Postmortem: Treyarch's 2002 hit, Spider-Man

PRINT

By Jamie Fristrom

In this reprint from the August 2002 issue of Game Developer magazine*, Spider-Man *developer Jamie Fristrom writes about what went right and what went wrong with the game's development process. Fristrom's current project,* Energy Hook*, strongly builds upon some of the design principles first explored here and in* Spider-Man 2*.

Treyarch was finishing up *Max Steel* and *Tony Hawk's Pro-Skater 2* for Dreamcast when we agreed to do a game that would tie in with the *Spider-Man* movie, and release it simultaneously on all three of the next-generation consoles: Playstation 2 (PS2), Xbox, and Gamecube. We formed a new team out of parts of the others to begin work on the proof-of-concept and design. This team of four programmers, four designers, four artists, and a producer wasn't starting totally from scratch; we had Activision's previous *Spider-Man* game for the Playstation (PSX) to look at.

We wanted to improve some of the things about the game, such as giving the web-swinging more freedom, and play to that game's strengths, such as the hostage modes and variety of ways in which you could be Spider-Man. We also wanted to add a whole new type of gameplay: aerial combat, the ability to take on flying villains as you swing around Manhattan. Sony/Columbia gave us a ton of concept art and stills from the movie on which to base our work.

The PS2 was our lead SKU, because it has the largest installed base and was the easiest to get development kits for. We figured if we could make our game run on the PS2, we could make it run on anything. We began work on the Xbox a few months after the PS2 and the Gamecube a few months after that.

What Went Right

1. Good people.

"All you need is good people," says the submarine commander in *Das Boot*, after the chief engineer repairs the boat and saves their lives. It's true, with good people you don't need to enforce process because it happens automatically. Everyone makes sure they do a good job: they volunteer for code reviews, they write their own unit tests, they find better ways to do things instead of the ways passed down from on high.

Our team was made up of a number of talented individuals, who each made their own unique, lifesaving contributions. Even the interns were amazing.

Because I'm not in HR, I don't know how we got these people, but I think part of the reason is that the people in charge of interviewing know their stuff, whether it's art or code. Part of it is the referral bonus we give employees for recommending new hires; part of it is that Treyarch is an environment not too many people are willing to leave. (We've seen what's out there, and it's worse.) Part of it may be the free soda, and free dinners during crunch time, and part of it is our policy of not hiring just anyone to fill a position. (We do hire out of desperation occasionally, but rarely. And when we discover we've hired someone who is just average, we let that person go.)

This team was not only talented but also motivated. Even though adding unplanned features was discouraged, many of us stayed late, on our own time, to get stuff in we thought would make a real improvement to the game. This is where the playing-as-Green-Goblin mode came from, and the secret bowling level, the rats in the sewer level, and a lot of special effects.

2. Developed cross-platform libraries and intermediate file formats.

Treyarch was developing a few titles for the next-generation platforms, and it was obvious that having one crossplatform library to do the rendering, which all the teams shared, would be a big advantage. We formed a new team, Next-Generation Libraries (NGL for short). The *Spider-Man* graphics/PS2 programmer split off from our group to lead the team, providing the architecture and API to which the various platform graphics libraries would be written, and he developed the initial PS2 graphics library.

NGL had disadvantages as well as advantages: the advantage of more efficient use of coder resources was balanced with the worry that we might not be able to rely on a separate team. The NGL team wasn't always kept in the loop about when our deliverables were due and sometimes weren't available on a weekend or night when they were needed. Finger-pointing would occur: is it an NGL bug or a client-side bug? And sometimes finger-pointing didn't happen when it needed to: a client-side bug would linger on the plate of an NGL programmer who didn't know how to fix it.

At times, the NGL programmers became de facto *Spider-Man* programmers. They were building the *Spider-Man* code base on their machines to make sure their changes worked with it, to optimize for the game's worst cases, and to

make sure the Xbox and Gamecube matched the PS2 close enough. In the end, NGL worked out great, finishing features and fixing bugs in time to ship.

---

3. Engine reuse.

The first decision we had to make was what engine to use. (Writing one completely from scratch was out of the question for an 18-month project, a lesson we learned the hard way with *Draconus*.) We had access to the previous *Spider-Man* PSX engine through Activision, but our designers and programmers were used to the engine we used for *Max Steel* and *Draconus*, an engine that was already next-generation and cross-platform. Our engine had a powerful scripting language, but it also had slow turnaround times and was never intended for a *Spider-Man* game.

Despite its shortcomings we felt that this engine was the one to use, and it turned out we were right: we were able to get Spidey swinging through Manhattan on the PC in record time. Now all we had to do was port it to three platforms and add as many features as we could.

4. Good process.

At its heart, our methodology was "code and fix." However, there were many semiformal processes that prevented our software cowboyism from totally spiraling out of control.

The artists and designers were scheduled in Microsoft Project. For the coders we started with an XP-like system of file cards representing weeks, posted up to a large corkboard. But the schedule changed so frequently that this method stopped being good enough, and we switched to Joel Spolsky's method from www.joelonsoftware.com. This worked pretty well; I would go through the schedule every morning to make sure everyone was keeping their schedules up to date. Also, once we switched to this system our estimates tended to be more accurate, which was a nice side bonus.

Using Microsoft Access, we started logging bugs right away. People would log bugs by sending an e-mail to the producer, who would first log the bug into the database and then later print out paper lists for people. We broke bug priorities down into: 0 — don't leave your desk until it's fixed; 1 — fix ASAP; 2 — fix ASAP unless you're holding somebody up; and 3 — this one can slide until the next milestone. Our policy was to fix the bugs first, which meant that most bugs were marked with priorities of 2 or lower. As the project grew, it became clear that the work of maintaining the bug database was sucking down most of our producer's time, and after trying an off-the-shelf bug tracker that didn't meet our needs, we built one in-house using Streamline Technology's Seven Simple Steps, which all of the teams at Treyarch use. It's an Access back end with a web front end that e-mails you when you have a bug. This not only freed up a lot of producer time but also became one of the main tools we used to communicate tasks and bugs to others and to make sure those tasks got done.

On average we broke two levels a day, usually due to simple things like not checking in a new file. To protect ourselves from these errors, we did daily build and smoke tests on the PC and the PS2. We did the daily build on a machine that would do a complete update of the data and source depositories, rebuild and recompile the various intermediate files into their final output, and run an automated test of every level, just to see if they ran. (Generally we've found that 95 percent of the bugs we introduce are of the variety that makes levels stop loading at all.) This way we'd catch bugs up to a day after they were introduced, instead of discovering the hard way, days or weeks later, that a level nobody had touched in a while didn't work.

On the art and design side of things, we would storyboard a movie or create concept art for a level before we began animating or modeling it. A professional writer wrote the script with all the voiceovers. Before we designed a level, we would have a level implementation meeting, where the participants in creating that level would discuss what the level was going to be before it was modeled and scripted. Because of this process, several of our levels were very close to good-enough-to-ship on the first iteration, although several other levels had to be revisited several times before we signed off on them. We had a concept of "level alpha," which meant the level was basically ready to go in the box except for cutscenes (scripted or animated) and voice-over, and game designers didn't work on the next level until they had their previous one at "level alpha."

Finally, we had a small internal testing department. It's fairly standard in the game industry to wait for a game to reach so-called "alpha" (a completely nebulous term) and then put it into QA. Then QA tests the game out of sight of the developers and submits bug reports. Although we did do the standard external QA with Activision, before the game was at alpha we had internal testing. At first this was just one person, but as we got closer to finishing, the number grew. And once the game hit alpha, Activision sent some of their best and brightest testers over to Treyarch (which was easy, since our offices are one block from theirs) to test the game on-site, so they could demonstrate bugs in person, work with the very latest revisions, do testing on development kits, and ask us when they discovered bugs whether or not the bugs were important. They became part of the team, and I think that because we could meet them and see them face-to-face we accorded them more respect than the faceless testers at Activision. By the end of the project, half of our bugs were caught internally; although there were many duplicates, there were many more bugs that external QA never saw.

5. Communication.

The layout of our office was geared to encourage communication between the people who needed to communicate. The leads' offices were fairly central in an L-shaped office suite. The game designers were close to the programmers who supported them, which was key in getting things to happen and encouraged additional lunchtime communication. The programmers who dealt with specific platforms were farther away. NGL was on a different floor, but their lead was in a nearby office, and NGL representatives would share offices with us when it was useful for them to do so.

All of our design documents were made available on an intranet web site, along with short documents on how to use the tools and a small FAQ that had some common troubleshooting answers.

Finally, we had a lot of management. Greg John oversaw the entire project. There was a producer watching deliverables. There was a creative director. There was a level-modeling lead. There was an animation lead. There was a game design lead. There was a lead programmer. The engineering group — the programmers who wrote the platform-independent, gameplay-related code — had their own lead. Each console had a lead go-to guy who understood that console best. NGL had a lead programmer and a producer. In all we had about one lead for every five people, and all the leads met once a week.

Although communication was good, it could have been better, particularly between the art and design departments. That's something we'll have to work on for the next project.

What Went Wrong

1. Not enough staff soon enough.

Some of our other problems can be traced back to staff size. When the project started, we had about four programmers, four game designers, three modelers, one texture artist, one producer, and one creative director. NGL was only in our imaginations at this point. Treyarch was eagerly looking for more staff, but staff doesn't just come out

of thin air. Most important, we didn't have a concept artist, we didn't have a writer, and we had so many people working on the proof-of-concept that we didn't really have someone to come up with a full design.

Later, when we rolled into full production of our inadequate design, things got even worse, because some of our veterans were taken away to work on other projects, without adequate replacement. Eventually we were fully staffed and then some; we had more than 40 people at points during the project, and more than 50 worked on it at one point or another. We had to make up in the end for what we lacked in the beginning.

2. Inadequate initial design.

I think designing is overrated; coming up with a 200-page design document only to scrap most of it seems like a waste of resources. Still, you need some design — a bad plan is better than no plan — and you need more than we had.

By the time we finished our proof-of-concept, the design consisted basically of a list of bosses, a list of levels, and some ideas on how our new AI system might work. Each item had about a paragraph devoted to describing it. Our story was a page long. By the end of the project, we had a wealth of design: we had the output of each level implementation meeting; we had the script; we had a couple of pages on each boss, describing the moves he would be capable of; we had concept art and storyboards; we had concept art for the front end. If we had come up with that material earlier in the project, we would have been rudderless for less time and could have made more game before shipping.

3. Audio and voice-over problems.

Because we didn't hire a scriptwriter until fairly late in the project, we didn't get the script approved by Activision and Sony/Columbia until even later. By the time the people who make these decisions had decided to accept the script and get Willem Dafoe and Tobey Maguire to do the voice-over, we were long past the drop-dead date we had given to Activision. It was already alpha, and we had very few of the cutscenes done, because it's a waste of resources to do the cutscenes until you have final voice-over. But we soldiered on. We got the final voice-over recorded and transplanted animators into the *Spider-Man* animation sweatshop to get all of the cutscenes finished as soon as possible. The end result is that a lot of our cutscenes don't look as good as we would like.

Audio also suffered from massive disorganization. We didn't have anyone on our team dedicated to managing audio resources. What we needed was a good directory structure, file-naming conventions, and the like to make dropping in final sound effects easy. Too many times we put in a placeholder sound effect without changing the name, resulting in massive confusion about which sound file went with what. After those issues were all resolved, we discovered that the console versions didn't sound the same as the PC version, due to discrepancies created by the different tool chains on the different consoles. We were working out these problems right up until we shipped.

4. Too much done in script engine.

One of the most valuable things about our game engine is CHUCK, the script language named after its creator, Chuck Tolman. It's a C++-like script language that emulates multithreading, allowing you to have multiple game entities processing their individual scripts at the same time. It's used to script game events, trigger audio, change AI states, speed up or slow down the whole game or individual entities, and even place front-end widgets and text on the screen. One can prototype whole new kinds of gameplay in CHUCK with very little programmer help; this is where the stealth mode of *Spider-Man* came from, for example. The game designers on *Spider-Man* are really programmers in their own right. They are empowered.

Because CHUCK is such a powerful script language, however, game designers would often take on tasks that could have been handled in code better. CHUCK doesn't really have arrays, and it doesn't have a debugger, so for anything complicated, C++ is the way to go. Unfortunately, it wasn't always the way we went.

The worst example of this may have been the front end. Although programmers did the work, we thought it would be clever to write it in CHUCK instead of C++ with the idea in mind that this would open it up for more people to maintain it, if necessary. This idea worked in a way: when we got to the eleventh hour and we still hadn't finished the front end, the game designers stepped in and helped finish it off. If we had done the front end in code, we would have finished it sooner and the game designers wouldn't have had to step in at all.

Again, this is a result of our staffing difficulties, because if we'd had more staff sooner, programmers could have provided these features in code. Instead, the game designers got fed up and did it themselves, costing us efficiency in the long run.

---

5. Not enough QA before alpha.

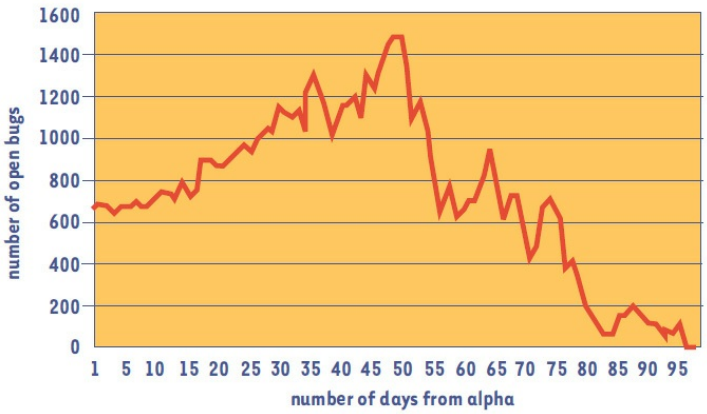
Nothing prepared us for how many bugs we'd find in this project. The total came in at around 16,000, about half of which were internal and half of which were external. This was more than twice as large as the largest bug count we'd

ever seen on a project at Treyarch before, partly because we were on three platforms and therefore had more bugs. (The same bug on three different platforms might show up as three or even six bugs, as PAL SKUs were being tested independently from their NTSC counterparts.) Still, even though our find rates were high due to the number of platforms, our fix rates were quite ordinary. Our open bug graph looked like Figure 1.

Watching our bug count was like watching the stock market, and we felt lost at sea. On previous projects we'd been able to guess fairly accurately how many bugs we'd have and develop a good idea when we'd be complete. On this project, all we knew was that it was going to take longer than we thought, and possibly a lot longer. All we could do was work extra overtime and mark "will not fix" or "as designed" on as many bugs as we could. At this point in the project, it's a gray area between a "bug" and a feature that "really has to be there." Although we fought against as many changes as we could during this period, most of the battles were lost, as we reluctantly agreed that the opportunity justified the risk of these features. If we'd fought any less hard, or lost any more of those battles against unplanned feature changes, we would not have made our ship date.

Normally we like to have the game in testing for a week after we hit zero bugs, to make sure that there aren't any lingering, hard-to-detect problems. We didn't have that luxury on this project. A week before our final date to submit we still weren't at zero bugs. All we could do was be careful during this week with our fixes: access to SourceSafe was restricted; the team was admonished that they mustn't fix any more low-priority bugs, lest they introduce a stop-shipment bug; and we did informal inspections on all source changes. When we finally hit zero bugs, we worked overnight to get the burns out and submitted the next morning.

While our submissions were cooking at the console manufacturers, we continued testing at Activision, so we could find any problems before the console manufacturers did. We found about half a dozen problems that we would have loved to fix, but they were either very rare or not serious enough to warrant a resubmit, although they were quite embarrassing.



We haven't worked out the details of how to reduce our bug count for the next project; more and better and different QA is necessary, but how much and what kind? Formal code reviews? Unit tests? More black-box testing? More soak tests with random monkeys? Maybe we should make asserts and developer-eyes-only error messages fatal, to force people to stop and fix these problems instead of working around them. Maybe we should rely less on the PC version, so we can catch console-specific bugs sooner. Whatever we do, I hope it'll go into the "What Went Right" section of Treyarch's next Postmortem.

One View of Many

This article represents data sifted from the internal postmortem we did at Treyarch. After we shipped the NTSC versions, everyone wrote their lists of what went right and what went wrong on the project. Categorizing that data was difficult, and my lead programmer bias has seeped into what's been presented here. If senior producer Greg John, creative director Chris Soares, or design lead Tomo Moriwaki had written the article, you would have seen a much different view.

Due to the massive movie marketing, the *Spider-Man* phenomenon is huge. Being a part of it, seeing *Spider-Man* paraphernalia everywhere, is exciting, and partly makes up for the frustration.

But what really makes up for the frustration is the fact that we somehow pulled it off: an original title, three new platforms, 18 months. And so far, it's doing very well.

This post-mortem originally appeared in the August 2002 edition of Game Developer magazine.

Return to the full version of this article