CASE STUDY · PROOF OF PLAY

# I owned a revenue-generating game, and built the tooling that let two people run it.

Pirate Nation Arcade was an experimental web3 spin-off of Proof of Play's flagship IP. I owned it end to end and built the content pipeline that turned an intractable data schema into balanced, shippable game content, so a tiny team could operate a live product.

Role: Product owner + tooling

Team of 2

~$500K in the first months

Abstract chain


Pirate Nation Arcade, a pirate roguelite spin-off I owned and operated.

## The challenge

Proof of Play wanted to spin its flagship IP, Pirate Nation, into a standalone arcade product on the Abstract chain, fast and lean, as an experiment. The bet only worked if a very small team could stand it up and keep it fed with content. There was no budget for a full studio.

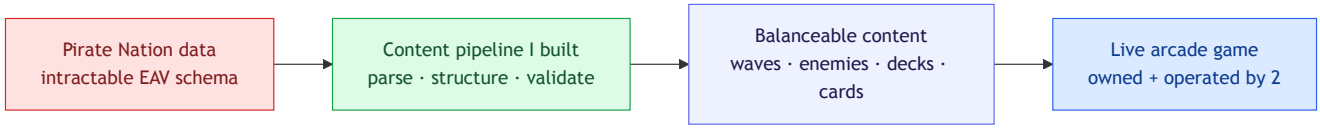
The hard part was the content. The game's data lived in an **EAV schema** (entity-attribute-value): properties scattered across thousands of rows, effectively unqueryable, and impossible to balance by hand. Authoring waves, enemies, decks, and cards by reading raw rows would have been slow and error-prone, exactly the bottleneck that sinks lean teams.

## What I owned and built

I owned the product end to end, scope, content, and operations, and personally built the **content and analytics tooling** that made a two-person team viable. The core was a pipeline that parsed Pirate Nation's intractable EAV data into clean, hierarchical structures fit for analysis and balancing, plus a data viewer that let me inspect and tune content directly.

This is where AI earned its place: I used it to turn a near-unqueryable schema into queryable, structured content, collapsing what would have been weeks of manual data wrangling into a repeatable tool. The tooling, not the headcount, is what let the product ship and keep shipping.

And the content itself was mine. I designed the cards, the enemies, the decks, the wave progression, and the meta the tool surfaces. So I sat on both sides of it: the designer deciding what should exist, and the engineer who built the system to author, inspect, and balance it.




The leverage: my tooling turned unusable raw data into balanced content a two-person team could ship on a schedule.

## The tooling, on screen

This is the PN Arcade Manager I built, narrated in my own voice. It inspects every card, enemy, deck, and wave, computes drop odds and average damage per turn, calculates leaderboard reward payouts to

player wallets, diffs test-net against production, and reads full battle logs to spot broken strategies.

## The product, in motion

## The outcome

The Arcade generated roughly **half a million dollars** in its first few months, run by two people. The point is not the headline number, it is the *shape* of it: a real revenue product, owned and operated by a team small enough to fit in a sentence, because the unglamorous work, wrangling data into shippable content, was solved once in tooling instead of paid for over and over in labor.

That is the pattern I keep coming back to. Find the bottleneck that forces a big team, automate it, and let a small team punch far above its weight on a live, revenue-generating product.

## By the numbers

**~$500K**
Revenue in the first few months

**2**
People running a live product

**504**
Commits in the content tooling I built solo

**1**
Pipeline that made an unqueryable schema balanceable

**Owned**
End to end: scope, content, and operations

**web3**
Abstract-chain spin-off of Pirate Nation

> *An owner-operated experiment at Proof of Play. The transferable lesson is leverage: solve the bottleneck in tooling and a tiny team can own and run a real product. Happy to walk through the pipeline in a conversation.*

Ben Young · Special Projects Lead, Proof of Play

**benyoung.ai** · **o.benjamin.young@gmail.com**