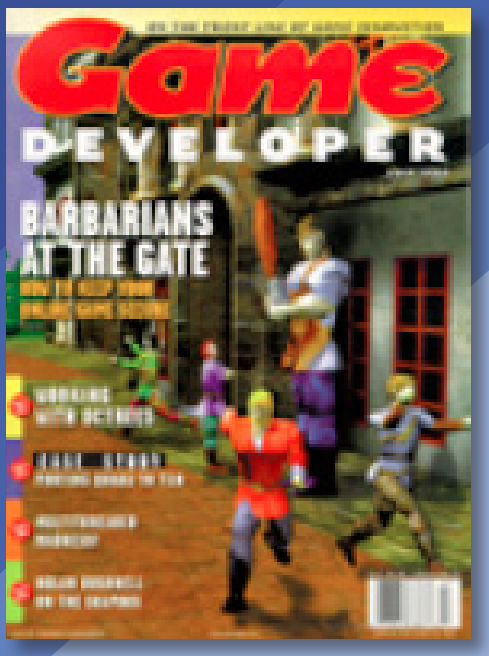
gd

GAME DEVELOPER MAGAZINE

JULY 1997



UBM
Tech

Beat the Cheat

Here are the Diablo Cheat Files. A couple notes on using some of the codes:

-For Superweapon Godly Mana must be off, then just cast healing.

-To use Leach, kill yourself (using firewall) down to 1 hp and then cast Bonespirit, everybody thinks you are dead.

-Townkill will not affect other players and others will not even see it unless they also have townkill turned on.

-To use the item dupe just pick up the item you want to dupe, put it down and then click an empty spot in your inventory.

-To use the Crash code just turn it on, enter a game, and everybody in that game except you will get an illegal operation error and their Diablo will crash.

This selection was excerpted from of one of the many web pages devoted to cracking Blizzard's DIABLO. DIABLO, one of the best selling games of 1997 thus far, also happens to be hold the dubious distinction of being one of the most widely cheated games during the same period. No surprise there. A game as popular as DIABLO often attracts a proportionate share of attention from crackers armed with NuMega's SoftICE or some other system-level debugger. What made DIABLO especially fun for people to tamper with was the game's multiplayer online component, Battle.net. After all, what's more rewarding to a cheater than showing off his/her exploits to the rest of the world by manipulating the game environment in the full view of others? It's a stage for cheaters to demonstrate their abilities, their "technical superiority."

As the business models and the technology that underlie online games evolve, security breeches will not only become more common, they'll grow more serious. Many people feel that the next leap in online gaming will be the creation of pay-to-play persistent worlds, not unlike MERIDIAN 59 and the forthcoming ULTIMA ONLINE.

The allure of pay-to-play online worlds will be far too great for crackers/cheaters to resist. However, unlike the Battle.net scenario, where other players are the biggest victims of cheaters, soon it will be game developers who get burned.

First, in policing players that modify game parameters in order to give themselves superhuman powers, steal others' possessions, kill other players, or crash the system itself, companies will spend time and money. However, the long-term damage may be much more costly. Consider a scenario where a cheater gives superhuman abilities to an online character. That cheater uses this character to kill others, solve puzzles, and "upset the natural balance" of the game in other ways. Honest players get fed up, quit the game, and tell friends considering purchasing it not to waste their money. Word spreads, people stop playing, and the revenue stream dries up for company. Net result? A developer that has invested thousands — or millions — of dollars on the game's development in the expectation of recouping it through a monthly revenue stream goes under.

You can't blame customers for quitting. It's bad enough to log into a free game service like Battle.net and inexplicably get robbed of all your hard-won possessions by the character standing next to you on the screen. But if you're paying $9.95 a month for that privilege, forget it. Cheat-proofing online games could become a thorn in our collective sides soon.

Now Batting Cleanup

I'm sad to have to say adios to one of our veteran columnists, Chris Hecker. Hecker's getting a hall pass good for some time away from the magazine until he gets his game out the door. His technical expertise will be missed by everyone, and we all look forward to his return sometime soon.

On a happier note, Brian Hook joins our pages. Hook is a recent addition to the team at id. His column "Graphic Content" will keep tabs on the 3D programming scene and present technical tutorials. Welcome, Brian. ■

Game Developer
ON THE FRONT LINE OF GAME INNOVATION

EDITOR IN CHIEF Alex Dunne
adunne@compuserve.com
MANAGING EDITOR Tor Berg
tdberg@sirius.com
EDITOR-AT-LARGE Chris Hecker
checker@bix.com
CONTRIBUTING EDITORS Larry O'Brien
lobrien@msn.com
Brian Hook
bwh@wksoftware.com
David Sieks
gdmag@mfi.com
Ben Sawyer
bensawyer@worldnet.att.net
ART DIRECTOR Azriel Hayes
ahayes@mfi.com
ADVISORY BOARD Hal Barwood
Noah Falstein
Susan Lee-Merrow
Mark Miller
Josh White

COVER IMAGE 7th Level

PUBLISHER KoAnn Vikören
ASSOCIATE PUBLISHER Cynthia A. Blair
(415) 905-2210
cblair@mfi.com

REGIONAL SALES MANAGER Tony Andrade
(415) 905-2156
tandrade@mfi.com
SALES ASSISTANT Chris Cooper
(415) 908-6614
ccooper@mfi.com

MARKETING MANAGER Susan McDonald
MARKETING GRAPHIC DESIGNER Azriel Hayes
AD. PRODUCTION COORDINATOR Denise Temple
DIRECTOR OF PRODUCTION Andrew A. Mickus
VICE PRESIDENT/CIRCULATION Jerry M. Okabe
GROUP CIRCULATION MANAGER Mike Poplardo
SUBS. MARKETING MANAGER Melina Kaplanis
NEWSSTAND MANAGER Eric Alekman
REPRINTS Stella Valdez
(916) 983-6971

Miller Freeman
A United News & Media publication

CHAIRMAN/CEO Marshall W. Freeman
PRESIDENT/COO Donald A. Pazour
SENIOR VICE PRESIDENT/CFO Warren "Andy" Ambrose
SENIOR VICE PRESIDENTS H. Ted Bahr,
Darrell Denny,
David Nussbaum,
Galen A. Poss,
Wini D. Ragus,
Regina Starr Ridley
VICE PRESIDENT/PRODUCTION Andrew A. Mickus
VICE PRESIDENT/CIRCULATION Jerry M. Okabe
SENIOR VICE PRESIDENT/SYSTEMS AND SOFTWARE DIVISION Regina Starr Ridley

www.gdmag.com

OpenGL vs. Direct3D Redux

Scrapping the Direct3D Immediate Mode would benefit the game industry. The distinction between low-end gaming boards and high-end professional accelerators is completely artificial and was introduced only because the first (and most of the second) generation accelerator chips rushed to the market were far from being worth full-featured OpenGL support. Features were missing, or the boards performed too slowly to have several features enabled at the same time. I suspect that within a year, this distinction will have vanished entirely. The only difference between low-price and high-price solutions will be how a given board is outfitted: one or more texel processors, more on-board memory, dedicated memory, separate buses, faster memory, additional geometry processor, and so on. Soon, even the cheapest board will provide enough features to be worth an OpenGL driver. If we use OpenGL as the standard to start with, extensions to cutting-edge hardware will continuously creep down into the mass market, following the hardware's improvements and falling prices.

The entire graphics programming community will benefit from experiences originally gained on the high-end workstations. After all, that's one of the major benefits of using OpenGL: a decade of experiences to build upon. I'd rather have the ARB members discussing OpenGL API modifications to support procedural texture hardware, backdrops and depth-mapped 3D sprites, and vertex-free representations or multiresolution representations than wait until Microsoft manages to push Direct3D up to OpenGL's level of maturity. OpenGL provides a head start in a hardware accelerator market that is going to explode in several divergent paths some day. When that day arrives, keeping any API up to date is going to be a full-time task.

I'm glad Hecker wrote that column, and I'm glad *Game Developer* published it.

Bernd Kreimeier

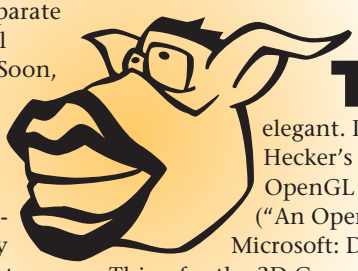
Tastes Great...

I think the magazine looks great. There seems to be a "fun" edge to it now that it never had before. The content, as usual, remains an important reason to keep a library of back issues. I for one would appreciate an opportunity to buy a GD binder for them.

Also of special note, I enjoyed the editorial. I am helping a friend of mine with a magazine, and before starting, he and his entire staff decided to donate their time for the first year. We were excited to read that *Game Developer* started in a similar way. We're just starting our third issue.

Keep up the good work. *Game Developer* will always remain an invaluable part of my development library.

Wade Winningham



This new look is much more elegant. I really enjoyed Hecker's column on OpenGL vs. Direct3D ("An Open Letter to Microsoft: Do the Right Thing for the 3D Game Industry," April-May 1997). The column didn't beat around the bush. It got straight to the main issues. Kudos to your writers.

Mike Whitmire

I think it's great that there's a magazine devoted to game development issues, and I like the format. I particularly appreciate the fact that you're printing your articles readably (not overwhelming the text with fashionable graphics). It's also easy to tell text from advertising, which is not always true of technical magazines. Your preview of CGDC was helpful. I was glad you included Annie Fox among the CGDC interviewees. In my view, getting past the (mainly male) core gamers to a wider audience means taking note of what the young female market wants — beyond Barbie — and designing games to suit. Annie Fox is an eloquent presenter of that side of things. Overall, the look is right and the content interesting and informative.

Charles Cameron

...Less Filling

I first started reading *Game Developer* a year and a half ago, and I was immediately hooked. I loved the magazine and cherished every issue, however recently it seems that you guys are aiming more and more towards the big-time developers and leaving us garage and closet compilers to fend for ourselves. What I'm getting at is that I want to be able to make use of the information I find in here without having to have a Silicon Graphics machine. Other than that I think that this magazine is the greatest. Keep up the good work.

Jason Smith

Tzvi Freeman ("Monitoring Devices in Games," April-May 1997) talks about how "Numbers and things... feel better at the bottom." Perhaps you could apply some of the article's suggestions to your magazine's layout. I am constantly looking at the bottom of the page to find the page number, only to realize that it's now hidden under my thumb!

Jared Mathes

Glad to hear you're going monthly. This is the first issue that I've been able to find in awhile. The online issue on Internet gaming was interesting reading and has definitely influenced the gaming project I'm currently working on. I look forward to your next special.

I like the new look, and the wider focus of the articles. Focusing on all areas of game development looks promising (but don't forget the meaty programming articles that I buy the magazine for). Hecker's column on OpenGL was very interesting (as are most of his writings). I also liked all the other articles except the puff piece on the CGDC. The little info box on the first page of the article and a one page "cool stuff to see" would have been adequate.

Rob Basler

INDUSTRY WATCH

by Ben Sawyer

NET GAMING IN SEARCH OF THE SIGNATURE PRODUCT. When TEN makes a press release out of hitting the 25,000 paying-subscribers mark, you get the impression that things aren't going so well for Internet game networks. That announcement, made a few months ago, indicates to me that there is a general malaise in this market sector. Currently, I see but three games making a name for themselves with their online capabilities: DIABLO, QUAKE, and ULTIMA ONLINE (and ULTIMA ONLINE isn't even shipping yet). But QUAKE and DIABLO simply aren't the breakthrough games that Internet-based gaming needs, and ULTIMA ONLINE is still a largely unknown quantity.

The slow growth of Internet-based gaming certainly isn't due to a lack of commercial game services. Quick, how many game networks can you rattle off? I can think of seven, not counting some of the smaller match-up services/free networks being deployed by the likes of EA and CUC.

The two most prominent networks, TEN and Mpath, are representative of the struggles facing all such services. TEN now claims 30,000 subscribers, plus some limited ad revenue. At $19.95 per subscriber per month, they're just shy of $6 million in annual revenue (plus some cash from ads). To me, that sounds like chicken feed, but so far TEN seems to be sticking to its pay subscription model.

Mpath isn't doing much better, but it's beginning to diversify somewhat. First, the company launched an advertiser-supported model, the Mplayer Freezone, which offers free access to multiplayer products. Mpath will also offer a premium service with extra features for $29.95 per year. Further, Mpath announced a pay-per-play business model for games that don't have a

Life Forms 2.1

CREDO INTERACTIVE has released Life Forms, a 3D tool designed as a niche product for the animation, planning, and playback of human movement.

The new version features a host of import/export plug-ins, including BioVision .BVH motion capture data, VRML 1.0 and 2.0, 3D Studio, Infini-D, Extreme3D, SoftImage, ElectricImage, and Alias.

Life Forms 2.1 includes something called an integrated walk generator. Users build a walk sequence by specifying parameters such as distance, step length, velocity, and step count or by specifying a line in a path for a figure to walk along. The package's figure editor provides body control with its direct manipulation of forward kinematics. There is a timeline that displays graphical representations of key shapes over time. Timeline editing functions include cut/copy/paste, shrink/stretch, and mirror. And there is a set of predefined animation libraries, including runs, walks, tumbling, skating, diving, swimming, and jumping.

Life Forms 2.1 is available for MacOS or Windows 95/NT. The list price is $299. Life Forms 2.0 users will be upgraded with a free electronic version. Life Forms 1.0 customers can upgrade for a time-limited price of $99. Prices do not include shipping, duties, or local taxes.

- Credo Interactive Inc.
  Burnaby, B.C., Canada
  (604) 291-6717
  E-mail: lifeforms@cs.sfu.ca
  Web: fas.sfu.ca/lifeforms.html

Poser 2

FRACTAL DESIGN CORP. is also talking character animation with the latest version of Poser. This is a significant upgrade, featuring new import/export capabilities that complement Fractal Design's other products, as well as some of the more popular 2D and 3D design applications.

One of the nice things about Poser 2 is the collection of body figures specially commissioned from Viewpoint Datalabs. Users can also substitute 3D models for body parts, which might be helpful in creating cyborgs and other-worldly creatures. These objects, along with parts such as wigs and rings, can be linked to the body for smooth and unified motion in animations.

Poser 2 also combines keyframe animation and Inverse Kinematics, so users can simply set the beginning and ending keyframes, and Poser interpolates all the in-between movements. Animations can be saved as QuickTime or AVI movies with masking information.

Fractal Design Poser 2 is available for Windows 95 or Windows NT. The List price is $249. Current Poser users can upgrade for $69.

- Fractal Design Corp.
  Scotts Valley, Calif.
  800-846-0111 or
  408-430-4000
  Web: www.fractal.com

HiProf

TRACEPOINT Technologies, a newly formed subsidiary of Digital Equipment Corp., has just introduced HiProf, a graphical hierarchical profiler for analyzing code and enhancing the performance of 32-bit C and C++ applications.

HiProf is based on what DEC is calling Binary Code Instrumentation (BCI) technology. Using BCI, HiProf can instrument and profile compiled and linked .EXE files or DLLs more quickly than a profiler operating at the source or object code level. HiProf offers an intuitive GUI with easy reporting and the ability to import results to an Excel spreadsheet.

HiProf lists at $599. Interested coders can download it from TracePoint's web

site, or purchase it from more traditional retail channels. HiProf runs under Windows 95 or Windows NT and supports all Win32 applications developed using Microsoft Visual C++ 2.x or higher and Microsoft Developer Studio 4.0 or higher.

- Tracepoint Technology Inc.
  San Jose, Calif.
  408-283-5377
  WWW: www.tracepoint.com

Fuseworks

FUSEWORKS, A RECENT startup in Hull, Quebec, Canada, has announced the availability of its Fuseworks Server 1.0 and SDK, allowing developers to create multi-user content for web-based gaming using Java and Shockwave.

Fuseworks has built a high level of functionality into the Fuseworks Server, eliminating the need for server-side CGI programming on the part of the game designer. Fuseworks has put the emphasis on game design, offering their SDK free for download and opening up access to the Fuseworks Server for testing. The company is hoping to promote a community of developers and players that will feed content providers/publishers, who will host multiplayer gaming sessions by running Fuseworks-based multimedia applets on a Fuseworks Server. To spark the interest of developers, Fuseworks is offering a $25,000 cash prize to the designer of the most compelling Fuseworks-based game in its "Ultimate Fuseworks Game Competition."

Again, the SDK is free, but you need to be versant in Java or Shockwave. Contact Fuseworks directly for information on the Fuseworks Server. Check the web site for information about the contest.

- Fuseworks Inc.
  Hull, Quebec, Canada
  819-771-8182
  Web: www.fuseworks.com

Catalyst

IN A SIMILAR VEIN, NEWFIRE Inc. has released Catalyst, its authoring tool for creating web-based games.

Closely following the introduction of Torch, Newfire's Internet entertainment player, Catalyst gives developers an extensible game authoring tool with a plug-in–based architecture. Catalyst supports art imported from many animation and rendering packages, such as 3D Studio, and gives designers control over scene composition and interactivity. The package includes capabilities for real-time playback and analysis for testing purposes, optimization of 3D scene geometry, and project management.

The Catalyst base product is priced at $1,995 for a single-user license. An SDK for developing plug-ins is in the works. Interested developers can join Newfire's Ignition developer program on Newfire's web site.

- Newfire Inc.
  Saratoga, Calif.
  408-996-3100
  Web: www.newfire.com

Nichimen Fast Track

KEY COMPONENTS of Nichimen Graphics' game development toolkit, N•World, have just been encapusulated in a new product called Fast Track. Fast Track lets you create polygonal characters and scenes in real time, paint directly on their surfaces, blend seams, and reduce the final colors. Fast Track can preview objects and texture data on the PC and convert to DirectX and VRML 2.0 formats. The product is available on both SGI and Windows NT platforms. Pricing is $9,995 per platform, but a special deal offered through July gets you both versions for $6,495.

- Nichimen Graphics
  Los Angeles, Calif.
  310-577-0500
  Web: www.nichimen.com

retail component. Finally, the Mpath Foundation was formed to help game companies build their own gaming networks (which seems to be increasingly popular for game developers such as Blizzard).

Mpath also debuted the OLO (Online Only) Incubator with the goal of helping developers launch online titles. This move indicates Mpath'a interest in jumpstarting original title development for their service.

At the CGDC, it was announced that games sporting a strong online multiplayer component sell 20-30% more units than those without. That's good news, but the types of games we're seeing today are not going to buoy the online game services for long. How are game networks responding to slow growth? Right now, the common words and phrases in the lastest batch of press releases include:

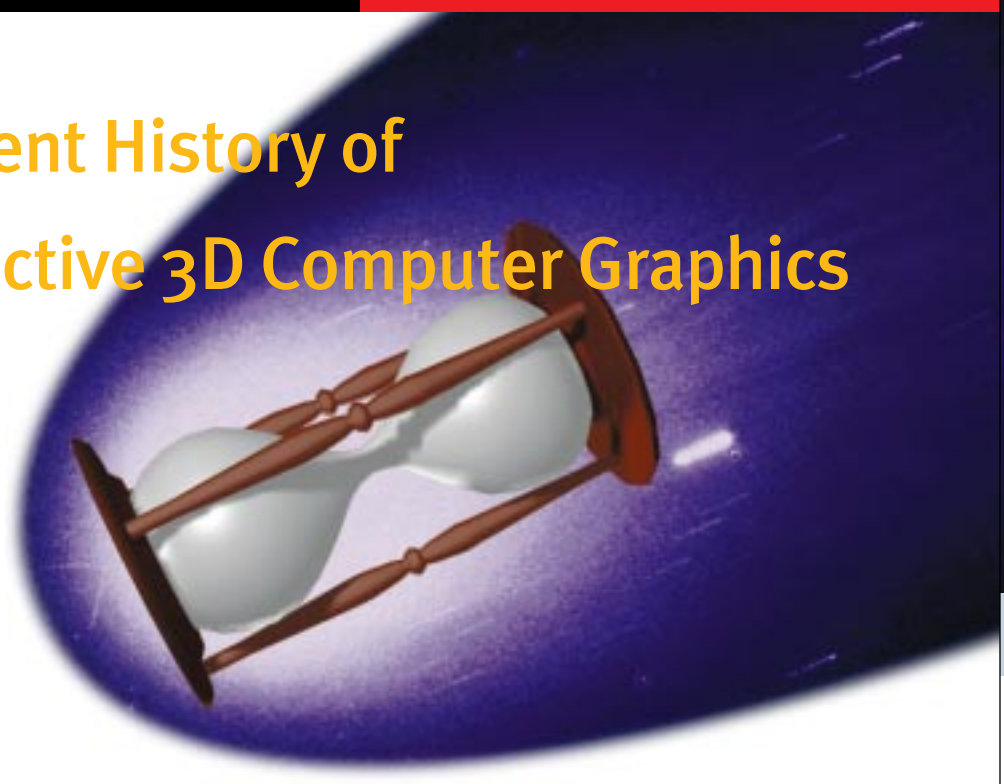
- **"Exclusive." (Indicating a different line up of titles you won't find elsewhere.)**
- **"New pricing model." (Also known as "deep price cuts.")**
- **"Advertising." (Here's where deep price cuts get compensated.)**
- **"Free." ("New pricing model" +"advertising" = "New revenue model.")**

In a year or so, unless big things happen, the next press releases might feature words like "acquisition" and "merger." I feel that this market *will* take off, but the combination of too many providers (which confuses customers), not enough compelling games (which depresses customers), and no critical mass (which scares investors and developers) is holding the market back.

What's needed is a signature product, just as Nintendo needed MARIO, Genesis needed SONIC, and the PC needed DOOM. The world of Internet-based multiplayer games is searching for a breakthrough title. That title might be ULTIMA ONLINE or some other persistent RPG world. Regardless of which title fulfills this prophecy, many are awaiting it eagerly.

NEWS RESOURCE OF THE MONTH.
Catherine S. Kirkman sends weekly e-mail of news in the multimedia/games world. You can subscribe by sending "subscribe multimedia-list" in the message (with no subject) to majordomo@case.wsgr.com.

A Recent History of Interactive 3D Computer Graphics



As I'm typing this column, it's spring in northern California, which is fairly irrelevant for most northern Californians, since we only have two seasons — cool/wet or cool/dry. The point is that by the time you're holding this magazine, it will be summer, and things in the world of 3D will have changed since I typed these words. So while I'd like to address the state of interactive 3D graphics as I see it today, remember that a lot can happen in one season (or "one quarter" as we cool pretentious types like to say).

No More Fixed Point

The past few years have given us a lot of advances in computing technology, many of which are directly applicable to 3D graphics. A major technological advance of the past two years, and one that is often taken for granted, is the widespread adoption of processors with fast floating-point units, including the Intel Pentium and Pentium Pro and the Motorola PowerPC. This means that fixed-point math could largely go away, giving 3D worlds more dynamic range and greatly simplifying the process of writing 3D pipelines. No more worrying about overflow or underflow, no more assembly routines for doing fixed-point multiplies and divides, and no more typing into a debugger "i/65536.0" in order to see a variable's *real* value. Isn't that a beautiful feeling?

Game developers now can do the bulk of their work in floating point if they so desire, unless they're actively targeting the 486 market (there are rehab programs for those that insist the 486 is a viable game platform). Unfortunately, because of the vast legacy code base, very few of today's games are shipping with floating-point support. But that's not a technical issue; that's a "gotta ship this quarter to generate revenue" issue. You know who you are.

Not for the squeamish or faint of heart, the newest member of the *Game Developer* team rings in with his take on the state of 3D graphics programming.

3D Hardware

By far the biggest news of the past year has been the arrival of real 3D graphics hardware — hardware that is actually faster than software rendering (no snickering, the first few generations of 3D hardware were often slower than software rendering — what were those companies *thinking*?!), sits on the PCI bus, and can actually do some things software can't. This new generation of accelerations, led by reasonably fast and/or feature-laden chipsets such as the Rendition Verite, 3Dfx Interactive Voodoo, and 3DLabs PERMEDIA, are

slowly washing away the unpleasant taste left behind by the first few weak attempts at hardware acceleration.

We're not exactly over the hump just yet, though. Consumers and developers are still suffering through a few pretty lame 3D accelerators that, unfortunately, have managed to get into a lot of computers because of their strong Windows acceleration (and the considerable influence their manufacturers wield). Still, by this time next year, natural selection will have hopefully killed the weaker companies in the herd. Just like watching a lion take down a gazelle on the Discovery Channel, watching the death throes of a company trampled mercilessly by its competitors and misguided stock analysts isn't necessarily pleasant, but it's a necessary part of the life. To be honest, right now this herd we call the 3D chip industry *needs* some thinning.

On a related note, another class of 3D hardware came into being within the past two years — the 3D-oriented console. These include the Nintendo N64, the Sony PlayStation, and the Sega Saturn, all of which raised awareness of 3D technology in many consumers. Jane Consumer might not know the difference between bilinear filtering and point sampling, but she can tell the difference between VIRTUA FIGHTER 2 and SONIC THE HEDGEHOG.

Finally, the other notable event in the world of 3D graphics hardware was Microsoft's announcement of their Talisman 3D graphics architecture. The announcement was greeted by a mixture of ambivalence ("It sucks" and/or fierce yawns) or fear ("Microsoft is doing *hardware*?!").

3D APIs

The arrival of 3D hardware forced developers to deal with a new problem — finding a way to access the capabilities of every chipset without wanting to rip your eyes out of your head. This was a daunting task since register-level programming is very different between different 3D accelerators. Not only that, the feature sets and performance of competing 3D accelerators were rarely similar, so developers had to deal with unpredictable feature sets and performance across different brands of 3D hardware.

Because of the sheer complexity of programming 3D hardware directly, developers turned to card-specific APIs, such as 3Dfx Interactive's Glide, Rendition's Speedy3D, and Videologic's SGL. While making the task of supporting a single chipset reasonably simple, by definition these proprietary APIs do not solve the fundamental problems of different feature sets and vastly varying performance levels of the different chips.

So the industry held its breath, waiting to see if and when a Grand Unifying Library would emerge. This mythical library would perform many tasks, such as software emulation of missing features, feature set reporting, and most importantly, hiding the details of programming a specific chipset.

3D-DDI. The first stab at this was provided by Microsoft, in the form of the 3D-DDI. The 3D-DDI, or 3D-Device Driver Interface, was the 3D analog to the 2D-DDI that interfaces to a Windows accelerator. It was supposed to be the official 3D graphics driver interface, accessible to developers (even though this was officially discouraged), providing a uniform driver back end for OpenGL and for Microsoft's newly acquired retained mode graphics library, Rendermorphics' Reality Lab. The one thing it lacked was a software emulation layer — it wasn't an API so much as a driver interface, and for this and other reasons it died without ever really hitting the market.

Direct3D Immediate Mode. So Microsoft made another attempt at a 3D API, this time with their Direct3D Immediate Mode (D3D IM) library, a part of their DirectX SDK. This API was more feature rich than 3D-DDI and included software emulation of a limited set of features, although not simultaneously with 3D acceleration (a.k.a. "mixed-mode" rendering). It also introduced two new complementary but controversial interface mechanisms — the Common Object Model (COM) and execute buffers, both of which seem to be universally reviled by developers from all walks of life.

Aside from the many technical problems Direct3D has faced, early releases of Direct3D were also plagued by poor management, buggy releases, and lack of proper staffing. Paradoxically, it was both rushed and late, resulting in startlingly buggy software and drivers and extremely bad documentation and sample programs. Bugs in the sample programs often wound up in commercial programs, announcing their existence in a very public fashion. Not only that, but Direct3D indirectly suffered from a rash of problems that had more to do with DirectX as a whole, such as notoriously flaky setup programs that wantonly trashed users' systems. Finally, while this software version of *Waterworld* was running its course in Windows 95 Land, the Windows NT versions of DirectX (including Direct3D) were often unavailable, incomplete, driverless, or only partly functional.

But that was then, and this is now. Things are, in theory, looking up. As I type this column, DirectX 5 is in beta and hopefully will ship as part of Windows 97 (which is looking more like it's going to be Windows 98), easing a lot of the configuration woes users were suffering. Between the release of DirectX 3 and DirectX 5 (they skipped DirectX 4 for some reason), a lot of things changed at Microsoft. Massive staffing changes occurred within the DirectX group, and the new Direct3D team became heavily focused on providing a robust, well-defined, and intuitive API for developers and hardware vendors. The all new immediate-mode interface is far simpler to use than the execute buffers that were rammed down developers' throats last year. And this Direct3D team seems dedicated to providing strong support for hardware vendors.

OpenGL. So Direct3D is the answer, right? Not necessarily. In the dark period between DirectX 3 and DirectX 5, developers began to look for alternatives to Direct3D. During this time, developers learned of "the other graphics API," OpenGL. OpenGL is a portable and open 3D graphics API that runs on a variety of platforms, including Windows 95, Windows NT, Silicon Graphics Irix, DEC Ultrix, HP-UX, BeOS, Apple MacOS, IBM OS/2, and IBM AIX. Unfortunately, OpenGL has had the stigma of being slow, mostly because early implementations on Windows NT *were* slow.

Furthermore, developers assumed that it was targeted at workstations, so you would need a pretty beefy machine to get reasonable performance. But probably the single greatest thing holding back acceptance of OpenGL was that it was primarily targeted towards Windows NT, not Windows 95. It wasn't until 1996 that Microsoft released a version of OpenGL that worked with Windows 95, and even then it didn't have hardware acceleration capabilities. Even today, hardware accelerators are "just around the corner," but we've been hearing that since summer of 1996.

For these and other reasons, most developers discounted OpenGL out of hand. Until one day John Carmack of id Software came along and stirred up a hornets' nest. John did two things that were previously unthinkable. For starters, he publicly denounced Direct3D, calling it "broken," and stated that he had better things to do with his time than trying to get it to work. And, by far the most earth-shattering event, he released an OpenGL-based version of the best-selling game QUAKE. GLQUAKE ran on systems ranging from $2,000 PCs with Orchid Righteous 3D boards all the way up to Silicon Graphics Onyx2/Infinite Reality systems. GLQuake was fast, looked good, had few technical support hassles, and demonstrated, once and for all, that OpenGL could be used just fine for games, thank you. Except for the fact that no one had OpenGL drivers on their Windows 95 machines.

Coincidentally, right about this time, Microsoft announced they were working on a new driver model for OpenGL to replace the old and cumbersome Installable Client Driver (ICD). This new driver model, the Mini-Client Driver (MCD), was supposedly easy to implement, robust, and fast. So all of a sudden it seemed like OpenGL was going to be a viable contender for the hearts and minds of developers.

Today, the two APIs are roughly comparable — Direct3D has a marketing advantage because of far better driver support and because it's Microsoft's anointed games API. OpenGL, on the other hand, has portability, openness, good documentation, ease of use, and John Carmack's seal of approval going for it. Still, DirectX 5 may even the playing field somewhat with respect to documentation and ease-of-use factors.

Time will tell which API will garner more support, but in the short run, both will probably have have limited success, and hopefully neither will die outright. Direct3D is a strategic asset for Microsoft since it locks developers into the Windows market and, unlike OpenGL, it's completely under Microsoft's control. This is good for Microsoft, but probably really bad for us little people. Hopefully, OpenGL won't just die. Besides, Microsoft needs to support OpenGL to attract key UNIX applications to its operating systems.

As it stands today, the industry hasn't seen Direct3D's "killer app," so even though there are more Direct3D titles available, they pale in comparison to GLQUAKE, the only commercial OpenGL game available today on PCs.

A Technical Overview of Direct3D and OpenGL

Now that the history lesson is over, it's time to move on to something less academic — what Direct3D and OpenGL mean to game developers today and tomorrow. For a number of reasons, many experienced game developers, even those with lots of 3D experience, haven't even looked at Direct3D or OpenGL. Some developers are still trying to ship products that were started before Direct3D was even a trademark. Others think that 3D APIs are useless and slow. And a large majority are just waiting to see what's going to happen in the long run before they waste time learning something that may be irrelevant. This last bit is founded in reality, since a lot of programmers struggled long and hard to learn to deal with execute buffers, only to see much of that experience go to waste when the new procedural interface to Direct3D was announced.

So today, two graphics APIs exist that can be used for games. Microsoft's official position is that Direct3D is for games, and OpenGL is…not. However, GLQUAKE confused the situation drastically — here was a visually stunning and fast game that used the not-officially-sanctioned-for-games 3D graphics API. So with that in mind, I'm ignoring Microsoft's rather odd and irrational view that OpenGL isn't suitable for games (proof is in the pudding, as it were), and treating Direct3D and OpenGL as competing APIs, each with its own strengths and weaknesses.

I don't have the space to get into a lot of detail about the technical differences between OpenGL and Direct3D, but I'll try and give a brief overview. I recommend that you look at Chris Hecker's column ("An Open Letter to Microsoft: Do the Right Thing for the 3D Game Industry," *Game Developer*, April-May 1996) on the subject.

Direct3D

As stated earlier, Direct3D is Microsoft's officially sanctioned 3D graphics API for games. It is a part of their overall game development SDK, DirectX, and is tightly coupled to DirectDraw, the 2D component of DirectX. Direct3D has the features you would expect in a 3D graphics API — texture management, triangle rendering, Gouraud shading, texture mapping, Z-buffering, and what not. It also provides the facility to query a driver to see what capabilities are supported. This allows an application to determine what features are supported in hardware and what features may need to be emulated by the application.

Triangle Rendering. Prior to DirectX 5, Direct3D used a data structure known as an *execute buffer* to handle the majority of communication between an application and a Direct3D driver. An execute buffer is a chunk of memory that specifies vertices, triangles, and state information. Execute buffers have been notoriously difficult to use. For example, since they have a specific format that must be adhered to, an application must generally know how many triangles, vertices, and state commands are going to be inserted into an execute buffer before that buffer is created. This involves a lot more bookkeeping than just throwing triangle data at the API. Also, the optimal size of an execute buffer often depends on the particular driver, complicating code even further. Finally, execute buffers are very difficult to debug since their execution occurs completely outside of the application. The most information a devel-

oper can count on is whether an execute buffer failed or succeeded during execution.

The incomplete code fragment in Listing 1 is based on a tech note at the Microsoft Mediadev Web site (http://www.microsoft.com/directx) and renders a screen space Gouraud-shaded triangle by filling in an execute buffer and executing it.

This is a pretty complicated mess just to render a triangle. While it would seem obvious that coding with the above style is not a good idea, it took the release of DirectX 5 before someone decided to address the problem. Microsoft decided to create a new interface, `DrawPrimitive`, which can either replace or coexist with execute buffers. `DrawPrimitive` will be an integral part of Direct3D when DirectX 5 ships, and it bears a strikingly similarity to other procedural APIs — a huge benefit for developers. The burdensome task of managing execute buffers can be removed from the application, and application developers can now spend more time generating data instead of figuring out how to get that data to the API.

The much simpler code snippet in Listing 2 illustrates the `DrawPrimitive` analog of the Gouraud-shaded triangle illustrated earlier.

Data Structures. Microsoft APIs in general define and export many data structures that an application is expected to use, which leads to a plethora of headaches. For example, issues with structure packing and alignment, or differences in the size of enumerated types, can cause grief for those developers brave enough to use a non-Microsoft development environment. An even more significant problem is that later versions of these libraries may implement new features that older structure definitions don't support, forcing the constant introduction of new "extended" data structures.

In the case of Direct3D, the primary data structures that will affect an application are the Direct3D vertex structures, of which there are three: the `D3DVERTEX` for specifying untransformed vertex position and normal information; the `D3DLVERTEX` for specifying untransformed vertex position and color information; and the `D3DTLVERTEX`, which specifies transformed and clipped screen coordinate information. All three vertex structures also specify texture coordinate information.

Since these data structures are tightly packed, for maximum performance an application will likely have to reformat its data into `D3DVERTEX`es either at start-up or on the fly. Neither of these options is particularly attractive to developers. The former often kills code modularity, since a DirectX/Win32-specific data structure is now an integral part of the application, and the latter exacts a monumental performance penalty for games that process a lot of vertices.

Capabilities Reporting. A core feature of Direct3D is its ability to report back the capabilities of a particular Direct3D driver, whether it is a software driver or a hardware device. This reporting feature, handled through capability bits,

was originally considered necessary for games, but in reality it has proven to be of only minor use. When a driver declares the *absence* of a capability, real information is gathered — a game knows what it definitely cannot rely on. However, when a driver reports the *existence* of a capability, this is generally useless information — a feature's mere existence indicates little about its usability or performance. With today's 3D accelerators, we've already seen many examples of this, where a particular feature is supported, but is often slower than software emulation.

Another weakness of capability bits is their inability to inform an application of mutual exclusions and other nonboolean attributes. For example, you cannot have a capability bit that says, "This device supports perspective-correct texture mapping, but only if texture wrapping is disabled," or one that says, "State changes are a bad idea with this hardware, so use Z-buffering instead of sorting. While you're at it, Z-rejection is pretty fast, so try to render bigger and nearer polygons first."

Industry Support. One area where Direct3D has a massive advantage is industry support. Almost every major 3D hardware manufacturer places Direct3D driver support at the top of their list of priorities. For many game companies, supporting any API *but* Direct3D is considered a suicidal strategy. This strength of support has lent Direct3D a lot of inertia, making it highly unlikely that it will go away anytime soon.

LISTING 1. *Triangle rendering using an execute buffer.*

```
// this code lets us lock a previously allocated execute buffer
// so that we can fill it in with the relevant data
memset( &d3dExeBufDesc, 0, sizeof( d3dExeBufDesc ) );
d3dExeBufDesc.dwSize = sizeof( d3dExeBufDesc );
lpd3dExecuteBuffer->Lock( &d3dExeBufDesc );
memset( d3dExeBufDesc.lpData, 0, EXECUTEBUFFERSIZE );

// we fill in our vertices
lpVertex = (LPD3DTLVERTEX)d3dExeBufDesc.lpData;
memcpy( lpVertex[0], &appvertex[0], sizeof( D3DTLVERTEX ) );
memcpy( lpVertex[1], &appvertex[1], sizeof( D3DTLVERTEX ) );
memcpy( lpVertex[2], &appvertex[2], sizeof( D3DTLVERTEX ) );
lpVertex += 3;

// this is the command to process the vertices
lpInstruction = ( LPD3DINSTRUCTION ) lpVertex;
lpInstruction->bOpcode = D3DOP_PROCESSVERTICES;
lpInstruction->bSize   = sizeof( D3DPROCESSVERTICES );
lpInstruction->wCount  = 1U;
lpInstruction++;
lpProcessVertices = ( LPD3DPROCESSVERTICES ) lpInstruction;
lpProcessVertices->dwFlags      = D3DPROCESSVERTICES_COPY;
lpProcessVertices->wStart       = 0U;
lpProcessVertices->wDest        = 0U;
lpProcessVertices->dwCount      = 3; // number of vertices
lpProcessVertices->dwReserved   = 0;
lpProcessVertices++;

// this is the command to draw a triangle
lpInstruction = (LPD3DINSTRUCTION) lpProcessVertices;
lpInstruction->bOpcode = D3DOP_TRIANGLE;
lpInstruction->bSize   = sizeof(D3DTRIANGLE);
lpInstruction->wCount  = 1U;
lpInstruction++;
lpTriangle = (LPD3DTRIANGLE)lpInstruction;
lpTriangle->wV1    = 0U;
lpTriangle->wV2    = 1U;
lpTriangle->wV3    = 2U;
lpTriangle->wFlags = D3DTRIFLAG_EDGEENABLETRIANGLE;
lpTriangle++;

// close the execute buffer
lpInstruction = (LPD3DINSTRUCTION)lpTriangle;
lpInstruction->bOpcode = D3DOP_EXIT;
lpInstruction->bSize   = 0;
lpInstruction->wCount  = 0U;

// unlock the execute buffer
lpd3dExecuteBuffer->Unlock( lpd3dExecuteBuffer );

// execute it
lpD3DDevice->Execute( lpd3dExecuteBuffer, lpViewport, D3DEXECUTE_UNCLIPPED );
```

OpenGL

I'll be honest — I don't think anyone thought OpenGL was a serious contender for game graphics. I mean, *OpenGL*? The 3D graphics API used on quarter-million dollar Silicon Graphics machines? The API that doesn't understand fixed point? The API that wasn't available on Windows 95 until this year?

But against the odds, OpenGL has at least managed to get its foot in the door of the game development community. While few developers have committed to supporting OpenGL, many developers who previously dismissed it without a second thought have come back to give it another look. So, for those of you curious about what OpenGL is like, here's a brief description.

Triangle Rendering. OpenGL has a vast array of mechanisms for rendering a triangle or a group of triangles, but the most common one, and the one most people associate with OpenGL, is the immediate-mode procedural API. No data structures are exported, and as a matter of fact, OpenGL practically

LISTING 2. *Triangle rendering without using execute buffers.*

```
lpD3DDevice->Begin( 0, Vertex, TriangleList ); // begin drawing a triangle list
  lpD3DDevice->Vertex( lpVertex1 );            // vertex 1
  lpD3DDevice->Vertex( lpVertex2 );            // vertex 2
  lpD3DDevice->Vertex( lpVertex3 );            // vertex 3
lpD3Ddevice->End( 0 );                         // end drawing
```

bends over backwards trying to prevent an application from having to reformat its data at all. The code fragment in Listing 3 is the OpenGL equivalent of the Direct3D triangle rendering code presented earlier, without the requirement that the incoming data be in any API-specific, predetermined format.

While the **glBegin/glEnd/glVertex** paradigm is the most common way of specifying a triangle using OpenGL, it also provides other useful methods of specifying triangle data, including display lists, straight vertex arrays (**glDrawArrays**), and indexed vertex arrays (**glArrayElement** and **glDrawElements**).

Capabilities Reporting. Unlike Direct3D, OpenGL lacks the ability to report back what features are accelerated in hardware and what features are not. Instead, OpenGL demands that all implementations *must* provide *all* functionality dictated by the OpenGL specification. This makes software development significantly easier, but at the cost of inconsistent performance across a wide variety of hardware. A game *must* benchmark performance with a variety of features enabled, adding cumbersome code to the game. But unlike Direct3D's capability bits, this will actually generate accurate, empirical data that can be applied directly by the user when faced with a tradeoff between visual quality and performance.

Industry Support. OpenGL has either much better or much worse industry support than Direct3D, depending on which industry you're talking about. If you look at the world of 3D graphics in general, OpenGL is by far the most dominant 3D API in existence. It is available on more brands of CPUs and operating systems and across a greater range of hardware than Direct3D. OpenGL has more thorough documentation available in print and electronic form than Direct3D Immediate Mode, and the number of programmers versed in OpenGL vastly outnumber the number of programmers versed in Direct3D.

However, if you look at the world of 3D graphics in the PC gaming space, OpenGL is at a significant disadvantage. The conventional wisdom, often directly related to the poor performance Microsoft's original implementation of OpenGL on Windows NT, is that OpenGL isn't suitable for games. Most game developers don't take it seriously, and many hardware vendors targeting the consumer space have completely ignored OpenGL. This situation has slowly been improving, thanks in part to GLQUAKE and to the availability of a Windows 95 driver architecture for OpenGL. Still, OpenGL has a long way to go before it can even consider itself a bit player in the 3D API games market for Wintel platforms.

OpenGL is definitely fighting an upward, maybe impossible, battle for acceptance in the game community today. As a game developer, you would be hard pressed to convince a publisher that exclusive support for OpenGL is reasonable. But to ignore OpenGL completely would also be doing yourself a disservice, since OpenGL provides a lot of benefits that Direct3D has never targeted, such as image processing, high-resolution output, and scalability to high-powered graphics workstations. This last one is a potentially huge issue if you need to create custom tools for your artists and modelers — tools that may not run particularly well on a lowly Wintel machine, but may work great on an Intergraph or Silicon Graphics workstation.

LISTING 3. *Triangle rendering with OpenGL.*

```
glBegin( GL_TRIANGLES );
    glColor3fv( vertex[0].color );
    glVertex3fv( vertex[0].position );
    glColor3fv( vertex[1].color );
    glVertex3fv( vertex[1].position );
    glColor3fv( vertex[2].color );
    glVertex3fv( vertex[2].position );
glEnd();
```

Wrapping It Up

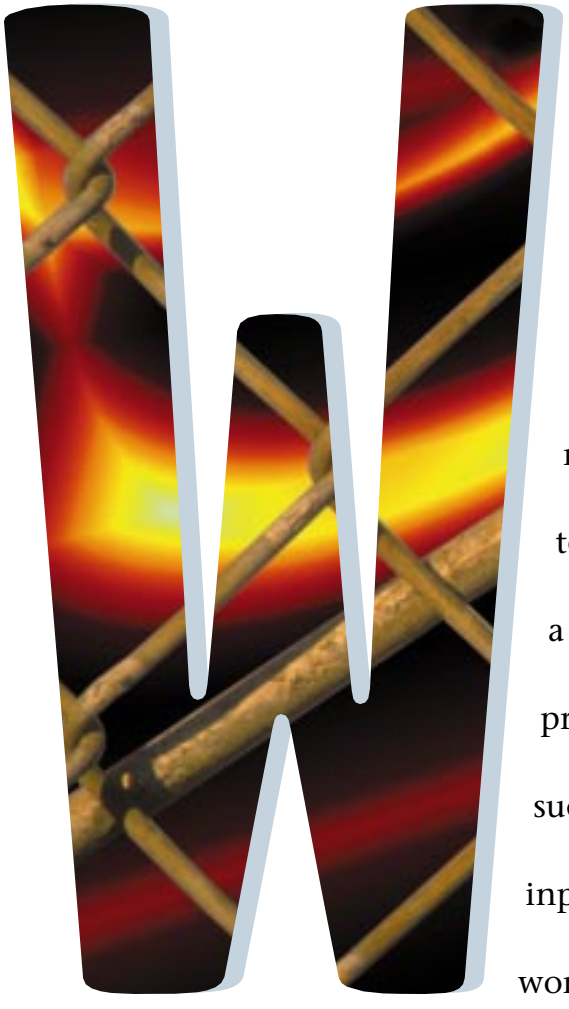
Well, this establishes the 3D graphics scene as I see it today. There are many more differences between OpenGL and Direct3D than I can describe in this limited space, such as texture memory management, lighting models, material management, DirectDraw integration, openness, and portability to non-Microsoft platforms, but for now, the differences outlined here are probably the most relevant to developers who are just now beginning to examine the 3D API situation.

I'm looking forward to learning about the cool new things happening in 3D graphics as they occur, and in turn writing about them in the hopes that we can help each other try to keep up with the rapid advances in the computer industry. And, of course, I hope that those of you with comments, bug reports, criticisms, and corrections will write me and *Game Developer* with your insights and experiences — I'm bound to make mistakes and incorrect assumptions, an unavoidable by-product of the massive changes we're seeing in this industry every day.

In the future, I plan on writing about all manner of things related to 3D graphics for games. There's a lot I want to talk about, from my experiences with Direct3D and OpenGL to accomplishing real world tasks using the 3D hardware and software available today. My columns may be editorials, journalistic reports, academic essays, or workshops (especially workshops!), but in the end my goal is the same — to make sure information is spread instead of stockpiled. It is my sincere hope that anyone who gains from these columns will learn something new and build on it. And, if they are so inclined, turn around and disseminate what they've learned. ■

Brian Hook was formerly an engineer for 3Dfx Interactive and a contractor in the games and semiconductor industries. Today he's a programmer for a small game company known as id software. He'll be working on QUAKE 2 and TRINITY.

SECURITY IN ONLINE GAMES



We are witnessing the biggest revolution in games since the introduction of home computers: online games, played via modem over the Internet. Many companies now insist that every game they develop must have an online component. The most ambitious online games are persistent worlds, which immerse hundreds of players in a single, shared environment. Traditionally, game programmers have faced well-known challenges, such as artificial intelligence, fast 3D graphics, and input devices. In the development of persistent worlds, there is an entirely new set of problems.

BY ANDREW & CHRISTOPHER KIRMSE

Consider the following:

- A single game can last for years. When you leave the game, information about the state of your session is saved. New players can join the game at any time, and old players can stop playing altogether.
- An online game runs simultaneously on thousands of machines. Typically, a central server complex accepts connections and synchronizes communication with client processes running on players' machines. A planetwide network — the Internet — carries data between the clients and the server (in large-scale games, it is impractical for clients to send all data directly to other clients).

It seems that the introduction of every new game is followed by the introduction of web sites dedicated to cheating the game. In one- or two-person games, cheating is a minor issue, since it only affects one or two people at a time. However, a single cheater in an online game can affect thousands of people and have lasting implications.

Of all the issues the online-game developer must be concerned with, security might initially seem like a trivial problem. However, it is central to keeping a system running. A design error in the communication scheme or graphics engine might lead to suboptimal performance, whereas a design error in the game's security is the first step to ruin. In a scenario where customers pay for continued access to the game, letting hackers run free is a sure way to lose a large part of a paying customer base.

Consider the following scenario in a persistent online game. Ben is an avid player of the game, and one day he shows it to his friend Alyssa, who is a computer science undergraduate student. Out of curiosity, Alyssa writes a program to examine all network traffic generated by the game. To further her knowledge, she extends the program to randomly modify pieces of some of the messages the client sends. Unbeknownst to her, the server contains a bug that accidentally incorporates some of this random data into the persistent world and corrupts its database. About a week later, without warning, the server crashes, and the system administrators are forced to rewind the database to the previous week's state. Outraged, thousands of customers cancel their accounts, and the game dies a violent death. Given the damage that a simply curious programmer could cause, imagine what a hacker bent on revenge might do to exploit the smallest security hole.

In this article, we identify areas where security has historically been a problem, and where a client/server system is typically weakest. Figure 1 shows several ways a hacker can attack a game system. We also suggest defensive mechanisms for the most common attacks, and we describe what can happen when hackers get through.

Real-World Adventures

Lest you think that security is just a theoretical problem, consider recent events in some popular games. It appears that Blizzard's Battle.net service contains some major security flaws at the outset. As any experienced DIABLO player on Battle.net will tell you, people have found ways to gather incredibly powerful equipment without earning it. Someone even went so far as to hack the game so that characters in a multiplayer game can be stolen right over the network. In QUAKE, hackers have created automated programs called bots that can automatically destroy opponents. These are free game networks; imagine the amount of effort hackers would exert to attack a pay-by-the-hour system, or if someone offered a Ferrari as a prize in an unregulated QUAKE tournament.

During the development of MERIDIAN 59, we received a shocking reminder of how dedicated hackers can be. At the start of the game, a player can customize a character by assigning numerical values to certain attributes. After the player selects the values, the numbers are sent to the server. Unfortunately, at one point we changed the selection method without changing the check in the server that ensured that the values were legal. Naturally, someone soon modified the client executable to send outrageously large attribute values, and they shared this hack with their friends. Soon we had godlike characters wandering around the world. Our first fix added the values together and checked that the total was under the legal limit. Within a week, someone had found that certain attribute values, when added together, caused the sum to overflow back into the legal range, allowing them to resume cheating. The lesson is that hackers can be extremely clever and persistent, so the lazy solution of security through obscurity is generally not enough.

It's not always the players who are to blame for security problems. MERIDIAN 59 has an administration mode, accessible only by game administrators, that gives complete control over the server. However, during our beta test, a careless administrator learning the system made some poorly considered modifications to the game world. The effects didn't show up until hours later, when suddenly characters' names started disappearing, and monsters began popping up in their inventories (they even attacked the players who were holding them!). We were forced to restore the game state from an hours-old backup. In a commercial system, this would have been quite disasterous. To avoid careless mistakes like this, you should be as restrictive as possible when handing out administrative powers.

Persistent World Architecture

Most persistent worlds share the same basic architectural features. To play the game, a user either installs client software from a CD-ROM or downloads it from the Internet. The client software contains code that communicates with the game server using a custom protocol designed for the particular game. Large static data, such as graphic files, sounds, music, and level layout are typically part of the initial installation onto the client.

The peer-to-peer approach used for most LAN-based games does not scale well to large-scale persistent worlds. When the game lasts longer than a single session, it becomes difficult to deal with players dropping in and out of the game arbitrarily. Performance also becomes a problem, since the amount of game state increases linearly with the number of players, while the amount of network bandwidth remains constant. A simple networked game, such as a shooter, has very little game state — perhaps as little as the coordinates of all the players and monsters, their weapons, and their ammo. This allows each game client to know the entire game state, and makes it possible to transfer to everyone else all the game states changes made by one client.

To solve the problems of a large-scale game, on the other hand, one or more game servers are set up in a central location to keep track of the game state. Each of the game clients communicates its changes exclusively to a game server, which communicates those changes only to the clients that need to receive them (Figure 2). For example, if one user moves his or her character, the server only needs to tell other clients in the user's vicinity. This reduces the bandwidth use of the game to an amount that will not overload users' modem-based connections.

Some virtual worlds last many years after their initial creation. This is one of the benefits of creating a persistent world instead of a transient game. To extend the life of a persistent world, developers often grow the game over time, by adding more areas to the world and new features to the game play. To support this, the client software must have a way to upgrade itself, preferably without any user intervention. This is one of the most important parts of the initial release of any persistent world, because although the world may initially be quite primitive, it has the potential to be improved dramatically over time.

Besides the game servers, there are likely other processes required to make the whole system work in a commercial environment. User account information might be stored in an external database, dynamic game updates might use FTP servers, and automatic mailings could require an SMTP server. These processes could run on the same machine as the game, but for performance reasons, they usually run on separate machines.

Security Precautions

There are two security goals in an online game:

1. Protect sensitive information, such as players' credit card numbers.
2. Provide a level playing field, so that it is as difficult as possible to cheat.

Protecting sensitive information is mostly a matter of configuring the server complex correctly. Each machine connected with the game, such as any web, FTP, or database server, needs to be secured. All game servers should be behind a firewall that only allows data through the ports that the game needs to operate. Finally, the server complex should be located in a locked area, since physical access to the machines circumvents all other security precautions.

As you can tell from our previous example with MERIDIAN 59, some amount of internal security is also necessary to prevent employees outside the development staff from damaging the game or leaking information. In general, administrative powers should be given out strictly to those staff members who require those powers to do their jobs. Once you grant power, it is almost impossible to take it away, so

be extremely sparing. Limit serious power, especially the ability to change account information, to a few trusted, on-site individuals.

Detecting Cheating

A good first step in foiling cheaters is to set up the system to detect cheating automatically. Record all player logins and logouts, as well as important game events such as player advancement. Keep track of key quantities in the game, such as the total amount of money and numbers of rare items in existence. If you review these statistics regularly, you can tell when there is a major problem, such as the appearance of multiple copies of items that are supposed to be unique, or the overnight doubling of the money supply.

Also, talk to your players occasionally. They probably have more experience actually playing the game than even the game designers do, and they are anxious to ensure that other players don't have an advantage over them. If they suffer at the hands of a cheater, they will complain to you loudly. On the other hand, since players by design have a limited understanding of how the system works, they tend to report far more incidences of cheating than actually occur. Consider creating an e-mail account specifically for players to report instances of cheating. Be aware that sifting through all the complaints can be a full-time job; we recommend waiting until a problem is reported at least several times before spending time to investigate it further.

Attack and Defense

Clearly, the consequences of a security hole in an online game are much more serious than in a standalone game. A malicious player might use a security hole to cheat or to compromise the integrity of the game. Letting hackers run free is a sure way to lose a large part of a paying customer base, so the game design should address this problem from the beginning. There are three places in the game that are vulnerable to attack: the data files, the client/server protocol, and the client executable itself.

A simple approach to securing data files on client machines is to encrypt them. Since there is a performance penalty at run time for decrypting files, however, only those files that contain important information need encryption. There is probably little or no harm in letting a player examine and modify music, sound, and graphic files, whereas access to levels or speech files might give someone a substantial advantage.

Encryption is not quite enough, though. Merely renaming or copying one data file over another might allow someone to fool the client into thinking that a player is in a location other than where the server places the player; this could have consequences ranging from odd player behavior on other people's machines to granting the player access to otherwise restricted areas. The solution is to have the server verify that the client is using the correct file data, not just the correct file name. When the client loads a file, it can perform a checksum of the file and send the checksum to the server (given that the file is encrypted, the checksum can be quite simple and inexpensive to calculate). If the server finds that the checksum doesn't match its expectations, it concludes that the player has cheated, and either logs this information or takes immediate corrective action. Note that it's not enough for the server to simply ask the client to deal with the problem, since a malicious user might block such a message.

In fact, there are many possible attacks on the client/server protocol.

FIGURE 1. *Possible Attacks.*

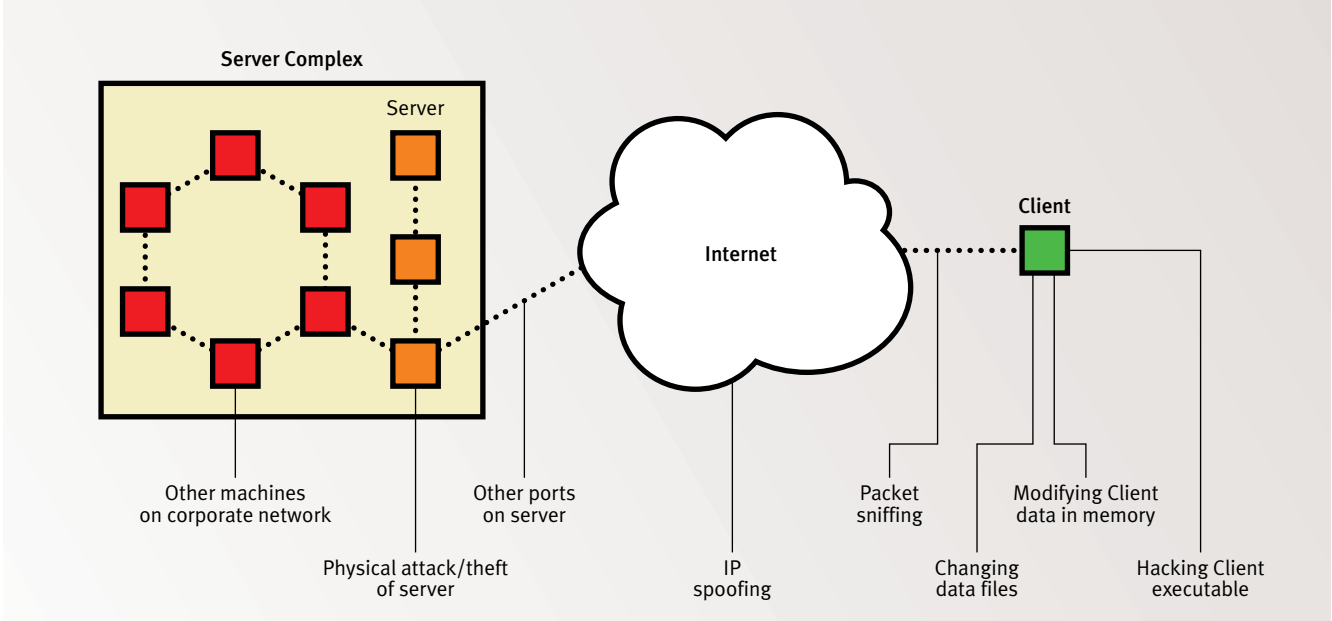
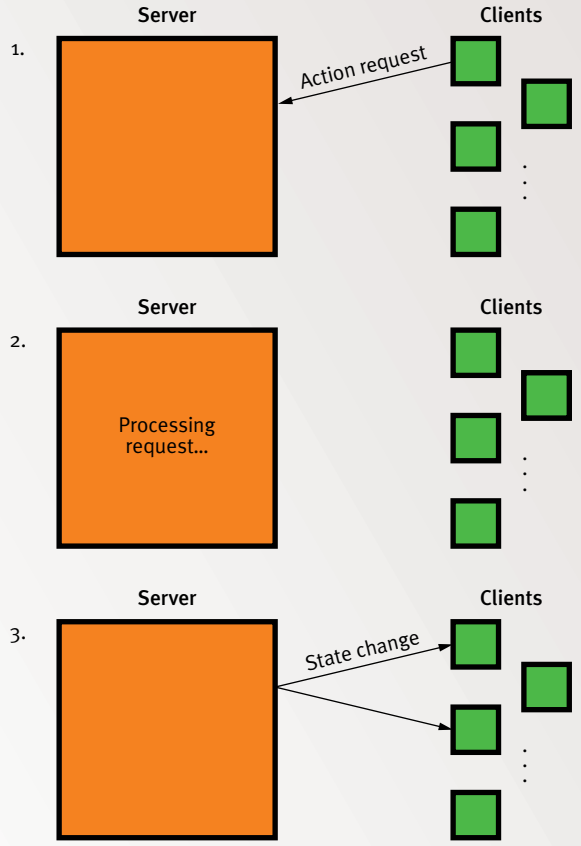


FIGURE 2. *Client/Server Architecture.*



It's fairly easy for a hacker to use a packet sniffer, which is a program that displays all data transferred over the network (many such commercial programs are readily available). Armed with a sniffer, a hacker might try to reverse engineer the client/server protocol with an eye to changing packets as the client sends them. Encrypting the protocol effectively solves this particular problem, but there are others. Even when packets are encrypted, a hacker can capture an outgoing packet and resend it, possibly hundreds of times (an attack called "replaying packets"). If the packet is a request to fire a spaceship's lasers, replaying packets could give someone a significant advantage by making the ship's lasers fire rapidly.

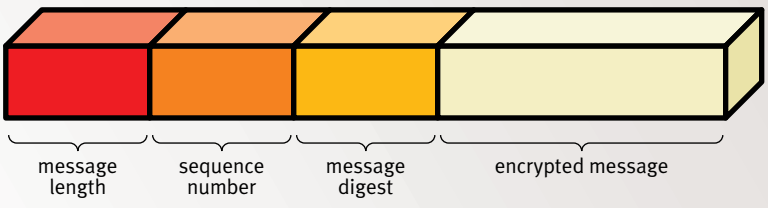
A good way to thwart packet replay is to assign each packet a sequence number, which changes from one packet to the next. If the server receives a packet with the wrong sequence number, it knows that the sender is cheating. Of course, the sequence number should be more than just an integer that increments with each message; that scheme is far too easy for a hacker to replicate. Instead, the sequence number could be a mix of anything that acts as a state variable for the protocol. For example, it could be the total number of bytes that have been sent since the client started. See Figure 3 for a diagram of a typical packet.

Something else a cheater might do is use a packet sniffer simply to block certain messages from reaching the client or the server. Assume for argument's sake that the player's health value is stored on the client, and that messages that inflict damage are blocked from reaching the client. The player blocking damage messages would become invulnerable. By storing all important data on the server, it's possible to prevent message blocking from giving anyone an advantage. However, message blocking attacks can be subtle, and each protocol message needs to be examined to see what would happen if someone blocked it.

The client executable resides on the player's machine, and thus is vulnerable to tampering. Worse, the client possesses important information when it runs, including the contents of data files, which must be decrypted as they are loaded. A cheater might use a debugger to view memory while the client is running, or even modify the executable to disable other security checks. Unfortunately, there is little defense against such attacks. Any user sophisticated enough to modify the executable will probably be able to get around any attempts to prevent tampering. One way to contain the damage is to limit the client's knowledge of the game as much as possible. After all, a hacker can only learn as much as the client knows. By storing data on the server, a network roundtrip is required to retrieve it, which hurts performance and complicates the code. Data security is a balancing act between performance concerns and the danger that players may find a way to learn all of the information in the client.

Another possible security problem is the use of automated programs called bots to simulate user actions. Such programs are easy to write in a windowing environment, since external programs can send arbitrary window messages to the game client. A common use for a bot is to perform

FIGURE 3. *Protocol Message.*



an action repeatedly, faster than a person ever could. To thwart bots, the client should require that a certain amount of time passes between two successive user actions. Depending on the particular operating system, it might be possible to do more to discourage bots by restricting message sources (for example, you can detect window messages sent by external programs under X Windows). On the other hand, some game designers might consider bots useful convenience features for certain purposes, and might even encourage their use. As an illustration, a player has written an add-on for MERIDIAN 59 that expands the number of available alias keys by simulating user input.

Dealing with Cheaters

What should the server do when it detects a security problem? One answer is to immediately disconnect or possibly suspend the offender's account. Suspension might be too severe — even one misdirected account suspension is a serious customer relations problem. On the other hand, simply disconnecting an account has the disadvantage of letting the hacker know that his or her attempt has failed. A more subtle approach is simply to log the security breach, and periodically review the security log. This approach discourages casual hacking at the cost of letting a few security violators through for a short time. It is best suited for less severe security breaches, where this cost is acceptable.

For more severe security problems, you should be relentless in removing cheaters from the system. Although the number of people with the technical knowledge to hack your system is probably small, there are many more who are willing to use other people's hacks. In a commercial system, the terms of service should disallow tampering with the system, so that you can disconnect serious cheaters instantly. It might be more difficult to keep cheaters out in a free system, because cheaters can just keep signing up.

When you find a bug or security hole that cheaters have exploited in a persistent world, you often have to write code to set things right again. In MERIDIAN 59, there were several instances where players found obscure ways of getting free money in the game. Each time, we had to write special-purpose code to find large stashes of money and reduce them down to reasonable sizes. Such code is error-prone, because there are many special cases, and because it is impossible to tell whether the money was actually obtained illegally. Be sure to test corrective code extensively. We actually set up special testing servers with external players to deal with problems like these.

A fact of life in the security business is that no defense is foolproof. Given enough time and money, an attacker can always defeat any security scheme. A hacker can disassemble and rewrite your entire client, dedicated hardware can break your encryption scheme, or a terrorist can drive a tank through the door to your server room. Your job is to make cheating prohibitively expensive, so that potential attackers have little desire to try. You have to judge for yourself how secure is secure enough, taking into account what's at stake if someone breaks through. Also, keep in mind the fact that the time you spend beefing up security is time that you could have spent improving the game in other ways.

Applying What We've Learned

Let's apply the principles above to a concrete example of a near real-time action game. All persistent data is stored on the server, and in most cases, the client sends a message to the server each time the player requests an action. The one exception is motion; to provide a realistic environment, the game must show the player moving as soon as a key is pressed. This one fact has important consequences for the game design and the protocol.

A hacker who finds out that the client is sending motion messages might try to modify the messages to make his character move more quickly. Encrypting the protocol should be enough to prevent this attack. However, the hacker might record outgoing motion packets, and then resend them later, achieving the same effect. Adding a sequence number to each message should stop this problem. For added security, the server can attempt to validate player moves. Simple checks, such as calculating the distance between two successive moves, can find the most serious cheats. Another alternative is to perform full collision detection to validate every incoming motion message. In many cases, collision detection is too expensive to perform on the server, but in some games it is possible. Where it is too expensive, the server could verify only a randomly selected sample of moves; this can even be done offline or by a separate process on another machine.

In a strategy game, it isn't as important for motion to take place in real time (consider a turn-based game like CIVILIZATION). In these cases, it might be acceptable to send every move to the server, which would verify the move and send back a reply if the move is legal. If the game is too slow when every move generates a network round trip, moves can be batched into larger groups and sent to the server for verification all at once. This is a good example of how security concerns can affect a game's performance.

Take It Seriously

Security is a serious concern in all online games, but especially in persistent worlds. A single security hole can make your customers disappear overnight. As more games move to the Internet, we'll hear about security more often, since players will break through the weak protection in most games. With proper planning and eternal vigilance, however, you can avoid becoming a casualty of the online revolution. ■

Andrew and Chris Kirmse are the inventors and primary developers of MERIDIAN 59, one of the first graphical Internet games. They have been writing computer games together since 1982. Andrew holds degrees in physics, mathematics, and computer science from MIT. He can be reached at andrew@invasiongames.com. Chris graduated from Virginia Polytechnic Institute in 1996 with a degree in computer science. His e-mail address is chris@invasiongames.com.

Octree Partitioning Technique

Octree Space Partitioning (OSP) algorithms are used for the correct representation of solid objects in a 3D environment, and are the basis for many modeling and rendering systems. The primary objective of the OSP is to reduce the number of comparisons required to determine which surfaces in a scene need to be processed for ray tracing, collision detection, visibility determination, or similar calculations. Although primarily used in static environments, they can also be used in dynamic environments with a few modifications. Octrees can provide a significant reduction in the time needed to sort polygons in a scene for proper display and are ideal for high-performance games that consist primarily of empty space where the objects within this space show large variations in relative size (for example, flight simulators).

In this discussion, we will review the two canonical forms of the OSP:

1. OSPs that operate on objects in the scene.
2. OSPs that operate on the volume of space comprising the scene.

Both types of octree algorithms store their data in a tree structure made up of three or more child nodes per branch. The resulting data structure can be traversed in three dimensions, unlike the Binary Space Partition (BSP), which can only be traversed in one (for more information on Binary Space Partition trees, see "Advanced Binary Space Partition Techniques," *Game Developer*, October/November 1996).

In contrast to the construction of a binary tree (where the surfaces in the scene are inspected and assigned to either the left or right node, depending upon which side of the partition plane they are located), with octrees you assign to the nodes of a tree not the surfaces themselves, but the *volumes of space* that contain surfaces.

Hierarchical Bounding Volumes

The hierarchical bounding volume form of the OSP algorithm stores its data in an n-tree structure. An n-tree differs from a binary tree in that its parent nodes have varying numbers of child nodes, rather than having either zero or two child nodes. The root node of the tree contains all of the objects in the scene. Each branch node going away from the root contains fewer and fewer objects, until you reach leaf nodes, which contain only a single object or surface. Figures 1 through 3 illustrate the steps to the construction of the tree and is explained in the following text. The purpose of this algorithm is to manage and sort information about the objects in the scene relative to each other, minimizing the time needed to render the surface in the correct order: front-to-back or back-to-front.

Creating the Tree

To create the initial tree, obtain the maximum dimensions of the scene that is occupied by your surfaces, then adjust the dimensions to form a cube.

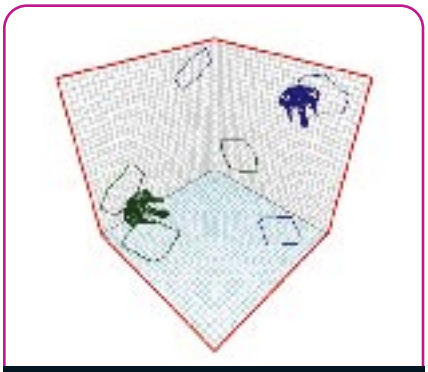


FIGURE 1. *The volume of the root node.*

This is the bounding cube for all the surfaces, and it is assigned to the root node of the octree. Figure 1 represents two spaceships in a volume of space. Note the use of shadow outlines against the background grids to make it easier to see the position of the ships. Figure 1 is the root node of the tree that holds the entire volume.

Next, select a subset of the surfaces based on an object hierarchy construction algorithm (OHC). This algorithm determines which surfaces are selected, in what order they are processed, and how many are allowed in each subset. You can choose from a number of different algorithms, depending on the needs of your game. In this example, I'll use a simple OHC that selects and groups surfaces based on the volume they occupy. The OHC randomly selects any surface from the scene, which I'll call the "seed surface," as it's the first surface from which a single branch of the Octree grows. The algorithm then finds the nearest surface to the seed surface. If both surfaces fit within a volume of space less than a predefined limit, then both surfaces belong to the same set. We call this predefined limit the "limit volume." In this example, we'll use a simple limit volume that is just a bounding box; more complex shapes can be used, depending on the characteristics of the game. The initial limit volume is usually large enough to contain the largest free-standing object in your scene.

You continue adding the nearest surfaces to the set until the total volume of the set reaches or exceeds the limit volume. All of these surfaces are assigned to a child node of the tree. This node also stores the coordinates of the volume occupied by these surfaces. Select another seed surface from the remaining surfaces in the scene and repeat this procedure until all surfaces in the scene belong to a set and are included in the tree. You have now completed the first level of the tree.

FIGURE 2. *Limit volumes forming nodes on the tree.*

Figure 2 shows the limit volumes around each of the ships. Note that the orientation of the left ship causes it to occupy a larger limit volume than the right ship. This difference is caused by our selection of a bounding-box OHC algorithm; other algorithms can provide tighter fits. Figure 2 is the tree with nodes for each of the ships.

Now we reduce the size of the limit volume, usually by splitting it in half, and recursively repeat the process. Each branch of the octree now points to a single set of surfaces. You repeat the seed-surface selection and subset creation for each of the branches of the octree, which adds a new level to the tree. Figure 3 shows the smaller limit volumes around components of the ships. For clarity, Figure 3 shows only two of the lowest level limit volumes for each ship.

Repeat this process recursively until the limit volume reaches the size of the smallest surface, or until every surface occupies its own branch of the tree. You may have to split some surfaces to accommodate the single surface per branch rule.

If you allow the algorithm to continue until the limit volume reaches the size of the smallest surface, your scene will display correctly, but it may take too long to create the octree. Using a limit volume that allows more than one surface into each branch improves performance, but sacrifices final quality. In this example, I used a bounding box limit volume, which improves performance of colli-

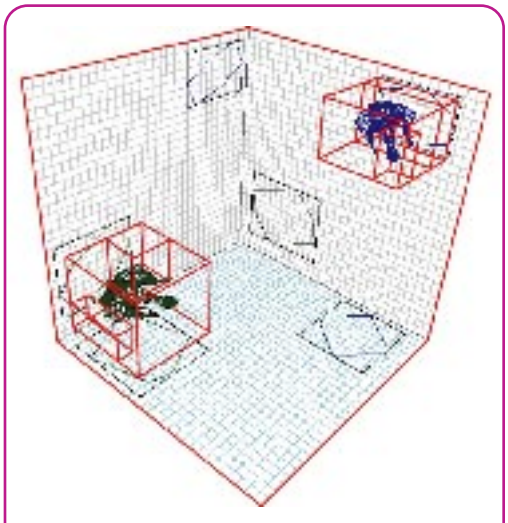


FIGURE 3. *Further reduction of the limit volumes*

IN MANY SPACE-BASED GAMES AND FLIGHT SIMULATORS, OCTREES ARE IMPORTANT IN MODELLING AND RENDERING. by Mike Kelleghan

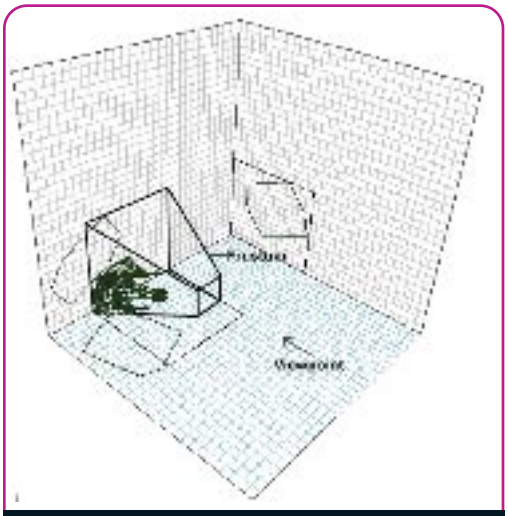


FIGURE 4. *The camera's viewing frustrum.*

sion detection and object motion, but reduces that of ray tracing.

How you select the seed surfaces will determine the number of child nodes at each level of the tree. Ideally, you want a seed surface to be at the center of a cluster of surfaces in order to reduce the number of nodes in a level. Rubberbanding algorithms that find the center of a cluster are very useful in selecting an optimal seed surface. Seed surfaces that are too isolated from their neighbors generate too many nodes at certain levels, which in turn increases search times when doing collision-detection or ray-tracing calculations.

Using the Tree

Once the n-tree is complete, you have created a data structure that describes the entire scene in an efficiently increasing level of detail. Now you can use this tree to determine which surfaces are visible from any given camera position. First, calculate the viewing frustum of the camera (Figure 4). Then, determine whether the volume of the viewing frustum intersects with the volume described by each of the branches of the Octree that represent a bounding box (Figure 5). If the two volumes intersect, then recursively follow each branch, checking whether the child nodes intersect the frustum. Move down each node in the tree until you reach the leaf nodes. You now have a path through the n-tree that contains only those surfaces that are included in the frustum (Figure 6). These surfaces can now be fed to a Z-buffer or other rasterizer.

This technique also can be used to obtain line-of-sight calculations.The line-of-sight is treated as a degenerate frustum or is tested for intersection with a volume. Collisions can be calculated by treating the path of an object as a line through the scene and applying the line-of-sight technique.

Handling moving objects with octrees is a little more awkward. If you include moving objects in the algorithms described previously, you will spend a lot of time adjusting the subsets of the tree to include or exclude the moving object from a particular node. Depending on your application, it may be faster if you consider each moving object as a separate and independent leaf node, regardless of the size of the volume it represents. This moving-object leaf node can then be attached to a static node as it moves through the scene.

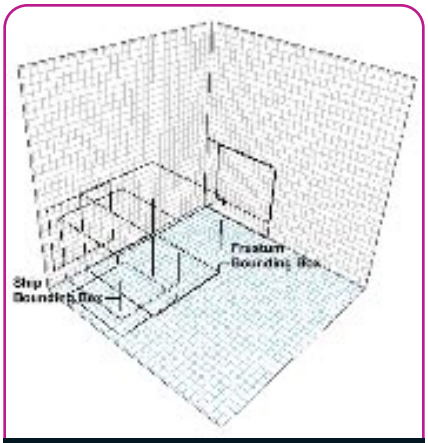


FIGURE 5. *The intersection of the viewing frustrum with a bounded volume described by a node on the Octree.*

Canonical Space Subdivision

The Canonical Space Subdivision form of the OSP algorithm is ideal for applications where most of the space is occupied by objects, and the objects are all approximately the same size (for instance, in maze-type games). The purpose of this algorithm is to manage spatial information about the scene using an octal-tree structure (in which each parent node has eight child nodes) to store its data.

The root node of this tree contains all of the surfaces in the scene. Each branch node contains an octant (one-eighth) of the parent space until you reach a level where each octant is the size of the smallest surface in the scene.

Creating the Tree

To create the initial tree, first obtain the maximum dimensions of the scene that you wish to manage. Adjust the dimensions to form a cube. This is the bounding cube of the entire scene and is assigned to the root node of the tree (Figure 1).

Next, subdivide the bounding cube into eight smaller cubes, known as cubelets. For clarity, Figure 7 shows the cubelets represented by red lines against the background grids. Each one of these cubelets will be held in a child node. If the cubelet contains any surfaces, assign the surfaces to the node. If a surface straddles more than one cubelet, split the surface into subsurfaces. Store the coordinates of the cubelet in the node. Assign an identification number to each cubelet based on the octant it represents (the upper-left-nearest cubelet is assigned number 1, the next cubelet number 2, and so on).

Repeat this procedure (Figure 8) until the size of the cubelet reaches a prede-

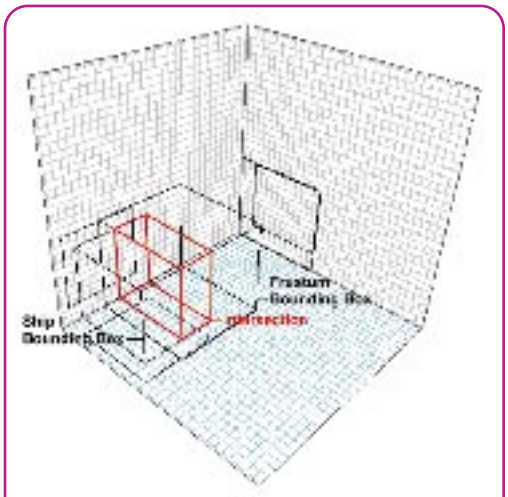


FIGURE 6. *Viewing path cutting through the n-tree.*

fined lower limit (usually the size of the smallest surface), or until each cubelet contains a single surface. The tree is now complete.

Using the Tree

Once your tree is complete, you have a data structure that describes the entire volume of space in a regular, hierarchical form. You can use this data structure to determine which surfaces are intersected by a given line, say the path of a missile or the viewer's line of sight (Figure 9). Obtain the two intersections of the given line with the root cube of the tree, and then select one of these intersections as the start position of the algorithm. Traverse the tree until you find the smallest cubelet that contains the start position. If there are any surfaces in the cubelet, test for intersection of the line with the surfaces in the cubelet.

Using this starting cubelet and the 3D slope of the line, you can obtain the next cubelet that the line passes through using a 3D variation of Bresenham's algorithm. This variation is simply the same 2D line-drawing algorithm extended to 3D and is termed a "3D Digital Differential Analyzer" (3DDDA). Repeat this procedure until you reach the other endpoint of the line.

You can extend this technique to determine what surfaces are inside the viewing frustum. First, obtain the eight endpoints that define the limits of the viewing frustum. Then, obtain the slopes of the lines defined by the points. Traverse the tree until you find the smallest cubelets that contain the endpoints of the lines. Select the four cubelets that contain the endpoints of the lines that define the near clipping plane of the viewing frustum. Using the slopes of the frustum lines and the endpoint cubelets, apply the 3DDDA to obtain all the cubelets through which the near clipping plane passes. Use these cubelets, the remaining slopes and endpoints, and the 3DDDA to obtain the remainder of the cubelets in the frustum. Pass all the surfaces contained in the cubelets to the rasterizer.

Handling moving objects is fairly straightforward. If you set the predefined lower limit of the cubelet to an integral factor of the smallest distance an object can travel in the world, keeping track of which cubelet contains the object becomes a simple matter of applying the 3DDDA.

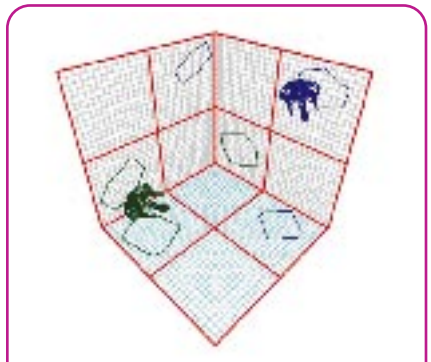


FIGURE 7. *Dividing the volume into cubelets.*

FIGURE 8. *Further subdivision into cubelets*

Using the octree algorithms as the centerpiece of a game lets you display true 3D objects with a high degree of visual precision and performance. Using these algorithms correctly can give you enough extra horsepower to add a little transparency, or real-time physics, providing the difference between just another game and a beautiful work of art. ■

Mike Kelleghan builds 3D real-time articulation engines for various game companies in Los Angeles, and is currently recovering from a case of tendonitis caused by too much joysticking on flight simulators. He has, so far, been unsuccessful in convincing the insurance company that it should buy him a more ergonomic joystick.

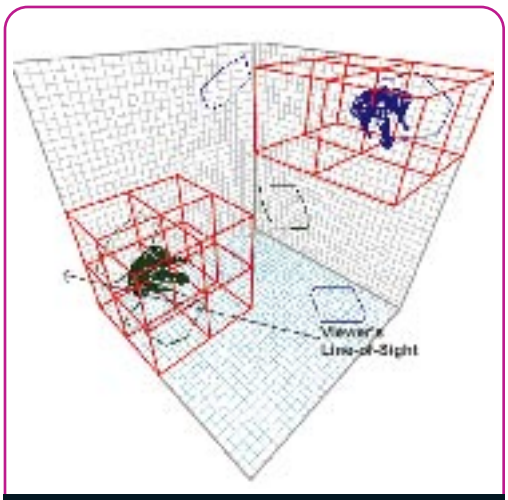


FIGURE 9. *Determining line of sight in an Octree structure.*

FOR FURTHER INFO

C++ Object-Oriented Data Structures, Sengupta S, Korobkin CP, Springer 1994, pp. 659 - 674.

"On Visible Surface Generation by a priori Tree Structures," Fuchs, et al, *Computer Graphics* (SIGGRAPH '80 Proceedings), 14(3), pp. 124-133.

"Near Real-Time Shaded Display of Rigid Objects," Fuchs, et al, SIGGRAPH '83, pp. 65-72.

"Linear Time Voxel Walking for Octrees," James Arvo, *Ray Tracing News* 1(2). E-mail edition available under anonymous FTP from weedeater.math.yale.edu.

"Fast Ray Tracing Using K-D Trees," Fussel and Subramanian, Technical report TR-88-07, Dept. of computer science, The University of Texas at Austin, 1988.

"Heuristics for Ray Tracing Using Space Subdivision," MacDonald and Booth, *Proceedings of Graphics Interface '89*, pp. 152-163.

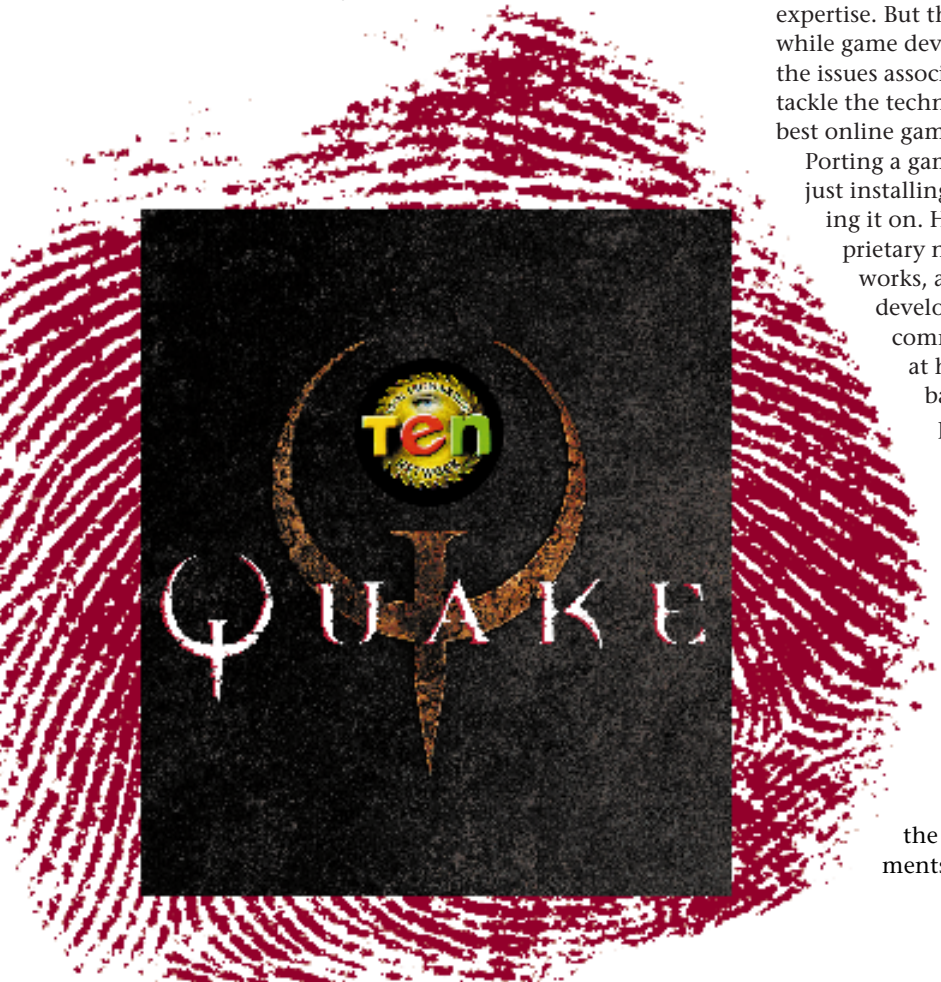
"Automatic Termination Criteria for Ray Tracing Hierarchies", Subramanian and Fussel, *Proceedings of Graphics Interface '91*, pp. 93-100.

The mother of all ray tracing web pages, with more stuff on Octrees than you can read in a year, can be found at www.cm.cf.ac.uk/Ray.Tracing/

CASE STUDY

Ho QUAKE Wa Po ed o TEN

by Ned Purdom



Whether played across LANs, the Internet, or on commercial gaming services, multiplayer gaming is hot. For the gamer, what better way to express competitive impulses than to establish Web-wide bragging rights? For the developer, online play offers a means of extending the life of a particular title, realizing royalty revenue, and enhancing retail box sales. As a result, many developers now offer titles with some level of online support.

While the online capabilities provided by game developers are usually a sound basis for online play, game networks are able to optimize these games by taking advantage of rapidly changing networking infrastructure, technologies, and expertise. But this is a matter of focus as well as expertise; while game developers have their hands full dealing with the issues associated with title development, game networks tackle the technical and social challenges of creating the best online gaming experience.

Porting a game to a commercial network isn't as simple as just installing the game onto a network server and turning it on. High-performance server hardware and proprietary network APIs are commonplace to game networks, and these present hurdles for game developers who are looking to host their game on a commercial service. In this case study, we'll look at how one game network, the San Francisco-based Total Entertainment Network (TEN), ported id's QUAKE to their network.

Evolutionary, Extensible Game

Many aspects of QUAKE make it compelling from a game play standpoint, and quite challenging from the perspective of a developer getting it ready for online play on a managed network service. The game itself is quite open, fostering continuous evolution. id Software spurred this evolution by providing their own programming language, QuakeC.

While QUAKE is online-ready out-of-the-box, the developers at TEN felt that several enhancements were warranted. First, they wanted to

accommodate a broad spectrum of online gamers — those with relatively slow 14.4 baud modem connections through those fortunate enough to have high-speed T1 lines. Equally important, TEN wanted QUAKE players on their service to have a similarly consistent game-playing experience from wherever they happened to connect.

As TEN began bringing the title online, it was apparent that the major areas of technical focus would be porting QUAKE to TEN's network infrastructure, optimizing gameplay performance, and providing value-added or unique elements to differentiate the TEN QUAKE experience from other online efforts. At the same time, the company knew that they had to make readily available those constantly evolving elements of QUAKE that the playing public demands.

Tier 1 ISP Service

Before actually porting QUAKE to its network infrastructure, the developers at TEN first had to determine what architecture would best support the game. After considerable deliberation, it was decided that a forked server approach would provide the best gaming experience. In this scenario, multiple arena servers are set up and a separate instance of the game is created within one of the arena servers when a new game is started. Once this architectural approach was decided, the process of moving QUAKE to the network proceeded.

Developing a low-level network transport for QUAKE was the first requirement in the porting process. Since TEN requires Windows 95 on client machines, and QUAKE is a DOS application, a special interface had to be provided to transfer UDP data between the DOS application and Windows' WinSock.

QUAKE uses the UDP Internet protocol as its networking backbone. A slightly older version of the commonly used TCP/IP protocol, UDP offers several advantages for game play. With UDP, each packet is transmitted one time, but there is some packet loss associated with the protocol. Conversely, TCP/IP retransmits packets to ensure reliability — a key consideration in mission-critical networking. What UDP lacks in reliability, it gains in performance, which is certainly a major issue for gamers. With a lot of packet loss on the network, UDP is more effective for game play because it does not try to resend stale data.

By using the TEN datagram library for DOS, the company was able to provide this data transfer using a virtual device driver and a hidden Windows application. The datagram library allows a DOS application to use the UDP networking protocol within Windows 95. The development of this special code was one of the first efforts undertaken. The code to implement this can be found on the *Game Developer* web site.

Porting Quake to Solaris

Providing enough server resources to accommodate peak playing loads was the next concern. TEN anticipated that up to 700 players would play QUAKE in a given day, with some 140-150 playing simultaneously. This load was certain to tax QUAKE's DOS/Windows/NT server environment.

The TEN service — for all games — is based on the Sun/Solaris operating system running on ultraSPARC servers. This platform was initially selected for its superior performance and because TEN felt it could provide better support if it only had to manage one operating system.

Not only does the Solaris/SPARC platform offer the performance necessary to handle the anticipated load for QUAKE play, but equally important, it provides the scalability to accommodate growth in QUAKE play, as well as TEN's overall subscriber base. So the decision to port the DOS QUAKE server to Sun's Solaris operating system was an easy one to make.

The process of readying QUAKE for TEN's network infrastructure and porting the game to the Solaris operating system was conducted by four of TEN's engineers. The total process required about three months. The first month was devoted to setting up a build environment similar to id's. Once the source code was obtained from id, the porting process, testing, and optimizing required another two months, and was conducted exclusively by TEN.

Because of the inherent quality of QUAKE, including its modular architecture, network readiness and extremely clean code, the game engine itself was virtually unchanged as it became part of the TEN service. As a result, TEN developers focused primarily on optimizing data transfer to improve game performance, and adding features for enhancing network play.

Once complete, TEN forwarded a CD of its game version to id. The executables were downloaded and id staffers played QUAKE on the TEN service. At this point, the game was approved by id.

According to developers at TEN, id's role was limited to providing source code and approving the final game. However, they note that because QUAKE was such a good starting point and an Internet-ready title, TEN's role was one of integration and enhancement. TEN notes that in some cases, network capabilities must be added or code cleaned up so that a title is suitable for online play.

Enhancing the Game Experience

Once QUAKE was ported to the network infrastructure, TEN realized that attracting and keeping players was the next — and ongoing — technical challenge. Part of this challenge was providing a unique, inviting, and evolving environment for an extremely broad range of QUAKE players. An equally important challenge was optimizing QUAKE performance, player by player, during actual online play. This required striking a balance between the design and game playing elements specified by id Software — logos and other art — and those added for the TEN service.

A major concern was making the service easy to use for the new player, while providing flexibility and extensibility for the more experienced QUAKE gamer. The opening game screen allows players to chat among themselves prior to establishing a game. When ready, even a game involving several mods can be started with a few mouse clicks, as opposed to command lines common to standard Internet play.

Rankings & Tournaments

Inherent in online play is a desire among participants to play against others; a natural outfall from this competition is player rankings. Rather than simply construct a database ranking players by wins and losses, TEN extended the QUAKE server to gather a stream of data from each game. This information includes what weapons were used,

who shot whom, percentage of success, and so forth. With these rankings, a QUAKE community grew, not unlike the communities associated with statistics-driven sports rotisserie leagues.

Optimizing Game Play

Game developers conduct quality assurance throughout the development process. While these tests address the stability issues associated with the game, TEN needed to get real-world insight on high-load, multiplayer gaming. Consequently, TEN enlisted hundreds of presubscribers to stress test the entire service, the user interface, and early versions of QUAKE and other titles.

During early product testing, evaluators reported performance problems, especially at modem speeds below 28.8 baud. The company considered this a serious issue given the large number of gamers with 14.4 baud modems. Evaluators noted a "skating" feeling during play, where they did not have complete control. Other manifestations included sound dropping out, which prevented players from hearing each other's approach.

The QUAKE server was continually broadcasting complete game updates to clients during play. It was apparent that there was simply too much data for slower modems to handle. TEN carefully studied bandwidth profiles during game play and determined that the only way to provide consistently good performance on slower modems was to fundamentally change the way that QUAKE sends data to players.

The result was a new capability called Dynamic Attribute Reduction Technology (DART). With DART, the TEN QUAKE server keeps track of what each and every client knows at each point during a game. For instance, the server keeps track of whether a player has opened or closed a particular door, and avoids resending that information until the object has changed its state. DART filters out redundant (unchanged) data and permits the server to broadcast only changed information. In effect, DART optimizes the game for each particular player continuously throughout a game by reducing the information sent to players by up to 70%. Furthermore, DART allows TEN to manage server loads better and handle more players on the existing servers. See the code examples on the *Game Developer* web site, which show how DART is typically implemented.

Moving forward with QUAKE and other online titles, it's clear that the major technical issues for service providers such as TEN are how best to maintain the original spirit of the game, maximize the online experience of each supported game, keep pace with the evolution of online games, and do so in a simple-to-use and reliable manner. ■

Ned Purdom of San Anselmo, Calif., has been writing about technology for more than 15 years. He has profiled a number of leading game developers, including Sega, Trilobyte, Mechadeus and Crystal Dynamics. *The author wishes to thank TEN engineers Howard Berkey, Norman Morse, and Bill Lipa.*

Activision Finds Rebirth in the Apocalypse

Activision was the original third-party console developer, rising to hundreds of millions in sales on the back of the Atari 2600 VCS, as well as Mattel's Intellivision. Games like RIVER RAID, PITFALL, and KABOOM! were what brought Activision to the peak of the industry in the early '80s.

After falling on hard times, the company has come back from the brink with recent hits such as ZORK NEMESIS, PITFALL II, MECHWARRIOR, and EARTHWORM JIM. Those titles, plus a promising lineup that includes QUAKE level packs, HEXEN II, INTERSTATE '76, and DARK REIGN have made this developer one of today's consistent hitmakers.

Activision's APOCALYPSE is the company's first internally developed PlayStation title. It's the kind of 100% pure-action game that made gamers fall in love with older 8- and 16-bit consoles in the first place. Despite its name, APOCALYPSE signifies a beginning — the start of a round of PlayStation development by Activision.

The APOCALYPSE team: (left to right) John Spinale, Director; Danny Matson, Art Director; and Michael Kirby, Producer.

A basic story of Sony PlayStation development seen through the eyes of Activision's first in-house console product game.

Facing the Reality of Console Development

For many developers, the biggest test of console development is deciding whether they're up to it at all. Unlike PC development, the startup costs and factors of console development are an altogether different ballgame. Costs rise, talent pools shrink, demographics shift, and the market dynamics are skewed toward a few top-tier titles and companies with big time marketing and distribution muscle. APOCALYPSE director John Spinale summed it up, "There are actually a lot of angles that you need to be aware of when you jump into the realm of console development. One that may or may not be interesting to other developers, but is definitely a reality of life, is the economics of console development. You have a much larger startup cost in terms of the development hardware you have to buy, and the developer pool is much more limited.

"In addition to that, once you get down to creating a title, the economics of actually getting that title out to market are substantially different from a PC game, where you have shareware, low-cost development, or the ability to find a publisher later in the development stage."

The Birth of APOCALYPSE.

The story of APOCALYPSE is deeply rooted in console gaming. In the early years, console games were quite different from PC games because of their emphasis on pure, arcade-oriented action.

"The main difference between a console game and a PC game is the interface between the player and the game. With a PC, you have a mouse, a high-resolution monitor, and a keyboard. It's a different experience than when you're sitting on your living room floor in front of a wide-screen TV with your PlayStation jacked into the Dolby stereo system," said APOCALYPSE producer Michael Kirby.

Activision wanted APOCALYPSE to be something more akin to the pure shooters of yesteryear. The designers wanted players to have a quick experience and quick gratification. The team felt that action and game play elements were lacking in many of today's games, and they wanted to re-address the genre and create an action game similar to popular titles in the previous console generation.

A specific problem pointed out by many — especially around Christmas '96 — was the plethora of also-ran titles. The APOCALYPSE team felt that there was a lack of good games on all of the 32-bit systems. They longed for CONTRA on the Super NES or GUNSTAR HEROES on the Genesis — games with "pure, thumb-pumping, action-oriented stuff."

The shifting demographics of the console market have long played a big part in game design. The early adopters that once drove the 32-bit console market have given way to younger players. Kirby believes that "thumb-pumping action" is a key to success on the PlayStation.

Kirby believes there are three fundamental rules in the action genre (whether it's a game or a film). They are:

1. Your personal world is threatened.
2. The enemies are huge and getting bigger.
3. If you do nothing you will perish.

He says that these guidelines are effective in games no matter what age the player is. "Everything you need to know about our universe, you learned in kindergarten," Kirby quipped.

APOCALYPSE represents Activison's continued emphasis on real-time 3D environments. But the team also wanted to develop a character-based world.

"What we were able to do on the PlayStation was build real-time 3D environments rather than simple 2D backgrounds," Kirby explained. "That opens up a whole new realm of possibilities for depicting main characters and integrating cinematic techniques within the world and creating a sense of drama and character that you couldn't achieve in the 16-bit world."

APOCALYPSE NOW

Here's the premise of the game: APOCALYPSE is set in a future where an evil madman known as The Reverend has managed to rerelease the Four Horseman of the Apocalypse. Bruce Willis plays a scientist who has uncovered the plot and he enlists you, the player, to help him.

The premise sets up Willis's character as a your partner. You interact with Willis's character, which has its own AI and motives.

Because the scientist is so central to the player's experience, a lot had to be done to create a memorable and realistic character. To embue the scientist's character with more realism, Activision authored a system they call ActiVation. ActiVation consists not only of a labor-intensive cut scene process, but the AI and real-time character programming as well.

The cut scenes of the game were entirely outsourced. Bruce Willis was cyberscanned by Viewpoint Datalabs. Body and facial movement were animated using motion capture software. hOuse of mOves provided the motion capture facilities and produced the files. Later, the resulting animations, physical and facial, were attached to the mesh model, and the cut scenes were animated. Although they were assembled by outside art house Equinoxe, all of the scenes were supervised by the APOCALYPSE art staff, which provided complete oversight, storyboards, and instructions for both the capture sequences and resulting animations.

Willis, who received some Activision stock as part of his compensation for the project, didn't just lend his name. Willis contributed as much as he could to the team, based on his skills as an actor. "We put some spinning and crashing shots into the motion-capture production using a harness that Bruce specifically requested," Kirby said. "Once I laid out what was possible, he was very quick to construct his own representation in a drama, and he was also cognizant of his own limitations. I wouldn't say Bruce has logged a million hours of game play, but he's a remarkably fast learner."

Using motion capture files and cyberscanning was fine for the cut scenes, but the APOCALYPSE team felt that their skills were better suited for constructing the game engine. Cyberscanned models and their accompanying motion capture files are too complicated to use in real time. What Danny Matson and his APOCALYPSE art team found was that after you removed all the extra frames from a motion-capture file, you're almost right back to where you started if you had done the animation from scratch. Instead, APOCALYPSE's real-time graphics are all hand-created. Another problem for APOCALYPSE was that the PlayStation only has 2MB of RAM for artwork and other game elements at any one time. Thus, meshes and other art elements have to be highly optimized.

Many of the game's environments are dominated by an individual Horseman, who the player faces at the end of each of four key points in the game. Players progress through the game, interacting with Willis's scientist character, destroying bad guys, picking up powerful weapons, all while trying not to stand in one place for too long. If this sounds like the basic plot of most shooters before APOCALYPSE, it's entirely intentional. However, the perspective of the player is quite different from other games in this genre.

The player runs through entire 3D worlds using a third-person perspective, à la TOMB RAIDER. Spinale emphasized the use of different camera angles

to provide almost three different shooting games for the price of one.

"We use a dynamic camera to set up the scenes and the action," Spinale explained. "It boils down to three basic modes of game play. One is a tracking shot, similar to TOMB RAIDER. The second is a side-on shot, where you're looking at a profile of the character. This view is similar to the mechanics of the side-scrollers of yesteryear, where you move in and out of the screen, as well as left to right. Lastly, there is a very high top camera, giving a 360-degree field of view around the character. We're dynamically going to each view on the fly, depending on the area the player has entered."

One problem that this on-the-fly camera creates is a constantly changing angle for the player, which might make it hard to differentiate left from right. The design for APOCALYPSE is prepared for that. Noted Spinale, "We've set up the control structure to make the player's aiming relative to the camera positioning. As we switch the camera angle and the player moves, right is always right and left is always left. You don't need to switch your aiming to compensate for any change on screen."

Artwork and the Armageddon

Perhaps the biggest difference between console- and PC-based development is content construction. The console world revolves heavily around the display offered by a living room television set. That creates a host of issues that the APOCALYPSE team had to work around. Matson explains, "Developing art for the PlayStation is a little more involved because you go through additional steps to eventually render the art as it will appear on a user's television set. The TV actually aliases art for you, but at the same time contrast and color differences are very different from what you might be used to on a PC monitor. So you have to adjust for that by trial and error. On the PlayStation, to test the display you have to go through about six more loops than you would on a PC product."

While some tool manufacturers offer products that are supposed to help preview art that will be moved from a PC to a console system, Spinale pointed out that these tools are still not perfect. "You can never really tell [what it will look like]. Everyone is trying to take a stab at the visualization issue, which is good, but at the end of the day you never know what you're going to get on an NTSC monitor until you see it there live."

Trying to get the right color mix on a TV screen was therefore a burning issue for the APOCALYPSE artists. In the process, they found that saturated colors tend to show up better than subtle shades of colors on a TV screen.

The other problem faced by the team was that TV display variance is wider than the variance in the monitor market. Manufacturers are constantly fooling with display characteristics and TV components of different sorts and qualities. Summing it up, Matson simply stated, "This is still a serious problem."

Programming the APOCALYPSE

The PlayStation development kit provided by Sony consists of a PCI-based PlayStation card and development software. Developers need to be able to burn PlayStation CD-ROMs, which requires specialized software because PlayStation CDs are unlike other formats. Once you have the SDK, you'll need to work through all the basic development libraries that come with the console. In many cases, these libraries need extra work. Activision's APOCALYPSE programmers used all the major development libraries supplied by Sony as starting points and

Development Profile

Development Team. The team consisted of about 16-20 people: a producer, a director, an art director working with a team of 5-6 artists, a lead designer working with an additional 3-5 assistants, and a lead programmer with 3 assistant programmers. Artists were organized in concentrated areas such as conceptual, modeling, and texture maps, while the programming team consisted of the lead engine designer, with others chipping in tools and technology aspects required by the engine specification.

Language Used. C and assembly.

Content Tools Used. Adobe Photoshop, Autodesk 3D Studio MAX, Alias Animator for cut scenes, Lightwave for special effects, Debabelizer (Mac), and internal tools for optimizing 3D meshes.

Special Libraries. The standard PlayStation libraries were rewritten for speed and to reduce memory requirements.

Budget. The APOCALYPSE team refused to release the budget, but other reports have pegged it at over $1 million. "In terms of budget, a similar PC product probably would cost 25-30% less," said Spinale.

Time of Development. The development schedule had three main stages: six months of true conceptual and preproduction, six months of creating the engine and supporting assets, and six months to add the game and story elements and build it into a finished game. "In terms of development time vs. PC games, it's about the same, maybe a little bit longer — basically 18 months. If you've got an engine and toolset in place, you can certainly cut down some of the time," said Spinale.

enhanced them to create their own in-house development libraries

With this foundation in place, the APOCALYPSE team began constructing the game engine. The lead programmer coded the real-time aspects of the ActiVation technology, an engine data structure, a rendering engine, and animation libraries. The APOCALYPSE engine will see further life in other Activision titles down the line.

Spinale recommends smaller teams for console development. He feels that a small team is well suited to coding the run-time engine because of the customized and relatively expensive development hardware. He says that the lack of polished libraries and support utilities that are common to PC game development also hinder the process. Fewer people manipulating the console code therefore makes it easier to create a fast, efficient engine. "You need people who own the code from the start," Spinale said.

Heed the Warning

Marketing a console title, begins as early as possible. Some game magazines such as *Next Generation* have already shown previews of APOCALYPSE, even though it's not slated for release until later this fall. The product is also making the rounds internally to all of Activision's retail partners, as the company drums up orders and shelf-space commitments for the title — especially from large accounts such as Toys 'R Us. Activision made APOCALYPSE their featured game at this year's E3. While the game will ultimately have to be judged on its own by consumers, a key marketing campaign is crucial to the success of the title.

Consumer gaming magazines will provide ample coverage, but Activision is also banking on coverage and exposure for APOCALYPSE by the mainstream media such as CNN, *Entertainment Weekly*, and *Time Magazine*. In

APOCALYPSE's case, the partnership with Bruce Willis will help break through the clutter of titles in the marketing and PR mix.

In the end, though, there are two major resources you need to pull off an attempt at console stardom. If APOCALYPSE succeeds, it will not only be because of Activision's monetary and

branding commitment, but because of the APOCALYPSE team's commitment to push out a title worthy of the financial commitment it takes to play at this level.

Ending the interview with his own biblical warning to other developers, Spinale advises, "In general, we at Activision have really taken on a lot of challenges with APOCALYPSE, especially to have such a fast engine and new character AI ideas. We definitely bit off a very large chunk for our first in-house PlayStation project. I would definitely put out the warning now: If you don't have the best resources, money, people, and marketing, be very careful about jumping in on console development." ■

Based in Portland, Maine, Ben Sawyer writes and consults about the interactive and consumer technology industries. He recently finshed a report on Microsoft's Internet strategy for Jupiter Communications. He can be reached at BenSawyer@worldnet.att.net.

Market Profile

Debut. APOCALYPSE is scheduled for release in late September, just in time for the Christmas '97 shopping season.

Distribution. Retail CD-ROM SKU being distributed by Activision.

Marketing Campaign. A major advertising campaign is slated. The high-profile launch will surely include a round of mainstream press interviews with Bruce Willis. Early preview articles are already appearing in such magazines as *Next Generation*.

Competition. With a Christmas release, the product is up against everything. In terms of direct competition, it will hit up against any company putting forth next-generation, TOMB RAIDER-style action products. Shiny's MDK, were it on the PlayStation instead of Windows 95, seems to be a product with similar designs. For the moment, the positioning seems sound.

Outlook. Activision has a resurgent line of titles garnering it closer to the top-level position it once held, and the product is hitting the PlayStation at its peak year. The Bruce Willis star power could help it with mainstream PR. All this bodes well, but as with last year, will there be so many titles released in the fourth quarter that even fine games like this could get bowled over? Early previews have been well received, and a strong showing at E3 and top-notch marketing campaign could get it the critical mass needed to rise above the fray.

GameFx Accelerates Past the Competition

Crazy! That seems to be the concensus midway through a GameFx presentation. It's crazy for them to be developing a game wholly reliant on cutting-edge hardware — MMX processors and 3D accelerators that leave the mass market behind. Crazy to separate story from the game itself,

to not involve the player in some explicit heroic journey, to abandon the glitzy graphics and soap-operatics of cut scenes. Crazy for a company barely a year old — without a single commercial title under its belt — to flout industry wisdom by attempting these things with its debut product. Crazy, that is, until the presentation reaches a hands-on demonstration of the game, still pre-alpha, and the stunning quality of its graphics.

"Once they see the demo, they do a complete about-face," laughs artists/designer/producer Walter Wright. "Forget that they just said there's no market. Suddenly they want to be involved!"

Wright's enthusiasm is genuine, as is his conviction that the path GameFx has chosen is a valid one. The small group of talented iconoclasts in their Arlington, Mass., office are together in large part because they share this vision: that tomorrow's games don't have to be saddled by the limitations or conventions of the past, that the place to be is ahead of the technology curve. They're determined to make eye-popping, jaw-dropping, cult-forming games, and they're willing to leave a big chunk of the market behind to do it their way. The philosophy itself is infectious, but it's accompanied by an acknowledgment that success won't be won with eye candy alone.

"If we can make the game play as well as we know we can make the game look, then we have a winner. I think." The hesitation isn't uncertainty regarding the GameFx team's ability to create a truly fun game, but awareness of powerful forces outside any game developer's control: the vagaries of the marketplace and, in this new era of Manifest Destiny — Go online, young developer! — the power of buzz.

Through with Looking Glass

The origins of GameFx might be traced to a meeting-of-the-minds of sorts that occurred at the 1996 Computer Game Developers Conference. There, a group from Looking Glass Technologies, rallied by Noah Davis — then director of the Advanced Technology Group at LGT — began riffing on the state of the industry and the future of computer games. The group agreed that this was an exciting industry but, like many businesses, it was slow to accept change. It was suggested that the computer game may grow into the popular artform of the next century. It was debated what role story has in a game. And it was realized that, in many respects, right or wrong, this group saw things differently than most of the short-timers and hardcore veterans thronging the conference.

The folks at GameFx are making a render-on-the-fly game so great-looking you'll think it's a prerendered rail-shooter till you grab the joystick and see for yourself. So why do many game companies think they're crazy?

One major difference was the value they placed on cutting-edge technology. Eight-bit graphics and 486 processors represent, to most game developers, a teeming market. To the GameFxers-to-be, this legacy technology represented limitation and frustrated potential, a barrier — self-imposed by the game developer — that retards the evolution of games and the industry.

Another difference was the growing conviction shared by the group that in many games the current role of story had become an anachronism. While text-based adventures and low-resolution, eight-bit graphics clearly invited the adoption of traditional story elements to engage the player, and while LGT itself had taken the role of the interactive story to a new plateau with their '94 release of System Shock, the sort of visceral games this group longed to make were, they felt, less reliant on story. In such cases, a traditional handling of story would only fight for attention with the graphically-rich game play that should be driving the game.

To this group of developers, subculture was more important than story. They felt there was a sort of gestalt — a buzz — that had begun to develop around certain very successful multiplayer games, elevating them from the solitary pastime of lonely nerds to the centerpiece of vibrant, online commu-

nities. Backstory may play a role in the formation of game subculture — providing a superficial setting for the game's conflict and establishing a sort of core vocabulary for initiates — but backstory shouldn't get in the way of the pure fun of game play and shouldn't try too hard to shape the group dynamic. That dynamic was seen as the thing to be cultivated, and multiplayer possibilities beyond "deathmatch" beckoned as a virtually untapped resource.

Episode One

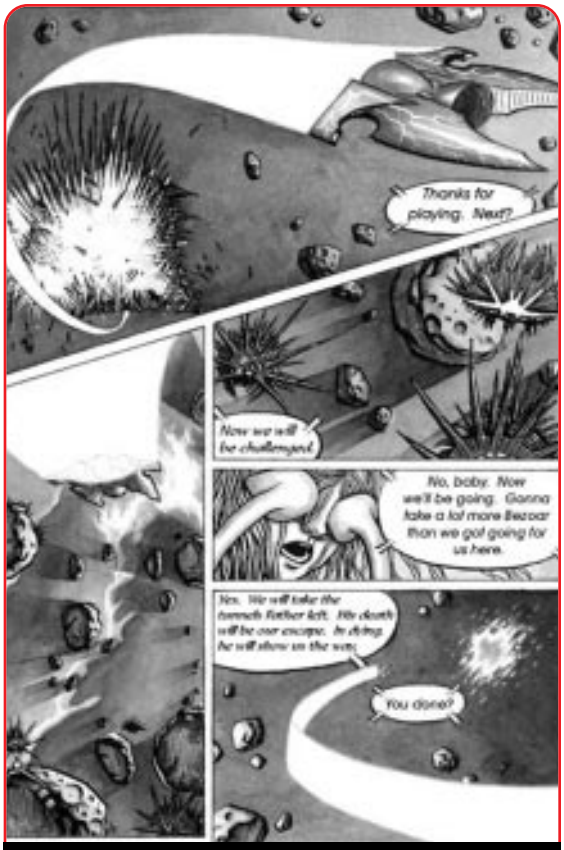
Unsurprisingly, a new company grew from this seminal meeting. Along with several other like-minded LGT alums, most also from the Advanced Technology Group, Noah Davis formed Advanced Gaming Technologies as a subsidiary of 3Dfx Interactive, a manufacturer of graphics accelerators. Their pilot task was to create a demo showing off the sort of great real-time 3D they — and 3Dfx — knew would be available to game developers once accelerated graphics had found a home in the mass market. Their plan from the outset was to grow this simple but flashy spaceflight demo into a full-blown game that could be embraced by more than just 3Dfx customers.

3Dfx itself is part of high-tech entrepreneurial visionary Gordon Campbell's Techfarm family of companies. Campbell recognized that the more interest that could be generated around hardware acceleration and accelerated graphics, the better for everybody. Last December, he cut the strings binding Advanced Gaming Technologies to 3Dfx and established Davis's small band of LGT expatriates as GameFx Inc. Their mission: to ride the technology wave and push the gaming envelope for all it's worth. Their first envelope-pushing title: Out of the Void, Episode One.

Rule #1: No Rules

As might be expected, Out of the Void breaks a number of cardinal rules. It's not simply optimized for graphics accelerators and high-speed MMX processors; it's reliant on them. Running in 16-bit color at 640×480 with real-time lighting, it's successfully pushing 6,000-7,000 textured polygons at 30fps or better. An even higher polygon count is anticipated once culling operations are in place. But that's only if you're running their target system: a well-tuned 200MHz MMX Pentium equipped with hardware-accelerated 3D.

Though cynics say there's no market for hardware-optimized games, GameFx says "just you wait." They're developing for the systems expected on the market by the fall release of Episode One. Besides, GameFx points out, there's little hope for a tiny start-up to battle megalithic publishers for shelf space. The company's leading-edge positioning should lead at least to hardware-bundled distribution, or perhaps a distribution deal with some forward-thinking giant. There's already interest in that area, and by the time this column sees light of day, such a deal may already be inked. As artist/producer Walter Wright observed, everyone who sees the demo wants to figure out how they can get involved.



Backstory for Out of the Void: Episode One is provided in a comic book, to allow for more linear game play on the screen

Another rule being broken relates, of course, to story. As the name suggests, Episode One does have a story, but GameFx refuses to allow it to intrude on the game. Their belief is that they should take full advantage of the power of the leading-edge system for which they're developing by creating visceral entertainment for the player that doesn't break away from intoxicating, highly-detailed, twitch action just to show a B-movie-quality cut scene.

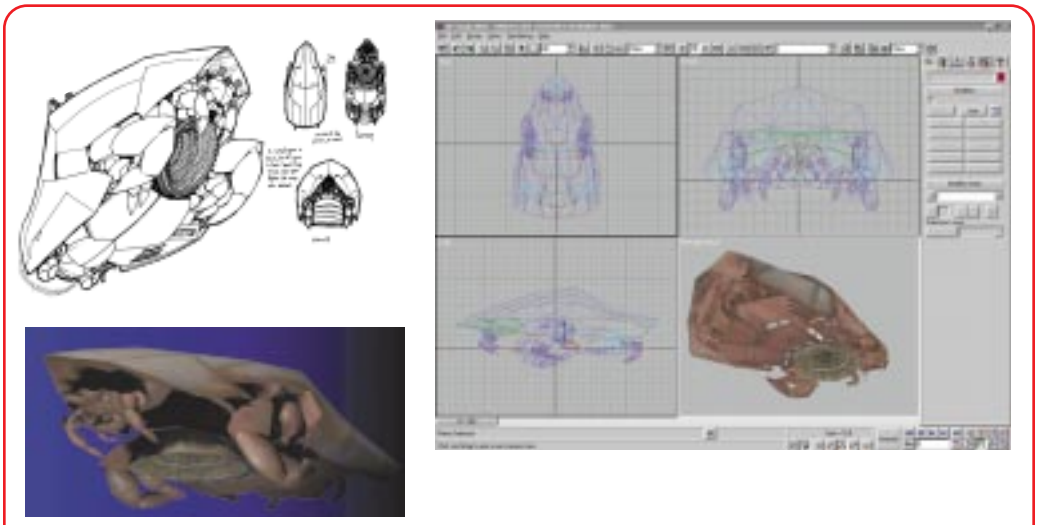
The backstory for Episode One formed in the mind of GameFx president Noah Davis, commingled with his ideas for a new direction for computer games and a new company to forge that direction. Rather than presenting the story within the game itself, Davis collaborated in the creation of a comic book with part-time game artist and aspiring comic artist Gareth Hinds. Not only does Davis's story set the scene for Episode One and a string of sequels, Hinds' art establishes the visual style for the whole Out of the Void pre-alpha universe-in-the-making. (The comic can be seen online at the GameFx web site: www.gamefx.com.)

Backstory aside, the goal is to put the player directly into the action. The comic-book backstory may help set the scene, but it isn't necessary to understand the straightforward objectives in the game; avoid the asteroids, shoot the aliens, unlock the jumpgate to the next level where you do it all again, and above all have fun.

Another part of letting the player dive into the game and focus on the

Next Gen Preview

For an preview Out of the Void: Episode One, including a game play analysis and lots of cool screenshots, check out the July 1997 issue of *Next Generation* magazine.



Due to an innovative game editor that works as a 3D Studio MAX plug-in, the process of taking a game element from concept sketch to model to fully animated space villain is straightforward and can be performed by the game artists themselves.

action is to not clutter up the screen with complex controls. Status of the player's ship systems is self-evident from the appearance of the ship itself. There isn't even a targeting reticule; just aim for the center of the screen.

One reason for keeping the game and its interface so focused is to allow an easy port to coin-op. In a bustling entertainment center, there's no time to tell a story or read a user's manual to decipher controls. But even outside the arcade, GameFx believes that if the graphics are good enough and the game play fun enough, elaborate plots and recondite controls are superfluous, if not outright distracting. The GameFx philosophy might be summed up with a line from the Immortal Bard: "The play's the thing!"

The Art of the Void

For a company that wants to stay ahead of the technology curve, long development cycles are to be avoided. A game that's been a long time in development has already slipped from the cutting edge by the time it comes to market. Central to the GameFx plan, therefore, is an episodic structure that will make for shorter development cycles. The core game systems developed for EPISODE ONE can be built upon and refined for subsequent releases. Likewise, art resources can be added to, rather than rebuilt from scratch, for each new game. Thus, new episodes can be brought to market more quickly than could totally new games.

Thanks to the efforts of GameFx programmer Brian Jacobson, development of EPISODE ONE has been facilitated by the creation of a game editor that works as a plug-in for 3D Studio MAX. This allows models complete with mapping coordinates to be exported directly from MAX into the game. In addition to populating the game environment with objects such as alien spaceships, asteroids, and jumpgates, the editor can be used to export paths and positions for all these things. Game elements can therefore be manipulated with a rich toolset in an intuitive, visual environment.

Here, GameFx breaks another rule in an industry where artists are often narrowly defined as texture-mappers, modelers, or animators and are rarely allowed to contribute significantly to game design. With such a visually-oriented game, the input of artists is considered important to the overall success of the game's different levels. The MAX plug-in editor facilitates the involvement of artists in that design process. An avid gamer as well as lead artist for EPISODE ONE, Kurt Bickenbach is able to take models of his own creation and deploy them in levels of his own design, levels assembled within MAX and exported to the game. It's a rare treat for a gamer/artist to enjoy such wide-ranging control. In fact, Bickenbach, Jacobson, Wright, and all other members of the team have input into design, which helps give everyone a better appreciation for the working of the whole.

One fact this practice has impressed upon artists at GameFx is the worth of level-of-detail models (LODs). Though EPISODE ONE's space setting simplified game mechanics for the programmers, it presented a challenge for the artists-turned-level-designers, who found that, without intervening walls to restrict view volume, every object on the level could too easily wind up onscreen at once. The high polygon budget made possible by hardware-accelerated 3D allowed them to build models of rare complexity for a real-time game — some level bosses consist of over 4,000 polygons. But if too many highly complex models appear onscreen at once, performance will still lag.

Objects, therefore, were represented by models of varying complexity; as many as seven different so-called levels-of-detail. The most complex version is used at close range to the viewer, and increasingly simplified models are then substituted at appropriate distances where the reduced detail is less noticeable. This cuts down the number of polygons that must be computed for the scene at run time and helps keep the frame rate within an acceptable range. The trick to LODs, GameFx artists found, is to reduce polygon-count without significantly changing a model's silhouette. Reduced detail in a distant object can often go unnoticed, but changes to the object's outline cause an unwelcome visual "pop" when one LOD is substituted for the next.

Artists also were able to experiment with using the editor to attach real-time lights to the ship's pulse weapons, so each shot would light up surrounding objects as it passed. It looked great, but the cost in system resources became high once aliens started firing back and the scene filled with mobile lights, whose effects all had to be calculated on the fly. It is wistfully acknowl-

edged that lighted shots may not make the final cut.

In addition to encouraging its artists to delve into other areas of production, GameFx also expects each to tackle the full range of art-related tasks called for in the creation of a game. Though OUT OF THE VOID comic artist Gareth Hinds provides the concept sketches that define the game's look, the other artists take the ball from there. Each must create texture maps, models, LODs, animation, and whatever else is necessary. Pigeon-holing has no place here. Hinds organizes an informal weekly life-drawing class in the GameFx office, a tradition that the other artists enjoy and credit with loosening-up their approach to 3D modeling. There's nothing like a couple hours of rapid fire gesture drawings to remind you that art doesn't begin or end on a computer screen.

GameFx

GameFx Inc.
645a Massachusetts Ave.
Arlington, MA 02174
617-643-3293
Fax: 617-643-3286
Web: www.gamefx.com

What's After EPISODE ONE?

Hinds has already created concept art for several alien races, though only one race is slated to appear in EPISODE ONE. In addition to trotting out other exotic extraterrestrial species and their idiosyncratic starship designs for future episodic extensions of the space shooter, GameFx plans to evolve the game and the game engine to take players out of the void of space and bring them down to Earth — or Mars.

Continuing their knack for parlaying contract jobs into the building blocks for great games, GameFx is taking a Mars terrain database they've assembled as a side project for Intel and using it as the foundation for a terrain-based OUT OF THE VOID episode. The goal is to use the same editor and game engine with enhanced object animation capabilities to allow models that are more articulated than the spacecraft in EPISODE ONE. After that, they expect OUT OF THE VOID will go underground for a high-detail corridor shooter with an even further-enhanced game engine.

But all of these future plans hinge on EPISODE ONE acquiring enough of a following to warrant subsequent episodes. That's where "the cult" comes in, the subculture that GameFx sees as more important than story.

"They'll all know the story," Wright points out. "They'll have seen the comic book and all, but they'll be in some multiplayer arena bashing one another in a way that has nothing to do with story. What's important is the social interaction that builds up around the game."

Central to keeping this subculture buzzing is follow-on; supporting the released game with value-added features. GameFx plans a level editor, certainly, so players can design their own levels. But the goal is also to make the game editor easy enough to use to let players insert new models, even their own models. Of course, it would require a savvy user with significant computer graphics skills to customize the game to that extent. But players are already doing amazing things with level editors for existing games. The idea of players importing their own models into the game is just another example of the envelope pushing GameFx believes so fervently in. They look forward to their web site being a dissemination source for new, homegrown OUT OF THE VOID levels and models.

Another planned follow-on feature is multiplayer game options beyond so-called "deathmatch." The true six-degrees-of-freedom maneuvering of space in EPISODE ONE will, they think, lend itself well to far-out versions of football and capture-the-flag. They'll be on the lookout for other possibilities during play testing, but the real hope is that "the cult" will take on a life of its own and concoct play styles GameFx hadn't predicted. Money can't buy buzz like that. ■

In addition to writing the Artist's View as a Contributing Editor for Game Developer, *Dave Sieks is a Creative Director of 1711 Software, a developer of online entertainment. You can e-mail him at gdmag@mfi.com.*

by Nolan Bushnell

A Product Whose Time Has Come...Again

The video coin amusement business in the late '70s to early '80s exceeded $18 billion annually before declining. It has been stagnant for over a decade, and today brings in only $6 billion a year.

But that is rapidly changing. In fact, I believe location-based entertainment has the potential to be a thriving $20 billion market — in a remarkably short time.

The Internet is driving this change. By giving the public greater variety of entertainment, by pricing it fairly, and by promoting it like crazy, we can tap the one thing that people who frequent hospitality venues all have in common — the desire to have fun.

"Game-Driven" Marketing Is Dying

From the time Pong came out in 1972, the video game market has been title-driven. "Make a successful game and they will come." That strategy has been applied to console video games, to standalone PCs, and now to online gaming in the home.

In large part, this strategy has worked. But it ran its course in the arcades, and is approaching its limits in the home. In location-based entertainment, what matters most is tailoring a customized and rapidly-changing entertainment package to the tastes of particular locations. With online technology, we can do that.

Rather than developing games only for young males, we must reach out to many different audiences. To the travelers and business people that frequent hotel lobbies. To the upscale "latté crowd" that frequents Starbucks Coffee Bars. To the eating and drinking crowds in restaurant chains and bars.

Games are played by less than 10% of typical bar patrons. The recipe for success now must also include games, communication via e-mail and chat, shopping, tournaments, and music. Each location must find the right mix for its customers.

We Need a New Set of Rules

We've used any number of explanations for why the floor dropped out of coin-op, including: coin-op failed to be significantly better than home games in terms of technology — polygons per second, real-time ray tracing, MIPS, and so on; if we only had a $1 coin...; if only the manufacturers would produce fewer games, build cheaper games, not release games into the home so fast, and build games that would last for 100 years; home games killed the coin-op business; all the industry needed was a great hit; and the number of locations has fallen.

But none of these arguments hold water. If coin-op games need to be superior, why has MEGATOUCH (with a Z80) out earned CRUISING and MORTAL KOMBAT in bars over the past few years? If the type of coin is so essential, why have tokens of high value failed every time they've been tried? The only bright spot on the U.S. landscape is the nickel arcade. The home computing market comprises 15 million households out of over 100 million in this country. There's a brave new world out there in public places, and we are just beginning to find it.

The New Rules

It's going to take a shift in our thinking to make this happen. Here's my recipe:

- *The value of a game has to be higher or its cost reduced.* Compare the cost of a first run movie — $7.00, or roughly $3.50 an hour — to the $24 it costs for an hour of entertainment; arcade players shell out a buck for every two-and-a-half minute game.
- *ROI must come from multiple sources, not blockbuster hits.* Despite the doomsayers who predicted that TV, VCRs, and finally cable would kill off theaters, the movie industry has never been healthier, hitting record sales levels last year. But notice which movies dominated the Oscars this year — independent films. Not the big budget studio movies. The best movie of 1996 had less than a six-month run, and the average movie ran for less than two. We've got to go that same route by reducing development costs and adjusting to the reality of shorter life cycles and lower returns on individual games.

CONTINUED ON PAGE 63.

"Soapbox," continued from page 64.

Let's take another page from movie marketing — seasonal and holiday fare. For example, is a Halloween game that is downloaded at the beginning of October and taken off the first week of November such a bad idea?

• *Variety is the spice of location-based life.* Why would someone pay $60 for a software title when a similar game could be played in a public place for a quarter? And when those quarters can be spent on 10 new games instead of just one? Variety has always been more important in leisure time than price. Why is that any different now? Perhaps we haven't seen enough variety in arcades up to now.

• *We've got to bring entertainment to the places where most people socialize.* I travel a lot. I see games in sports bars, but not in hotel lobbies. I see fast food locations without games. I see coffee bars with no games, and many chain restaurants with no games or amusements. In most of these locations, a pinball machine would look strange, but a countertop web terminal with easy touch screen menus and game play would not.

For those of you who remember, Pong was in a wood-grain cabinet that went into places that didn't want garish graphics and decals of monsters in their locations. I find little that can be purchased today that will fit the decor of a Starbucks or a United Airlines terminal. Has our industry kept current in meeting the demands of an older, more upscale crowd?

Networked Entertainment is the Bridge to Newfound Revenues

Now, with online networks, entertainment providers can change games at will, experiment to find out what works best for a particular location and customer base, and put in those games that provide the greatest income. Instead of a hit game that brings declining revenues over time — the old arcade model — we will find that the hardware makes *more* money as the software is finely tuned to each location.

With networked entertainment, revenue streams can multiply in locations through national and even international tournament events with big-time promotional sponsors and through advertising and the sale of merchandise via location-based terminals. We don't have to market to get the people to come to us, we are bringing the entertainment to the places they already frequent — and spend a lot of money — to have fun.

The customers are there. There's no shortage of talent to develop entertainment for them. But we've got to challenge ourselves to invent new rules and jump-start this next wave of location-based entertainment. ■

Nolan Bushnell is the Director of Strategic Planning for PlayNet Technologies, which has launched a networked entertainment system aimed primarily at the hospitality marketplace. Bushnell's creation of Pong and founding of Atari in the 1970s is credited with launching the video game revolution.