AI SYSTEMS · SYSTEMS PAPER WITH MEASUREMENTS · RESEARCH PROTOTYPE (SIMULATION)

VDSG: A Commanded Admission-Control Runtime for Autonomous Agents

Rules that decide, evidence that learns, orders that only narrow, and an eye taught by the world. We state the runtime's contract and its order semantics as propositions, derive the estimators behind its perception and its sense of pace, size its recovery from being stuck out of what it sees and how fast it moves, and measure the whole thing at the wheel of id Software's DOOM (1993) and Wolfenstein 3D (1992) – with every failure kept in the record.

Perslis Research · ViZDoom 1.3 on DOOM1.WAD v1.8 · ECWolf on the WL6 data · 2026-09-26

Download PDF ↓ | The recorded run, replayed →

Systems paper · research prototype · nothing here is qualified for use

Admission before the planner | Orders that only narrow | An eye taught by the engine | Negatives published

ABSTRACT

An autonomous agent is usually trusted because it appears to perform well; nobody can say in advance what it will refuse to do, or why it did what it did. VDSG takes the opposite stance. It is a runtime that, at every decision, computes the set of goals an agent is *permitted* to pursue from explicit rules over facts, lets an operator narrow that set in ordinary language, learns from

evidence only inside it, and writes down the reason for every choice. We formalise the contract — facts, an admissible set, a fixed-priority rule policy, a bounded learner — and prove three small but load-bearing properties of orders: an order can only narrow the admissible set, the learner can never leave it, and an unsatisfiable order is refused with a reason rather than silently widened. We then add senses the runtime owns rather than assumes: an *eye* that labels and ranges the rendered frame from colour statistics it is taught by the engine while it plays (an image-space estimator whose one constant is learned from the projection law of a raycaster), an *odometer* that measures the agent's real pace and turn rate from what each command produced, and a *stuck rule* calibrated by that pace whose escape is sized from the deepest open space the eye can see. We measure all of it on the real 1993 DOOM shareware engine and on Wolfenstein 3D: the depth buffer's scale (7.64 map units per step for walls, 7.35 for sprites; 1,703 credible samples), the eye's agreement with its teacher (78–92% on DOOM, 79% on Wolfenstein) and its ranging error (±7% on Wolfenstein; ±25–65% on DOOM, and why), the measured pace (47 units per decision, calibrated within two seconds), a unit error the calibration exposed (150 stuck events per 1,014 decisions, then 9), and the level-clearing results (E1M1 cleared on the third difficulty, not on the fifth; E1M2 not cleared, with the cause of death measured: 144 of 177 points of damage from hitscanners beyond 15 m never in view). This is a research prototype in simulation. Its claim is not that it plays well; its claim is that every action it takes is explicable from the facts it was handed, that an operator's word binds it, and that its senses report their own error.

Intelligence proposes. The runtime decides what may be done, says why, and can be told no in plain words — and the no is a guarantee, not a preference.

FOR EVERYONE — WHAT WE ARE TRYING TO DO, IN PLAIN LANGUAGE

The problem. Machines that act on their own — a car that drives, a drone that flies, a program that plays a game — are mostly trusted because they seem to do well. When one does something wrong, nobody can point to the moment it decided and say what it was allowed to do, what it chose, and why. Trust built that way is a hope, not a check.

What VDSG is. A referee that sits underneath whatever is doing the acting. At every instant it looks at the plain facts of the situation — what is nearby, what is in view, how much health and ammunition there is, where the exit is — and writes down the short list of things the actor is *allowed* to do right now. The actor picks from that list and nothing else. A person can shorten the list by speaking: *don't fire*, *only use the shotgun*, *hold position*. The referee never invents anything from a command; it can only take options away. If a command cannot be obeyed — *only use the shotgun* when there is no shotgun — it says so, out loud, every moment, until it can.

Why a video game. DOOM and Wolfenstein 3D are cheap, fast, unforgiving worlds with doors, keys, enemies, ammunition and no mercy. We can run them thousands of times, break the referee on purpose, and measure everything. Nothing here is aimed at games for their own sake: the game is the wind tunnel.

What we want. (1) To show that an actor with *no neural network*, governed by a referee, can play a hard game and explain every action it took. (2) To let a person command it and be sure that a command is a guarantee. (3) To give it senses it learns from the world itself — an eye that learns what a door looks like because the world tells it where the doors are, and a sense of its own speed so it knows how far things are in seconds, not just in metres. (4) To make it recover from being stuck the way a person would: turn toward the open space you can see, walk for as long as that space is deep. (5) To measure honestly and publish the failures next to the successes.

Where this goes. The same referee, under a robot, a vehicle or a drone: the rules change, the contract does not. Our earlier work put this referee under a driving simulator; this paper is the version with senses of its own and a person at the microphone.

What it is not. A product, a certified system, or a claim that the actor plays well. It dies a lot. The paper says exactly where and why.

1 · Introduction

Two earlier reports from this group studied admission control beneath an autonomous controller: a stateless shield of composed invariants under benign, hostile and frontier-model drivers in a photoreal simulator [1], and the formal account of admissible motion it instantiates [2]. In both, the controller *proposes* and a deterministic runtime decides what may reach the actuator. The runtime was stateless, its invariants were fixed, its perception was the simulator's ground truth, and no person could speak to it while it ran. This paper removes those four restrictions one at a time, keeping the contract fixed, and measures what each removal costs. The result is VDSG: a runtime that (i) chooses among *goals* rather than clamping a scalar, from a fixed-priority rule policy over an admissible set; (ii) accepts standing orders from an operator in ordinary language and treats them as constraints on that set; (iii) learns from its own evidence, but only inside the set; (iv) owns two senses — an eye and an odometer — that it calibrates against the world rather than assumes; and (v) recovers from being stuck with an escape whose direction and length come from those senses.

We chose the DOOM and Wolfenstein 3D engines as the testbed deliberately. They are deterministic given a seed, run thousands of times faster than a vehicle simulator, expose ground truth (object lists, per-pixel labels, a depth buffer, sector geometry) that lets every learned estimate be checked against a teacher, and are unforgiving: a pilot that turns on the spot at a wall for 1,300 decisions dies, and the record says so. The engines are the real ones — id Software's 1993 shareware IWAD under ViZDoom [3], and the 1992 WL6 data under a tapped ECWolf [4] — not simplified reimplementations.

Contributions

1. **A commanded admission contract with proved order semantics** (Section 2): orders only narrow, the learner is bounded by the narrowed set, and unsatisfiable orders are refused with a reason. The proofs are short; the properties are the ones an operator needs to be true.

2. **An eye taught by the world** (Section 3): a per-column colour-statistics labeller and an image-space range estimator whose single constant follows from the raycaster projection law and is learned from the engine's own distances. We give the estimator, its calibration procedure, the measured scale of ViZDoom's depth buffer, and the estimator's error on both engines — including the case where it is poor and the reason.
3. **An odometer and a calibrated stuck rule with sized escapes** (Sections 4–5): pace and turn rate measured from what commands produced; a "not moving forward" test judged against the expected displacement; an escape whose pivot angle comes from the deepest visible open space and whose length comes from the measured pace. The calibration exposed a unit error that had been costing one false stuck every eight decisions.
4. **Navigation and acquisition on the engine's own geometry** (Sections 6–7): a grid from sector data with one-way ledges, doors, lifts and keyed doors; an engine-exact door protocol for Wolfenstein; and a "go and get a gun" behaviour with a give-up memory, with the cause of death on the second level measured rather than guessed.
5. **Measurements, negatives included** (Section 9): what cleared, what did not, and the four engine facts that had to be measured rather than assumed.

We are explicit about scope. This is a **research prototype in simulation**. Nothing in it is qualified for use, and the pilot's competence is not the claim.

2 · The contract, and what an order is

2.1 · Facts, goals, and the admissible set

Let s_t be the engine's state at decision t, and let $F(s_t)$ be the *situation report*: a structured, model-free extraction of facts — enemies with bearing, distance and whether they are in view; pickups; health, armour, ammunition and the weapon in hand; position and heading; whether the pilot was hit recently; whether there is ground behind it to fall

back over. F is the only domain-specific stage. Everything after it is the same across engines.

Let

$G = \{\mathrm{HEAL}, \mathrm{DODGE}, \mathrm{RETREAT}, \mathrm{ATTACK}, \mathrm{SEARCH}, \mathrm{RESUPPLY}, \mathrm{EXPLORE}\}$

be the goals. An *applicability* function $A : F(s) \mapsto 2^G$ returns the goals that can be executed in the current situation: a goal is admissible only when its object exists — ATTACK needs an enemy in view, HEAL a health pickup in view, RESUPPLY ammunition or a wanted weapon on the radar, RETREAT ground behind. Three goals — DODGE, SEARCH, EXPLORE — are always admissible. The *rule policy* π_R chooses one goal from $A(s)$ by a fixed priority (survival, then self-defence, then engagement, then progress); it holds no state and has no parameters that are fitted.

An *evidence memory* M may override π_R: it ranks goals by an estimate of return-to-go and failure probability learned from the pilot's own past, but only among the goals offered to it. Finally an *actuator* α turns the chosen goal into a button vector, and the engine advances k tics ($k = 4$ on DOOM at 35 tics/s, $k = 8$ on Wolfenstein at 70 tics/s; both give about nine decisions a second).

2.2 · Orders

A standing order O is a constraint an operator places on the pilot in words. We define it not as text but as its effect: a map from a situation to a subset of goals and a set of button-level restrictions,

$$O = (\,\Gamma_O,\ \beta_O,\ w_O\,), \qquad \Gamma_O : F(s) \mapsto 2^G, \quad \beta_O \subseteq \mathcal{B}, \quad w_O \in \{\bot\} \cup \mathcal{W},$$

where $\Gamma_O(s)$ is the set of goals the order still permits, β_O is a set of buttons the order removes (the trigger, every movement button), and w_O is a weapon the order names, or none. The parser that produces O from text is a fixed phrase vocabulary with explicit negation. It is not a model: text it does not recognise is refused, never guessed. The *admissible set under an order* is

$$A_O(s) \ = \ A(s) \cap \Gamma_O(s),$$

and when this intersection is empty the runtime falls back to $A(s) \setminus \{\mathrm{ATTACK}\}$: an order that cannot be met never puts the pilot into engagement it was not already going to enter. The order is *in force* when its precondition holds in s (the named weapon is carried, for instance) and *refused* otherwise; in both cases a receipt naming the reason is emitted every decision.

> **Proposition 1 (orders only narrow).** For every situation s and every order O, $A_O(s) \subseteq A(s)$. No goal outside $A(s)$ is ever executed under any order.

Proof. $A_O(s) = A(s) \cap \Gamma_O(s) \subseteq A(s)$ by definition of intersection, and the fallback set $A(s) \setminus \{\mathrm{ATTACK}\}$ is likewise a subset of $A(s)$. The button-level restriction β_O can only clear buttons; the single button an order may set — the weapon w_O — is set only when w_O is carried, which is a precondition of the goal set $A(s)$ already permitting a weapon change. □

> **Proposition 2 (the learner is bounded).** Let g_t be the goal executed at decision t with the memory M active. Then $g_t \in A_O(s_t)$ (or in the fallback set when $A_O(s_t) = \varnothing$) for all t, whatever M has learned.

Proof. M is given the narrowed set as its domain: $g_t = \arg\max_{g \in A_O(s_t)} U_M(s_t, g)$ when it overrides, and $g_t = \pi_R(s_t)$ restricted to $A_O(s_t)$ otherwise. In neither branch does a goal outside the domain enter the maximisation. The actuator receives g_t and applies β_O again before the engine, so a branch that forgot the order still cannot leak a removed button. □

> **Proposition 3 (refusal is sound and complete).** An order is reported in force at decision t if and only if its precondition holds in $F(s_t)$; a refused order changes no button; and an order becomes in force at the first decision at which its precondition holds, with no action by the operator.

Proof. The receipt is computed from $F(s_t)$ alone, every decision, by the same predicate that gates the order's effect on the vector, so the two cannot disagree; a refused order contributes neither to Γ_O nor to β_O nor sets w_O. Because the receipt is

recomputed every decision from the current facts, satisfaction is detected at the first decision it holds. □

These are not deep results. They are the properties that make an order worth giving. We state them because the failure modes they exclude were all observed during development before the runtime enforced them: a negated weapon obeyed backwards, comma-separated clauses merged into one, and an unsatisfiable goal order silently widened to ATTACK — all caught by an adversarial review and all now excluded by construction and pinned by per-site regression tests.

One consequence deserves stating. Because orders bind the memory (Proposition 2), *learning cannot undo an order*: a pilot that has learned that ATTACK pays cannot be talked into it by its own evidence while an operator holds the trigger. The evidence memory ranks inside what is left.

3 · The eye: perception taught by the world

The runtime's first witness to the world is the engine's own state — the object list, the label buffer that names every rendered sprite, the depth buffer, the sector geometry. A second witness is useful for two reasons: it can be checked against the first, so the runtime can report how well it sees; and it is what remains when the first witness is absent, as it is on the Wolfenstein engine, which exposes no depth buffer, and as it would be on a physical platform.

3.1 · Working image and colour

The rendered frame is shrunk by nearest sampling to $W \times H = 64 \times 40$ and each pixel is quantised to a colour bin $q(r, g, b) \in \{0, \dots, B-1\}$ with $B = 512$: three levels-of-eight per channel, with the level thresholds placed non-uniformly toward the dark end of the range. The choice was forced by measurement: an earlier two-bits-per-channel quantiser ($B = 64$) put roughly 80% of the pixels of a lit DOOM corridor into a single bin, and the eye's agreement with its teacher fell to 28% as more classes began to vote.

3.2 · Class models and the column label

For each class $c \in C = \{\text{ceiling}, \text{floor}, \text{wall}, \text{door}, \text{enemy}, \text{pickup}\}$ the eye keeps a histogram H_c of colour bins with total count N_c, and a Laplace-smoothed likelihood

$$P(b \mid c) \;=\; \frac{H_c(b) + \alpha}{N_c + B\alpha}, \qquad \alpha = 1.$$

A column x of the working image is labelled by the thing class that best explains the rows it occupies,

$$\ell(x) \;=\; \arg \max_{c \in \{\text{wall},\text{door},\text{enemy},\text{pickup}\}} \; \sum_{j \in R(x)} \log P\big(q_{j,x} \mid c\big),$$

where $R(x)$ is the row extent of the thing in that column (Section 3.4), and a class may vote only once $N_c \geq N_{\min}$. Confidence is a function of the log-likelihood margin Δ between the best and second class, $1 - e^{-\Delta/\kappa}$. Adjacent columns with the same label form a *card*: kind, dominant colour name, bearing $\theta = (\bar{x}/W - \tfrac{1}{2})\,\Phi$ for a horizontal field of view $\Phi = 90^\circ$, confidence, and — once the eye can range — a distance. Histograms are halved once they grow past a cap, so a new level's lighting is learned rather than argued with.

3.3 · The teacher

The eye is *taught*, not trained. While the pilot plays, the engine supplies a class and a distance for every column: on DOOM, the label buffer names the sprite under a column (enemy or pickup by type), the navigator's grid says whether a door lies along the column's ray before the depth says the ray stopped, and the depth buffer gives the distance; on Wolfenstein, a ray per column over the tile map gives the first wall or shut door and its distance, and the radar's sprites overwrite the columns they cover when nearer. Each look adds the column's pixels to the taught class's histogram; ceiling and floor are learned from the rows above and below the thing. Agreement is scored on every look as the fraction of labelled columns whose label matches the teacher.

Three teacher errors had to be found and removed, each by measurement: the status bar at the bottom of the DOOM frame (rows 404–479 at 640×480, depth 0 and label 0) had been taught as *floor*; the pilot's own weapon, drawn bottom-centre and carrying the

player's own label, had been taught as floor; and close walls reaching the top and bottom edge rows had been taught as ceiling and floor. Each contaminated a model that the range estimator (below) depends on.

3.4 · Ranging from apparent height

A raycaster draws a vertical surface of world height H_w at perpendicular distance $d_\perp$ with an on-screen height h that follows the pinhole projection law,

$$h \;=\; \frac{f\,H_w}{d_\perp} \quad\Longrightarrow\quad h\,d_\perp \;=\; fH_w \;\equiv\; k,$$

so the product of apparent height and distance is a constant k per class of surface, and $\hat{d}_\perp = \hat{k}/h$. The eye measures h as the run of rows around the horizon that look more like the thing than like the ceiling above or the floor below, with a two-row hysteresis so a dark texture band is not mistaken for the ceiling. It learns $\hat{k}$ as the median of the samples $h_i \cdot d_{\perp,i}$ the teacher provides, per class, after a minimum number of samples; where the teacher can measure the true extent itself (DOOM's depth buffer holds one depth per wall column, so the wall is exactly the run of rows with that depth) the sample uses the true height and $\hat{k}$ becomes exact for that class. The euclidean distance along a column's ray is $\hat{d} = \hat{d}_\perp / \cos\theta$. The reported error is the median relative error of the eye's *own* estimate against the teacher's distance, $\varepsilon = \mathrm{median}\;|\hat{d}_\perp - d_\perp|/d_\perp$, and it is shown on the console beside the agreement.
On Wolfenstein every wall is one tile (64 units) tall; with a 320×200 render shrunk to 40 rows the law predicts $k = 32$ rows·tiles. The eye learned $\hat{k} = 33$.

3.5 · The depth buffer's scale, measured

ViZDoom's depth buffer is an 8-bit image whose relation to distance is not documented for the purpose of measurement. We calibrated it against the map's own geometry: for a column, the nearest line of *any* kind along the column's ray had to be a one-sided wall — so that nothing two-sided (a step, a window, a raised floor) could lie between — and the depth at the horizon row was regressed on the perpendicular distance to that wall. Over 1,336 such columns on E1M1 at 640×480,

$$\text{depth} \;=\; 0.1309\, d_{\perp} \;-\; 1.04 \quad (\text{walls; median residual } 0.9 \text{ steps}),$$

$$\text{depth} \;=\; 0.1360\, d_{\perp} \;-\; 0.82 \quad (\text{sprites; } 367 \text{ samples; median residual } 0.2),$$

that is, 7.64 map units per step for walls and 7.35 for sprites, saturating at 255 (about 61 m). Consistent with the renderer's source, in which wall depth is derived from the column's inverse scale and sprite depth from the sprite's, with different constants. The linear law in the *perpendicular* distance — not the euclidean — is what a column-scaled renderer produces, and it is what the eye's projection law needs.

3.6 · Two witnesses

Where the engine supplies a depth, the pilot's cards carry the engine's distance as the first witness and the eye's own estimate beside it ("eye says 3.8 m"); where it does not, the eye's estimate is what the pilot has. The range profile that sizes an escape (Section 5) follows the same rule. The eye therefore never overrides the map: it is a second witness whose disagreement is a number the operator can read.

3.7 · Results

Quantity	DOOM E1M1	Wolfenstein MAP01	Note
Agreement with the teacher	78–92%	79%	120 s runs; four classes voting
Range constant $\hat{k}$ vs theory	4,166–7,753 (varies)	33 vs 32	DOOM sector heights vary; Wolfenstein walls are one height
Own ranging error ε	±25–65%	±7%	DOOM: floor and wall share colours at range in the dark palette
Cost of a look	18 ms median	15 ms median	every second decision

Two negative findings are structural, not tuning. First, on DOOM the wall height H_w varies by sector (72-, 128-, 256-unit rooms), so k is not one constant and the median is the

typical wall; an estimator built on the projection law cannot do better without knowing which sector a column belongs to — which the map does, which is why the map is the first witness. Second, in DOOM's dark palette the floor and the walls become the same colour bins at range, and a colour-only edge scan cannot find the base of a distant wall. The eye's number says so; we did not tune it away.

4 · The odometer: a measured pace

A distance is useful to a pilot only with a pace. Rather than assume the engine's movement constants, the runtime measures them from what each command produced. Let $u_t \in [-1, 1]$ be the forward command sent at decision t, σ_t the strafe command, and $\Delta x_t = \|x_{t+1} - x_t\|$ the displacement it produced. A *clean* sample is one with $|u_t| \geq \frac{1}{2}$ and $\sigma_t = 0$. The pace is the 80th percentile of the normalised clean samples over a recent window,

$$\hat{p} \;=\; Q_{0.8}\left\{ \frac{\Delta x_t}{|u_t|} \;:\; t \text{ clean} \right\},$$

a percentile rather than a mean because a sample taken while bumping or sliding along a wall is near zero and must not lower the estimate of what the pilot does when it is clear. The turn gain is measured the same way from heading changes under turn commands. Then the expected displacement of a command is $E_t = \hat{p}\,|u_t|$, and the time to any distance d is $\mathrm{ETA}(d) = (d/\hat{p})\,\Delta t$ with Δt the decision period.
Measured: DOOM 47 map units per decision at full forward — 12.8 m/s in the engine's 32-units-per-metre convention — with the estimate stable after eight clean samples, about two seconds; Wolfenstein 0.7 tiles per decision. Every card the eye produces then carries a time: *door · grey · 12° right · 4.2 m · 0.4 s away*.

5 · The stuck rule, calibrated and sized

A rule that binds every goal that moves: if the pilot is not moving forward, it pivots; if the same movement keeps recurring, that too is stuck, and it breaks out rather than

replanning into the loop. It was set after being measured absent: 1,300 consecutive decisions turning at one wall on E1M1; 686-second timeouts circling 7% of E1M2.

5.1 · Two rules

Rule 1 (not moving forward). Let $m_t = \|x_t - x_{t-1}\|$. The pilot is still at t when $m_t < \tau_t$, and stuck when it has been still for n_1 consecutive decisions while asking to move. Uncalibrated, $\tau_t = \tau_0$ is a fixed line. Calibrated by the odometer,

$$\tau_t \;=\; \max\!\big(\tau_0,\; a\,E_{t-1}\big), \qquad E_{t-1} = \hat{p}\,|u_{t-1}|,\; a = 0.3,$$

so a command that should have moved 47 units and moved 8 is judged still, whatever a fixed 5-unit line says. Decisions in which the pilot chose not to move (holding position to shoot) are excluded from the count: they are not being stuck.

Rule 2 (the same movement recurring). Over a window of n_2 decisions the net displacement $\|x_t - x_{t-n_2}\| < \delta$. This is deliberately net displacement and not "the same action repeated": walking straight down a corridor repeats one action every decision and is the opposite of stuck. A loop is movement that goes nowhere.

The thresholds are in the engine's own units. That sentence was earned: the Wolfenstein lane ran for a day with the DOOM constants — five *tiles* a decision as the line for "moving" — and recorded 150 stuck events in 1,014 decisions, one every eight, because no Wolfenstein pilot moves five tiles a decision. In tiles: 9. The calibration work found it, which is what calibration is for.

5.2 · The escape

When either rule fires, the navigator learns the cells it was trying to enter as blocked, forgets its route, and commits to an escape: a pivot, then a walk clear, then replanning. The escape used to be fixed — about 96° and seven decisions, alternating sides. It is now sized from what the eye sees and how fast the pilot moves. Let β^* be the bearing of the deepest open space in the eye's range profile (the per-column euclidean depth, smoothed over five columns) and d^* that depth. If $|\beta^*| \geq \beta_{\text{ahead}}$, the pivot angle and its duration are

$$\theta = \operatorname{clip}\big(|\beta^*|,\, \theta_{\min},\, \theta_{\max}\big), \qquad n_{\text{pivot}} = \left\lceil \frac{\theta}{\omega\, g} \right\rceil,$$

with ω the engine's full turn rate per decision and g the measured turn gain, and the walk clear is

$$n_{\text{walk}} = \operatorname{clip}\left(\left\lceil \frac{\operatorname{clip}(\tfrac{1}{2} d^*,\, d_{\min},\, d_{\max})}{\hat{p}} \right\rceil,\; n_{\min},\; n_{\max} \right),$$

half the seen depth at the measured pace. If instead the deepest space is straight ahead ($|\beta^*| < \beta_{\text{ahead}}$) yet the pilot is not moving, whatever blocks it is something the eye cannot see — a ledge, a body — and the sides alternate as before. Every escape states its case on the console: *"deepest open space 42° right (14.7 m) — pivoting right 45° (1 decision), walking clear 5 decisions"*. The constants ($\theta_{\min} = 45^\circ$, $\theta_{\max} = 150^\circ$, $n_{\min} = 3$, $n_{\max} = 20$) bound the escape; the direction and length inside those bounds are the world's.

6 · Navigation on the engine's own geometry

A training arena needs no navigation; a level does. The runtime rasterises the engine's sector geometry into an occupancy grid (16-unit cells on DOOM) and plans on it with breadth-first search. What the grid knows, and where each fact comes from:

- **Walls**: one-sided and flagged-blocking lines. A cell is blocked when its centre lies within the player's radius of the wall — an exact distance, not a cell-and-neighbours inflation, which had closed every 48-unit passage on E1M2.
- **Steps**: a two-sided line whose floors differ by more than the engine's climb limit (24 units) is crossable *downhill only*. Blocking steps both ways fenced the exit of E1M2 off from the start.
- **Doors**: a two-sided line into a sector with less headroom than a player. Passable for planning; USEd on arrival; never learned as a wall. **Lifts** from the map's own line specials, treated as doors. **Keyed doors** blocked until the key is held — and

"held" is inferred from a key card that was on the radar and vanished while the pilot stood on it.

- **The exit**: from the map data's own line specials, verified against the engine's spawn position.
- **Learned blocks**: anything else it cannot make progress into, evidence rather than guess. **Live geometry**: a signature of every floor and ceiling height, rechecked every few decisions, so a lift that has come down is walkable.

Planning priority is: a wanted pickup, the exit, a key the exit needs, the nearest unvisited cell, then coverage laps. Every decision on the console is a sentence about this: *to the exit switch*, *at a yellow door without the yellow key*, *picked up the red key*.
On Wolfenstein the map is a 64×64 tile grid and the door protocol had to be made engine-exact. The engine's USE acts on exactly one tile — the one adjacent to the player's tile in the facing cardinal direction — and toggles a door: pressing USE on a door that is already opening reverses it. A pilot pushing at a half-open door and pressing USE every 0.7 s was therefore closing the doors it was trying to open. Measured before: 583 of 1,275 decisions spent at doors, with a nine-tile oscillation at one of them. Now USE is pressed once, when the adjacent facing tile is a shut door, and not while it is opening: 371 of 1,184 decisions at doors in the next run.

7 · Wanting a gun, and how the pilot dies

The second DOOM level killed the pilot in every attempt, and the record showed why: it passed within 8.8 m of a chaingun and 16.7 m of a shotgun, neither ever *in view* — behind a partition, on a podium — and died with the pistol after sixteen kills. A gun the radar knows of is now *wanted* when it shoots (weapon slot three and up), is not carried, and is within reach: the wanted set $\mathcal{W}(s)$ is nearest-first, and RESUPPLY becomes admissible with it and routes to it on the grid. A pickup the map cannot route to, or that is not reached within a patience of 150 decisions, or beside which the pilot stands for 20 decisions without getting it, is remembered as *unreachable* and never wanted again that episode. The patience is counted per pickup and survives interruptions — a fight in the middle does not reset it — because the first version, which reset on every ATTACK, spent 281–514 decisions a try wanting a chaingun it could not have.

After: the shotgun is taken at 10–20 s in every attempt, the red key the exit needs is held in every attempt, and the pilot still dies at 112–207 s with 16–18 kills. We measured the death rather than guessed at it. Of 177 points of damage in one full attempt, all came from Zombiemen; 144 of them from beyond 15 m; in every hit the shooter was *not in view* and the goal was SEARCH — turning toward the *nearest* radar enemy, which stood behind a wall while a Zombieman 30 m the other way did the shooting. Two rules followed.
SEARCH now turns toward the enemy most likely responsible (facing the pilot, in range) rather than the nearest; that is a correction of an error and stays. The second — a pilot hit by something unseen while hurt, with ground behind, RETREATs to break the line of sight before it searches — was a hypothesis, and it measured worse: a mean of 123 s alive over three attempts (112, 125, 133) against 187 s over the four before it (167–207). It was reverted, and the record keeps the number. At these sample sizes the variance is large; the honest statement is that breaking the line of sight is not, by itself, a survival rule on this level, and that survival there remains unsolved.

8 · Learning without a model

Two learners sit inside the contract, both bounded by Proposition 2. The first is the evidence memory from the arcade work [5]: per situation signature and goal, a return-to-go estimate and a failure rate, with the rule policy as a prior it may override with a stated cause. On the training arena the result is parity, not a win — random 3.2, rules 17.9, rules with memory 20.3 mean return on 32 seeds, paired $t = 0.78$ — and we keep the number that way. The second is a trace of typed evidence tiles, one per *encounter* (a contiguous run in one situation with one goal, not one per frame, which had produced 79,000 cards whose rates were all near zero), from which rules of the form "this goal is harmful whenever…" are widened by abstraction over the situation and admitted only when their harm rate clears the base rate by a margin. Of six such rules, four reproduced on held-out episodes. The mechanism of the trace and its admission gate is described in the Peel report [6]; this paper only uses it.
One credit-assignment defect is worth the record because it inverts a headline. The engine's kill counter is the *map's*: it counts monsters that kill each other. A pilot ordered never to fire — verified, zero trigger presses over eight episodes — was still credited 55 kills. Decomposed on the same seeds, moving without firing was worth most of the return

and shooting a little more, inside the noise. Rules "five times random" is mostly locomotion. The console's headline counters are damage dealt and shots on target, which are exactly zero when the pilot does not fire.

9 · Measured

What	Measured	Where
Random / rules / rules + memory	3.2 / 17.9 / 20.3 return on 32 seeds — parity (paired $t = 0.78$)	Freedoom arena
Kill credit	a never-fire pilot credited 55 kills; damage dealt is the honest number	defect, published
E1M1, third difficulty	cleared: exit switch at 52 s, 4 kills	DOOM1.WAD
E1M1, fifth difficulty (Nightmare)	not cleared in 8 attempts; dies at 23–37 s	DOOM1.WAD
E1M2, third difficulty	shotgun at 10–20 s, red key held; dies at 112–207 s, 16–18 kills; 144/177 damage from unseen hitscanners beyond 15 m	DOOM1.WAD
Depth buffer scale	7.64 units/step walls, 7.35 sprites; 1,336 + 367 samples; residuals 0.9 / 0.2 steps	ViZDoom 1.3, 640×480
Eye agreement	78–92% (DOOM), 79% (Wolfenstein)	120 s runs
Eye ranging	±7% (Wolfenstein, $\hat{k} = 33$ vs 32); ±25–65% (DOOM, own estimate)	120 s runs
Pace	47 units/decision = 12.8 m/s (DOOM); 0.7 tiles/decision (Wolfenstein); stable after 8 samples	odometer
Stuck rule in the wrong units	150 → 9 events per ~1,000 decisions	Wolfenstein MAP01
Wolfenstein doors	583/1,275 → 371/1,184 decisions at doors; USE once, when shut, on the engine's tile	MAP01

What	Measured	Where
Wolfenstein progress	route to the elevator shortens from 126 to 25 tiles in 150 s; elevator not yet reached; 3 deaths (respawn)	MAP01

Every figure measured in-house on the machine that recorded the public replay; negatives included. Attempt counts are small (four to eight) and are reported as such; nothing here is a statistically powered comparison except the 32-seed arena result, which is a null.

10 · What we learned

1. **Measure the engine; do not assume it.** The depth buffer's scale, the tile USE acts on, the door toggle, the map's kill counter, the units of a threshold — each was assumed once and measured later, and each assumption had a cost in the record. The methodology that worked was always the same: find a case where the engine's answer is beyond doubt (a one-sided wall with nothing in between), collect many, regress, and keep the residuals.
2. **The first witness is exact; the second learns.** Where ground truth exists, the runtime uses it and shows the learned estimate beside it with its error. Where it does not, the learned estimate is what there is, and its error was measured where it could be. This is the honest way to introduce a learned sense into a runtime that must explain itself.
3. **An order that can only narrow is the safe interface to a learner.** Proposition 2 is one line, and it is the line that lets an operator trust a system that learns.
4. **Calibration finds bugs.** The unit error was invisible while the threshold was a constant and obvious the moment the threshold became a fraction of a measured pace.
5. **Patience must survive interruptions.** Any give-up counter that resets on a context switch is not a give-up counter.
6. **Rules are mostly locomotion.** Most of the return of a rule-based pilot over random is moving at all; the memory's contribution is inside the noise at this scale. Say so.

7. **Failure avoidance alone does not learn a coupled objective.** The arcade finding [5] — the same mechanism gains on one game and loses on another where risk and objective are coupled — is the open problem here as well. The second DOOM level, where the pilot must take fights to progress, is where it is being measured.
8. **Publish the deaths.** A record with the failures in it is the only kind an operator can use.

11 · Limitations and what is next

The pilot does not clear the second DOOM level and does not clear the first on the fifth difficulty. Its own ranging on DOOM is poor for the two structural reasons given. It has no vertical field of view and no notion of cover beyond the trail it walked in on. The Wolfenstein pilot does not yet reach the elevator, and spends a third of its decisions at doors after the protocol fix, which means the remaining cost is elsewhere and unmeasured. The orders vocabulary is fixed and small by design; that is the point, and also a limit. None of this is qualified for use.

Next: survival on the second level, measured the way the death was measured; the elevator; the same runtime under a physical platform, where the eye is the only witness and the odometer is what the wheels report; and the coupled-objective problem, which no amount of engineering here has moved.

12 · Related work

Runtime assurance. The Simplex architecture [7] and shielding for learned controllers [8] place a verified component beside or beneath a high-performance one. VDSG differs in sitting *before* the planner (the planner never sees the full action space), in choosing among goals rather than clamping a control, and in accepting an operator's constraint as a first-class input with proved semantics. *Deterministic game agents.* Goal-oriented action planning [9] and behaviour trees give game agents explainable structure; VDSG's rule policy is closer to a fixed-priority selector, and its novelty is not the selector but what is around it — the admissible set, the orders, the bounded learner and the taught senses.

Learned Doom agents. ViZDoom [3] has hosted a decade of reinforcement-learning agents and, more recently, language-model pilots handed a structured state; VDSG uses the same state contract without a model and with a runtime that can explain and be commanded. *Perception by colour statistics.* Per-column classification from colour histograms is deliberately the smallest thing that works; its role here is as a checkable second witness, not as a vision system of record. *Engines.* The DOOM and Wolfenstein 3D renderers are described exhaustively in [10, 11]; their projection laws are what Section 3.4 uses.

13 · Conclusion

VDSG is a runtime, not a player. It decides what an agent may do, from rules over facts; it can be narrowed in words, and the narrowing is a guarantee; it learns only inside what it is allowed; and it perceives through senses it calibrates against the world and whose error it reports. Measured at the wheel of two unforgiving engines it clears one level and dies on the next, and the record says exactly where, why, and what was assumed that should have been measured. That record — not the score — is the product.

References

1. Perslis Research. *Runtime Admission Control on a Photoreal Driving Simulator: Hostile and Frontier-Model Controllers Under One Verification Floor.* 2026. research.perslis.com/carla-admission.html
2. Perslis Research. *Admissible Motion.* 2026. research.perslis.com/motion.html
3. M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jaśkowski. *ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning.* IEEE Conference on Computational Intelligence and Games, 2016.
4. ECWolf, a source port of Wolfenstein 3D (GPL). maniacsvault.net/ecwolf
5. Perslis Research. *The Arcade Floor: symbolic failure memory on real Atari ROMs* (Space Invaders +32%, Freeway −12% on identical code). 2026. perslis.com/defense/evidence

6. Perslis Research. *Peel: a typed evidence floor with a functional hard veto.* 2026. research.perslis.com/peel.html
7. L. Sha. *Using Simplicity to Control Complexity.* IEEE Software 18(4), 2001.
8. M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, U. Topcu. *Safe Reinforcement Learning via Shielding.* AAAI, 2018.
9. J. Orkin. *Three States and a Plan: The A.I. of F.E.A.R.* Game Developers Conference, 2006.
10. F. Sanglard. *Game Engine Black Book: Wolfenstein 3D.* 2017.
11. F. Sanglard. *Game Engine Black Book: DOOM.* 2018.

How to cite

```
@techreport{perslis2026vdsg,
  title       = {VDSG: A Commanded Admission-Control Runtime for Autonomous Agents},
  author      = {{Perslis Research}},
  institution = {Perslis Research},
  year        = {2026},
  month       = {9},
  note        = {Systems paper with measurements; research prototype (simulation).},
  url         = {https://research.perslis.com/vdsg.html}
}
```