The Dream That Won't Compound

https://gerolds.github.io/textbook/textbook/posts/the-dream-that-wont-compound/

Contents

The Dream That Won't Compound

There is a particular kind of studio that is easier to recognize from the outside than to describe honestly from within.

It has committed people, shipped titles, some funding history, and enough public legitimacy to avoid looking like a failure. It may win small awards, attend conferences, and present itself as a serious independent studio with long-term ambitions.

But internally, the core aspiration and the operating reality have drifted apart. The stated goal is usually something like this: make premium games, earn a living from players, and build a studio that can sustain itself on the strength of its work. The problem is that the games do not earn enough to support that goal. A €500K production that returns less than half of that in its first year is not a temporary shortfall if the same pattern keeps repeating.

What makes this hard to diagnose is that the games are often competent in a narrow sense. They ship. They function. They may even be well intentioned. But they do not create enough demand to spread through recommendation, community, or word of mouth. The issue is less technical execution than the absence of a process that reliably produces work people strongly want.

The studio survives anyway through grants, subsidies, publisher advances, and periodic acts of internal sacrifice. That survival matters, but it also obscures the central question: *is the studio building a system that can ever produce the outcome it says it wants?*

The dream stays alive not because the evidence supports it, but because the life-support machinery works well enough to stop the obvious question from being asked: *can this team actually do the thing it says it wants to do?* This essay is about that pattern.

---

1. The structure that eats the dream

The studio’s aspiration is clean: make premium games, live on revenue, answer to an audience instead of a funding body. It’s

worth having. It would clarify everything if the studio actually lived under it.

Instead, projects are funded through non-repayable grants, government subsidies, and low-tier publisher deals with full first-euro recoupment and milestone tranches. The money arrives in installments tied to deliverables written in a pitch document months before anyone understood the game. The milestones cannot be renegotiated without jeopardizing the agreement.

Every funding structure teaches a studio what to care about. This one teaches a single lesson: *complete the checklist*.

The studio's real risk, day to day, isn't "will players love this?" It's "will we finish the agreed deliverables before the money runs out?" That risk shapes everything:

- **Scope locks to the pitch**, not to the game. The pitch was written to win funding, and the milestones are locked to it. Every creative discovery that diverges from the original document becomes a liability instead of an opportunity.
- **Cuts and pivots become threats.** A pivot means renegotiating with a funder who doesn't understand why the game changed. It's easier to build what was promised, even when the team already knows it isn't working.
- **The budget is the ceiling.** Revenue, if it comes, arrives long after the work is done. Nobody wakes up thinking about the customer, because the customer is not who pays the bills this quarter.
- **Nothing compounds.** Each milestone is treated as a standalone task. Nothing snaps to a coherent whole because the whole was defined in a pitch no one fully believes anymore.

The deepest consequence is simple: **the team never feels bottom-line risk.** The grant absorbs mistakes. The subsidy absorbs drift. The publisher absorbs mediocrity. The team calibrates its intensity to match the actual stakes.

This isn't laziness. It's adaptation. People read their real incentive landscape with perfect accuracy, no matter what leadership says out loud.

Over time, the team gets better at winning grants and worse at making games anyone would buy. The muscles atrophy invisibly. People can still build features, write code, make art. What they can’t do is prove the game works while there’s still time to change it. The funding structure never required that proof, so the proof never came.

---

2. **How the projects actually go**

Everyone on the team has read the articles, watched the talks, nodded along to “fail fast, iterate, find the fun.” The actual daily process looks nothing like that.

Projects start too big. Grant applications reward ambition, not realism. A modest pitch for a tight game the team can actually deliver loses to the studio down the street promising a genre-defining experience.

So the project begins over-scoped, under-staffed, on a timeline just short enough that there’s never slack. There’s never room for the only thing that matters: **building the core loop first and proving it works before building anything else.**

Instead, the team builds systems. Inventory. Dialogue. Camera. Progression. These are building blocks that *could* be used to make a game, but they are not the game. They are infrastructure for a game that hasn’t been found yet.

Months pass. The systems exist. The game does not. Milestones are due. The budget is thinning. Nobody has a playable ten-minute slice that makes a stranger say “Oh, I get it” without a founder hovering nearby to explain what is supposed to be working. The team enters a quiet crunch where everyone is busy and little improves.

The game ships. It is not good enough by any bar a paying audience would accept. Players don’t hate it. They don’t finish it. They don’t talk about it. It sits on a store page with a handful of mixed reviews, generating revenue that looks like a rounding error next to the budget that produced it.

And the market the game shipped into is no longer the one the studio imagines. Games do not exist as isolated artifacts anymore. A game is also its story: its gestation, what it means to the people who play it, the identity it offers, the fifteen-second reel that makes a stranger stop scrolling.

The community, the clips, the developer's visible commitment: these are not marketing bolted onto the product. They are part of the product. A game that cannot survive compression into a short-form clip has very little chance of breaking out, because the clip *is* the discovery layer now. This does not mean every successful game is designed for TikTok. It means every successful game has something in it—a moment, a feeling, a visual, a verb—that travels without explanation. The studio in our pattern does not think about this. It builds the game, ships the game, and then wonders why nobody came.

Some games do break out despite ignoring all of this. But those cases usually rest on a specific lever—an IP, a community, a moment in culture—that cannot be replicated or planned for. Treating the exception as the model is one way studios avoid dealing with the selection pressures that govern discovery. Even innovation itself, miscalibrated, can work against you. A game so novel that it resists compression, that can't be explained in a sentence or shown in a clip, faces an uphill battle not because it's bad but because the funnel doesn't know what to do with it (https://gerolds.github.io/posts/when-the-funnel-eats-the-work/). The studio must design for legibility *and* depth. Treating them as opposites is a luxury the market no longer affords.

The studio tells itself this was a learning experience. Then the next grant cycle begins and the same pattern repeats, because the pattern isn't a mistake. It is the operating system.

---

3. **The organization that can't hold itself accountable**

The studio has no structured performance reviews. No structured work-item reviews. No instrumentation for observing whether the team is learning or just staying busy.

Essential roles (art direction, game direction, design direction, QA, marketing) are nominally assigned but not genuinely authorized. Someone has the title but not the mandate, the time, or the air cover to make the hard calls the role demands.

Values and expectations live in “common sense” and “personal responsibility.” In practice, nobody can be held accountable because nobody agreed on what accountability looks like. Rewards and consequences are illegible. People don’t know what earns recognition, what earns correction, or why. So they default to the safest behavior: pleasant meetings, smoothed-over disagreements, and nothing changing.

The culture is encouragement-only. It feels kind. It is kind, in the narrow sense of rarely making anyone uncomfortable. But when the only feedback is degrees of warmth, the team cannot distinguish between good work and merely adequate work. Intrinsically motivated people push themselves anyway. Everyone else calibrates to the ambient standard, which drifts downward so slowly that nobody notices until the game ships and the audience notices for them. Across the team, lessons are learned, shared, sometimes documented, and then ignored. The same issues resurface project after project, rediscovered by a different person at the same cost.

There is no trusted process. No method the team believes in enough to follow when it gets uncomfortable. So everyone improvises, and improvisation without a feedback loop is just drift.

---

4. **The seniority mirage**

Small studios cannot afford the seniority they need, so they relabel what they have. A mid-level developer becomes a “senior.” A junior artist becomes the “art lead.” Someone who shipped one game as a contributor is now directing.

The titles are aspirational, not descriptive. The consequences are not. Demands rise to match the title. Capacity stays at the actual level. The gap creates bottlenecks around the few people who genuinely have the experience—usually the founders—who become the single point of resolution for every design question,

every technical dispute, every creative disagreement. Decisions queue. The team drifts. Vision diffuses. Communication overhead eats the schedule.

The fix is not hiring more seniors. The studio can’t afford them. The fix is **scoping to actual capability**: building the game this team, at its real skill level, can ship well. Not the game the title-inflated version of the team might theoretically pull off. That team doesn’t exist. This one does.

---

5. **The stories the studio tells itself**

Studios in this pattern almost never describe their situation publicly. That silence is itself a signal. The indie postmortem genre is dominated by two shapes: the success story (“how we made it against the odds”) and the clean failure (“we ran out of money and here’s what we learned”).

The messy middle almost never gets written up: years of shipping games that technically exist but don’t sell, sustained by grants and political relationships, never quite failing and never quite working. There is no dramatic arc. There is no lesson that flatters the teller. There is just a slow, comfortable erosion that is hard to narrate without admitting that the dream was never backed by a mechanism.

When studios *do* speak honestly, the pattern is remarkably consistent. Introversion Software’s Tom Francis wrote openly about the economics of small indie games: even modest success on paper often means years of work for below-minimum-wage returns. Jake Birkett of Grey Alien Games published data showing that the median indie game earns almost nothing, and that studios routinely misread their own position on the distribution. The Vlambeer founders talked publicly about how the market shifted beneath them faster than their model could adapt. Kitfox Games’ Tanya X. Short has written about the brutal math of trying to sustain a small team when each game is essentially a fresh bet with no compound advantage from the last one.

But these are the articulate exceptions. For every studio that shares the math, dozens quietly close or quietly persist in the

pattern described here, telling themselves stories that prevent the real examination from happening.

The stories sound like this:

5.1. **“We spent €500K and earned €200K in year one. That proves we can make games.”**

It proves the studio can lose €300K on a single project. A 60% loss. In any other industry this triggers a fundamental reexamination. In indie games it is reframed as a promising start—partly because the comparison set is so grim that *any* revenue feels like validation. But the math does not care about the comparison set. The question is not “did we do better than the median?” The question is “can this model sustain a company?” The answer, at these numbers, is no. Run this experiment three more times and the studio is dead.

The deeper problem is that this story prevents learning. If €200K on €500K is “proof we can make games,” then nothing needs to change. The team that just demonstrated it cannot make a game people want in sufficient numbers has declared the demonstration a success. The next project inherits the same process, the same assumptions, and the same outcome.

5.2. **“We need PC and consoles to break even.”**

The cost of a cross-platform release—QA, certification, porting, platform-specific bugs, split marketing—is systematically underpriced. The team treats it as a line item. It is a multiplier on every other cost. The received wisdom that you need multiple platforms is inherited from studios with larger teams, established pipelines, and existing audiences. For a team of ten shipping its third game to a nonexistent fanbase, a single-platform release that is genuinely good will outperform a multi-platform release that is merely present. Platform breadth without product depth is just the same thin game available in more places where it will be ignored.

5.3. **“Our reputation is our most valuable asset.”**

Reputation with whom? If the answer is funding bodies, political contacts, and award juries, the studio is optimizing for the grant

cycle. Reputation with players—the only reputation that generates revenue—is earned by shipping games that convert strangers into evangelists. Everything else is résumé maintenance disguised as strategy. The studio with a strong grant reputation and no audience has built an excellent career in a profession that is not game development.

These are not exactly lies. They are survival stories. They help the studio avoid admitting that the current system may be incapable of producing the outcome it claims to want. They also persist because the public conversation about small studios provides few honest reference points. When nobody around you is saying “we did everything right and it still didn’t work because our operating system was wrong,” it becomes easier to believe the system is fine and the missing ingredient is just a better break.

---

6. **The way out is smaller than you think**

The studio does not need a transformation. It needs a **primitive**: a small, repeatable process that connects intention to evidence.

The primitive is the same one described in The Studio Primitive (https://gerolds.github.io/posts/studio-os/): **Promise → Proof → Cut → Ship.**

Promise: one sentence that binds the room. A constraint statement about the player’s experience that makes tradeoffs non-personal.

Proof: a ten-minute playable slice that works without progression, narrative, or a founder’s sales pitch. If a stranger can’t say “Oh, I get it” unprompted, the proof is not done.

Cut: every feature that doesn’t increase the probability the promise lands is guilty until proven innocent.

Ship: release the thing that was proven and cut, not the thing that was pitched.

But the primitive only works if there is a commitment (https://gerolds.github.io/posts/finding-the-commitment/) underneath it. This is where the studio in our pattern fails before it even begins.

A commitment is not a theme, a genre, or a mood board. It is the thing the team refuses to betray. It is what makes cuts non-personal and tradeoffs structural. Without one, every hard decision becomes a negotiation between tastes, egos, and whoever is most tired. With one, the room has a question it can ask: *does this serve the commitment, or does it erode it?*

The studio in our pattern typically has no commitment. It has a pitch. It has a set of features. It has a genre tag. But it does not have a single sentence that everyone in the room would defend under pressure.

This is why the team cannot cut: there is no principle to cut *toward*. It is why the team cannot prioritize: there is nothing that ranks one feature over another except effort and personal preference. It is why the game ends up incoherent. Each part was built to satisfy a different person’s idea of what the game should be, and nobody had the authority or the shared language to unify them.

The hardest part of adopting the primitive is not learning to prototype or learning to cut. It is committing at all. A commitment you keep is worth more than a better commitment you abandon. A team aligned on a good direction will outperform a team fragmented over the perfect one. The studio that commits to a B+ idea and moves will outship the studio that spends six months debating A+, because commitment compounds: every decision made under it reinforces the next, the mechanics and art and tone start to rhyme, and the game develops a coherence that cannot be faked and cannot be achieved by committee.

The studio’s current process produces the opposite of compounding. Each project starts from zero. Each debate is relitigated. Each person carries a slightly different mental model of the game. The commitment, if it ever existed, dissolved the first time it got hard—and no one noticed because nobody had written it down, tested it, or agreed to defend it.

---

7. **What compounding actually looks like**

Everything described so far is a pattern. Patterns are easier to see when you hold them next to the thing they fail to become.

I have worked with teams that compound. Not all of them shipped hits. But the ones that did share a texture that is unmistakable once you've felt it. It looks nothing like what the non-compounding studio imagines.

The first thing you notice is energy. Not the performative kind. Not motivational posters or all-hands pep talks. A quieter, more obsessive kind. Everyone on the team can describe the game in one sentence, and they do, repeatedly, to anyone who will listen. At lunch, at the bar, to partners, to strangers at a conference. They are not performing marketing so much as working through an idea they are deeply committed to. When someone describes the game and the listener's eyes light up, that reaction matters. When they don't, the team thinks about why.

The second thing you notice is the loop. Not as a concept on a whiteboard, but as daily practice. These teams build the core loop first, before systems, before content, before art. They play it ugly, in greybox, with placeholders everywhere, and they play it repeatedly. They playtest constantly. Not scheduled quarterly reviews: weekly, sometimes daily. They hand the controller to someone who has never seen the game and they watch. They do not explain. They do not hover. They watch where the player hesitates, where they lean forward, where they put the controller down. Then they change something and do it again tomorrow.

The third thing you notice is cutting. These teams cut with a speed that looks reckless from the outside. A feature that took two weeks to build gets removed in an afternoon because it didn't serve the loop. Nobody mourns it. Nobody argues. The commitment is clear enough that the cut is obvious. The team trusts the process more than it trusts any individual piece of work.

The fourth thing, and this is the one the non-compounding studio gets most wrong, is the origin story. The narrative the struggling studio tells itself is that success requires a long journey of mastery: years of paying dues, building skills, learning the craft, until one day the team is "ready" and the breakout happens. That story is rarely how it actually goes. The teams I've seen break out often do it relatively fast. Not because they are technically superior. Often they are not. Their code is messy. Their art pipeline is held together with duct tape. They have gaps in their skills that would make a

hiring manager wince. But they have two things the non-compounding studio lacks: zeal and a tolerance for constant reality checks. They commit hard, they test early, they adjust without ego, and they improve at a rate that compounds because every cycle is short and every cycle produces real signal.

This is where people reach for “talent” as the explanation. It usually isn’t. What looks like talent is often preparation that happened long before the studio existed. The people on these teams have often been training themselves to make games for a decade or more, privately and obsessively, as a hobby that was always more than a hobby. They played thousands of games not only for entertainment but for study. They modded. They made jam games. They argued about design on forums at 2 AM. They developed intuitions about what makes a loop work, what makes a moment land, what makes a player lean forward, years before they had the vocabulary to describe any of it.

When these people go to design school, if they go at all, they don’t learn design. They learn the *names* for things they already understand. They acquire models and language that let them talk precisely about what they’ve felt for years. They gain confidence to commit, because now they can articulate why their instincts are right. School doesn’t give them the skill. It gives them permission to trust the skill they already have.

The non-compounding studio is not usually full of people like this. It may have one or two. But the median team member chose game development as a career, not as a calling. They are competent professionals doing a job they find interesting. That is fine for many industries. It is not enough for an industry with extreme power-law distributions, severe oversupply, and a market that punishes “good enough” with invisibility. Making a game that cuts through usually requires people who cannot stop thinking about what makes games work, not because they are paid to, but because they can’t help it. That obsessive, self-directed preparation is the hidden substrate underneath many “overnight successes.” It just doesn’t show up on a résumé.

These teams still hit walls. They plateau. They have months where nothing works and the mood turns dark. But when they hit those walls, they have something to sustain them: **early signals that**

the game is real. A hundred thousand wishlists. A demo that went viral on its own. Playtesters who ask unprompted when they can play again. A Discord that grows without paid promotion. These signals are not luck. They are the compound interest on months of loop-first, cut-hard, test-always development. These signals make it rational to keep going. To grind through the plateau on savings and minimum wage for four years if necessary, because the evidence says the payoff is on the other side.

Compare this to the studio in our pattern. It also grinds. It also works long hours. It also “believes in the project.” But it grinds without signals. It persists without evidence. It calls this dedication when it is actually denial. The compounding team earns its stubbornness. The non-compounding team borrows it from hope and pays interest in wasted years.

The uncomfortable truth is that the gap between these two modes is not mainly innate talent, and it is not budget. It is long periods of self-directed preparation, combined with a commitment they can test, a process that generates signal, and the honesty to act on what the signal says.

The energy, the cutting, the speed, the breakout: none of it is mysterious. All of it is earnable. But not by a team that treats game development as a job. Only by a team that treated it as a serious study long before it became a job.

---

8. **Why it stays hard**

Most studios in this pattern will not change. The cost of change is felt immediately—discomfort, confrontation, the loss of familiar rhythms. The benefit arrives later, in games that earn, in skills that compound, in a team that trusts its own process.

The funding structure punishes change. Grants reward familiar shapes. Publishers want predictable milestones. Political contacts want a stable, reputable partner, not a studio that cuts half its features three months in. The team itself has adapted to the current system. People who thrive in low-accountability environments will resist a shift to high-clarity ones. Not out of

malice, but out of rational self-interest. The culture has selected for comfort.

Changing it means some people will leave. For a studio of ten, that feels existential. So the dream persists. The studio keeps aspiring to something it has no mechanism to achieve. The grants arrive. The projects ship. The games don’t sell. The team is tired but not confident. And the distance between the studio and its dream does not close, because the dream was never the problem. The problem is a company that survives on promises and heroics instead of on a baseline it can repeat.

The way out is not a better pitch or a bigger grant. It is a different operating system: one that treats evidence as currency, cuts as kindness, and the audience as the only funder that ultimately matters.

That operating system is small. It is learnable. It does not require genius or luck. It requires one thing the studio has never done: building a process it can trust more than it trusts its own hope.

Drafting assistance: Claude. All claims mine; errors my responsibility.