GAME DEVELOPER MAGAZINE

JANUARY 2002

UBM Tech

Return

GAME PLAN

LETTER FROM THE EDITOR

War Games

The game industry overall did an admirable job of weathering the public criticisms lobbed its way immediately following September 11. Some popular genres, such as real-time tactical shooters, especially those featuring counter-terrorist operations, came under fire for hitting too close to home in post–September 11 America. In just one among many similar actions by game publishers, Ubi Soft announced shortly after September 11 that it would be delaying the release of Red Storm's ROGUE SPEAR: BLACK THORN indefinitely to modify sensitive content.

What a difference a month makes. Not only was the modified BLACK THORN released with fanfare less than a month after its original ship date, but by then Ubi Soft was proudly publicizing the fact that they had recently licensed the ROGUE SPEAR engine to a contractor working for none other than the U.S. Department of Defense (DOD). New maps and scenarios will be added in order to train Army troops in urban counter-terrorist strategy and tactics.

Technology sharing between the game and defense industries is nothing new, of course. But our industry's collaborations with DOD suddenly takes on a whole new significance as ground troops, possibly trained in part on technology that originated from game developers for entertainment products, begin fanning out through Afghanistan. Perhaps some of the game industry's most outspoken critics will find themselves unsure of whether to continue to chastise the game industry or raise a hand in salute. Strange times tend to make for strange bedfellows.

We've come a long way since 1980, when Atari's tank simulation arcade game, BATTLEZONE, caught the attention of Pentagon officials who wanted Atari adapt it for military training. To say that most game developers — then and now — don't fit the mold of your typical straitlaced defense contractor is an understatement. Nonetheless, collaboration efforts have stepped up in recent years.

One organization to keep an eye on is the Institute for Creative Technologies (www.itc.usc.edu), founded in 1999 as a joint effort between the Army (who provided $45 million in funding), game developers, Hollywood talent, and the University of Southern California, which recently announced development of two projects that will have both military and commercial applications. C-FORCE is being developed for consoles by Future Combat Systems (a venture between Sony Pictures ImageWorks and Pandemic Studios), and CS-12 will come to PC from Quicksilver Software. Effectively, these are the first two commercial games ever commissioned by the military and developed with DOD's direct input, through the Army's Training & Doctrine Command bureau.

Such efforts could end up being good PR for the game industry at a time when public interest in the military is particularly high. DOD-backed games could also serve as a recruitment aid for the Army (much the same way *Top Gun*, produced under the beaming auspices of the U.S. Navy, sent aspiring Mavericks to Navy recruiting offices in droves). However, we are also opening our industry up to a whole new level of scrutiny. What headway we've made convincing the general public that videogames aren't sophisticated training for would-be high school snipers may well reemerge when people realize that the Army is co-developing commercial applications for its own training purposes, and with taxpayer dollars. (The difference, of course, is that the Army has no interest in using these applications to train soldiers to shoot weapons. The emphasis, say Army officials, is strictly on strategic and tactical training.) It would be wise for our industry to continue to keep tabs not only on our known critics but also on those who would ride our coattails to further their own goals.

Welcome, Hayden. This month, we're pleased to welcome Hayden Duvall as our new Artist's View columnist. Hayden brings with him a wealth of experience in both art and the game industry, and currently works as lead artist at Confounding Factor on their upcoming console title, GALLEON.

Jennifer

GameDeveloper

600 Harrison Street, San Francisco, CA 94107 t: 415.947.6000 f: 415.947.6070

Publisher
Jennifer Pahlka jpahlka@cmp.com

EDITORIAL

Editor-In-Chief
Jennifer Olsen jolsen@cmp.com

Managing Editor
Laura Huber lhuber@cmp.com

Production Editor
Olga Zundel ozundel@cmp.com

Product Review Editor
Tor Berg tberg@cmp.com

Art Director
Audrey Welch awelch@cmp.com

Editor-At-Large
Chris Hecker checker@d6.com

Contributing Editors
Daniel Huebner dan@gamasutra.com
Jonathan Blow jon@bolt-action.com
Hayden Duvall hayden@confounding-factor.com

Advisory Board
Hal Barwood LucasArts
Ellen Guon Beeman Beemania
Andy Gavin Naughty Dog
Joby Otero Luxoflux
Dave Pottinger Ensemble Studios
George Sanger Big Fat Inc.
Harvey Smith Ion Storm
Paul Steed WildTangent

ADVERTISING SALES

Director of Sales & Marketing
Greg Kerwin e: gkerwin@cmp.com t: 415.947.6218

National Sales Manager
Jennifer Orvik e: jorvik@cmp.com t: 415.947.6217

Senior Account Manager, Eastern Region & Europe
Afton Thatcher e: athatcher@cmp.com t: 415.947.6224

Account Manager, Northern California & Southeast
Susan Kirby e: skirby@cmp.com t: 415.947.6226

Account Manager, Recruitment
Raelene Maiben e: rmaiben@cmp.com t: 415.947.6225

Account Manager, Western Region & Asia
Craig Perreault e: cperreault@cmp.com t: 415.947.6223

Sales Associate
Aaron Murawski e: amurawski@cmp.com t: 415.947.6227

ADVERTISING PRODUCTION

Vice President, Manufacturing Bill Amstutz
Advertising Production Coordinator Kevin Chanel
Reprints Stella Valdez t: 916.983.6971

GAMA NETWORK MARKETING

Senior MarCom Manager Jennifer McLean
Marketing Coordinator Scott Lyon
Audience Development Coordinator Jessica Shultz

CIRCULATION

BPA INTERNATIONAL

Game Developer is BPA approved

Group Circulation Director Catherine Flynn
Circulation Manager Ron Escobar
Circulation Assistant Ian Hay
Newsstand Analyst Pam Santoro

SUBSCRIPTION SERVICES

For information, order questions, and address changes
t: 800.250.2429 or 847.647.5928 f: 847.647.5972
e: gamedeveloper@halldata.com

INTERNATIONAL LICENSING INFORMATION

Mario Salinas
t: 650.513.4234 f: 650.513.4482 e: msalinas@cmp.com

CMP MEDIA MANAGEMENT

President & CEO Gary Marshall
Executive Vice President & CFO John Day
President, Business Technology Group Adam K. Marder
President, Specialized Technologies Group Regina Starr Ridley
President, Technology Solutions Group Robert Faletra
President, Electronics Group Steve Weitzner
President, Healthcare Group Vicki Masseria
Senior Vice President, Human Resources & Communications Leah Landro
Senior Vice President, Global Sales & Marketing Bill Howard
Senior Vice President, Business Development Vittoria Borazio
Vice President & General Counsel Sandra Grayson
Vice President, Creative Technologies Philip Chapnick

CMP United Business Media | GamaNetwork

WWW.GAMANETWORK.COM

Return

SAYS YOU

THE FORUM FOR YOUR POINT OF VIEW. GIVE US YOUR FEEDBACK...

Sound Design Kudos

I just wanted to thank Andrew Boyd for his kind words and intelligent observations about the sound design of MEDAL OF HONOR and MEDAL OF HONOR: UNDERGROUND in "Escape from Bad Audio" (October, 2001). We here in the audio department at Electronic Arts Los Angeles/DreamWorks Interactive have been enjoying reading it and we're pleased that you so astutely picked up on some of our techniques. It was an excellent article.

Erik Kraber
Senior Sound Designer
MEDAL OF HONOR
MEDAL OF HONOR: UNDERGROUND
via e-mail

I enjoyed Andrew Boyd's "Escape from Bad Audio." I am a member of the AIAS Best Music category panel, and I can tell you your points are right on. We are all very dedicated to great music and sound. One of the problems we have faced is that many noteworthy games don't make it for our review because the game developer or publisher didn't submit them. We have been working to improve this by communicating throughout the year about games we think are great, which should improve the pool of excellent games for final review.

Thanks for bringing audio to forefront of thought among developers and publishers with this article.

Lennie Moore
via e-mail

Mobile Platforms?

The issue of games for mobile devices is interesting. While RJ Mical gives some lip service ("Some Assembly Required," Soapbox, November 2001) to the idea that we might take new risks and have a different development cycle than the current standard one, most of his article seems to suggest that mobile platforms have lesser hardware and what we really want is better hardware, so we can play on them the same games that we play on PCs.

On the game design side, it might be interesting to see games designed for mobile platforms, instead of adapted to them. Thinking games (which include adventure games) seem like a good match to PDA-style games, because they fit the control method and the goal of saving battery life. For action games, the control limitations may also not be that serious, especially when targeting non-hardcore audiences. My father liked some action games on my old VIC 20. These days, he doesn't play any action games, although he likes MINESWEEPER and such. I'd imagine that one reason is the complex control methods of current games. Simpler hardware could mean a simpler interface, and that might not be a bad thing.

Eyal Teler
via e-mail

Let us know what you think: send us an e-mail to editors@gdmag.com, or write to *Game Developer*, 600 Harrison St., San Francisco, CA 94107

Apples vs. Codewarrior

Jamie Fristrom's product review of ProDG in the November issue of *Game Developer* ("SN Systems' ProDG 2 for PlayStation 2," Product Reviews, November 2001) implies a comparison of Codewarrior on Dreamcast to ProDG on Playstation 2. We feel that this could be misleading to readers of the magazine in that this is not an apples-to-apples comparison, and implies that CodeWarrior is not a mature enough product on PlayStation 2.

The situation is quite the contrary. Codewarrior is the industry-leading product for PlayStation 2 and Gamecube development. Feature for feature, we match or exceed those described in the product review, but also have some unique products, features, and services exclusively for Codewarrior users.

We strongly encourage any licensed Playstation 2 or Gamecube developers out there to evaluate our product themselves. If you're interested, let us know by sending a message to games@metrowerks.com.

Brian Gildon
Director, Entertainment Products
Metrowerks Corp.

PRODUCT REVIEW EDITOR TOR BERG RESPONDS:

While it is not our practice to run comparative reviews of development tools, in the case of Jamie's review of ProDG, both he and I felt that mentioning a competitive product established a tone for the review that would be helpful to the reader. Treyarch had evaluated Codewarrior for Dreamcast and found it to be inappropriate for the studio's specific development needs. This experience led Jamie and the Treyarch team to examine SN Systems' ProDG 2 for Playstation 2.

In today's game industry, development tools are generally too specialized to compare to one another. Feature sets are rarely even similar, let alone equivalent. So a feature-by-feature comparison is inevitably unfair to one product or the other. Thus, we regret implying a comparison between ProDG and Codewarrior.

Codewarrior has indeed been an industry leading product for a long time and has been used in the development of many cutting-edge titles. We look forward to reviewing Codewarrior on its own merits in the near future.

CORRECTION

In the information box included in the review of SN Systems' ProDG 2 for Playstation 2 on page 9 of the November 2001 issue of Game Developer, *the price of the product was listed incorrectly due to an editing error.*

The correct price for ProDG 2 for Playstation 2 is $5,000 per unit for the first nine units, with reduced volume pricing for larger orders.

We apologize for any confusion that this error may have caused.

Return

INDUSTRY WATCH

daniel huebner | THE BUZZ ABOUT THE GAME BIZ

EA improves, Midway lags, SNK closes. Electronic Arts managed to post better sales and a smaller second-quarter loss than last year, but the company's net loss still reached $32.8 million on revenues of $240.2 million. EA lost $38.9 million on sales of $219.9 million in the second quarter one year ago. The results were largely within analyst expectations. Prior to releasing its second-quarter financials, the company made a move to improve the performance of its online subsidiary, EA.com, by announcing layoffs of 200 to 250 staff from the unit. EA executives said the cuts had to be made in order for the company to make that unit, which trades as a separate tracking stock, profitable by fiscal 2003.

While EA boosted its sales and cut its losses, Midway announced a steep drop in revenue. The company's revenues slid 40 percent from last year to just $28.3 million. Despite the lower sales, Midway managed to cut its loss to $7.7 million from last year's $10 million.

Japanese hardware and software developer SNK Corporation has ceased operations entirely. The company, which created the NeoGeo arcade system as well as legendary games like VANGUARD, filed for bankruptcy last April, and had earlier closed down many of its offices.

McNally family donation increases IGF Grand Prize stakes. The Independent Games Festival is sweetening the grand prize pot by posting an additional $5,000 to the giant check handed out with the Seumas McNally Award for Independent Game of the Year. The total amount of the prize is now set at $15,000, with the additional funds coming from the Seumas McNally Memorial Fund created in honor of the lead programmer of Longbow Digital Arts. The IGF Grand Prize was posthumously named in honor of Seumas McNally after his death of Hodgkin's lymphoma shortly after the festival in 2000.

Acclaim, Activision, and THQ raise expectations. Acclaim reported fourth-quarter earnings of $2.9 million on revenues of $46.5 million, marking a return to profitability after a loss of $63.5 million in the same period last year. The company said that its DAVE MIRRA FREESTYLE BMX and CRAZY TAXI franchises were responsible for much of the improvement, leading Acclaim to raise its guidance for revenue and profit for the fiscal first two quarters of 2002.

LEGENDS OF WRESTLING, helped Acclaim Entertainment return to profitability in its fourth quarter.

Activision also reported encouraging financials, posting solid second quarter numbers despite lower revenues and profits than last year. The company's net revenues for the quarter were $139.6 million, down from $144.4 million last year. The company's profits were also off from last year's results, down 47 percent to $2.2 million. Despite the slip, Activision's results exceeded analyst expectations, and the company raised guidance for the year-end quarter and predicted continued strong sales.

THQ, for its part, translated a strong Game Boy Advance catalog into a 144 percent increase in third-quarter profit. The company reported net income of $3.2 million, compared to $1.3 million last year. Revenues for the quarter reached $68 million, up from $53.3 million a year ago. 51 percent of THQ's sales for the quarter came from Game Boy and Game Boy Advance titles. THQ also increased its forecast for the coming year.

3DO quarterly sales fall by half, Hawkins underwrites new stock. 3DO cut costs in its second quarter but couldn't match its savings with increased sales. For the quarter, revenues dropped by more than 50 percent to $9.9 million from $20.2 million in the same period last year. Cost cutting helped 3DO trim its quarterly loss by 42 percent to $9.7 million from a loss of $16.7 million in the second quarter last year. These results were in line with analyst expectations for the company.

CEO Trip Hawkins again stepped in to provide the company with desperately needed working capital, using personal funds to purchase over $8 million of 3DO shares in a recently completed private placement round of financing that raised $9.75 million for the company. Hawkins bought 3.9 million of a total of 4.7 million shares. The shares we were issued at $2.06 per share.

Sega teams with Microsoft and THQ. Sega continues to cut deals in its bid to become the industry's largest third-party software provider. Sega and Microsoft unveiled a new partnership at the Tokyo Game Show which will bring a number of Sega games exclusively to the Xbox. Most notably, the next game in the SHENMUE series will only appear on the Xbox in North America, and Sega will also port PHANTASY STAR ONLINE to the Xbox as well. New online PC games may also result from collaborations between the companies. Sega and Microsoft also said their work together would extend into the arcade market by designing an arcade motherboard based on the Xbox technology.

Sega also reached an exclusive, multi-title co-publishing agreement with THQ to publish 16 Sega titles for Nintendo's Game Boy Advance. THQ said it will co-publish Game Boy Advance titles featuring Sega brands in North America through 2003.

UPCOMING EVENTS CALENDAR

MILIA 2002
PALAIS DES FESTIVALS
Cannes, France
February 4–8, 2002
Cost: variable
www.milia.com

AMERICAN INTERNATIONAL TOY FAIR
JACOB K. JAVITS CONVENTION CENTER
New York, N.Y.
February 10–14, 2002
Cost: none
http://www.toy-tma.com/AITF

D.I.C.E. SUMMIT
HARD ROCK HOTEL
Las Vegas, Nev.
February 28–March 1, 2002
Cost: variable
www.interactive.org

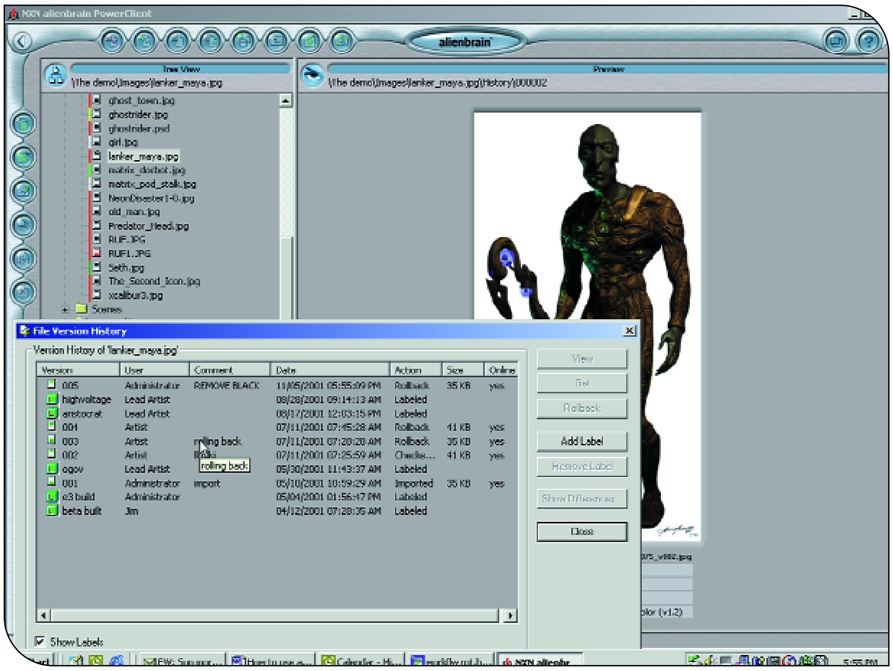
Return

NXN Software's Alienbrain 5

by chris corry

After spending a few weeks with the latest version of NXN Software's flagship product, Alienbrain, I find myself returning again and again to the same conclusion. These guys have taken the complicated and daunting task of enterprise-wide asset management and made it . . . sexy?

That's right, sexy. Alienbrain is a beautiful and altogether elegant client/server solution for managing a game project's files and multimedia assets. And version 5.0 moves the product toward NXN's ultimate goal of fully digitizing the entire production pipeline. While not without its flaws, Alienbrain comes closer to organizing and managing the entire cradle-to-grave life cycle of a development team's collective output than any other product. In fact, were it not for the product's 24-carat price tag, I'd be completely comfortable recommending Alienbrain as a must-have product for any development team. As it is, however, the story is a bit more complicated.



Version	User	Comment	Date	Action	Size	Online
005	Administrator	REMOVE BLACK	11/05/2001 05:55:09 PM	Rollback	35 KB	yes
highvoltage	Lead Artist		08/28/2001 09:14:13 AM	Labeled		
aristocrat	Lead Artist		08/17/2001 12:03:15 PM	Labeled		
004	Artist		07/11/2001 07:45:28 AM	Rollback	41 KB	yes
003	Artist	rolling back	07/11/2001 07:28:20 AM	Rollback	35 KB	yes
002	Artist	[illegible]	07/11/2001 07:25:59 AM	Checke...	41 KB	yes
ogov	Lead Artist		05/30/2001 11:43:37 AM	Labeled		
001	Administrator	import	05/10/2001 10:59:29 AM	Imported	35 KB	yes
e3 build	Administrator		05/04/2001 01:56:47 PM	Labeled		
beta built	Jim		04/12/2001 07:28:35 AM	Labeled		

Alienbrain helps you keep a complete version history of every file in a project.

As you're no doubt aware, keeping track of all the resources required to build a contemporary game title is a huge task. On a day-to-day basis, game developers must deal with many different file types, each of which may appear in a dozen different forms or stages within an art pipeline. We need to worry about source art files, sounds, scripts, exported art, preprocessed art, postprocessed art, bundled art, platform-specified art . . . you name it. Even a relatively small title typically needs to deal with thousands of different files, and unless the team is well organized and eternally vigilant, keeping track of all this inherent complexity can be an absolute nightmare. Alienbrain straightens out this dire state of affairs, helping team members coordinate the modification and creation of new assets in a controlled and methodical fashion.

At its core, Alienbrain is built around a version control engine that has been specifically designed to handle the media types commonly used in the development of interactive titles. In other words, Alienbrain is tuned to deal with exactly those sorts of large binary files that bring other version control systems to their knees. In fact, you can place any sort of file that you want into Alienbrain and it is likely to handle the data with aplomb. The main client's elegant user interface allows for rapid, real-time previewing of most standard file formats without resorting to external viewer programs that need to be spawned in a separate window. For those not-so-standard proprietary file formats that we all find ourselves relying on from time to time, Alienbrain has a robust and comprehensive plug-in architecture for referencing external viewers or, for the more ambitious among you, custom-built viewers that integrate seamlessly into the client. Overall, the main, artist-oriented Alienbrain client is a model of flexible user-interface design. While it can, at times, threaten to over-

CHRIS CORRY | *Chris is a brash but harmless fellow, often found lurking in the shadows of LucasArts Entertainment Company. He may soon be bald. You can reach him at chrisc@lucasarts.com*

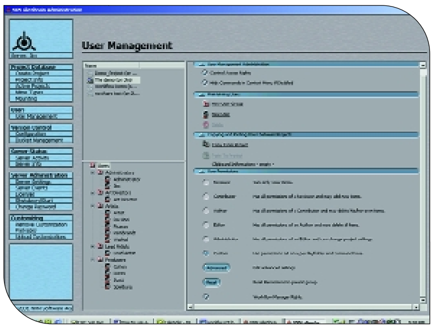
★★★★★ excellent

★★★★ very good

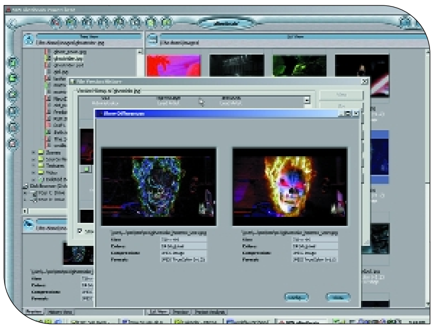
★★★ average

★★ disappointing

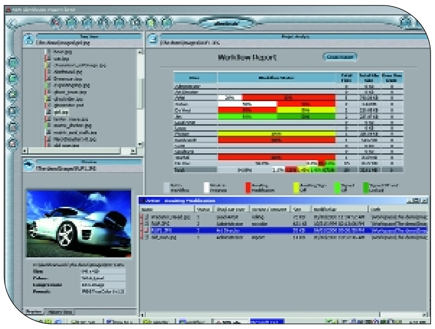
★ don't bother



The administrator can set up new users quickly and give each unique permissions.



View side-by-side comparisons of two files in a file history.



Comprehensive productivity reports offer a detailed snapshot of a project.

whelm you with its inherent customizability, virtually every user is assured of being able to set up the interface in whatever way he or she desires.

Alienbrain supports all of the standard version-control features and then some. The software tracks the history of a file as it evolves over time, encouraging users to associate comments with each new version and allowing effortless rollbacks to previous versions. Version rollbacks are nondestructive; the previous, rolled-back version of the file becomes the new tip of the revision tree, and you can still access any previous version. The main client even offers a fancy visual version comparison feature that, for certain supported file types, lets you compare multiple versions of an image file side by side. You can check files in and out directly from within common applications such as Photoshop, Maya, and 3DS Max, and the client even extends the Windows shell namespace, allowing you to manipulate version-controlled files directly from an Explorer window.

There are a few lapses. The Alienbrain server is only available on Windows platforms, and overall the whole product has a very Windows-centric feel to it. Macintosh (OS 9.x or higher) and Linux clients are available, but they provide only basic functionality such as check-in, check-out, and previewing. For text and source code files, Alienbrain does provide rudimentary differencing and merge facilities, but because NXN relies on the Windiff and WinMerge utilities, these features are far from best of breed. And although projects can opt to allow or disallow multiple check-outs on a per-file-type basis, some programmers might complain about the lack of support for branching and version pinning. While the requisite integration with Visual Studio is well done, a source-code-oriented version control system such as Perforce may still be a better choice for programmers. At the very least, if you do decide to commit your programmers to Alienbrain, you'll want to invest in a beefed-up utility such as Araxis Merge to help with differencing-related tasks. The good news is that NXN makes it relatively easy to substitute a third-party merge application, so a motivated project manager is likely to find a way to get the entire team using Alienbrain.

If the Alienbrain client is our sexy femme fatale, the server is her well-muscled bodyguard. Alienbrain's server-side databases support robust user-management tools that provide a tremendous degree of control over user rights and feature access. For example, you can easily set up a group that allows check-ins and check-outs but disallows file deletions from a project. The Alienbrain administrative user interface is well designed and powerful, allowing you to configure projects and monitor server performance in real time from any machine on the network.

Alienbrain is also a marvel of customizability. Beyond the aforementioned chameleon-like abilities of the client, there is almost no functional aspect of the program that cannot be modified through scripting or with a bit of programming elbow grease. Much of the Alienbrain client is actually implemented in JScript (Microsoft's take on JavaScript), and while there's a lot here to wrap your head around, once you've mastered the landscape of the programming environment, you can perform some remarkable modifications. One customization example that particularly impressed me was a script that hooked the client's "Post_CheckIn" event and used an external utility to process the newly checked-in file into a platform-specific format. Such functionality could ensure that users always had locally processed, platform-native resources that were ready for use. Because files can have custom properties associated with them, and because these properties are all exposed to the scripting system, the customization possibilities are nearly limitless. As if this level of control weren't enough, NXN also supplies an SDK that lets Alienbrain users modify and extend the product using C++. If all this talk of scripting and programming intimidates you, NXN offers a variety of consulting and customization services — at an additional charge, of course.

With version 5.0, Alienbrain is moving beyond simple asset management by incorporating tools to help manage and track the production process. As an art resource moves through an approval process, Alienbrain can attach a status to the file, indicating the current state of that file. For example, an artist can create a bitmap file with a status indicating that the file is still a "Work In Progress." Once completed, the file can be marked as "Awaiting Sign Off," after which time a lead artist can review the work and either approve it as final or bump it back to the artist for modification. Alienbrain can be customized to accommodate your company's specific development workflows, while an innovative color-coded reporting feature makes tracking and reporting progress a cinch. These features are likely to be a godsend for projects with complicated production processes.

Alienbrain offers a host of other features that I haven't the space to describe fully here. These run the gamut from an extensive reporting engine to an integrated instant messaging system to a secure, VPN-like feature that allows remote clients to access off-site Alienbrain databases safely and securely.

Make no mistake about it, Alienbrain is a large, sophisticated system that performs well and promises a tremendous amount of value to those customers brave enough to tackle it head-on — and to those customers with deep pockets. My biggest complaint about Alienbrain is its price tag. This is a powerful product that has much to offer, but a typical installation using the full-featured Power Client will cost approximately $2,000 a seat. You can purchase a less expensive, lower-fidelity Base Client for people like programmers, who don't need all of the Power Client's bells and whistles, but at the end of the day you're still looking at a significant expense for migrating an entire team to Alienbrain. On top of the high price, NXN Software has an utterly bizarre policy of not providing prospective customers with evaluation copies. In an industry that's used to paying thousands upon thousands of dollars for state-of-the-art 3D modeling tools, maybe some of the larger studios can roll the dice and justify an expense like this with little or no hands-on experience. Most companies, however, cannot. If Alienbrain cost $750 a seat, I have little doubt that it would become a de facto industry standard. As it is, it's a terrific product that will sadly remain out of reach for most of us.

ALIENBRAIN 5.0 ★★★★

STATS

NXN SOFTWARE
Sonnenstrasse 19
80331 Munich, Germany
+49 (0)89 27 32 240
www.nxn-software.com

PRICE
Pricing varies depending on the size and requirements of the project. Contact NXN for more information.

SYSTEM REQUIREMENTS
Server: Windows NT 4 SP 5/2000; 256MB RAM. Client: 128MB RAM; Windows 98/ME/NT 4 SP 5/2000, Mac OS 9.x or higher (with MacOS Runtime for Java), or Linux Server 4 SR 2 or later.

PROS

1. Flexible and effective main client user interface.
2. Customization options are legion.
3. Powerful project management and production tracking tools.

CONS

1. Price.
2. Geared more towards artists than programmers.
3. Servers are available under Windows only.

GIMPEL PC-LINT VERSION 8.0

by herb marselas

Gimpel PC-lint 8.0 (Lint) is arguably one of the most powerful, and yet daunting, tools for C/C++ programmers. The power of Lint lies in the exhaustive number of warnings, errors, and informational messages that its static code analysis generates, informing you of virtually every potential peril and pitfall waiting in your code. The messages generated by Lint range from strict C/C++ error information to coding recommendations from Dan Saks and Scott Meyers (the author of *Effective* C++). The latest version of Lint doesn't disappoint, adding nearly 90 new messages and 40 enhanced messages and options to the range of information supplied.

The problem with all this power is that it can lead some programmers to dismiss Lint's bountiful and seemingly extraneous messages. This complaint is not unjustified. Although our own code base at Ensemble compiles without a problem on our compiler's highest warning level, a fresh install of Lint generated a 56,000-line message file when run on just one of our files. Within a few hours, I was able to identify and disable all of the extraneous messages and get down to a meaningful set of potential problems that needed to be examined and addressed.

Beyond its power of static code analysis, Lint's appeal lies in the broad range of platforms and compilers that it supports. For Windows users, Lint is released as an executable. But for other platforms, Lint is released as a shrouded, or obfuscated, set of source files. Lint also doesn't favor one compiler or library over another, supporting configurations for more than 30 compilers and many common programming libraries out of the box.

One of Lint's few compiler-specific features is its new support for Microsoft Visual C++ Developer Studio Project files (.DSP). Visual C++ users can now feed their .DSP file directly into Lint, eliminating the need to copy the preprocessor settings from the Visual C++ IDE into Lint's configuration files. However, Visual C++ users must still manually add other settings, such as the include directories, from the Tools Directories menu to the Lint configuration files. As long as Lint already supports the Visual C++ project files, Gimpel should have gone the extra step and included the ability to read additional configuration information directly from the Microsoft Windows registry.

Perhaps the most important new feature in Lint 8.0 is the interactive value tracking. When it's enabled, interactive value tracking attempts to verify and track values passed between functions by making up to six passes over the code being analyzed. This function can help find value-out-of-range conditions or other situations in which parameter passing could cause problems. While I was able to see this functionality at work with the sample programs that Gimpel supplies with the Lint package, in practice I was unable to find any issues of this type using a sampling of files from Ensemble's large source code base.

Another change with this version is that the documentation is now completely online in Adobe Acrobat format. The documentation is relatively complete, though it could include more examples illustrating the messages that Lint generates. NuMega's BoundsChecker, for instance, includes a small code example for each of its errors; a similar glossary could only enhance Lint's usability.

The only major problem that I encountered while testing Lint 8.0 was a crash bug that surfaced while parsing some of our template code. However, Gimpel's technical support staff was able to get us a workaround after just a few days and a half-dozen e-mails. Unfortunately, we had to wait almost a month for a patch to fix the crash, so in the meantime we had to modify our code whenever we wanted to run Lint.

I also found that Lint doesn't correctly add itself to the path during installation if you're running Microsoft Windows NT, 2000, or XP. This seems like a glaring oversight, as Gimpel has added other Windows NT/2000/XP features, such the ability to reduce the priority of the process if you want to run Lint as a background task.

If you don't have Lint, get it today. If you already have the last major version, carefully evaluate whether these new features are enough to justify the upgrade. Regardless of your choice in compiler and platform, using Lint can only improve the quality of your code if you take the time to configure it properly and use it regularly.

★★★★½ | PC-lint 8.0 | Gimpel Software
www.gimpel.com

Herb Marselas is a programmer at Ensemble Studios.

EFFECTIVE STL: 50 SPECIFIC WAYS TO IMPROVE YOUR USE OF THE STANDARD TEMPLATE LIBRARY

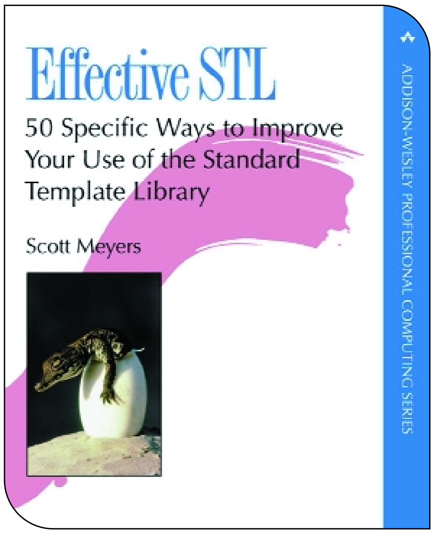
BY SCOTT MEYERS

reviewed by noel llopis

Have you read *Effective C++*, also by Scott Meyers? No? Go buy it right now, read it, reread it, and then come back here. I guarantee that it will make a huge difference in the way you work.

O.K., welcome back. Here's the good news: *Effective STL* is to STL what *Effective C++* is to C++. Meyers's latest book is equally great and is written in the same light, conversational tone that made reading his earlier work such a pleasure. It assumes that the reader has a basic knowledge of STL (and C++) and builds on that knowledge to show how to use it effectively and avoid common pitfalls.

The Standard Template Library (STL) is made up of generic data structures and algorithms that can be used across many compilers and platforms, including most current consoles. The library is implemented with templates, so the resulting code is quite efficient, possibly even more so than hand-coded structures and algorithms. Additionally, STL has been implemented, tested, debugged, and optimized by thousands of people, so it's generally reliable code. STL also empowers its users by putting powerful constructs at their fingertips. Previously, a common approach was to throw a bunch of objects into an array and search through them in linear time (does that sound familiar?). STL lets you put these objects into a hash table and thus have constant-time access to them.


Effective STL
50 Specific Ways to Improve Your Use of the Standard Template Library
Scott Meyers
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Effective STL begins with a fairly thorough discussion of containers (vector, list, map, and so on) and iterators. Meyers immediately goes beyond a typical description of the containers and their O(n) performance characteristics by addressing important, real-world questions: Is the memory for the elements allocated contiguously? Are the iterators invalidated when the elements change? What is the most efficient way of removing elements for a specific container?

Meyers then moves on to algorithms and functors. Just as with his discussion of containers, rather than simply listing all the available algorithms, he points out common mistakes and how to deal with them. Meyers reveals, for example, different ways of sorting elements, or how `std::remove` really works (and why it doesn't really remove anything).

The final chapters present more general, but still very useful, information on the STL. Meyers discusses when to use STL algorithms and when to use your own, style guidelines, or even how to deal with the broken STL implementation and template support in Microsoft's Visual C++.

But STL isn't perfect. Readers will gain some ideas of where STL is lacking, but these issues aren't spelled out explicitly. Sometimes you'll need to read between the lines and consider how STL applies to game development. For example, memory allocation can be an issue, especially for those doing console development; game developers will probably want to write their own allocators. *Effective STL* has a brief discussion of allocators but doesn't provide enough information to guide developers who need to write their own.

Another of STL's oft-cited shortcomings is the difficulty debugging STL code. Developers working with STL often face cryptic multi-line error messages and experience difficulty viewing the elements of a container in the debugger. *Effective STL* shows readers how to parse the intimidating error messages and includes pointers to STL resources on the Internet.

Effective STL works well both as a book to read cover to cover and as a reference later on. Meyers only uses source code where it's necessary; he doesn't bore his readers with pages and pages of pointless code. As a matter of fact, this book won't bore you at all, since it packs a lot of information into a mere 250 pages. Overall, you simply must read *Effective STL* before you decide to use (or not to use) STL in your next game or tools. If you're currently using STL, then this book should already be on your bookshelf.

★★★★½ | *Effective STL*
Addison Wesley | www.aw.com

Noel Llopis a software engineer at Meyer/Glass Interactive, where he specializes in 3D graphics.

Return

Jay Stelly, Technically Speaking

Valve's Jay Stelly.

Jay Stelly is a senior engineer at Valve. He was on the HALF-LIFE team, and he's leading the engineers developing the technology for Valve's next generation of games. We recently caught up with Jay to discuss his take on the challenges of balancing the roles of programmer, designer, and manager.

Game Developer. How much of your time do you spend managing versus programming? Do you try to stay out of the critical path on your own code?

Jay Stelly. The management demands change as the project goes on, but most of the time I spend about one to two days per week doing that. I'm still on the critical path, but my schedule does reflect having fewer hours to spend on coding. I still love to write code. I have to be on the critical path to enjoy my job!

I'll get a few hours each day to code without distractions. I'll shut off e-mail and the phone and just focus on what I'm coding. For some of the other time, I'm still coding, but I'll be interrupted often with various meetings or issues. Obviously, there are some tasks I can only get done with the proper amount of focus — so it takes a little extra planning as well. But regardless of process, most of the management comes back to hiring. I think that having the right people is a bigger part of the problem than how you organize those people.

We try to hire great people and stay out of their way as much as possible. We don't have any hierarchy in our engineering group. Everyone is responsible for a product; there are no technology-only developers at Valve.

GD. You have the legacy HALF-LIFE code, licensed physics, and a large team of programmers doing custom technology. How do you integrate all this, and is this heterogeneity the future of high-end games?

JS. Games have already reached this level of complexity. Many developers are reusing code now; I'd say it's the present as well as the future. Gamers are asking us to push the bleeding edge but not take any steps back along the way. Our code base has become more and more component oriented as it has grown. We've got clear interfaces between components, and really tight communication about the overall architecture.

GD. With long ship cycles, how do you make sure components that are developed early on are still relevant to the game that actually ships?

JS. This has been a really interesting shift. At the high level, our overall technology goals don't change much over the course of a project. But underneath, we're incredibly iterative. We constantly bring in new bits of technology or upgrade existing code. We certainly aren't just implementing something that was completely designed up front. We're attempting too many things that we didn't know quite how to do when we started. So, we take small steps, and keep the engine running and stable at all times. When big new features come online, there is usually some loss in stability, but it's temporary. We do throw away some code, but I believe that building the code we throw away is a necessary step to getting the final code just right. Having things split out into components makes it much easier to manage the iteration. Much of the change happens behind interfaces, without large effects on the rest of the system. We've got enough automation built in to allow us to keep the content up to date as we make changes.

GD. What's the interaction between game design and engineering at Valve? How do you keep the game design iterative and flexible without constantly churning and rewriting code?

JS. This is the hard part, isn't it? In the beginning of development, we got a bunch of people together across disciplines to make the high-level game design and technology plan. Most of the new technology was driven directly by our game design. But once we started to implement that technology and build the game, we had to deal with a bunch of challenges. Probably the biggest cost when adding new technology is the amount of time it takes for the artists and designers to get comfortable with it. At that point, they have lots of really good ideas for improvements and further development. So we try to bring those guys into the process as early as possible. Each new technology has at least one artist or designer acting as an advisor and tester far in advance of the rest of the team.

This procedure lets us figure out a bunch of the design problems earlier, and lets programmers fix these problems before they become overly difficult or costly. It also means that we can start working on second-order features sooner, because we've gotten that early feedback. It's kind of odd, but another part of the solution to this problem is to avoid being too flexible with the code. Our game designers come up with incredibly creative solutions to problems when faced with technological constraints. We're not trying to simply remove these constraints from the design process wherever possible. Some of the best stuff happens when designers are forced to think about how to create the game in the face of constraints. So it's important to be careful about choosing which constraints to change — and not to make these decisions without giving the design enough time to react. Having constraints be inflexible is really valuable. Lack of certainty about technology will paralyze a game design.

Return

Mipmapping, Part 2

Let's continue our quest for higher-quality mipmaps that we began with last month's column ("Mipmapping, Part 1," December 2001). Last month we saw that Kaiser filters produce output of a fairly nice quality. According to the tenets of signal processing, the more CPU we are willing to spend, the wider we can make our filters and the higher the output quality will be. But when we tried to make our filters very wide, we saw disturbing ripples in our output images, so we settled on a filter that was not very wide after all.

We suggested that these ripples were caused by oscillations in the frequency response of the filter. But the output ripples didn't go away even as the filters became ridiculously large and their frequency responses approached the ideal. This suggests that additional causes lurk behind the scenes. As it happens, identifying the guilty phenomena is important for our understanding of much of computer graphics, not just mipmapping.

Ditching Filters

Wide Kaiser filters are just approximations to an infinitely wide sinc pulse, which we said would remove the high frequencies from our image. Sinc does what we want because its Fourier transform is a rectangular pulse; when applied to the Fourier transform of our signal, this pulse multiplies the low frequencies by 1 and the high frequencies by 0.

In applications such as live audio processing, you want to respond to a signal coming over a wire with low latency; for these cases, the filtering form of antialiasing is well suited. But because we are mipmapping as a batch process, we have the option of just performing the Fourier transform on our texture map, manually writing zeroes to the high frequencies, and then transforming it back.

Mathematically, this is the same as filtering with an infinitely wide sinc. Since we are perfectly enacting the rectangular pulse frequency response, we won't have any problem with a finite filter causing frequency response ripples. Therefore our output will be perfect, or so we might expect.

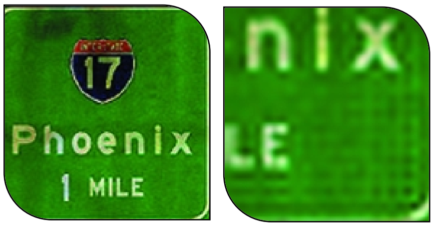


FIGURE 1A (left). Original image of a road sign. FIGURE 1B (right). A close-up of the corner after mipmapping via the discrete Fourier transform. Note the unwanted bands of color ("ringing").

But our output isn't perfect. The texture map still has horrible ripples in it, as you can see in Figure 1. It looks much like the output of last month's Figure 4, which was processed with a 64-sample Kaiser filter.

Life isn't simple, so there are multiple causes for these ripples. We'll look at the two most important causes. First, though, I'll mention that this month's sample code (which can be downloaded from www.gdmag.com) implements the Fourier transform method of mipmapping. On some images, such as the human face in the Choir Communications billboard, the results are better than the Kaiser filter. But on some other images, the results are terrible. So what's going on?

Darn

The biggest problem is that our concept of a texture map is mathematically ill-defined. Texture maps are pictures meant to evoke in the viewer's mind the nature of the represented surface. We need to think about the mathematical consequences of the way in which we store texture maps and compare that math to our mental model of the corresponding surface. We will see that the two are in conflict.

Figures 2 and 3 show a simple texture: each pixel is the same shade of green, except for one spot that is white. We interpret this as a surface that is uniformly green everywhere except in that white spot. Ideally, all the math we perform on the texture should be consistent with this mental interpretation, so that the processed image matches our expectations. Our mental model of the image surface is a continuous function, but the texture map is not continuous. It exists only as a series of samples, which is why Figure 3 is drawn the way it is.

Let's talk about sampling and reconstruction. When we sample an arbitrary (not band-limited) continuous function at n points, then those points are the only information we have. In between, there could be anything at all. So suppose we take a set of

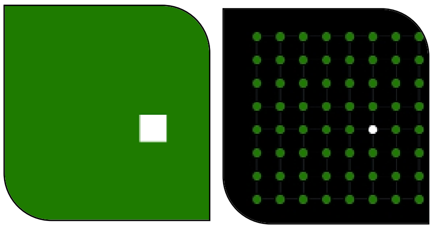
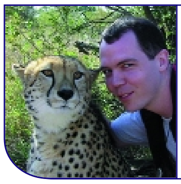


FIGURE 2 (left). A solid green texture containing a single white spot (pixels depicted as big squares. FIGURE 3 (right). The same texture as Figure 2, drawn differently to emphasize the point-sampled nature of the data.


JONATHAN BLOW I *Jonathan Blow is a game technology consultant living in San Francisco. His e-mail is jon@bolt-action.com. Game that influenced this article: Konami's POLICE 911, arcade version. That game rocks!*

samples and try to reconstruct a continuous function. We need to interpolate all the values in between, but because we threw away most of the original data, no interpolation algorithm will get us back to where we started. We just have no idea what was supposed to be between those samples.

Here's where digital filtering comes in.

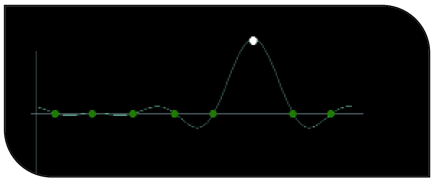


FIGURE 4. A one-dimensional cross-section of the texture from Figure 2, viewed from the side; the cyan curve shows the band-limited continuous function represented by these samples.

The Shannon Sampling Theorem is a cornerstone of information theory. It says that if you restrict your input functions to contain only a certain range of sinusoid frequencies (that is, the function is band-limited), then you can reconstruct the continuous function exactly from the samples. That's a powerful idea.

But there's a consequence to that idea. When reconstructing, we're not allowed to dictate what the function does in between the sample points; we have to take what we're given. Because the continuous function must be band-limited, it will oscillate between samples in ways that we may not expect. Figure 4 shows a one-dimensional cross-section of our green-with-white-spot texture and the continuous function it represents when we use sinusoids as our reconstruction basis. Note that the continuous version is not flat in the green area, as we would imagine it. Instead, it ripples. It can't possibly be flat, because the band-limitation constraint means that the function is unable to ease down to a slope of zero.

As far as the math we are using is concerned, here's what happened: To generate our texture, we started with some band-limited function with ripples all over the place, but when we sampled it, our sample points just happened to hit the right spots on the ripples to generate constant intensity values. Because the samples represent the entire continuous function, the ripples are still there, and our signal processing operations will reproduce them faithfully. When we shrink the texture to build mipmaps, we are essentially zooming out our view of the continuous function. Visible ripples appear because the change of scale and the added low-pass filtering disrupt the delicate coincidence of our sample point positions.

If we enlarge the source texture instead of shrinking it, we will see the ripples at our new sample points. As we move to higher and higher resolutions, we converge toward the continuous function in the limit.

Last month I mentioned that Don Mitchell, who has helped with this column by discussing many of the ideas, has a filter named after him. The Mitchell filter resulted from experiments about enlarging images without making them look icky. It is intentionally imperfect from a signal reconstruction standpoint, because perfect reconstruction isn't very much like what our brains are thinking about. But the Mitchell Filter doesn't stray too far from the perfect upsampling either, because then nasty artifacts would appear.

In the same way that the Mitchell filter is not a perfect resolution-increasing filter, our mipmap filters, intended to decrease resolution, look best when they hover in a sweet spot between badness and goodness.

Do We Really Want to Antialias?

There is another important point here. The band-limited continuous-function representation, with all the unacceptable ripples, is a prerequisite to digital sampling. That is, if you walk up to a real-life green wall with a white spot on it, and you wish to create a texture map from it, then to prevent aliasing you must low-pass the incoming image before digitizing it. This filtering produces the unacceptable rippled function, and the ripples will show up in your samples. Thus, digital cameras don't try to antialias this way, and they don't collect point samples to use as the photographed image. If we then treat the image as a collection of band-limited point samples, a practice that seems to be in vogue, then we introduce ripples that weren't there in the natural lightfield.

Antialiasing causes the ripples. We don't want the ripples. Therefore, antialiasing is not really what we want.

What we actually want is for all of our graphics to have infinite resolution, so that we can draw surfaces at arbitrary scales and orientations with impunity. Unfortunately, we don't yet have the means to do that, so we're using this digitally sampled image stuff instead. Antialiasing is an important tool that helps us cope with the constraints imposed by digital sampling. But it also causes some problems, so it should not be visualized as a goal in itself.

Antialiasing is usually touted as something that only fixes problems, that has no downside. That's because an awful lot of people in computer graphics (and game programming) work from day to day using a set of concepts that they don't understand down to the core. They're just repeating what they heard from someone else, and what we get is much like the game of telephone, where we whisper in each other's ears, and after a few transmissions the message "I biked with Pete on Friday" becomes "I'd like to eat a fried egg."

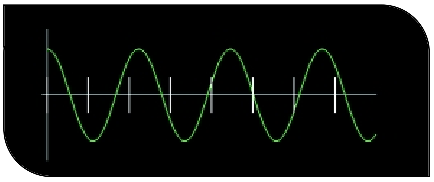
That's not to say that anyone should be blamed — it's hard to understand all this stuff. I didn't understand it until writing this article. But there you have it: Antialiasing is not really what we want. We want elimination of objectionable artifacts, which is a different thing. Antialiasing can help in that regard, but it's not a total solution, and it can hurt too.

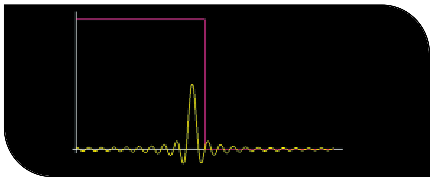
And that's assuming that we actually know how to antialias — which we don't. Here's why.

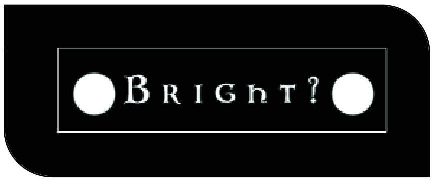
Questionable Interpretations

The Fourier transform is all about treating your data as a big vector and projecting it onto a set of (complex-valued) sinusoids that serve as basis functions. Each sinusoid is of a different frequency. The length of the vector in the direction of each basis function tells us something about the frequency content of the signal for that given frequency.

But what it tells us is not the actual frequency content. If you have a single cosine wave of the same frequency as a basis function and you push it through the Fourier transform, you'll get what you'd expect: a single spike in the output at the appropriate frequency. But if that cosine wave has a frequency somewhere in between the basis







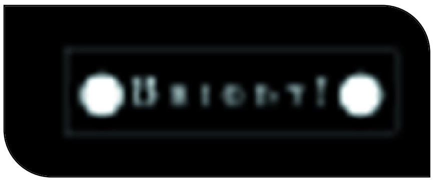


FIGURE 5 (top left). A low-frequency cosine wave (green) that could be sampled easily at the intervals marked in white. FIGURE 6 (top right). The magnitude of the frequency content of Figure 5 (graphed in yellow), as reported by the Fourier transform. It is the sum of two sinc pulses. The red step function depicts the cutoff pulse of an ideal low-pass filter. FIGURE 7 (bottom left). A high-contrast texture at full resolution. FIGURE 8 (bottom right). The high-contrast texture reduced by three mipmap levels (box filter). The text has become somewhat dim, and the outlining rectangle has become extremely dim. This is energy loss due to the gamma ramp.

functions, you don't get a spike anymore. You get a smear of frequencies across the entire spectrum. Figures 5 and 6 illustrate this. The peak of the Fourier-transformed data is in the right place, but what is going on with all that other stuff?

Even though each Fourier basis function consists of a single frequency, in the end they're still just basis functions. If you ascribe any interpretation to the transformed data, aside from "the inner product of my input data with each basis function," then you do so at your own risk. "Frequency content of the signal" is just an interpretation, and it's a slightly misleading one.

When we low-pass filter in order to antialias, we are acting on this misleading interpretation. Consider that the cosine in Figure 5 could be downsampled into a mipmap just fine, as it stands. But when we filter it, we cut off all the high frequencies to the right of the red line in Figure 6. These frequencies aren't really there; when we subtract them from the cosine wave, the shape changes. We were shooting at ghosts, so we hit ourselves in the foot. This is important: some kind of imaginary true antialiasing would have left the cosine unscathed; but all we know is how to eliminate aliasing as the Fourier transform sees it through its own peculiar tunnel vision. This causes problems.

If we can't even antialias a dang cosine wave without messing it up, then what can we antialias?

Moving On

We've looked at some difficult issues underlying not only mipmapping but many other areas of computer graphics as well (pixels on the screen are, after all, samples). It's important for engine programmers to understand these issues. But as of today there's no magic bullet for them. I can't just give you code that magically shrinks your textures while simultaneously preventing aliasing, bluring, and rippling. Maybe in the future we'll have a new paradigm to help sort out these problems. Maybe, if we're vigilant and our aim is true, the community of game engine scientists will develop the new concepts and techniques. Who can say?

But this is an action-oriented column, so I'm obliged to provide some news you can use today. The good news is that there's something we can do, having nothing to do with choice of filter, that improves the quality of our mipmaps.

Conservation of Energy

An important goal of mipmap filters is that they don't add or subtract energy from an image; mipmap levels of a single texture should all be equally bright. This is equivalent to saying that as you sweep a filter across a given sample, the sum of all that sample's contributions does not exceed the sample's original magnitude. Or:

$$\sum_i f_i s = s$$

where f_i are the filter coefficients and s is the sample value. This implies

$$\sum_i f_i = 1$$

Thus, the sum of the filter's coefficients is 1.

We have a problem, though. We want the same number of photons per unit area to shoot out of the user's monitor, no matter which mipmap level we're displaying. But that doesn't happen, even though our filter's coefficients sum to 1. This is because we tend to do our image processing in a color space ramped by the monitor's gamma, which messes up the energy conservation properties.

Figures 7 and 8 demonstrate this. Figure 7 is a high-contrast texture at full resolution. Figure 8 is a lower-resolution mipmap constructed in the naive way; the areas containing fine features have gotten significantly dimmer.

Here's a quick gamma recap: Unless we do weird tweaking with our graphics card's color lookup-tables, the amount of light coming out of a CRT is not proportional to the frame buffer value p. It is proportional to $p^{s\gamma}$, where $s\gamma$ (gamma) is a device-dependent value typically above 2. Our eyes interpret light energy logarithmi-

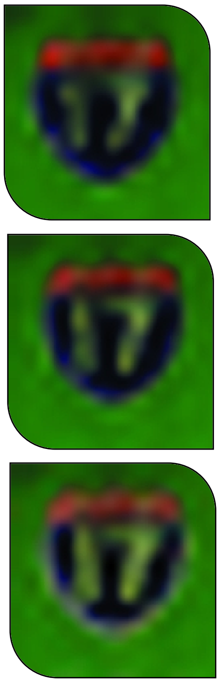


FIGURE 9A (top). A highway shield mipmapped using a typical game's box filter. FIGURE 9B (center). A Kaiser filter does a better job of preserving the shapes of the numbers and the sign border. FIGURE 9C (bottom). Moving the Kaiser filtering into light-linear space, we maintain the contrast between the numbers and the background, and we preserve the white border around the edge of the shield.

cally, so this exponential outpouring of energy looks somewhat linear by the time it gets to our brains.

We store all our texture maps in a nonlinear way — they expect the CRT to exponentiate them in order to look right. This serves as a compression mechanism. We would need more than 8 bits per channel if each channel held values proportional to light energy.

Suppose we pass a simple box filter, coefficients [.5, .5], across a sample of magnitude *s*. We get 2 contributions to the image, of magnitude .5*s* and .5*s*. That adds up to *s* again — so far so good. Now the CRT raises these values to the power gamma. Now we have two adjacent pixels of magnitude ($.5^{\gamma} s^{\gamma}$). For simplicity, suppose gamma is 2. Then the total amount of light energy is ($.25\ s^2 + .25\ s^2 = .5s^2$) . . . but this is only half as bright as it should be (the unfiltered pixel would have brightness s^2). Our image got dimmer.

We can fix this by filtering in a space where pixel values are proportional to light energy. We convert our texture into this space by raising each pixel to the power gamma. Then we pass our filter over the texture, ensuring conservation of energy. We then raise each pixel to the $1/\gamma$ to get back to the ramped space so we can write the texture into the frame buffer. (The CRT will raise each pixel to the gamma again during output.)

It would be cleaner to set up the frame buffer so that all values stored in it are linear with light energy; the RAMDAC would perform any necessary exponentiation. High-end film and scientific rendering all happens in linear-light; game rendering will switch to this paradigm soon. When the frame buffer is linear-light, you can correctly add radiance to a surface just by adding pixel values in the frame buffer. (Right now, when we shine multiple lights on a surface, we just add the values together, but the gamma ramp makes that wrong. This is one reason why lighting in PC games is dull and limp.)

This month's sample code implements mipmapping in linear-light space, and you can see the results on a game texture in Figure 9.

ACKNOWLEDGEMENTS

The "Phoenix 1 Mile" texture in Figure 4 is by Dethtex, see http://users.nac.net/schwenz.

Thanks to Don Mitchell, Sean Barrett, and Jay Stelly for fruitful discussion.

FOR MORE INFORMATION

Blinn, Jim. *Jim Blinn's Corner: Dirty Pixels*. Morgan Kaufmann, 1998.

Hamming, R.W. *Digital Filters*. Dover Publications, 1998.

Poynton, Charles. *Digital Video and HDTV Algorithms and Interfaces*. Morgan Kaufmann, 2001.

Building the Future, Part 1: Architecture

Why is it that we play videogames? Is it to learn something about ourselves as human beings? Is it to hone our social skills and make us more attractive to the opposite sex? Do videogames strengthen the fabric of society? Do we find answers to life's most profound questions when we plug in our consoles and fire up our joypads? Or is it just entertainment? Call me shallow if you like, but the game industry is, by its very definition, in the business of entertaining.

O.K., some games come along that give us insight into the microeconomics involved in running a chain of pizza restaurants or simulate what it's like piloting an Apache helicopter. But we still play them for fun. Games that attempt to be about real life, like their TV counterparts, run the risk of becoming tedious, and once you remove the element of voyeurism (which is unlikely to play a large part in a game, anyway), you can easily be left with nothing more than the mundane. Which is, of course, no fun at all.

With this in mind, it's easy to see why such a large percentage of games are set in a fantasy world that doesn't actually exist. The ability to design a game that takes players beyond the things they can experience in their real lives, as well as the creative freedom that this affords, is hard for game developers to resist.

As with film, the choice of fantasy settings often splits broadly into either a sword-and-sorcery troll-bashing dragon fest, with pointy weapons and more than its fair share of beards, or the sci-fi staple that is the world of the future.

In this column, I will take the second of these and look at some ways in which we, as artists and designers, can put together an environment that successfully gives players the feeling that they are in the future, with specific reference to the architecture.

Buildings Are Not Fun

What makes a game's visuals exciting? Insane particle effects when you fire your plasma cannon? Exquisitely detailed zombies tripping over their own intestines? Beautifully animated female ninjas with real-time physics in all the right places? The answer to all of these questions is, of course, yes, but as sexy as the central characters and their effects may be, without a high-quality environment to give the game its context and location, the atmosphere and overall feel of the game will suffer.

Architecture is an indication of the time period in which the game is set, but more than that, it can be both a backdrop and the stage on which the action takes place. As game environments become increasingly interactive, and as we are constantly being allowed to use greater detail as well as a wider variety of visual tricks in our worlds, creating the architecture, exteriors and interiors, is no longer the thankless task it once was. The days when the lowliest of all art monkeys was assigned the job of making "some buildings for the background" are long gone, and the quality of our architecture is now a strong contributor to a game's visual impact.

To the DeLorean, Marty...

Having established that architecture within a game is important, and that this is particularly the case when designing the future, we now need to decide what future we are going to be dealing with.

Perhaps the first and most obvious question is how far into the future we're going. Near-future games, like near-future films, are based in a world that we know and recognize, albeit with a few, often superficial, changes. Travel farther into the future and the restrictions of our current surroundings begin to evaporate. Extrapolation gives way to hypothesis and the possibilities become more varied.

Even when avoiding extremes like the diaper-wearing future world of Sean Connery's *Zardoz* or that of H. G. Wells' hairy, subterranean Morlocks in *The Time Machine,* the most common future archetypes are that of utopian and dystopian civilizations (Figure 1). These reflect the positive and negative possibilities we imagine as potential futures, and are polar opposites of what will, for us, likely turn out to be a combination of both.

The utopian ideal often describes a place where technology has been used to produce a world of beauty, peace, equality, and more often than not, a whole lot of white molded plastic. In the utopian future, the designer is concerned primarily with aesthetics, and

HAYDEN DUVALL | *Hayden started work in 1987, creating airbrushed artwork for the games industry. Over the next eight years, Hayden continued as a freelance artist and lectured in psychology at Perth College in Scotland. Hayden now lives in Bristol, England, with his wife, Leah, and their four children, where he is lead artist at Confounding Factor.*

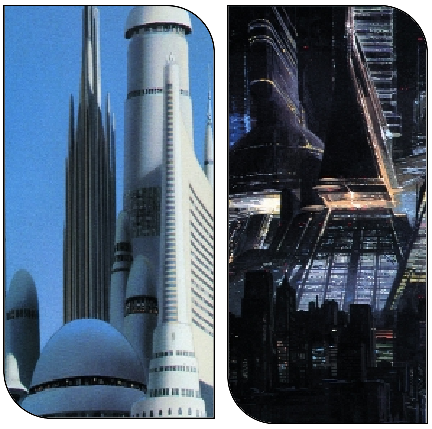


FIGURE 1A (left). Utopia. FIGURE 1B (right). Dystopia.

architecture becomes less about function and more about form. Advances in science have allowed the city of the future to become a clean, bright, happy place, where the harmony of human achievement is reflected in the surroundings.

Dystopia, however, is a little bit darker and a whole lot dirtier. Rutger Hauer may have seen it as an ideal holiday destination in *Blade Runner*, but the dystopian city of the future takes the worst of society's ills and straps on decades of industrial madness.

Both versions of the future present the game artist with a variety of challenges. As always, the easiest environment to deal with in a game is a small, interior space. Once we venture outside, things tend to get larger, and the areas that we need to fill with meaningful visuals get larger also. Whether organic or man-made, large spaces are both labor intensive and resource hungry, and solutions involving complex systems of LOD, texture streaming, and clever layout can all help ease the pain.

Although our game worlds often combine elements of both utopian and dystopian futures, as far as the creation process for the game artist goes, it's useful to break it down into the three areas of surface, structure, and scale.

Surface: Shiny Happy People and Urban Decay

In this context, surface refers to the materials that we are attempting to portray, their properties, and how we convey these within the framework of our engine and platform limitations.

When looking at large-scale city buildings, the move from brick to concrete and then toward shiny surfaces is clearly visible. Highly reflective materials became associated with high-tech, and as the 1970s saw the gratuitous proliferation of mirrored sunglasses, so the 1980s began in earnest to produce huge, monolithic mirrors at the centers of our largest cities. The portrayal of future cities of chrome and glass has long been a science fiction staple, but besides being a slightly dated approach, real-time environment mapping and refraction within a world that is packed with shiny, semitransparent surfaces will have any programmer bleeding from his or her ears in no time at all. Despite huge advances of late in areas such as hardware transformation and lighting, reflective surfaces on a large scale within a game are still not practical.

In a similar vein, one indication that we are indeed dealing with the architecture of the future is the presence of exotic or out-of-the-ordinary materials. Examples of this have been evident for some time in prerendered backgrounds or FMV sequences, and with the graphical grunt behind today's gaming platforms, the game artist can now achieve some of these effects in real time, without the hardware they're working with bursting into flames under the strain.

Taking materials out of context and building with them can be used to suggest a futuristic world, and using the detail of organic surfaces such as tree bark or coral can be a good starting point. Combining these kinds of natural materials with some element of human design can produce interesting results. In graphics, the mathematical precision of computer-generated features and geometry is usually a problem that detracts from the realism of a scene. Our eyes are extremely sensitive to this kind of perfection, as the real world is generally full of flaws and shapes that are much less exact. We can, however, use this regularity and order to place features within an organic texture to show that it is being manipulated by humans and used as a material for construction.

The simple addition of fabrication joins, rivets, bevels, and the like can be all that's needed to give a surface the extra detail necessary. This combination of the natural and the manufactured can provide some unique surface qualities (Figure 2).

Once we begin to look toward a dirtier future, the kind of surfaces we are dealing with become more recognizable to us. It's nigh on impossible to talk about a dystopian future city without reference to *Blade Runner* (see, I've already done it twice). But without lapsing into spasms of worship at the alter of Syd Mead, his vision of a future city has had a vast impact on visual representations of our future in films and games alike.

Using the dirtiest, most unkempt pieces of a contemporary city as a starting point, and extrapolating out to a seething mass of giant, twisted structures, decaying, part derelict, the surfaces we begin to encounter would be at home in an abandoned chemical plant. Taking materials and dirtying them down is certainly part of the way toward the look we're after, but another important element is that of making the underlying technology visible.

In a dystopian future, the aesthetics of design are crushed under the weight of shoddy workmanship and dereliction. Instead of a single, coherent whole, in which all elements are combined in harmony, things are grafted together, creating

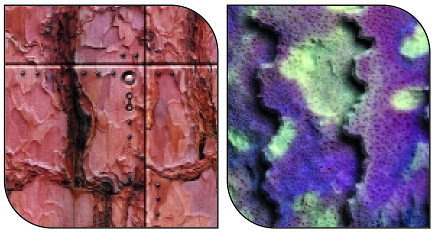


FIGURE 2A (left). Utopian texture 1. FIGURE 2B (right). Utopian texture 2.

ugly hybrids. What are usually hidden workings break through to the surface, with pipes, ducts, and wiring all becoming exposed (Figure 3). Architect Renzo Piano may have won huge acclaim for designing the Pompidou Center in much the same way, but combine this externalization with dilapidation and a good coating of dirt, and the result is somewhat less attractive.

When dealing with the surface (in other words, textures), it is often useful to add mechanical detailing, or to expose circuitry. High-magnification images of micro-

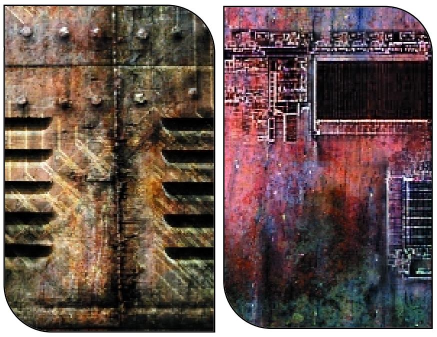


FIGURE 3A (left). Dystopian texture 1. FIGURE 3B (right). Dystopian texture 2.

processors can be extremely useful as a starting point for adding the precision-machined look to a surface. "Greeble" plug-ins are also useful if you wish to render out an image to use in texture generation, as these can take most of the work out of generating a large amount of detailed geometry.

One additional aspect not to be overlooked when considering surfaces in a game is the way in which they are encountered by the player. Detail in a texture is, of course, wasted if the player is never going to be near enough to appreciate it, especially when dealing with potentially large-scale objects such as buildings. As always, a sensible allocation of resources will achieve the best results.

Structure: The Shape of Things to Come

Buildings have historically been fairly square. Maximizing usable space, and minimizing the complexity of the physics involved has generally led to boxlike structures. Even when departing from a cuboid, straight lines remain an important feature.

Seeing this as somewhat of a challenge, architects of recent times have availed themselves of vastly improved technology, as well as huge advances in the understanding of the science involved in construction, to create buildings that are anything but angular. Some err on the side of crazy (the Guggenheim Museum in Spain, for example). The trend, however, has been to challenge convention, and in the city of the future, we are free to break all the rules.

The battle of the curve, struggling to overcome its idiot cousin, the straight line, has featured throughout most areas of industrial design. In architecture, especially when dealing with large structures, the restraints are somewhat more stringent than when dealing with a vacuum cleaner, for example. But as we probe the future for possibilities, one potential direction to explore is a move toward organic forms.

A simple indicator that the player is roaming around in the future is the presence of structures that would not be possible to build in the present (Figure 4). It is also vital to remember that any structures must function within the context of the game in which they exist. Gameplay considerations and level design have to take precedence over pure aesthetics, and successfully combining the functional with the beautiful is the ideal.

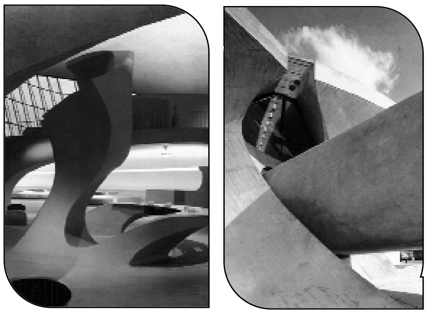


FIGURE 4A (left). TWA inside. FIGURE 4B (right). TWA outside.

Scale: How Big Is Yours?

One common view of the future city is that it will somehow have to accommodate vast numbers of people, whether for work or simply as a place to live. This idea of expansion in all directions is already in evidence in places such as Tokyo, where severe limitations of available land have caused the city to expand vertically and also to compact more into less space.

This high-density living has, on occasion, been used to show that the post-apocalyptic landscape is no longer inhabitable (Mega City One in *Judge Dredd* for example), but it can also simply be a caricatured version of the overcrowding seen today in most of the world's major cities. Whatever the underlying rationale, architectural scale is a useful component of the future landscape.

The well-worn science-fiction idea of the huge, all-powerful corporation that runs everything from the police to the burger-matic fast food vending machines is not that hard to imagine (I think we can already name a few contenders), and cities of the future like to base these corporate monsters in buildings scaled to reflect their immensity.

Specifically, the height of a structure has always shown man's defiance of natural laws and his ever-growing command of the elements. In the future, we can theorize that our architecture will doubtless take advantage of incredible scientific breakthroughs, and the scale of our largest buildings will increase accordingly (Figure 5).

When building a game environment, an overindulgence in large-scale architecture can, however, be counterproductive. Unlike the real world, our perception of scale within a game is almost completely relative, and if too much of a character's environment is built on a massively large scale, we run the risk of creating the impression that it is actually the character that is in miniature.

Constructing the Future

Successfully creating a game world set in the future obviously depends on a whole range of contributing factors. Architecture is certainly one element that we, as artists, can use to locate the player in the desired time period and create an atmosphere that enhances the experience. There is, of course, no correct way to construct

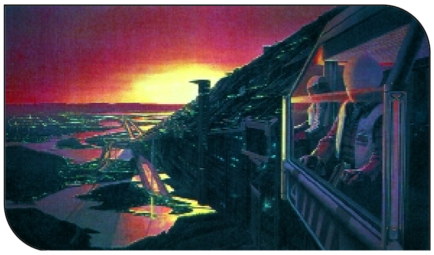


FIGURE 5. Megastructure.

the buildings of the future, as every game has its own particular demands and every world we build can be unique.

Despite this array of possibilities, we have the present to use as a starting point, and there is a wealth of visionary architects whose work already challenges convention and can give us a glimpse of what may be to come. Our task is to pin it down and make it work.

Return

2001 The Front Line Awards

It gives me great pleasure to introduce *Game Developer*'s 2001 Front Line Awards. In this, our fourth annual awards program, we've found that the best and brightest of the game industry's development tools mirror the evolution of the game industry itself. This year's crop of Front Line Award honorees tends to reflect efforts to formalize and codify, and in a way professionalize and legitimize, the practice of game development.

Indeed, the common thread that connects most of our winners is their utility in creating and enforcing standards, best practices, reusability, version control, and extensibility, all hallmarks of traditional, so-called corporate software development. Our industry has arrived in the mainstream, ladies and gentlemen, and the advent of products such as Alienbrain, Quantify, and the GeForce 3 are plain evidence of this professional maturation.

As always, *Game Developer* magazine owes a huge debt of gratitude to our Front Line Award judges. Our judges this year threw themselves into this project with aplomb, enduring a rigorous period of hands-on evaluation and open debate. Furthermore, since our judges were drawn from the ranks of working game developers, I'm confident that their concerns and final choices are relevant to the readers of this magazine. So without further ado, I would like to recognize the contributions of our 2001 Front Line Award judges.

Programming: Independent developer Scott Bilas, Chris Corry of LucasArts, Mark DeLoura of Sony Computer Entertainment America (and formerly of *Game Developer* magazine), Herb Marselas and Matt Pritchard of Ensemble Studios, and Michael Saladino of Presto Studios.

Art: Hayden Duvall of Confounding Factor, Spencer Lindsay of Etribe Studio, and Todd Siechen of RealEyz Imaging.

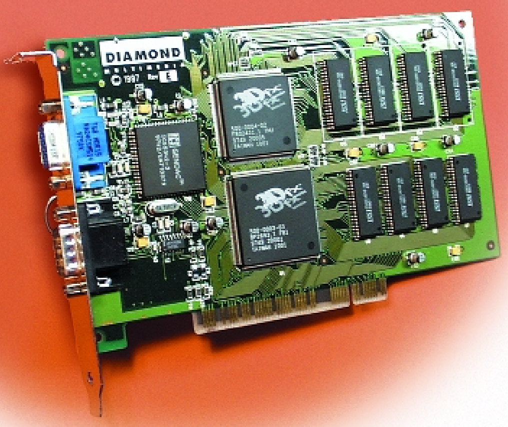
Illustration by Peter Ferguson

Audio: Chuck Carr of Sony Computer Entertainment America, Tom Hays of Novalogic, Aaron Marks of On Your Mark Music, Robb Mills of Stormfront Studios, independent audio developer Gene Porfido, and Rob Ross of Sound Endeavours.
Production: John Williamson of Zombie, Troy Dunniway of Microsoft, and Alex Dunne of Gamasutra.com (formerly of *Game Developer* and, in fact, the founder of the Front Line Awards).
Hardware: The hardware category was open to balloting by all of our judges in all categories, and the results reflect the disparate roles and opinions of our judges.

Finally, it seems appropriate to thank you, our readers, first for your help in nominating an initial list of over 90 remarkable tools, and second for your continued interest in the Front Line Awards and in *Game Developer* magazine.

—Tor Berg

HALL OF FAME AWARD



3DFX VOODOO CARD

3Dfx

The Front Line Hall of Fame is for products that have made a lasting impact on the direction of game development. All Front Line judges from all categories debate the significance of various historic development tools until they final agree on a single inductee.

The original Voodoo card, along with its killer app, GLQuake, caused game developers everywhere to stop in our collective tracks. Before Voodoo's arrival on the scene, we had never seen a video card targeted at the end user that elevated the performance of our games to something that was clearly more than the sum of the CPU, clock speed, and installed memory. So convincing was Voodoo that we added a new variable to the equations that made up our games: The Hardware 3D Accelerator. And Glide, 3Dfx's 3D API, ripped the covers open on this wonderful new device and laid its inner workings open for us to use and manipulate with a minimum of interference. And though there had always been some of us who had used 3D in our games, the presence of the Voodoo accelerator and the rewards that it gave was what push so many of us who had never before considered 3D to start the journey. For the next few years, a lifetime in the computer industry, 3Dfx evolved the Voodoo chip and stayed on top of the consumer 3D industry.

Though eventually overtaken by its competitors, Voodoo's impact was enough to change the very shape and direction of mainstream game development. And that is perhaps how we will remember it in the future: as a signpost on our journey, marking the spot where the heart of PC gaming changed direction from 2D to 3D.

—Matt Pritchard

3Dfx • www.3dfx.com

PROGRAMMING

The fastest way to compare and merge your files and folders
Araxis Merge 2001
for Windows 95/98/NT/2000
Araxis
Installation Guide

PERFORCE 2001.1

Perforce Software

Nearly all game companies are using version-control systems for code, and many are using such systems for game assets, which seem to grow exponentially in size each year. Perforce easily handles the huge files that are becoming common in today's production environments. It uses a client-server model built on top of TCP/IP, making syncs to large databases extremely fast. A defining feature of Perforce is its model for version control, the change list, which is a logical grouping of files that are changed for the same purpose. Change lists are submitted to the database automatically, receive the same comment, and have one entry in the history listing. This can be a powerful tool for organizing and tracking game development. Pending change lists are also viewable by the entire team, so it's easy to see what teammates are working on. Perforce supports integration with a variety of popular programming tools, and also comes with C++ and Perl APIs to enable automation or integration with custom tools. Ready to switch over right now? Perforce even has a tool to convert SourceSafe, RCS, CVS, and ClearCase databases.

—Scott Bilas

Perforce Software • www.perforce.com

ARAXIS MERGE 2001 v6.0 FOR WINDOWS

Araxis

Araxis Merge is a tool that you won't think you need. Use it once, however, and you'll question how you ever got any work done without it. Merge is hands down the most fully featured, easy to use, graphically intuitive file and directory differencing/merging utility available. Merge is what you get when you take a tool to its extreme, polish it to a high shine, and then add a few more features for good measure. Its differencing/merging interface is fast and easy, and is friendly to both mouse and keyboard users. Among its notable features are unlimited undo, in-place editing, and highlighting of changes within lines. For those times when you know some differences are safe, Merge provides the ability to ignore differences via a regular expression match. Its default GUI is two-way, but a unique three-way option enables two versions to be merged into a third. This feature is particularly useful for engine licensees needing to merge the latest code drop against their own local changes, as well as simultaneously differencing with the previous code drop. Araxis Merge is truly one of this year's most innovative and useful programming products.

—Scott Bilas

Araxis • www.araxis.com

FINALISTS

Game Programming Gems 2 by Mark DeLoura – Charles River Media; Perforce 2001.1 – Perforce Software; Visual Assist – Whole Tomato Software, Merge 2001 v6 – Araxis; Vtune Performance Analyzer 5 – Intel; Mathcad 2001 – Mathcad; *3D Game Engine Design: A Practical Approach*

RATIONAL QUANTIFY v2001A

Rational Software

Rational Quantify is a first-rate call graph profiler that should be in every programmer's toolbox. What really sets it apart from other profilers is its combination of power and simplicity in providing many angles of attack to optimizing CPU usage. Any programmer can pick up Quantify, do a few profile runs, and quickly find out where the game is slow and how to speed it up. Instrumenting an executable is hands-free, fairly speedy, and is done at the binary level, so there's no need to recompile. For analysis, Quantify provides four separate views into the profile databases — an innovative pie-chart-style view, a global table view, a call graph, and the inevitable source code view, complete with per-line profile stats. Navigating these views is simple and fast, which is vital to make sense of the millions of function calls that happen in just a few seconds of game execution. Multiple sessions can be loaded at once, and merged and differenced to provide new databases for further analysis. Quantify also exports several APIs, including one that lets the game control sampling sessions, which can be used to enable fully automated profiling. Few tools provide this much bang for the buck, and in such a polished, friendly package.

—Scott Bilas

Rational Software • www.rational.com

Xbox Development Kit – Microsoft; Codewright 6.6 – Starbase;
by David Eberly – Morgan Kaufmann; Quantify 2001– Rational

PRODUCTION

ALIENBRAIN 5

NXN Software

With Alienbrain, NXN Software has created an entirely new class of development tool. While past efforts to implement various flavors of asset management systems in game development have resulted in clever but quaint collections of scripts and other homegrown tools, NXN's Alienbrain is an extremely powerful version-control system straight out of the box, and only gets better as users uncover its extensibility and ease of customization. (See page 8 of this issue for a more detailed review of Alienbrain by FLA judge Chris Corry.)

NXN built Alienbrain to handle file types commonly associated with interactive software development. The interface is a wonder of GUI design, presenting a wealth of information about a file's status and history in a clear and well-organized manner. Alienbrain's functionality is well-integrated with many standard game development tools, such as 3DS Max, Photoshop, and Maya. And if Alienbrain doesn't already do what you want it to do, NXN's extremely eager support team will fix it so that it does.

Unfortunately, Alienbrain remains really pricey. With a product so heavily dependent on its human (and expensive) component, NXN must charge its customers a relatively high licensing fee.

This expense is going to stand in the way of smaller studios purchasing Alienbrain, which is a shame, because NXN has done an admirable job of trying to fill a gaping hole in game development production needs.

—Tor Berg

NXN Software • www.nxn-software.com

FINALISTS

Alienbrain – NXN; Project 2000 – Microsoft; ICQ 2000b – ICQ

ART

MAYA 4

Alias|Wavefront

Maya is something of a small miracle in 3D production. It remains true to its PowerAnimator heritage and really shines by bringing the tools to a polished and usable state for customization in the pipeline. The way Maya can be molded to fit any sort of environment, user, or task is ironically synonymous with its ability to allow the artists and programmers to craft fantastic imagery in short periods of time with minimal obstacles beyond their own imaginations. Maya often leaves you with a feeling of amazement at its ability to work in the way that you had always wished that a piece of software would work. There is no doubt that with future versions of Maya, we can expect further refinement of key areas such as subdivision surfaces and rendering.

Undoubtedly, May has a steep learning curve, but there is a very big payoff once that learning curve has been tackled. Molding Maya's interface to your own personal style will make you soon almost forget about the interface altogether and pay more attention to the craft itself.

—*Todd Siechen*

Alias|Wavefront • www.aliaswavefront.com

3DS MAX 4.2

Discreet

This latest incarnation of Discreet's 3DS Max is a product that really shows its maturity. In addition to a completely redesigned IK and bones toolset, the very Maya-esque quad-view toolbar is exceptionally useful when one is deep in a modeling or animation task. The interface is completely redefinable. Users can modify the positioning, colors, and keystrokes of the tool bars with MaxScript and Visual MaxScript.

Although in the past, 3DS Max has had a bad rap as a modeling package, the extensibility of the standard polygonal modeling tools via plug-ins and scripting has made this version a really great modeler. The subdivision surfaces (NURMS) are logical and easy to work with, and Patch Modeling is a breeze. The NURBS modeling portion of Max 4 is still not up to the level of Maya or Rhino, but it proves to be a very reliable modeler in its own right.

With the release of Character Studio 3.0 and the addition of many standard interactive controls such as the Wire Editor and manipulators, character animation is easier than ever.

I've been using Max since it was 3D Studio DOS beta 1.0, and this version is by far the best one yet.

—*Spencer Lindsay*

Discreet • www.discreet.com

FINALISTS

Maya 4 - Alias|Wavefront; Bones Pro 3 – Digimation; FaceGen – Digimation; 3DS Max 4 – Discreet; Deep Paint 3D – Right Hemisphere; Reactor – Discreet; Zbrush 1.23 – Pixologic; Vroom for Maya – Testarossa

DEEP PAINT 3D

Right Hemisphere

Excellent integration with Photoshop plug-ins allows digital artists to benefit from Deep Paint 3D's extensive customizable set of painting tools in conjunction with those they are most likely familiar with already. Effects available in the 2.5D mode give artists the ability to give work a false depth, allowing the easy creation an image with a hand-painted look (compatibility with Wacom Intuos tablets makes this a useful function).

In 3D mode, Deep Paint 3D works seamlessly with 3DS Max, Maya, and Softimage, allowing models to be painted directly, with control of color, bump, opacity, shine, and glow.

Documentation and online tutorials provide an excellent introduction to the package, and a friendly interface allows the artist good access to Deep Paint 3D's powerful features.

Although many of Deep Paint 3D's tools are not necessarily going to be useful to the average game artist, this powerful program can provide the user with more control over their 2D work, as well as some additional help when working in 3D.

—Hayden Duvall

Right Hemisphere • www.us.righthemisphere.com

AUDIO

CHANNELSTRIP

Metric Halo

ChannelStrip from Metric Halo is a Mac OS plug-in for MAS (MOTU Audio System, the format for Digital Performer) and Pro Tools that brings the look, feel, and sound of a high-end digital console to the computer. It includes a gate, compressor, and six-band fully configurable EQ, along with animated visual graphs of each parameter (just like the big boys).

Not only is ChannelStrip's design and concept right on the money, the audio quality is exceptional. ChannelStrip's EQ has been able to capture that sound for which many of the classic recording consoles and elite outboard boxes (all hardware) are famous. The high-end EQ is notably as clean and clear as you could imagine, thanks to 64-bit processing, and adds an open air to music and voice-overs that's magic. The EQ's ability to morph from a shelf or wideband through the most intricately accurate notch is also impressive. On vocals and delicate instruments such as acoustic guitar or violin, the quality truly shines through. Add the noise gate and compressor and you'd think you just dropped an SSL into your Mac.

To top it off, company support is unrivaled in attention to user satisfaction. The bottom line is that Metric Halo simply got it right all the way around with ChannelStrip. It has excellent sound, a great price, and the people to back it up. I don't record or mix anything without it.

—Gene Porfido

Metric Halo • www.mhlabs.com

FINALISTS

ChannelStrip – Metric Halo; SpectraFoo – Metric Halo; C4 Multiband Parametric Processor Native – Waves; SoundMax with SPX – Analog Devices/Staccato; GigaStudio – Nemesys

HARDWARE

GEFORCE 3

Nvidia

Until this year, 3D hardware acceleration had seen little improvement in the functionality of chips since the original 3Dfx Voodoo. Improvements to triangle setup and fill rates were increasing the speed of the triangle engines, but specific feature innovations were at a standstill. Eventually, all chips handled bilinear filtering, mipmapping, and every blend mode that Direct3D offered. This stagnation was causing most 3D games to appear the same. Then Nvidia introduced the GeForce 3 and changed the progress of 3D hardware.

The GeForce 3's programmable vertex and pixel processing units finally gave game programmers the true flexibility to create a unique vision. Nvidia's first demos showed real-time shadows, cartoon shading, hardware-accelerated mesh skinning, volumetric fur . . . and this was just the beginning. The GeForce 3 has also added speed to triangle setup and pixel fill rates that is truly impressive.

However, creating a jump in technology was not enough for Nvidia. The company also created an SDK with technical support that was unmatched. Nvidia exposed its chip's new functionality through the Direct3D and OpenGL APIs instead of trying to enforce a new standard. Nvidia's efforts to instruct programmers in the correct ways to use its chip have been outstanding. The Nvidia GeForce 3 is a leap forward in technology and support, making it an amazing tool for game programmers.

—Michael Saladino

Nvidia • www.nvidia.com

TITANIUM POWERBOOK G4

Apple

Apple's latest Powerbook brings the multimedia studio on the road in style. Powered by the G4 PowerPC chip with Altivec at speeds up to 667MHz, the Titanium has a beautiful 15.2-inch display, DVD-ROM drive, Mac OS 9 and X, and it's all squeezed into a 1-inch thick, 5.3-pound titanium shell. Priced between $2,199 to $3,299 for the top-of-the-line system, the Titanium Powerbook is truly a sight to hold and behold.

The Titanium Powerbook fits in perfectly for the multimedia developer. Audio and graphics applications that take advantage of the G4's Altivec Velocity Engine are opening doors for artists and musicians that had been closed tightly without it. Photoshop and some of its plug-ins move at burning speeds. A new plug-in for Mark of the Unicorn's Digital Performer from Audio Ease even gives the user the ability to sample and save different architectural spaces to process audio through as authentic reverbs. And there's more to come.

Apple has created the perfect portable studio with the Titanium Powerbook. The combination of speed, size, and software, along with the Mac OS's legendary productivity, is why Apple's latest Powerbook design deserves to be honored.

—Gene Porfido

Apple • www.apple.com

FINALISTS

Titanium Powerbook G4 – Apple; ATI Radeon – ATI; GeForce 3 – Nvidia; DVD-R/CD-RW SuperDrive – Pioneer; Vaio Superslim Notebook – Sony

Return

Keeping Up with the Sims

Managing Large-Scale Game Content Production

CHARLES LONDON | *Charles has worked in the game industry for 10 years and is currently the executive in charge of production at New Pencil Inc., which provides outsourced game art content for all platforms. Previously, Charles spent four years at Electronic Arts/Maxis as the art director for THE SIMS franchise, developing the game's look and feel and leading the content creation team. He lives in San Francisco with his wife, Lori Orson, and his goldfish, Mr. Fish.*

With project budgets in the multiple millions of dollars and virtually no margin for error, more and more development teams are under tremendous pressure to come out on top of the entertainment software market's cutthroat competition. No team manager wants to contemplate dropping the ball when creating the vivid graphics necessary to help make a game a success. Electronic Arts' THE SIMS franchise is an excellent example of this pressurized situation. With record-breaking sales on its initial release of THE SIMS, as well as the subsequent successes of the expansion packs THE SIMS: LIVIN' LARGE and THE SIMS: HOUSE PARTY, EA had a lot riding on the success of its content generation strategies.

Much of THE SIMS' appeal is due to the very same element of the game that presents one of EA's strategic production challenges: the sheer number of highly detailed, technically complex objects that populate its game world and provide interaction for the Sims, the little creatures that are at the core of this unique world. Designing, specifying, engineering, and troubleshooting these objects is no small amount of work; EA counts on balancing that effort by sending the actual production of much of the sprite art assets for these objects out of house. New Pencil has been the studio charged with delivering on EA's demanding vision for these critical assets, from providing auxiliary production capacity on the original version of THE SIMS to the creation of most of the sprite assets on later expansion packs, such as the recently released THE SIMS: HOT DATE.

This article highlights the critical issues that govern the high-volume asset production needed for today's most demanding games and some of the techniques upon which New Pencil has relied to create the artwork for the Sims franchise. These principles were developed at New Pencil and at Maxis over the long association between the two studios, and New Pencil has found these techniques to be fundamentally good practices. Although New Pencil uses these principles to provide high-volume content outsourcing services, they can be employed by any internal development team with excellent results. If you are in the position of managing this kind of production capacity, there are some things you are going to need to consider.

As New Pencil developed its systems over repeated contracts with Maxis, four strategic targets emerged that helped us generate

SHOWN FROM LEFT TO RIGHT:

Yuan Zhang	3D artist
Josh Nadelberg	3D artist
Greg Faillace	creative director
Michael Murguia	CEO
Adam Murguia	art director
Disco Dog	traditional artist
Mat Smiley	3D artist
Scot Tumlin	project manager
Charles London	executive in charge of production
John Beebe	lead technical artist

a large body of game content on time and on budget while still holding quality as the uppermost value: time estimation and cost modeling, style matching and cross-team consistency, asset staging, and project tracking and team management.

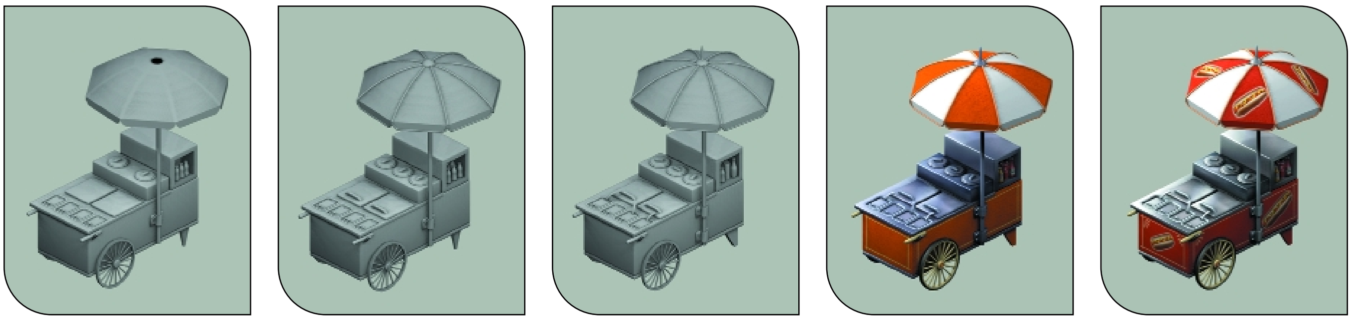
These four fundamentals are cornerstones of a solid production foundation. Omit one, and your production may collapse. Let's take a closer look at what each target entails.

Time Estimation and Cost Modeling

No project is going to be delivered on time and looking great without good projections of how long each asset will take to make and how much it will cost to make it. Too often, teams stop short and focus exclusively on the underlying technical issues that support time estimation without going to the next step. This is especially true when work is being done internally, where budgets often deal with large numbers of assets as a group line item such as "animations."

The reason for doing these projections goes beyond just wanting to hit a budget and a timeline; it goes to the deeper issue of quality. Good time projections allow you to reserve buffer time for extra polish, integrate new techniques, and create tailored resource plans to handle special concerns in the production without having to rush too fast or cut too many corners on the look of any asset.

Once the time estimates are in place, it is then possible to institute good cost models. A solid cost model allows internal teams to make the best use of their budgets and outsourcers to defend their profit margins, which in turn allows them to be more flexible and service-oriented toward their clients. It seems easy just to take the average time you think an asset will require and multiply that by the number of assets. At that point, you need to then add 20 percent right away as a rule, as a rough stab at accounting for the unknown developments that are sure to occur. No seasoned client or production manager, however, wants to depend on that kind of superficial analysis; he or she knows that there are far too many efficiencies to be found and dependencies to be contended with for any such simple formula to reflect reality. In order to develop a more in-depth cost analysis, we focus on the following three general areas: the pipeline, the approval process, and the aesthetic target.



FIGURES 1–5 (left to right): FIGURE 1. Design evaluation: The rudimentary blocking of the object to see if it's even in the ballpark. FIGURE 2. First model: The model has most of the geometric detail it will receive. FIGURE 3. Final model: The model has all the geometric detail it will get and is ready for texturing and lighting. FIGURE 4. The first pass on materials and lighting. FIGURE 5. The final result.

The place to start in developing good time estimates is the pipeline. Time estimates need to be based on a pretty solid production pipeline, or else they are meaningless. This is not to say that a pipeline can't be upgraded or amended during production, but unless the baseline is well understood by everyone who is going to use it, those upgrades cannot be evaluated for their impact on the schedule. Begin by posing a few questions about the pipeline: Who are the experts in the use of the pipeline, and is there access to those individuals? If assets have already been through production on this pipeline in the past, what were the timeframes associated with them? What technical taboos or requirements are there in the creation of the assets, and are they documented? How modular is the pipeline? If an error has been made somewhere during production, how far back in the creation must the artist regress to bring the asset up to specification?

We relied successfully on Maxis to provide instruction and support on their homegrown export tools and worked to make sure the pipeline that we set up at New Pencil was as close as possible in implementation to the pipeline at Maxis. THE SIMS' sprite pipeline was well developed before Maxis went casting for a contract house to expand their capabilities. Just as importantly, Maxis worked hard to ensure that New Pencil had access to its experts who had the inside track on how the pipeline really worked. Maxis had clear expectations as to how modifications were to be made to assets and at which point in the pipeline each asset should be modified. New Pencil also had a team with a long history of working together, which is a huge plus when developing time estimates. Many new studios or teams are forced to rely on experiences with other teams that may not reflect their actual present skill sets, exposing them to potentially large errors in time estimation. As a result, New Pencil had a high level of confidence in estimating the time the average basic asset would require, excluding revisions.

Understanding the approval process and the approval team is the next most important factor. Are the members of that group in good communication with each other? Are the list of reviewers involved in regular critiques of the submissions restricted to only those with clear responsibility for the final result? Is the review group casting a well-aimed net to garner feedback from other individuals with influence on the product's future, such as marketing or the publisher? Is there someone on the review team whose task is to manage the review team to timely consensus and with the freedom to make a command decision when the team is at loggerheads? Is there a clear progression of stages that an asset will pass through to avoid changes to previously approved work? Can the art team expect to receive feedback from the reviewers in a timely manner? Is there a general aesthetic vocabulary in use by the reviewers, and is the art team conversant with it? In the case of THE SIMS, we were able to answer yes to all of the preceding questions about the review process due to the amount of preproduction work that Maxis had done, its own existing production pipeline, and the close-knit nature of its team leadership. If the answer that you get to more than one of these questions about your approval process is "no," or if the questions cannot be answered at all, get ready for rising costs. Sadly, it's easier to say that the costs will rise because of a faulty approval process than it is to say by how much, but a good rule of thumb is to multiply the costs associated with any existing approval process by 1.5 when there is risk of multiple iterations.

For internal development teams, calling attention to this cost inflation, if done diplomatically, can be a useful way of generating the leverage at higher levels of the company to address the problems within the team that underlie the approval process's instability. No executive wants to see money being thrown away because there are questions about how solid the review team is. For outsourcers, this is a more dicey proposition. Ideally, if your relationship with your client is solid, you can consider a gently candid conversation about your concerns. If the relationship is not that strong, you're left with few options besides simply charging more for the work in order to reduce your exposure to these higher costs. However, in the bargain, you're also throwing away your competitive edge. In any case, make sure the contracts that you sign clearly state time limits and the format of feedback so that you can protect yourself. It's no substitute for being able to help the partnership address review-related challenges, but it may help you avoid eating too many revisions.

Having a clear progression of stages for the assets was also crucial. It helped focus the feedback from Maxis on the aspects of the asset at issue and made sure that the likelihood of large revisions was greater in the early part of development, when changes were less expensive. The process that resulted worked to achieve in-game integration as early as possible, giving Maxis lead time to support New Pencil should the assets not perform as anticipated. The process was divided into the following stages:

Design evaluation. The asset is shown as a high-resolution render in 3DS Max, with no texture and only enough model detail to communicate the general direction of work (Figure 1). The key elements of the asset as described in the asset specification document are referred to, but may not be completely modeled.

First model. The first model is shown as an in-game screenshot with little or no texture (Figure 2). Color may be used to help differentiate the model's substructure but is not yet a subject for review. All comments from the design evaluation feedback are addressed, if they are relevant to this stage. In some cases, the asset may be well developed enough to skip the next step and go on to texturing.

Final model. The final model is shown as another in-game screenshot, where the model is complete with respect to geometry (Figure 3). All sprite animation states are represented, and the animations function in the game. All comments from the first model submission are addressed, if they are relevant to this stage. In some cases, there will still be model comments, but they will be small enough to be addressed in the forthcoming first texture stage.

First texture. Materials and lighting are shown for the first time at this stage (Figure 4). All comments from the final model stage are addressed.

Final candidate. Ideally, all comments have been addressed and the asset is ready to be delivered (Figure 5).

The last concern for time estimation and cost modeling is the aesthetic that you're trying to match. It's safe to say that this is a larger concern for outsourcers than for internal development teams, due to the separation between the conceptual artist or art director who has the vision that sets the bar and the art team executing the work. Nonetheless, the more elaborate and esoteric the style is that has been chosen for the work, the more it will cost to produce. Don't confuse this issue with the technical complexity dealt with in the preceding pipeline discussion. Rather, this is the cost of having to undertake the revisions necessary while the artists internalize the aesthetic they must match.

The questions that need to be posed regarding the style matching have more to do with one's own capabilities than what one needs from the client or the art director: Do you feel that you understand this aesthetic? Are the reference materials both broad enough to support a consistent look across all assets and deep enough to communicate clear individual asset identities? Are the various skill sets that the project's aesthetic demands present in a balanced fashion across the art team? Is the requisite familiarity with software present in enough art team members to support general productivity? Is there someone on the team who has a particular understanding of the reviewer's vision who can share this insight with the art team? Again, we were able to answer yes to all of these questions; this was essential to New Pencil's confidence that the final assets could be produced with roughly the number of submissions planned.

So let's say you've asked all these questions of yourself and of your client or development team and you've come up with some time estimates and costs that you feel pretty sure about. Next, you should gauge your assumptions with a test run. Don't just sign up for the whole shebang if you can help it. There are going to be hidden costs, surprise developments, and plain old reversals of fortune you may or may not have anticipated. Choose a representative set of assets — a few easy ones, a few medium ones, not too many hard ones — and set yourself a milestone deadline and a team structure that reflects your best guess as to time and costs. It's important that everyone, especially your client or management, know that this is a test, and that the estimates will likely change as a result of how the test works out. It's a good way to blow the bugs out of the system and get everyone on both sides of the art fence accustomed to working together, without committing the entire project to deadlines and cost expectations that just aren't realistic.

Early on, New Pencil and Maxis took the time to learn to work together by beginning with more modest goals and then evaluating the realities of production that emerged. As a result of these early lessons, the negotiated price for content did rise by some 30 percent, but because the standard of quality had been set and the relationship forged in a climate of good communication, these price increases were perceived as reasonable and did not have to be repeated later, which would have endangered the long-term relationship between the studios.

Style Matching and Cross-Team Consistency

The name of the game is consistency; if it doesn't look like the reference, the product won't hang together. Once you've

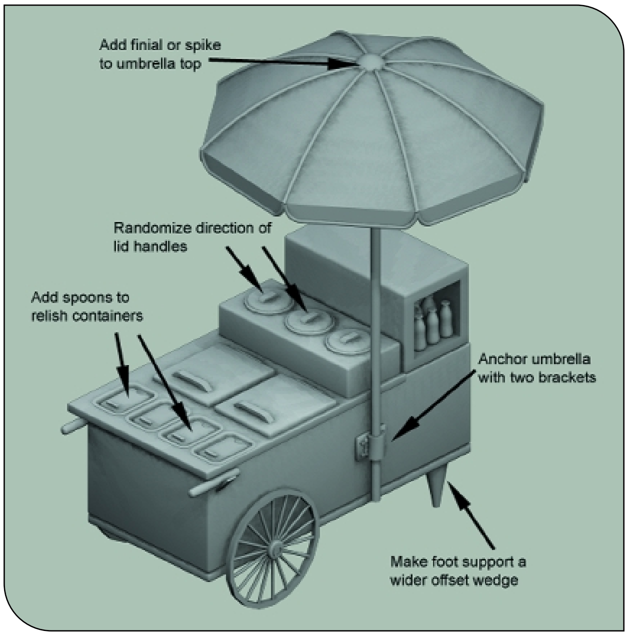

Add finial or spike to umbrella top
Randomize direction of lid handles
Add spoons to relish containers
Anchor umbrella with two brackets
Make foot support a wider offset wedge

FIGURE 6. Tight feedback rocks. The submission from New Pencil is returned with Maxis's comments included in the image.

actually begun production, it takes a bit of time for the understanding of the visual style to percolate through the art team. In the period where the team is working to internalize that vision, an aesthetic checklist is a key tool in making this period as short as possible. This checklist enumerates techniques to be employed and elements to be checked for in pursuing the aesthetic at hand. It's a supplement to the reference package that incorporates the most common feedback points that the team receives. The sidebar "The Aesthetic Checklist" shows the aesthetic checklist developed for the sprite content for THE SIMS.

Putting assets in-game very early allows feedback to be done visually, by marking up the screenshots and sending them via FTP so that the feedback can be stored on a server and remain accessible to the whole team. Working with annotated game screenshots (see Figure 6 for an example) instead of only e-mail comments did wonders for centralizing art direction.

In addition, group reviews and the development of the checklist was crucial in allowing New Pencil to improve consistently its ability to implement art direction feedback across relevant assets, thus helping to ensure that feedback provided for earlier assets was incorporated into later assets that shared those aesthetics.

Another vital tool to assure consistency across assets is the asset library. Developing an asset library is a goal that should be pursued immediately upon production. As materials or model elements are developed to approval, reusable elements that serve as style signifiers should immediately be archived in a file structure that is easily accessible to the entire team. The ability for one artist to reuse the work of another does far more than simply save time in construction; it's also a powerful way to keep the general look and feel of the project consistent across numerous artists. In addition, such reuse compensates for the inevitable inequities in skill sets among team members, allowing artists to complement each other's capabilities while working on separate items.

It's critical to get the team looking at each other's artwork. Often, an asset doesn't seem quite right, and it takes a few sets of eyes to determine why. Besides regular group reviews, the asset library reinforces this behavior by getting artists to assist each other in the integration of their various library contributions.

Template files, like asset library files, are another powerful way to help keep everyone's work in the same vein. Perhaps the most useful template file to New Pencil was a standardized lighting set, introduced after THE SIMS: LIVIN' LARGE and first used in developing a small set of downloadable objects. The lighting set is a 3DS Max file containing a fast spoof of radiosity, which all artists would merge into their workflow very early in creation. The set is not only a great time-saver with regard to the labor required to make an object seem lifelike and volumetric as per the checklist, it also goes a long way toward establishing a uniform appearance through subtle ambient lighting (Figure 7). After a few additional tweaks, this set became a standard part of asset creation both at New Pencil and at Maxis.

Asset Staging

Working on THE SIMS franchise means being faced with a very large number of objects to produce. Most recently, that figure was 100 animated object sprite sets be delivered within a 10-week timeframe. Such a high volume of work cannot be done effectively with a completely flat team. We employ a modified assembly-line process that allows some artists to specialize in their strongest skills and others with broader abilities to float, adding capacity on the part of the line that may be under pressure. The assembly line is broken up between modelers and texturers/lighters, with approximately one floater for every two craft-dedicated artists.

The art director who leads the whole team serves as a liaison between the Maxis art director and the New Pencil art team, making sure that work done at New Pencil is being held to internal standards of quality as well as the detailed direction from Maxis. In order to make sure this art director isn't spread too thin, we usually designate a lead texturer and a lead modeler. These are primarily service roles for solving technical problems and providing critique in advance of the art director's review. In addition to these resources, keeping up with THE SIMS requires a project manager to track the progress of each asset and help make projections about future bottlenecks or overloads. Figure 8 shows how all these parties interact to maximize efficiency.

Keeping things moving forward is a primary goal. New Pencil works to make sure that all artists understand their weekly workload, and the art director or project manager communicates the priority of each object within that work list. The artists are directed to focus their attention on the highest-priority asset that has outstanding feedback issues. Should an asset of higher priority come back from evaluation, the artist is expected to set aside the asset currently being worked on and address the requested changes on the high-priority item quickly. This technique keeps

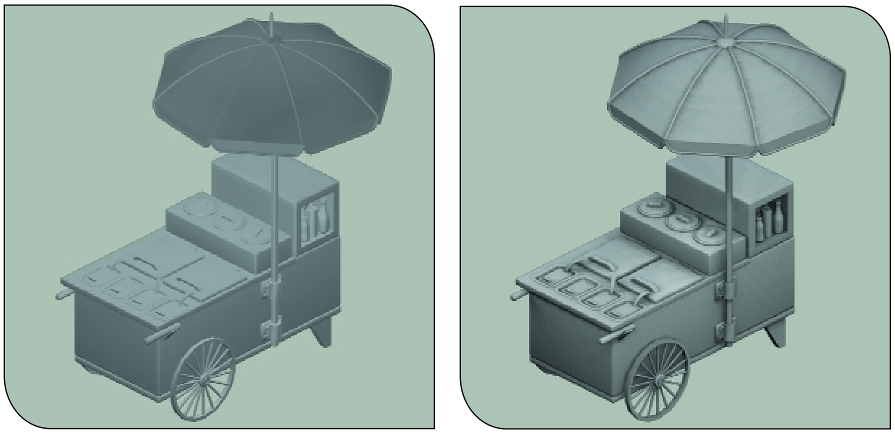


FIGURE 7. The lighting set at work. The difference between the images using default ambience (left) and the ambient light provided by the lighting set (right) is dramatic.

the most advanced assets on the front burner, speeding their process through the system and helping to achieve a spread of assets across different stages of approval. This process helps avoid bottlenecks by making sure that there are always models coming off the line for texturers/lighters to work on, and helps New Pencil keep to a regular weekly output level.

Priorities are determined using a variety of criteria: New Pencil tries to interleave difficult items with easy ones in order to avoid a hard patch of slowdowns due to too many difficult assets. (In addition, because invoicing was done by blocks of assets finished, making sure there were enough easy objects in every block helped New Pencil stay on time and thus ensured regular payments. Any internal development team can also appreciate the benefits of regular output. Even if it's not tied to payments, it's in the best interest of every team to be seen as reliable and consistent.) Prioritizing to stay within a regular band of productivity also makes it easier to beat deadlines every now and then by getting the team used to a baseline average work pace, rather than an irregular one.

Asset library needs are another criterion for setting an asset's priority. Often, the development of one asset will set the look and feel of all the others. A good example of this was the development of the castle-style objects for THE SIMS: LIVIN' LARGE. Heavy, dark wood, velvet, stone, and iron were the dominant elements in the Castle furniture and architectural items that New Pencil created. By frontloading a small set of items that used all of these archetypical materials, the Castle library elements were quickly laid down, speeding the development of the rest of the set while allowing them to be interleaved effectively with other assets as priorities required.

Lastly, work on some items must wait until the end of the project. Due to the complexity of the code that underlies some of the items, Maxis was often unable to provide us with the game files needed in order to proceed on certain assets at the beginning of production. However, because Maxis had done a good job detailing the rest of the asset list, New Pencil was able to remain productive with the objects on its plate and give Maxis the time it needed to develop the remaining objects correctly. In later con-

The Aesthetic Checklist

This checklist is a recap of all the elements that are often reiterated in feedback and that comprise the basic bar of quality against which an asset will be measured, regardless of aesthetic subtheme. The idea here is to call out all elements that need to be checked before an asset is submitted. The main purpose is to reduce revisions and to keep the burden on Maxis's review team low.

Modeling

- Have all artists been given the opportunity to ask questions regarding the object's construction?
- Are all the relevant template dimensions built to tolerances?
- If animations are already implemented in the game, has the model been checked against them?
- Do all edges have the right amount of real-world finish, such as flanges on the ends of openings, grilles, chamfers, and so on?
- Is there enough subtle low-level detail to make good use of lighting?
- Has the model been previewed at game resolution to check the visibility of all model detail?
- Does the object display properly in the game?
- Does the model conform to all called-out features of the spec?
- Has the entire team been given the opportunity to provide feedback?

Textures

- Have all artists been given the opportunity to ask questions regarding the object's materials?
- Do the materials look realistic and similar to spec examples (woodgrain visible and natural; metal shiny, brushed, or patinaed as appropriate; colored metals rich and non-plasticky; chrome well developed in reflections; and so on)?
- Are materials and/or textures being appropriately reused when items are intended as an associated or matched set?
- Are there evident repeats in the texture?
- Where a texture from one geometry has been reused on another of different proportions, has it been checked for distortion?
- Are the colors mapping well to the run-time file?
- Is dithering being used properly to defeat banding?
- Has the palette been polished where necessary?
- Has the entire team been given the opportunity to provide feedback?

Lighting

- Have all artists been given the opportunity to ask questions regarding the object's lighting?
- Is the lighting set current and installed properly?
- Is a keylight being used to properly develop box lighting for cubic forms, dihedral angles, and planar objects?
- Are keylight highlights being effectively used to set off shadows and shiny surfaces?
- Are surfaces that overhang other geometry being properly shadowed?
- Are interior spaces slightly darker than exterior surfaces?
- If any part of the object lights up, is it effectively casting light back onto itself where relevant?
- Is the pattern of cast light representative of the illuminated object's apertures (grilles, shades, and so on)?
- If any part of the object lights up, is there a satisfying level of contrast between lit and unlit states?
- Is the shape of the object well revealed by the lighting set?
- Are lighting gradients being used to relieve large, flat expanses?
- Are subtle gradients being used across standing elements to help give a feeling of verticality?
- Are lighting levels being corrected for dominance in sprites that display together but export separately?
- Has the entire team been given the opportunity to provide feedback?

tracts, as the relationship between the studios has become more mature, Maxis has chosen to wait to detail some 20 percent of the assets until the majority of the work could be seen together, thus choosing items that would best finish out the set.

Project Tracking and Team Management

All the good plans in the world won't help you to manage large numbers of assets if you don't have some sort of tracking system. New Pencil's system is a simple but effective Microsoft Access database. The project database allows the project manager to list assets individually, define a series of states that the assets can pass through on their way to final, track which artist is assigned to these assets, and timestamp the asset for when it was submitted and when feedback was returned or when it was finalized. Furthermore, the system allows the generation of particular reports, such as project overviews by state or artist, or reports of assets that are blocked and what is blocking them. While these reports are invaluable, it's important to make sure that the system is easy to use and not overly detailed. Too many reports are a sign that you may be confusing the asset database with the project itself; ideally, you'll be spending as little time on it as possible, so don't burn too many hours setting it up.

The project manager who manages the tracking system is also the person who serves as the exclusive technical liaison with Maxis. The project manager notifies Maxis of files that are needed, of evaluations posted, and of requests for technical support. Likewise, the project manager also notifies the New Pencil art team of arriving feedback, priority assignments, predictions of bottlenecks or chokes, and any technical changes that have arrived from Maxis. Keeping a single contact for these kinds of matters ensures that Maxis always knows whom to contact in case of some snafu or concern; on projects like THE SIMS expansion packs, time is of the essence. One of the most important things an art team can do is adopt the no-surprises rule when dealing with its clients or individual team members. While being the bearer of bad news is never pleasant, providing a heads up early enough for a solution to be formulated builds confidence and acclimates the team to problem solving and avoiding panic.

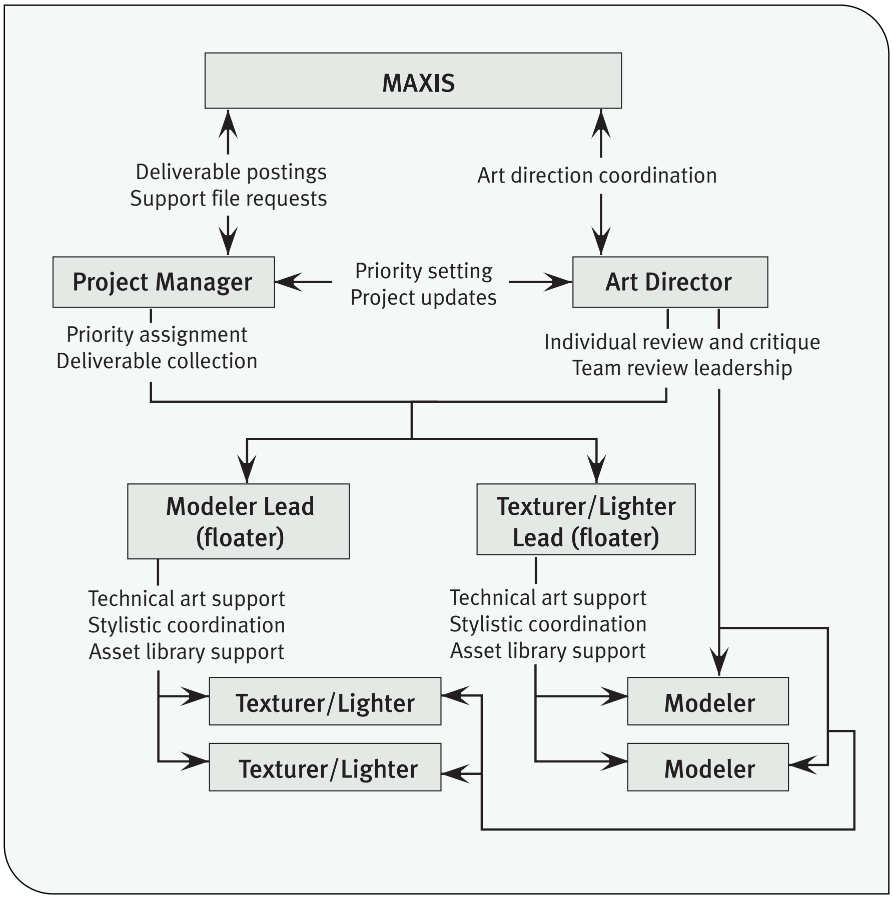


FIGURE 8. A schematic of the production pipeline that exists between Maxis and New Pencil.

Some simple but solid personnel management techniques will be needed to stay on the plan that you've worked hard to create and make sure that the work is of the highest caliber. First and foremost, the team needs to understand the task set before them and be motivated to achieve it. Make sure that everyone gets a chance to review all of the art materials. While you don't want to bury artists working on one asset group with the minutiae of references that doesn't relate to their assignments, it's important that everyone understand the broad context that relates to all the work. Make time for questions from all team members, both individually and in a group setting, and make sure that questions are answered fully. Some of the most important technical challenges will be spotted not by the management but by the people on the ground who actually have to take the hill.

Get feedback from the artists as to the usefulness of the reference material at hand and make clear action items to supplement the reference in places where artists clearly are not getting the concept. Where possible, give artists the opportunity to choose assets to work on for themselves instead of just being assigned their workloads. A sense of ownership will do wonders in helping the artist bring his or her own vision to the work in service of the existing aesthetic.

All-nighters and seven-day weeks are a fool's game. Nothing superb ever gets done by overtired people whose lives have been turned upside down. Sure, the game business requires the occasional supreme effort, but the smart manager does everything he or she can to avoid such excesses. Put your artists first. If they feel you are behind them, they'll be more than capable of providing the exceptional work that you've guaranteed.

marek spanel and ondrej spanel

Bohemia Interactive Studios'

OPERATION FLASHPOINT

GAME DATA

PUBLISHER: Codemasters
FULL-TIME DEVELOPERS: 10
PART-TIME 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: 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

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

MAREK SPANEL | *Marek worked on home-grown games even as a child back in the late 1980s, with his brother Ondrej. In 1992, he started working at a game distribution company, and in 1998 he started to work full time on his first PC game, eventually released as* OPERATION FLASHPOINT. *In 1999, he co-founded Bohemia Interactive Studios, the company that undertook development of* OPERATION FLASHPOINT.

ONDREJ SPANEL | *Ondrej coded his first games in the late 1980s. After graduating from Charles University in Prague in 1995 with a degree in computer science, he started work as the lead programmer for research and development of a PC game that was completed in 2001 under the title* OPERATION FLASHPOINT.

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.

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.

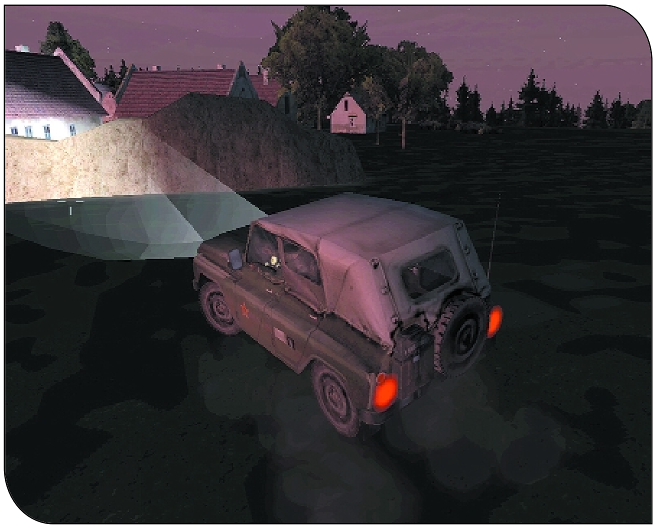
The publisher's test staff started playing the game Sunday afternoon. Everything went smoothly at first, but later they discovered one serious scripting bug in one of the campaign missions that made it unplayable. Late in the evening, they called the team leader about the bug, and he had to drive to 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 enables the player to use any vehicles, including gunships and planes.



Daytime and weather changes dynamically in the game so the are looks pretty different in various daytime and weather conditions.

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.

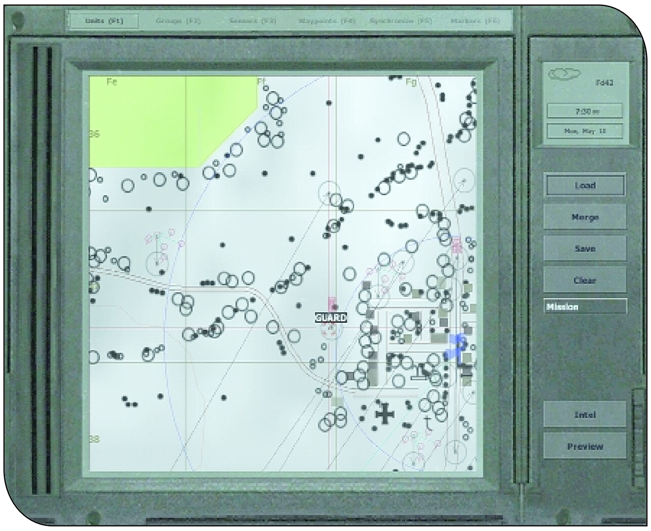
Since the release of the public demo version (around three months prior to the release of the final game), the community following has become much bigger. The only downside to the increase in the community base is that the community has become much less focused and less mature, as the average age of 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.

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 and properties 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.

The Operation Flashpoint demo enabled millions of players around the world to taste the game around three months prior to its commercial release.



The built-in visual mission editor opens endless gaming options with easily created missions.

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.

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.

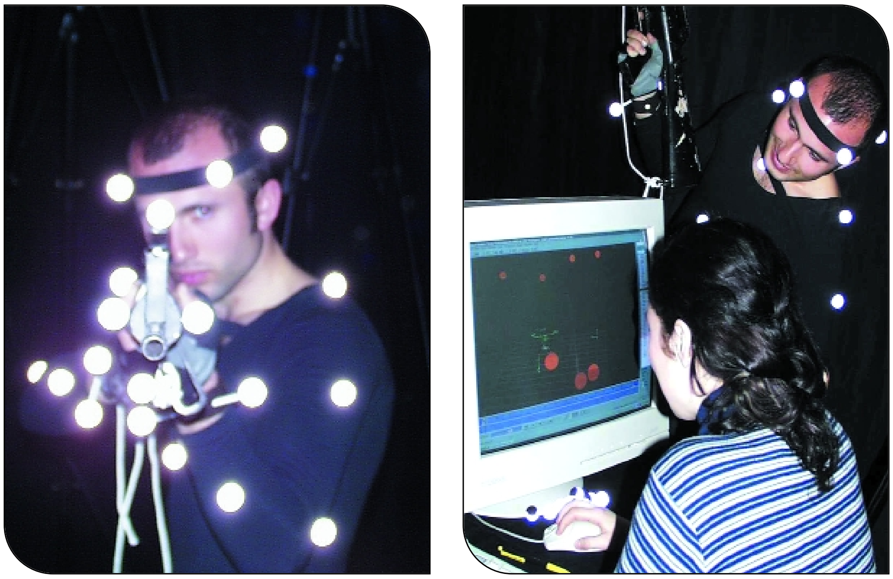
First of all, some technologies in the game were a bit outdated after more than four years. We didn't know at the outset that the game would be still in development after so long, and we hadn't left time at the end to rework parts of the engine. Some of the criticism of OPERATION FLASHPOINT addresses 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.

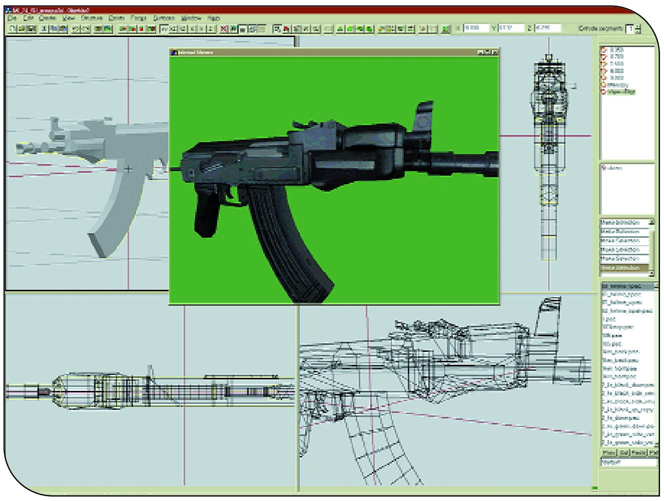
The main problem wasn't that the development cycle was too long per se, but that the development was so much longer than we'd expected. Next time, we will work much more diligently to better estimate our development time — and we will probably try to aim higher with the detail of our artwork, even if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

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.

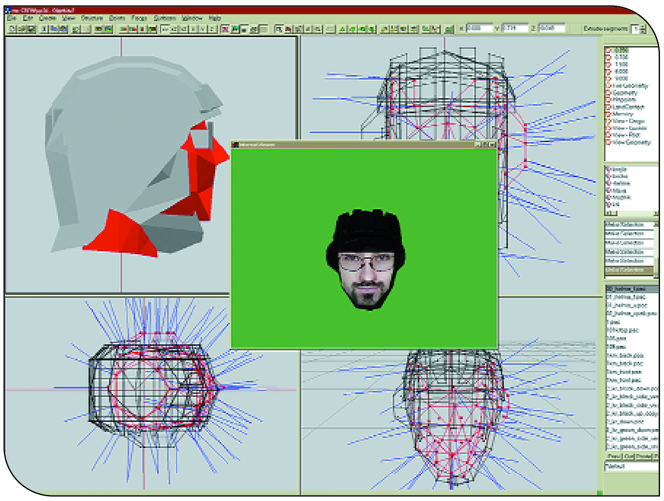
While we'd never believed too much in designing the game on paper, the real problem was that we never even had documentation of the things that we'd finished. This situation led to incredible problems in the final stages of development. Many tasks could only be done by one person on the


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

The combination of infantry combat with full simulation of vehicles is a crucial part of the design of OPERATION FLASHPOINT.



A Soviet AKSU rifle shown in Bohemia Interactive's proprietary modeling tool, Oxygen, with direct real-time previewing using the game's engine.



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.

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.

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.

OPERATION FLASHPOINT's mission was to deliver the tension of full-scale conventional military conflict.

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.

SPECIAL ON GAMASUTRA.COM

Step back in time and check out two of Bohemia Interactive's early demos of OPERATION FLASHPOINT from 1996–1997. Visit www.gamasutra.com in mid-December to download.

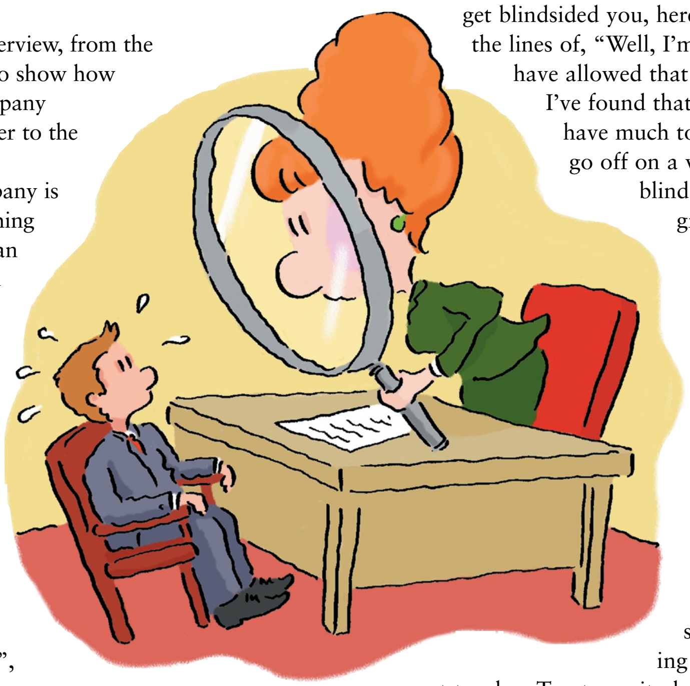
Return

jake simpson

Interviewing at Game Companies

Job hunting the game industry — that most fun of occupations. Recently I've run that gauntlet, jumped through that hoop, been into that bag, or whatever idiom you prefer. I tend to think "descended to that particular ring of hell" sums it up fairly well. I dunno, have you ever had an interview where you didn't sit there squirming with anticipated humiliation?

The whole point of an interview, from the interviewee's perspective, is to show how brilliant one is, how the company should be making a huge offer to the interviewee immediately, and indeed, how lucky said company is that the interviewee had nothing better to do that morning than to be there inspecting it. Said company's representative conducting the interview should do so without putting any questions to the interviewee that he or she might not be interested in answering. In fact, said interviewer shouldn't really ask any questions apart from, "How big do you want your office to be?", "Would you please fill in this blank check yourself?", and "The masseur arrives on a Wednesday afternoon; is that good with you?"



Interviewers should not indulge in programming tests, require proof of any kind of technical ability, and specifically not ask for references. And if they do, they should not follow up on them. What kind of company wouldn't trust you? If I say that I personally wrote all of the rendering code for the new DOOM engine for John Carmack as a personal favor, then that should be the end of it. Be properly impressed and reach for the checkbook, please.

For some reason, some companies ask you to bring proof of your abilities: demos, portfolios, examples, and so on. Why this seems to be an accepted practice is beyond me. Maybe interviewers just need something to do in the evenings, like play your games so they can see your brilliance showing through or rip off your undoubtedly fantastic ideas.

Some companies try to trap you by asking questions such as, "What's the worst bit of code/management situation you've done/been in?" This is a trap, people. Be prepared. I am.

At this point in the interview, I just point out that everything I've done has been nothing short of inspired, then divert the line of questioning by pointing out the window and asking if it's rained frogs recently.

Sometimes you'll have to admit that something you were involved in wasn't perfect. Try to avoid this if you can. But if you get blindsided you, here's a good tip. Say something along the lines of, "Well, I'm not perfect — if I were I wouldn't have allowed that other person to make that mistake." I've found that, for some reason, interviewers don't have much to say after that. At this point, you can go off on a wild tangent about the color of the blinds or how the interviewer would look great with orange contact lenses.

Other Tips I've Picked Up Along the Way That You Might Find Helpful

Ensure you've gone out and gotten a skinful of beer the night before. Nothing says "I am a social animal" to interviewers like a faceful of last night's beer. And nothing says, "I am totally comfortable talking to you" like letting a nice big belch out to play. Trust me, it always goes down a treat. At least I think it does, judging by past interviewers who've cried with laughter. I'm assuming it was laughter, anyway, though they were definitely crying.

Try not to mention black magic too often, or ask the interviewer if he or she has a daughter, as this often offends. (It's usually safe to ask about a sister, though.)

Be sure to visit Best Buy or Electronics Boutique before going to an interview so that if they ask what you're currently playing, you can drop some of the names of what's currently on the shelves. I mean, it's not as if you have time play anything, what with your busy social schedule. And if they ask about bad games, be sure to mention one of their company's titles. That way you can really show them how cool you are, and how you're not afraid of saying what you think. They can't fail to be impressed.

Illustration by Scot Ritchie

continued on page 63

Don't worry at all about lying on your resume. I looked it up and "resume" is actually Latin for "fiction," so you're O.K. there. Just make sure you don't lie too obviously. Saying that you were CEO of Microsoft for three years is probably not going to pan out — someone might spot that. On the other hand, that 18-month stint you did for shoplifting women's underwear is something you definitely want to keep quiet.

A really good get-out-of-awkward-question phrase that I discovered is, "I'm sorry, I'm contractually bound not to discuss that." It's right up there with, "I'm sorry, for me to tell you that would contravene the Official Secrets Act." For example:

"You worked on project X?"

"I'm sorry, I'm contractually bound not to discuss that."

Or, "Who was the 17th president of the United States?"

"I'm sorry, I'm contractually bound not to discuss that."

Or, "Wow, crappy weather, eh?"

"I'm sorry, I'm contractually bound not to discuss that."

Actually, you could probably answer that last one. Just be careful and never mention "vibrant moonbeams." People will look at you funny.

So, to recap: Be yourself, and make sure you impress the interviewer with your wit and sophistication. Some good potty humor usually breaks the ice. You can't fail to win any job you set your mind to.

Oh, and you don't need to thank me.

JAKE SIMPSON | *Jake works for Maxis, appearing every day dutifully and cashing the checks as soon as he gets them. Hey, you never know, right? He's been in the game industry for a few years, working for Midway, Raven, and others that have vanished into the mists of time. Wow, that sounds poetic doesn't it? If you want to read more of his drivel, check out www.jakeworld.org. He can be contacted at jakesimpson100@yahoo.com, but please, he's a sad, sad individual. Treat him with caution, and above all, don't buy him beer.*