Postmortem: Bohemia Interactive Studios' Operation Flashpoint

By Marek Spanel,Ondrej Spanel

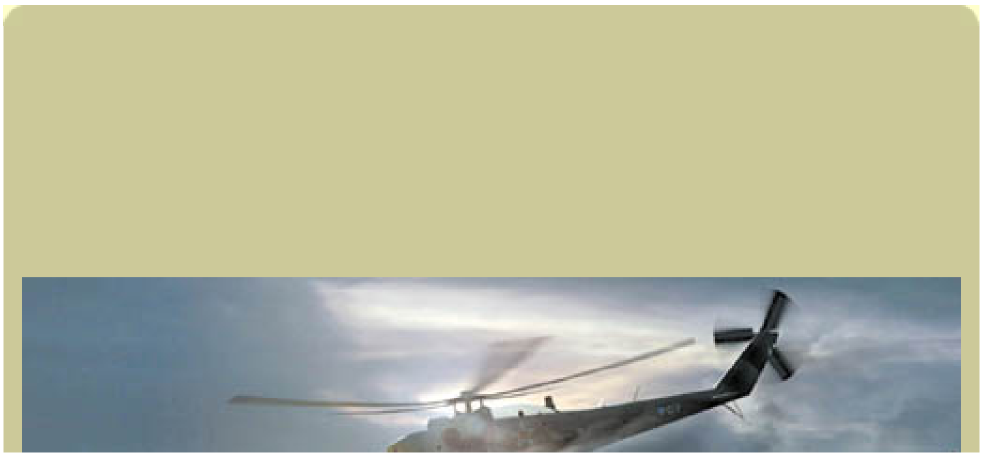
The story of *Operation Flashpoint's* development is quite unusual in the game industry these days. For one thing, the team didn't start out as professionals; originally only the lead programmer was allowed to work on the game full-time. Switching publishers three times, starting a new company, growing the team from one to 12 full-time members, and moving offices five times during the game's development were just some of the hurdles we had to clear. Only the team's vision and obsession for the game remained consistent from the very first playable version until the end. It's not possible to describe whole story in the space given for this article, so let's just jump directly to the final moments.

It was 8 p.m. on Friday, May 25, 2001. Our publisher's representative, who had been in Prague for the last few days to make sure everything was going O.K. as we were finalizing the gold master, left Prague feeling confident that things were going well — the disc was almost ready and could be sent to final testing and then to manufacturing after some weekend testing.

Meanwhile, our lead programmer (to make matters even more exciting, he was then working at his temporary home in France for couple of weeks) was trying to resolve some serious graphical anomalies with the hardware transformation and lighting (HW T&L) rendering. If he were to fail, HW T&L would not be included in the final release. If he solved it, some data organization changes would be necessary to suit the needs of the HW T&L. He spent nearly the whole day resolving some random crashes that appeared in the game during the last day, going back and forth over e-mail with an Nvidia support engineer. The crash was fixed by late afternoon, and by 10 p.m. it looked like the HW T&L problems were at an acceptable level. Around midnight, the tools that would perform the data format change were ready.

On the other front, the team had received the final localized strings for the game. However, the file containing the core strings of the game that had been delivered by our publisher appeared to be untested and unusable. After spending a couple of hours dealing with it, most of the team had to go home to have some sleep. Still, the team leader stayed behind at the office, trying to use the new HW T&L data format, going over each step by phone or e-mail with the lead programmer (while also trying to implement new localized string tables and fix some problems in the campaign and missions). At 3 a.m. it looked like all the data had been converted — and both the lead programmer and the team leader could go have some sleep.

Saturday morning, our publisher realized that the gold master hadn't actually been delivered. Tensions rose even further, and nerves began to unravel. Only two days remained before mass production was scheduled to begin. Everyone on the team had been working since early Saturday morning, but at times a successful end to these last-minute crises seemed to be so far away. By around 5 p.m. on Saturday, most of the important issues in the code had been resolved, and the lead programmer decided to take another look at the HW T&L implementation. Luckily, within a few hours, he suddenly discovered the root of all of the HW T&L problems and fixed them. The plan was to deliver the gold master to our publisher via FTP by that evening. Nobody expected that it would actually take until Sunday morning. After a long, sleepless night of playing through the game and fixing any problems that appeared, everything looked fine, and most of the team could finally go to sleep again.



***Operation Flashpoint*'s mission was to deliver the tension of full-scale conventional military conflict.**

With some relief, we finally started the game upload on Sunday around 9H But were we done? Not yet. Suddenly, a seagull stopped flying in some of the in-game cutscenes. The team leader called to wake up the lead programmer in France: "The seagull is not flying. What should I do?" We had to stop the upload until the lead programmer delivered necessary code fix. After the project leader received the updated files from the lead programmer, he started to rebuild the game in Visual Studio. It was Sunday around noon, and the game had finally gone to the publisher for final testing.

The publisher's test staff started playing the game Sunday afternoon. Everything went smoothly at first, but later they discovered one serious scripting bug in one of the campaign missions that made it unplayable. Late in the evening, they called the team leader about the bug, and he had to drive to the office after sleeping just a couple of hours over the past three days to fix the bug as quickly as possible and then upload the fixed version to the publisher's server in the U.K. Around midnight Sunday night, the disc was finally ready to go.

Three weeks later, hundreds of thousands of copies of the game were available in stores worldwide. In the meantime, the development team was playing the game, terrified of finding a disastrous bug. Fortunately, no such critical bug appeared. Considering the amount of work we'd done on the game in those last couple of days and hours, the risk of finding some major problems was pretty high. On Friday, June 22, the game was released, and it immediately became the top-selling PC game in many countries. The team knew that their mission was successfully completed. The passion and hard work of every single member of the development and publishing teams started to pay off.

What Went Right

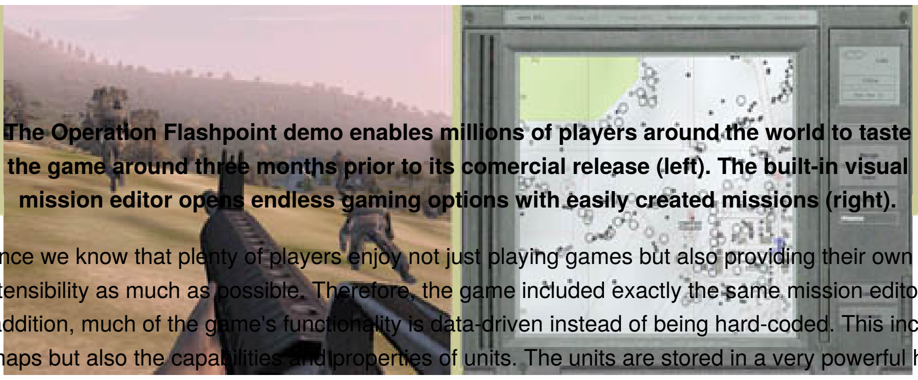
1.The team. Probably the most positive thing we encountered during development of this game was the people working on it. Almost all of the people who joined the team really helped to improve the game and remained fully dedicated to it from the first day until the final moments. While we were understaffed almost all the time, we are happy to say that while the team was growing steadily, it was very stable — almost all the people who participated in the development of *Operation Flashpoint* are still working at our studios now that the game is finished. Another advantage was that the team was very cooperative. Some roles on the team were not demarcated very strictly. Still, between the programmers, designers, and artists, everyone could comment on any part of the game, and everyone's opinions were taken seriously.

While this often made communication more difficult, it definitely helped the design and development process and enriched the game in many areas. One of the most powerful tools we used for internal communication was our intranet news server, which proved to be an invaluable tool during the whole design process. The game was mostly designed on the fly, and the newsgroups made the design process not only convenient, but really quite enjoyable.

2.The community. *Operation Flashpoint*'s public following is incredible. We ourselves are surprised by the number of people creating fan sites, developing new content for the game, or just keeping in touch with the community by reading news and forums. Currently there are hundreds of different sites dedicated to Flashpoint, and dozens of them are of a truly professional quality with news updated almost hourly.

Another great thing about the community is that some of the people have been following this game for years already, and they still carry a great deal of enthusiasm for it. Some of the first great fan sites started out more than two years before the game was released, and we managed to keep people's excitement going with regular updates about the improvements we were making to the game throughout its development. We take it as a good sign that most of the sites are still online and updated regularly. About halfway through development, we invited some people from the community to participate in the design of the game directly, via external forums and newsgroups. Their skills in both military and gaming areas were invaluable, and we constantly used their feedback to improve the game.

Since the release of the public demo version (around three months prior to the release of the final game), the community following has become much bigger. The only downside to the increase in the community base is that the community has become much less focused and less mature, as the average age of those visiting the web sites has descended notably.
But the community still looks really vital and the most-visited fan site has counted millions of page views already. Recently, fans have been creating many custom-made tools and enhancements to the game in addition to the various web sites and services.


The Operation Flashpoint demo enables millions of players around the world to taste the game around three months prior to its comercial release (left). The built-in visual mission editor opens endless gaming options with easily created missions (right).

3.Open architecture. Since we know that plenty of players enjoy not just playing games but also providing their own content for them, we wanted to enable this extensibility as much as possible. Therefore, the game included exactly the same mission editor that we had used to design our missions. In addition, much of the game's functionality is data-driven instead of being hard-coded. This includes not only the mission files and world maps but also the capabilities andproperties of units. The units are stored in a very powerful hierarchical configuration tree with inheritance capability, yet they are relatively easy to edit. By using these configuration files, it is possible to add completely new units and worlds to the game or add modified versions of existing ones, which is what many players are doing to create their own content, thus lengthening the product's lifespan. We wanted to use configuration files to shorten coding time as well, because we figured that the fine-tuning of most values could be done by designers or testers instead of programmers. But this didn't work as we expected, mostly because only the programmers really knew meaning of the values.

Our scripting language also featured highly in the game's development and extensibility. We started to build the mission editor as a visual tool, but we soon recognized its limitations in certain areas. Seeing this, we added an expression evaluator for trigger activation, which was surprisingly powerful, and a full scripting language soon followed. When the game was released, we knew immediately that scripting was a really good choice, as many user-made mission used scripts to implement specific new functionality. Looking back, we can say scripting proved to be much more powerful than we expected. We only wish we had added it sooner in the development cycle, so that some of the functionality that we hard-coded into the game could have been scripted instead, including some AI behavior.

At a later stage of development (after the European release, in fact), we extended the game's ability to support add-on content. Single files stored in specific folders could add, for instance, new models, units, or islands. Besides some official add-ons that we have introduced since the game's release, there is already a massive number of user-made add-ons, all available for free on the Internet.

***Operation Flashpoint* enables the player to use any vehicles, including gunships and planes (left). Daytime and weather changes dynamically in the game so the are looks pretty different in various daytime and weather conditions (right).**

4. Creative freedom. From the very beginning of the project, we tried to create the game that we really wanted to play. We didn't look too much toward other games for inspiration, and we virtually ignored games that might be considered our competition. We also gave little regard to whether the market would like the game or not. Our relationships with publishers were never too strong (actually, we changed publishers several times), and all important decisions were made by the core development team, keeping us relatively independent throughout the game's development. In fact, changing publishers so many times wasn't strictly a negative thing. Even though it led to some financial uncertainties, the creative freedom we enjoyed instead of some more money was more than worth it.

The result of this creative freedom is a game that is really distinct. Our design-on-the-fly approach (or "design by playing," as we prefer to think of it) made the game very enjoyable and a different experience from any other. Most design decisions were first discussed (mostly in newsgroups as mentioned earlier) and then tried out in the game. No matter how nice a design idea might have seemed at first, only the elements that worked well in the game ended up being included.

5. Long development cycle. The unusually long time that Flashpoint was in development was very beneficial. In terms of gameplay, the game is very mature, which would not have been possible in a shorter development cycle, especially because this was our first major game. Two or three years into development, the game started to be really enjoyable. In fact, it was polished enough then to suit our original plans for the release version. But due to various things (mainly on the publishing side), the game wasn't released at that point, which gave us more time to polish and improve it. We were able to incorporate various features that we originally hadn't planned to develop, either because they were too difficult or required excessive CPU or memory resources.

We were also able to incorporate feedback from various external testers — our friends as well as colleagues at well-known gaming companies who evaluated the game throughout its development. We refocused and redesigned the game a couple of times, always opting to keep everything that worked well and change the things we felt could work better.

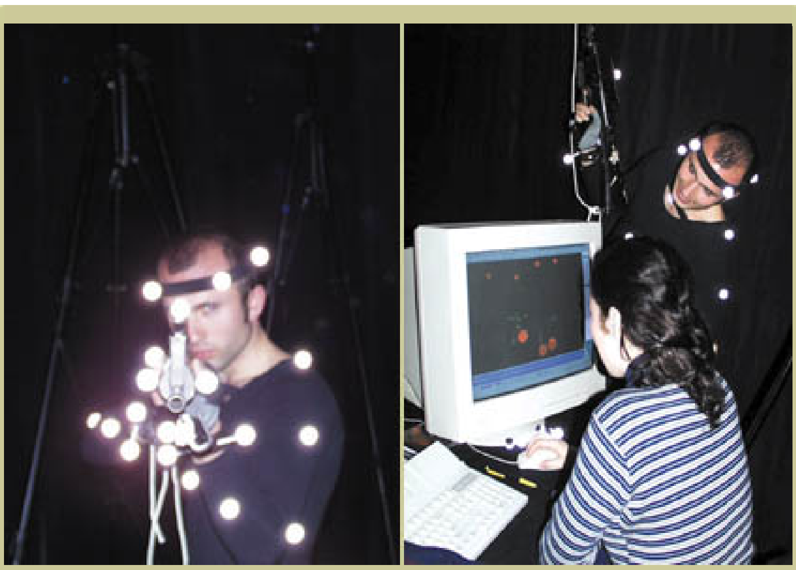
What Went Wrong

1.Development cycle much longer than expected. Ironic as it is, we have to start What Went Wrong exactly where we left off with What Went Right. Despite some of the usefulness the extra development time offered us, in the end th cycle was probably too long, and in optimal conditions would have been at least 20 percent shorter. It may have been possible to shorten the cycle, but we experienced various external events or internal missteps that prevented us from accomplishing that.

First of all, some technologies in the game were a bit outdated after more than four years. We didn't know at the outset that the game would be still in development after so long, and we hadn't left time at the end to rework parts of the engine. Some of the criticism of *Operation Flashpoint* addresses the amount of detail in the textures and some models — and we have to admit that these could have been better.

Furthermore, the excessively long development cycle led to burnout and heavy exhaustion, particularly for the people who had been working on the game since very beginning (the lead programmer, lead artist Jan Hovora, and the team leader). In some cases, we weren't able to sustain some of the features we'd implemented two or more years prior. They just disappeared somehow in the process of reworking parts of the code, and we didn't even notice. Newcomers to the development and testing process never knew such features had ever existed.

The main problem wasn't that the development cycle was too long per se, but that the development was so much longer than we'd expected. Next time, we will work much more diligently to better estimate our development time — and we will probably try to aim higher with the detail of our artwork, even if it seems insanely detailed for present and predicted hardware capabilities.


Bohemia Interactive's in-house motion capture facility has proven very beneficial in creation of lifelike human movement animations.

2.Documentation. Lack of documentation is a common affliction among game developers, but some aspects of this problem were so severe in our case that they are worth mentioning.
While we'd never believed too much in designing the game on paper, the real problem was that we never even had documentation of the things that we'd finished. This situation led to incredible problems in the final stages of development. Many tasks could only be done by one person on the whole team. In other cases, hours were spent trying to investigate how something had originally been meant to work. We recognized these problems and tried to improve them, but apart from a few instances, our effort wasn't really successful.

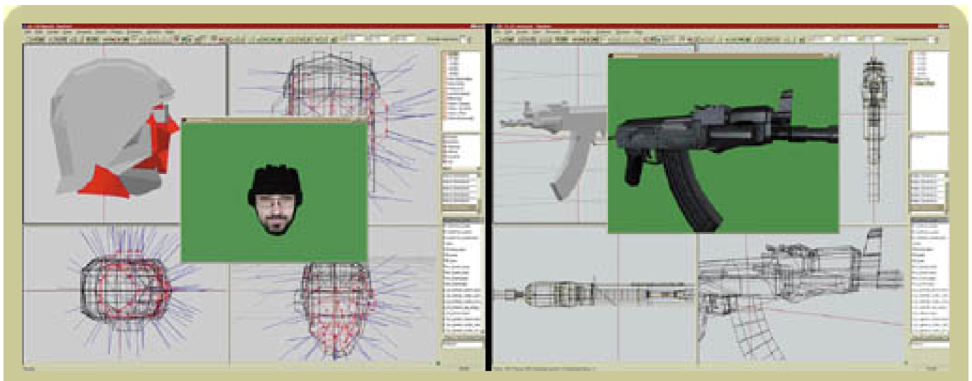
As the development team grew, the missing documentation was becoming a more serious problem. But the final project deadlines were getting closer as well, so it was nearly impossible to find the time to address the problem.

3. Quality assurance. We experienced various problems in communication and cooperation with our publisher. Generally, their focus and assistance in some areas of the production of the game (design suggestions, voice-overs, translation, scriptwriting) were really helpful. But in other areas, we experienced some events and circumstances that slowed down and complicated the game's development rather than moving things forward.

One of the most unsatisfactory areas was the way the QA procedures were managed and designed to work by the publisher. We never succeeded in achieving a common bug database, and the publisher enjoyed an illusory feeling that its QA database really covered the project. The truth was that such a database (even without any direct access for the development team) hardly said anything about the project's status because it covered just small fraction of all the problems we had tried to fix. Even when the publisher dedicated a pretty big testing team to the game, it sometimes seemed something of a waste of time for everyone involved in it.

In the end, we had to largely ignore the publisher's QA reports, because they contained too much useless information and very few real bugs. We tried to focus on very limited external testing managed directly in the very late stages of development to ameliorate this problem — but this approach could hardly replace real, full-time testing of the game.

We admit the situation was very difficult for the QA team as well — in no small part because of the lack of documentation on our side — but we still believe this process could have been handled much better. One of the mistakes we made was assuming that the publisher's QA would be sufficient, so we didn't build a strong testing team in-house. We definitely will find a solution for any future projects, because the way it was done for *Operation Flashpoint* wasn't satisfactory.



The Head of a tank crew in Oxygen. You can see face of Petr Pechar, one of the artists on *Operation Flashpoint*, under the helmet (left). A Soviet AKSU rifle shown in Bohemia Interactive's proprietary modeling tool, Oxygen, with direct real-time previewing using the game's engine (right).

4. Some content was not under our full control. One very concerning thing was that our final CD was still manipulated by the publisher. The publisher applied SafeDisc protection to the final code, which caused some unexpected compatibility problems that we weren't able to control. The mixing of various SafeDisc versions and a serious compatibility problem with Windows 2000 that was present in the first European batch of CDs could have been avoided.

In addition, we weren't able to finalize the English language in the game because we didn't have a native English writer on the team. We also didn't oversee the voice recording and voice actor selection, which led to some results that were unsatisfactory to us.

5. Multiplayer API. From the very beginning we were aware that our game had huge multiplayer potential, especially if it were implemented as a massively multiplayer online game. But we knew we would be unable to deliver such experience. Instead of aiming for such an unrealistically high goal, we decided to implement mission-based multiplayer that shared as much code as possible with single-player game. Even this effort proved to be extremely difficult. *Operation Flashpoint* was always developed primarily as single-player game with a strong story, but for a very long time we were thinking about multiplayer functionality, and we tried to design the game code in such a way that it would help us incorporate multiplayer later.

For implementation, we wanted to avoid low-level network coding and instead use a high-level API. As we were already using Direct3D for graphics and DirectSound for audio, and both suited our needs quite well, we decided to use DirectPlay as our network API. DirectPlay offered high-level handling of all network communication, including Voice Over Net capabilities. Unfortunately, our experience with this API was extremely bad. Often when trying to get some high-level functionality working, we realized it contained bugs that rendered it almost unusable.

We had to implement our custom code for things we thought DirectPlay would provide, but that was sometimes very hard, as we did not have the low-level control that we needed. We also encountered many performance problems, some very strange, such as significant (particularly server-side) slowdown even with no traffic over the network. This along with the lack of documentation and a lack of stability resulted in many problems that were hard to debug.

Another drawback that we didn't recognize beforehand is that DirectPlay is Windows-only, but many dedicated servers for games currently being played online run on Linux. Overall, selecting DirectPlay as our network API was one of the most unfortunate decisions in the whole game's development.

Future Dreaming

Most start-up game developers dream of developing a number-one title. We weren't any different. With *Operation Flashpoint*, this dream has come true for us. The game achieved the number-one position in sales charts of various countries and regions, including the U.S., Germany, the United Kingdom, Benelux, Scandinavia, and Australia. More than 500,000 copies sold worldwide in just three months, proving to us that our last four years of effort were worth something.

We always stayed focused on the game, and we didn't have too much regard for those who believe that the success of any game is mainly a question of marketing, securing a big license, or working on a sequel. We always believed that it's the game itself that makes the difference between success and failure.

The combination of infantry with full simulation of vehicles is a crucial part of the design of *Operation Flashpoint.*

We're still playing the game and we still like it. After such a long time, it's hard to believe that *Operation Flashpoint* remains the favorite choice of games for most of the development team. We're still working on new content for Flashpoint out of pure enthusiasm. In the end, we don't feel ourselves to be anything more than proud members of a big and healthy Flashpoint fan community that has arisen around the world.

We know that someday we will have to leave Flashpoint to its own destiny. But currently, we still feel too involved in the game. We are already looking forward to future projects, but it will take months for us to start one. We consider the success of *Operation Flashpoint* as the pole position for our next race. We plan to use all the experience and resources that we have gained during the last couple of years to push gaming even further in the future.

Game Data

Operation Flashpoint

Game Data Publisher: Codemasters

Number of Full-Time Developers: 10

Number of Contractors: 3

Estimated Budget: $600,000

Length of Development: Over 4 years

Release Date: June 22 (worldwide except North America), August 29 (North America), 2001

Development Hardware (Average): Various PC systems from 266MHz Pentium IIs to 1GHz Pentium 4s and 1.2GHz Athlons with 20GB hard drives and Voodoo 2 or GeForce 2 graphics cards

Development Software: Windows 98/2000, Linux servers, Visual C++ 6, SourceSafe, Adobe Photoshop 5.0, 3DS Max, Microsoft Office, TextAloud (for voice prototyping)

Proprietary Software: Oxygen (3D low-polygon modeling and texturing tool), Visitor (landscape editor), and some other proprietary data conversion and packing tools

Notable Technologies: DirectX, Vorbis Ogg, Vicon 8 motion capture system

Project Size: 10,000+ files, 250,000 lines of C++ (some assembly), 5,000 textures, 800 3D models, 100,000 words (localized into six other languages), more than 60 single-player and multiplayer missions

---