Postmortem: 2K Boston/2K Australia's BioShock

By Alyssa Finley

[Gamasutra is proud to be publishing notable Game Developer magazine postmortems online for the first time - starting with project lead Finley revealing the creation of 2K Boston/Australia's seminal BioShock.]

The story of developing *BioShock* is an epic one and isn't easily expressed in 10 postmortem points. The team and the game changed remarkably over the course of development. A company was acquired. The team size doubled. The product focus changed from RPG hybrid to shooter.

It's easy to talk about the processes we used to develop the game, but it's harder to describe the creative spark that somehow managed to turn the most unlikely of premises (a failed underwater art deco utopia set in the 1960s) into a marketable shooter. It took a visionary to make the creative choices to guide the game, and an incredibly talented and hardworking team to bring that vision to life.

What Went Right

1. Every demo tells a story.

Demos were galvanizing moments for *BioShock*. They led to a unified team vision, identification of problems and solutions, external excitement, and internal support. For example, the project was signed after GameSpot ran an exclusive feature based on a single-room graphics demo.

Since *BioShock* was a relatively unknown IP outside the game development community, the public's impression of it would be critical to building the buzz we needed to make it a commercial success. As a result, every time we took the game out in public, we put great thought into the message we wanted the demo to deliver and the level of polish of the presentation.

Our first public presentation was at E3 2006. We had developed a great deal of content before that point, but hadn't yet built a space that really demonstrated the game experience to our satisfaction. The E3 demo forced us to focus the whole team on what the user experience should be. We defined a message for the demo- player choice-and built a narrative around that message. Even though the experience was highly scripted at the time, it effectively demonstrated the feel of the game we wanted.

Another example of demo-inspired development was the "Hunting the Big Daddy" demo. Though Big Daddies and Little Sisters had been part of the game in some form since the beginning, initially the player could confront Little Sisters directly without necessarily needing to dispatch the Big Daddy that protected them.

During the development of this demo, the team discovered that with some polish and tuning changes the act of dealing with a Big Daddy could be a truly epic battle in itself. This led to the realization that Big Daddy battles should be the key to player growth, essentially providing a roving boss battle that players could undertake at a time and place of their choosing.

Another example is the graphical effects on the player's hands when using plasmids, which came out the first *BioShock* trailer created with Blur Studios. In that cinematic, the player uses a hypodermic needle to make his arm into a weapon; after the injection the protagonist's skin blackens and swells and angry hornets burst out of it to attack the Big Daddy.

When working with Blur to develop the trailer, we knew that the sequence didn't accurately reflect the game's visuals, but we did it because it really captured the vibe of what the "genetic modification" part of the game was all about.

2. Course corrections.

One of the true successes of *BioShock*'s development was our ability to identify and react when the game was not shaping up to become what it needed to be. For example, the first vertical slice prototype we built was an non-navigable linear corridor shooter that looked like it took place in an abandoned box factory.

It didn't provide a compelling experience as either an RPG or a shooter. In response, we threw away that prototype and started again from scratch with the goal of building a single room that felt like the ruined underwater utopia we were trying to build.

First we did concept art passes. Once we got a concept that worked, we built it. Then we used it as a demo space. We used that single room (now Kashmir Restaurant in the first level of the game) as an artistic reference that guided us in creating an aesthetic unlike any other game on the market. (For more about the artistic style of *BioShock*, see the free art book downloadhere.)

Each department went through a similar crisis moment over the course of the project. These frequently came as the result of the demos, but not always. At one point, when facing a shortfall of programmers and an overflow of tasks, we proposed removing physics objects from the game entirely in favor of having only large, constrained physics actors.

This would have allowed us to spend much less time tuning the physics of individual objects while allowing the world to seem somewhat dynamic. However, doing so would have removed a huge level of interactivity from the game, so that decision was corrected relatively quickly.

In terms of design, we created a depth and density of game systems that fit into a game about character building and choice, but would not have been competitive as an FPS. Around the time that the game went into alpha, we took a hard look at that gameplay and realized that, although there were many choices, they weren't very compelling.

This was because we hadn't been thinking as much about making a shooter as we should have, and many of our key interactions (weapons tuning, plasmids, length of AI engagement) were designed and tuned for a slower and more cerebral experience. To put it another way, nerdy RPG-like stat changes just didn't seem meaningful in the vibrant and dangerous world of Rapture.

Once we recalibrated the game to be more like a shooter, we simplified many of the deeper systems tremendously so that the user would be able to understand them. We also put more polish time into the core interactions of the game, such as the weapons, plasmids, and user interfaces. We ended up with fewer choices overall, but each one of those choices was infinitely more functional, understandable, and fun than the previous ones.

It was inevitable that we lost some progress due to these major corrections. But the team's ability to pull together and address the fundamental problems was amazing, and the results were well worth it.

---

3. Input from outside.

The first external *BioShock* focus test was meant to be a sanity check: to get a better sense of what was working well but needed polish and what wasn't working at all.

At this point we had already done one small round of internal focus testing with friends of friends, which had turned out mostly positive feedback. So, just after the first beta, the entire design team plus a contingent of 2K producers headed off to see how a group that knew nothing about our company or *BioShock* would react to the first level.

It was brutal.

The first level, they said, was overly dense, confusing, and not particularly engaging. Players would acquire new powers but not know how to use them, so they stuck to using more traditional weapons and became frustrated. They didn't interact with the Big Daddies, and they didn't understand (or care) how to modify their characters. They were so overwhelmed by dialogue and backstory that they missed key information. A few of the players did start to see the possible depth of the game, but even they were frustrated by the difficulty of actually using the systems we had created.

Based on this humbling feedback, we came to the realization that our own instincts were not serving us well. We were making a game that wasn't taking the initial user experience into account, and we weren't thinking enough about how to make it accessible to a wide variety of players.

After the focus test, we went back to the drawing board for the entire learning sequence of the game. We scrapped the gameplay in the first two levels entirely and re-architected them to be a much slower paced experience that walked the player through the more complicated gameplay verbs, such as "one-two punch"-combining weapons and plasmids. We changed the medical pavilion from having sandbox-style gameplay to using a series of locks and keys that were set up to ensure that the player knew how to use at least a few key plasmids. And we made a development rule that future changes would be data-driven, not based solely on our own instincts.

After the first round of changes, we had two rounds of internal 2K play testing to gather more data about the user experience, releasing builds of the game to several 2K studios and soliciting feedback about how far people got and which weapons or systems they enjoyed. We received feedback from 2K game analysts, Microsoft, and a few other advisors.

When we brought the demo back to focus testing, which was barely a month before we were (then) scheduled to complete the game, the experience was very different. Although players still got stuck and frustrated at various spots, they understood the game systems and saw the potential inherent in them. While we still had work to do to make the game more accessible, at least now the problems were much more easily solvable.

4. Small empowered teams.

While developing our first internal demo, we realized just days before completing it that it was on the wrong track. By that point it was too late to take on all the problems in the demo, but we decided to try to improve the core interactions. We used a small, focused strike team approach to target and solve AI problems, choosing one problem at a time, analyzing and tackling it, then moving on. Although this approach wasn't enough to salvage the original demo, it was recognized in our internal postmortem of the demo as an effective process that we should do more often.

One of the most visible successes of the strike team system is the tuning of the weapons of the game. All the weapons had been in and working for several months, but as the game got closer to content lock, they still weren't feeling as good as they should.

To tune each weapon, a team consisting of one designer, an animator, a modeler, a programmer, the effects specialist, and an audio designer held a kickoff meeting where they analyzed and brainstormed about each aspect of a single weapon. They came up with a task list for each team member, went off to work for a day or two on their tasks, then came reviewed all the results. When they were satisfied, they moved on to the next weapon.

Over the course of development, we created multidisciplinary strike teams to work on a wide variety of problems, including AI, animation, visual effects, and cinematics. The results of those teams were universally better than the previous non-iterative process.

The 2K Boston *BioShock* team.

5. Talented people, flexible staffing.

BioShock was initially scoped to be developed in about two years with a small team of 30 people-25 in Boston focused on gameplay and five in Australia working on the core engine. As the team completed successful milestones and demos, and made strong cases for more development resources it became clear that we needed to tap into the Australian office.

Initially, Australia was intended to supply a small core technology team that worked on the renderer, engine, and core tools and processes for console development. The Australians had a tremendous impact on development because by taking care of the core engine and pipeline tasks, the Boston programming team was free to focus on gameplay systems and production.

One of the fastest and easiest ways to staff up any newly-opened position on the *BioShock* project was to pull from the Australian team. By the time *BioShock* went gold, almost everyone in the Australian office had worked on the game in one way or another.

The huge advantage to using the Australian team resources was that they already knew the engine and the game, and had easy access to the core technology team. They came up to speed incredibly quickly, and could be productive almost immediately upon getting project tasks. And although the time difference made communication a challenge, it also meant that critical bugs could be worked on literally day and night.

What Went Wrong

1. Evolving Product Positioning.

The spec of *BioShock* changed so much over the course of development that we spent the majority of the time making the wrong game- an extremely deep game, and at times an interesting one, but it was not a groundbreaking game that would appeal to a wide audience.

We knew from the start that we'd have to make late changes to really bring the game to life-we had even built our original schedule to allow for six months of finalizing-but the amount of change that we ended up needing seriously exceeded our remaining schedule. Ultimately, we were very lucky to get an extension in the eleventh hour.

Part of the reason for the late course change came from not having our internal product message clear from the beginning. *BioShock* had initially been positioned as a hybrid RPG FPS. The decision to reposition the game as a focused FPS came later, after our initial production phase in summer of 2006. Had we been working with an FPS mentality earlier, we could have made better use of our time.

Another contributing factor to the late switch was that the game had been more or less proceeding according to plan throughout development, so there didn't seem to be any emergencies that needed intervention from higher levels of management. Milestones were completed, goals were met, development seemed to be proceeding uneventfully. But as the game neared alpha, key people began looked more closely and saw that *BioShock* wasn't on track to become an accessible and marketable game.

As mentioned in the first What Went Right point, the real turning point for *BioShock* came when we had to present the game to the outside world, which forced us to carefully consider the story and takeaway message. In retrospect, we should have tried to develop some of that thinking sooner.

2. Narrative content development happened late.

We had many drafts of the story over the course of development, but the final draft turned out to be an almost complete rewrite. To make matters worse, we failed to fully exercise the narrative production path in early versions, so once the final draft was complete and recorded, many implementation and pipeline problems appeared for the first time that should have been caught and resolved earlier.

The core issue was that a giant pile of content came online well past beta, and the team had to scramble to get that content correctly installed while also fixing bugs. Competing demands for time and resources meant that, unfortunately, some of the important narrative details of the game weren't created until the final rewrite, and therefore required quite a bit of work to retrofit them into an existing game.

To add to our woes, the first focus test feedback on the narrator's voice came back extremely negative. People found the character extremely off-putting, so we recast the part at the last possible moment. On the positive side, this allowed us to refine several areas of the game (including the intro sequence) to ensure that the player knew what to do at the right time. On the other hand, it was difficult to get all the content in, debugged, and polished in the remaining timeframe.

3. Scaling vision to team size.

Our goals and vision pretty consistently overreached our production capacity. Ideas that started out small turned out to require a tremendous amount of coordinated support to reach a polished state. Although we were able to add resources regularly and make some cuts late in the game, our ability to plan for how much work it would take to bring any single idea or space from concept to completion was poor.

In addition, we didn't have an effective internal review process set up until very late in the development cycle. We would work on levels to the definition of a milestone, get feedback, and then set it all aside for a while or leave it in the designer's hands to polish. It wasn't until we created a more regular review cycle, where all the key players sat in one room and watched the gameplay session and someone logged all the bugs, that we were really able to define the amount of remaining work to bring a feature or level to completion.

The ultimate reason we were able to pull off the game we made to the level of polish we did, was the sheer dedication of the team working on it. Even though we had padded the finalizing schedule, we still far exceeded it.

It's to the team's credit that they stepped up to the challenge with incredible dedication. The final crunch period on *BioShock* was long and hard. People who had been pacing themselves for six more weeks of work had to reset to three more months of work rather suddenly. Had we understood our polish cycle requirements sooner, or had known about the additional time earlier, we could have paced ourselves better.

---

4. Inefficient processes and tools.

Many of the processes and tools we used to develop *BioShock* were inefficient or confusing in implementation, leading to slow iteration cycles and bugs. Using a modified version of the Unreal engine, which the team had already used to ship two previous games, gave us a huge head start in developing *BioShock*. The gameplay team was able to mock up a playable version of the core game mechanics in just a few months, and the team's familiarity with the tools allowed us to get new gameplay spaces up and running quickly.

However, the ease and familiarity of the workflow often led us to accept a solution that was faster to implement but slower to use rather than taking the time for a more efficient implementation.

For example, there was no good convention for how to name script actions. Depending on the system, one script action might be called "Change < SystemProperty > " while another would be called "Set < SystemProperty > " or "Modify < SystemProperty > ". With hundreds of scripting actions available, designers often spent way too much time searching for the right tool to use. This could have been avoided with a scripting code standard.

The content baking process for the console was time-consuming and difficult to troubleshoot. Frequently the only way to either identify or resolve a bake problem was to re-bake at the cost of up to an hour of work, and if the tools were actually broken in some way, it would take at least another bake cycle to be able to work effectively again.

Once we reached crunch time, it was extremely painful to have to wait for the bake process to complete when people could have been working productively instead. We should have put more energy and time into speeding up the bake process sooner.

5. Poor data collection.

One of the most frustrating things about our decision to be more data-driven in tuning the game was the lack of actual good data to base that tuning on. Our game log system was barely adequate to analyze single play-throughs and became completely unwieldy when trying to analyze a single log file containing data from multiple play-throughs. We had no good methodology to define what information was logged and at what level of detail, so the job of parsing out the logs into understandable "gameplay metrics" was painful, slow, and ended with inadequate results.

To further complicate the problem, most people in the office used shortcuts or cheat codes at the start of a level rather than playing from the beginning of the game, which caused us to base a tremendous amount of early tuning on a shaky set of assumptions about how players would choose to build their characters.

Blockbusting

Our goal when we set out to make *BioShock* was very clear. We wanted to get to the next level, moving beyond our suite of critically acclaimed games to make a blockbuster. A lot of factors aligned to make this possible: the commercial backing of 2K; the game design knowledge we'd acquired from building *System Shock 2*; the technological familiarity with our Unreal-based engine that we'd built with previous games. But we still had to figure out how to make it all big-blockbuster big.

A lot of our problems came from underestimating how big the task of making a triple-A product for multiple platforms and multiple regions really is. And other problems came from over-estimating our capacity to solve those problems using our existing procedures and staffing levels.

If there's an over-arching theme of our development, it's that we, like many other developers, believe that ultimate success in this industry comes from iteration. You have to build, evaluate (and have others evaluate) and be prepared to throw things away and rebuild.

The products we make are just too complex and our industry reinvents itself too rapidly to do anything else. But we believe that if you are truly prepared to turn a critical eye on your own product and honestly respond to that criticism you'll get quality at the end. As to whether you get a blockbuster, only time will tell.

Game Data

Developer: 2K Boston and 2K Australia

Publisher: 2K Games

Platform: Xbox 360 & PC

Release date: August 21, 2007

Development time: 3 years

Number of full time developers at peak: 93 in-house developers, 30 contractors, 8 on-site publisher testers (see the sidebar on pg. 22 for details)

Hardware: PC; AMD Athlon X2 dual core or Pentium 4 Intel-Duo dual core processors; NVidia Geforce 8800 graphics cards; Xbox 360 dev and test kits

Software: Microsoft Visual Studio 2005, Perforce, Xbox 360 SDK, Xoreax Incredibuild, Visual Assist X, Araxis Merge, BoundsChecker, Purify, VTune, 3ds Max 8, Photoshop CS2, ZBrush, Flash 8, SoundForge 8, Sony Vegas, Acid, Ableton Live

Technology: Unreal Engine, Bink, Havok, Fmod

Number of files: 3,775

Lines of native C++ code: 75,8903

Lines of script code: 187,114

Team Breakdown

In the Boston studio:

Programmer: 1

Artists And Animators: 15, plus 2 borrowed from Firaxis

Designers: 6 in-house, 1 contract

Audio Developers: 2 in-house, 7 contract

Producers: 3 in-house, 2 contract

Testers: 13 contract, plus 8 on-site publisher testers

In the Australia studio:

Programmers: 12

Artists And Animators: 10

Designers: 5

Audio Developer: 1

Producers: 2

Testers: 1 in-house, 7 contract

In the Shanghai studio:

Artists And Animators: 12

Designers: 3