GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Wideload Games' *Stubbs the Zombie*

By Alexander Seropian

Introduction

Three years ago, after leaving the helm at Bungie I found myself without a job and on the sidelines of the game industry. It was weird. After spending six months tooling around at home, I realized I had to get out of the house and do something—be productive, set an example for my kids, take over the world—that kind of thing. Games are what I know and love, but I faced a real dilemma in trying to figure out how to get back to making games on my own terms.

To be an independent developer in the current climate of publisher consolidation and rising costs seemed impossible, but somehow Wideload was created. I challenged myself to create a company with a set of commandments essential to my personal and professional happiness.

The Commandments

- First Commandment: We shall establish our game's creative direction.
- Second commandment: We shall own our intellectual property.
- Third commandment: We shall not let a third party determine our success, such as the publisher who's doing (or not doing) the marketing, or the funding source (likely a publisher) making demands that are not in-line with our goals.
- Fourth Commandment: We shall have a small manageable team. We don't want 50 employees making one game over three years in house (we want low overhead), and we don't want to suffer the churn of ramping up and down for projects.

Development Theory

In a time when ten million bones hardly gets you a game, and development teams are crossing double-century headcount, I realized the key to these commandments was size—as in small size. Figuring out how to make quality games with a small team would solve the challenge of making original games, while remaining independent and having a shot at surviving that way.

Here's how the theory works. If the team is small, the overhead is low. Time equals money, so low overhead gives you lots more time to experiment and prototype (good for originality).

Additionally, every project starts small and ends big. But if you think of each project as a cycle of life, your company goes extinct pretty quickly when you have 75 people wrapping a project and then you only need 10 or so to start the next one. Staying small was the key.

Everything Works in Theory

There's no getting around the fact that shipping a major console title requires a lot of talented people. We took a page from Hollywood's playbook and decided to hire the "above the line" talent as the core Wideload team, but use outsourced independent contract talent to staff our production department. This would allow Wideload to have a consistent and manageable burn rate, yet work with a wide array of people who could provide the exact resources we would need. We also decided early on that we would license engine technology rather than create our own, as we did not want to spend the time investment and internal headcount cost to compete with the likes of Bungie, Id, and Epic.

Stubbs the Zombie

Our first project is *Stubbs the Zombie*, in which Stubbs, a wisecracking zombie, takes on an ultra-modern city of the future using nothing but his own carcass and the weapons of his possessed enemies. The gameplay consists of eating brains to create zombie allies, piloting various vehicles, and possessing enemies via a detached hand. Though the subject matter is mature, the mood and atmosphere is light.

We decided at the very beginning that Wideload had to establish itself as a brand. Our games should have a common thread that identifies them as something uniquely Wideload, and that thread is humor. What I'm most proud of in *Stubbs* is that everyone who has played it, reviewed it, loved it and, well...maybe didn't quite love it, agreed that it's funny.

Stubbs just shipped, and we built it using our outsourced production model. Putting our theory into practice was, politely put, a learning experience.

What Went Right

1. Brainstorm to living room in one easy step.

We only have twelve people on staff, and we all work in one big room. Our original business plan said nothing about the creative dynamics of

large versus small teams, but this is probably the single best result of our model. Granted, the acoustics of our wood-floored, brick-walled loft space stink because I'm too cheap to buy rugs, but we have a comfortable, casual, light, and quick workflow that allows anyone to blurt out ideas and have them propped or chopped on the spot. A small team doesn't need a lot of hierarchy, management, team meetings, strike teams, and a lot of organizational overhead, so we can focus our energies on being creative. It's hard to believe we didn't consciously plan this, but our emergent culture turned out to be a great side effect of our model.

For example, substantive, creative conversations often began when someone cracked a joke and the rest of us riffed on it. Unlike at a big company, where authority is always out of earshot, at Wideload we put those gems straight into the game! The dance battle in *Stubbs* emerged this way.

2. Freedom from the publisher's shackles.

Because we're small and our overhead is low, we were able to spend nine months working on various game ideas, mechanics, and story elements before we signed a publishing deal. This allowed us to take our time and experiment. It also gave us the power to say no to a few deals that weren't consistent with our commandments. We actually came within a few feet of signing a publishing deal early on with a major publisher, but it would have required us to give the publisher final say on creative decisions. We thought the deal could have been a disaster in a worst-case scenario, so ultimately we passed on it. Being able to reject a publishing deal could be considered having leverage, and for independent developers that's pretty rare.

We also had the ability to miss milestones. We didn't want to miss milestones, but because we could effectively manage our cash flow without having to live hand-to-mouth with each milestone payment, we avoided the situation of our publisher using money to take control of our project.

3. Cost structure.

A big problem facing game developers is the budgeting process. In order to secure project funding, you need to estimate cost and schedule on day one. Developers don't have a constant development platform (hardware and software are always changing) so making this projection requires a little voodoo science. Publishers tend to expect developers to stick to their budget and schedule, and often times that doesn't work. *Stubbs* shipped four months behind our initial plan, but two circumstances made our lateness workable. First, our overhead was low because our staff is small, and second, our contractors deliver specific assets to us for specific fixed prices, so if scheduling failed, the asset price did not increase like it would have if it were being developed by a full-time employee. As a result (to a certain degree) our contractors assumed schedule risk on our behalf.

The table below shows how much money the Wideload method saves per schedule slip, compared to the costs incurred by a fully staffed team.

Model	Headcount	Monthly Burn	Cost of Four-Month Slip
Wideload	12	100,000	$400,000

Traditional	45	375,000	$1,500,000

There were also points in the development of the game when we needed a little time to design something before opening the asset floodgates. We were able to prototype animation ideas for feel and gameplay without having a bunch of animators waiting around for us to get the plan together. Once our animation system and mocap list was ready, we pulled the trigger and got it done quickly.

What Went Right (Continued)

4. Staffing.

We only had twelve seats to fill. The team was split evenly between between artists, designers, and engineers. Basically, we had enough engineers to take the Halo engine and bend it to our will, enough artists to prototype and manage contractor submissions, and enough designers to write and script levels.

We assembled our core team very quickly. The prospect of trying to recruit and relocate 45 game developers to Chicago is daunting for a startup. If it costs roughly $10,000 to recruit and relocate for each position, then we saved $330,000 by keeping the team small. It's also hard to convince a publisher that as soon as they write that first check, 20 more people will come to work the next day.

There was also this great side effect to our model that when contractors didn't work out, we could simply fire them, which may sound cold and heartless, but the fact is that hiring presents a risk. When you bring someone on staff you create a semi-permanent bond. To break it is intellectually and financially expensive. When we had contractors that weren't cutting it, they were fired and we moved on. No one had to relocate. It was just business. No hard feelings. That fact made us quite a bit more maneuverable in terms of staff.

In addition, since our whole model is set up to find and manage contracted talent, we were able to add extra contractors when we needed to speed up production. We found ourselves pretty late in the game without enough scenery objects and no main menu. If we had to rely on already scheduled internal team members to handle these tasks, we'd have been screwed. We were able to find a lot of help externally to complete these two parts of the project.

The market for independent artists, designers, and programmers in the game industry is definitely growing. It's still small though, especially compared to film or television, where everyone is independent. Finding great talent is really important to us though, so we took advantage of all the tools we could to locate talent. We created a database of every company we could find that was doing contract work. We populated it with my personal contacts and those we found through resources like Gamasutra.com and Conceptart.org. We then began to make calls, evaluate reels, and meet with potential hires. Ultimately, and this should be no surprise, we had the best luck with the people that we had history with or who had experience with the tools we were using.

5. The Internet.

We had contractors all over the world contributing to *Stubbs*. SourceOffSite and instant messenger were invaluable tools for us. An important part of our process is making iteration time as short as possible. The more versions of something we did, the better it turned out. Giving our contractors the ability to create game-ready assets remotely and put them into our source control database allowed us to review submissions immediately, in-engine, and in the form that end-users would see.

Once we had that process set up, the feedback loop with the contractors tightened a lot. We also used online forums to develop concept art. I never thought in a million years it would make sense to do concept art with contractors, but it worked great. For our main character, we were able to get different artists to contribute their ideas simultaneously, which we could review and white-board online in real time.

What Went Wrong

1. Complicated Tool Chain.

Engine licensing is important to our model. We don't have the staff or desire to spend years creating an engine from scratch. On *Stubbs*, we used the *Halo* engine, which was great for us because the engine kicks ass and we all knew how to use it. But, with that said, the *Halo* engine had never been licensed before; there was no documentation and no internet forum or third party support for it. Any training our contractors got on asset creation had to come from us. The *Halo* engine has its own unique asset path and idiosyncratic behaviors, so the learning curve slowed us down and wasted time.

In some cases, we decided it wasn't worth training a contractor to produce game-ready assets; we'd just bear the burden of cleaning and importing the assets ourselves. This was tremendously inefficient. In other cases, we had to devote art director time to basic training. I think in the future we'll dedicate someone to tool training so as not to create a production bottleneck internally.

The many faces of death - Wideload commissioned a number of artists for *Stubbs*' concept art, and used the best of each.

2. Contractor Selection.

We could have done a way better job of vetting potential suppliers. We got lucky and found some incredible people to work with, but our selection process had three problems. First, not every asset class that went into production had a shippable asset reference to go with it. We made it a goal to develop the first version of every object type (character, vehicle, environment, scenery, weapon) internally and send that as the level-setting reference, but we got a little too enthusiastic in some cases to wait for that. This made it difficult to set the bar for everyone.

Second, not every contractor was required to submit a test asset. This was another goal we set, but again, we jumped the gun in a few instances, which was a mistake. Omitting this step allowed incorrect expectations to emerge and caused underbidding. In the future we'll set expectations of quality and scope for potential contractors before they submit a bid and start working.

Third, we underestimated how important good management and art direction is for contractors. We worked with one art house in particular that was stretched too thin and sold us on the A team, but gave us the B team. They experienced a bad cash flow squeeze during production, which strained our relationship. Additionally, their art director was not experienced enough, which made it really difficult for us to manage quality across their team. Had we discovered all this in the selection process, we would not have had to waste time replacing the contractor during the middle of production.

What Went Wrong (Continued)

3. Contractor Management.

We underestimated just how much time was required to manage contractor submissions. We knew it would be time consuming, but even with that expectation, the combination of art directing and art production was more work than we had time to do. We were short on producers and our artists were scheduled to produce content on their own. We didn't have enough bandwidth available for reviewing submissions in a timely manner. We realized too late that our production phase requires an intense focus on the work coming in from the contractors. Focusing our internal efforts on the contractor feedback loop should have been a higher priority for our art direction team.

4. Not enough producers, falling through the cracks.

Our project director doubled as our producer. This was bad. We let a few aspects of production fall through the cracks and as a result ended up dealing with our scenery object build and the game shell during post-production. It's hard to believe we didn't have the foresight for this, but our model requires serious production management. There are tons of assets to track and multiple parties contributing to the process. To think we shipped this game without a full-time producer is nuts.

5. Crunch Avoidance System—Failed.

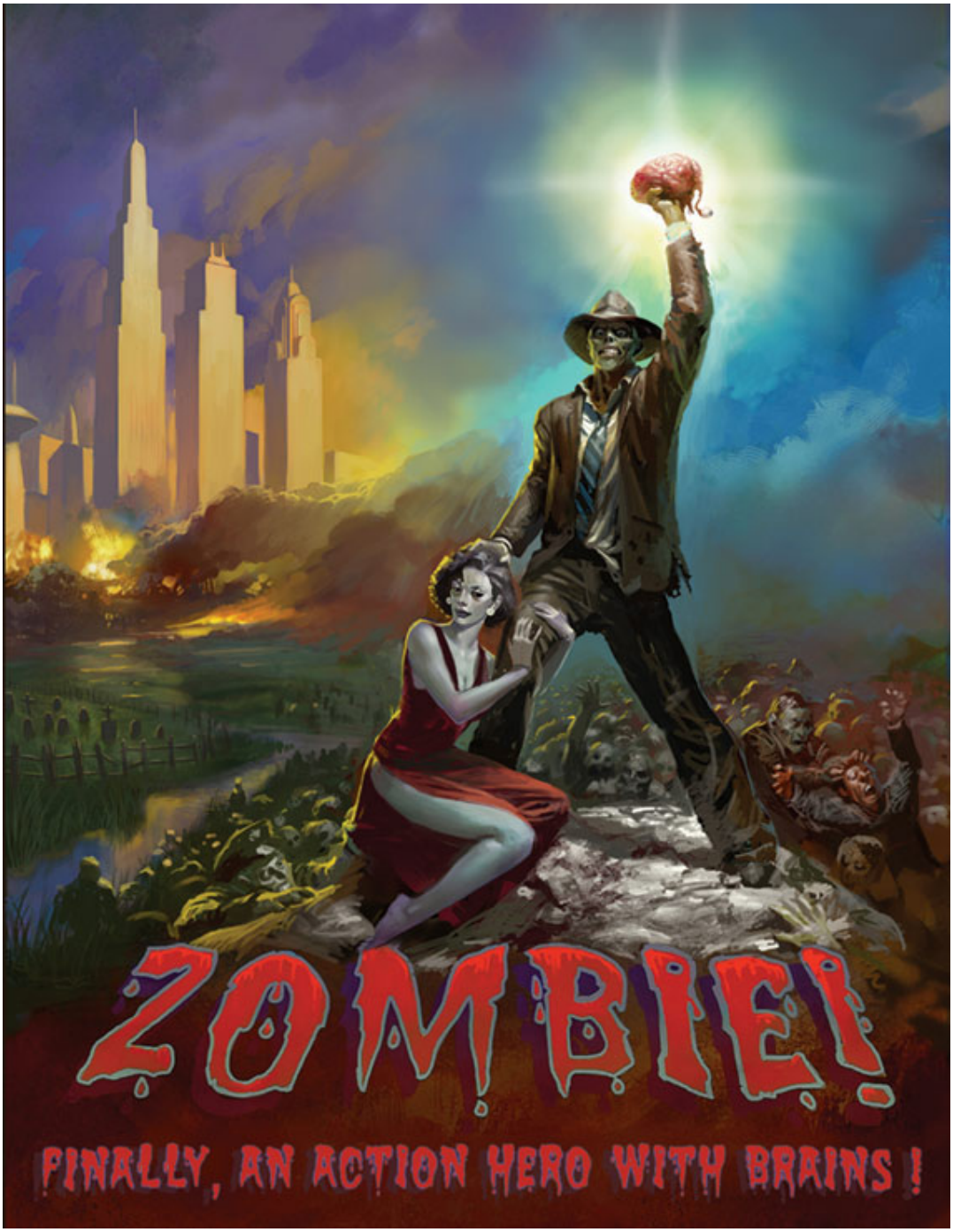
I had this crazy idea that since the bulk of the work was being done by contractors, we would be managing them to deadline and we at Wideload wouldn't have to work crazy long hours. This logic was used to form the basis of our "crunch avoidance system," and it was an abject failure.

We crunched for a solid three months, which isn't too bad relative to past experiences, but is way worse than zero. Because we let some major components slip into post-production and were four months behind schedule, and we let ourselves get behind on contractor approvals, we ended up with more than post-production tasks during our post-production phase. Quality of life is a big issue for game developers. It certainly is for me, having young kids at home. I still hope to create a better work/life balance using this model.

On future projects, we will endeavor to keep production phase deliverables comfortably within our production phase. For us, the key to this is good production management and timely feedback to our contractors. If we can focus on post-production during the last three months, we can avoid working double duty through the end of the project.

Experimenting Zombies

Throughout the development of *Stubbs*, I received lots of calls from industry friends who asked me, "How goes the experiment?" *Stubbs* was released for Halloween and has gotten some solid reviews. Wideload survived the process and will strike again.
Pragmatically speaking, the experiment has been a resounding success. Adhering to our commandments resulted in an original game that has succeeded in large part because we had the freedom to take risks. Had we done things the old-fashioned way, where the publisher has all the leverage, our lovable undead anti-hero might have met a tragic end at the hands of a focus group.



That said, I wear new battle scars and have tattooed new lessons to the back of my hand so as to never forget. For anyone hoping to follow our blazing trail, or at least learn something from our foibles and fables, I hope this article helps.

Return to the full version of this article