gd

GAME DEVELOPER MAGAZINE

APRIL/MAY 1997

UBM Tech

Chapter IV: A New Hope

When engaged in a long-term project, it's helpful to stop what you're doing once in a while and assess your progress. Are you on schedule? Will you reach your goals? Perhaps more importantly, are your goals still valid, or has the marketplace changed enough to warrant a change in your plans?

Here at *Game Developer*, we decided that it was time to do just such an assessment of the magazine. It's been three years since a group of editors (notably Alexander Antoniades — thanks Sander!) dreamt up a magazine aimed at game programmers. With a little convincing, we got the green light to launch *Game Developer*. Thanks to a fantastic staff (which, incidentally, volunteered all of it's time during the first year this magazine was published) and you, our readers, the magazine has steadily grown. Now that we've craned our necks around to see where we've been and where the industry is going, it's time to tweak things a bit.

Most commercially successful games these days are created by a small to medium team of people comprising programmers, designers, artists, animators, sound designers and composers, and so on. While the programmer is a pivotal role responsible for weaving all manner of digital assets into the final product, none of the roles mentioned above operates in a vacuum. Programmers work with sound effects people, game designers interact with animators, and so on in a complex web of communication. And due to our respective fields of expertise, communicating across these job function boundaries can be challenging. That's why we're widening *Game Developer*'s scope of coverage — to try to help bridge the gaps between team members by featuring articles written from various functional viewpoints. Our subtitle has been changed from *Programming for Fun* to *On the Front Line of Game Innovation* to underscore our belief that games require more than just solid code — they require outstanding execution at a number of different technical and creative levels, including (but not limited to) programming.

With this issue we also kick off our new design, which is more dynamic and visual than our old look. It's also more playful, which is certainly in keeping with the spirit of our creative community. I hope you'll appreciate the new aspects of the design, and we trust you'll give us your feedback on what's working and what isn't.

This issue also marks the beginning of a new program designed to promote the talents of game artists and animators. Each issue will feature custom artwork by artists in the industry. You can check out a short bio about the artist and the tools they used to create the cover image on the contents page. If you're a game artist or animator and you're interested in the opportunity to get your work on our cover, check out our web site for further details.

On the back page of the magazine you'll find a new column, "Soapbox." This column is exactly what it's title proclaims: a forum for industry visionaries to comment on topics that they are concerned about. In the inaugural column, Sid Meier explains why more and more successful games come from small producers, rather than the bloated firms that seemed to be taking over the industry as recently as a couple of years ago.

Finally, beginning with the June issue, the magazine will go monthly. It's extremely gratifying to me to reach out twice as often to readers and touch upon so many more subjects in print. Our first stab at distributing additional magazine content via our web site has shown promise, as witnessed by our *Online Game Development* special report that's currently available for purchase and download there. We'll likely be taking one or two more stabs at electronic distribution projects this year, which opens even more doors for us to circulate game development information.

As we take the wraps off of our new design and polish it over the coming months, our commitment to providing the most authoritative and valuable information in a clear, attractive format continues to guide us. Please let us know how we're doing.

ON THE FRONT LINE OF GAME INNOVATION
Game DEVELOPER

EDITOR IN CHIEF Alex Dunne
adunne@compuserve.com
MANAGING EDITOR Tor D. Berg
tdberg@sirius.com
EDITORIAL ASSISTANT Chris Minnick
cminnick@mfi.com
CONTRIBUTING EDITORS Larry O'Brien
lobrien@msn.com
Chris Hecker
checker@bix.com
David Seiks
dsieks@msn.com
Ben Sawyer
bsawyer@worldnet.att.net
ADVISORY BOARD Hal Barwood
Noah Falstein
Susan Lee-Merrow
Mark Miller
Josh White

COVER IMAGE Vector Graphics

ADVERTISING SALES STAFF

ASSOCIATE PUBLISHER Cindy Blair
(415) 905-2210
cblair@mfi.com
WESTERN REGIONAL SALES MANAGER Tony Andrade
(415) 905-2156
tandrade@mfi.com
SALES ASSISTANT Chris Cooper
(415) 905-2379
ccooper@mfi.com

PUBLISHER/GROUP DIRECTOR KoAnn Vikoren
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 MANGER Mike Poplardo
ASSIST. CIRCULATION MANAGER Jamai Deuberry
SUBS. MARKETING MANAGER Melina Kaplanis
NEWSSTAND MANAGER Eric Alekman
REPRINTS Stella Valdez

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/ SOFTWARE DEVELOPMENT DIVISION Regina Starr Ridley

www.gdmag.com

"Says You" Question about "Sez U!"

In the "Sez U!" section of *Game Developer*, June/July 1996, Chris Hecker mentions that one can generate the gradients for perspective texture mapping using the plane equations. In the equations you give, I assume the second *u* should really be *v*.

What are the other coefficients? Are the *a,b,c,* and *d* you give the plane equation coefficients? If so, what are *e,f,g,h,* and *i*?

Anthony Tessier
VIA E-MAIL

CHRIS HECKER REPLIES: *Yes, you're right about the second* u *being an error. As for the question, the* a,b,c *and* d *are not the plane equation coefficients as you normally think of them. An* ax+bx+cz+d=o *plane gives no information about the orientation in the plane, so you can't really use that for placing a texture.*

Instead, when I say "plane equations" in the context of texture mapping, I mean you define the texture plane in view space (3D space with the eye at the origin looking down the z axis) using three vectors, O, U, and V, so that a point on the texture plane in this space is parameterized by the texture coordinates, u,v *as follows:*

$$P = O + u * U + v * V \quad (1)$$

In this equation, the vector O points from the eye point to the origin of the texture (meaning the place where u,v = o,o*), and the U and V vectors form the plane of the texture. Now, Eq. 1 is a vector equation, or actually three scalar equations. If you project the 3D points on the texture using*

$$P'_x = \frac{P_x}{P_z}$$

and

$$P'_v = \frac{P_v}{P_z}$$

*(the primed points are the 2D screen-space projected points), then you have two rational equations in two unknowns (*u *and* v *are the unknown texture coordinates we're trying to find, and* P'_x *and* P'_y *are known since they're your screen space coordinates in your polygon):*

$$P'_x = \frac{P_x}{P_z} = \frac{(O_x + u * U_x + v * V_x)}{(O_z + u * U_z + v * V_z)}$$

and

$$P'_v = \frac{P_v}{P_z} = \frac{(O_v + u * U_v + v * V_v)}{(O_z + u * U_z + v * V_z)}$$

If you solve these equations for u *and* v *you'll get two rational linear equations in* P'_x *and* P'_y *like I had in the original mail.*

Please Disclose the Nondisclosable

I would like to see more console coverage. I'm not sure how this would effect nondisclosure, but it would be nice. A good article would be one on cross-platform development. Being a lead PlayStation programmer on a soon-to-be-released title, I have encountered (actually, the entire team has encountered) pitfalls in multiplatform game development.

Another thing is that it would be nice if the community as a whole stressed quality over quantity. It is almost impossible now to develop a good game in a year, as opposed to what you could quickly turn around for the 16-bit market.

John Bryant
VIA E-MAIL

EDITOR ALEX DUNNE REPLIES: *One of our goals with the relaunch of the magazine is to bring in more coverage of console titles. Stay tuned.*

As for your second comment, don't expect the quantity of games on the market to diminish anytime soon. Game developers now more than ever have reasonably priced development tools, access to pertinent technical information, a growing customer base, and in the case of PC-based games, a cheap and effective distribution medium in the Internet.

Today's large number of titles means that some high-quality games get overlooked — sometimes the almighty marketing dollar wins out over superior play. We just have to hope that word of mouth among game players compensates for this.

Optimizing BSP Trees

I read Michael Kelleghan's article on Advanced BSP Trees ("Advanced BSP Techniques," Oct/Nov 1996), and I wanted to point out a few of the areas in the article that should have had some more explanation. I noticed a lot of references to Z-order of polygons. I have spent a lot of time developing real-time 3D BSP engines in the last couple years, and this struck me as a little counter to the whole purpose of a BSP tree.

In a true 3D environment, the camera can view the scene in any direction from any point in world space. Since the Z-order of polygons depends on the camera's current viewing direction, Z-order changes constantly. This is why a static BSP tree is so useful. It requires no runtime reclassification of polygons with respect to the camera. For complex structures, the BSP tree is built once, then sorted by recursively classifying the eye point with respect to a particular BSP node and then drawing all polygons at that node. This is a highly efficient and perfect sorting algorithm for 3D environments where the camera (but not the scene) is changing all the time. The author, however, seems to present the camera as static, in which case there are probably much faster ways to perfectly sort a 3D scene, such as building a Z-buffer of the background scene, then drawing the dynamic objects in ascending Z-order.

As far as BSP optimization, the author didn't mention a very important technique: minimization of polygon splitting. When a BSP tree is built for a complex model, polygon counts can increase anywhere from two to four times if partitioning planes are selected blindly from the top of the list. Polygon-count increases can be brought down to between 0%(best case) and 25%(worst case) if the entire subset of polygons is searched for the one whose plane produces the least amount of splits. It takes a lot of time to build an optimal tree, so it should be done offline. Trees can be optimized even better if the modeler inserts his own planes to separate curvature in the model (a lot of cross-splitting occurs between nonlinear sections in a model).

Miguel J. Gomez
VIA E-MAIL

AUTHOR MIKE KELLEGHAN RESPONDS: *Upon reflection, I can see that the article would give the impression that the camera is static. Thanks for the clarification. Your comment about the lack of BSP optimizations is correct. I didn't mention a number of optimizations, as I had to keep the article short to meet the limitations of the magazine's page count.*

INDUSTRY WATCH

by Ben Sawyer

With the CGDC and E³ around the corner and sales figures back from the holidays, companies are taking stock and moving to focus their plans and killer products for E³ in hopes of building better momentum for sales success by next Christmas.

DIGITAL PICTURES LAID OFF about a third of its staff, mostly support people; it hadn't been a good holiday season, especially after several titles didn't ship in time. Sanctuary Woods, 3DO, and Sega also laid off people as strategies shifted or sales didn't produce.

Sega dismissed a number of marketing and director-level staff members as the company merged its hardware marketing into a single-division focus. The Bandai merger hasn't yet produced layoffs, but as the merger goes through, there will certainly be some thinning of the ranks.

GTE Interactive, which was originally formed to build content for GTE's MainStreet Interactive TV initiative, announced that it would shut down as the Interactive TV market failed to materialize.

Sometimes personnel make their own strategic adjustments. Ron Millar made headlines when he jumped from his position as senior designer at Blizzard to Activision to work on DARK REIGN, Activision's real-time strategy title. Millar is also forming Redline Games with James Anhalt to produce titles for other publishers. Millar's responsibilities on DARK REIGN will include character development and mission building.

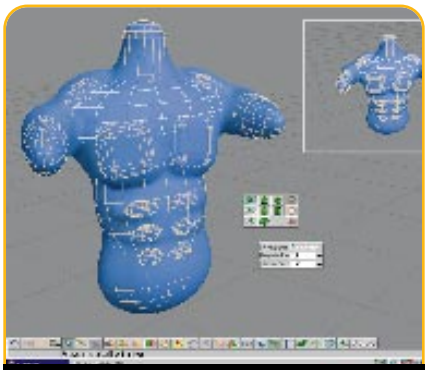
"I chose to work on DARK REIGN," says Millar. "I have my pick of what I want to do, and I think DARK REIGN is by far the coolest thing on the market. This game is going to rock."

Lookin' Gooood

Artistic software dominates the new products scene.

trueSpace3

CALIGARI CORP. HAS released the latest version of its 3D modeling and animation tool, trueSpace3. The package offers a broad-ranging toolset and the ease of working in a single, integrated, modeless environment.



Real-time object deformation with trueSpace3's Live Skin.

To list some modeling and rendering features, trueSpace3 offers a real-time metaball modeling tool called Live Skin, 3D paint and brushes, and plastiform surface engraving and deformation. A host of animation tools includes built-in collision detection, the ability to specify the physical properties of objects such as rubber balls and styrofoam cups, and inverse kinematics for the manipulation of models. There is even a 3D sound tool for attaching directional sound to an object.

trueSpace3 is also VRML 2.0 enabled and actually includes a VRML browser. But it's not just a VRML tool, as it supports AutoCAD, .DXF, .3DS, WaveFront, Imagine, LightWave, .FLC, and .JPG file formats, along with many others.

Plus, you get a trueClips CD full of royalty-free textures and 3D Objects.

You'll need a 486/DX processor (although Pentium is the preferred configuration), Windows 95 or Windows NT, 16MB of RAM (better 32MB; better still a whole lot more), and 20MB of free disk space. Price for the whole thing is $795. trueSpace2 users can upgrade for $199.

■ Caligari Corp.
800-351-7620
415-390-9600
www.caligari.com

Texture Creator 2.0

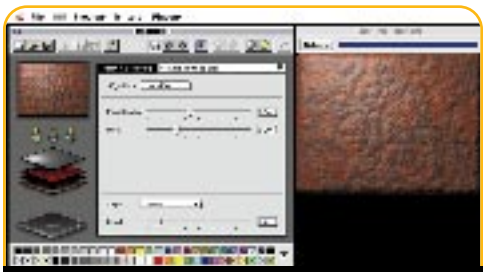
THREE D GRAPHICS IS offering a photorealistic texture creation solution with its product Texture Creator 2.0. Based on procedural rendering, Texture Creator builds a specific texture from layers that vary four properties: lighting, with which users control one to three spot lights to cast highlights and shadows; shader layers, which set the material (such as wood, marble, and so on), color, and blending of the layers; geometry, which allows manipulation of waves, ripples, and bumps; and edge, which gives certain types of borders to the texture tile.

The interface is handy, with precise control over variables such as number of layers (one to seven), opacity of layers, and color balance. One neat feature matches the edges of the texture tile for seamless tiling on web pages.

A Texture Wizard guides users through the process of creating textures for a wide range of purposes. And the package

ASTS

OF GAME DEVELOPMENT



Wrinkled Slab, one of the preset textures that comes with Texture Creator.

includes over 200 preset textures and exports a wide range of image formats.

Texture Creator 2.0 is shipping for PowerMac (MacOS 7.5.3 or higher), Windows 95, and Windows NT. You get a tidy little CD-ROM for $129.99.

- Three D Graphics
  800-913-0008
  310-553-3313
  info@threedgraphics.com
  www.threedgraphics.com

NetImmerse

NUMERICAL DESIGN LTD. has released its 3D game development toolkit, NetImmerse. NetImmerse is a run-time library written in C++ that lets developers create optimized, scaleable 3D content. The lowest-level rendering layers can be specified as OpenGL, Direct3D (immediate mode), or NDL's own proprietary software rendering layer.

Images from an upcoming F22 game by NDL customer Interactive Magic.

The toolkit includes routines for creating real-time morphing terrain, 3D spatialized sound, and highly optimized collision detection. Source code is also available.

- Numerical Design Ltd.
  919-929-2917
  sales@ndl.com
  www.ndl.com

The IKaID Library

THE IKaID LIBRARY FROM Blaze Software contains math and physics routines for resolving movement of 3D models. Included are inverse kinematics, dynamics, a library of 3D transformation routines, and a nonlinear equation solver.

All routines are included as 32-bit C++ class libraries for Windows 95 and Windows NT. You can get your IKaID Library for $169.

- Blaze Software
  310-772-8155
  blazing@GTE.net
  home1.get.net/blazing/fast.htm

Smacker 3

SMACKER IS A VIDEO-, animation-, and sound-data compressor. Smacker 3 has just been made available by RAD Game Tools. New features in this version include data–rate-based compression, direct AVI compression, and DIBSection support. The SDK is available for DOS, 16-bit Windows, Windows 95, Windows NT, Win32s, Mac, and PowerMac, and apparently will soon be available for Sega Saturn.

The Smacker SDK is licensed on a per-product or per-site basis, starting $3,000 per-product. The Smacker utilities will let you use previously created Smacker videos and can be downloaded for free from RAD's web site.

- RAD Game Tools
  801 322-4300
  www.radgametools.com/smk3.htm

PERSONNEL ISN'T THE ONLY thing that gets shuffled in reorganizations. 3DO, Interplay, and Acclaim were among those that announced they were canning titles to focus sales and costs. However, those coming off of a hot sales year went in the other direction. THQ, Eidos, EA, and Activision all leaped into new deals.

THQ AND PSYGNOSIS INKED a deal whereby THQ will publish many of the Saturn versions of Pysgnosis' games. Psygnosis itself is getting ready to open a string of US development centers.

EA bought into Don Daglow's Stormfront Studios (Daglow was a key EA producer for many years). Activision signed a deal with Bruce Willis for an upcoming title and acquired the rights to the RPG universe HEAVY GEAR from Target Games.

GT Interactive, fighting the skepticism reflected in its stock price, kept up its breakneck growth plan by forming its own internal development group, CaveDog Interactive. GT also signed 5D Games to a world-wide publishing agreement and purchased One Stop Direct, a UK-based budget software publisher.

Interactive Magic announced that Interactive Creations Inc. would be merged into the company and become I-Magic Online. Interactive Magic will pursue a "virtual battlefield" plan by publishing six titles, including AIM2 ABRAHMS.

Developers and analysts should stay alert for all the adjustments in companies' strategies leading up to the CGDC and E^3.

ON A FINAL NOTE, DIABLO recently shipped to a huge reception throughout January and February. Next issue, I'll be highlighting the string of recent announcements that have the RPG genre making a big comeback.

NEWS RESOURCE FOR THE MONTH. The news resource for this month is Intel's custom news service, which arrives via e-mail. You can subscribe to it at www-pwn.intel.com/.

As Intel pushes the SIPC, MMX, and other major multimedia and home computer initiatives, you'll find this easy way to keep in touch with the company's announcements.

by Chris Hecker

An Open Letter to Microsoft: Do the Right Thing for the 3D Game Industry

A debate is raging in the game development community on an incredibly important topic: 3D APIs for the PC, and specifically, Direct3D versus OpenGL. This debate has its share of contentless flames, but at its core is an issue that will affect the daily lives of 3D game developers for many years to come. Being one of these developers, I care deeply about how this issue is resolved. To put it very bluntly, I believe it would be best for the 3D game industry if Microsoft canceled Direct3D Immediate Mode and put all their 3D immediate mode resources behind their OpenGL team. This article will give my rationale for that statement.

The API issue has many twists and turns. Debating it is like wrestling Jell-O; every time you think you've got it cornered, it squeezes out somewhere else. For this reason, I'm going to proceed very methodically and lay out a logical framework for my opinion. I'm sure to leave some holes through which someone can squeeze if they are intent on disagreeing with me, but I think I'll provide a vast preponderance of evidence to back up my claims. Note that most of the ideas I'll present have been stated by other people in various places, so I can't claim to have originated many of these arguments myself.

Background

First, some clarifications: When I refer to Direct3D in this article, I mean Direct3D Immediate Mode, not Retained Mode. Retained Mode will be useful to some developers, but high-end 3D games probably won't use it since they need more control over database traversal and culling. Just to be totally clear, Direct3D is a Microsoft-designed API; OpenGL was originally designed by SGI, but is now handled by an independent Architecture Review Board (the ARB, where Microsoft, SGI, and several other vendors have voting seats). In the space I have here, I'm not going to be able to describe either API, so I'm going to assume you're familiar with both (check the references at the end of the article for more information on the APIs themselves).

Next, the history: This debate has been going on for a long time. I stopped looking for the original "Direct3D versus OpenGL" Usenet posting when I found one (a reply, no less) dating all the way back to March 1996. The conventional wisdom used to be that OpenGL was inherently slow — too slow for games — and that Microsoft had to design their own API. I bought into this wisdom when I was at Microsoft (and even helped spread it there) about two or three years ago. In fact, everyone I knew was convinced OpenGL was big and slow, and the only solution for games was a new and different API. In retrospect, I realize that didn't understand the technical issues; what's worse, I didn't know that I didn't understand the issues. Even worse yet, no one within earshot understood the technical issues well enough to explain why we were all wrong. The people who really understood OpenGL were in the workstation business at this time, and didn't realize 3D games were about to become an important market segment.

Where's Physics, Part 4 ?

I'm going to take a short break from our physics series to cover a topic of great importance to the 3D game industry. I'll be back with physics next issue.

My opinion started to change last year, after a few important events. First, a bunch of people at SGI got fed up with Microsoft's game evangelists telling developers that OpenGL was inherently slow. They decided to prove that it was at least as fast as Direct3D — if not faster — with a demo at Siggraph '96. This event got everyone's attention, and indeed, focused it on Microsoft's implementation of OpenGL, which was starting to get pretty fast as well. I took an interest in the issue at this time, and started reading Usenet posts by knowledgeable 3D engineers from many different companies. Eventually, I became convinced not only that OpenGL wasn't inherently slow, but that it was inherently faster than Direct3D at the limit, for reasons I'll detail below. Next, as everyone probably knows, John Carmack of id Software released a position statement in which he made known his choice of 3D API: OpenGL. He ported Quake to OpenGL to prove that the API has what it takes for the highest end game programming. Finally, Microsoft announced plans to update Direct3D to address some of the issues people were raising.

That's the history in two paragraphs. You'll notice I keep using the term "inherent" with regards to performance. I should describe what I mean by this. When I say API A is "inherently faster" than API B, I mean that on the vast majority of hardware, the driver and application writers for A will have more opportunities for optimizations, and programs running on top of A will be faster in general, than equiva-

lent programs running on B. So, clearly a bit of hand waving and faith is involved in saying one API is inherently faster than another. Still, I think it's possible to take knowledge of the problem domain and make a convincing case for "inherent speed" based on things like memory bandwidth and access patterns, bus speeds, existence proofs embodied in current high-end hardware, and so on. The inherent speed is important, since we don't want to run into performance ceilings imposed by our API.

This argument takes place on many levels. I'm going to show that OpenGL is inherently faster than Direct3D. However, even if they're just equal in performance, the conclusion that OpenGL is superior still holds, as you'll see. I'm reminded of one of Dave Baraff's dynamics papers that I read recently. To paraphrase, "We can always solve this problem because A is never singular, but even if it is singular we can solve the problem anyway for this other reason."

Performance

Let's start with performance. The first and most telling thing about the performance issue is that no one at Microsoft (or anywhere, actually) has been able to give me a single technical reason why Direct3D is even OpenGL's equal in 3D performance, let alone its better. I'm talking about technical engineers on the Direct3D team; when asked point blank for an architectural reason why Direct3D is (or could be) inherently faster than OpenGL, they admit they have none. The only people who routinely say Direct3D is faster for games are the Microsoft game evangelists, and they're marketing people, not technical engineers. Microsoft constantly refers to Direct3D as being "designed for games," but when pressed, no one seems to be able to come up with how that translates into actual performance improvements over OpenGL.

Although this lack of technical response is pretty damning for Direct3D, I'll still cover the specific technical reasons that OpenGL is an inherently faster immediate mode API. If you already agree that OpenGL is inherently as fast or faster than Direct3D and you just want to see all the other overwhelming reasons why OpenGL is superior, you can skip this whole next section. It's going to be heavy reading — remember, I need to try to prevent the Jell-O from squeezing out.

The performance comparison has two aspects. First, we'll compare OpenGL to Direct3D execute buffers. Then, we'll compare OpenGL to the new Direct3D DrawPrimitive API.

Execute Buffers

When comparing OpenGL's output model to Direct3D's execute buffers, we must consider the three types of 3D vertex data: static data, where the vertices of the model don't change relative to one another (like the fuselage of an airplane or the arm of a

hierarchical model); dynamic data, where most vertices change every frame (like undulating water, morphing geometry, or a continuous-skinned animating figure); and finally, partially static data, which is a mix of the previous two.

STATIC DATA. For static data, OpenGL's display lists are clearly better designed than Direct3D's execute buffers, because they're opaque and noneditable. By "opaque," I mean the application doesn't know the format of the display list. By "noneditable," I mean once created, the display list's internal data cannot be changed. These two features together make it possible for driver writers to compile the display list into a format appropriate for their hardware and to upload the list onto the card, even into memory that's inaccessible to the application.

Contrast OpenGL's display lists with Direct3D's execute buffers, which have a fixed format dictated by Direct3D and are editable by the application after being created. These features make life much harder — if not impossible — for the driver writer who'd like to optimize for performance. Direct3D simply does not allow 3D hardware manufacturers the flexibility they need to optimize static vertex data rendering. In the interest of full disclosure, I should point out that Direct3D does have a little known and currently unimplemented function you can call to "optimize" the execute buffers and make them noneditable, but it's unclear whether this feature will ever be implemented. Even if it is eventually implemented, it still wouldn't allow Direct3D's theoretical performance on static data to match OpenGL's theoretical performance; OpenGL display lists have other performance-enhancing features execute buffers do not, such as the ability to invoke other display lists.

DYNAMIC DATA. 3D hardware accepts your application's rendering commands in one of two basic ways: IO or DMA. In an IO-based architecture, the 3D card memory maps its data registers and backs them up with a FIFO, allowing the data to be written directly to the card by the CPU. In DMA-based hardware, the 3D card starts up an asynchronous memory transfer to read the data directly from main memory without needing the CPU. There are, of course, hybrids that use both. Each technique has advantages and disadvantages, and which is faster is still an open topic of research (for example, the current high-end SGI hardware switches dynamically between the two, and in the PC game world, Rendition prefers DMA and 3Dfx uses IO). Since there's no consensus as to which is best, it's vitally important that the 3D API allow either or both, optimally and without preference.

Consider how OpenGL's data-formatless function-call model handles dynamically changing vertex data on both types of 3D hardware. On IO hardware, the OpenGL model allows the application to generate the vertex data and get it out to the accelerator with as little data conversion and cache pollution — and as much fine-grained parallelism — as possible. On DMA

hardware, the model allows the data to be written directly to the card's DMA buffers in exactly the optimal format for the hardware, without conversion to or from an intermediate vertex format. The OpenGL model scales from rasterization-only hardware (such as current PC boards) all the way up to high-end hardware that can accept the model-space vertices directly.

Direct3D execute buffers, on the other hand, deny you maximum performance for dynamic vertex data in one of several ways. On IO-based cards, the application must write out an execute buffer full of vertices to main memory, which will then be parsed by Direct3D and written to the card. This means that your vertex data came out of your application, was reformatted into a big, bandwidth-munching main memory buffer, then read out of main memory, munged by Direct3D, and sent to the card. This extra layer of memory traffic and reformatting is death for high-throughput rendering. OpenGL sends vertex data either directly to the hardware or through a much smaller and cache-friendly single-primitive buffer.

Now consider how Direct3D execute buffers work with DMA-based designs. The best case here (or perhaps I should say the "least-worst" case) for Direct3D is if the 3D hardware can somehow directly DMA and parse the execute buffer format. Consider what is involved in this task alone; the hardware must be able to read all current and future Direct3D vertex and primitive formats, all commands, and all flags. Even the Direct3D engineers admit that this isn't likely to happen, and the hardware engineers from the PC 3D hardware companies I've talked to agree (no hardware that I know of does this, either). In addition, the application must cope with execute buffers that are different sizes on different cards, none of which might be the optimal size for your application's data.

It gets even worse for Direct3D if the DMA-based card doesn't parse the execute buffers. In this case, the application writes out an execute buffer, then the Direct3D driver must parse the buffer, write out the data again to the card's DMA buffers, and then start the DMA transfer. This is yet another layer of memory traffic over and above the IO-based card example.

Finally, some have proposed making the execute buffers writable and resident in the card's memory. In this case, the hardware designer not only needs to handle all the parsing problems mentioned previously, but also must design the hardware to allow the CPU random access to — and even reading from — the card-resident execute buffers and to have interlock protection to prevent modification of the buffers while they're being executed. No one, to my knowledge, has implemented this kind hardware.

It's clear that for dynamic vertex data, OpenGL allows much more flexibility for both the application and the 3D hardware. This flexibility directly translates into better memory access characteristics and higher performance. I should point out that a rasterization-only card will almost always deal with dynamic vertex data. This is obvious with a moment's thought; as the camera moves in 3D, the screen coordinates of all the primitives will change in each frame. Thus, this section directly applies to the vast majority of currently available commodity 3D cards for the PC.

PARTIALLY STATIC DATA. Finally, we come to partially static vertex data. If I had to pick the weakest part of my performance argument, it would be here. If you assume that you only need to update a few vertices per frame, it seems that editable execute buffers could have some benefit. However, this is only the case if the card can directly DMA and parse the buffers. If Direct3D has to parse the buffer itself, then you might as well have dynamic vertex data, and then all the previous section's criticisms apply. Also, I've yet to hear a really compelling and uncontrived example for partially static vertex data. The more the static vertex data gets, the easier it is to break up into multiple, completely static sets (OpenGL's display lists support this mixing of static data by allowing them to invoke other lists). The less static the vertex data is, the more it starts to resemble dynamic vertex data. Add to that the disadvantage that you still have the data reformatting issues with Direct3D, and I'd say it's at best a wash for both APIs on this one.

So, I think that's all she wrote for execute buffers from a performance perspective — OpenGL trounces Direct3D. I'd be very interested to hear from anyone with any other points I've missed, either for or against execute buffers.

DrawPrimitive

In some ways, it seems that Microsoft agrees with my conclusion about execute buffers. The latest feature to be added to Direct3D is the DrawPrimitive API. This is a way to draw triangles without batching them up in execute buffers. I've heard different rationalizations for adding this API, from the performance limits of execute buffers that I just discussed, to execute buffers simply being "too hard for people to use." Whatever the reason, DrawPrimitive isn't going to avoid all of Direct3D's performance problems, although it will avoid some of the memory traffic problems associated with execute buffers. Direct3D will still have the data reformatting problems, for example (the API will take Direct3D's standard vertex formats), so your application will still have to read from its database and convert its vertices into Direct3D structures, and then pass a pointer to those structures to the API. One might assume applications could use Direct3D's vertex structures internally and save a conversion, but this puts onerous constraints on the application. OpenGL doesn't have these format constraints (see the paper comparing OpenGL to PEX, referenced below, for more details on this formatting issue).

Basically, at this point, Direct3D can choose between emphasizing execute buffers and running into the performance problems I've mentioned, or it can emphasize DrawPrimitive, in which case we're stuck with an immature and poorly designed clone of OpenGL that's missing some of the architectural decisions that make OpenGL fast. Either way, game developers — and players — lose as we give away performance and get nothing in return.

Everything Else

Personally, I feel performance scalability should be the primary issue Microsoft considers when deciding which API to support for game developers; I think I've shown that OpenGL wins handily in this arena. However, there are scads of other reasons for choosing OpenGL, even if we

ignore performance. I'll go into those now. The reasons are so convincing that even if OpenGL and Direct3D were somehow only evenly matched in performance, OpenGL would still be a better API for the game industry.

As I said before, no one at Microsoft was able to give me a technical reason why Direct3D was better than OpenGL. That didn't stop them from giving me a bunch of nontechnical reasons, which I'll address here.

DRIVERS. Currently (February 1997), there are more Direct3D drivers than OpenGL drivers available for Windows 95. However, this discrepancy is simply a matter of time, resources, and evangelism. All major 3D card manufacturers have now committed to doing OpenGL drivers (glQuake certainly helped motivate people on this front). Microsoft could intensify this development and shorten the gap by giving their OpenGL team more resources.

DIRECTX INTEGRATION. Direct3D is, by definition, integrated with DirectDraw. Still, nothing says OpenGL can't be integrated as well. In fact, by the time you read this, Microsoft's OpenGL-DirectDraw bindings should be announced and possibly available in beta. As with drivers, this will be nonissue in a matter of months.

EXTENSIONS. Some people at Microsoft claim they'll be able to extend Direct3D for game developers' needs faster than OpenGL could be extended. This is not only incorrect, it's backwards. OpenGL has an official and time-tested extensions mechanism, where vendors can do proprietary extensions first, then move them to multivendor extensions, and finally move them into the next OpenGL specification upon ARB approval (of which Microsoft is a voting member). Microsoft's OpenGL team has done published extensions already, as have SGI, 3Dlabs, and others. The only way for a vendor to get extensions into Direct3D is to beg Microsoft to put them in — there is no way to do it themselves. Hardware vendors always mention this as a major problem with Direct3D. What's more, OpenGL's structureless interface makes it much easier to seamlessly integrate extensions into the API; adding multitexture support to OpenGL is going to be trivial, while adding it to Direct3D is going to require creating a new vertex type.

To sum up, Microsoft can extend its OpenGL just as fast as they can extend Direct3D. As an added benefit to game developers, individual vendors can try out extensions without needing Microsoft's approval. This process is already in place and working in the OpenGL community. So, if you want to do a nifty 3Dfx-specific trick in your next game, you can do it with OpenGL, but not with Direct3D.

FOR FURTHER INFO

Usenet News
The never-ending Usenet threads are available on www.dejanews.com. Search for OpenGL and Direct3D and then hunker down for a week's worth of reading.

John Carmack's OpenGL Position Statement
http://redwood.gatsbyhouse.com/quake/jc122396.txt
If that site is down, you can find it on DejaNews. You can also find Alex St. John's reply to Carmack's statement on DejaNews.

SGI's OpenGL Page, including the 1.1 Specification
http://www.sgi.com/Technology/OpenGL

A Comparison of OpenGL and PEX
ftp://ftp.sgi.com/opengl/doc/analysis.ps.Z

Microsoft's Direct3D Immediate Mode pages
http://www.microsoft.com/msdn/sdk/platforms/doc/sdk/directx/src/directx_400.htm
http://www.microsoft.com/mediadev/graphics/drawprim.htm

Microsoft's OpenGL Docs
Microsoft has good OpenGL documentation on their web site as well, but I can't figure out how to get the direct URL. Basically, go to the following URL and select the "Documentation" tab, then "3D Graphics" in the little drop-down box, then wait for the Java applet to load. Good luck.
http://www.microsoft.com/msdn/sdk/default.htm

Brian Hook's Online 3D Papers
http://www.wksoftware.com/publications.html

OPENGL IS ONLY GOOD FOR SGI HARDWARE. I used to believe this myself, but I did a little checking. It turns out that people have done OpenGL (or its predecessor, IrisGL) on just about every type of hardware imaginable, from span renderers to full-geometry pipelines. Furthermore, consider the number of different OpenGL vendors. Check out glQuake running on a 3Dfx or an Intergraph. The more you think about it, the more you'll realize Direct3D's exposed vertex formats and execute buffers dictate the design of the hardware much more so than does OpenGL. That means there's less room for innovation by hardware designers, and again, we developers lose.

OPENGL HAS NO CAPS BITS. Direct3D has a mechanism by which you can test whether a driver supports certain features, like a Z-buffer or alpha blending (you test capabilities bits, or caps bits). In contrast, OpenGL requires that all features be implemented, whether in hardware or emulated. At first glance, it seems that Direct3D has the advantage here, but that's not the case. First, caps bits can't express the richness of 3D hardware. For example, a card that can Z-buffer and have destination alpha — but not both at the same time — is a real possibility, but you can't express that in caps bits. Second, just because a caps bit says a feature is supported doesn't mean you want to use it, since as we've all seen on this first generation of hardware, it might be slower than software (or it might not even actually be supported, given an unscrupulous vendor). The only true solution to this problem is to profile each feature at installation time and give the user a choice. You can do this equally well on either OpenGL or Direct3D.

NonDebatables

I think I've covered most of the debatable points. In addition, there are some points going for OpenGL that no one debates.

EASE OF USE AND ELEGANCE. No one argues with the fact that OpenGL is easier to use and more elegant than Direct3D. This was the central thesis of Carmack's position statement. He feels usability is even more important than the performance of the API. For more of

his opinion, you should read his statement for yourself. It's in the references.

SPECIFICATION AND CONFORMANCE. It's also inarguable that OpenGL is better specified. You can read this specification on the Web; again, check the references. The spec is based on years of 3D experience, and it does a great job of balancing specificity and ambiguity, allowing hardware manufacturers room to innovate while providing software developers with a consistent, high-performance API. In addition, each OpenGL implementation must pass a battery of conformance tests. Direct3D has no comparable specification, and no are conformance tests available as of this writing.

DOCUMENTATION AND SAMPLES. OpenGL is also better documented, with many good books about it available in stores. The few Direct3D books available only cover Retained Mode in any detail. There are tons of well-written OpenGL samples available on the Web, also in contrast to Direct3D.

Summary

Can't Microsoft fix these problems? Of course they can. They can continue to change and patch Direct3D until, as Carmack says, it "sucks less." However, it'll take them years to reach the level of polish and performance that OpenGL has right now; why should we developers have to put up with it? For the same effort, Microsoft can give their OpenGL team more resources to make OpenGL even better, and the whole industry benefits and advances.

Can't we have both APIs and let them compete for mindshare? Sure, but currently the Microsoft OpenGL team is prohibited from evangelizing OpenGL to game developers because that would run counter to the current "strategy." That's not fair technical competition. Also, I showed a draft of this article to engineers from several top-echelon PC 3D hardware manufacturers to get their technical review; they all said they'd say the same thing if they weren't afraid of Microsoft and the repercussions it would cause for their companies. Again, this is not competition. And again, we developers lose. As a final reason for not having both APIs, hardware vendors are forced to spread themselves thin to support more than one API, and driver support suffers.

So, for all these reasons — for the good of the game development industry — I urge Microsoft to cancel Direct3D Immediate Mode and fully embrace OpenGL as the immediate mode game development API of choice. I also urge game developers to take a closer look at Microsoft's OpenGL. If you like what you see, use it. Remember to share your opinion with Microsoft and 3D hardware manufacturers. Finally, if you're a 3D hardware manufacturer, get your OpenGL drivers done as soon as possible.

Chris Hecker usually inserts a funny bio here at the end, but he feels strongly enough about this topic that he'll refrain for an issue. He can be reached at checker@bix.com.

MONITORING DEVICES IN GAMES

What is a monitoring device, you ask? Monitoring devices are all those powerbars, radar screens, ammo counts, and cardiographs that generally clutter up the screen. Monitoring devices provide information that can't be implied in the game play environment. Monitoring devices are used for the players' sake. Good interfaces communicate vital information that:

- let players set goals
- tell players how they're doing
- and empower players with knowledge, so they can get deeper into the game.

BY TZVI FREEMAN

Monitoring devices also enhance the game itself by:

- increasing thrill and suspense
- and providing rewards for achievements.

Monitoring devices should not:

- distract the player from the fun of the game
- clutter the screen, especially if valuable play space is covered
- confuse the user with ambiguous data
- or detract from the meaningfulness of information that is truly important. (This relates to the "noise to signal ratio," which I'll qualify later.)

So, as with any powerful tool, balance is important. You can't just say, "Hey, let's throw in a turbosprocket rust-level display!" You have to use some discrimination. That discrimination is the topic of this article.

Doing Without

The first question is, do you really want any monitoring devices at all? Is the distraction really worth it? One of the most basic elements in any form of entertainment is something the author Samuel Coleridge called the "suspension of disbelief." You want the players to forget about reality, to forget that they are sitting in front of a computer and playing a game. You want the game to become reality.

If a monitoring device is not integral to the game environment, then it takes the players out of that environment. It reminds players that there is a computer in front of them running a game that's only software.

Of course, there are games where a console full of monitoring devices is a natural part of the game play environment, such as a car racing game or a flight simulator. Even then, players should have the option to hide the console so that they can totally absorb themselves in the scenery and the action.

But when it comes to playing an organic being — such as in a fighter or a pistol-in-hand shooter — life-bars and such are out of context. Personally, I got in a lot of fights as a kid, but I never once saw a life bar flashing above my head. However, it is true that my heartbeat rose significantly, and I may have lost some blood and collected a few perforations and bruises. Now aren't those far more meaningful monitoring devices than a linear bar?

POWERBARS, SCORES, AMMO COUNTS, AND OTHER INTERFACE DISPLAYS CAN EITHER ENHANCE GAMEPLAY OR DISTRACT AND CONFUSE. HERE ARE EXAMPLES OF WHAT TO DO — AND WHAT TO AVOID.

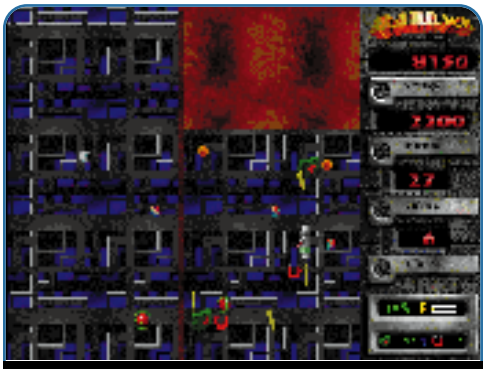
BUSHIDO BLADE is fighter game in a very realistic 3D environment. The designers did without life bars altogether.

Technical and budget considerations may make it necessary to go for non-contextual monitors, but that doesn't mean that they're good. The more realistic an environment you provide, the cornier all these devices appear when laid over it. Once you get into true virtual 3D, where context is everything, they get plain bad. When it comes to action games that addict the player by demanding constant full attention, distraction becomes a major fault.

A little background on how our eyes plug into our brains will help here. We have two fields of vision: central vision, the stuff we see in our "line of focus," and peripheral vision. The eye and the brain, being amazingly efficient devices, treat these two fields differently.

Targeting your central field of vision is a spot on your retina known as the macula (Latin for "spot," as in immaculate — spotless. See, Latin isn't so hard after all!). Here are heavily concentrated photoreceptors that can take in information at high resolution. When this information gets to the brain, a likewise heavy concentration of neurons are there to process it. That's why what you are thinking about is generally related to what you are looking at; there's simply more of your brain's CPU time allotted to that field.

Your peripheral vision doesn't have such fine resolution. But peripheral vision does pay attention to motion at least as well as central vision does. This is a survival issue; if you didn't notice a brick flying at you from the periphery, you might not survive very long. Attackers aren't usually considerate enough to attack from within your central vision.



BARRACK, from Ambrosia, is a stimulus-rich, high-action game — so much so that just a glance over at the numbers on the right and you're dead. So why bother taking up valuable real estate with something the player won't even see?

Now, go over to a friend who is deeply engaged in a task and make a sudden movement of your hand within her peripheral vision. Note the movement of her eyes, head, and upper torso. You may need also to duck. Here is another survival instinct: unexpected movement in the peripheral vision will distract the observer and direct her full attention toward the movement.

An important implication of these dual fields of vision is that if you want to give players the pleasure of completely immersing themselves within the game action, don't distract them with extraneous lights flashing and objects moving in the periphery.

Of course, in some instances you may want to distract the user — à la smoke and mirrors — such as when you want to cover up for suggested action that isn't really there, or create excitement when nothing is really happening. Otherwise, when your game depends on the addictive quality of basic, neurological, immediate stimulus-response, distraction is something to be avoided.

Alternatives

There are plenty of alternatives to powerbar-type monitoring devices, some of which I've touched on already. I call these implicit, as opposed to explicit, feedback devices. There are a surprising variety of ways to provide feedback implicitly in a game.

APPEARANCE. Try altering the way people and objects appear. If you can't find any other way, just put numbers directly on the objects. Before you do that, though, ask yourself, "What would people expect to happen to this object when it reaches this state?"

Perhaps you could apply different texture maps to your objects. Or you could remove (or add) a few significant polygons. Perhaps things could smoke or crackle with sparks. Perhaps the player's mobility could become impaired, rendering him clumsier or slower. There's a whole bag of tricks that's barely been explored.

SOUND. Certain game states can trigger sound effects and music that fit in context. Games are getting better at this every day.

Sound deserves a few comments. Anybody who drives a stick shift knows how much we use sound to monitor what's happening. In real life, we rely on the many advantages that our ears have over our eyes.

Sound plays a dominant part in the "fight-or-flight" reflex. Generally you first hear, then see. Sometimes you never even get a chance to see. So danger is generally first communicated by sound, not sight. This could explain why sounds have so much more influence over human emotions than images.

To observe how sound dominates your psyche, try this experiment: Watch a super-violent movie, but with the calm music of Beethoven's Pastorale in the background. Now try playing the movie's soundtrack to a Care Bear cartoon. Don't let your kids see the latter — they'll have nightmares for weeks.

In short, sound can be a far more effective way to impart a sense of danger, success, failure, and all those other feelings you wish to communicate with your monitoring devices.

The most compelling reason for using sound, however, is that it leaves the eyes free to concentrate on the task at hand. More than that, it leaves the brain free to concentrate on the task. The brain is a true multitasking device — it uses different banks of neurons to process sounds and sights. So, if you distribute information using both sight and sound, you can communicate far more effectively than if you try to drive everything up just one channel. Thus, your players are more engaged. More of their brain is immersed in the experience, so they're more likely to become obsessed with playing it.

Some sounds that you may not have considered for monitoring status include

- Heartbeat
- Sound amplitude
- Sound frequency
- Music tempo
- Tone of voice
- Sirens
- The sounds your car makes when something's wrong
- The sounds you make when something's wrong
- The sounds you make when things are all right.

What about voices? This option seems to escape many designers, but there is no reason you can't just outright yell at a player, "Hey, you've got only one life left!" Or even, "You have five lives left."

Much of what I've said about multitasking the sight and sound channels of the brain applies to speech as well. Processing speech is quite different than processing sight or sound. Speaking to players engages yet another channel in their brain. In other words, players are even more engaged.

An important factor to remember when using speech is the influence of a speaker's tone of voice. Try saying something nice to your mate in a mean voice, and you'll see what I mean.

Perhaps you've read the case of the air-traffic controller who was proud of her cool, collected composure in the face of disaster. When a stewardess and passenger were sucked out of a plane in midair, she smoothly managed the landing and alerted the ground emergency crew. The ground crew, however, came unprepared, not expecting the kind of emergency they actually encountered. They laid the blame on the air traffic controller, who was severely reprimanded for lacking a tone of urgency in her voice.

It's worth it to use voice professionals. Crisp enunciation and knowing how to talk into a microphone make a big difference.

SCREEN LUMINANCE. I haven't seen this used anywhere yet. I haven't even tried it myself. But luminance has been used for decades to set mood in movies and TV, so why not in games?

SURREALISTIC TOUCHES. This category might include effects such as auras. I know people who claim they see these things around all of us. Apparently, a lot of information can

In TERMINAL VELOCITY by Terminal Reality Inc., you look up at you status/radar device, which feels quite natural. This device is packed with data — if you know how to read it. The rest of the displays, however, are hard to read and poorly placed. They may have been better off in the bottom left.

be gleaned from auras. If I can't see them in real life, why not let me see them in my play life?

For the uninitiated, an aura or halo has size, color, and shape. All of those factors can provide plenty of information. It's not technically difficult, either.

The "Pros" of Those Messy Things

Now that I've got you ready to throw out every life bar, gauge, and flashing number on the screen, let me say that you often really need them. You can't always rely on sound alone. For one thing, sacrilegious as it may be, players may well turn off the sound. For another, sound has a temporal quality to it; you hear it, and then it's gone. And looping sound clips — as anyone who's driven with a back seat driver knows — can be really annoying.

Images, on the other hand, are always there when you need them. You can read them at your own pace — when you're ready — and rely on them to be there when you need them again. Images have persistence. Furthermore, images can communicate complex information in an immediately useful way. Most people raised in our highly visual culture have difficulty translating audible data into meaningful information. Give them an image, however, and they've got it. One picture can be worth a megabyte of beeps. Factors to look at include:

- How urgent is the player's concentration on the action? Can he afford to take his eyes away for a moment? In a war game, race, or flight simulation, player can often snatch a glance to the side during short breaks of relative calm. In many arcade-style games, however, there's barely time to breath.
- Is this information that the player will need to refer to often throughout the game? A map, radar display, or shopping list are examples of such information.
- That primal question of game development: How long is the game play? Longer game play makes a better case for recording data and making it available.
- Is this complex information that needs some sort of visual representation to be clear to the player? Again, a map is a good example. But there also may be information that you can represent with a chart or representational image.

So you may well want to use sound and sight in tandem to communicate, leaving players the option of easily hiding the visible displays. That's not a licence to use distracting displays *gratis* — you still need to justify everything you place on the screen.

WARCRAFT has a map that is at once simple and highly informative. Information about your soldiers and peons is represented as a list of numbers, but this is O.K. in a game that doesn't demand constant fast action.

Guidelines for Displaying Data

Now we're left with the other issues of clutter — ambiguous data and signal-to-noise ratio. Following are some suggestions on how to deal with clutter.

THE BOTTOM-LEFT PRINCIPLE. Know that the most valuable piece of real estate on your screen, next to middle-center, is towards the bottom-left. Interface designers generally feel that the human eye, when left to its own, will fall to the bottom left of the screen. Game designers can implement this tidbit of information by placing an upward-thrusting image in the bottom-left, thus keeping the eye moving and engaged. Nintendo makes use of this theory by placing the most vital piece of data in big letters somewhere near that corner. Whatever you do, don't put auxiliary or semi-useless data there.

It's quite conceivable your design will obviate this principle. Other objects, or the design of your console may lead the eye in different ways. But it's something you must be aware of when laying out your screen.

id made a great move in using a face to represent your state of being. A number here would have sabotaged the idea being communicated.

TOP, SIDE OR BOTTOM? Try this experiment: Ask some people to imagine something they've seen many times. Observe the direction each person looks. Try it with several people several times. Now ask these people to make a difficult decision. Observe again.

Neat, eh? You can sometimes even follow a person's thought process by the directions the head tilts; one looks up to recall a piece of information and looks down to consider its impact on whatever he's thinking about. We strongly associate looking down with deep thought and looking up with visualizing. Makes sense, since there's usually a lot more to see when you look up than when you look down. My point is that for us thinking animals, this is a stereotyped reflex. In other words, it feels natural.

You can take advantage of these idio-

syncracies in the placement of your devices. Maps and things that require players to visualize ("Where am I?" "Who's coming after me?") feel best at the top of the screen. Numbers and things that players use for making and implementing decisions feel better at the bottom.

LOGICAL ORGANIZATION. Applying logic to screen layout seems obvious, yet many game designers completely ignore it. You need to group your devices by the relevance and category of the information they convey.

HOW MANY? Just as with students in a classroom, every monitor you add detracts attention from those that are already there. The human mind can only handle so much information at a time. Expert flight simulators and such are obvious exceptions. In those cases, the complexity of the console is part of the game.

If the game is heavily action-oriented, three devices is a lot. In any case, six is pushing it.

HIGH-DENSITY INFORMATION. "But," you cry, "I've got so much I need to tell the players! And it all makes the game so much more fun!"

So find ways to communicate a lot with a little. That's the way it is in real life. We are used to taking in images that, at one glance, tell a world of information — such as the expression on your boss's face when he says he wants to talk to you, or the look of the soup your fiancé ordered for you at the Chinese-Italian restaurant.

In science and in business, we do the same thing. With a single chart, we communicate a lot of vital data. We use color (which, on its own, carries much information), x and y coordinates, thickness of lines, and so on. We provide some information graphically, some with text or numbers, and some both ways.

Status displays that use color, depth, height, and numbers can do the same. So can a well-designed radar or map. The advantage of providing information this way is that in a single glance at the same place every time, players can find vital information. And it takes up less space. And it also looks neat.

MINIMALISM. Minimalism demands skill and talent. It means, "Just say what needs to be said and no more." One of the most annoying things a person can do is to ramble on and on when all that was asked for was

Game Design Defined

Recently, a representative from Nintendo told my class that his company's top priority is to define the "form" of electronic games. After all, a generation is now growing up taking technology for granted. To them, anything is possible — you can forget trying to wow them with bleeding-edge technological dazzle alone. They're going to want good games. Period.

To produce good games consistently and not waste money on mediocrity, companies like Nintendo need to be able to determine whether a game is good even before the first prototype is begun. And they're going to need to teach people how to make those great games. The only way to do that is by first defining what makes a good game.

At DigiPen, we've started setting out some of the elements of the electronic-game form in a way that can be discussed with clarity, and maybe even taught.

The only thing more exciting than playing great electronic games is designing them. It involves the diverse feats of technical know-how, artistic talent, storytelling skills, a good feel for psychology and biophysics, and the ability to articulate your ideas on paper so others can implement them — plus the humility to listen to other people's ideas, too. Game design is definitely one of the greatest challenges the human mind can face.

The best place to begin is with something called "Human Interface Design" — an area into which much thought and research has been invested over the last two decades. Human interface generally refers to the two-way street where technology and its human masters meet. But in electronic games, technology is the human interface — an interface between human imaginations.

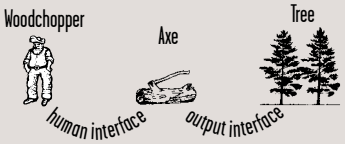
For a lucid illustration of this point, let's start with an abstract model of human activity. We can break any such activity into three parts: the actor, the activity, and the effect. If this is a technological activity, two objects will be involved: the technological device being used to perform the activity and the object being affected by it.

Let's say there's a fellow out there using an axe (a technological device) to transform trees into firewood.

Notice that there are now two interfaces. The axe needs a sharp output interface to the tree, but it also needs a human interface so the wood chopper can wield it without chopping his own hands. The same model can be applied to swords, pens, typewriters, automobiles, or even, lo and behold, computers.

But now, let's look at what happens when we aren't working, but only playing a game. In games, as we were all told many times during primary school recess, nobody gets hurt. In other words, the players all pretend they're doing something, but nothing of any significance is really getting done — except in the minds of the players. In their minds, they can achieve goals far beyond the realm of practical reality. But as far as the real world is concerned, the effect of their activity is only an abstraction.

If our woodchopper was actually the woodchopper's three-year-old kid playing "chop the trees," we would model his activity like this:



The diagram is similar to our previous model, except that the ouput interface and the object being acted upon — the tree — only need to exist in the player's mind.

That's why the kid can get along with a toy axe. It doesn't need the sharp output interface because it doesn't really need to chop the tree. The child is not out for firewood, or any other real result. All that interests him is the activity as an end in itself. And when you don't need results, you can fill in an awful lot with just your imagination. As long as you can swing it, and it looks like an axe, it's good enough.

The beauty of the electronic game, how-

Continued on p.30

one vital piece of urgent information. An image that displays more information than needed is similarly annoying.

What you are aiming for is immediacy — a rapid flow of knowledge. You want the player to throw one glance at this image and get the whole picture in a flash. If you throw in all sorts of extraneous frills, you're just getting in the way. Just as you want to find the fewest words possible to communicate the most content, you should find an image that communicates the most with the least.

INTEGRATION. Getting back to that "suspension of disbelief" issue, you don't want the device to take players out of the game. The more you integrate the image into the theme of your game, the better you can avoid this.

THE CONCEPT OF IMMEDIACY

$$\frac{\text{Information}}{\text{Detail}} = \text{Flow of knowledge}$$

A powerbar may fit in a sci-fi shooter, but I really can't see it working in a medieval fantasy. There, an aura may work better, or perhaps the tone of music would be enough to indicate a change in status.

If your game uses a Japanese Ninja theme, find metaphors from that world to represent whatever you need to communicate. Similarly, if the action takes place in a laboratory, LCD displays and line-graphs are in order.

You must also consider the matter of initial appeal. When your prospective players first look at your game, likely a major factor in their decision to play it will be the look of the devices. These will tell them not only how neat the game is, but what type of a game it is as well.

Which brings up another major decision: whether to use a console display. If your game takes place in an environment where a console makes sense, you may want to have one there initially, just to communicate to the players what this game is all about. Then you can give them the option of hiding it. In my observations, most players, if they are able, end up hiding these things after they have familiarized themselves with the game.

If there's no good reason for a console, try floating your displays over the scene. A good artist should be able to pull this off in a way that looks as though the monitors are on a distinct plane from the play environment, while still looking natural.

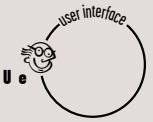
WHEN ARE NUMBERS IMPORTANT? Why not use numbers whenever you can? Numbers, percentages, and such are often the best way to communicate. Many times, however, they are just clutter, or at least a distraction. A good rule of thumb is to first determine whether the information you are attempting to represent is a quantity or

Game Design Defined (Continued from p. 29)

ever, is that the activity itself is only an abstraction. In an electronic game, you're not really doing anything. Actions only happen in that wonderworld of cyberspace that doesn't really exist — at least not like the existence into which we were born. To paraphrase Woody Allen, virtual reality is neat, but when you want a good steak....

In an electronic game, you wield a virtual axe, which is just the way that some game designer is representing an axe so that you, the game player, can allow yourself to believe (we call that "pretend") you're chopping wood. But there's nothing really there. Not even a toy axe.

In other words, the virtual axe and the virtual woodchopping are only interfaces between two imaginations — the author's and the player's — and between the player's imagination and the player. If it is a multiplayer game, it is also an interface between multiple imaginations. Here is our diagram of the electronic game:



This line of discussion brings us to something that many people have been trying to say in different ways: The ultimate game is nothing more than an interface between human imaginations. Anything in a game that is not fulfilling that function is just bird droppings on the windshield.

Sounds exciting, doesn't it? I don't think we're really there yet. We're going to have to pull a few more technological stunts first. But there's a lot we can do with the technology we have right now — so long as we keep the goal in mind and follow a few basic guidelines.

The Table at the end of this sidebar lists some of the most important guidelines that extend from what I've just said.

A final caveat is in order. The game itself doesn't determine whether it's good or not. Neither do the people who made it. It's the players who play it. We can struggle to create some guidelines, but in the end there are no Absolute Truths of Game Design — it's all entirely subjective.

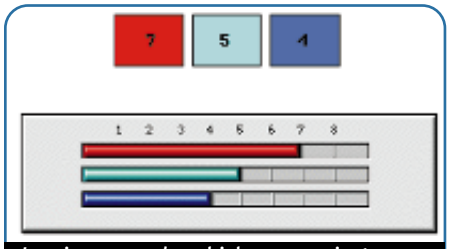
But we'll to try anyway, 'cause that's all we can do.

HOW TO TELL THE GOOD GAMES FROM THE BAD.

The Good Game	The Bad Game
A good game empowers your imagination.	A bad game gets in the way.
A good game makes you feel in charge.	A bad game restricts you with artificial restrictions.
A good game is transparent. You only feel your own mind, the other players, and the ideas	A bad game keeps reminding you that there is a game here.
A good game lets you into its creator's imagination.	A bad game feels like a machine wrote it.
A good game lets its players feel each other's personality.	A bad game only lets you touch each other superficially.
A good game fits the human being like a glove.	A bad game demands the human be fit into it.

a quality. How many bullets are left is a quantity. So is the score. How alive a character feels is really a quality — no matter how many games represent it with a number.

If the information is a quantity, ask yourself how important accuracy is in this case. Is it vital to distinguish between four and three? Or is it enough to have just a relative idea of where things are?



In microseconds, which communicates faster?

If accuracy is important, use a number. If an image also helps, you can place the number within the image. If, on the contrary, this is a case of vital accuracy where an image may distract, put a really big number with no accompanying image in the bottom-left corner. In many games, the number of lives remaining would fit into this category. In a racing game where you can't see all the other racers, your position in the race (First, Second, or Third) would fit into this category.

If, however, what you are trying to represent is a quality, a number is just noise over the signal. A number simply confuses what it is that is being communicated. You want to communicate a feeling or a sense; a number is saying a shallow, cold fact.

Another very important point to keep in mind is that numbers are an abstraction — that is, they are not analogous metaphors to the information they represent. The mind has to take numbers and translate them into a real idea. In many cases, this is an extra burden on the player, and a failure of immediate communication. This is especially the case when the player needs to compare data. In such a case, a series of bars — a "bar chart" — works faster and better. We all learned in first grade that 14 is more than 12. But it's still easier to see that one bar is taller than the other.

Similarly, even when a number is used to show how much ammo is left, if we're talking about small numbers, it helps to display a line of bullets. The more concretely you can communicate, the closer you are to the mind to which you're communicating.

What about numbers that pop up on the screen when you blow up things? I'll get all hell for saying this, but while this technique may be O.K. for an arcade-style game, it's really corny in a shooter. Though you want to keep the players' eyes on the screen, you couldn't find a better way to remind them that this is a computer game and they're not really out there shooting anybody.

How about using voice instead? An electronic digital reading assistant could fit well. Remember, the advantage of images over sound is persistence; there's no persistence here.

Pump Blood Faster.

After my tirade against distraction, here I am advocating it. (Well, I'm the first to admit there are no absolute truths in game design.)

Once you have designed your monitors, you can use them to increase the tension of a given moment. This is an art in itself, and those who are good at it know to use it sparingly.

I think back to how my father would make table tennis into a battle of the psyches. As soon as there was a little tension in the air, he would play on it, making his opponent aware of all the reasons to be nervous. Only those with steel backbones could withstand his tactics.

That's what you are accomplishing by flashing displays when ammo is low or when an enemy is approaching — as long as it's happening infrequently enough. You can make an only mildly dangerous situation appear catastrophic if you add the right touches. Similarly, you can pump up the thrill players experience when they're successful. Think of displays as something like the canned laughter and gasps in TV sitcoms.

As I said, there are no absolute truths in interface design. I can only discuss the issues and techniques involved. The final word is: Know the effect you are trying to achieve with your game, with its design, and with each and every one of those data displays you scatter around. The more you can articulate your goal, the better effects you'll achieve. ■

Tzvi Freeman teaches Game Design and Documentation at Digipen School of Computer Gaming in Vancouver, British Columbia, Canada. He has designed several commercial games and acted as a consultant on many others. He can be reached at Tzvi@compuserve.com or TzviF@aol.com.

De e mi i g Yo Game A dio Re i eme

George Santayana (1863-1952) said, "Those who cannot remember the past are condemned to repeat it." During the evolution of my career in computer gaming, I have tried to keep all of my notes so that I could remember the mistakes of the past and not repeat them. This article is drawn from those notes. If my past mistakes can be explained to you, then you might better understand my present methods of operation. While reading and contemplating this article, also remember another quotation from Mr. Santayana, which I will paraphrase: Skepticism should not be surrendered too soon nor to the first comer. There are many ways to approach audio in computer games. My methods work for me — they may not work for you.

In this article, I will review some important considerations regarding sound in computer games. We will start with general considerations and ultimately cover the more detailed resource and technical considerations regarding audio that should be determined early in the game development process.

Under my philosophy of game development, all members of a project team begin the project by getting into each team member's business. This is the appropriate time for project team members to share ideas, regardless of whose ultimate responsibility it will be to act on these ideas. Nothing is sacred at this point. No one is chastized for having an idea, no matter how ridiculous it may be. Everyone digs deep at this time and takes chances.

At the same time, team members take notes on how each idea would impact their areas of expertise. All ideas are taken at face value and are assumed to be attainable. The overriding question in each mind is, "How can I implement this idea?" This is the time when you (the audio team leader) should consider some hard facts and reality.

In truth, most of the projects that I have worked on have followed this system haphazardly, if at all. Because the project teams usually have been small, we could manage changing directions at a moment's notice. Larger project teams would have major problems if they worked this way. Team members must do everything possible to head off problems before they evolve. What hard facts and reality should the audio team leader consider in the early stages of a project?

Decide Who Decides

The foremost thing that I want to know is who is responsible for making the final decisions regarding sound. For goals to be met, resposibility for the audio has to move from the project team to the audio team leader and one other person maximum. Once the project is locked into a forward direction, only the sound person and his supervisor should be involved in decisions made about sound. This is not to say that the audio team should be isolated from other team members. At times, the audio people will have to

work closely and directly with artists, animators, designers, and programmers. Hopefully, the audio team members will remain open to ideas and suggestions from others. Still, these others should not be part of the final audio decision-making process.

I have been involved in projects where there was audio management by committee, and it simply doesn't work. Everyone has an opinion, and looking for consensus is nigh on impossible. Lock down from the start exactly who makes audio decisions. If you are a contractor, put it in a memorandum to the project supervisor, so you can point to the memo when you realize that the audio is being passed around for everyone to judge.

Hold onto Original Recordings

The release platform(s) for the project is the next thing to consider. id's CASTLE WOLFENSTEIN, which I worked on, was developed for the Intel platform, period. During development, id didn't consider the possibility that the game would be ported to the Mac or console platforms. The original audio was recorded directly to a sampler; the dialog was recorded using a high resolution and sampling rate that was later downsampled to 7k, 8-bit. Because the project team didn't have a network or large hard drives, they

IN THE EARLY STAGES OF GAME DEVELOPMENT, IT'S ESSENTIAL THAT YOU MAKE CERTAIN DECISIONS TO ENSURE THE HIGHEST QUALITY AUDIO FOR YOUR TITLE. by Bobby Prince

didnt keep a copy of the original digital files. Thankfully, I did keep a copy of most of the files on diskettes. These saved the day when the sound was ported to other platforms.

The moral of this story is that in planning sound, always assume that your project will become a Hollywood movie. Start with the highest quality sound available and work down to the release platform(s) for your game. Certainly, it takes longer for digital editors to massage 44.1k/16-bit/stereo files, but the extra time will pay off later.

Sound effects require many of the same considerations as the digital audio music. Taking a lesson from my Wolfenstein experience, all of the sounds that I created for DOOM were a minimum of 44.1k/16-bit in their original format. As in Wolfenstein, no one on the DOOM project team ever envisioned that the game would make appearances in movies and television programs. As for myself, I was only thinking of the possibility of ports to video games. During development of DOOM II, I was working with id Software on a port of Wolfenstein to a video game platform. Because of that work, I had decided to make sure that all of the DOOM sound effects were

archived at high resolution and sampling rate. This came in handy for the movie/TV situations and finally knocked into my head the conviction to start from the top and work down.

Regarding storage, a good removable hard drive is a cheap investment toward the future. Plan for future platforms (even nonelectronic ones).

Choosing the Sound Engine

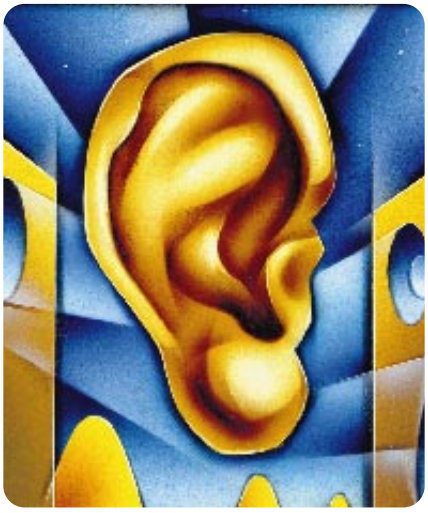
The next big thing to consider is the sound engine. Many of the hours I could have spent working on sound for a project have been devoted to helping the sound coder debug his new engine. Everyone wants to reinvent this wheel "the right way." The basic sound engine needs to be locked down very early in the project. Time is never on your side here. There are always tasks more pressing than some special feature for the sound engine. You have to be able to rely on certain basics. At the same time, you have to hedge your bet in case some of the special features do become available. When sound for the project is to be interactive, you have to be extra careful. Let's say that the project team hopes to have interactive music and has decided that MIDI is the music platform. But the sound engine is still in some coder's mind. It would probably be best to compose each song in modules that can be combined into one file when the interactive music engine turns out to be a dumb looping engine.

Music Platform Considerations

The platform for the music should be the next major consideration. The audio team leader should be careful here to ensure that the other audio decision maker fully understands the ramifications of each decision. The terminology surrounding audio is a big mystery to many project directors. As everyone else does, they want the best audio. But many times, project directors don't understand that there are tradeoffs no matter which direction is chosen. Much of the confusion can be averted if the audio team leader takes the time to explain the terminology and give examples of the different approaches. It takes patience to teach this stuff, but you have to take the time to do it or everyone will be disappointed by the results of early decisions made on erroneous assumptions.

If MIDI is to be supported, a decision must be made as to which synthesizers will be supported. If more than one type of synthesizer is supported, will there be a sequence file for each one, or will one GM file suffice? I have worked with sound engines that allow MIDI tracks to be designated for specific synthesizers. This method works especially well since it requires only one MIDI file per musical selection.

If Red Book audio or digital audio (.WAV, .AU, and so on) is to be used, important decisions must be made as to the sampling rate and resolution. If you've worked on many projects, I'm



sure that you've had a project director demand "CD audio," until you explained the 10MB/minute storage requirement. What you ultimately do depends on storage limitations, sound engine capabilities, desired audio clarity, and the like. Again, the best rule of thumb seems to be to start with a platform that would be appropriate for a movie or audio CD and work down from there. If MIDI files are used in the production of Red Book tracks, it is possible to make a basically acceptable GM version of the music. If live musicians are used, it would be prudent to have them use MIDI controllers so you can record a sequence of their performance. Then, if Red Book audio has to be dumped from the project, you would have the basics for your MIDI tracks.

The choice of sound tools is equally important. Until I worked on DUKE NUKEM 3D, I never knew the luxury of simply plugging a song or a sound effect into a game and listening to it (watching for sync) within context. Until then, I had to depend upon (and take up the time of) a programmer to "hard wire" these things.

While I was working on DOOM and DOOM II, John Romero usually did the hard wiring of the music and sound effects for me. I was set up in one room and he was in another. After I had what I thought was a workable sound effect or song, I would ask John to "make it so." He would make the new sound effect play, and I would watch and listen for the timing. Then I'd run back to my room and edit the sound effect to make it fit better. Then we'd do it all over again. If John wasn't there, I was stuck and had to wait to try out my new ideas. The benefit of this situation was that because John had excellent taste in sound effects and music (he placed all of the music in the final levels), I would frequently get new ideas from his comments. The downside to this situation was that I could have done even more music if so much time had not been required to get the sound effects synchronized to the action.

Proper sound tools can save a tremendous amount of time. In considering a project schedule, I need to know how much work is necessary just to hear possible music and sound effects in context.

The Sound Itself

Now that the more general considerations have been faced, it is time to decide what audio needs to be implemented for the project. How should the audio be planned?

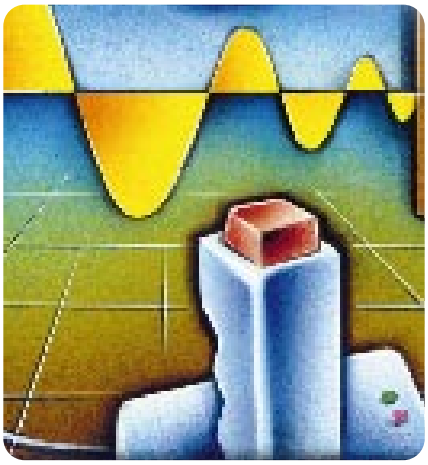
In my early projects, sound was planned in only the most general sense. The project team discussed the general direction for the music, but little time was spent planning for sound effects. Usually, I did sound effects that I thought were important, and other effects would be completed as someone saw the need for them. This sounds like (and probably was) laziness in planning, but it actually turned out to be a good approach; start from a minimal number of sound effects and add until believable sound (given the game environment) is achieved. Then stop and don't mess up a good thing.

Since these early projects, I have been involved with projects that had

some major effort on the part of the game developer in planning for the sound. Reviewing correspondence on some of these, I counted the proposed sound effects for one project in particular. There were 136 sounds required to start with, many that would be playing at the same time during the game. My ears ached just to think about it. Before the game was actually released, most of these sounds were dumped in favor of the more important ones (sometimes limited bandwidth and storage space can be a blessing). My projects that have won greatest acclaim for sound effects have had significantly fewer sound effects than those receiving little or no acclaim. I think it all has to do with focusing the game player's (or game reviewer's) attention in just one audio direction at a time.

Planning sound effects requires more than listing all the sounds one can think of at the time. Certainly, they can be planned ahead of time, but part of that planning should involve deciding what *not* to include. If the graphics have already been roughed out in storyboard fashion, an audio storyboard should be completed, too. As the art storyboard helps define the visual focal point of the scene, an audio storyboard can define the audio focal point. The audio storyboard can also be used to plan the music for the game.

Most of the projects I have worked on have started out with only the most general direction for music. Often the direction is more of a style request than anything else. This was the case with the DOOMs. From the start, the id team decided that the music should be heavy metal throughout. I sequenced several cover tunes of their favorite artists for trial runs. They were very excited about the results and wanted me to start writing original songs. When the game was at a point that the music could be put into context, everyone quickly saw that heavy metal music wasn't going to work on most of the levels. The same thing has happened in other projects, where the original direction of the music just didn't cut it in the context of the game. This leads to a rule of thumb for composers: Never showcase a composition out of context. Until a song can be played within the game, don't play it for audio decision makers. More than anything else, one song I wrote for DOOM taught me this lesson. I had this idea of using the GM instruments to say the word "doom" in a song. On wavetable synths, the oooh instrument starts with a "d," so I had the "doo" part. By messing around with the Santur patch, I was able to get a somewhat acceptable "mmm" sound; I had what I needed. I played the song out of context, and no one liked it. So much for my great idea! During DOOM II, I pulled the song out of mothballs and included it with several others that were being "wired in" to levels. The song was a great hit with everyone — no one remembered having heard it before and rejecting it.



Focus the Player's Attention

How does one determine what sounds, voices, and music will be necessary in a project? The answer to this is simple and easily overlooked. It can be stated in one word: Focus. Keep asking yourself where you want the game player's attention. If one single sound (music or sound effect) meets this goal, don't use anything else. If more is required, add it. Look at the movies that have won academy awards for sound/music. They are lessons in focus and simplicity of sound.

Before I start creating sound effects, I like to know a lot about the story behind the game. If there is no story, I make up one. I ask for biographical sketches of all of the game characters — including the nonhuman ones. It sounds pretty crazy to ask questions about whether a particular demon was hatched or born, whether it was raised by its parents or left to fend for itself, whether it had friends growing up, and the like, but it definitely paints an audio picture in my mind. The more that audio people know about a project's subject matter, the better they'll deliver appropriate content. Of course, this would be true for artists, too.

Who Is Involved?

One overarching consideration during the planning and execution of the audio for a game is the scheduling of the work. It pays to consider whether a demo is going to be produced well ahead of project completion. Planning for a demo can change the normal order that you might follow, and it is often an afterthought of even the most carefully planned projects.

Should audio development be outsourced? The answer to this question is another question: Is there enough audio work to keep an audio team busy? In the case of larger companies, an in-house audio team probably makes sense. Development teams that are centrally located are convenient, especially at the end of the project development cycle when last minute changes must be made quickly. At the same time, I have been involved with projects that had turned stale by the end of development, because everyone on the team had begun to think alike and no one was looking at things from a fresh perspective. While it is not always a requirement for a project to come together successfully, a good contractor will often travel to the developer's site and spend the last few weeks with them to tie the ribbons on a project.

The greatest thing that a producer can do is hire or contract with people who know what they are doing and trust them to do an excellent job. The most successful projects I have worked on were developed with this philosophy. It's always true that the whole in a creative endeavor is much greater than the sum of the parts. ■

Bobby Prince has had the pleasure and fun of contributing the music and sound effects to WOLFENSTEIN 3D, DUKE NUKE, DOOM, and many other notable games. Bobby can be reached via e-mail at gdmag@mfi.com.

Ho Pla fo m Choice Dic a e De ig

Game design is all about tradeoffs. Occasionally, design decisions can be made purely on the basis of providing for superior game play — which way is more fun? Far more frequently, an entire spectrum of other considerations comes into play. A good designer must consider tradeoffs in the areas of programming, art, sound effects, music, and writing. Often, factors more distantly related to game play are important, including the hardware on which the game will likely be played, marketing considerations, and even the retail distribution system. Finally, there are the practical tradeoffs, such as what resources (people, hardware, money, and expertise) are readily available and what "political" concerns must be managed (does the publisher hate games with full-motion video, does the producer have a bias towards games with rock music, and so on). A good game designer can juggle these competing concerns and still come up with a first-rate game.

Platform Pitfalls

One of the more interesting tradeoffs concerns deciding on a platform; is it a coin-op game, will it be launched on a console system such as Nintendo or PlayStation, or is it aimed at the PC market? Usually, a designer is handed this decision by the publisher or has already specialized in one of these platforms. Still, the constraints and opportunities implied by the platform choice can dictate many of the fundamental game design decisions during production. The range of game genres and styles varies widely from one platform type to another, often for subtle reasons. This article will touch on the unique qualities that the platform choice dictates to the designer, cover the pitfalls of converting games from one platform to another, and conclude by exploring the promise of future platforms.

A Night at the Coin-Opera

The coin-operated video game boom of the early 1980s introduced much of the world to the excitement of computer games. Close on its heels came the successes of the Atari VCS and the Mattel Intellivision, letting people bring the interactive entertainment experience to their living rooms. Lurking in the hobby market at that time were the first PC games, written for the TRS-80, the Apple II, and the Atari 800 home computers. Coin-op games are popular primarily with teenagers, consoles are favored by teenagers and young adults, and PC games tend to appeal to all ages, depending on the genre of game. Specific genres of games are popular on each platform. These three platform types have kept their differentiation to the present day, and all three formats are fairly healthy and potentially profitable.

But the variety of types of coin-op games has lessened. Fifteen years ago, there was a wide range of game types, with new technological and creative innovations surfacing frequently. Now,

fully 95% of all coin-op games are fighting games (like the MORTAL KOMBAT and VIRTUA FIGHTER lines), racing games, shooting games, or sports games. What's going on here?

Designers haven't run out of ideas. They've become trapped by market forces, shackled by a rigid distribution and sales system. Coin-op games have some advantages over the other platforms. They can have custom-designed hardware of great durability, including expensive extras like force-feedback joysticks, fancy gun controllers, or dedicated 3D accelerators that would be prohibitively expensive on a PC, much less a console system. But coin-op

The coin-op market is dominated by fighting games, such as VIRTUA FIGHTER.

THE HARSH MARKET REALITIES OF PC, CONSOLE, AND COIN-OP GAMES DETERMINE WHAT YOU CAN AND CAN'T GET AWAY WITH IN GAME DESIGN.

by Noah Falstein

games face a more rigorous testing process than the other formats. When a coin-op game is fully playable and fairly bug-free, it is put "on test" in an arcade. Unlike beta testing of a console or PC game — where players are brought into a room to try what is clearly a prototype game — those playing a coin-op game on test have no idea that it's any different than the other finished games around it. This makes for a very effective but harsh testing environment. If a game on test is not fully as popular as its competitors (measured in how much money it makes over the course of a week), it won't be released.

The most significant ramification of this process is that new and different coin-op games are culled during the testing process. Where a PC game might have a chance to be released, get reviewed, and find a market, a coin-op game has to live or die in direct competition with its peers.

But wait, it gets worse. The distribution system for coin-op games exacerbates the problem. Test results are reviewed not only by the game developer, but also by the distributors, who buy hundreds or thousands of units and resell them to the operators, who place them in arcades, bars, and restaurants around the world. The operators only want the top money-bringers, so a very hit-driven mindset edges out the more modest earners. The bad news for designers is that true innovation is discouraged. Because coin-op games don't come with manuals, it's vital that the players be able to understand how to play instantly. Furthermore, players must feel that they get their money's worth even with a short first game, and the game must have sufficient depth and surprises to keep them coming back over and over again, all without allowing more than a few minutes per play. A coin-op game that is extremely popular and averages ten-minutes game play will earn less than a game that is half as popular, but averages two minutes per play. These market forces dictate very narrow game design tradeoffs, which in turn encourage designers to simply take last year's successful game and make a slight variation for this year. Above all, the necessity of making lots of money quickly and motivating the player to come back for more necessitates an intense and exhilarating game experience, precluding the possibility of more slow-paced and lengthy games, such as a RPG or strategy game, making it in the coin-op market.

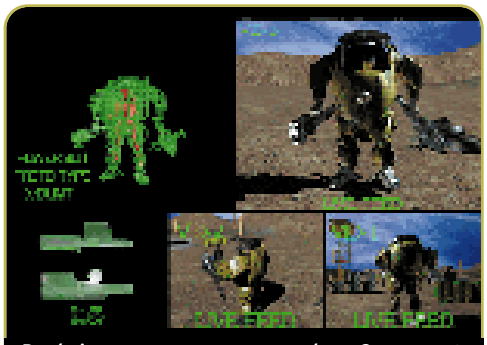
The Consolation Prize

The console market, currently dominated by Nintendo, Sega, and relative newcomer Sony, have an easier time of things — but not much. The console platforms have a few unique advantages. Unlike PC systems, a game for a given console will play the same way on every unit. The thorny issues such as minimum processor speed, memory, or CD-ROM speed that can plague the PC game designer are immaterial to a console designer, because the systems are standardized (with a few minor exceptions, such as optional controllers or peripherals). Unlike their coin-op cousins, console games have no time limitations, so quests can last for hours and sports games can take as long as their real-life counterparts without affecting profitability.

These advantages have costs. As high-powered as the latest 32- and 64-bit game consoles are, they tend to lag behind their PC and coin-op cousins in areas such as available RAM or mass storage options. And their standardization freezes them in time, so games for the Super Nintendo written today run just as quickly — or slowly — as when the system debuted years ago.

The creative range of titles for consoles is wider than for coin-op, but still somewhat limited — and for similar reasons. Console makers offer differing degrees of freedom to developers interested in making games for their systems; the

most restrictive of them have testing programs similar to those described for coin-op, and they exercise strict controls over distribution as well. In general, the longer a given console system is on the market and the bigger the installed base, the less restrictive the controls. Some very innovative design concepts have debuted on consoles, and Shigeru Miyamoto, Nintendo's secret weapon, does most of his groundbreaking work for their system. Few people realize that the now-popular PC genre of real-time strategy/action games such as WARCRAFT II and COMMAND AND CONQUER owe their origin to a modest game called HERZOG ZWEI on the Genesis in the late 1980s.



Real-time strategy games, such as COMMAND & CONQUER, are enjoying increasing popularity amongst PC gamers. This type of game evolved from a little-known Sega Genesis game called HERZOG ZWEI.

Home, Home Has the Range

PC games cover the widest range of genres and audience appeal. It's an interesting marketplace, in which the top two sales successes of the year can be a peaceful, cerebral, slow-paced exploration of a mystical island, and a slam-bang gore-fest where the player blows away demons with guns and grenades. This diversity comes from several sources. First and foremost, all console systems and coin-op systems are bought expressly as game platforms. In a tragic miscarriage of the natural order, most PCs are still used for such unimportant and trivial applications as word processing, spreadsheets, and general business management. Still, despite this backward thinking by the game-impaired public, many powerful systems with lots of RAM and huge hard disks are purchased as home office computers or for businesses and then used for gaming. In essence, the game play is subsidized by the commercial expenditures; a family that won't buy a $200 console system for their children may consider letting them use the $2,000 home computer to play the latest Disney extravaganza, and a 55-year-old executive who normally would find games too trivial to consider buying may be persuaded to try a "serious flight simulation program" or indulge in a golf program to check out some exotic courses. Because of this crossover, many different types of games are potential money earners. Designers have the widest range of creativity open to them when developing for the PC platform.

In a Galaxy Far, Far Away...

So how do these differences in markets and game genres translate into the decisions a designer makes? As mentioned before, often the choice of platform is given to the designer before the project starts. In some instances, the starting point is a concept or possibly a sports or movie license, and the platform decision is open. How does this affect game design?

Let's take the example of the Star Wars licenses. This popular license has seen several games made for just about every platform since the early 1980s. In each case, the designs are based on actions, situations, and characters from the movies. However, each owes as much to the conventions of the chosen platform as it does to the source material.

The coin-op Star Wars games from Atari were all fast-action shooter games, with intense first- or third-person combat situations. They looped through a series of action sequences and rapidly escalated in difficulty to ensure average play times in the two- to four-minute range.

Console games, such as SUPER STAR WARS for the SNES, also focused on the action component of the films, but added play depth and many levels, taking advantage of the longer game time players have at home. The game play was shifted to side-scrolling in deference to technology limitations and the popular conventions of contemporary console games.

The PC games have a wider range, from the first-person action of DARK FORCES to the simulation campaigns of X-WING and TIE FIGHTER to the strategic angle of the soon-to-be-released REBELLION. Here, a more complex range of actions is possible with both a keyboard and the time to learn a wide variety of controls. PC games also differ from coin-op and console games in their attention to storyline and plot development, common interests of the somewhat more sophisticated PC audience.

With these examples, we see how the target platform has a strong influence on the design decisions made when creating a hit game. More can be learned by studying the successful and unsuccessful attempts to convert a game from one platform to another. In general, designers have been more successful in taking a hit coin-op game and bringing it to the console market. The age range of the players is similar, the styles of games popular in coin-op are all also popular in console, and an exciting game made to the demanding restrictions of coin-op can still captivate a console player, once additional levels and depth have been added for longer play time. A few hit PC games have sold fairly well on console, although there have also been expensive failures, such as SIMCITY NINTENDO. For the most part, though, successful conversions in any other direction are rare. Very few console or PC games have ever been hits

LucasArts has managed to support a wide range of genres with its PC-based Star Wars games, such as the first-person shooter DARK FORCES, shown here, and REBELLION, a real-time strategy game.

when brought to coin-op, and coin-op and console games have enjoyed modest but generally limited success as PC games, possibly because most of the young action-oriented players already had played the games on console or in arcades.

The Empire Strikes Out?

Let's look at Microsoft's recent plan to put out Pentium-based arcade machines so developers can easily port existing PC games to the arcade market. One of the biggest obstacles to this plan is the 60-year-old coin-op hardware distribution system. Just getting the machines into arcades is going to be a tough job; impossible unless there is software that can bring a similar coin-drop (arcade-speak for the weekly dollars a given machine takes in) to existing dedicated machines. But Microsoft has lots of money and persistence, so let's assume that some brilliant game written expressly for their arcade box brings in about as much as the latest VIRTUA FIGHTER or MORTAL KOMBAT, and an installed base of machines is quickly built. What kind of follow-up games will be good to port? From the examples above, it's clear that most PC titles are just the wrong genre or play style for the arcade game market. Microprose almost went under in a disastrously expensive attempt to port their hit PC game F-15 STRIKE EAGLE to the coin-op market. But perhaps you have a great first-person 3D shooter game (there certainly are plenty on the shelves these days) and you take the time to adapt it to the arcade market, simplifying the controls so you don't need a full keyboard and structuring the level progression and relative difficulty to work for the arcade. Then you put it on test in an arcade, and it earns about one-fourth as well the current hit game written expressly for the coin-op market by professionals with decades of experience. Not a bad start, right? You'll probably sell about one-fourth as many units as the current hit game, but that's O.K., you can work your way up. Wrong.

The operator who buys the games from the distributor won't buy one-fourth as many units of your game as she will of the big hit. She'll buy several units of the big hit and perhaps a few others from the top-ten list of new releases. But if your game hasn't performed well enough to put it in the top ten or twenty, it won't sell any units, and you won't get that chance to release a merely satisfactory game in order to finance your learning curve on the way to making a spectacular game.

X-WING VS. TIE-FIGHTER, a simulation campaign, is another example of LucasArts ability to release PC-based titles over a wide range of genres.

This distribution bind is strongest in coin-op, but shelf space issues make it an increasing problem for console and PC games as well. You may be able to make some money by having a game in the top fifty instead of the top ten, but that's still a small percentage of the thousands of new games released every year. Software distribution is the straitjacket of the industry. It has become the number-one consideration in design tradeoffs for coin-op designers, and could well overwhelm home games in the future.

So, to paraphrase Obi-wan Kenobi and Yoda, was that distribution mechanism our only hope? No, there is another.

A New Hope

Many small game developers have turned to the Internet as their best chance of salvation. While the Internet isn't a new hardware platform *per se*, the ability to connect many players to games at once and the unique distribution possibilities put games designed for the online environment into a distinct category. The success of online distribution of shareware titles or teasers such as DOOM and DESCENT has given hope to a future where most software is distributed online; rather than browsing scores of titles on display in a software store, thousands or tens of thousands are a mouse-click away. Already on the Internet, we've seen the genesis of some of the first new genres of interactive entertainment in ten years. Online soap operas are struggling to make money, but may be a viable hybrid of the computer world and mainstream television. With the porting of games such as Berkeley Systems' YOU DON'T KNOW JACK to the Internet, can online game shows be far behind? But it isn't only new forms of entertainment that can thrive. If a developer wishes to spend $2,000 making a simple war-game scenario focusing on a single battle in the War of the Roses for an existing game engine, he may be able to find the 100 people in the world willing to pay him $20 to break even. More to the point, a development group spending $100,000 to make a low-budget but very original title can achieve breakthrough status through word-of-mouth on the Internet. These kinds of developers don't stand a chance in today's store-based distribution system.

Still, several important advances must come about before this rosy scenario is possible. High-speed downloading must become commonplace. The new 56K baud modems will help, but online game distribution will require the megabytes per second speed of cable modems or similar technology. Systems to prevent unauthorized duplication of games and to assist secure credit card transactions will have to become commonly accepted. And most importantly, the public will have to get used to browsing and shopping online. Is this going to happen? That topic deserves an article in and of itself. For now, let's all cross our fingers and hope. ■

Noah Falstein is a freelance interactive designer and producer. He's been making games since 1980 and is best known for his years at LucasArts, where he worked on their Indiana Jones graphic adventures and combat simulation games. Noah can be reached at nfalstein@aol.com

He e Come he CGDC

It started in developer Chris Crawford's living room and has evolved into a yearly pilgrimage for thousands of interested souls. Each year designers, programmers, artists, writers, producers, journalists, and analysts descend upon the Computer Game Developers Conference (CGDC) to get a wide-eyed perspective on where their industry is and where it's headed. If you haven't been to a CGDC before, you've been missing out on what many industry types consider the single best expenditure of time apart from developing products.

Now in its 11th year and with over 5,500 expected attendees, the CGDC promises to set the stage for a number of major industry developments. Contributing Editor Ben Sawyer posed some questions to five of this year's conference speakers in a series of individual interviews to gain some insight into where this diverse group sees the conference and the industry headed as April 25 approaches.

COMPUTER GAME DEVELOPERS CONFERENCE™

When:	April 25 – 29 (Expo: April 27 – 29)
Where:	Santa Clara Convention Center, Santa Clara, Calif.
What:	The CGDC is the conference and tradeshow focusing on the tools and technologies for creators of interactive entertainment software. Produced by Miller Freeman Inc., in association with the CGDA.
Estimated Attendance:	5,500 (Conference and Expo combined)
Cost:	The "Classic CGDC Pass," which gets you in to the three-day conference, plus full breakfast and lunch Sunday – Tuesday, is $1,095. A three-day Expo Pass is $75, available at the show.
Miscellany:	100+ exhibitors. Job fair running concurrently. 200 conference sessions.
To Register:	Call 617-821-9212, or go to the CGDC web site and use the online registration form at www.cgdc.com.

The Speakers

PHIL STEINMEYER, POP TOP SOFTWARE. Phil has programmed, codesigned, or designed three major strategy games for New World Computing, including the popular HEROES OF MIGHT & MAGIC series. His next effort is a yet-to-be-announced real-time strategy product.

BRYAN NEIDER, ELECTRONIC ARTS. The vice president of distribution for Electronic Arts, Neider has been in the entertainment software industry since the early 1980s. With a CPA background, Neider has worked for cartridge manufacturer Data East USA, Spectrum Holobyte, and Electronic Arts. Now he oversees EA's distribution business. This includes EA's affiliated label program, its Toys 'R' Us program and

emerging markets. EA distribution alone is a $90 million business.

ANNIE FOX, ELECTRONIC EGGPLANT ENTERTAINMENT. A designer of several top game titles, including an educational series of games based on the best selling Madeline books, Annie Fox is now producing a web community for LiveWorld, a company started by several former executives from Apple Computer. The site will help promote, organize, and support teenage volunteer projects and organizations around the country.

DENIS LOUBET, ILLUSION MACHINES. Denis's first work as a computer artist was the startup screen for the ULTIMA I, which he produced with an Apple II and a graphics tablet. Moving on, he eventually became Origin's first staff artist. Denis is now part owner and head artist at Illusion Machines, a company founded by Origin alumni.

LARRY GUTERMAN, DREAMWORKS/ PACIFIC DATA IMAGES. A graduate of USC film school, Guterman was the live-action director of DreamWorks Interactive's GOOSEBUMPS CD-ROM. He is now working at Dreamworks/Pacific Data Images (PDI) as codirector of *Ants*, an upcoming 3D animated film starring the voice of Woody Allen.

IN THE LAST YEAR OR TWO, A LOT HAS CHANGED. WE'RE GOING TO DEAL WITH ISSUES WHERE GAMES HAVE CHANGED AND EXPLORE WHETHER THINGS HAVE WORKED OUT AND WHAT WILL HAPPEN IN THE FUTURE.

The Questions

? First, please tell me about the presentation you'll be giving at the confernce.

PHIL STEINMEYER. I'm running a roundtable called "The Future of Strategy Games." Strategy games have changed a lot in the last couple of years. Three years ago, strategy games were either CIVILIZATION-type world builders or hex-based wargames. In the last year or two, a lot has changed. The big thing has been real-time and multiplayer games, such as COMMAND & CONQUER and WARCRAFT. Just look at graphics — strategy games used to have the worst graphics in the industry. Now, in many cases, they're among the best. So my roundtable will deal with issues where strategy games have changed and explore whether things have worked out and what will happen in the future.

BRYAN NEIDER. I'll be talking about distribution options for today's market. We'll focus on understanding the target audience for a product and the various publishing and business models out there. I'll talk about copublishing, affiliated label deals — which have changed dramatically over the years — and, lastly, what it's like to go out there and publish and distribute a product. Then I'll summarize with a discussion of today's retail market — all this from a game developer's perspective. Throughout this discussion, I'll talk about various financing options that accompany each agreement.

ANNIE FOX. We're doing a panel called "Women in Interactive Entertainment." I've put together a panel of women who are involved in work that covers different aspects of getting a product out the door. These include producers, directors of video and marketing, publishing executives, and me as a game designer. We're going to talk about our own personal experiences working in these different aspects of product development. Because we're all women, we'll discuss what that means in terms of our interest in certain products and the kind of products we'd like to design for female audiences.

DENIS LOUBET. This is my first conference, so I don't have any idea what to expect. I've been to Siggraph and CES, but not the CGDC. I'll be running a roundtable titled "Art Tools and Techniques." At this roundtable we'll be discussing how, in the 10 years since I started working with an Apple II and a paint program, we've advanced to using what was considered back then extremely high-end. Macs and PCs are pulling up to workstation-level performance, so we'll explore how high-end art tools affect the game development process. Pretty soon, all the packages will do all the cool things, and the feature sets of all the programs will be pretty interchangeable. At that point, it'll be strictly the artist's talent and vision that will get the best image across.

LARRY GUTERMAN. My session is titled "Cinematic Direction in Interactive Products." I'll be talking about the cinematic aspects of directing the live-action portions of

interactive games. I'll be talking about staging and shot design, visual-effects integration, working with actors and their performances, and editing for live-action film.

A good example of what I'll dive into is an especially important aspect of interactive products. There are subjective and objective points of view; how do you segue between subjective and objective points of view when you're trying to do something that looks cinematic? You often want to be able to bounce between the two, so when you segue into the fully interactive aspect of a product, you want to end up in a totally subjective setting so that the viewer has the experience of being involved. I'll be using lots of examples from past films and my work on the GOOSEBUMPS CD-ROM.

THERE ARE WOMEN WHO HAVE LOTS OF TALENT AND DON'T KNOW HOW APPLICABLE IT IS TO THE INTERACTIVE INDUSTRY. I HOPE THAT I CAN ADD THAT UNDERSTANDING TO THE CONFERENCE AND WOMEN ATTENDEES.

? What do you hope to add or gain by speaking at this year's conference?

PHIL STEINMEYER. [The conference] is a good forum for ideas. I've always liked the roundtables that let the small guys toss out ideas – a lot of times there are some really valid ideas. Something interesting happened last year during the AI roundtable I hosted. One of the things I wanted to know was, are other people out there using fancy neural networks and fuzzy logic in games? As much as everyone wants to talk about it, no one is actually doing it! One thing I hope to gain is an understanding of where the development curve is.

BRYAN NEIDER. I hope [developers] take away some tools that help them better prepare and manage their goals. I'd like to give them the tools to achieve their goals so they can honestly appraise the market and what it takes to be successful.

ANNIE FOX. I often find that there are women who have lots of talent and don't know how applicable it is to the interactive industry. I hope that I can add that understanding to the conference and women attendees.

DENIS LOUBET. I want to impart a sense of history. I know there will be some people at the [CGDC] that have been working in the computer game industry longer than I have, but I'll wager not many. I can bring a perspective of where it's been and where it's going.

I'd also like to gain some perspective myself from the conference. I haven't spoken very much to the big names in computer game graphics and it will be interesting to talk with other people in my position.

LARRY GUTERMAN. The GOOSEBUMPS CD-ROM was the first game that I worked on, so my experience with games is more limited — it isn't as fully explored as some others. I hope to convey an understanding of how to present things cinematically, so that [a game] sustains a viewer's attention both visually and in terms of a story. Dramatically, in between those points, where the player gets to interact, there really is a synergy of emotion, story, and gameplay. For me, I can't wait to see what others are doing.

? Why should someone attend your presentation?

PHIL STEINMEYER. Anybody who has worked, is working, or is planning to work on strategy games should come to find out what's going to be big next year — not what was big three years ago or even last year. It's a way to get a feel for the direction of the industry — and make sure you're on top of it and not behind the times.

BRYAN NEIDER. Developers who come to my session will come away with a sense of the level of competition in the market, and they'll find help in finding a niche they can fill.

ANNIE FOX. Well, I would suggest women come to my presentation — not exclusively, but women who are trying to break into the field or find a niche for themselves. It's still a very young industry. Because that's the case, it's easier to break into. We're not

Is there anything special you hope to get out of this year's conference for yourself?

PHIL STEINMEYER. Well there's going to be some more AI roundtables, so I'll be checking in on those. I also like to look at some of the business presentations, because I'm hiring employees. I'm also moving more into 3D now than in the past, so I'll be looking at the presentations of both software and hardware companies offering products. The whole conference is really a place to stick your big toe in a lot of different pools that you're not really up to date on.

BRYAN NEIDER. Well, the top topic is going to be the Internet. It'll either be Internet gaming or how to make money on the Internet. I think at a high level, a very interesting issue is going to be how products are going to be delivered, and who's going to be managing the back end. Is it going to be the TelCos or the cable companies or something else, like the power company? How will this distribution change the what developers do?

ANNIE FOX. Now that the Internet is opening up at the astonishing rate that it is, we've got a whole new frontier here for writers, designers, and producers. I think that's what these conferences ought to be about — letting new people in and inspiring people in this industry to widen their view of what is possible and what is entertainment.

DENIS LOUBET. Game-wise, of course it will be online gaming. We know it's possible, but what about profitability

THE WHOLE CONFERENCE IS REALLY A PLACE TO STICK YOUR BIG TOE IN A LOT OF DIFFERENT POOLS THAT YOU'RE NOT REALLY UP TO DATE ON.

talking about a Hollywood old-boy network yet. As a result, the way things are done are still up for grabs, and creative people can not only find a niche, but create new niches.

LARRY GUTERMAN. I would say specifically, from my understanding, the degree to which we try and integrate [cinematic] elements hasn't been attempted before. The specific tack [DreamWorks/PDI] took hasn't been attempted before. I've seen a number of other games where they do have live-action material, but often it's an interactive movie type thing, which isn't what GOOSEBUMPS is. We really blended a whole range of techniques, wide-lens shooting, puppeteering, miniatures, live-action, and 3D sets. People might want to come and see how we've worked these media together in a seamless integration.

BRYAN NEIDER. The CGDC is important. It gives you ideas of what to setup and have ready for E^3. The CGDC gives you a sneak peak into what you should be announcing and doing at E^3.

What is the big topic you or other people will be discussing at this year's conference?

PHIL STEINMEYER. I think you'll see the financial impact of the Internet more in the next year or two — and more at this conference than you have in the past. People have been talking about the Internet for a while, but it really didn't make a whole lot of sense up until now — I think that's going to start to chnage. 3D boards are also going to hit critical mass. But the Internet is really going to be key. There are now a lot of different models besides TEN and Mpath to discuss.

and all the various details pursuant to that? Do you want a huge megaworld that everyone goes into or a lot of tiny worlds that people pick from? On the art side, people will be talking about 3D Studio MAX and all the plug-ins that go with that.

LARRY GUTERMAN. To be honest, it's a tough question. From my perspective the idea of mixing, 3D on-the-fly graphics with cinematic direction from a master director — the issues of lighting and staging — are absolutely applicable and hold some new promise for synergy between live-action direction skills and game development.

What is the biggest problem you or other people will be discussing at this year's conference?

PHIL STEINMEYER. I don't know about others, but my view of the biggest problem in the industry is

marketing person to have a base level of understanding of what the technologies in a good product are. It doesn't mean they have to be an expert, but they have to be able to make a quick judgement about the hardware constraints, the tools being used, and the strength of a title. It's sort of knowing enough to be dangerous.

ANNIE FOX. Anyone who is just starting out, who is wondering how they might work what they do and love to do into this new industry, should come. Women in particular, I think, feel intimidated because of their lack of technical expertise. As a writer I always say to people, "I write in English." I'm not a programmer. I know just enough about the capabilities of the systems that I'm designing for to know what they can do.

THE THING THAT EVERYONE WILL BE DISCUSSING MOST WILL BE ONLINE GAMING, AND THE BIGGEST PROBLEM PEOPLE WILL BE DISCUSSING WILL BE ONLINE GAMING.

that there is too much "me-tooism" in the industry. I'm amazed because financially, it doesn't make a lot sense. Clearly, the big money makers have tried to achieve a little bit more originality.

BRYAN NEIDER. I think the issues we've dealt with traditionally, such as the channel problem with 4,000 titles. I don't think it's new news, but it's still a very significant issue. I think a bit more mundane problem will be trying to make great games with MMX technology and 3D boards. It's a very tangible subject and problem that people can get their hands on now and do something with.

DENIS LOUBET. The thing that everyone will be discussing most will be online gaming, and the biggest problem people will be discussing will be online gaming. There are only so many people out there that can connect to these games, and they're going to have to divide their time up between all these games that are coming out.

LARRY GUTERMAN. I think that in terms of problems, there is still the sense that the technology is driving the creativity sometimes. A lot of the conceptual aspects of games seem to not be as fully explored aesthetically and creatively as much as technologically.

? What advice do you have for new CGDC attendees?

PHIL STEINMEYER. Get to the roundtables and seminars early; as early as possible. Take in a lot of different things. Don't get caught up in the same track. Talk to people when you can; most speakers and people are very open. Be sure to go to the corporate sponsored night and grab all the cool stuff you want.

BRYAN NEIDER. I think it's very important for a business person or a

DENIS LOUBET. I haven't been to the CGDC yet, but at Siggraph, it's basically a place for all the companies to spy on each other. I've been told that the CGDC is a great place to catch up on what everyone else is doing.

? In terms of new announcements or new technologies, is there anything you'll be evaluating?

PHIL STEINMEYER. Microsoft will probably be talking more about DirectX, so I'll look for that. I'll be looking mostly at Direct3D to see if it's ready for prime time.

BRYAN NEIDER. I'll be trying to get a general read on what the next base level of gaming is. We have to be competitive at retail; how does that affect my affiliated label program and anything else I'll be evaluating between the CGDC and E^3?

DENIS LOUBET. I, of course, will be going around and looking at the art quality in any titles shown. I'll see

what plug-ins are being developed [for 3D programs.] A lot of vendors will be bringing their equipment and showing how it can help game production. So I'll see which tool makers really understand the game industry's needs.

LARRY GUTERMAN. Especially after seeing SUPER MARIO 64 and TOMB RAIDER, real-time 3D graphics technology will be of interest. I want to see if people are feeling like it's all going to jump another step. I'm also really curious about where the interface technologies for 3D are headed. These interfaces are a lot more complicated. I want to see what the limits are in terms of how many options you can offer the user.

? Where do you think the industry will be by next year's conference?

PHIL STEINMEYER. I think people will still be talking about the Internet. The hype will have fallen off a bit. It will become a little bit more old hat. But people will have a better grasp of where it's at. It will continue to grow — and the industry too — at a pretty fast clip. I don't think DVD will be that big of a factor next year. It might be, but I don't think anybody is serious about it right now.

BRYAN NEIDER. I think next year's conference will probably see the first full implementation of products with MMX technology. We'll have far more robust 3D environments than before. 3D will be the base level by next year. Not for every game, but by next year the question will be, what's built on top of the 3D? Is it streaming video or Internet hooks? The Internet, by next year, will be clearer. We'll have narrowed the list of issues, be able to put action plans in place, and be able to offer consumers a richer experience.

ANNIE FOX. The social aspects of online gaming will be important. You're looking at a whole new dynamic called the social interaction. It's what happens when people get together as a team. What's the most fun of the scavenger hunt? It's that you're running around the neighborhood with other kids. It's the fun you have pulling ideas, brainstorming, and problem solving with the other people that are a part of the game.

DENIS LOUBET. Unless there's another DOOM that wipes everybody out, I can't see the furor over online gaming dying anytime soon. This year, it seems like people are just starting to scramble and try and get something on there. Next year, we'll have found out what happens when you get on, and we'll see what fallout comes from that. I still think that online gaming will be the big problem and the big promise because we'll have data finally.

LARRY GUTERMAN. I think in terms of promise-pushing, the 3D side of things will still be there. I feel badly that the promise of live-action and digital video is still sort of [seen] in the old conventional wisdom. It would be nice if we could push that further, but these things tend to progress in waves. Perhaps by next year we'll be able to move [live-action video in games] forward more. ■

Different Perspectives on 3D Sprites

Side-scrolling games, such as the venerable MARIO BROTHERS, made sprite creation a relatively easy task for game artists. The intrepid plumber and his ilk trod bravely forth on a well-worn path from left to right. Not all that many frames of animatics were required to cover the necessary repertoire of movements: a few each for run, walk, jump, crouch, climb, and throw. On those occasions when the player reversed direction — backtracking from right to left — the right-face sprites could simply be mirrored to create a left-face version. Simplicity!

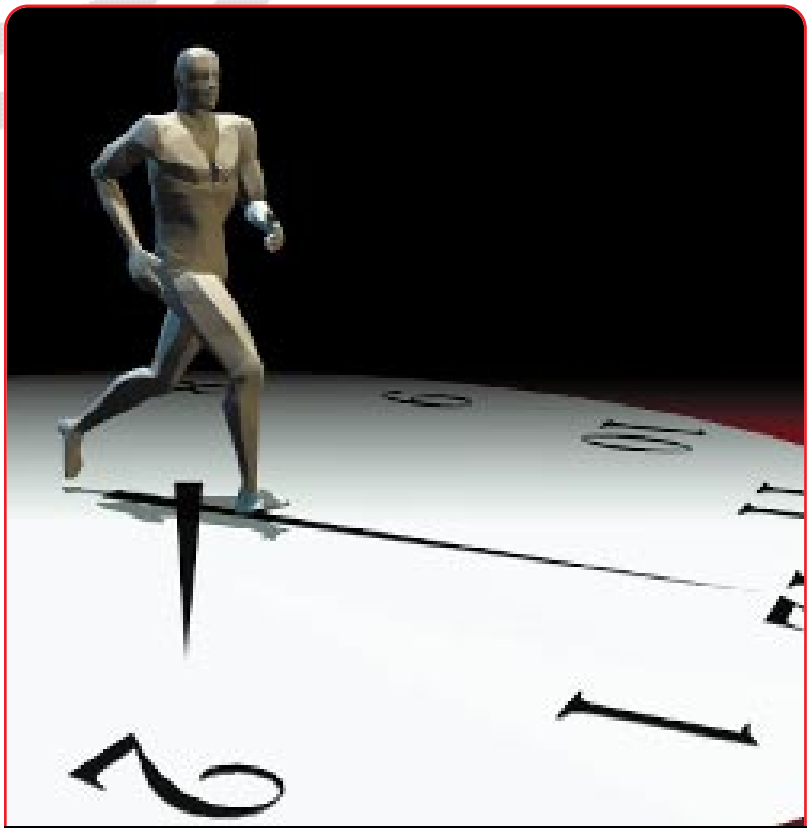
Even Mario has ventured into the Z dimension now, though, following much of the game industry in a stampede to offer players a 3D view of the action. First-person, isometric, cinematic... it's a rare game that hasn't moved to a more dimensional environment. More often than not, the game screen is now called upon to portray depth, and the on-screen heroes, villains, and hapless victims of these games must be well-rounded digital actors whose "good side" is every side.

The ideal-universe solution would call for a nicely detailed 3D model animated as necessary with in-game transforms: that is, real-time manipulation of vertices to accomplish movement. My programmer friends, however, tell me that this kind of environment requires too much processor power to be a practical solution for many game types and target systems. Some games can pull it off — for example, fighting games featuring a limited number of opponents in a small arena, or driving games where 3D models can more or less be pushed around the screen without complex rotation of vertices. Other solutions must be sought for more involved game scenarios where players explore larger 3D environments and encounter numerous characters. Two such workable solutions are kin to the humble 2D sprite. Somewhat confusingly, both are known as 3D sprites. And both present different sets of attractions for the game developer and challenges for the game artist.

It's A Bird, It's A Plane

Familiarized by id Software's paradigm-shifting hit, DOOM — which not only changed the way people looked at games, but also let them look behind the stage curtain and handle the props — the more common kind of 3D sprite is really 2D trickery. It's something I call a "planar sprite" because the 3D image is simply painted on a 2D surface that faces the player's viewpoint. As with 2D sprites, each

Two very different ways to "fake" 3D action, each with its own pros and cons, each commonly known as a "3D sprite".... It's enough to make an artist start coining his own terms.



A clock metaphor will help you visulaize the number of viewing angles from which you'll need to render your sprites.

possible action (walk, run, and so on) requires a brief animatic routine consisting of a short series of images. Unlike 2D sprites, however, these action snapshots must be shown from several view angles, so that the actor seems to inhabit its 3D environment realistically.

The number of angles required depends on the ways the actor will be viewed by the player during the game. A figure the player can walk around or sneak up behind should be shown from enough angles to afford an all-around view. Additional angles would be called for if that figure might also be seen from above or below. The challenge is to accomplish all this without exceeding memory limitations, which means the total number of sprites must be kept in check. As each frame of each action must be shown from every possible angle, the sprite-count can add up quickly. (For a more technical look at this sort of 3D sprite, see "Real-Time 3D Modeling" by Josh White, *Game Developer*, August/September '95.)

The first step in controlling the proliferation of sprites is to take advantage of symmetricality. Just as 2D sprites in a side-scrolling game can be mirrored so that the right face also serves as the left face, 3D sprites can be mirrored to avoid unnecessarily repeating visual information. As a simple example, imagine a figure standing on a clock face such that a view from the 12 o'clock position shows the figure head-on, a view from 3 o'clock shows the right side, 6 o'clock shows the backside, and so on. Given a reasonably symmetrical figure whose right side looks more or less like his left, the views from the 7 o'clock through 11 o'clock positions could be dispensed with, as mirror copies of the 1 o'clock through 5 o'clock views could be used in their place. By this means, the number of view angles needed can be cut nearly in half. Little details — such as a gun held in the right hand suddenly appearing to be in the left hand — generally can be overlooked in the interest of economy. The hope is that the player will be too involved with gameplay to notice or care.

If the first approach to economizing is at the expense of accuracy (which hand was that gun in?), other sprite-cutting tactics cost in smoothness. *Rotational* smoothness depends on the number of degrees between each view; in terms of the clock-face example, how many increments are between 12 and 6 on the dial? The more finely subdivided that space, the more smoothly a rotation of the actor can be depicted. Of course, for each position, a full series of sprites is needed for all actions; the fewer increments you can live with, the better off your sprite budget.

Animatic smoothness is sacrificed by reducing the number of frames in any given movement routine; cutting your walk cycle down from 12 frames to 6, for example. Remember, each frame saved is multiplied by the total number of view angles: the number of rotational increments discussed previously. Savings add up quickly as frames are cut, but movement becomes choppier and less satisfactory. It's another trade-off to be carefully balanced.

Planar sprites can be created any number of ways — using hand drawings or photographs, for example — but one common approach is to render out a series of view angles of a model from a 3D software package. This technique was used for MICROSOFT NBA FULL COURT PRESS, a 3D basketball game for Windows 95 that makes use of planar sprites. A look at their experience helps illustrate some of the other opportunities and challenges presented by this technique.

MICROSOFT NBA FULL COURT PRESS makes use of planar sprites mapped onto a 3D model of a basketbal player. The jersey numbers were also achieved through some clever graphics trickery.

Artists at Microsoft started with a 3D model of a basketball player — purchased from Viewpoint Datalabs — and imported it into Softimage 3D. Weighing in at over 20,000 polygons, this model would have been inappropriate for use as an in-game 3D model. For the purpose of rendering out sprites, however, the detail was welcome. One advantage of this sort of sprite is that it allows the representation of greater detail than would be possible with actual in-game geometry.

The Microsoft artists attached a "skeleton" to the model in Softimage so that skeletal deformation could be used to control animation. A certain amount of editing was called for to align the model's vertices so that they would move properly with the underlying skeleton. The decision was made to base animation on motion-capture data, as the project called for a whopping 250 realistic movement routines. BioVision of San Francisco, Calif., provided motion-capture services, allowing a live actor's movements to be recorded. The difficult part was paring down the smooth, high-resolution movement thus acquired to a practical number of frames for each movement routine. The "sprite budget" dictated that fewer than one frame in four could be used.

The "channel" feature in Softimage made it easy to animate the model using this motion-capture data. About four-fifths of the data translated readily to the 3D model; some routines

matched up less well, and the basketball player would wind up with one leg over his head or his torso turned 180 degrees. Most of the editing that was called for, however, was to exaggerate the realistic movements so that they would "read," given the small size of sprites on the game screen. Eventually, seven artists spent five months hand-editing every frame of animation to get the desired movement quality.

A very creative approach was used in depicting jersey numbers on the basketball uniform, so that a single sprite could be used to represent different players with different jersey numbers. Segmented numbers, such as those on a digital clock, were incorporated into the texture map used to detail the uniform on the model. Each segment of the number was painted with a unique RGB value not used elsewhere in the game's palette. If uniforms were to be white with red numbers, for example, each uniquely colored segment would render *red* if needed for a particular number. If not needed, it would render as *white* so as to appear the same color as the uniform. Very clever!

The development team, however, experienced an unexpected stumbling block. Since lights were used in Softimage to shade the model and define its form, the number segments got shaded along with the rest of the texture map. This meant that they were no longer the pure, unique RGB value expected. Instead of the appropriate uniform number, each player sported a crazy, multicolored 88 on his jersey! The painstaking solution was to "sew" individual polygons onto the model's uniform over each segment of the number. "Constant" shading was then used to color each of these polygons, so that when rendered, they retained their unique RGB value.

Talk about a high body count... readily available editing utilites for id's industry-transforming DOOM let every aspiring game developer see for themselves just how quickly the sprite count adds up.

Geo Whiz

Though jersey numbers caused the artists at Microsoft a few headaches, a lot of the complication and difficulty of working with planar sprites comes in managing the quantity and position of view angles and the multitudinous bitmaps that result. Another sort of 3D sprite neatly sidesteps these issues while imposing a few rigors of its own. I call these "geosprites" because they are geometry-based rather than bitmap-based.

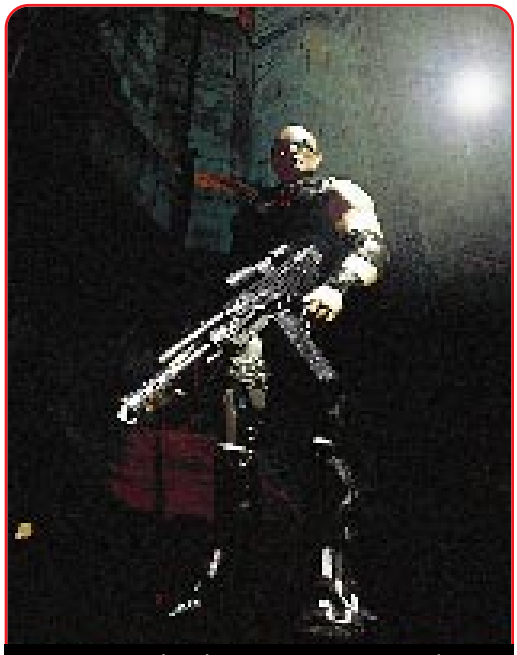
Geosprites are actual in-game geometry: 3D objects in the 3D environment. As such, view angles don't need to be calculated and prerendered, as with planar sprites. Whatever the player's viewpoint, the sprite appears in the appropriate orientation.

As with other sorts of sprites, geosprites are animatic; though the geometry is brought in-game, it isn't animated using real-time vertex transforms. Instead, during development, artists use the 3D software package to make a series of 3D "snapshots" by freezing and outputting the model at several points throughout a movement routine. These static models correspond to the frame sequences used with other sprites. By rapidly swapping one model for the next in the series, the illusion of movement is created on-screen.

Compare the artist's task in creating planar sprites versus geosprites. Say our task is to create a movement routine consisting of five frames. To make planar sprites, we would first need to determine the exact position of the camera for each desired view angle. That's a job in itself. We must then render our five frames from every view angle — say, thirty camera positions — and manage the resulting 150 bitmaps so that the appropriate sprite can be referenced in the game when needed. With geosprites, we output a 3D snapshot of our model at those five frames. That's it. Five 3D models versus 150 bitmaps. When you start thinking about the number of different characters and the various movement routines needed for each, the file management duties involved in using planar sprites becomes a monstrous task.

Put in these terms, geosprites sound like the easy way out. However, the same memory limitations that necessitate a "sprite budget" when using planar sprites also demand "polygon and texture-map budgets" for geosprites. The challenge, therefore, is to build extremely economical models and textures that retain enough detail to please the eye.

Psygnosis used this sort of 3D sprite in the creation of TENKA, its fully 3D first-person point-of-view shooter for



In TENKA's rendered sequences, 100,000 polygons are used to create the hero. Getting the polygon count down is an art in itself.

the PlayStation game platform. Given the amount of action the game called for on-screen, the PlayStation's processor couldn't be expected to handle real-time animation and maintain a desirable framerate. The use of geosprites enabled Psygnosis to create TENKA as the true 3D environment they envisioned without ever letting framerate drop below 30 fps.

The bad guys in TENKA's sci-fi world are half-human half-robot "bionoids," mutant monstrosities, and small droids. The models range from 300 down to 50 polygons. Compare that to the over 20,000 polygons for the basketball player mentioned above or the 100,000 polygon version of TENKA's hero used by Psygnosis for the rendered intro and final sequences, and you begin to appreciate the difficulties of building such "simple" models. Though Psygnosis also used Softimage — which has excellent polygon-reduction tools — during development, most of these models were built by hand rather than optimized, simply because the target polygon count was too low to automate the process successfully.

The Softimage tool that facilitated the use of geosprites on the TENKA project is a stand-alone feature called evalScene. This grabs a 3D snapshot of a Softimage scene at a specified frame. Not only did Psygnosis use evalScene to create geosprites, they got game environments this way as well. Scenes were created, mapped, and lit in Softimage, an evalScene snapshot taken of the environment, then a custom-built converter used to bring the data into the PlayStation. This way, models and environments could be created by artists in an artist-friendly application: the 3D software with which they were already familiar.

These two approaches to the 3D sprite present some identical challenges, but each also demands of the artist some different creative solutions and artistic skills. Whether your next project uses what I've chosen to call planar sprites or so-called geosprites, either form of 3D sprite is a handy way to milk more multidimensional action out of today's limited system resources. Just make sure that when your boss calls upon you to create some "3D sprites" for the new project, you know which kind is meant. ■

David Sieks is a Contributing Editor at Game Developer. You can contact him via e-mail at gdmag@mfi.com.

FOR FURTHER INFO

BioVision
415 292-0333
info@biovision.com
http://www.biovision.com

Softimage
514 845-1636
Fax: 514 845-5676
http://www.softimage.com/Softimage/default.htm

ViewPoint DataLabs
800 328-2738
801 229-3000
Fax: 801 229-3300
http://www.viewpoint.com/

by Ben Sawyer

Skimming the Voxel Surface with NovaLogic's COMMANCHE 3

When it comes to combining cutting-edge technology with games, NovaLogic is one of the first game developers to leap to mind. Like other companies — such as id — that have also hit the fine line between cutting-edge program technology and game development, NovaLogic has organized itself and its product line to accentuate its succesful marriage of technology.

That's why *Game Developer* interviewed several NovaLogic employees. With new technologies such as Direct3D, 3D sound, and MMX coming to the market in a big way in 1997, a strong technology perspective seems especially interesting as the bar is raised on performance by a new generation of consumer machines.

From a Technology Standpoint

Game development at NovaLogic follows a single thread — strong base technology. For several years now, that base has been Voxel Space, a real-time implementation the volume pixel technology used by high-end graphics programmers — especially those doing medical imaging.

Volume pixels even found their way into an article in *Wired* magazine recently, subtitled "Getting below the surface." Still, when it comes to games, who cares what's below the surface?

"As far as the *Wired* article — which I only scanned when I didn't see our name in it — there are voxels and there are voxels," says NovaLogic CEO John Garcia. NovaLogic's voxels aren't the same as the voxels used in processes like medical imaging, which concerns itself with the entire volume of a volume pixel. NovaLogic programmer Kyle Freeman earned a patent for a subset of traditional voxel technology that relates to the surface. What NovaLogic has done by concentrating on the surfaces of voxel imaging is to figure out how to render surfaces with incredible speed.

"We are so technology driven that what makes this process fun for us is figuring out how to make things cooler and faster than what's already been done — not only by our competitors, but by ourselves as well," says lead programmer Freeman. "When you do a game, you want to pick something that will be different than the rest."

Voxels are different because they have a lot of 3D detail and they render very fast (within Voxel Space.) there can be a lot more of them. As Freeman explains, rather than representing a kilometer of ground with one large triangle, it can be represented by thousands of little blocks, which can represent the actual dirt and rocks. This approach has some unique advantages, according to Freeman. "As something such as a tank rolls over those blocks, it can actually be influenced by the surface texture of the ground."

"We are so technology driven that what makes this process fun for us is figuring out how to make things cooler and faster than what's already been done — not only by our competitors, but by ourselves as well."

Voxels, though, do have drawbacks. "A tank with a moving turret is costly to translate. You don't want to use voxels for mobile objects," says Freeman. In these cases, NovaLogic has been working to perfect its games by adding in polygon technology to its Voxel Space techniques.

These methods require speed; at NovaLogic, that speed is gained through the use of assembly language. "All of our games and tools are written in 32-bit assembly code," explains Freeman. "Using assembly, we can actually access 64-bit operations at the processor level that you can't access using other higher-level languages. We're slipping in MMX instructions where they're appropriate. Our games seem to run about 10% faster on an MMX CPU, and we haven't gone nearly as far as we'd like to concerning MMX optimization."

Charging Forward With MMX.

Intel's MMX, being a processor-level technology, offers a company as schooled in assembly as NovaLogic some unique advantages. So what does Freeman think of MMX?

"I love the MMX instructions. I wish they had been in there from day one. As the MMX installed base grows, I'm sure we'll be using MMX instructions as much as we use normal instructions today."

Continuing, Freeman explained the basic reason MMX is so interesting to an industry faced with loads of 3D graphics cards.

"MMX is nice because it's in the CPU, so it has an inherent advantage. It's also general-purpose, which is very important. 3D cards tend to accelerate certain types of graphics. I have yet to see a 3D card that accelerates voxels.

"We can make the MMX do anything we want, so we can make it accelerate our polygons *and* our voxels. Plus, the MMX is in the CPU and has full-bandwidth access to all the memory in the computer, whereas a 3D accelerator card would typically only have a small amount of texture memory on the card — which currently would limit the look of the game."

So is a company like NovaLogic looking at card technology at all? "We're looking at graphic cards, but in the very short term," says Freeman. "We won't have any game that takes advantage of them — the cards are changing so rapidly. We have problems just keeping track of them. I have no clue what the consumers are going to end up doing."

Not Just Graphics

As the creator of Voxel Space, Freeman uses voxels to sum up his entire philosophy of technology-based development. "I like technology, especially if it's different or unique; voxels certainly offer an advantage and I'm certainly one to monopolize on them. Stick 'em where the do the most good!" However, before you think Freeman only means graphics when he says technology, consider his work with 3D sound technology.

According to Freeman, "We're the only company to be shipping a project with real-time Dolby surround sound." Explaining the virtues of 3D sound, Freeman relates that, "Dolby ProLogic provides the advantage of sounding great on two speakers no matter where you position yourself. It also sounds good if you drop down into mono, whereas a lot of the other 3D sound systems don't. A lot of the speaker systems coming out now are incorporating ProLogic, and the truth is that if you run one of the other 3D sound systems through ProLogic, it sounds horrible. ProLogic is the most generally compatible system to generate 3D sound."

Freeman developed the drivers on his own, and NovaLogic contacted Dolby to find out if the company was interested in what Freeman had accomplished. At first Dolby was skeptical about the idea. Then they heard it.

"They weren't entirely convinced you could do Dolby Pro-Logic in real-time," Garcia said. "But Kyle showed them." Much as he had shown the world real-time voxels.

From Volume Pixels to Volume Sales

It's one thing to develop cool technology and another to be able to jump right into cutting-edge technology such as MMX. At NovaLogic, they've used those items as catalysts to their development of top-selling games such as COMMANCHE and ARMORED FIST.

"We're very technology-oriented," says Garcia. "We try to not only perfect the technology that's out there, but to create new technology — from that comes the games. We've gravitated toward vehicle simulators because [those games] like that technology."

This philosophy is best represented by the original version of COMMANCHE. As explained by product manager John Seeholzer, "The game ended up as a helicopter simulation because Voxel Space was created. The original version of Voxel Space dictated the type of game. It was a technology that didn't

What is Voxel Space?

Patented technologies need to be unique. While voxels are not, Voxel Space is. The simplest way to think about voxels is to imagine you are extruding a pixel on a 2D image. Each time you extrude the pixels another step, you're building another layer of 3D data; extrude them a lot and each to a different height and you can create some extraordinary 3D images. Voxels (volume pixels) have been used in medical imaging and other 3D visualization fields for some time. For users in those fields, however, their affinity for voxels is based on the graphic technology's ability to provide top-notch 3D renderings of objects — especially successive layers, which might help locate tumors or other data deep inside an object.

Game developers want top-notch 3D graphics, as well. But they have the added overhead of trying to produce them in real-time on far slower hardware. Game developers aren't concerned with different layers because the player isn't going to see them. Drawing only what needs to be seen is where Voxel Space — and NovaLogic's patent — comes into play.

NovaLogic and Kyle Freeman developed a process that allows them to calculate and draw only the surface portion of a volume pixel. By disregarding the interior of the voxel (which is the medically important part), they're able to bring the beauty and precision of voxels up to the speed of real-time graphics. Now working on the third generation of the technology, NovaLogic has continued to push the ability of Voxel Space, extending its speed, viewing area, and resolution, as well as adding polygons and other 3D techniques to render tanks, planes, and other moving objects where voxels don't work well. With Voxel Space, NovaLogic has taken the volume out of the pixels and put it into its sales.

look good from really high up, because you see very far. Its strength was in modeling things close to the ground, but that also didn't look great from right on the ground. So NovaLogic was well into the development of COMMANCHE before we realized that it was going to be COMMANCHE."

Seeholzer explained the difference between the way NovaLogic develops games and the way many other companies go about creating a product. "I think that we are a somewhat unorthodox company in the way that we go about game development. Over the years, I think our industry has left the time when every company had their own approach. As the industry has gotten bigger and the companies have gotten bigger, there are more companies doing the same things. So, as that has happened, we are more and more defined as an unusual company in the way we do things."

Product development being the way it is at NovaLogic, the process revolves around the lead programmer. "We approach development from a fairly flat, matrix-oriented perspective," says Seeholzer. "The lead programmer on a project — because of the fact that our products are so technology-driven and oriented — is the person to whom all

of the links are attached." Everything revolves around that programmer. "We form a ring around the programmer with our art staff, assistant programmers, and designers," continues Seeholzer. "What we don't have is a top-down approach, where you have a producer who creates a large design spec and then hands out large stacks to minions who then do the work."

This process is meant to let the technology that NovaLogic creates shine through and drive the development process. "We have a very collaborative environment," says Seeholzer. "Nobody is really the sole dictator in our projects, though there are some people who have veto power to settle disputes and set direction. The general approach is to have a programmer who is juggling a lot things and knows basically what the code is going to do. We want to do as much cool stuff as we can with what we can program. It's the code that dictates what sort of things the program can do. We'd hate to be in a situation where the programmer could do something amazingly cool, but it was not written up in some design spec, so it didn't go in. Nor would we want to force the game to do something arbitrarily that its code wasn't good at."

So without that large spec and top-down approach, what general scheduling rules drive the development? Seeholzer explains, "Certainly, as is true for anyone producing games these days, pushing the process is a challenging and delicate business. It can vary a little bit depending on the level of new technology we need to create from scratch.

"For example, COMMANCHE 3 is an entirely new engine, so a lot of the effort that went into programming went into the technology. However, ARMORED FIST 2 is benefiting a lot from that new engine by needing less time spent on the technology; more of the effort is spent trying to make a product out of it. In general terms, we like a product never to take more than 18 months. If you spend too long, by the time it comes out, there is no longer a market for it. Our common range is about 12 months, given that a certain amount of R&D has already been worked on."

So What's Being Developed?

"Our next PC product is COMMANCHE 3, which is a gigantic leap forward for us technology-wise," says Seeholzer. "ARMORED FIST is expected to master within a month and a half of COMMANCHE." As for some other projects, "We've wanted to do a fixed-wing project for some time. Once we got experienced with polygons for Voxel Space 2, we realized we had enough experience with polygons to do a fixed wing project, which became F-22."

NovaLogic's technology base and its previous products' strong sales have made the company a strong producer of sequel.

Says Garcia, "The desire to try again at something we were successful with, but we knew we could do even better, now leads us to doing sequels. We were all very proud of COMMANCHE. As time has gone on, we've no only learned massively more about what the real Commanche is, but also about flight models and other military simulation aspects. This has led us to wanting to do a much better COMMANCHE."

COMMANCHE 3 is that better COMMANCHE. Freeman led the development of the new version and brought a desire to push the game in lots of new directions. "COMMANCHE 3 started right after the original COMMANCHE shipped,"he recalls. "Limitations that were identified in the older version were methodically elimi-

ADVERTISER INDEX

nated by the new Voxel Space engine and the new game. Hindsight was in ample supply after the original COMMANCHE."

The new version works well with far-away object visibility, as well as close-up views — the main problem of the earlier technology. It also has a very complex physical model. One of the things Freeman wanted to do after the first COMMANCHE was to make every object in the universe behave consistent in its own universe and by the laws of physics.

The game's physics model was embellished so much that Freeman himself has been surprised by the game's actions. "I have tremendous fun playing the game, because it is extremely hard to predict. There are no canned sequences anywhere; it's all just following physics in a live universe."

Applying Design to Technology

While a company like NovaLogic lets its technology thrust propel the project, it still applies a great deal of additional design work to each product. The technology primarily builds the visual, aural, and physical aspect of the game. Because NovaLogic has put so much into the design, the game itself has a lot to live up to. That means the design has to be top-notch.

Over the years, design at NovaLogic has caught up to its passion for pushing the game engine. NovaLogic has steadily built up a lot of internal expertise about military simulations. The company applies and reapplies that expertise it as often as possible.

"We may be doing a tanking game and learn a lot about how American and Russian tanks move around," explains Seeholzer. "In another game — such as a helicopter simulation — we might want to have some tanks moving around on the ground. We can reapply that knowledge base across the board.

"We have a whole company of people on the lookout for stuff in books, TV, and other popular media. All of our employees in development are really keyed in to this process. It's not uncommon to come into work at NovaLogic and have an artist run in with a tape from The Discovery Channel with the first ever recording of a certain tank firing. The entire staff will analyze it over and over for art perspectives, design ideas, and so forth."

Adds Garcia, "We've been doing this for awhile, and a lot of doors are beginning to open for us. In the case of ARMORED FIST, the Marines were very helpful. We spent a day over at 29 Palms playing with the tanks. With COMMANCHE, both Kyle Freeman and I were invited by Sikorsky to come out to Connecticut and see how the helicopter was developed and play with their simulator."

The company also looks to military consultants. Explains Seeholzer, "When we did the original COMMANCHE, we found the editor of a military defense helicopter magazine. We used him as a consultant. In more recent days, we've interviewed pilots of Commanches and Russian Werewolfs."

"We also go the manufacturers to seek unclassified information," Seeholzer continues. "They're most definitely aware of the games, and in recent years, everyone has become much more cooperative. We get some of the best help from people actually working with the vehicles we're modeling."

Customers are also a potent source of feedback; that type of feedback is more important in terms gameplay than in the details of the flight model. "We're always interested in making our products more successful while satisfying the demand to make them authentic," says Garcia. "If somebody out there says our flight model is too hard to deal with, we might introduce an alternate flight model to make it easier."

Building a Strategic Advantage.

All companies — especially those in an industry as competitive as computer games — must find a strategic advantage. For NovaLogic, that advantage truly comes from its commitment to finding a way to evolve product design from a technology-first standpoint. The way it has modeled its company, and subsequently its design process, around technology has resulted in success any company would want to emulate.

However, to go to the extent NovaLogic has gone to build a strategic advantage through technology — including a patent on Voxel Space — is not a route that many companies have traveled. As NovaLogic demonstrates, that route requires a commitment to wait for the technology to offer up the product, and then make the most of that sudden opportunity.

In an industry that continues to push itself to meet market ship dates with 16-hour workdays, perhaps the biggest strategic advantage NovaLogic has found is that it has the experience of knowing what lies at the end of a "wait-for-the-technology" based approach. We'll see examples of this commitment when COMMANCHE 3 and ARMORED FIST 2 ship this spring. ■

Based in Portland, Maine, Ben Sawyer writes and consults about the interactive and consumer technology industries. His latest book, The Digital Camera Companion *(Coriolis Group Books) is out now. He can be reached at BenSawyer@worldnet.att.net.*

NovaLogic Profile

NovaLogic has 70 employees. The company concentrates on several core titles a year and usually releases one title from its previous PC stable for the Mac each year. All art, design, mission creation, tech support, and marketing are in done in-house.

NovaLogic is an affiliated label of Electronic Arts, which frees it from the responsibility of managing accounts or inventory. The company enhances this arrangement by having its own sales force, which makes calls on key accounts.

COMMANCHE 3 development was done almost entirely in assembly on PC systems, targeting DOS and Windows 95 platforms. The lead programmer coded 95% of the title, while other pieces of code (such as network code or a map mechanism) were contributed by other programmers working at the company.

A title like COMMANCHE 3 had 12 artists working around the clock. Most art went through four or five generations before it was placed in the game.

SOAPBOX

by Sid Meier

A Revolution

The computer gaming industry is undergoing a revolution! This mantra has been used to promote dozens of technical innovations that have promised to radically alter the experience of playing a computer game. From virtual reality headware to interactive movies, CD-ROM, and DVD, our industry has spawned almost as many buzzwords as products over the past few years. Along with the new technologies came a headlong rush to expand development teams, gobble up Hollywood talent, and sink millions of dollars into the latest bells and the loudest whistles. The exponential growth in computing power and storage space has attracted media attention and spurred growth in the market, but the fundamental qualities that make a good game have remained unchanged and elusive. Consumers still flock to buy original, addictive, and fun games, leaving many flashy products with million-dollar budgets languishing in the $9.99 bin. These costly failures demonstrate that the consumer does not desire a cinematic experience, but rather a quality gaming experience.

The real changes are occurring in the computer gaming industry — not in the nature of the games themselves. Increasingly, the industry is splitting into two groups: the large publishers, who handle sales, marketing, and distribution, and the small producers who provide the publisher's content. Consider the top five products of 1996: WARCRAFT II, MYST, DUKE NUKEM 3D, CIVILIZATION II, and COMMAND & CONQUER. With the exception of CIVILIZATION II, all were produced by small, primarily independent developers. The prime example of this new trend is id Software, which showed that eight or ten people working apart from the large, established companies could turn the industry on its head with wonderfully designed games. No longer is it considered necessary or even desirable to have a cast of thousands working on a single product.

This system of small developers working with large publishers is evolving because it makes sense from both a creative and a business perspective. Many characteristics of a small, motivated group facilitate the production of high-quality games. Development of a game is enhanced by a close-knit, intimate environment where the group shares a vision and has the flexibility and control to fully implement that vision. Members can communicate easily, promoting a free flow of ideas, and the team is able to respond quickly to market trends — minimizing missteps and shortening development cycles. The marketing, sales, suffocating bureaucracy, and high overhead costs are left to the large publishers, allowing the developers a tight focus on playability and fun.

However, the greatest advantage of the small producer is in its insulation from the quarterly fiscal pressures faced by larger publishers. Such separation allows for the organic evolution of a game, where ideas are tried and implemented based on how well they enhance the gameplay. Prototyping is the most important ingredient in a successful development cycle. But prototyping is difficult to schedule and it cannot be shortened to make the numbers come out properly in a spreadsheet cell. We have all seen projects released before they were ready, not because the designers thought they were complete or the playtesters could not find any more bugs, but because the company had a responsibility to its shareholders that conflicted with the interests of its customers. By breaking the link — separating the process of game creation from the business of selling a product — the whole industry moves forward.

The large publishers have an inherent advantage in that they are insulated from the "make-or-break" syndrome experienced by smaller companies. For a company like Microsoft, no single missed game would ruin revenue projections or send it's stock into a tumble. Consequently, the company can allow its external developers more time for the all-important prototyping and play balancing that truly differentiate a quality product from shallow hype.

For producers of computer games, the divergence of the industry is encouraging and even heartwarming. Now is the time to pursue a designer's dream: when large publishers desire quality content and actively seek independent groups that demonstrate both a vision and a plan. For those whose business is the creation of worlds, development is returning to the heady days of the mid-1980s, when a few people with a garage and a vision really could revolutionize the computer gaming industry. ■

Sid Meier is an old hand at game development. In addition to cofounding MicroProse Software, he has authored many popular game titles, including F15 STRIKE EAGLE, SILENT SERVICE, PIRATES!, F19 STEALTH FIGHTER, and the classic CIVILIZATION. He is currently involved in developing MAGIC: THE GATHERING.