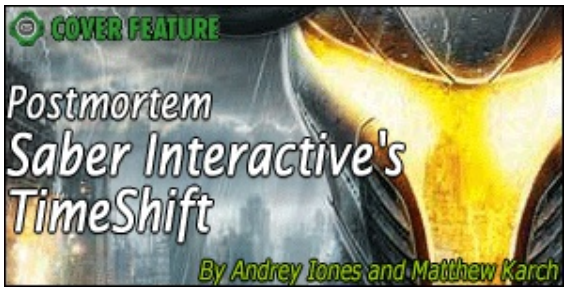
GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Saber Interactive's *TimeShift*

By Andrey Iones,Matthew Karch



[Saber Interactive's PC/console shooter TimeShift was released late last year after a "strange and convoluted" - but fascinating - development cycle, and in this exclusive Gamasutra postmortem, the creators detail what went right and wrong in its creation.]

Few games have gone through a development cycle as strange and convoluted as *TimeShift*'s. After four years of development, three extensions, two publishers, three re-recordings of voice acting, three iterations of the story, two sets of discrete FMVs and multiple iterations of Saber's technology base, the game finally made it to the shelves for Holiday 2007.

The game started off as a small proof-of-concept demo in 2003, which we completed a few months after we released our first title, *Will Rock*. Atari picked it up a year later with plans to publish the game on the PC and Xbox.

The title showed a lot of promise early on -- and visually, it was ahead of the curve -- so Atari made the decision to move over from the Xbox to the Xbox 360. Two years into development, Sierra acquired *TimeShift* from Atari. Atari was having some cash flow issues and needed to sell off some of their projects.

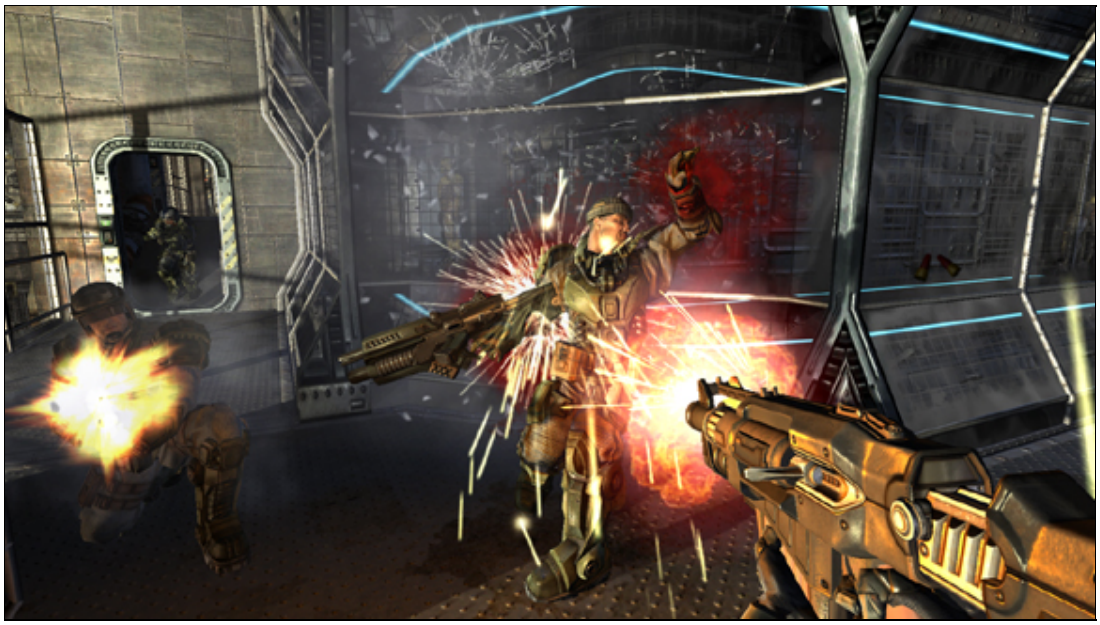
The original plan after the extension was to give the product another six months to enable Sierra to improve on some of the production elements of the game. Dennis Quaid, Michael Ironside and Nick Chinlund were hired to add their voice talents to the game and extra effort was spent in improving visual quality and gameplay elements. Literally on the eve of GMC, Sierra made the rather unorthodox decision to give *TimeShift* another year of development.

While extensions are common, they are less so when a game is essentially ready to ship. This was not only a vote of confidence in the title, but also one in Saber's ability to turn the game into a AAA franchise.

The thought behind the extension was that *TimeShift* was built on a very solid concept, but one that could use additional refinement. It was also decided to add another SKU to the game -- the PS3. This meant that Saber had about 10 months to make vast improvements to the game on all levels -- visuals, gameplay and polish -- and also to write a PS3 engine and port the game to that console. While this was a tall task, we ultimately proved up to the challenge.

In the course of a year all FMVs, voiceover acting, story, the beginning third and ending levels of the game, characters, animations, vehicles and weapons were completely thrown away and redone.

Finally, in the fall of 2007, the new and improved *TimeShift* was released to the public. While the competition has been stiff, the game has really resonated with the gaming public and user feedback has been exceptionally good.



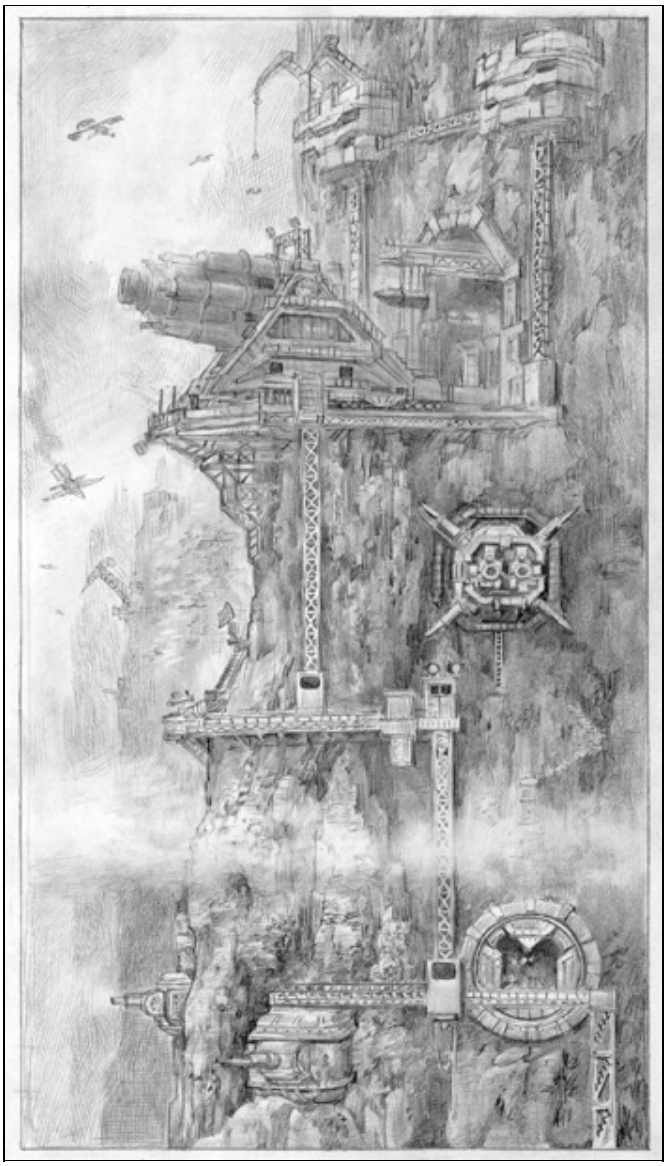
What Went Right:

1. **Dealing With the Scope of the Project.** Back in 2003, Saber had a small team of 25 talented people. It was enough to put together a working prototype which demonstrated our innovative time-control mechanics, but to handle a AAA project coming out on three SKUs, we needed to expand.

By the time we shipped the game, we expanded our studio to over 90 people, all working on the title. Undergoing such a major expansion was not easy. Talented people are hard to find anywhere, but even more so in Russia, where there are very few teams working on larger titles, and therefore there are very few people with the requisite experience to handle big projects.

Fortunately, our core team was comprised of the most talented game developers in Russia -- people with 10+ years of experience in the industry, including console work. This was the core team that worked together on our first game, *Will Rock* -- a game that was put out in less than a year from start to finish. As we kept hiring, most of the core team members were promoted to manager or senior positions, and headed up newly created departments within the company.

What was once simply the "art team", with a flat structure run by Dmitry Kholodov, transformed into a multi-discipline department that includes Concepting, Texturing, Lighting, Art Outsourcing, Asset Modeling, Level Modeling, UI Design and Special Effects -- all led by their own managers. Establishing these departments allowed us to build "centers of growth", streamlining production and communication pipelines.



When a new person was added to the team there was always a manager available who kept track of the tasks in a particular discipline and who was able to train a new hire.

While we knew it would be hard to find the right people, some other issues related to this rapid expansion were rather unexpected. Finding space for all of these new people was one of them. Because the project was ongoing, we decided to stay in the same office building rather than move to a larger office and disrupt development.

Eventually we ran out of space. For a while, we couldn't hire more people because our office was at full capacity. Fortunately, we were able to work with our landlord to relocate other tenants and take additional space on our floor.

As part of our move into next-gen territory we needed to improve our production pipelines and supporting infrastructure. During normal development, we used to make one build a month that was sent to our publisher.

That was enough to demonstrate the progress we were making. By the end of the project we were pushing 15 builds a day, every day (this includes two or three regional builds on each of three SKUs, for the full and demo versions of the game).

We had to establish a special Build Department which worked closely with our QA team to ensure that builds were made and tested daily so that the problems were immediately communicated back to the team.

We built a number of server racks with "hardware farms" to help us churn all this data on a daily basis, compiling code for all three SKUs, re-exporting art assets and assembling builds, compiling the shader database, and doing the uploads of the final assets back to the publisher. The process was working around the clock, and sometimes we were already working and testing the new builds while the ones from the day before were still uploading.

---

2. **Thinking Big.** It was in September of 2006 when Sierra made the decision to give the project an extra year. Everyone at Saber was in a state of shock from the decision. At the time, the game was nearly done, and it was looking really good.

We took it through two QA passes at Microsoft and were down to seven open bugs -- that was all that we had to do to get the game on the shelves. We were at the end of a long marathon of sleepless nights, everyone was tired, the finish line was so close... and *bam!*, here comes the extension. We all had to take a deep breath and gear up for another long year.

After having some time to reflect, we realized we were given a rare opportunity to really take the game to the next level of quality and polish, as well as to put it out on another major SKU, PlayStation 3.

Of course, as a young independent developer it is important to prove that you can deliver a game on time (which we had) and to get the game out on the shelves, but in the overall scheme of things, the ability to put the game on the PS3 and to improve the overall quality across all platforms outweighed the delay in shipping the game.

The primary question posed by the extension was what to do to stay competitive in this crowded genre a year later. We were very fortunate that we had visionary people on the team who could think big.

In the course of a year, the entire rendering engine was redone by Anton Krupkin, Denis Sladkov, and our rendering team, adding a large number of cutting-edge features such as dynamic shadow maps and a sophisticated material system. Our lead animators Alexander Myala and Sergey Boginsky traveled to LA to House of Moves and replaced thousands of hand-animations with mocapped ones.

The story was rewritten by Michael Hall -- a professional writer with Hollywood credentials. Rewriting the story obviously required redoing all the VOs (therefore the lines recorded by Nick Chinlund and Dennis Quaid were gone), FMVs and localization assets. Our HUD and menus

were redesigned from scratch.

Our multiplayer team led by Stas Zainchkovskiy completely rewrote our networking system coming up with a state-of-the-art client-server / peer-to-peer engine; Pavel Rusin (a member of Russian *Unreal II* pro team turned pro programmer) spent the year balancing and tweaking multiplayer gameplay code.

The art and scripting teams led by Dmitry Kholodov and Sergey Larionov completely redid the beginning and ending of the game, meticulously recreating the atmosphere of Krone Era (needless to say, all characters, weapons and vehicles were redone as well).

Finally, Anton Lomakin, our SFX Guru, came up with some amazing rain effects which were highly praised by the gaming press -- imagine stopping time and seeing individual rain droplets frozen in mid-air, complete with distortion and refraction effects. This is something that was never done before.

This effort paid off well. By the time we shipped, *TimeShift* did not feel like a game with a "facelift" -- it was an entirely new game. The public's reaction to the new *TimeShift* was extremely positive.

3. **Scheduling**. Sierra fully embraced the changes we suggested during the planning phase and they were very supportive of us. However, we knew that even though the game was becoming much better, we still needed to hit a Fall 2007 release date, on all three SKUs. Missing holiday sales simply wasn't an option.

A full year of extension seems like a lot of time in the planning stages. However, we quickly realized that if we wanted to ship a stable quality product we needed to hit Alpha by March 2007, and Beta by June. All said and done, we only had a few months to do a near-complete overhaul of the product.

We all felt fairly confident we could hit our dates on the Xbox 360 and PC, but whether or not the PS3 version would be on time remained a question. We got our first kits around November 2006, and we had a relatively slow start.

It took us about three months just to set up the hardware, compile the code on a new compiler, master the dev tools and render the first polygon. However, we realized that releasing *TimeShift* on PS3 in 2007 was a great opportunity, so we kept adding more senior engineering resources onto it. The challenge was that the PS3 version of the game was not a port -- all three SKUs were being developed at the same time, with new assets and technologies coming online as the PS3 engine matured. Things were moving, but moving slowly.

In early June, at a meeting in our St. Petersburg offices with Sierra's senior management, we were asked to give the probability of us hitting on time on different SKUs. The Xbox 360 and PC were estimated at 90%, but we could not give the PS3 version more than a 60% chance. We had a host of technical issues to resolve with the SKU, ranging from fitting into memory, performance, and technical requirements (TRCs).

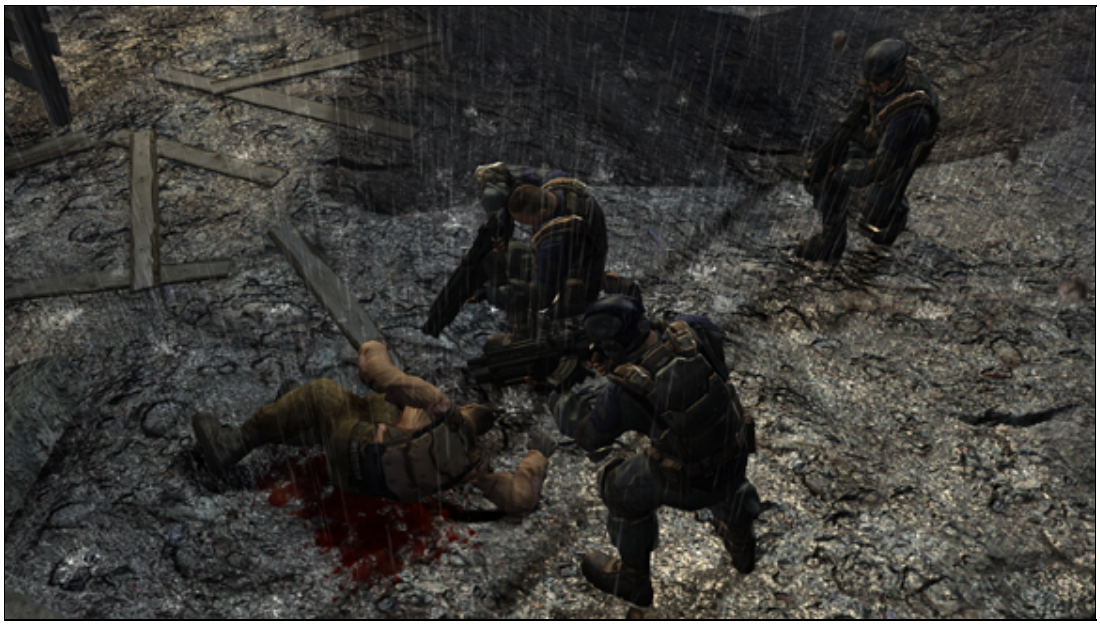
Multiplayer was also totally broken at the time, as we continued to rework the architecture. Because we were so close to the finish line, Sierra and Saber together made a tough decision to focus on the Xbox 360 and PC SKUs and make sure they didn't slip, and give PS3 version a lower priority.

It turned out to be a great decision because it provided the team with focus, which is vital on a project where the stakes are so high. However, we were not happy at all with the prospects of missing 2007 on PS3.

At an internal meeting with Andrey Grigoriev, who leads our engineering team, we brainstormed ideas on how to keep the PS3 SKU on track without putting Xbox 360 and PC at risk. We understood that we couldn't put a lot of senior resources on PS3 full-time, but it was vitally important to keep the PS3 build "alive" -- the code needed to be compiling and running, stability issues had to be in check, tons of small kinks with the build processes needed to be identified and worked out, and TRCs needed to be dealt with.

So we designated a small team of people (actually, two programmers and one QA specialist) to work full time on this SKU. When they stumbled upon issues they could not resolve themselves, other members of the team were asked for help with such specific areas as audio, multiplayer or UI.

Miraculously, this approach worked out perfectly. By the time Xbox 360 version was close to completion in early September, and we had more resources becoming available, we still had some pretty major technical problems unaddressed on the PS3 SKU. We were severely out of memory (both main and RSX), loading times were unacceptable, performance was sub-par, and a large number of TRCs were still open.



However, we had a pretty stable build which we could make and test on a daily basis just like on two other SKUs. This was the key. Our entire technical team was now ready to address all of these issues. We were at a complete asset and feature freeze. And we still had a full month before we had to make our final submission to Sony.

In addition, many of the technical hurdles on PS3 are similar to those on Xbox 360 -- and we had a lot of people who just handled that SKU, so they had the experience and the tools. Things were working out. For example, the very complex task of optimizing the loading times was handled by Roman Lebedev in under a week, reducing loading times from whopping four minutes to a TRC-compliant 25 seconds.

In addition to this strategy which was put in place and successfully executed, a few other factors played a critical role. Mark James, Sierra's Tech Director assigned to our project, helped us to forge direct relationships with the first party hardware vendors and our middleware providers.

Whenever we hit a wall (which oftentimes was happening at 3 AM Russia time -- which is an afternoon on the West Coast of the U.S.) we could call the technical experts at Sony or Microsoft directly and ask specific questions and sometimes creatively negotiate a solution which would satisfy the technical requirements and was feasible within our timeframe. This allowed us to keep the ball rolling the following day and not waste any time. Using email would certainly have slowed things down tremendously.

Finally, developing a robust set of tools paid off in a big way, especially during the final sprint towards GMC on the PS3. A great example of one of these tools is our Performance Analyzer. This relatively simple tool allows us to put performance brackets in the code and outputs performance counters into a file, which can be viewed and analyzed later on.

Entire play-throughs of the levels were recorded and dissected later on by performance experts, identifying code bottlenecks, threading issues or art problems. (Frankly, if the format of this article would allow for an extra "what went right" section I would put tools there, because having a robust toolset is second only to having a great team of people. Now that *TimeShift* is finished and is on the shelves, we are putting nearly our entire engineering team on development of the tools for our future projects).

---

4. **The Act of Balancing and Tuning.**As we approached beta, we started to show the game to focus groups and to Sierra's internal balance team. The results were pretty stunning. In the first interaction in our demo level, where we only had three opponents with the weakest weapons -- people kept dying over and over again.

Not a single member of Saber team thought this section was at all challenging. That was indicative of the major problems we had with balancing and overall accessibility of the game to the general public.

Our solution to the problem was to conduct more focus groups and refine the gameplay. We brought people in both at the Sierra and Saber offices and had our level artists and scripters watch people play and make observations. The experience was very painful. We could not believe that people could not figure the way out of a room with three doors, two of which were permanently locked.

They just kept bumping into those two locked doors forever, and eventually gave up. The solution was to clearly mark the permanently locked doors with large visible locks, or red signs. James Bonti, a Sierra AP working out of our St. Petersburg office, compiled a bible of "things frustrating the player", which included instant deaths, unusually difficult areas, clunky controls, objects which were placed in the player's path and confused people who did not know where to go, unclear objectives, and other similar things. After the fixes were made, a version was presented to the next focus group, and the cycle continued.

We also listened to the members of the press that saw the game the year before and could compare the two versions. One of the biggest criticisms we were getting was our simplified time control.

In 2006, the user could select any of the three time controls at any time, which not only took up a lot of buttons on the controller, but also made the learning curve much steeper. After the extension it was decided -- after yet a few more focus groups -- that the users learned how to use just one time control function and never experimented with the other two.

Therefore a decision was made to automate the timeshift selection -- now there was only one button to press, and the game (or rather the player's AI-controlled suit) would choose the function automatically.

This not only allowed for easier controls, but also allowed the designers to designate certain areas of the game where specific time powers were made unavailable (the biggest challenge of *TimeShift* circa 2006 was that the user could press Time Reversal at any moment, which significantly limited our ability to include complex scripted events, and affected even simple things such as opponent spawning).

However, this also frustrated a lot of people who felt that the primary feature of the game -- the ability to freely control time -- was taken away. This made us get back to the design board and figure out a solution which would combine both schemes. The end result certainly pleased the more knowledgeable members of the press, as well as the gaming public.

5. **The Proverbial Crunch Mode.** Nobody likes overtime, especially when it happens during the summer. The three months of the summer in Russia is the only time when the weather is good and it's nice to be outside.

This is the time when most people take holidays. Unfortunately, summer also happens to be the "high season" at work when the project is being wrapped up for Christmas release. Because of the extension, our team had to spend two summers in a row in the office.

Saber's senior management knew that crunch was coming, and it was inevitable. We also knew that the last thing we wanted to happen was to burn the team so much that somebody would snap in the middle of the project and quit -- so the goal was to make this crunch feel reasonable.

We also had to come to terms with a few over-zealous people at our publisher, who "expected" the team to go into crunch ideally sometime in January, knowing that the project wouldn't end until mid-October. That certainly wasn't an option. Mandatory 55-hour work weeks for most people at the team were finally established starting in May.

The primary tools we had at our disposal were proper planning and established production processes. We knew that our primary "peopleware enemy" was the frustration factor, and we tried hard to minimize it. Luckily, we learned the lessons of the previous "wrapping-up mode".

There's nothing more frustrating than to work overtime and then not be able to leave the office when your work is done because a new feature was implemented just before the build is due to go out and it needs to be tested.

Our team leads established a weekly schedule with strict cut-off times for code and data check-ins. A build for each SKU was compiled nightly and tested the following day, so if something was wrong with the code, assets, or the build tools -- we knew about the problem right away, and there were few surprises when the time came to make an "official build" for Sierra.

Every Saturday we sent a number of weekly builds on all SKUs back to Sierra QA for testing. Mondays, Tuesdays and Wednesdays were dedicated to "normal work", with features being written and bugs being fixed. Towards the end of the week, we would go into "wrap it up" more.

On Thursdays, we did the code branch and made a "build candidate" which went into thorough internal testing. Only essential bug fixes could be checked in into the "release branch", and only with the specific approval of the leads. The same applied for art changes. This way we had two or three days to test the build internally and stabilize it.

But because assets and code were fully branched, those people who did not need to do fixes for that week's build could fully focus on other things working towards other goals and maintain high productivity.

This branching system also allowed us to "break things" in a big way when needed to make major fixes or improvements without affecting the stability of the overall build. A lesson we quickly learned was that it was of utmost importance to consistently send quality builds to Sierra. Not only did it greatly increase the efficiency of testing, but also helped to cement belief in us hitting the final dates.

At the height of this "build pushing" we had 12 QA people, including Saber and Sierra team members, working on-site. These were the guys who were hit the worst with the crunch. They had to leave late, oftentimes after midnight, having tested the build, and come in early to report the bugs and be available to team's requests.

Kirill Morozov, the head of Saber QA and our "Build Master", setup remote access from home and was monitoring and tweaking build assembly processes through the night, sometimes getting frantic calls from QA guys in LA at 5 AM when something was going wrong.

We also did a few other things right when managing this unpleasant crunch period, ranging from proper financial incentives to serving lunches and beer on Saturdays. The team morale was holding up pretty high, and we achieved our goal -- nobody quit -- either during the crunch, nor after it ended.

What Went Wrong:

1. **Prolonged Dev Cycle.** As we mentioned above, *TimeShift* was in development for more than four years. We went through a publisher switch. Even at that moment, the game was somewhere around beta and was already undergoing serious QA at Atari, and a demo was already out on PC and Xbox 360; a year later we had to clear up the confusion when people were downloading the OLD demo and thinking they are playing the new one.

A total of four executive producers, two writers, and a large number of creative people worked on the game. By the time we finished, we were operating under Amendment 17 of our initial contract. An insane number of piecemeal two to four month extensions never allowed us to neither bring the game to the finish line, nor make a big enough overhaul. The game as it was in September 2006 was a mishmash of way too many creative visions.

Certainly the last extension was an exception to this and allowed us to bring the game to the level of quality it deserved, piecing the elements together. It worked out for the most part, but the story came out relatively weak.

We were constrained by the fact that we could not really take the game through a proper preproduction and planning phase, and could not change a lot of levels to accommodate to the final creative vision. Certain members of the gaming press criticized us sharply for that.

Another negative effect of the game being so long in production, and being close to the finish line so many times, was that by the time it was getting close to being released it had already been shown to the press over and over again.

How many exciting stories can you tell to the same journalists who've seen the title three or four times already? The surprise factor was totally lost, and Sierra's PR staff had to overcome a lot of skepticism. Because the game was shown primarily to the U.S. press, PR efforts were easier to carry out in Europe where the game was fresher. That's probably one of the reasons we are getting consistently higher review scores in that region.

2. **"Not Invented Here" Factor.** At Saber, we strongly believe that it's absolutely essential to rely on our own technology. *TimeShift* is built using our proprietary Saber3D engine and toolset. Having full control over the entire production pipeline has allowed us to accommodate new features easily and to optimize our production processes.

We fully believe that we would be unable to implement one of the most challenging technical features of the game -- time reversal -- had we not had built the game on our own engine. Having said that, we realize that there's a great value in using engine components provided by third parties. For *TimeShift*, the primary components included GameSpy for networking needs, Fmod for audio support, Bink for video playback, and Havok for physics handling.

Adopting these systems was a great idea which allowed us to focus on the game rather than on pure tech. All of them are tested solutions, utilized on a large number of projects in the industry. We therefore expected them to be as reliable and robust as the CLIB base library. Unfortunately, that was not the case.

Sometimes the issues were known and there were pre-existing workarounds or fixes available, but every once in a while we were stumbling upon an untested piece of code, or an unanticipated use of a subsystem, or a lack of a vital piece of documentation which made some concepts unclear to our engineers.

What came as a surprise, though, was that whenever an issue arose with any of these components it oftentimes required massive efforts to nail it down. Conference calls were scheduled, mini-apps that demonstrated the problem in a testbed scenario were built and sent to the providers, and support trips were set up to bring the engineers on-site to address the issues "right on the spot".

All of that slowed down the development somewhat and was always a "scare factor" to the senior management. What if an engineer from one

of these middleware providers would not be available for a quick trip to our offices?

At the end of the day, though, we were able to address all of the issues and ship on time. Many kudos go to all of those middleware providers who helped us by going that extra mile, flying in from all over the place to St. Petersburg, or making themselves available at odd hours to talk to our team.

The lesson learned is that it's certainly a great idea to utilize third-party solutions, but because those providers don't work for you they may have other priorities, so enough time needs to be allocated to deal with the possible issues. It's also crucial to integrate and test all third-party components as early in the dev cycle as possible to be able to accommodate to any contingencies.

And it's essential to get the source code for all the components so your engineers can at least see exactly what's happening at the source code level and report specific info such as callstacks, the values of registers or variables back to the middleware support team. Establishing direct relationships and lines of communication with tech support people is invaluable, with real-time communication means such as ICQ or other IM services being preferable to slower email or web-based ticketing systems.

---

3. **Thinking Big -- The Flipside.** Like we mentioned above, it was critical for us to think big and to try to advance every aspect of the game in order to stay competitive and come out with a AAA game in 2007.

This was the only viable strategy -- but it had a flipside. We had minimal time to spend on preproduction and feature testing in the early stages. We had to be working on assets at the same time as the technology to support those assets came online. Obviously, we were stumbling upon unforeseen issues all the time.

One of the biggest technical challenges we had was with our ShaderCache system (see Wrong #4) -- but that's not the only example. ShadowMap technology only became functional at the time we were at Alpha, so Olga Cheremisova and our other lighting artists had a very limited time to go through all the levels and set up proper lighting.

Sometimes we did not have the time to properly upgrade the tools to accommodate the volume of new assets. Our lightmapping tool was developed two years ago, but it was never tuned to handle the very large scenes that we were coming up with.

In the worst cases, it was taking up to 45 minutes to calculate lighting for some of the larger scenes, which is obviously totally unacceptable and made the life of our lighting artists even tougher. The only viable option we had was to continually upgrade hardware and give multiple computers to the key lighting artists.

Another issue we faced was production speed due to constantly evolving technology. Although our engine was highly optimized in time for ship, we were not able optimize the engine fully during development, which affected the work of artists and scripters who unfortunately had to struggle with the game running at sub-optimal framerates.

It was only at the end of production that we had time to get the engine running fast. Had we been able to do this earlier, the design department would have had an easier and more efficient time scripting our levels.

Finally, because the final dev schedule was so compressed, most assets were coming online late -- even though everybody involved was pushing as hard as possible, the sheer volume of new assets was mind-boggling. This included delivery of mocap, VOs, localization data, FMVs, and other components.

It was our first attempt to use mocap, and probably because of that, it took us over a month to get the first mocapped animation in the game and nail down the pipeline (of course, when we finally got that first animation in we lost Ruslan Vizgalin, our lead animation programmer, for three weeks -- he was in the hospital fighting pneumonia).

This whole process was wreaking havoc on our lead AI programmer Sergey Mironov, the scripting team and the producers -- most animations and animation transitions were not working for about a month, the game was not playable, and it was impossible to easily tell what the source of a bug was -- was it one of the many animation glitches, or a scripting issue, or an AI navigation problem?

It was not much easier with VOs and localization assets. VOs were recorded and localized at the same time, even before integration of English-language VOs in the game was complete. Anna Naleushkina, Saber's AP responsible for those production tracks, was frantically coordinating from our St. Petersburg office the movement of those tens of thousands of assets between the recording studio in LA, the processing house in Vancouver, and localization team in Dublin. Surprisingly, nothing was "lost in translation".

To be honest, we don't know if there's a good solution to handle this -- other than add more time and make the project more sane. That clearly wasn't an option for our team, though. When you have such a massive number of assets coming in and you need to do the integration very quickly, it simply creates a project-wide havoc, and it feels as if the skies are falling.

There was never enough time to fully update the tools or pipelines, so all we could do was to clench our teeth and plow forward. A good enough number of Saturday night beers helped us succeed.

---

4. **ShaderCache.** For *TimeShift*, our shader guru Denis Sladkov developed a state of the art material and lighting system (one of the first fully supporting dynamic shadow maps). Its design started right at the beginning of last year's extension, and it was amazing.

For the first time we managed to bridge the gap between the shader programmers and technical artists. Evgeniy Davydenko, our most tech-savvy artist, was excited to be able to tweak shaders in realtime and see surface properties change in the engine.

As the engine side of the technology was slowly becoming available, the artists had to start working on textures and materials right away, using our previous-gen tools. The entire texture set for the game and supporting materials had to be reconverted many times to keep up with the changes the tech team was doing.

However when the system was finally ready and the supporting shaders were written another major problem arose. We could not play the game at reasonable framerates because we did not have a shader-precompilation system.

Because shaders were compiling at run-time, the game was chugging every other frame at any decent visual quality setting (it takes up to 30 seconds to compile an average shader in *TimeShift* on PC, and we had thousands of those shaders). In other words, all we could do was to switch to shaders 2.0, position the camera, then switch to shaders 3.0 -- and then after a five minute pause a beautiful picture appeared.

The problem was massive. We assigned Victor Streltsov, one of our most seasoned and creative programmers, to address the issue. We hired another senior programmer, dedicated to helping Victor with that assignment. Four months later, our ShaderCache tool was born.

Think of a huge distributed computation system which is installed on all 100+ workstations we have in the office, governed by a server-based app which farms out individual shader compile tasks to the clients. Even with a system like this it took us over 12 hours to build a complete shader database for all platforms (by the time we finished, it took no more than an hour to complete this task, thanks a number of optimizations we did).

While this system was being built, however, the life of texture artists was pretty miserable. They could work with test scenes just fine, but seeing the results of their work in the full scenes was nearly impossible.

Unfortunately, even once we were able to pre-compile all the shaders for our scenes, another pretty big issue emerged. We had a few thousand unique shaders per scene, and because they were very complex -- it took over five minutes to create them all during level loading on PC.

Obviously, having the user wait five minutes for a level load was unacceptable. Also, for some hardware configurations the game was crashing while creating all those shaders.

Our lead rendering engine designer Anton Krupkin was working out a solution with NVIDIA and ATI engineers. It was another external dependency which we identified pretty late in the process. Another set of custom builds had to be made and delivered via secure channels to the hardware vendors.

Some of the problems that we saw could never be reproduced on their end because we used some specific combination of graphics cards and the drivers -- so we brought their engineers to our New Jersey office. Ultimately, thanks to the unparalleled support of NVIDIA and ATI staff, the issue was resolved and driver updates were issued on time for the release of the game.

---

5. **Testing the Networking Code.** Even though we nearly shipped a game with a decent networking code a year ago and had an understanding of what it takes to get it done -- we still underestimated the complexity of the task. We set a high quality bar, which was to create a robust multiplayer engine rivaling those in the best products -- and we did not want to settle on anything less than that.

Our multiplayer mechanics were very strong and fun to play, so we had to support them with proper tech to allow 16 players to play online. In

order to reach the mass-market we established pretty low bandwidth requirements, and to achieve this goal we had to do a pretty major re-architecturing of the networking engine.

Once the commitment was made and the work started -- there was no going back. Our old networking code was no longer compatible with the rest of the engine, but the new system was insanely complex and needed to be properly debugged.

Normally, you want to stay away from "insanely complex" systems because they are hard to develop, debug and maintain. But it looks like it's the nature of the business when it comes to multiplayer. All the issues we had in single player were now multiplied by a factor of 16 (the maximum number of players *TimeShift* supports).

You want to test stability? Performance? Bandwidth handling? Great! All you need to do is to find 16 people that are available at request, ideally sitting in different offices (St. Petersburg, NJ and LA), make sure they all have the same build, and do the test. Then repeat as needed.

The amount of testing we had to go through was absolutely insane. We came up with a concept of mini-builds, which allowed us to distribute builds faster. We ran dozens of tests on a daily basis assembling bandwidth handling stats, trying to provide the multiplayer team with enough information to do the analysis.

A dedicated multiplayer QA team in LA was running daily tests with their counterparts in India, running multiple 16-player matches with different settings on various bandwidths. But a week before release we were still unhappy with the results -- even though our lead multiplayer engineer, Stas, was leaving the office at 5 AM nearly every day of the week. It still remains a mystery how he pulled it off -- but he did.

All the crashes were addressed, bandwidth handling streamlined, and we shipped with very solid code which allowed people from various parts of the world to play on a server with minimal lag and use the voice over IP feature.

Even though we finally succeeded, this entire experience was scary, and worked totally against our normal planning. We are always trying to implement "big systems" first to minimize the risks and bring the game to a shippable quality long before the finishing time, which needs to be reserved for minor tweaking and polish.

That was certainly not the case with multiplayer. A lesson we learned was that we underestimated the massive scope of testing required to bring the multiplayer component to the quality level we wanted. Also, multiplayer coming together so late in the process did not allow us to have any beta testing of this component of the game.

Surprisingly, though, based on the user feedback, multiplayer implementation is very solid in *TimeShift*. In fact many consider it to be better than the more established shooters that released this season.

Conclusion

Working exceptionally long hours, dealing with unexpected technical hurdles, managing multiple SKUs, and adapting to ever-changing business requirements, working on *TimeShift* was a very trying experience for everyone involved, both at Saber and at Sierra.

We could have never anticipated what it would take to put together a AAA product under such tight deadlines, but we were determined to make it to the finish line and to create a great game. We emerged from this experience as a much more mature team ready to take on even bigger challenges.

TimeShift is Saber's first console game, our first AAA title and probably the highest profile game ever to come out of the former Soviet Union this side of *Tetris*. We are very proud of what we accomplished and look forward to a normal development cycle where we can really show off our talents as a team

Game Data

Developer: Saber Interactive

Publisher: Sierra

Platforms: PC, Xbox 360, PlayStation 3

Release date: Fall 2007

Development Time: 4 years

Number of full-time developers at peak: 92

Number of contractors at peak: over 50

Hardware: over 120 hi-end PCs with latest graphics hardware, including various "farm racks"; about 30 Xbox 360 dev and test kits; about 25

PS3 dev and test kits

Software: in addition to standard dev tools, we used SN DBS (making distributed builds on PS3), gcc 4.1.1 (PS3 compiler), SN Tuner (PS3 performance analysis), ProDG, NvPerfHUD, NvShaderPerf, ATIGpuPerfStudio (graphics performance analysis on PC), VTune Performance Analyzer and Thread Profiler, PIX & XbPerf (performance analysis on Xbox360)

Middleware technology: GameSpy, Fmod, Bink, Havok

Return to the full version of this article