GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: *Tom Clancy's Splinter Cell*

By Wu Dong Hao

I started my career in game industry with Ubi Soft's Shanghai studio in 1998 as a producer. During my five-year adventure with Ubi Soft, I've worked on a number of different games, but *Splinter Cell* for the PS2 was the most challenging one. The technical breakthroughs we needed to achieve and the tight project schedule we faced were almost overwhelming. When we set the schedule in November 2002, almost no one believed we would make it. To add even more pressure, the success of the Xbox port of the game made everyone realize that there was no way to make just average-quality PS2 port of *Splinter Cell*. We had to match that quality.

Sam Fisher climbing in *Splinter Cell* for the PS2

Fortunately, we achieved satisfactory results. The port of *Splinter Cell* to the PS2 certainly wasn't an ideal project (especially in terms of cost effectiveness), but it's a good example of applying extra resources to hit milestones.

The project was your typical porting job. With the success of *Splinter Cell* for the Xbox, we had a solid starting point and a very clear vision of what we had to achieve. Thus the challenges were technical and schedule-related, and weren't related to content creation.

As producer on the project, I mainly focused primarily on planning, personnel management, risk management, team coaching, and establishing good working processes. In this postmortem, I'll share my experience as it relates to these aspects of the project:

- Prototyping and pre-production
- Personnel management on a huge and multi-cultured development team
- Planning and production management
- Quality assurance

We started working on the port of *Splinter Cell* in April 2002, with a small team that focused on prototyping and technical research. The full production began from the second week of October. Between that time and February 2003, when Sony Computer Entertainment Europe (SCEE) approved the first master, the team grew to around 90 people. People from other Ubi Soft studios joined the project at different phases, including some developers of the original Xbox version from Canada. We also had developers from France and Italy.

What Went Right

1. A prototype that helped determine resources and scheduling. Compared to the full production team, our prototype team was tiny. We started with five people, comprising two engineers, one level designer, a 3D artist and myself. We added four more engineers and an animator two months later.

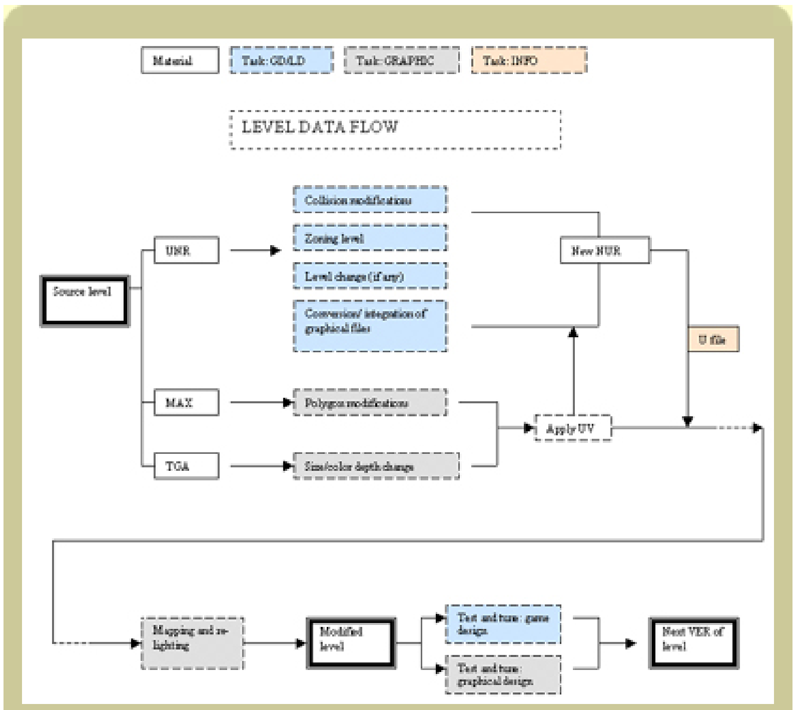
Besides proving to upper management that we are able to solve the technical issues of porting, the prototype we first worked on was supposed to provide a solid base for production. By August 2002, we were quite confident of the outcome: The systems for generating

Splinter Cell's lighting and shadows on PS2 were integrated into the engine, as were other special effects. What made us even more confident was that we had to create these systems from scratch; it wasn't possible to re-use this code from Xbox version.

We studied how best to allocate hardware resources between the CPU, GPU and memory under the *Unreal* engine. By the time we completed the prototype, our team had a detailed list of how much memory was allocated to each part of the game (map data, character/animation, engine, UI, textures, and so on), and understood how to work with the constraints of the map data to ensure the good frame rate. We tried to be pessimistic while setting those constraints, which turned out to be a good decision -- later on we encountered some bad surprises related to system resources, but because of our earlier estimates, we had built in a sufficient margin of error to deal with them.

The prototype also proved crucial for organizing the production schedule and estimating the necessary project resources. The effort required to port one map was split into several steps, and we used our experience creating the prototype to accurately define the data flow between steps and estimate the overall resource costs.

Here is an example of how we defined the data flow:



Click for larger image

After this chart was created, we could estimate how long each step would take for 1 person, from which we could generate accurate plans that took into account the schedule and quantity of maps. That would in turn help us determine how many people we needed on the project.

2. Pre-production documentation and staffing decisions. The pre-production phase started three months before the production. It was during this stage that we formed the team, trained people, set up the production model, and made project schedule.

One of the most important steps we took during the pre-production stage was to create Technical Design Documents (TDD), into which we poured all the knowledge we gleaned from our prototype. It wasn't easy to create the TDDs, though. The prototype development team was so busy that nobody had time to document what we were learning. Ultimately I had to force the team to devote time to creating the TDDs by asking people stop their work for 1-2 days to concentrate on writing it. It proved to be a worthwhile exercise, though, since it saved us even more time later on in the project. The TDDs were used to train new people who just joined the team, and helped everyone understand the major technical issues.

The TDDs also included a detailed analysis of all the maps in the original Xbox version of *Splinter Cell*. The level design team, which had grown to seven people during pre-production, took the in-progress Xbox map data and rehearsed how we'd port them over to the PS2, given what we knew about the process and constraints after working on the prototype. At the same time, level designers made an optimization plan for each map. With those rehearsals and documentation, the workflow between teams and the production workload was better understood before we started on the real production. We even had a good idea of which maps would be more challenging to optimize, which enabled us to allocate the appropriate resources to them. The most difficult tasks were assigned to the most skilled developers to try to prevent bottlenecks in process.

On the personnel management side, besides adding more people to the team, we also made adjustments to the team leader's position. When we established the prototype team, we expected those team members to later become the leaders of the functional teams in the main production stage. With their knowledge of the project and experience working on the prototype, they were expected to be good coaches for the rest of the team. What we didn't foresee was that the production team would eventually turn into the biggest team ever in Ubi Soft's history, requiring some team leaders to manage 20-30 people. As such, management skills were actually more valuable than technical skills and experience, so we appointed new leaders for some teams. These people weren't on the original prototype team, but they were experienced team managers. That let the supplanted team leaders focus on technical issues, rather than spending time on administrative tasks.

3. Emphasis on accurate scheduling. The production of *Splinter Cell* for the PS2 took no more than four months. We didn't have much say when it came to setting the due date for master; Ubi Soft was adamant that the game be released by the end of its 2003 fiscal year. Yet we couldn't start production early, since the Xbox version was still under development.

A four-month production cycle is obviously very short for a game, which is expected to be of AAA caliber and up to the standards set by the Xbox version. Our milestones were naturally aggressive to satisfy these constraints. Here's what our schedule looked like:

- From beginning of production to Alpha: 9 weeks
- From Alpha to Beta: 5 weeks
- From Beta to Master: 4 weeks

This schedule appeared very unrealistic, yet we tried to be extremely realistic when we were considering what it would take to achieve it.

Everyone understood that the objective of the project was to make the game by a specified date and of a given quality. To achieve this, we had much more freedom to get the required resources than "normal" projects - the company was willing to dedicate people and spend money to gain time. From the beginning, everyone understood this strategy.

Prior to the start of production, we worked hard to round up the resources we knew we'd need for the project. The Shanghai studio made *Splinter Cell* its primary focus on by allocating many experienced developers to the game and recruiting new people. In addition, Ubi Soft's studios in France and Italy also sent development teams to Shanghai.

We could have had those teams stay in their respective countries and work remotely on *Splinter Cell*, but we gave up this idea. We decided it would be difficult to manage multiple teams across several countries - it added big risks to the project, and we didn't have enough time in our schedule to accommodate that. Having people come to China was obviously more expensive, but it was absolutely necessary to minimize our scheduling risks.

We tried our best to make the production plans as detailed and as accurate as possible, but we realized that the schedule was apt to change for unforeseen reasons. So we regularly followed up on our schedule and made adjustments to it as necessary to ensure we always had an up-to-date "vision" of the project. We had project status meeting every two weeks in which we discussed our current major problems and analyzed their effects on the schedule. We also explored actions we could take to offset these problems.

Before each important milestone, we re-checked the schedule using a system that focused on the critical path to reach the target. For example, say there are 10 tasks to complete to reach the Alpha, and of those the 10, five tasks can be done in parallel. The other five tasks must be done in a specific order, and assigning more people to those tasks won't necessarily save much time. In this example, these five tasks are the critical path to reach Alpha. Estimating how long those five tasks will take helps us know if we have a scheduling problem.

4. Matrix management, frequent personnel evaluations. *Splinter Cell* for the PS2 had the shortest development schedule in Ubi Soft Shanghai's history, not only because we had the biggest team, but also because the team was the most efficient. The development team was made up of six functional teams: engine, gameplay programming, animation, graphics, level design and sound. We also had a data manager who was dedicated to database management.

The *Splinter Cell* (PS2) Graphic and Animation Team

The graphic artists and level designers had to work closely on data production. So inter-team groups were established in a matrix-management fashion. Each group consisted of a level designer and two or three artists, and each group was responsible for the maps they worked on, and the individuals were physically located close to each other in our office to ensure good communication.

The graphics team was the biggest team, made up of over 20 people. Besides the team leader, we had a graphics technical director to ensure the whole team's technical skill and he is also responsible for communication with other team on technical issue.

Team leaders were given the full responsibility of his team and reported to producer. To bolster the efficiency and morale of the whole team, we conducted evaluations of team members frequently throughout the production cycle. The evaluation was result-oriented, which means we evaluated people only according to the outcome of his/her work. The result-oriented evaluation has two advantages over a traditional personnel evaluation:

- It is simple. There's less to consider since the evaluations are so frequent. It was possible to do evaluations very rapidly - every two weeks.
- The evaluation helps the managers keep track of the project's progress.

People with good evaluation results were rewarded by monthly bonus. Management talked to those who received bad evaluations, to apply the appropriate pressure. In the middle of the production, there was so much pressure on us that we considered setting a more severe work policy, in which each team had to assign the "improvement needed" rank to at least one person during each evaluation period. We also we talked about punishing people who continually received this rank. While harsh, the goal of these rules was to spur people to improve their work no matter how good their team was. By having to single out at least one person on each team, everyone would be under pressure to improve to avoid being that person. In the end, though, we never implemented these rules - we knew it would adversely affect morale, and we preferred more positive atmosphere for the project.

Throughout the whole production schedule, the team worked very hard, and the team leaders played an important role in ensuring the best performance from their teams. For project with huge teams, it is critical to choose good team leaders from the beginning.

5. QA management. The Xbox version of *Splinter Cell* provided a benchmark for how good the PS2 version should be. Yet it was a challenge to ensure game quality with more than 70 developers working under such a tight schedule.

Our regular team evaluations helped us to closely track our quality, and in addition to those, we also had more formal evaluations that served as a quality check-up prior to milestones like Alpha and Beta. These milestone meetings usually took two or three days and were split into two parts: engine evaluation and data evaluation.

During the game data evaluation meetings, we arranged a meeting room and gathered all the team leaders. We checked each map with the level designer and artist who worked on this map. We generally checked four aspects of the map:

- Memory use
- Frame rate
- Gameplay
- Graphics

These topics covered the major risks that concern game quality. If certain map was below the quality standard, the team working on that map might be asked to re-work it, using overtime. Another evaluation would be arranged shortly thereafter to check it again.

Engine evaluation was not so straightforward. Since many engine features cannot be visualized, we had the game's lead programmer and the studio's programming director (the latter of whom wasn't working on the project and could provide an objective opinion) to check the quality and progress of each programmer's work.

We also appointed an art director after we reached the Alpha stage, who was responsible for ensuring that all the maps had the same artistic style - the *Splinter Cell* style. We realized that with lots of people working on their own maps, we might get some maps that looked like another game, and this promoted artistic consistency.

Prior to the Beta stage, we found it difficult to control the quality and stability of the game. We had a huge team with people operating at different skill levels. The quality improvements became so detailed that team leaders had to spend more time supervising team members. When we started debugging, the situation deteriorated even more, as less-skilled developers sometimes introduced new bugs as they tried to fix existing ones. To remedy the situation, we started the "SWAT team" organization.

SWAT team is a concept we picked up from the *Splinter Cell* Xbox development team. The idea is to reduce the team size after a certain stage of production. Only the best people continue to work on the project, and their expertise is fully leveraged. A side effect of this is that it's also easier for managers to control quality; they don't have to deal with lots of people at different skill levels.

This organization was implemented with the graphics team (the biggest team with approximately 30 people). The SWAT team was made of the six best artists who worked closely with art director and technical director to improve the quality of the graphics.

During the final debugging stage, when the reported bugs were small in quantity but very complicated to fix, the whole graphics team was reduced to just two people-the graphics team leader and the technical director - to minimize the risk of generating more bugs.

Ubi Soft's Shanghai Office

What Went Wrong

1. Cultural barriers within the team. We wouldn't have completely this project on time without the help from other Ubi Soft studios. But the team's diverse composition made working together difficult. Many problems that we encountered stemmed from cultural differences.

The most obvious barrier was language. Three different languages we spoken by our team: Chinese, French and Italian. We had to use English as the working language, but not everybody was fluent in the language, and some people didn't even speak it at all. The first time it became an issue was during staff training. Since all the technical documents were in English and the technical director for the graphics team spoke only Chinese and English, it was very difficult to train developers who didn't speak English.

Before production began, we planned to mix people from different countries into one big team, thinking that it would help people get to know each other quickly. Then we realized that the language problem made that unfeasible. So we created two teams instead: local and foreign. Each team had a team leader who would translate project information to his team. Yet this structure created two other problems:

1. It made the decision-making process more difficult. The two team leaders would sometimes have different opinion on techniques, working arrangements, or requests to other teams. The worst was when these two team leaders gave conflicting information to their respective teams, which naturally caused confusion.
2. It created feelings of competition. People subconsciously - and sometimes consciously -- considered developers on the other team their competitors. While that can be good in a way, generally the situation made conditions uncomfortable, prevented a sense of teamwork and inhibiting the sharing of knowledge.

We didn't realize these problems until some bad situations occurred, at which point we tried to improve the situation. But since the production schedule was so short, we weren't able to completely solve the problems in time.

There are broader cultural difference between Chinese and Europeans, too. While these variations didn't directly lead to any problems, the differences were noticeable and concerned the project managers. Perhaps the biggest difference was that the local (Chinese) developers generally showed more respect to authority. They were more willing to take time to understand the technology and development process and follow those conventions. In contrast, European developers tended to question authority and try to find a better solution. Sometimes they managed to come up with better ideas, but sometimes they wasted time. On a project with such a compressed schedule, that hurt.

The power plant level for *Splinter Cell* (PS2)

2. Problems with data delivery. To deliver the highest quality game within the time constraints, we had to work on the raw data from the Xbox version of the game. That raw data consisted of 3ds max files, which were exported to the game editor to become game data.

When we started production in October 2002, we discovered that many of these raw data files were missing from the package we received from the *Splinter Cell* Xbox team. Consequently work on some maps could not start as scheduled, and our art team had to check all the maps to determine the scope of the problem.

We discovered that the problem was caused by the data archiving system. Since the engine only needs the exported data (the game data), the artists weren't obliged to archive the raw data, and sometimes didn't save files using the correct file names. So when the Xbox team packed up their database to send to us, some raw data files were missing and some files were named incorrectly. We had actually seen this problem arise earlier in the project, when we were working on the prototype. We asked the Xbox team to pay attention to the problem, but their team was also under tremendous pressure and sometimes the developers were too busy to adhere to rules that ultimately didn't affect their version of the game.

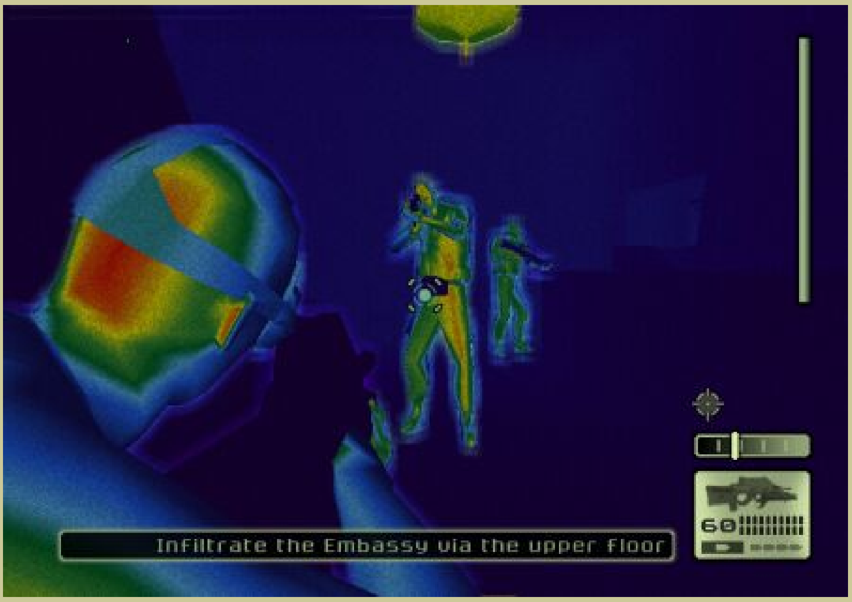
The missing data caused some bad feelings between the Xbox and PS2 *Splinter Cell* teams. The PS2 team felt that the Xbox team didn't care about the PS2 version's development, and the Xbox team felt that the PS2 team wasn't searching the package they sent carefully enough. Unfortunately we weren't able to completely solve the problem: some files couldn't even be found in the database managed by the Xbox team. But everything turned out fine, fortunately -- our engineers were able to create a tool that could recover the raw data from the game data files, so we were able solve the problem and extract the missing files.

If we could have foreseen this problem and started work on that file extraction tool before production started, we would have gained some precious development time. The lesson we learned is that when a risk is identified (e.g., not being able to get all the raw data), if one solution is not sufficient (asking Xbox team to be careful about storing data), we should have a backup solution (developing a tool).

3. Engine solutions vs. data solutions. Lighting and shadows are a very important feature in *Splinter Cell*. While working on the prototype, we encountered a technical problem that affected the accuracy of lighting maps and characters. The engine team thought it would be difficult to solve the problem quickly, but the prototype needed to have this feature working well. So instead of waiting for the engine team to solve the problem, we asked an engineer to create a utility for the artist to adjust parameters manually. It was temporary workaround.

While this was probably the best solution to complete the prototype on time, when production started on the real game, we were still using this solution on all the maps. Everyone on the art team was adjusting these parameters by hand. Later we discovered this created more problems. First, each map had its own set of parameters, making it difficult to control the overall quality. Second, because the quality of the lighting depended so much on the combination of certain parameter settings, small changes made to the tool would radically change all the setting, with disastrous results. The situation caused particularly big risks during the debugging stage.

If we had solved the problem in the engine in the beginning, the situation would probably have been much better. Although an engineer might have devoted more time to making the feature work properly, it would have save the art team lots of time. Above all, it would have been much better for quality control since just one people would have been controlling the quality of this feature for the entire game. This example illustrates that for features widely used throughout a game, an engine solution implemented from the outset is usually a wise choice.



Thermal imaging in *Splinter Cell* for the PS2

On the flip side, engine solutions are usually more complicated than data solutions. During the later stages of development (e.g. after beta), engine changes can be quite risky as they can break other parts of the game - which we also found out the hard way. During *Splinter Cell*'s debugging stage, as we attempted to fix bugs in some maps, we added new code. But those changes introduced new bugs into the maps, so we ended up going back to the data solution. So the caveat to my earlier rule is that if there's not enough time left to debug new code, a data solution can be a better choice.

Whether to choose an engine solution or data solution really depends on the scope of the problem - whether it's global or specific to a certain part of the game -- and what stage of development you're in. If it's the beginning of a project and the problem is a global one, spending time to fix the engine is worthwhile. If the problem exists only in certain maps and you're already at the debugging stage, patching with a data-oriented solution is safer. (Although these solutions can make a perfectionist programmer uncomfortable.)

4. The risk of AI debugging. *Splinter Cell* provides the player with so much freedom that it is difficult to hide problems, especially problems related to AI behavior. Sam Fisher, the player's character, is equipped with a wide range of gadgets that players can use to observe and interact with NPCs.

One major risk we constantly discussed during the making of *Splinter Cell* was how to debug the AI. As early as the Alpha stage we foresaw risks in debugging the AI system, and it remained risky up until the end of project. In fact we had to delay the first focus-group playtesting session because the game was full of AI-related problems.

There were many reasons why we had faced so many AI bugs in *Splinter Cell* for the PS2. The most important ones were the following:

1. We started development on the PS2 version using the Xbox version's pre-beta code, which was still quite buggy.
2. AI bugs are usually harder to spot than other bugs. Testing this aspect of a game is more complicated and requires better communication between testers and developers.
3. The programmers on the PS2 team were not the one who coded the original AI, so they were not familiar enough with the code to solve the AI problems.
4. The optimizations we made for the PS2 version introduced more bugs, which were quite difficult to track.

There's no obvious solution to problem 1, as we didn't have much choice if we were to adhere to our schedule.

For problem 2, we made some efforts to improve the communication between the programmers and testers. For instance, we arranged training for the testers in which the AI programmer explained how to identify NPC behavioral bugs. We also equipped the testers with video capture cards so that when they reported AI bugs, they were able to attach an AVI file which made it easier to explain and demonstrate the bug.

To address problem 3, we managed to get two of the engineers who coded the AI engine for the Xbox version to join our team. They were instrumental in debugging the AI.

We should have done better in relation to problem 4. Level designers made some of the optimizations, but not all the level designers knew the AI well enough to ensure bug-free scripts. They needed programmers to tell them what was wrong in their scripts and how to correct the problems. This led to a bottleneck in the engine team during the debugging stage. If we had provided sufficient training for our level designers about the AI scripts, we would have prevented plenty of bugs in the AI, and made the debugging process more efficient.

5. Some regrets. After *Splinter Cell* for the PS2 was released on March 27, 2003, I collected player feedback from the Internet. The general feedback has been quite positive; many players like the game - especially the "power-plant mission" which is exclusive to the PS2 version. Players also like "snow-suit mission". The graphic quality has been complimented, as has the game play in general. However, there have also been some negative comments from players:

- Player comment: "It's too easy."
  Some players think the PS2 version is too easy and too short, especially compared to the Xbox version. Some people also find the difference between the "normal" and "hard" mode too subtle.

  The fact is that we reduced the difficulty of PS2 version on purpose, since almost all of the playtesting of the Xbox version indicated that people felt that game was too hard. We keep in mind that there are more "casual" PS2 gamers than there are in the Xbox market. So in order to adjust the game to this market, we adjusted gameplay in *Splinter Cell* for the PS2 in a few respects, including player control, AI and level design. However, it's possible we might have gone too far, as some players seem to find the game not challenging enough.

  We didn't have much time to fine tune the game play for the "Hard' mode. Only some parameters are affected in that setting, like the damage dealt by NPCs, the quantity of Sam's ammo, and tolerance for triggering alerts. Also, there's no extra reward to player who play the "Hard" mode.
- Player comment: "It has a poor ending."
  Most players don't like the ending of the game. There's no really challenge in the last level, and no big reward either. This is partially due to the Tom Clancy-style, which is not adaptable to a "boss fight" concept. But it also comes from the fact that we didn't have much time to create something really special. We didn't even have time to make any secrets/bonuses for the game, as the final debug was so tight and we couldn't afford to add any new features, lest we break something or introduce new bugs. We actually did make a secret mode for the game, but ultimately we had to ditch it because there was no time to debug it.

- Player comment: "There's too much loading."
  I saw some players complain that the PS2 version's loading happens too frequently. This is due to the hardware constraints - the PS2 has less memory than the Xbox and PC. We knew about this problem from the beginning. We considered making the loading process more interesting for players by displaying special effects as it occurred (some animation or sounds, etc.), but we weren't able to come up with any good ideas. We used just one such "effect" in the game, and many people probably haven't even noticed it.

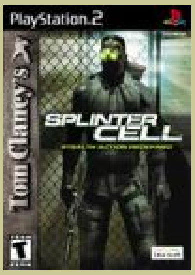
Summary

In *Splinter Cell* for the PS2, we succeeded with our strategy of applying a large amount of resources to meet a short production schedule. This strategy works for the porting projects which already have clear vision, based on the original version of the game. I don't think the same strategy would work for a game developed from the ground up, however.

Although we had very big production team, the team's size needed to be adjusted at different stages. In the early stages, a small but efficient team was a good choice. The team that built the prototype created solid base for the main production stage. Likewise, at the end of project, the team size needed to be reduced to efficiently control our debugging efforts.

It's very important but sometimes quite difficult (especially with lots of people) to ensure efficiency in your production team. I've found that the work done during the prototyping and pre-production stages is the key. A thorough assessment of potential problems and risks will identify impending issues you face, and with that information, you can construct plans to solve them before production begins.

Game Data



***Splinter Cell* (PS2)**

Publisher: Ubi Soft
Number of full-time developers: 76
Number of contractors:18
Length of development: 5 months
Release date: March 28, 2003
Target platform: PlayStation 2
Development hardware: PS2 dev tools, PCs avg. Athlon dual 1800+
Development software used: Unreal Warfare, Code Warrior, 3D Max, Photoshop, Ubi's animation tools, Optpix, Microsoft Visual SourceSafe
Project size: 3.47GB