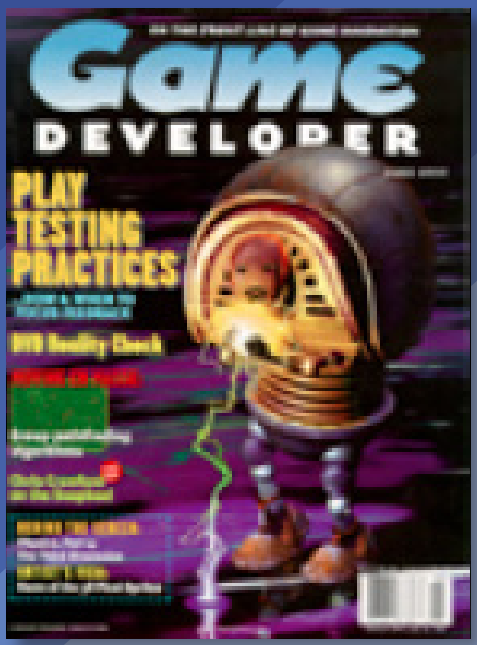
gd

GAME DEVELOPER MAGAZINE

JUNE 1997



UBM
Tech

Pick Up the Phone

There I was last February, in the middle of a developer conference hosted by Mpath. The room was dark and the audience was focused intently on the speaker on stage. Then, from somewhere near me in the dimly lit ballroom, there let loose the instantly recognizable and universally despised ring of a cellular phone. It rang. And rang. And rang some more. Dammit, I thought, didn't this joker know he was plummeting in the popularity polls?

About the sixth ring it happened. In the direction of the cell phone I saw a woman stretch over a desk to her left and pick up the ringing phone. I exchanged glances with the person in front of me, not quite sure what I was witnessing, and in our glance my equally surprised neighbor and I realized that this person had answered a stranger's cell phone in his absence. The woman calmly explained to the person on the other end that the phone's owner had stepped out, that no, she in fact did not know the owner of the phone, and that she would take a message. Then she commenced writing down a name and a phone number on a piece of paper.

I doubt that woman had ever been put in a situation where she had to answer a stranger's phone before. She had no idea who might be on the other end of the phone — a spouse, disgruntled client, the boss with some important news about a recent rash of cellular phone thefts in the company…who knows? She could have just ignored it, turned the phone off, or thrown it across the room. Instead, she acted in the best interests of its owner and of those around her.

Ironically, this is a perfect example of a characteristic that game networks like Mpath have to adopt to survive. It's called making up the rules as you go along. To my knowledge, Ms. Manners hasn't yet tackled the social implications of answering a stranger's cell phone. Ostensibly, that leaves the door open for people to do so without severe repercussions for exhibiting "bad form." Game networks must likewise experiment with their businesses to stay afloat. Not one of them, no matter how much they profess to the contrary, knows if flat-fee, pay-by-the-hour, or advertising-supported revenue schemes are going to work. Maybe in the long run all will survive in some form. Then again, maybe none of these models is viable. Perhaps loss-leaders like Blizzard's Battle.net that provide incremental sales revenue for the publisher will be the answer.

Conclusion: The tide is turning for the game networks. The days of exclusive game hosting deals are history. Game networks are courting developers with as much energy as they devote to courting prospective subscribers. However, today there are simply too many free games being played on corporate LANs, Battle.net-type sites, Quake servers, and so on to ask for money, no matter how much you preach the importance of your low-latency network. Therefore, game networks must stop reinventing their revenue models and start reinventing their content.

Reinvent how? In the short term, I'm betting persistent game worlds like Ultima Online will prove more viable for game networks than the latest first-person 3D action title. Only the most hardcore gamers play a game like Quake for more than 30-40 total hours, and only a subset of those pay to do so online. Dynamic online worlds that present fresh challenges and introduce players to each other outside of kill-or-be-killed environments will bring larger and more heterogeneous audiences to computer gaming, and hence more revenue.

What it will take is a commitment to a long-term goal by the networks. They must take a greater interest in the content that they dish out, and that will probably require partnering with a limited number of developers and working with them hand-in-hand. For some, the venture capital money might not sustain them through such fundamental changes. But for the companies that risk investment as they reinvent themselves and the current rules of Internet game play, I predict a brighter future. ■

ON THE FRONT LINE OF GAME INNOVATION
Game DEVELOPER

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

COVER IMAGE Rocket Science Games

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

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

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

Miller Freeman
A United News & Media publication

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

www.gdmag.com

INDUSTRY WATCH

by Ben Sawyer

CONSTANTLY MONITORING the newswire lets you catch interesting about-faces. Case in point the recent deal between Electronic Arts and Nintendo to bring EA Sports titles to the Ultra-64.

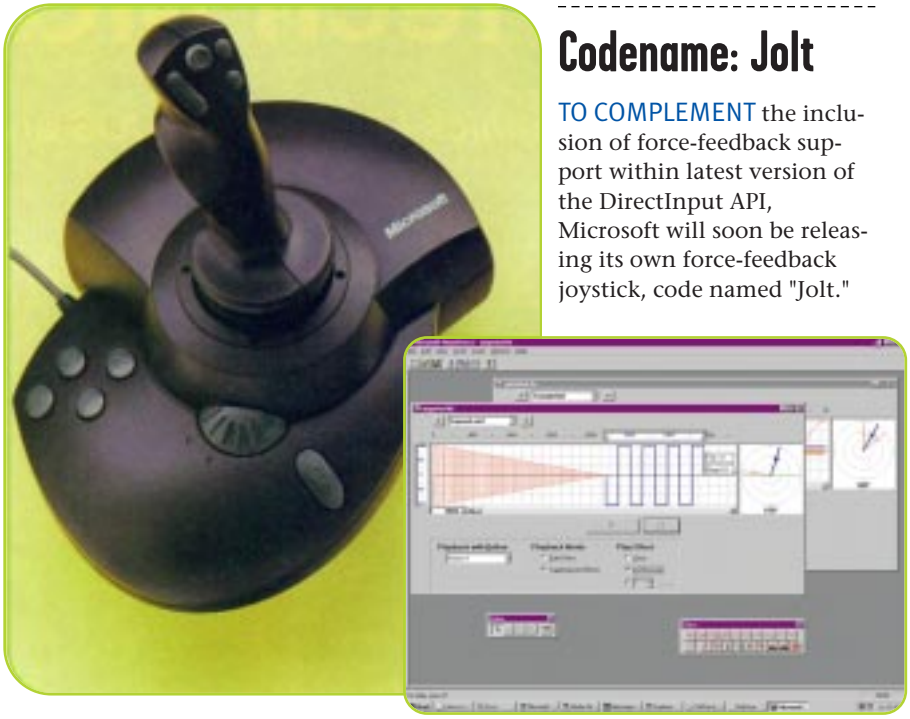
In a recent AP story, Bing Gordon of EA said, "We're pretty strongly in the camp that [thinks that] low manufacturing cost in media is important to the overall growth of interactive entertainment. I doubt we'll ever ship as many products for the Nintendo 64 in any year as we do for the PC or the Sony PlayStation." More importantly the wire headline read, "EA Cool on Nintendo." Was EA sending a message publicly because it wasn't impacting privately?

One month later, things heated up. On March 24, after the original comments hit the news, Nintendo and Electronic Arts announced a major multiyear, worldwide agreement under which EA will publish a line of its EA Sports titles for Ultra-64.

What changed EA's perspective? Or Nintendo's for that matter? Reports speculated that EA had negotiated a favorable royalty agreement. A lot of developers have said it's hard to make money on the Nintendo system. If EA got a sweetheart deal, does that mean Nintendo is going to have to cut such deals with other developers? Better royalties means more products, but Nintendo's own profits from software would be cut. Nintendo clearly feels that having EA in a more active support position was worthy of whatever they gave them.

EA put itself in a strong position. On Ultra-64, it could own the market, at least until another major third party got into it. The only other large sports third parties (Microsoft/Interplay/CUC/Sierra/Accolade) haven't announced Ultra-64 development.

LAST MONTH I SAID there was some resurgence in the RPG category. The suc-



Codename: Jolt

TO COMPLEMENT the inclusion of force-feedback support within latest version of the DirectInput API, Microsoft will soon be releasing its own force-feedback joystick, code named "Jolt."

The stick supports x- and y-axis rotation, throttle, an eight-way hat switch, eight buttons, plus a new shift button that can augment the functionality of the other buttons. For developers, a compelling story is the Visual Force Factory utility that comes with the joystick's SDK. Visual Force Factory lets you create effects using a graphical interface and immediately play them back to the joystick, letting you test an effect and make alterations as necessary by changing force parameters such as frequency and direction. These effects are stored as .FRC (force) files (which contain a single effect), or .VFX files (force resource scripts) that contain multiple force files. The joystick will be available in October, and the SDK is currently available.

- Microsoft Corp.
  Redmond, Wash.
  206-936-8643
  brettsc@microsoft.com

Virtual Internet Testing Environment

OUTSOURCED SOFTWARE testing provider ST Labs announced a virtual Internet test environment to isolate, control, and test variables affecting multiplayer game performance over the Internet. The company examines game performance characteristics during a real-world Internet session over a 24-hour period. These results are then used in a testing environment where latency, bandwidth, and dropped or reordered packets are manipulated to see how these variables affect the game. Armed with these results, you can recode or redesign your game to minimize adverse Internet effects.

- ST Labs Inc.
  Bellevue, Wash.
  206-974-0174
  www.stlabs.com

ASTS OF GAME DEVELOPMENT

HyperMatter

LONDON-BASED Second Nature Industries has developed an animation plug-in for 3D Studio MAX. HyperMatter will be distributed by Kinetix. Physics-based computations allow animators to simulate the real-world interactions of nonrigid objects.

Objects can be squashed, deformed, stretched, wobbled, twisted, and bent, all while retaining their real physical attributes. Animators can simulate the fluid, elegant movements of pliable objects such as people, pillows, balls, marshmallows, clouds, curtains, or anything else that can swing, drop, stretch, or hang by its corners.

Specific features include letting users rubberize or elasticize objects; dynamic simulation of physical movements; real-time interaction between objects, including full collison detection; and complete control over the behavior of objects.

HyperMatter is availble now at a list price of $800.

- **Kinetix**
  San Francisco, Calif.
  800-879-4233
  www.ktx.com

DirectX 5

THE LATEST VERSION of DirectX, version 5 (hey, what happened to version 4?), has been split into two layers: the foundation layer and the media layer. The lower level services are found in the foundation layer. Some of the changes found in release 5.0 include the new DrawPrimitive API for passing polygon information to the hardware using Direct3D; multimonitor support; USB support for game and audio devices; AGP support; MMX optimization; 3D sound acceleration; Talisman rendering features; optimized texture support; and the aforementioned force-feedback joystick support.

You'll also be happy to hear that they've improved the DirectX documentation and the DirectX Setup service.

- **Microsoft Corp.**
  Redmond, Wash.
  206-882-8080
  www.microsoft.com/directx

New Internet Game Server Software

RTIME INTERACTIVE'S recently announced Networking Engine 2.0, targeted at publishers who want to host their own game networks, enables better performance over the Internet and supports thousands of players and spectators. The engine is a client/server, software-only package that uses client frame-rate decoupling, dynamic motion modeling, a global timebase, and real-time information filtering. The company offers two configurations of the product: RTime 64, which supports up to 64 simultaneous participants per server, and RTime Unlimited which supports unlimited participants on each server. Game clients run on the Macintosh, Windows 95 and NT, SGI, and Solaris. The server engine runs on SGI, Solaris, and Window NT. The SDK is free, and depending on whether your game is pay-to-play or a free service included with a retail purchase (à la Battle.net), RTime charges a per-hour or per-box fee.

- **RTime Inc.**
  Seattle, Wash.
  206-281-7990
  www.rtimeinc.com

cess of Blizzard's DIABLO only fuels the fire for more RPGs or derivative works.

FINAL FANTASY VII represented a major win for Sony when it got the exclusive release of this RPG for the PlayStation. The product is slated for release in the U.S. in September as the lead Sony title for the holidays.

MERIDIAN 59 has become a central product for 3DO. A new update recently shipped.

Origin Systems' ULTIMA ONLINE has been profiled in many major game magazines (including this one). Another EA division, Bullfrog, is working on its major release for '97. DUNGEON KEEPER is a twist on the RPG plot in that players are trying to stop the heroes.

After John Romero left id, he said in interviews that a key to Ion Storm's product development plans is the creation of PC-based FINAL FANTASY-style RPGs.

Nintendo has ZELDA-64 coming up for Christmas '97 or early '98. It might be the lead title for its 64DD optical storage attachment for the Ultra-64.

Matsushita wants its M2 machine to be a major RPG platform. It recently showed off a M2 RPG, POWER CRYSTAL, from U.K. development house Perceptions. The screen shots alone should spark many fans' interest in RPGs on the M2.

Interplay and Sierra are planning several big RPG products, too. Sierra has DAEMON ISLE and BETRAYAL AT ANTARA, and Interplay has a major AD&D product, IRON THRONE, that will be launched online.

The RPG category benefits from larger storage capacity on consoles. The demand for persistant and interactive worlds and perhaps just a general pendulum swing back to deeper titles is also helping the genre's popularity. In any case it's interesting to note how important the specific RPG titles are to a number of heavyweights.

NEWS RESOURCE OF THE MONTH: *The Wave Report on Digital Media* is a great way to keep up on the realm of real-time 3D and 3D graphics animation.

To subscribe to *Wave*, send an e-mail message with "subscribe wave <your name>" in the body of the message to listproc@listserver.com. Previous issues of *Wave*, as well as other info can be found at http://www.fourthwave.com/wave

Physics, Part 4: The Third Dimension

It pains me to say it, but this is going to be my last column for a while. Writing these columns takes a lot of time, and right now I need to devote all my waking hours to my startup game company and to shipping our first game. Still, I'm going to stay on as *Game Developer*'s Editor-at-Large, so I will have input on the magazine — I might even write an article during the hiatus — but this is the last official Behind the Screen for at least a year. I love writing this column, so you can be sure I'll be back as soon as possible. In the meantime, remember that one of the reasons I write these articles is to encourage information sharing among game developers — if you have an idea for an article you'd like to write, don't hesitate to propose it to the editor. The more information we share, the faster our industry advances, and that's good for everybody.

As my swan song, I'm giving you this monster of an article to finish up the physics series. Twice the length! Twice the number of equations! Twice as late turning it in to my editor! Off we go!

Prelude

Personally, I think 2D physics is really cool. Still, you'll never forget the first time you see a physically simulated 3D object careen off a wall — especially when you wrote the simulator yourself. Also, for better or for worse, most of the games coming out these days are 3D. Unless you're writing the world's most realistic side-scroller, you're going to need the 3D equivalents of the first three colunms in this series. This installment is huge because I'm going to cram all three into a single column. To do this, I'm going to have to assume you know the material from the first three columns, so you might want to go back and read them again before going any farther.

Like the previous articles, this one is divided into a section on kinematics and a section on dynamics. The kinematics will tell us how to represent the 3D object's configuration and its derivatives, and the dynamics will tell us how to relate these kinematic quantities to each other and to external forces and torques. For the most part, the transition from 2D to 3D is intuitive, but as you'll see, the lack of a good parameterization for 3D orientation mucks up the works a bit. But I'm getting ahead of myself....

A swan song if you will, a desperate dash for closure if you won't. The physics series comes to a roaring conclusion by applying all you've learned so far to the deep dimension.

3D Kinematics

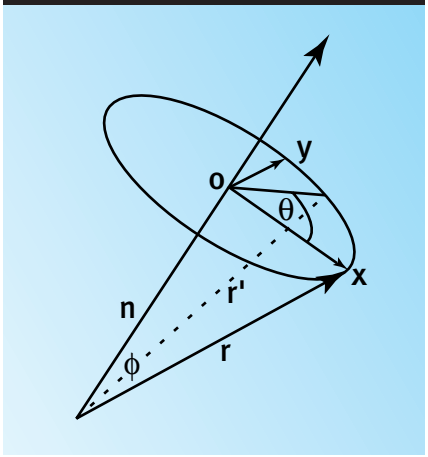
First, the easy part. The equations for 3D linear kinematics (position, velocity, and acceleration) are exactly the same as for their 2D counterparts. The two-element vectors turn into three-element vectors, and you're done.

Unfortunately, it's not so easy when we take 3D orientation into account. Consider the wonderfully elegant representation of an orientation in 2D: a scalar. It's hard to get simpler than this and still represent some useful information. As we've seen, the orientation value Ω, its time derivative ω, and its second derivative α are all scalars that nicely parameterize any orientation and change of orientation in two dimensions. However, a single scalar clearly isn't going to cut it for 3D orientation. But what representation will?

In order to answer this question and keep this article only two times larger than normal, I'm now forced to skip a ton of math. Rotation in 3D is an incredibly rich subject with deep ties to almost every field in mathematics. The classical mechanics text by Goldstein in the references on my web site (the URL's at the end of the article) has a 50-page chapter on 3D orientation, and yet there are still plenty of places in the chapter where Goldstein has to rein himself in to stay on course. Given the impossibility of covering orientations even superficially, we need to be content to spend only the next paragraph rationalizing our choice of representation, and then move on to describe our representation's properties.

There are three angular degrees of freedom in 3D (the three linear and three angular degrees of freedom add up to the oft-heard "6DOF"), so we need to use at least three scalars to describe an arbitrary orientation. At this point, math deals us a temporary setback. It's possible to prove that no three-scalar parameterization of 3D orientation exists that doesn't suck, for some suitably mathematically rigorous definition of "suck." I haven't done this proof (I think it uses some pretty high-end group theory, which I haven't learned yet), so I can't tell you

FIGURE 1. *Axis-angle rotation.*



exactly how it works, but I believe the gist of the proof is that no minimal parameterization exists that doesn't contain singularities. These singularities can take different forms — depending on how you allocate the three degrees of freedom — but according to the math, it's impossible to get rid of them. Anyone who's played around with the most common minimal parameterization of 3D, the set of three Euler angles (roll, pitch, and yaw are one possible set), has probably run into some of these singularities. Luckily, we aren't forced to use only three parameters. We can avoid the singularities by using more parameters, as long as we constrain our multiple parameters down to three degrees of freedom. This is exactly what we're going to do by chosing 3×3 matrices to represent our orientations.

Even though I said we'd only use one paragraph to rationalize our orientation parameterization, I'm going to cheat a bit and spend another paragraph describing what I mean by "constrain those parameters down." As a relatively intuitive example, let's say we want to represent a 2D position. The obvious way to do this is to use a 2D vector and be done with it. If we were feeling particularly nonoptimal — or if there was some problem with using 2D vectors — we could use a 3D vector to represent 2D position, as long as the tip of that vector was constrained to move in a plane. We could implement this constraint with a single dot product equation. If the dot product of our variable 3D position vector with another constant vector was always constrained to be a constant value, then the tip of the 3D vector must always move in a plane. This 3D vector minus the single scalar constraint leaves us with only two degrees of freedom to move in the plane — this is the same as using a 2D vector in the first place. As a rule, the original number of unconstrained degrees of freedom minus the number of scalar constraint equations leaves us with our final number of degrees of freedom. This concept of degrees of freedom and constraint equations becomes incredibly important as you learn more about dynamics (and about math in general). Mull this over for a while until you're comfortable with the idea.

Now, as I was saying, we're going to use 3×3 matrices to represent the orientations of our rigid body. Clearly, an arbitrary 3×3 matrix has nine degrees of freedom (one for each scalar in the matrix), so we're going to need some constraints to get down to the three degrees of freedom needed to represent a 3D orientation[1]. We get these constraints by restricting our matrices to be *special orthogonal*. The "special" part means the matrix is not a reflection — it can't turn a right-handed coordinate system into a left-handed one. The "orthogonal" part comes from the following matrix equation:

$$\mathbf{A}\mathbf{A}^T = \mathbf{1} \qquad \text{(Eq. 1)}$$

In English, A times its transpose, $\mathbf{A}^T$, yields the identity matrix, or put another way, $\mathbf{A}^T = \mathbf{A}^{-1}$ — transpose is the inverse. Eq. 1 also implies $\mathbf{A}^T\mathbf{A} = 1$. The theory of orthogonal matrices is at least as large as that of 3D orientations, so again I'm only going to touch on the highlights that directly affect us. Eq. 1 gives us our six constraint equations because it's a bunch of dot products of the rows of **A**. Three constraints come from the 1s on the diagonal of the identity matrix, meaning the rows are unit length. The other three constraints are from the 0s, meaning the rows are all at right angles to each other. Those constraints bring us down to exactly three degrees of freedom in **A**. Most people are aware that 3D rotations are orthogonal and obey Eq. 1, but it's also possible to prove the converse: that any special orthogonal 3×3 matrix is a rotation. As long as we enforce the six constraints of Eq. 1, we have a valid rotation. As a side note, those of you who have used 3×3 matrices to represent orientations have probably run into problems when the orthogonality constraints were not enforced in the face of numerical inaccuracy. In this case, your "rotation matrix" probably started scaling and shearing your objects instead of just rotating them.

We're still in mathematical fast-forward mode, so I'll just point out that a special orthogonal matrix operates on (or rotates) a vector through plain old matrix multiplication. This is one reason a 3×3 matrix is a more convenient orientation representation than a set of Euler angles — Euler angles require evaluating trig functions to rotate a vector. Also, the matrix product of two special orthogonal matrices is the cumulative rotation (applying the product **BA** to a column vector is the same as applying **A** and then **B**), which means the product must be another special orthogonal matrix. Finally, matrix multiplication is not commutative (**BA** is different than **AB**). This mirrors the noncommutativity of rotations; it's easy to construct a sequence of rotations that, when performed in a different order, result in a different final orientation.

I want to take a step back at this point and explain why I'm being slightly strange in my discussion of rotation matrices. Don't I understand that everyone knows that a matrix can rotate a vector, and that matrix concatenation works? Sure, and in fact I'm counting on you knowing this since I don't have room to explain that stuff in this column. However, I've found most computer graphics-oriented textbooks only explain how to construct rotation matrices ("put the sines and cosines in these places"), but they don't talk about many of the formal properties of rotation matrices. In dynamics, after giving our objects their initial orientations, we never again construct rotation matrices from scratch. Our orientations evolve as a result of the integration we perform on the dynamic system — knowing how to create a rotation around the z-axis doesn't help us much. Another important point is that the 3×3 matrix *is* the orientation. In graphics, we learn to use matrices to cause rotations, but in

[1] *Lots of people use objects called quaternions to represent orientations. Quaternions have four parameters and need one constraint. Usually the quaternion is constrained to be unit length.*

this column, the matrix simply is the orientation representation (in addition to having the nice property that it causes the rotation when multiplied with a vector).We're not, for example, using Euler angles and converting them to matrices in order to operate on vectors with them; we're storing the matrix as our only representation of our objects's orientation. So, if someone asks for object A's orientation, we hand him or her the whole 3×3 matrix, with assurances the matrix is special orthogonal so it really does represent an orientation. If we don't make sure it's special orthogonal, our orientation representation won't work properly. While we gain simplicity over Euler angles, we give back some of that gain because we're required to enforce the constraints on our matrices. I wish I could spend more time going into the subtleties of 3D orientation, but I can't, so you'll have to discover them for yourself from the references. Anyway, bear with me: Take your current knowledge of matrices, add it to anything new you learn here, and realize that we're talking about the same matrices in the end — now you just see them from a different side.

To warm up for the equation manipulation to come, let's prove a fundamental result for orthogonal matrices. We'll use this result later. Start with a rotation matrix **A** that transforms any vector **r** to **r′** by **r′** = **Ar**. Now, say we want to be able to apply a (possibly nonrotation) matrix **B′** to **r′** that will have the same effect as a matrix **B** that's applied to **r** before **A** rotates it. Symbolically, we want to figure out **B′** in **B′Ar** = **ABr**. Thinking about it another way, how do we "rotate" the matrix **B** by **A** so it will work in the primed space? We begin by noting that **r** = 1**r**. I can therefore insert the identity matrix into the right-hand side of the previous equation, giving us **AB1r** (inserting an identity matrix is a common linear algebra trick). The equality $\mathbf{A}^T\mathbf{A} = 1$ from Eq. 1 also gives us $\mathbf{B'Ar} = \mathbf{ABA}^T\mathbf{Ar}$. Comparing the two terms gives the following equation:

$$\mathbf{B'} = \mathbf{ABA}^T \qquad \text{(Eq. 2)}$$

Eq. 2 defines what is called in linear algebra a "similarity transform." It shows how to rotate **B** to get a matrix in the primed space that operates on primed vectors in the same way **B** operates on vectors in the unprimed space. Neat trick, huh? You could use Eq. 2 to find a matrix that will rotate an object around its local x-axis in world space: Create a **B** that's an x-axis rotation, then use **A**, the local-to-world transformation, to similarity-transform **B** (although in this case, it's probably easier just to rotate the object around the local x-axis when it's in local space before applying **A**, but if you didn't have the original **r** you'd need Eq. 2…hey, it's just an example).

Axis and Angle

We've decided on our kinematic representation for orientation, but we still need to pick representations for the kinematic derivatives: angular velocity and angular acceleration. To do that, we need to explore our orientation representation in a little more detail. I'll give you one more unproven fact, then we'll slow down and derive some results ourselves.

The fact is that any rotation (and this includes all combinations of rotations) can be described by a single unit vector and a rotation angle around that vector. This means you can concatenate any convoluted sequence of rotations you like, and if you simply tell me the start and the end orientations, I can give you back a unit vector and a scalar encapsulating the change in orientation. The scalar tells how far to rotate around the vector. This rotation will take you from your start to your end orientation in one step. (Note how many degrees of freedom we're talking about here: three for the elements of the vector, plus one for the angle, minus one for the vector's unit-length constraint leaves us with — surprise — three.)

We can also directly construct a rotation equation from the unit vector and the angle. Let's start with a unit vector **n**, an angle θ around that vector, and the arbitrary vector to rotate **r**. Figure 1 shows the situation, with **r′** as the resultant vector. If we look down **-n** onto the plane of rotation containing the tips of **r** and **r′**, we see the circle of rotation in Figure 2. As we know from trigonometry, if we consider the tip of **r** to be on the x-axis of this diagram, then the coordinates of the tip of **r′**, measured in this planar coordinate system, will be ($x = r\cos\theta$, $y = r\sin\theta$), where *r* is the radius of the circle. This (x,y) coordinate notation is just a shorthand way of saying the vector sum $\mathbf{o} + r\cos\theta\mathbf{x} + r\sin\theta\mathbf{y}$ or, "start at the origin **o**, go *r*cosθ units down the **x** vector, and then *r*sinθ units down the **y** vector." So, all we need to do is to form the vectors **o**, **x**, and **y** in the 3D space, then apply the 2D rotation formula to them.

First, we define the origin. The origin is the vector parallel to **n** with its tip on the plane of rotation. We can form this vector by projecting **r** onto **n** with a dot product, then moving the resulting distance down **n**.

$$\mathbf{o} = \mathbf{n}^T\mathbf{rn} \qquad \text{(Eq. 3)}$$

Eq. 3 uses the "matrix notation" for the dot product. If we transpose the column vector **n**, we get a row vector. A row vector times a column vector **r** is equivalent to a dot product and results in a scalar (for matrices, 1×n * n×1 = 1×1). The **o** vector moves us to the plane of rotation. We can trivially define the **x** vector as the difference between the tip of **r** and the **o** vector.

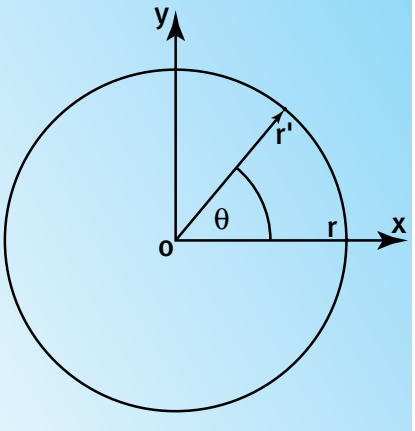
$$\mathbf{x} = \mathbf{r} - \mathbf{n}^T\mathbf{rn} \qquad \text{(Eq. 4)}$$

Note that we aren't normalizing **x** because its length is exactly what we want: the radius of the rotation circle *r*. Finally, we form the **y** vector using a cross product of **n** and **r**.

$$\mathbf{y} = \mathbf{n} \times \mathbf{r} \qquad \text{(Eq. 5)}$$

The cross product forms a **y** that is perpendicular to both **n** and **r**, and hence **x**, since **x** is a linear combination of the two. The **y** vector is also the perfect length, since the magnitude of the cross product is equal to $|\mathbf{r}|\sin\phi$ (**n** is unit length), which conveniently

FIGURE 2. *The circle of rotation.*



equals *r*, the radius of the circle, as you can see in Figure 1. Putting it all together, we get

$$\mathbf{r}' = \mathbf{n}^{\mathrm{T}}\mathbf{r}\mathbf{n} + \cos\theta\left(\mathbf{r} - \mathbf{n}^{\mathrm{T}}\mathbf{r}\mathbf{n}\right) + \sin\theta\left(\mathbf{n}\times\mathbf{r}\right) \quad \text{(Eq. 6)}$$

This is one form of a famous formula on whose name no one seems to agree. I've heard it called *The Rotation Formula*, *Rodriguez's Formula*, and a bunch of other names. No matter; we'll call it Eq. 6. Eq. 6 will rotate any **r** around **n** by θ. We're not actually going to use Eq. 6 to rotate vectors, although it would do the job just fine. Instead, we're going to use it to prove useful kinematics equations for 3D orientation. We could also construct a rotation matrix from Eq. 6 by "pulling out" the **r** vector from the right-hand side, but we're running out of space, so I highly recommend exploring that yourself. (Hint: Try to figure out the 3×3 matrix associated with each term, so that the matrix times **r** would equal the terms in Eq. 6. You'll need the "tilde operator," which I'll discuss later.)

Angular Velocity

In 2D, we used the time derivative of our orientation scalar as our angular velocity scalar. The angular velocity scalar, when combined with the perpendicular operator, was also useful for finding the velocity of any point in a rotating body. In 3D, our orientation is a 3×3 matrix. Is our 3D angular velocity required to be the time derivative of our orientation matrix? The answer is no, the angular velocity representation doesn't have to be the time derivative of the orientation representation. It's only important that we're able to calculate the orientation matrix's derivative so we can integrate it — we don't have to use the derivative beyond that. We'll see how to make the needed calculation later.

It may seem strange that we can use a fundamentally different representation for our angular velocity than we used for our orientation. Unfortunately, we don't have the space to go into why this is possible, but it's covered in most of the references on my web site. Let's work through a few derivations to define and get comfortable with the angular velocity.

First, we'll calculate the linear velocity of the vector **r** in Figure 1. If we assume **r** is rotating over time around a fixed **n**, then we can again reduce the problem to the planar Figure 2, and use similar arguments for the velocity of **r** as we did in my article on 2D angular velocity. The first argument from the 2D article showed the magnitude of the velocity vector as $r\dot{\theta}$, which we'll recognize as $|\mathbf{r}|\sin\phi\,\dot{\theta}$ from Figure 1. Next, we can see the velocity vector must point perpendicularly to **r** and to **n**. This is true because **r** is only rotating about **n**, so the tip of **r** is always moving normal to the plane containing **r** and **n** as it rotates. So, what's a vector expression that yields a vector of the right magnitude pointing in the right direction? How about this:

$$\dot{\mathbf{r}} = \dot{\theta}\mathbf{n}\times\mathbf{r} = \omega\times\mathbf{r} \quad \text{(Eq. 7)}$$

If we define the angular velocity vector ω as the current instantaneous axis of rotation times the rotation speed ($\dot{\theta}\mathbf{n}$), then we get an expression that is very similar to the one for 2D, only with a cross product replacing the perpendicular operator — I told you the two operators were related. Remember, like the 2D version, Eq. 7 is only valid if **r** is a constant vector — it can rotate around, but it can't change length and keep Eq. 7 valid.

Here's a totally different way to derive the same result: We can consider Eq. 6 as a function that describes the path of the vector **r**′ as it rotates by θ radians from its initial position **r**. If θ is a function of time, and **n** is a constant axis of rotation, we can differentiate Eq. 6 with respect to time.

$$\dot{\mathbf{r}}' = -\sin\theta\dot{\theta}\left(\mathbf{r} - \mathbf{n}^{\mathrm{T}}\mathbf{r}\mathbf{n}\right) + \cos\theta\dot{\theta}\left(\mathbf{n}\times\mathbf{r}\right) \quad \text{(Eq. 8)}$$

We consider **r** in Eq. 6 to be constant as well, since it's just the initial position of the nonconstant vector **r**′.

Finally, evaluate Eq. 8 at some time *t*. We can always define θ(*t*) to be 0 in Figure 2 by choosing an appropriate frame of reference. Specifically, we make the "x-axis" of the figure be the plane containing **r** and **n** at any given instant. Within this frame of reference, **r**′ = **r**, sin0 = 0, cos0 = 1, and we're left with Eq. 7! Remember, just because θ(*t*) = 0 in our frame of reference, it doesn't mean $\dot{\theta}(t) = 0$.

This vector ω is the representation we'll use for our angular velocity. The vector we've defined is "instantaneous" in the sense that it's a valid representation of the angular velocity at a given instant, but not before or after that instant. The instantaneous axis of rotation can and will change under the application of forces and torques. This means we can use it to calculate velocities of points at the instant it's valid, but we can't store it and use it later without keeping it up-to-date via integration. More on that later.

As a final derivation, we'll use Eq. 7 to calculate the derivative of the current orientation matrix using the angular velocity vector. This is a bit tricky, so hold on tight. First, we know from graphics that the columns of a rotation matrix are unit vectors in the transform's destination coordinate system. Well, Eq. 7 shows the angular velocity vector "differentiating" a column vector, and there's no reason we can't use the angular velocity to differentiate each column vector of the orientation matrix, resulting in the differentiated matrix. The only problem is that the cross product of a vector and a matrix isn't usually defined. However, we can represent a cross product as a matrix times a column vector, like this:

$$\dot{\mathbf{r}} = \omega\times\mathbf{r} = \tilde{\omega}\mathbf{r} = \begin{vmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{vmatrix} \begin{vmatrix} r_1 \\ r_2 \\ r_3 \end{vmatrix} \quad \text{(Eq. 9)}$$

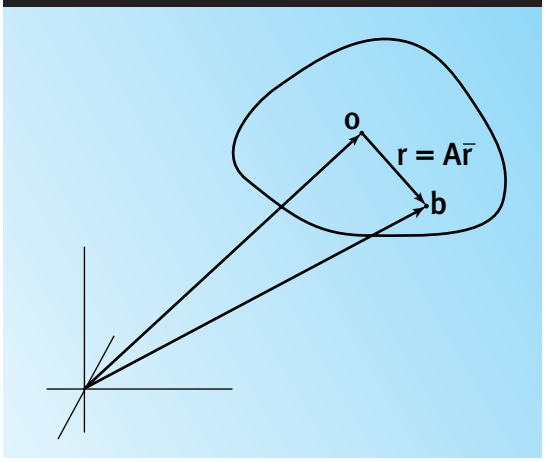
The tilde operator, introduced in the third term, takes a vector and creates the "skew-symmetric" matrix depicted in the final term. If you write out the matrix multiply by hand, you'll see it's equivalent to the cross product. We use the tilde operator to differentiate each column with a single matrix multiply.

$$\dot{\mathbf{A}} = \tilde{\omega}\mathbf{A} \quad \text{(Eq. 10)}$$

The right side of Eq. 10 will differentiate each column of **A**, which differentiates the whole matrix. We could have defined a vector cross a matrix as the column-wise cross product, or we could have just looped through the columns doing cross products individually. But then you would have missed out on the groovy new tilde operator in Eq. 9, so it was worth it. Plus, we'll use this operator again later.

It's important to stress a couple things about Eq. 10. First, the left-hand side is the instantaneous derivative of **A**, meaning it's only the derivative at the instant of time when ω is valid. However, that's all we need to

FIGURE 3. *A point on a rigid body.*



numerically integrate **A** forward to the next step, as we'll see. Second, the axis of the angular velocity vector and the axis of the rotation matrix can be different, and Eq. 10 still holds. In other words, if we have our current orientation **A**, and our body has some angular velocity, embodied in ω, then Eq. 10 will calculate how the orientation of **A** is changing at that instant under the influence of the angular velocity. This is how our body's orientation changes in the simulator — we relate the forces and torques to changes in ω, and use ω with Eq. 10 to integrate our body's orientation.

Kinematic Equations for a Point on a Moving Body

Let's use all the kinematics that we've developed so far to write the equations for a point's position and its derivatives. The position vector of the point **b** is given by the position vector of the origin of the body **o** plus the vector from **o** to **b** in the body, which we'll call **r**. Figure 3 shows this configuration. The vector **r** rotates with the body as shown in the figure. Since the body is rotating, **r** is the rotated world-space version of a vector we'll call $\bar{\mathbf{r}}$ that's constant in the body space. Now we can write the position equation for **b** in world space.

$$\mathbf{b} = \mathbf{o} + \mathbf{A}\bar{\mathbf{r}} = \mathbf{o} + \mathbf{r} \qquad \text{(Eq. 11)}$$

If we differentiate Eq. 11, we'll get the velocity of **b**. The **o** vector is easy, since it's just translating around, keeping track of the origin — its derivative is just $\dot{\mathbf{o}}$, or the velocity of the body's origin. There are two equivalent ways of differentiating the rotating **r** vector, though. First, we'll use Eq. 7 and differentiate the last term in Eq. 11 directly.

$$\dot{\mathbf{b}} = \dot{\mathbf{o}} + \boldsymbol{\omega} \times \mathbf{r} \qquad \text{(Eq. 12)}$$

Next, for a change of pace, we'll differentiate the middle term in Eq. 11 explicitly, using the product rule for derivatives.

$$\dot{\mathbf{b}} = \dot{\mathbf{o}} + \dot{\mathbf{A}}\bar{\mathbf{r}} + \mathbf{A}\dot{\bar{\mathbf{r}}}$$

Since $\bar{\mathbf{r}}$ is a constant vector in the body, its time derivative is 0. We can also substitute Eq. 10 into this equation and we get

$$\dot{\mathbf{b}} = \dot{\mathbf{o}} + \tilde{\boldsymbol{\omega}}\mathbf{A}\bar{\mathbf{r}} = \dot{\mathbf{o}} + \tilde{\boldsymbol{\omega}}\mathbf{r} = \dot{\mathbf{o}} + \boldsymbol{\omega} \times \mathbf{r}$$

In other words, both ways of finding **b**'s velocity are equivalent — score one point for math. We differentiate one more time to find **b**'s acceleration. (I'm only going to do it one way this time. You should try the other yourself.)

$$\ddot{\mathbf{b}} = \ddot{\mathbf{o}} + \dot{\boldsymbol{\omega}} \times \mathbf{r} + \boldsymbol{\omega} \times \dot{\mathbf{r}}$$

$$\ddot{\mathbf{b}} = \ddot{\mathbf{o}} + \boldsymbol{\alpha} \times \mathbf{r} + \boldsymbol{\omega} \times (\boldsymbol{\omega} + \mathbf{r}) \qquad \text{(Eq. 13)}$$

We should have expected the derivative of the angular velocity vector, the angular acceleration vector α, to show up, but what's the third term doing there? The math has magically produced the centripetal acceleration of a rotating point! In other words, if you look at the direction in which the third term is pointing, you'll see it's pointing back towards the origin of the body. This is the acceleration you feel if you're stuck to the wall of one of those spinning carnival rides. You actually feel it as a force pushing you into the wall, but that's only because the wall is accelerating towards the center to keep from being flung across the fairgrounds. (Mathematically, this is the restriction that **r** is constant in body-space.) I just love dynamics.

Interlude

What you just read was longer than any of my other columns, and we haven't even covered 3D dynamics yet. We have come a long way, though. We've chosen representations for the position, linear velocity, and linear acceleration, and also for the angular quantities of orientation, angular velocity, and angular acceleration (I slyly stuck this one into Eq. 13 as α, the derivative of ω). We've also shown how to use ω to differentiate vectors and matrices, and we calculated the velocity and the acceleration of any point on a moving body.

The only things left to do before we have a full 3D dynamic simulation algorithm are to develop the 3D dynamic quantities and equations, relate those dynamic quantities to the kinematic quantities, and show how to integrate them all forward in time.

3D Dynamics

Like 3D linear kinematics, 3D linear dynamics and 2D linear dynamics are identical, with the exception that the vectors now have a z element. In the 2D articles, we derived equations for the force and momentum of a single particle, then derived the position vector to the center of mass. Since the derivations are identical in 3D, I'll just state the results without proof. (Note that I'm switching from the superscripted indices that I used in the 2D columns to subscripted indices here so I don't confuse "total" values with the transpose operator. Sorry about that.)

$$\mathbf{F}_T = \sum_i \dot{\mathbf{p}}_i = \sum_i m_i \dot{\mathbf{v}}_i = \sum_i m_i \mathbf{a}_i = M\mathbf{a}_{CM} \qquad \text{(Eq. 14)}$$

Eq. 14 says the total force $\mathbf{F}_T$ equals the sum of all the momentum derivatives, which is equivalent to the mass of the whole body *M* times the acceleration of the center of mass $\mathbf{a}_{CM}$. If I know all the forces on the body, I take their vector sum and divide by the total mass to find the acceleration of the center of mass. I then can integrate the acceleration forward in time to find the new center of mass velocity and position.

The 3D angular dynamic quantities are, as you might expect, slightly different than the 2D angular dynamic quantities. First, we'll define the angular momentum of point B about point A in three dimensions. In 2D, the angular momentum was a scalar formed by a perp-dot product. We visualized this quantity capturing the amount of B's linear momentum "rotating around" A. Well, in 3D we need an axis to rotate around, so the angular momentum becomes a vector **L**. **L** is calculated with

a cross product, which conveniently creates a vector perpendicular to both the linear momentum of B, $\mathbf{p}_B$, and the vector from A to B, $\mathbf{r}_{AB}$. In other words, the cross product creates a vector that describes the plane of the momentum's rotation around A. The magnitude of $\mathbf{L}$ is proportional to the sine of the angle between the two vectors and measures the amount of momentum that's perpendicular to $\mathbf{r}$.

$$\mathbf{L}_{AB} = \mathbf{r}_{AB} \times \mathbf{p}_B \qquad \text{(Eq. 15)}$$

As in two dimensions, the derivative of the angular momentum is the torque, denoted by τ. A little bit of work will show the following identities hold:

$$\dot{\mathbf{L}}_{AB} = \tau_{AB} = \mathbf{r}_{AB} \times \mathbf{F}_B \qquad \text{(Eq. 16)}$$

The derivative of the angular momentum is the torque, and it can be calculated from the cross product of the vector from the point of measurement to the point where the force is being applied. The torque measures the amount of "rotating-around" force experienced from a given point.

The next thing we need to do is develop the "total" versions of these quantities. That is, what is the angular momentum for the entire body? As in 2D, the total angular momentum is just the sum of all the angular momentums of all the points in the body measured from a point (usually the center of mass).

$$\mathbf{L}_{AT} = \sum_i \mathbf{r}_{Ai} \times m_i \mathbf{v}_i = \sum_i \mathbf{r}_{Ai} \times m_i \dot{\mathbf{r}}_{Ai}$$

I've taken the liberty of rewriting the momentum of the point being measured as its mass times its velocity — I even went a step farther in writing it as the position vector's time derivative. This is the first step in linking the angular dynamic quantities with the angular kinematic quantities. The next step is to substitute Eq. 7 into the equation, leaving us with

$$\begin{aligned}\mathbf{L}_{AT} &= \sum_i m_i \mathbf{r}_{Ai} \times (\omega \times \mathbf{r}_{Ai}) \\ &= \sum_i -m_i \mathbf{r}_{Ai} \times (\mathbf{r}_{Ai} \times \omega)\end{aligned}$$

I flipped the order of the inner cross product, which causes the result to change sign. Finally, we use the all-powerful tilde operator from Eq. 9 to turn the equation into a matrix multiply: Both $\mathbf{r}$ cross products are replaced with $\tilde{\mathbf{r}}$, leaving ω on the right-hand side.

$$\mathbf{L}_{AT} = \sum_i -m_i \tilde{\mathbf{r}}_{Ai} \tilde{\mathbf{r}}_{Ai} \omega = \mathbf{I}_A \omega \qquad \text{(Eq. 17)}$$

The inertia finally rears its head in 3D, though it's now a matrix rather than a scalar! Since ω is constant over the whole body, as in 2D, we can pull it outside the summation. This leaves us with a matrix called the "inertia tensor," relating the angular velocity of a body to the angular momentum of the body. The inertia tensor obviously is a lot more complicated than the single scalar moment of inertia from 2D. To make matters worse, the inertia tensor changes as the object rotates because it depends on the world-space $\mathbf{r}$s.

If we ignore the change in the inertia tensor for a moment, we can actually begin to see how we might implement 3D angular dynamics. We can easily find the total torque on the body —

measured about the center of mass — by forming the vector sum of all the individual torques produced by force applications via Eq. 16. If we integrate this torque, we'll get the total angular momentum about the center of mass. Then, assuming we know the world-space inertia tensor, we can solve the inverse of Eq. 17 to find the current angular velocity for the body.

$$\omega = \mathbf{I}_{CM}^{-1}\mathbf{L}_{CM} \qquad \text{(Eq. 18)}$$

Once we've got the angular velocity, we're home free, kinematically speaking, since we already know how to integrate the orientation using the angular velocity to get the orientation's derivative. The only thing standing in our way is the inertia tensor.

The Inertia Tensor

When we did the derivations leading to the definition of the inertia tensor in Eq. 17, we were using world-space vectors and matrices. This is why the inertia tensor is giving us fits — it changes as the object rotates in world space because it depends on the world-space **r** vectors. However, it's possible to do the derivations in body space. You end up with an inertia tensor based on the fixed body-space $\bar{\mathbf{r}}$ vectors.

$$\bar{\mathbf{I}}_A = \sum_i -m_i\tilde{\bar{\mathbf{r}}}_{Ai}\tilde{\bar{\mathbf{r}}}_{Ai} \qquad \text{(Eq. 19)}$$

The body-space inertia tensor doesn't change (since the body is rigid), so we can compute it once at the beginning of our simulation and store it. We use the similarity transform trick we derived oh-so-long-ago in Eq. 2 to generate the world-space inertia tensor for the current orientation **A**. More interesting, perhaps, is the fact that since the body-space inertia tensor is constant, we can precalculate its inverse before we start. Then we similarity-transform the inverse inertia tensor, and avoid the inversion during the simulation when evaluating Eq. 18 to find the angular velocity vector.

$$\mathbf{I}_A^{-1} = \mathbf{A}\bar{\mathbf{I}}_A^{-1}\mathbf{A}^{\mathrm{T}} \qquad \text{(Eq. 20)}$$

The only piece still missing is a way to calculate the body-space inertia tensor in the first place. For continuous bodies, the summation in Eq. 19 turns into an integral over the body's volume, and for arbitrarily oddly shaped bodies, this integral can get arbitrarily complicated. It's fairly easy to analytically solve the integral for "easy geometry," such as boxes, ellipsoids, cylinders, and the like, and there are tables for other objects. Also, a paper referenced on my web site shows how to calculate the inertia tensor for an arbitrary polyhedron, but the algorithm is way too complicated to go into here. I should also note that if you can't calculate the exact inertia tensor, you can still use the inertia tensor for a tight-fitting approximation volume and the dynamics will be "mostly right."

3D Dynamics Algorithm

We now have the quantities and equations we need to implement 3D rigid body dynamics, and I've outlined the simulation algorithm in

LISTING 1. *The 3D Dynamics Algorithm.*

Initialization:

Determine body constants: $\bar{\mathbf{I}}_{CM}^{-1}, M$

Determine initial conditions: $\mathbf{r}_{CM}^0, \mathbf{v}_{CM}^0, \mathbf{A}^0, \mathbf{L}_{CM}^0$

Compute initial auxiliary quantities:

$$\mathbf{I}_{CM}^{0^{-1}} = \mathbf{A}^0 \bar{\mathbf{I}}_{CM}^{-1} \mathbf{A}^{0^T}$$
$$\boldsymbol{\omega}^0 = \mathbf{I}_{CM}^{0^{-1}} \mathbf{L}_{CM}^0$$

Simulation:

Compute individual forces and application points: $\mathbf{F}_i, \mathbf{r}_i$

Compute total forces and torques: $\mathbf{F}_T^n = \sum_i \mathbf{F}_i, \boldsymbol{\tau}_T^n = \sum_i \mathbf{r}_i \times \mathbf{F}_i$

Integrate quantities:

$$\mathbf{r}_{CM}^{n+1} = \mathbf{r}_{CM}^n + h\mathbf{v}_{CM}^n$$
$$\mathbf{v}_{CM}^{n+1} = \mathbf{v}_{CM}^n + h\frac{\mathbf{F}_T^n}{M}$$
$$\mathbf{A}^{n+1} = \mathbf{A}^n + h\tilde{\boldsymbol{\omega}}^n \mathbf{A}^n$$
$$\mathbf{L}_{CM}^{n+1} = \mathbf{L}_{CM}^n + h\boldsymbol{\tau}_T^n$$

Reorthogonalize $\mathbf{A}^{n+1}$

Compute auxiliary quantities:

$$\mathbf{I}_{CM}^{n+1^{-1}} = \mathbf{A}^{n+1} \bar{\mathbf{I}}_{CM}^{-1} \mathbf{A}^{n+1^T}$$
$$\boldsymbol{\omega}^{n+1} = \mathbf{I}^{n+1^{-1}} \mathbf{L}_{CM}^{n+1}$$

Listing 1. This listing focuses on the parts of the overall simulation loop that changed during the move to three dimensions, so it doesn't cover how collision detection and resolution fit into the picture. See the algorithm in the February/March 1997 "Behind the Screen" for the full loop (or look in the sample code). Let's step through Listing 1.

At initialization time, we need to determine the mass constants for the body. These can be calculated on the fly from the geometry of the object, or loaded in from a file, or even typed in by the user. We also need the initial conditions for the object. I've indicated the "step number" with a superscript, so the initial conditions are all step 0. For the linear quantities, we store the position vector of the center of mass, and its velocity. For the angular quantities, we store the orientation matrix and the angular momentum vector. Before I explain why we store the angular momentum, let's look at the next line in the initialization, `Compute initial auxiliary quantities`. The *auxiliary quantities* are those we derive from the other quantities — we don't integrate them directly. We first compute the initial world-space inverse inertia tensor by similarity-transforming the body-space tensor using the initial orientation matrix (Eq. 20). Then we use this world-space inverse inertia tensor and the initial angular momentum to compute the initial angular velocity (Eq. 18). So, part of the reason we store the angular momentum as a *primary quantity* is because we can compute the angular velocity from it conveniently. The angular momentum is also conveniently integrated from the torque, while the integration from the angular acceleration to the angular velocity is more complicated. (Try differentiating Eq. 17 to find the angular acceleration equation. Keep in mind the world-space inertia tensor's derivative is nonzero.) Finally, the angular momentum vector comes in handy when you want to compute the kinetic energy of the body, which is useful for debugging.

Once we're initialized, we can begin the simulation. The first step is to calculate what the external forces on our body are (from explosions, punches, rockets, or whatever), and where on the body those forces are applied. Once we have this information, we can calculate the total force and torque using Eqs. 14 and 16. Now we're ready to integrate over the timestep h. When looking at these equations, it's important to note the right-hand sides of all the integration steps use the quantities from step n, and the left-hand sides all specify the next step, $n + 1$. The new center of mass position is integrated from the current position and velocity. The new velocity is integrated from the current velocity and acceleration (using the definition of linear acceleration as force over mass, à la Eq. 14). Next, we integrate the orientation. The orientation's derivative is calculated using the current angular velocity as we saw in Eq. 10. In the last integration step, we integrate the new angular momentum vector from the torque. Finally, we need to enforce the orthogonality constraints on our orientation. If our integration was exact, we wouldn't have to do this reorthogonalization, but errors will creep into the orientation over time. There are many ways to reorthogonalize a rotation matrix, but they all amount to making sure the rows and

FIGURE 4. *The 3D collison impulse magnitude.*

$$j = \frac{-(1+e)\mathbf{v}_1^{AB} \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}\left(\frac{1}{M_A} + \frac{1}{M_B}\right) + \left[\left(\mathbf{I}_A^{-1}(\mathbf{r}_{AP} \times \mathbf{n})\right) \times \mathbf{r}_{AP} + \left(\mathbf{I}_B^{-1}(\mathbf{r}_{BP} \times \mathbf{n})\right) \times \mathbf{r}_{BP}\right] \cdot \mathbf{n}}$$

columns are perpendicular and unit length. See the sample code for one technique.

Now that we've got the primary quantities for step $n + 1$, we can calculate the auxiliary quantities from them. This gives us the up-to-date quantities needed for the next integration step. And away we go.

3D Collision Response

We're almost out of space, so I don't have room to derive the 3D collision response equation. However, the 3D derivation is very similar to the 2D derivation in the previous physics column, so you should be able to work it out yourself using the formulas in this article, especially Eq. 12. So that you can check your work, the final 3D equation for the impulse magnitude j is in Figure 4. Just remember, there's no such thing as $\frac{1}{\mathbf{I}}$ when $\mathbf{I}$ is a matrix, so you have to use $\mathbf{I}^{-1}$ and keep track of the order of multiplications.

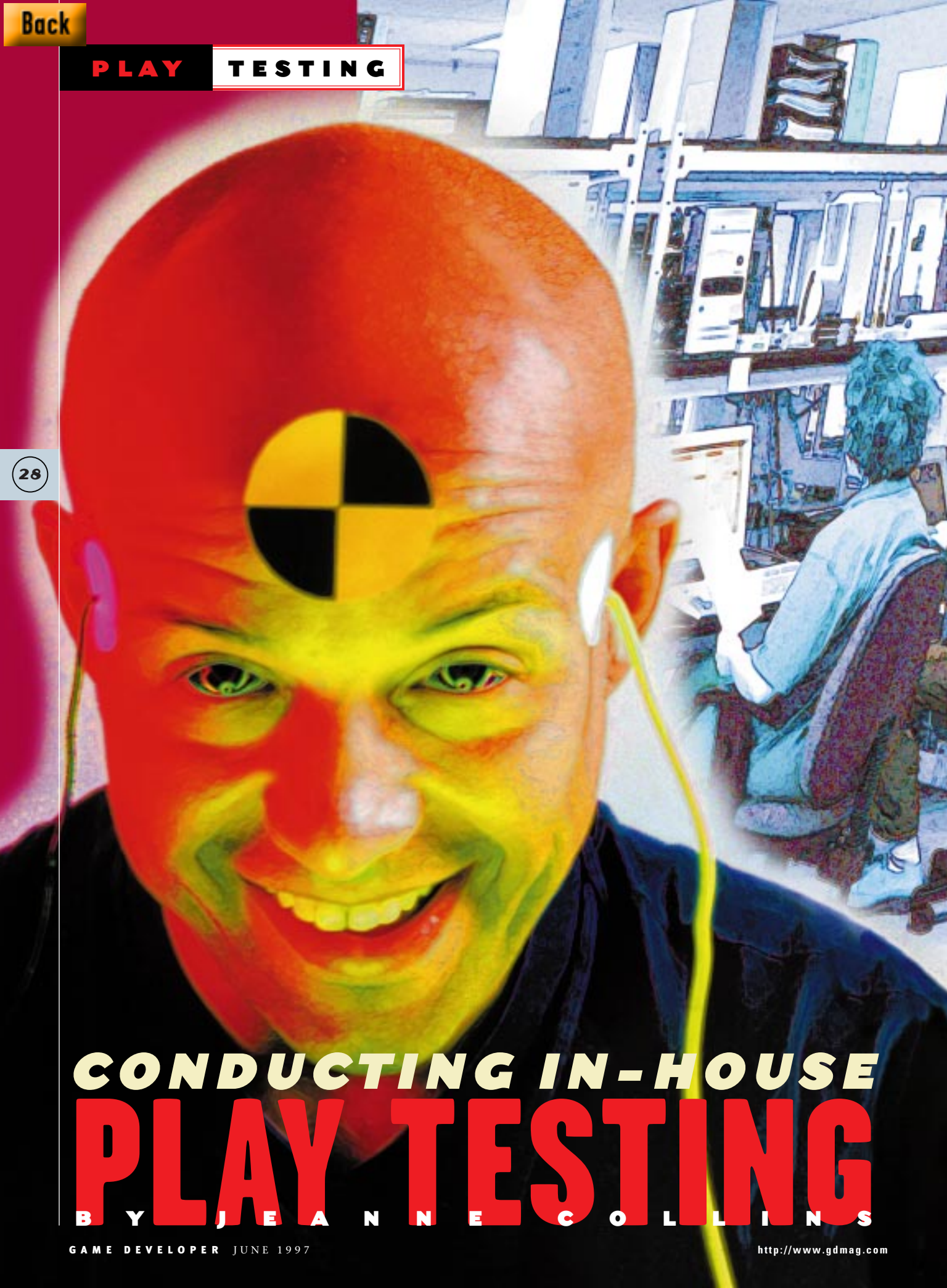
Postlude

That's it. With the information in this series, you should be able to add much more believable physics to your games and give the user a more immersive experience. However, you're far from done. Here are just some of the features we haven't covered:

- Contact. Our objects currently can't rest on the ground, which is pretty vital for a real game engine.
- Multiple simultaneous collision points. If you drop a box flat onto the ground, all four corners should hit at the same time.
- Modeling friction during contact and collision.
- Collision detection.
- Joints for articulated figures.
- Control for physically based creatures. Animation loops and simulation don't necessarily get along very well, so how to control creatures in a physically simulated environment is a huge issue.
- Numerical analysis. We covered the bare minimum needed to get our integrator running, but our Euler integrator probably won't do for a production-quality simulator. Numerical analysis is the study of how to implement all of these equations on the computer.

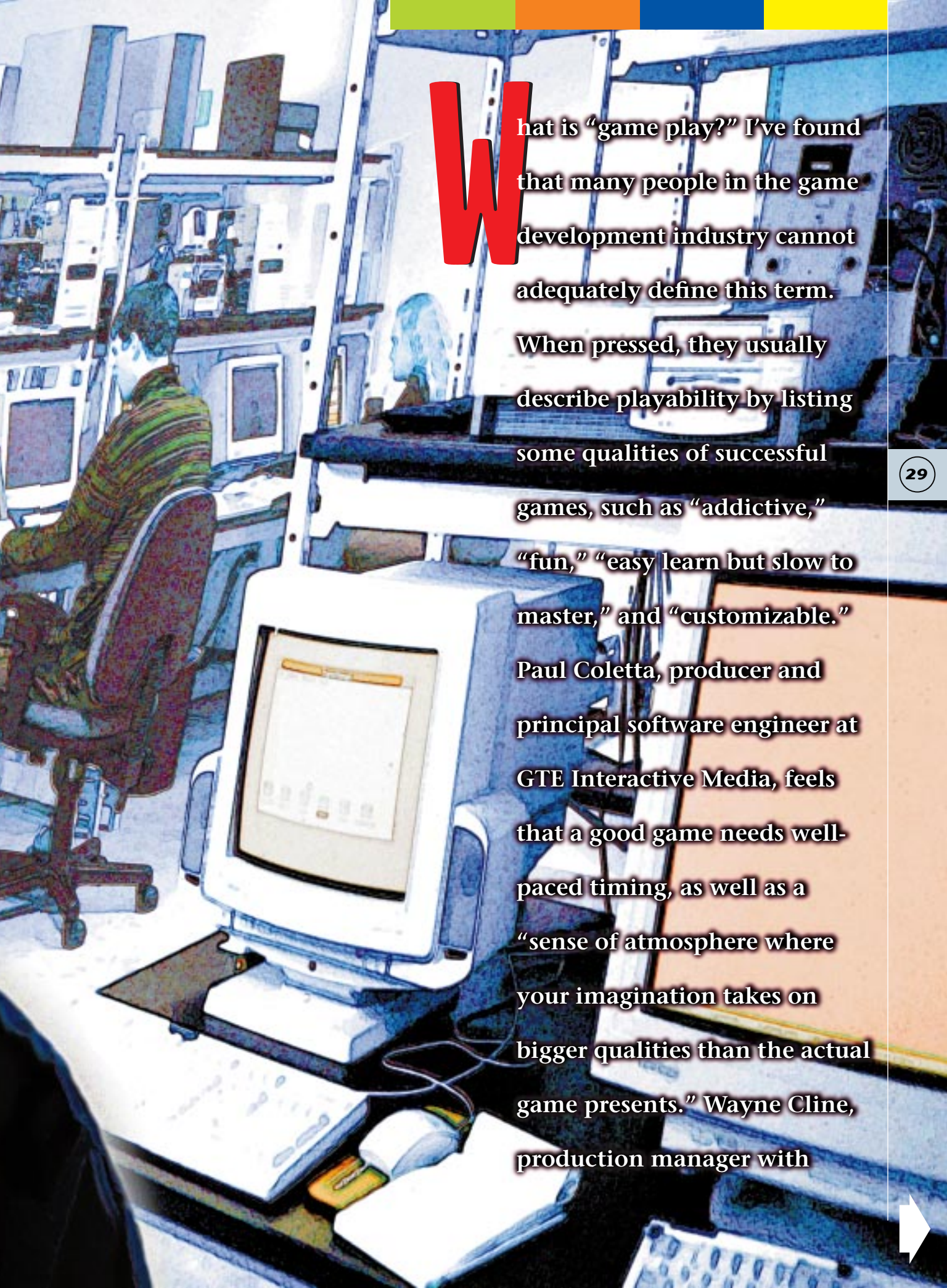
As you can see, there is a ton of physics out there to learn. We're in the dark ages of physical simulation in games at this point, and the material in these articles is just enough to get you started learning. So go read the references on my web site (http://ourworld.compuserve.com/homepages/checker), and get to work! ■

Chris Hecker's company, definition six incorporated, is putting its money where his mouth is by basing its first game on some pretty stoked physics. If the e-mail he's received during this physics series is any indication, lots of other companies are trying to do the same thing, so the next generation of games should finally start pushing the physics envelope in some interesting ways. Let him know how you're using physics at checker@bix.com.



CONDUCTING IN-HOUSE PLAY TESTING

BY JEANNE COLLINS



What is "game play?" I've found that many people in the game development industry cannot adequately define this term. When pressed, they usually describe playability by listing some qualities of successful games, such as "addictive," "fun," "easy learn but slow to master," and "customizable." Paul Coletta, producer and principal software engineer at GTE Interactive Media, feels that a good game needs well-paced timing, as well as a "sense of atmosphere where your imagination takes on bigger qualities than the actual game presents." Wayne Cline, production manager with

LucasArts, says that game play is "an unknown quantity that we're all trying to know." In short, there is no definitive magic formula for good game play.

Just because this quality is elusive, however, doesn't mean that it should be shoved aside as irrelevant. The ultimate responsibility for great game play usually rests in the hands of the producer. Still, before the code is written, the designer is often the person most concerned with "playability." More often than not these days, the producer and the designer aren't the same person. Games can be tweaked, improved, and enhanced during the testing phase, but if the game's basic design is flawed, it's already too late.

A good design should directly address components that allow the flexibility of altering game play. One of the best examples of such a component was designed and built by programmer Andy Caldwell (now with Screaming Pink Inc.). Caldwell's belief that good tweaking tools "relieve the programmer's time to work on programming while other people tweak" paid off in Street Hockey '95, a 16-bit multitap SNES game that unfortunately was undermarketed — it had great game play. Caldwell built the player attributes into a table structure and gave the testers access to the table. Each tester was assigned responsibility for the game play of certain characters. The designer and producer determined what each character's strength should be, and no other characters could be tweaked higher in that catagory. The testers had the ability to open up the table and change each character's attributes for shot accuracy, blocking ability, and so on. The testers then fed their improved attributes to the producer, who made sure that the testers weren't making "mega" players. The end result was that the programmer was able to concentrate on bug chasing and code performance improvements while the testers tweaked. Because of this tool, the game came in on time, under budget, and passed through Nintendo's approval system on the first pass — everyone on the team was empowered to do what they do best.

Starting and Controlling the Test Process

Play testing is usually accomplished in one of two ways: bringing in consumers (temporary play testers) and observing them while they use the product, or sending out beta copies of the game and eliciting feedback via a questionaire. Because conducting a wide-ranging beta test over the Internet is an article in itself, I'll only discuss in-house testing here. However, I do want to note that last fall I successfully used Internet Relay Chat (IRC) to conduct question and answer sessions with my external beta testers.

Conducting in-house play testing requires formal observation of temporary play testers playing the game over the course of several days. This type of testing shouldn't be confused with focus testing, which is conducted by your marketing team. The main purpose of in-house play testing is to put the game into the hands of each player and obtain individual feedback; marketing focus tests usually consist of showing the game to a group and obtaining group feedback. Sometimes people from an earlier marketing focus test might be invited back as temporary play testers, but usually these positions are filled through a variety of sources, such as recruiting friends of full-time testers, distributing flyers on local college campuses or at local arcades, posting notices on local Internet gaming bulletin boards, or advertising in local computer publications, such as The Computer-Edge in San Diego. Occasionally, good candidates can be found through temporary agencies, but most people don't boast of their gaming skills on résumés or job applications.

Wherever you decide to look for testers, make sure that you interview everyone before you hire anyone. Question interviewees about what types of games they most like to play. Don't hire somebody who only plays sports games to play test an RPG unless you want this individual to be one of those few purposefully hired to be unfamiliar with the genre.

The timing of play testing needs to be planned carefully. The game needs to be stable enough that the play tester doesn't spend too much time noting operational bugs, yet immature enough that effective changes can still be made to it. A minimum of one week's employment should be promised with the possibility of more. Since the hours that some play testers are available can vary, plan on double or late shifts for the regular testing staff during the weeks of play testing so as to accommodate those testers' schedules that only permit evening participation.

The ratio of temporary play testers to full-time staff testers monitoring them should be no less than 1:1. Each staff tester should always be observing, answering questions, and noting the temporary play testers' questions. Here are some key things for staff members to look out for:

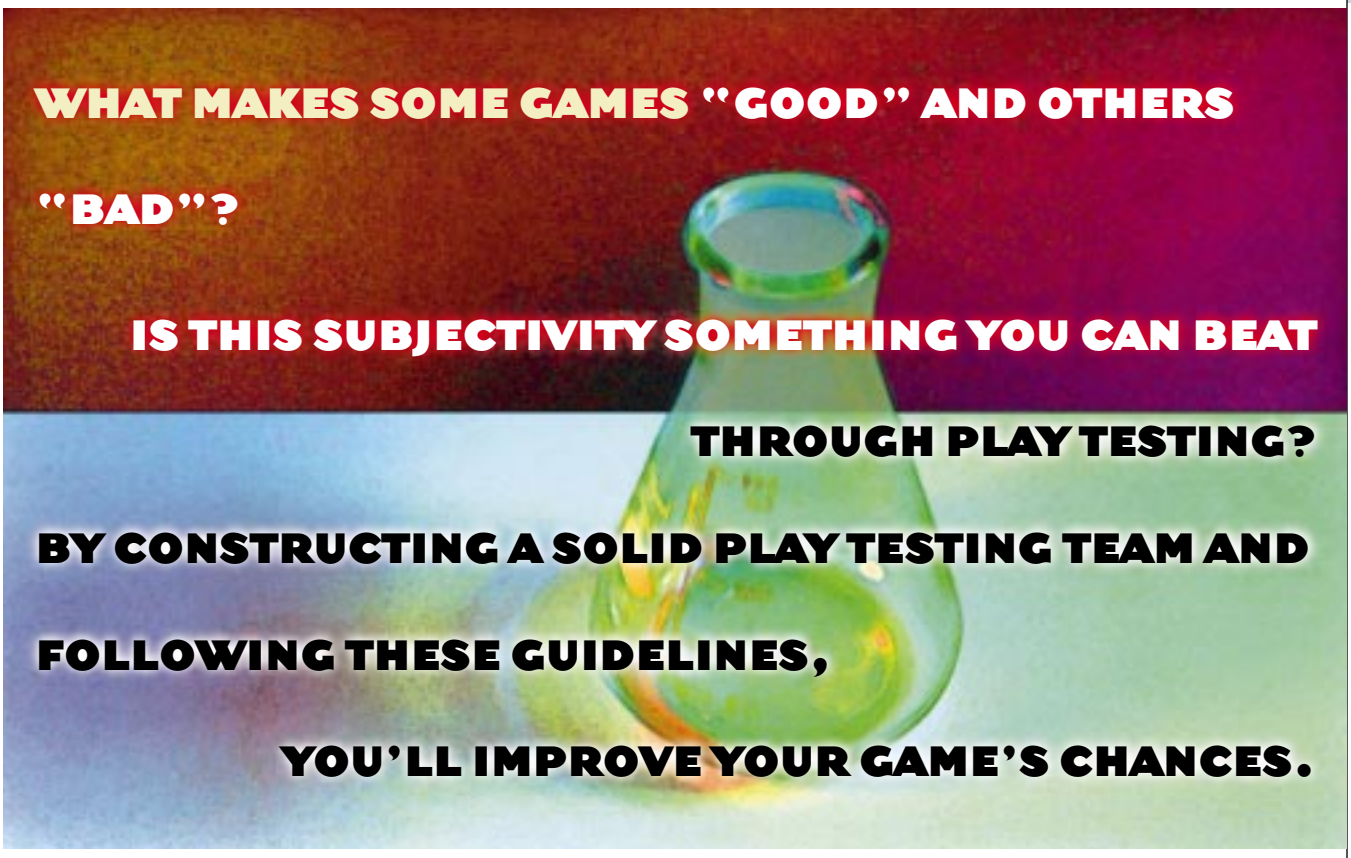
- Where do play testers seem to get stuck and ask for help from the staff? The staff testers working with the play testers need to rate each individual based upon their game skills. Although somewhat subjective, if one play tester can't even get the game installed and everyone else can, it would appear that this particular play tester doesn't posess adequate skills for the job. However, don't let this discourage you. Not everyone you bring in is going to live up to expectations.
- What kinds of features do the play testers have the most questions about? In the case of a sports game, set the game at the shortest playing time possible so that an entire game can be played in an hour or so. In the case of graphical adventure games that have a variety of different environments, be sure to spread the play testing across those various environments. Be sure coverage for the whole game — and not just the first part of the game's experience — is included in play testing. If there is a bonus environment that players can only get to after solving all the puzzles in other environments, provide shortcuts, jump codes, or previously saved games so that testers can jump to that bonus environment without having to solve everything else. Otherwise, what should be the best part of the game could turn out to be weak and bug laden.
- Do play testers get frustrated with the game easily? How closely does their frustration level relate to their skill level? Benchmarks need to be established prior to bringing in the play testers; additional benchmarks will be

added to as testing proceeds to measure key aspects of play testing. If the game uses puzzles, establish a minimum and a maximum amount of time for the play testers to solve each puzzle. If nobody can solve a certain puzzle in the expected minimum amount of time, don't stop the clock — let play testers continue until the maximum amount of time has expired. Find out if players really want to solve the puzzle or are becoming angry by their inability to solve it.

- Do play testers like the game? If they like the game, they'll be able to cite specific instances in the game that they liked or enjoyed. If they're bluffing, most likely they'll be unable to say any more than, "I just liked it."
- Are they complaining about the same things that earlier testers had noted in suggestion bug reports? If the play testers echo sentiments made during the earlier staff testing phase, and the items criticized were not fixed or changed, not enough attention has been paid to staff testers. These bugs will haunt you in product reviews after the game's been released.
- How long before the play testers become as bored with the game as the staff testers? A good testing schedule includes a lunch break after four hours and at least one 15-minute break every two hours. If the testers want to talk too much or need to take too many breaks, it could indicate that they are hitting the boredom stage. After a week of play testing (or at some other significant break during play testing), the play testers and their staff leaders should hold a group session to discuss the game. Prior to that, discussion between testers needs to be kept to a minimum so as not to alter opinions. During testing, the testers should be observed only — the producer and other "vested interests" shouldn't engage the testers in conversation — other than to ask questions — lest the testers be tainted by that interaction as well.

Play Testing Goals

Play testing should provide the producer with as much information as possible for making the necessary game play tweaks. Testing needs to provide more information than just crash and lockup problems. The producer needs to hear opinions such as, "I think the game is boring because…." Bug reports should include a category for subjective feedback, perhaps in headings titled "Opinion" or "Comment." Remember, the testing



When you have a significant number of play testers begging to have a copy to take home with them, you know you have a winner on your hands. But what if everyone seems to dislike the game? At this point in the schedule, too much money has been spent to throw it all away. It's time for the quality assurance (QA) manager to call a strategy meeting with testing, design, and production team members to review the usability test results.

The QA manager's primary objective is staffing each project with the right mix of play testing talent. Secondarily, the QA manager needs to assure that the information flow remains constant

— and pertinent — to the goals of the project. Often, QA managers' biggest obstacle is losing their best play testers to the production department.

Since turnover in the testing department can be fairly high, being able to identify and hire skilled testers is critical. The QA manager should look for excellent written and oral communication skills — the foremost prerequisite. I once made the mistake of hiring someone who couldn't write understandable bug reports. Even though this individual was a dedicated gamer with great ideas, it just didn't work out because this person couldn't communicate well.

Beyond communication skills, it helps for the tester to have a variety of experience in your game's genre. Also, throw in a few testers who know little or nothing about the genre, as this will broaden the insight you'll obtain about your title. Testers with less genre experience are often the ones who question the interface and yield improvements in areas where genre experts take things for granted.

The QA manager and the producer together need to choreograph a system of information sharing that will best help the project succeed. If you ask a tester and a producer why a game doesn't have good game play, you're liable to get two totally different answers. According to Paul Coletta, when testing says some aspect of the game is "wrong," the producer needs to interpret and evaluate whether that which is "wrong" affects the game's fun, pacing, or addictive qualities. Wayne Cline adds that the producer's biggest task is looking at testing reports and figuring out what will make the most impact on the game with the least disruption of the schedule.

The Dos and Don'ts of Managing Play Testing

DON'T BE DEFENSIVE ABOUT CRITICISM. Some producers get too defensive about their game design and concept, and they miss out on the best evaluations testing can give. Every effort should be made to make the testers feel that their opinions are important. Otherwise, they might fail to convey that one comment that could make or break the playability of a game, simply because they feel that their opinions don't matter or that they'll offend someone by giving honest feedback.

On the other hand, there will always be testers who can't say anything nice and advocate an entire revamp of the game. (Hopefully, the game didn't get that far in production if it really is that bad.) Don't put up your defenses too quickly, and try not to take these comments as insults. Glean as much information as you can from these testers.

QA managers should instruct testers to be specific when wording their feedback about a game. For instance, my favorite bug report was one where the tester stated, "The pencil sucks." This was in reference to a puzzle in a graphical adventure game where the player needed to move a piece of paper over a rock and rub a pencil on it to get the clue. The real problem was that the pencil was not easily manipulated to do the rubbing. Had the tester been more specific, time wouldn't have been spent trying to decypher this cryptic comment and the problem would have been solved more quickly.

STAND BEHIND OPINIONS. Testers should be taught to stick to their opinions, even if the producer tries to dissuade them from logging bug reports containing negative feedback. Some producers will go to great lengths to get their game through testing, but it's vital that the testing group report all issues they feel are important. Training testers to stick by their guns in the face of a direct challenge doesn't mean allowing them to become hostile. Testers who aren't perceived as thoughtful and helpful will get little cooperation from developers, ruining their chances to provide enough information or obtain enough support to do good work. According to James Bach, chief engineer with ST Labs, "Testers should be taught to give information, both positive and negative, without worrying about how developers will react to it." Furthermore, James advocates teaching testers that "the whole team owns quality, not just them. Testing is a process of revealing information that helps to make good decisions."

ENCOURAGE ESPRIT DE TESTING CORPS. Naturally, the size of a testing group should correlate to the number of games the group is expected to test at once. Full-time testing teams generally consist of at least one lead, one assistant lead, and three to six full-time testers, depending on the type and complexity of the project.

Full-time testers need to have a sense of community as a testing group, and should have a dedicated testing lab. Testers need to be located together in an area that promotes communication and cross-training between testers, particularly in the games industry, where few testers are actually trained in software testing methodologies, and most of their training is obtained on the job. Physically locating testers with the project developers they are assigned to —

and not with their fellow testers — could (and often does) hinder their objectivity. This doesn't mean that testers shouldn't have offices, just that their offices should be located near other testers. To counteract this separatism, testers (and particularly lead testers) need to be trained to work hard at developing strong communication with the developers whose products they are testing. They need to understand the basic architecture of the product they are testing to better find the bugs.

Ideally this community room will have all the necessary testing hardware. It can also double as a place to observe outside testers. A synergy of learning, communication, and discussion takes place in this setup. It promotes gameplay-oriented comments and critique.

MIX UP THE HARDWARE. Each project needs to be experienced on the minimum hardware configuration, as well as the closest thing possible to the maximum configuration and everything in between. The majority of testing needs to be conducted on the minimum configuration, because that is the promise to the customer. It's somewhat ghastly to see both "minimum" and "recommended" specifications on product boxes these days. What this dichotomy usually means is that the game will run on the minimum configuration, but if you want a decent experience, your machine had better have the recommended configuration. The difference between the two is causing a lot of unhappiness with customers.

Don't skimp on high-end testing either. Believe it or not, bugs can be found on the hottest machines around. I worked on one game that tested perfectly on the minimum specification, yet when customers attempted to install it on a machine that had 64MB RAM, the installer indicated that not enough memory was available to install the game. As it turned out, the game was looking for was 8MB RAM, and it only looked at the last digit. So it only installed on 8MB machines.

KEEP THE EYES FRESH. When staff testers look at nothing but the project to which they are assigned for weeks and weeks on end, they become blind to problems that they might otherwise notice. Therefore, it's useful to move testers around to other projects every now and then to gain a "fresh set of eyes." Sometimes, staff testers for one project can be used as temporary play testers for other projects. This is another reason for locating testers in a community area, rather than spreading them out all over a facility.

DISCOURAGE LEGACIES. How often have we caught ourselves passing down a project legacy to new testers? By "project legacy," I mean the harmful folklore used as justification for not solving an often-cited problem. For instance, one project I worked on spanned four CD-ROMs. Each time the tester started up the game, she needed to insert disk one into the drive, then swap to a second disk to resume play where she had left off. The reason she had to endure this disk swapping hassle (so the "pat" answer goes) was that correcting it would require an engine fix, and the engine "couldn't be changed." But making a change to the engine was possible; it was just that the programmer didn't want to do it, the producer didn't insist on it, and testers didn't make an issue out of the problem. We all had passed down the legacy that the engine couldn't be changed. A bug report for this problem was never even written, so when weekly meetings were held to review the reports, it wasn't ever discussed formally. Simply put, because of this "legacy," we had our blinders on when it came to that problem. Of course, this product's number one complaint once it went to market was the disk swapping issue.

OBSERVE YOUR TESTERS. The best producers spend time in the test lab — listening, not talking. They listen to the testers and they strive to derive and implement game play abstracts from the testers' concrete comments. As Cline says, "We know we have a good game if the testers are enthusiastic after weeks and weeks of play." However, I have seen producers who spend too much time with the testers. Often in these situations, each time a tester critiques an aspect of the game, the producer explains or defends why it is the way it is. The tester doesn't write up the problem because he believes it can't (or won't) be changed. Thus, new project legacies are born. Producers need to interpret and consider — not rationalize — any issues raised by the testers' comments.

REWARD YOUR TESTERS. Everyone works better and harder if they believe their hard work will be rewarded. To some staff testers, that reward might be recognition. To others, cold hard cash. Since the varieties are about as abundant as the number of people on staff, it is often difficult to reward everyone adequately. Some of the best (and most difficult) rewards include: recommending a tester for a promotion in recognition of a job well done, supporting a deserving tester when he or she applies for another job in the company (representing a step up the ladder), and recommending a tester for monetary bonuses. One of the easiest rewards is to spring for a pizza lunch and have a lunchtime game tournament playing the latest hot title whenever specific weekly goals for testing teams are met. Over the last couple of years, lunchtime tournament favorites in my shop have included DESCENT, DUKE NUKEM, and DIABLO competitions.

MAKE TESTERS AWARE OF THE COMPETITION. Make time for testers to review and analyze competitive products that are similar in nature to the one that they're expected to be testing. Make your testers the experts on the genre! Not only will you get better information from the testers, they'll appreciate the chance to play another game.

It All Boils Down To Teamwork

It's difficult to achieve that delicate balance between developers and testers during play testing. The guidelines addressed here don't encompass everything a game developer or publisher might want to do to test game play, but they're a place to start. The most important aspect of successful play testing is encouraging teamwork among the testers and developers. Listen to the testers, create an environment that is pleasant to work in, continually learn more about the craft, and stay fresh and honest. Play testing can be an ordeal, but when testers and developers work together, games ship on schedule, under budget, and with great game play.

Jeanne Collins is a quality assurance manager at GTE's Intelligent Network Services Group. She is sometimes referred to as a "self-proclaimed evangelist for quality assurance in the gaming industry" and chairs sigTEST, a CGDA Affiliate group. *Jeanne and* Game Developer *would like to thank ST Labs for the testing lab photo on pages 28-29.*

REAL-TIME PATHFINDING FOR MULTIPLE OBJECTS

Most game developers eventually face the problem of finding a path for a large number of objects in real time. Whether you're building a role-playing game or a real-time strategy game, there are times when you must create the illusion that multiple objects are more or less intelligent. At some point, this boils down to having those objects move from point A to point B. Because the objective is to simulate intelligence, both teleportation and ghost travel (walking through obstacles rather than around them) are out of the question. In theory, making a large group of objects

BY SWEN VINCKE

move around obstacles isn't very difficult. Making it happen in real time, however, can create quite a problem, especially when you're dealing with a large number of objects and moving obstacles. Pathfinding algorithms aren't always fast, and having to call them regularly can affect your game's performance.

To move a large number of objects in real time, you need a good, fast pathfinding algorithm. The algorithm has to find a path that is as close to optimal as possible, and it can't affect the frame rate when it is used repeatedly. Furthermore, you need a method to resolve object collisions, since the movement of a lot of objects in a relatively small space is bound to cause collisions. In this article, I examine the A* algorithm, which can be fast once it's been optimized. Then I look at several ways to avoid object collision. I've developed a small program in C called Move, which illustrates the methods that I describe. Move is available for download from the *Game Developer* web site.

A* Revisited

In the October/November 1996 issue of *Game Developer*, Bryan Stout presented an overview of several pathfinding algorithms ("Smart Moves: Intelligent Pathfinding," pp. 28-35). I'll assume that you have read his article (which is also available on the *Game Developer* web site) and are familiar with the A* algorithm and the associated terminology (Figure 1). Remember that the A* algorithm defines the heuristic evaluation function f'(n), which estimates the merits of generating a node n. The function f'(n) is an approximation of f(n), which yields the true merit of generating a node n. The estimate f'(n) is usually calculated by adding g(n) (which is the actual cost of a path between the initial node and the node n) and h'(n) (which is an estimate of h(n), the cost of getting from the node n to the goal node). A* uses these functions to guide its search; you could say that they represent the "knowledge" of the algorithm. One of the more important properties of A* is that if h'(n) rarely overestimates h(n), then A* will generate a path that rarely overestimates the optimal solution. In fact, you can generally trust A* to generate the fewest nodes necessary to find a solution.

FIGURE 1. *The components of A**

Initial Node (starting point)	Intermediate Point (node n)		Goal node
Optimal Route f(n)=	g(n)	+	h(n)
Estimated Route f'(n)=	g(n)	+	h'(n)

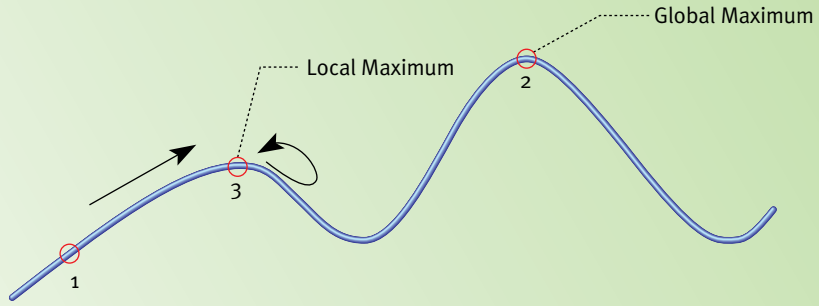
Unfortunately, A* isn't quite adequate for use as the motor of a real-time object movement system — it doesn't live up to expectations. This discrepancy can be attributed to the three inner loops at the core of the A* algorithm. Two of these loops check to see if a node (think of nodes as squares on a piece of graph paper, and the paper as the search space) has already been generated on either the Open (unchecked nodes) or Closed (previously checked nodes) lists. The third loop is necessary to pick the best node from the Open list. Since optimization of the third loop was discussed in Stout's article already, I will instead show how to eliminate the first two loops. First, however, we need to look at a more general and strategic problem.

The Disease of Heuristic Algorithms

As good as A* may be, it suffers from the same problem as every other heuristic search algorithm: the local extreme problem. A local extreme is a point in the search space that is either better or worse than any other point near it. By "better or worse," I mean that these points, when applied to the heuristic evaluation function, either maximize or minimize that heuristic evaluation function in a set neighbourhood. What's frustrating is that there often are a lot more local minima and maxima than there are global optima (usually there is only one global optimum). This is the reason that a heuristic evaluation function often searches in the wrong direction: It is following a local extreme. The only thing that can prevent this is a perfect heuristic, but these are hard to come by and generally require that you calculate the global optimal solution. Figure 2 illustrates the problem of local extremes.

Heuristic search algorithms that guarantee a solution (as A* does) work on the basis that they can get themselves out of local extremes to find the global. These algorithms work slowly because there is always the possibility that they were actually going towards the global solution. To illustrate this let's have a look at a heuristic search algorithm that doesn't guarantee a solution, such as the steepest-ascent hill climbing algorithm in Figure 2.

FIGURE 2. *The Steepest Ascent Hill-Climbing Algorithm.*



Starting from the initial node (1), the algorithm generates all the possible successor nodes and then selects the most promising node as the next node in the path to the goal node. It repeats this process until the goal node (2) is found or no better node than the current node is available for selection (in which case the algorithm reports failure). It's obvious that when this algorithm gets caught at a local extreme (3), it will fail. It has no built-in mechanisms to navigate past local extremes, and therefore gets tricked easily. On the other hand, its singlemindedness makes it one of the most efficient search algorithms available once it smells the global optimum.

FIGURE 3. *The bidirectional A* algorithm.*

1. Start with 2 OPEN lists, OPEN(0) and OPEN(1), which contain the start node and the goal node respectively. Set the g, h′, and f′ values. Set CLOSED(0) and CLOSED(1) to the empty list.
2. Until a solution is found, or an OPEN list is empty, do the following:
   For each OPEN list pick BESTNODE and perform the usual A* steps with it. However, before checking if a succesor of BESTNODE is already on OPEN or CLOSED, check if the succesor was already generated in the other search. If that is the case, a solution was found.

What we need is an algorithm that can get out of a local extreme without losing the ability to rapidly close in on a target. This is exactly what A* does. The h′(n) function allows the algorithm to head steadily towards an extreme, and the g(n) function makes sure that the algorithm is aware that every newly generated node comes at a price. Suppose we are in a search space where there is one local extreme L1 and and one global extreme G1. If the algorithm starts following L1, eventually the estimated cost f′(n) of the path leading to L1 will become larger than the f′(n) value of some arbitrary node. Thus, the algorithm will start expanding that other node. However, once it has expanded that node, the algorithm could well jump back to the path being expanded towards L1. Then, when a few additional nodes on that path are generated, the algorithm might jump back again, and so on. At some point, a node will be expanded that is more optimal than L1, and the algorithm will have gotten out of the local extreme. (Actually, in case of A*, the algorithm was never "in" the local extreme. It was just focusing its search there.) The problem is that getting out of a local extreme can take some time. When designing your movement system, you need to anticipate the situation I sketched here. It happens frequently, and there is little that you can do about it since otherwise, the algorithm wouldn't work anymore.

The No-Path Problem

A* gets into trouble when no solution exists. This possibility is often overlooked when discussing the algorithm, although any succesful real-time object moving system must be able to deal with the no-path problem. When there is no valid path, the search algorithm usually goes through the entire search space (essentially what a breadth-first search does). Since the breadth-first searches are notoriously slow, this can be a major problem. Unfortunately, there aren't many alternatives. Most often, developers cut off the search at a certain depth or stop the search when more than X nodes have been searched (which is better than cutting off at some arbitrary depth). Still, both options limit the algorithm's search horizon and, therefore, reduce its chances of finding a path.

What we need is an algorithm that recognizes, in a less costly fashion than a complete search, that there is no solution. Such algorithms exist, and in general rely on precomputed lookup tables. They are useless for our purposes, however, as we are looking to find paths in dynamic search spaces that can change regularly. Enter the bidirectional A* algorithm.

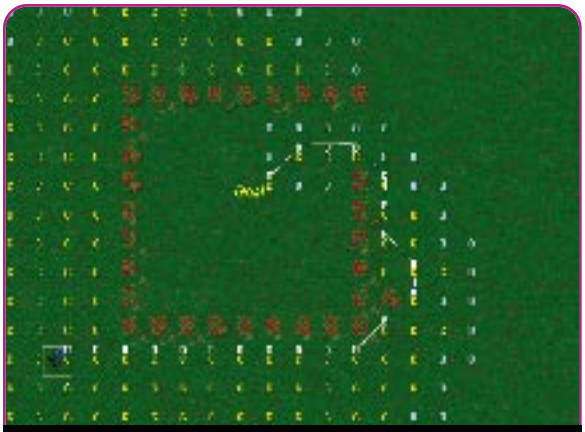
The bidirectional A* algorithm is shown in Figure 3. As you can see, it is basically the same as the A* algorithm; the primary difference is that the path is searched simultaneously from the start node and the goal node. If there is no solution to a path, bidirectional A* detects this problem more quickly than A*. Figure 4 shows the bidirectional algorithm at work in a no-path situation.

The risk of bidirectional A* is that the two simultaneous searches might miss each other (like ships passing in the night), causing the computer to double its work. Another problem is that for bidirectional A* to be efficient, a constant time function has to check whether a node belongs to one of the two searches. Hashing can easily take care of this.

Since bidirectional A* is a heuristic search algorithm, it (like A*) has problems navigating past local extremes. In the worst case, it deteriorates into a bidirectional breadth-first search. Fortunately, this situation is rare.

Optimizing A*

One of the bottlenecks of A* and bidirectional A* is caused by checking whether a node is already on the Open list or on the Closed list. If Open and Closed are linked lists, this checking can bog the system down tremendously. Fortunately, you can easily pare the search down (though at the expense of taking up additional memory) by storing the A* nodes directly in search space. When you define your search space (usually a matrix for real-time strategy games), reserve some memory for the A* data fields (typically a parent field, a link field, and a cost field) in the structure that describes one node. You'll also need a search ID field to determine if the node belongs to the current search, and some flags to determine whether it is on the Open or Closed list. Then, when you would normally expand the node by generating it and performing the standard A* checks, you instead use the node that is in the search space, eliminating the need to traverse the



This is a screenshot from the demonstration program. C means that the node was on the Closed list in A, O means the node was on the Open list, and B means that the node belongs to the final path. Note how several nodes that were on closed deviate from the path because of the problem of local extremes.*

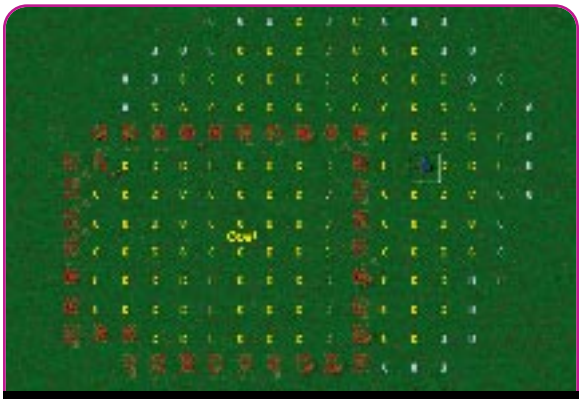


FIGURE 4. *The unit tried to enter the square but there is no entrance. If this had been the regular A* algorithm it would have tried to search the entire map for an entrance, but since bidirectional A* starts from the two sides, the no-path situation was detected rather quickly.*

Open and Closed lists. In my demonstration program (Figure 4), the searches that use this method are called "optimized."

Storing the A* data directly in search space also lets you easily implement frame/search distribution. "Frame/search distribution" is the term I use for the process of allowing the search algorithm to stop at any time necessary to do something else, then come back later to continue the search. The typical reason for stopping the algorithm is to let it prepare data to draw the next video frame or to draw the next frame. When I first implemented frame/search distribution, I couldn't change the search space (by creating new obstacles, for instance) when I paused the search, since the generated path might be incorrect. I could only change the nodes that weren't in the search yet, a condition that you can check for using the search ID field. Later, I will show that, in fact, you don't always have to be concerned about modifying obstacle data in nodes that have already been touched by the search algorithm. In my demonstration program, the searches that use frame/search distribution are called "distributed."

Let me offer two comments about the A* algorithm. First, you can forego the algorithm's propagation when it finds a better path on the Closed list. Of course, A* is no longer guaranteed to find the optimal path. In general, however, you'll find that without this propagation, the algorithm yields good enough results and only seldom makes obvious misses. This is even more true of bidirectional A*.

Second, try to limit the instances in which h'(n) overestimates h(n), and make sure that h'(n) reflects, as accurately as possible, the sum of the node traversal costs necessary to reach the goal. A fault in calculating h'(n) can severely drain the performance of A*. Spend time testing several values — it's worth it. Remember that when h'(n) overestimates h(n), you're simply doing what is called the "A search" (which is a breadth-first search).

Removing Interobject Collisions

It's a sad fact that the complexity of pathfinding increases considerably when the search space is changing all the time. There is no guarantee that a path that was optimal when it was first calculated will remain optimal once the search space changes. There isn't even any guarantee that the path will remain valid. The truth of the matter is that you would actually have to recalculate the entire path for every object every time there is a change in search space. If it's not immediately clear that this is a definite no-no, I urge you to try my demonstration program, and set the pathfinding method to GREEDY. (No reference to the greedy algorithm is intended. I call it greedy because it eats processor cycles.) Only in a turn-based game could this method be valuable. Let's look at several methods of dealing with interobject collisions.

Path Locking

Path locking is very straightforward. Every object calculates its path, flags the positions taken by that path as unavailable to the other objects, and gradually releases those "locked" positions as it moves forward in its path. This approach has its advantages and its disadvantages. Among the advantages are its speed, its ease of implementation, and the fact that the collisions are removed at the same time the path is calculated. Unfortunately, it's also a bit stupid. As more and more objects start to move, the search space gets more complex, and the paths taken to reach a goal become increasingly longer and more unnatural (Figure 5). In some cases, it can quickly become impossible to reach the goal. Another problem is that if the player decides to change the destinations of some of the moving units, some of the paths taken can meander terribly, even if there is a straight light from start to destination. To see these shortcomings, put several units together in my demonstration program and order them to an arbitrary destination. If these were troops in a live battle situation, you'd want the enemy to kill them. Still, in situations where objects are scattered sparsely over the search space, path locking might not be such a bad idea.

Cut-Path Locking

You can achieve better results with path locking when you limit the length of the occupied positions. This means that even if you've calculated a complete path, you only occupy a portion of this path. When an object reaches the end of the path that it occupies, it recalculates its path and again only occupies a portion of its path. It repeats this process until it reaches its destination. Figure 6 illus-

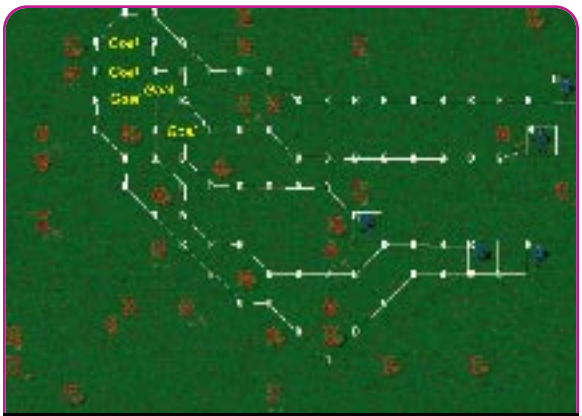


FIGURE 5. *The locked path method. Several objects are going to the same destination, and in doing so make the search space very complex. The paths are ridiculous.*

FIGURE 6. *Cut-path locking. Even though the unit knows the entire path, it only locks part of it and walks the part that it locked. Once the unit arrives at the end of the path it locked, it recalculates a path towards the final goal.*

trates cut-path locking. The advantage is that objects get better paths because the paths are more dynamic. The disadvantage is that you need more processor power to calculate your objects' movement.

If you consider implementing cut-path locking, you might consider altering the function that searches for a path in the following way. Once your search function is farther away from the start position than the maximum occupying length, don't bother with positions that are locked by other objects. This decreases the complexity of the search space. The drawback is that you start falling into the trap of limited search horizons. For instance, consider that several objects might plan to obstruct the route of another object. If you don't check for this kind of potential problem in the search algorithm, you won't have any way of preventing it from happening. Note also that if you limit the length of the occupied positions to one node, you have the GREEDY object-collision removal algorithm, as paths will have to be recalculated all the time.

Step-Based Evaluation

This is a very natural approach. Every object selects a path that doesn't take other moving objects into consideration. When a moving obstacle suddenly decides to block, that path there are two things an object can do. It can either wait for the obstacle to pass, or it can try to move around the obstacle (in which case, it recalculates its path, this time taking other objects into account). Of couse, the object could also stop moving altogether, but game players likely won't find that a sufficiently intelligent response — I'd rule that out as an alternative.

If the initial path doesn't take other moving objects into consideration, you can determine whether a path exists or not. If it doesn't exist, you needn't bother with trying to move the object anymore. If it does exist, however, and it is surrounded by other moving objects that are blocking it, you can take some sort of action to remove the block.

The decision as to whether to move or to wait is a hard nut to crack. Waiting, of course, is the least computationally expensive option, but if every object decides to wait, nothing will happen. On the other hand, moving can also be a catastrophe if the object is part of a larger group of objects trying to move through a choke point, such as a narrow bridge over a river. The search algorithm then behaves like the dreaded breadth-first search (searching every surrounding node in an attempt to get to the other side of the river).

This brings us to the interesting problem of internal ordering. It's possible that a path could be found for two objects A and B, only if object B starts before object A, but not if object A starts before object B. You can solve the problem of internal ordering using an algorithm such as the one presented in Figure 7, but my experience is that, for most games, this algorithm is overkill. You're probably much better off delaying object A and moving it a bit later. Just make sure you only delay A if in fact B is planning on moving. If B is idle, move around it, push it away, or take some other logical course of action.

Step-based evaluation works very well with frame/search distribution. Earlier, I stated that it isn't always necessary to modify obstacle data in nodes that have

FIGURE 7. *Object collision removal.*

```
1. For each object, do the following.
2. Is next node is clear?
   -If yes, go there and remove object from list.
   -If not,
      Is it because another object is standing there?
         -If no, it has to be because another object is en route to the node.  Take other
          move.  If no other move is available, mark object as delayed and remove it
          from list.
         -If yes, check if the blocking object is waiting for a move to be allowed.
            -If no, move is not allowed. Take other move and go to check. If no other
             move available, object becomes delayed and is removed from list.
            -If yes, are there other objects waiting for an allowable move and have
             they not been put on hold?
               -If yes, skip this object and proceed with other objects.
               -If no, the move is not allowed. Take another move and do step 2 again.
                If no other move is available, object becomes delayed and is removed
                from list.
```

Note that when a move is allowed, the node the object is going to becomes unavailable and the node it was on becomes free. If some objects can move faster from node to node than other objects, a node can be given a value indicating that it will be free in k amount of time and the object can be delayed by k before it actually makes its move. However, it has already marked its next node as being occupied.

Note also that in some cases it is better to wait than to take the other direction. We can use the f value of each direction and compare it to the f value of the preferred direction. If the difference is larger than some threshold T, we choose to delay rather than to take the other move. To prevent complete deadlocks, we can increase the threshold for every delay the object incurred up to a point where it will choose to take the other direction rather than stand still.

already been touched by the search algorithm; this is the case with step-based evaluation. Because this approach is an ad hoc method that deals with problems as they arise, it doesn't matter whether a path becomes invalid due to changes within the search space while the search is being conducted. Although the processor may do some extra work from time to time, you can continue moving objects that already have a path while a search is busy. This advantage should not be underestimated.

FIGURE 8. *The Real-Time A* algorithm.*

1. Set NODE to the initial start state.
2. Generate the successors of NODE, and quit if any of the successors are a goal state.
3. Estimate the value of each successor by performing a fixed-depth search starting at that successor using depth-first search. Pass the heuristic estimates up the search tree in such a way that the f value of each internal node is set to the minimum of the values of its children.
4. Set NODE to the successor with the lowest score, and move towards NODE. Store the old NODE in a table along with the heuristic score of the second-best successor. (This avoids deadlock.) If the node is ever generated again in step 2, simply look up the heuristic estimate in the table.
5. Go to step 2.

Final Thoughts

A lot more can be said about inter-object collisions, but unfortunately, not in the span of this one article. With the methods I have presented, you should be able to create an efficient real-time movement system. Note that the three object collision removal procedures I have presented operate under the assumption that the objects cannot communicate with each other. If objects are allowed to communicate (say that units have walkie-talkies), the locked path method suddenly becomes interesting, because then you can work with estimated time of arrivals. Examining the algorithms that take advantage of this knowledge, however, would take up another entire article.

Also note that my presented methods assume that an entire path is generated from start to destination. This doesn't have to be the case. For instance, you could use the real-time A* algorithm, which was not discussed here, but which I present in Figure 8. I advise using it only when you can't use the A* algorithm in its optimized form. Not generating the entire path from start to final destination means limiting the intelligence of your objects. ■

Swen Vincke is the overworked programmer behind THE LED WARS, *published by Ionos, and* THE LADY, THE MAGE, AND THE KNIGHT *to be published by Attic. He can be reached at lar@larian.com.*

DVD: A Game Developer's Survival Guide

The Digital Versatile Disc (DVD) is like an iceberg — a lot of marketing hype on top, and exciting, but challenging technology under the surface. The captain of the Titanic overlooked a seemingly innocent chunk of ice and regretted it. Likewise, if you ignore DVD's game potential because of its failure to live up to the initial publicity, you run the risk of sinking your future. In order to avoid such a catastrophe, here is some information necessary to start DVD development, including a review of the technology, an exploration of how DVD is being used by some game developers today and an examination of authoring and playback tools.

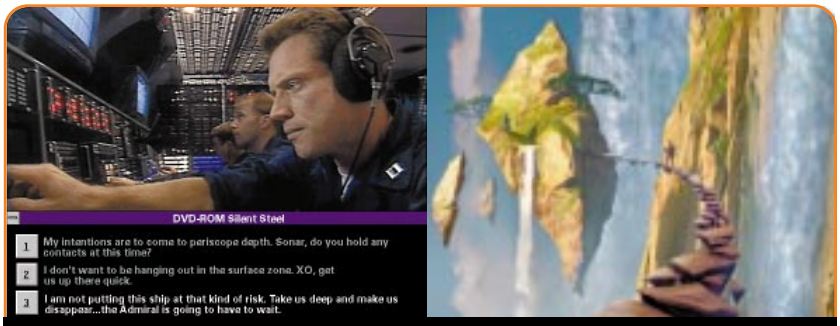
Although certain aspects of DVD technology (such as disc capacity and format) have been well publicized, many features are virtually unknown. This dearth of information is not accidental. The current cost to obtain the DVD 1.0 Specification is $5,000. While $5,000 may be an insignificant sum for a large conglomerate, some of us are hesitant to spend such a small fortune.

Like floppy disks, DVD has a variety of capacities. The majority of the early discs will use single-layer technology and can be either single- or double-sided (with capacities of 4.7GB and 9.4GB, respectively). In the future, more complex content will utilize single- or double-sided, dual-layer discs, which hold approximately 8.5GB per side.

DVD Compression Algorithms

DVD encompasses numerous formats, the most important being DVD-ROM and DVD Video. From a programmer's perspective, DVD-ROM can be considered simply a high-capacity CD-ROM. By contrast, DVD Video (or DVD Movie) enhances DVD-ROM by dictating how multimedia information is stored and played back from the disc.

Most DVD Video titles will use MPEG-2 for video compression/decompression, whereas MPEG-2 is optional for DVD-ROM. Although MPEG-2 has noticeably better picture quality than other compression schemes, it is extremely processor intensive (software decoding of a four megabit MPEG stream consumes 100% of a MMX Pentium, leaving no bandwidth for decompression of other DVD streams). As a result, DVD titles will require MPEG-2 hardware acceleration to enable playback for the foreseeable future.



Several game publishers have begun releasing DVD games, including Tsunami's SILENT STEEL (left) and Rocket Science's OBSIDIAN (right).

WHETHER DVD-ROM OR DVD VIDEO, THIS LURKING TECHNOLOGY COULD HAVE IMPORTANT IMPLICATIONS FOR YOUR FUTURE GAMES. TAKE THIS OPPORTUNITY TO MAKE AN INFORMED DECISION ON WHETHER OR NOT TO EXPLOIT DVD. by Linden de Carmo

In terms of audio, DVD Video uses Dolby's AC-3 audio compression technology. AC-3 was originally designed for theaters and the professional audio market, and provides multichannel surround sound without tricks (such as manipulating stereo digital audio data so that it appears to contain more than two channels). Each AC-3 stream contains six audio channels: left, center, and right channels for the front of the room, left and right surround channels, plus a sixth channel for extra bass sounds to reinforce crashes, eruptions, explosions, and so on. Unlike the other channels, the sixth channel may only contain sounds between 3Hz and 120Hz — as a result, it is nicknamed the ".1 channel" (.1 indicates a channel with limited frequency range). AC-3 is sometimes said to contain 5.1 channels of audio.

All North American DVD players must also be able to play uncompressed, linear Pulse Code Modulation (or PCM) streams. These streams have sample rates between 48-96kHz, and can be sampled at 16 or 24 bits. Players may optionally support MPEG-1 or MPEG-2 audio streams.

The Fledgling DVD Game Market

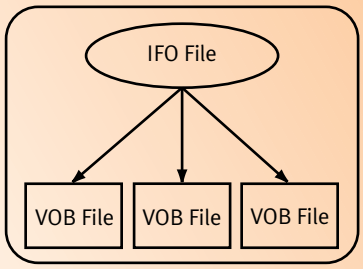
Although DVD is exciting technology, the immaturity of tools and questions about the initial market size have caused many game vendors to hesitate jumping into this arena. After all, cool technology doesn't pay the rent. However, a few cutting-edge developers have evaluated the DVD gaming market and believe it to be profitable. These vendors can be divided into the following categories: those who view DVD as a huge CD-ROM, those who are enhancing their existing CD products for DVD-ROM, and those who fully exploit DVD Video technology.

The first type of DVD game vendor uses DVD as a monstrous CD-ROM. Companies in this category transfer content from one or more CDs to a single DVD-ROM and then ship it. While this approach is clearly the most conservative, it allows access to DVD-ROM customers with minimal risk. It is also an especially effective technique for games that span several CD-ROMs, since players aren't annoyingly reminded to change the CD in the heat of a game play. One such product is OBSIDIAN by Rocket Science Games. The original CD-ROM version of OBSIDIAN is huge — it comprises five discs of QuickTime content. Kim Hilquist, OEM manager for Rocket Science, noted that not only has DVD eliminated the need to swap discs, but the increased speed of DVD-ROM drives enables smoother video playback.

The second type of DVD vendor enhances its existing games for DVD by utilizing MPEG-2 video and AC-3 audio. Xiphias, an edutainment developer, is one such vendor. Steve Kaplan, a project manager at Xiphias, is impressed by the technological potential of DVD. "For the first time, video quality [of a DVD product] does not distract the user. The picture quality is quite clear, and at some points, stunning," Kaplan says.

Although Kaplan is upbeat about DVD, he does point out some problems. First, the tools used to create DVD titles are very immature and can make title development laborious. For instance, in order to obtain movie content, Xiphias had to send video tapes to a third-party vendor to be converted into DVD's special .VOB file format. Since Xiphias uses their own custom navigation engine, they had to manually create navigation paths in the .VOB file.

FIGURE 1. *Illustration of a Video Title Set (or VTS). Each VTS is composed of IFO (or navigation) and VOB (data) files*



Another difficulty is the variation in performance and quality for MPEG-2 and AC-3 decoders. Since performance of hardware and software decompressors varies dramatically, game vendors must author content with data rates that are playable on a variety of decoders.

Like Xiphias, Westwood Studios also plans to soup up an existing CD title, COMMAND AND CONQUER, for DVD-ROM. Because the existing game uses two CDs, gamers get the immediate benefit of only needing one disc. Mike Sack, marketing director at Westwood, indicates that the company will also enhance this title with MPEG-2 video and AC-3 audio.

The final type of game vendor creates a true DVD title, utilizing all of the features in DVD Video. Tsunami Media is one such vendor, and they truly have been smitten with the potential of the DVD market. While most companies have serious reservations about DVD's revenue possibilities, Don Soper, a producer for Tsunami Media, believes this to be a lucrative market. As a result, Tsunami became one of the first vendors to release a game (SILENT STEEL) on DVD-ROM . The DVD-ROM version of SILENT STEEL is enhanced with AC-3 audio and MPEG-2 video and only requires one disc — as opposed to the four discs making up the CD-ROM version.

Now, Tsunami plans to leverage the multimedia enhancements in the DVD-ROM version of SILENT STEEL to create a DVD Video version of the game. Tsunami Media deliberately chose authoring tools that could generate either DVD-ROM or DVD Video titles. As a result, the DVD Video version of the game should have the look, feel and performance of the PC version, while retaining platform independence.

Soper indicates that the DVD Video version of SILENT STEEL will conform to the DVD 1.0 Specification and should work on any DVD-compliant player. When asked about programming restrictions imposed by DVD Video, Soper seems unfazed. "If you're creative, there's enough support to do a lot more things than one might think at first glance." He is also confident that the remote controls utilized by DVD players will be more than adequate for game play. This is significant since DVD remotes are slanted toward DVD menu input and button manipulation (remotes have a numeric keypad and menu shortcuts), with virtually no game-specific features such as a joystick control.

FIGURE 2. *The directory structure of DVD Video. This directory contains navigation (IFO) files and multimedia (VOB) files. XX indicates a two-digit VTS number and YY is a two-digit index number within the VTS.*

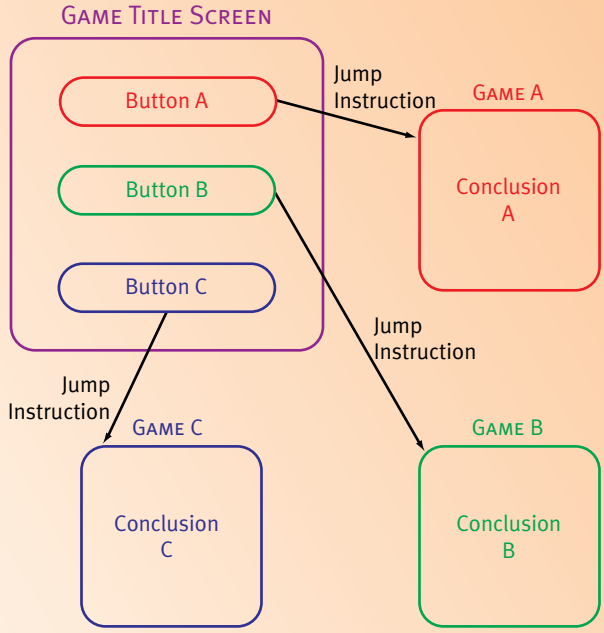
AudioTS
VideoTS
- **VIDEO_TS.IFO**
- *VIDEO_TS.VOB*
- *VTS_XX_YY.IFO*
- *VTS_XX_YY.VOB*

Items in bold must be present on every disc.
Items in italics are optional.

While Tsunami is upbeat about DVD, they realize that it has limitations for gaming. First, they have been affected by immature state of the title-creation tools. For example, debugging a title is an arduous process. A disc must be authored, sent to manufacturing, and then tested. Every time a bug is found, this cycle must be repeated.

A second restriction is the type of games that can be written in DVD Video. Although this technology excels at movie-oriented titles such as SILENT STEEL, Soper doubts that the current DVD specification can handle fast-action 3D-graphics games.

FIGURE 3. *Illustration of how buttons operate. Selection of a button causes the navigation engine to execute a command associated with the button. In this figure, the command causes the player to jump to a different scene in the game.*



The DVD Video Navigation Engine

Although DVD-ROM and DVD Video may share compression algorithms, DVD Video contains features not found on DVD-ROM. The most notable difference is that the content on DVD-ROM is platform specific, while DVD Video provides a platform-independent navigation engine for playing interactive movies (these movies potentially can be played on Windows 95, MacOS, and television-based consumer DVD players). This navigation engine requires a rigorous directory structure, which I'll explain briefly.

Every DVD Video disc must contain a VIDEO_TS directory, which contains only two types of files: .IFO and .VOB (Figure 1). These files are sorted by a DVD Video player to form Video Title Sets (VTS). A VTS is a grouping of all the files necessary to play a particular DVD Video title and is composed of one .IFO file and one or more .VOB files (Figure 2).

The .VOB extension is short for Video Object Set — it indicates that the file contains multimedia data. Many people are under the mistaken impression that .VOB files are equivalent to DVD Video. In reality, while it is possible to write a simple .VOB-only player, all of the interactive functionality in a .VOB file is abandoned when you don't play it with the associated .IFO file.

Both the Video Manager .IFO file and the VTS .IFO contain additional navigational data structures and a processor-independent interpreted language (sort of a miniature Java). These data structures are composed of the following objects: Program Chains, Part of Title, Programs, and Cells.

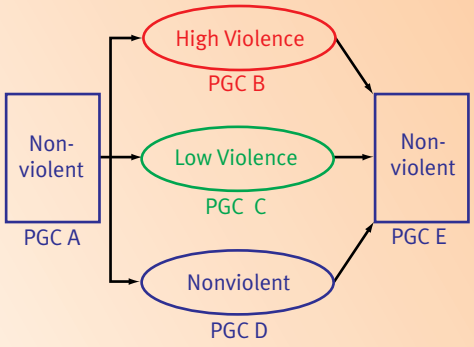
Program Chains (or PGCs) link related programs (or particular scenes) within a title. Their data structures govern how a given program plays. Simple titles may contain only one PGC. By contrast, multi-PGC titles containing two or more PGCs are used by complex discs requiring random access to a variety programs. A multi-PGC title can play programs linearly, randomly, or in shuffle mode.

Every program in a program chain is composed of elements called cells. Cells are the smallest navigational unit and tell the DVD player which portion of a .VOB file to decode.

Unlike program chains, which exist entirely in an .IFO file, cells are hybrid creatures; the data structures are defined in the .IFO file, and the multimedia content is found in the .VOB file. Each cell must start playback at a specific location in a .VOB file, referred to as a Video Object Unit (or VOBU). A VOBU is a container that houses both navigation packets, as well as multimedia packets (similar to the chunks found in an .AVI file).

While a VOBU is playing, the DVD player is able to obtain user input via on-screen buttons. You can tell the player how long a button (or buttons) should appear on the screen, and what to do when buttons are selected. Typically, the selection of a button causes the player to jump to a different location on the disc (Figure 3).

FIGURE 4. *Illustration of Parental Control Enforcement. DVD Video offers parents the ability to limit the violence, language, and adult themes their children may view. In this diagram, the game is authored with three different parental levels. The player will automatically select the appropraite content based on the parents' wishes. For example, if the player is set up for kids' titles, it will display program chain A, D (the nonviolent one), and finally E.*



Parental Controls May Open New Gaming Markets

One of the highly touted features of DVD Video is its ability to enforce parental controls over video content playback. While parental control is essential for the movie industry, it has interesting implication for games. Extremely violent games are a serious concern for many parents, and parental controls will prevent their children from playing such games. It's possible for DVD video game vendors to create low-violence and no-violence versions of their games for those who desire them. As a result, titles that previously could only be sold to mature audiences can now be sold to everyone. Unfortunately, parental control functionality is only available for DVD Video titles.

Because parental controls are activated at the program chain level, and not the higher VTS level, a DVD Video title that enables this feature should have two (or three) different versions of the same program chain, and each program should contain features specific to a particular parental level (Figure 4). This means that you only need to create different versions of the violent program chain(s) in the title — you don't have to create multiple versions of the entire title.

FIGURE 5. *System parameters and their uses.*

Parameter	Use
0	Menu Language
1	Audio Stream number
2	Sub-picture Stream number
3	Angle Number
4	Title number
5	VTS Title Number
6	PGC Number
7	PTT number
8	Highlighted button number
9	Navigation Timer
10	Title PGC Number for Navigation Timer
11	Audio Mixing Mode for Karaoke
12	Country Code for Parental Management
13	Parental Level
14	Player Configuration for Video
15	Player Configuration for Audio
16	Initial Language code for Audio
17	Initial Language code for Sub-Picture
18	Initial Language code extension for Audio
19	Initial Language code extension for Sub-picture

Unique Interactive Language

DVD Video offers content creators access to a device-independent language and a set of player parameters (or registers). There are 16 user parameters and 24 system parameters that hold the current state of the DVD player (Figure 5). Most DVD programs are interested in the

FIGURE 6. *Illustration of Navigational Commands. This example illustrates comparing a user parameter with a system parameter and jumping to a different location if the user has highlighted a different button (the system keeps the highlighted button number updated automatically).*

Sequence	Command	Comment
1	`SetSystem SysParam8,4`	Highlight the button number 4
2	`Mov Param1,3`	Set the user parameter 1 to 3
3	`Cmp Param2, SysParam8` `LinkTitle 3`	Does the highlighted button equal User param2? Yes. Go to title 3
4	`JumpTitle 2`	Otherwise, jump to a title 2

contents of System Parameter 8, which contains the currently selected highlight button, and System Parameter 9, which is a system counter.

The commands in the navigational language can be broken into the following categories: `Set`, `SetSystem`, `Goto`, `Link`, `Jump`, and `Compare` (Figure 6). The `Set` command offers primitive operations (such as compare or assignment) to manipulate the values of the 16 user parameters. You completely control the state of the 16 user parameters, but the player restricts access to system parameters. In fact, to touch the system parameters of the player, you must use the `SetSystem` instruction.

The `Goto` command is used to skip to a specific instruction number in the instruction stream (this is similar to the x86 JMP mnemonic). The `Link` and `Jump` command categories let you jump to various locations within a title or menu on the disc.

The `Compare` instructions let your DVD application test the values of either a system parameter or a user parameter. Assembly hackers will love this language, since you can squeeze up to two additional instruction categories (such as `Link` or `SetSystem`) into the space normally reserved for a single instruction category. For example, it's possible to set a system parameter, check the status of another parameter, and `Jump` to a different location with one instruction.

Copy Protection Can Protect Your Bottom Line

Since the dawn of computer science, software copy protection has caused conflicts between users and software vendors. Software companies like copy protection because it minimizes the chances that someone will steal their product. Users detest it since they can't backup their products for legitimate reasons. The promoters of DVD have tried to accommodate both sides by making copy protection of DVD Video titles optional. It's there if you need it, but you can create a DVD Video game without it.

The copy protection process initially is processor intensive — once the user's PC has been validated, however, the actual decryption of data isn't excessively processor intensive and shouldn't affect performance. To explain, before a title can be played, the DVD hardware in the PC verifies that the DVD-ROM drive is authorized to read the title (it ensures that the drive isn't a bootleg drive), and the drive verifies that the decompression software/hardware in the PC is legitimate (that is, it isn't a pirate machine setup to make copies). Once both devices have been authenticated, encrypted multimedia data can be sent from the drive and decrypted by the PC (Figure 7).

DVD-ROM Drives Make the Difference

DVD Video playback on a PC is made possible by a DVD-ROM drive (it is impossible to play a DVD Video or DVD-ROM title on a CD-ROM drive). DVD-ROM drives typically have between 10x and 12x data transfer performance and can read CD-ROM, CD Audio, and a variety of DVD formats (most DVD-ROM drives can't read CD Recordable discs). Although the drives are fast, their random access times aren't appreciably faster than a CD-ROM drive. Thus, it's important for efficient DVD Video software to mask this limitation when playing titles that require considerable searching (such as, titles that use parental control or display multiple angles).

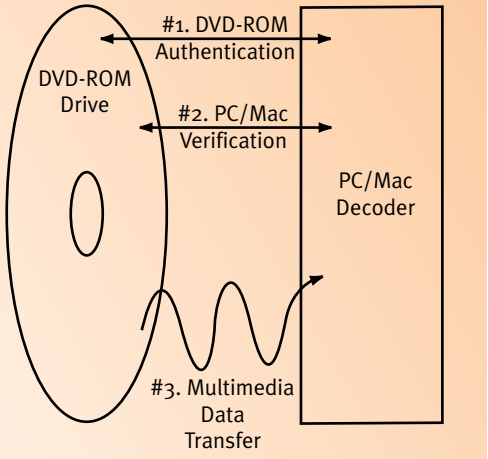
Besides enhanced performance, DVD-ROM drives support several new commands (the most important of which are copy protection and enhanced capability detection) that are accessible via SCSI or IDE command packets (depending on the interface of the drive). In fact, without the new copy protection commands, it would be impossible to play back an encrypted title.

Problems for Gaming

Although DVD Video is a dramatic breakthrough for movie titles, it wasn't specifically designed for the game market. As a result, it has weaknesses that you must work around. While most of these problems are minor irritations, others are more severe and will require careful planning to avoid.

The primary problem with the DVD Video format is the restrictions that it places on the number of instructions attached to a program. Commands may be attached to a cell or a button or placed at the start or

FIGURE 7. *The DVD copy protection architecture ensures that both the DVD-ROM drive and the PC-based decoder are licensed hardware devices before encrypted data can be read from the DVD-ROM drive.*



end of a PGC. Buttons and cells are limited to one command, while PGCs may have up to 128 commands attached both their heads and their tails. Fortunately, you can bypass these limitations in certain ways. For instance, it's possible to execute 128 commands, then branch to a bogus program (a program with no video or audio) that contains only additional commands. If further commands are required, you can continue to branch to additional hollow programs (Figure 8).

A second issue with DVD Video is the fact that all input must be captured by buttons. Since many full-motion video games use buttons for user interaction, this isn't a problem. In contrast, developers of fast-action games (those that require continuous streams of user input) are likely breaking out into a cold sweat as they read this. Fortunately, buttons don't have to be visible and can capture input for very small time intervals. By using buttons for directional input, an action game can appear to instantaneously respond to user requests.

The final issue is the lack of a graphics libraries in DVD Video. Unfortunately, DVD doesn't give you direct access to graphics memory or hardware-specific features. On the upside however, DVD Video doesn't have a windowing system — there are no device-independent layers.

New Tools Required

Because DVD Video, and to a lesser extent DVD-ROM, require new audio and visual compression technologies, the technology needs new content creation tools. These tools should offer the flexibility to choose between AC-3 and PCM audio, the ability to encode multiple audio and subpicture streams, access to all the subpicture special effects (such as wipe and scroll), and control over the bit rates of multimedia streams. Tools should have DVD player emulation as well, since emulation can greatly ease your debugging chores. Most importantly, tools should provide the ability to create the .IFO files to control navigation. This includes the construction of PTT, PGCs, cells, buttons, and navigational commands.

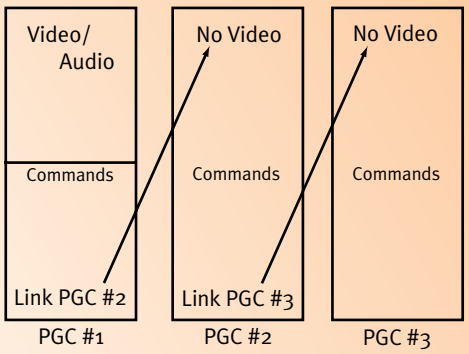
Testing Can Be Tricky

To test a DVD Video game, you'll need to obtain a DVD player. Consumer DVD Video players have been available in Japan since late 1996, and were released in the United States in early 1997. In addition to testing your title on a dedicated consumer player, you'll need to test it on a PC-based player, requiring that you install a DVD-ROM drive into your PC, and possibly an MPEG-2 decoder card and an AC-3 decoder as well.

A multitude of companies are claiming to support DVD Video in Windows 95. Whatever product you select should meet the following criteria:

- It should contain a hardware-independent architecture that can support a variety of hardware and software vendors.
- It should be able to read and process the variable bit rates contained in DVD streams (audio, video, and subpicture data). The product must be able to handle the maximum data rate of 10.08 million bits per second.
- It should support the full spectrum of DVD 1.0 features, including parental control, angles, multiple audio, and subpicture tracks.
- It should have certified AC-3 decoding support and the ability to decode all 5.1 channels of AC-3 audio. Software decoding of AC-3 is important if your product is cost sensitive.
- The product should support copy protection.
- It should be a true 32-bit, object-oriented product that uses ActiveMovie and isn't a warmed-over 16-bit MCI driver.
- It should have robust synchronization support for navigation content. Synchronizing navigation content is considerably more complex than .VOB-only synchronization.
- It should contain game-friendly features, such as a published API and progress notification messages.
- It should support dual-layer, and/or dual-sided discs.

FIGURE 8. *Illustration of working around 128-command limitation. This particular program must execute approximatelty 300 commands. By linking a couple of bogus PGCs (PGC 2 and 3), it is able to circumvent the 128 command per PGC limitation.*



Despite the rough state of the tools, DVD Video has the potential to forever alter full-motion video games. It offers tremendous video quality, surround sound with multiple audio tracks, and closed captioning. Furthermore, it provides a platform-independent navigation engine that enables interactive movies, parental control over content, and dynamic changing of angles of view. If your project is not dependent on 3D graphics or a platform-specific library, you should give serious consideration to a DVD Video version of your game. ■

Linden deCarmo is a staff software engineer at Oak Technology Inc. working on multimedia software. He currently is member of Oak's Windows 95-based Interactive DVD Browser project and can be reached at lindend@ibm.net

FOR FURTHER INFO

CPU Consumption
http://www.techweb.com/se/directlink.cgi?EET19970224S0007

DVD-ROM Performance Specs
http://www.pioneer-eur.com/products/multimed/optical/dvd.htm
http://www.hitachi.co.jp/New/cnews/E/970303B.html
http://www.toshiba-teg.com/diskdiv/eng/dvd.htm
http://www.creaf.com/mmuk/pcdvd/info/frames2.html

Ultimate Source of DVD Info
http://www.unik.no/~robert/hifi/dvd/
alt.video.dvd (newsgroup)

E3 Aie

For some, the Electronic Entertainment Expo (E3), taking place this June in Atlanta, Georgia, will be a big game festival. For others, it will be the one chance this year they have to prove that their companies are going to be around for next year's E3.

In only three years, E3 has become the most important event for the industry as a whole. While the secrets of developing and producing a game may be learned at the CGDC, the results of that knowledge are put on display for retailers, press, game players, analysts, and the world at E3.

E3 was is owned by the Interactive Digital Software Association (IDSA), a trade group comprised of 45 of the world's top entertainment software publishers. The IDSA itself was formed after many of the larger companies in the industry realized that they needed a unique representative trade group to boost the industry and represent it before Congress, foreign governments, and many other entities worldwide. E3 was started by the IDSA in conjunction with IDG to give entertainment software companies a trade show where they could promote themselves with a higher profile than had been available to them at shows like CES or Comdex.

In order to provide a preview to this year's conference, *Game Developer* interviewed four representatives from various industry positions to get an idea of what to expect at this year's conference. *Game Developer* also talked with Doug Lowenstein, president of the IDSA, to find out more about the conference and the organization. You'll find that extensive interview with Lowenstein on the *Game Developer* web site.

This year's E3 will mark a busy 12 months for the IDSA. The organization became involved in recent trade negotiations with China to ensure that critical aspects of those agreements protected the specific interests of copyright and piracy protection for game developers worldwide. In addition, the IDSA has partnered with the CGDC and the Interactive Services Association to create a special conference track at E3 for game developers attending the show. Lowenstein said that the IDSA is working harder to reach out to smaller developers and development staff at industry companies. The long-awaited IDSA web site will also launch sometime in the near future.

When:	June 19-21
Where:	Georgia World Congress Center and the Georgia Dome, Atlanta, Ga.
What:	Conference and trade show focusing on interactive content. Wholly owned by the Interactive Digital Software Association (IDSA), E3 presents the latest in interactive entertainment software and related products.
Estimated Attendance:	55,000
Cost:	Three-day conference and exhibits pass is $195; an exhibits-only pass is $45.
Miscellany:	400+ exhibitors. Tom Brokaw keynoting. Conference tracks include business management, retail trends, financing options, online trends, and content development. 30 conference sessions.
To Register:	Go to the E3 web site at www.mha.com/e3/index.html.

The IDSA has also played a crucial role in helping establish the self-managed games rating system, and has worked on alternatives to government regulation of the electronic entertainment industry. At this year's conference, Lowenstein and the IDSA will make available results from several ongoing research projects that will help developers better understand funding issues, online trends, and better business practices. Last year, the IDSA put out a report that was used to help many nongame industry press and people understand the substantial impact the industry has on the U.S. economy in terms of overall revenue and employment.

In addition to the IDSA-sponsored conference tracks and the orgainzation's research presentations, this year's show will feature 528,000 square feet of exhibit and meeting space, 56 first-time exhibitors, a keynote by Andy Grove and Tom Brokaw, and enough new titles to please even the most die-hard gamers. Perhaps the biggest thing about this year's conference is the location, which has moved from Los Angeles to a bigger venue in Atlanta to better accomodate all the large booths and crowds. While some West Coast developers jeered at the announcement of the move last year, Lowenstein hopes that, in addition to the benefits of a larger venue, the new locale will provide a chance for more East Coast companies (and hopefully East Coast mass-media outlets) to make the trip.

Four people sure to make the trip were kind enough to spend some time talking with *Game Developer*, sharing their unique views on what E3 will be and what they'll be doing at this year's conference.

The Speakers

W. BINGHAM (BING) GORDON, ELECTRONIC ARTS. Gordon is a cofounder of Electronic Arts and currently serves as the executive vice president of marketing. Gordon oversees marketing staffs located in San Mateo, California; Austin, Texas; Vancouver, British Columbia; and London, England. Gordon has also served as executive vice president of EA Studios, and, prior to that, was senior vice president of Entertainment Production, responsible for the design, development, and production of entertainment titles and creative properties.

TRACY GILES, MAXIS. Giles has spent five years at Maxis, where she first entered the electronic entertainment industry. She now works with Sam Poole to assist in establishing the Maxis sales operation, since the company left Broderbund as an affiliated label in 1993.

STEVE CRANE, ACTIVISION. Crane has been the vice president of technology at Activision for about 18 months; he previously held a similar position at children's game maker Knowledge Adventure. Prior to that, Crane worked at Electronic Arts on its first series of 3DO Products, which included ROAD RASH, SHOCKWAVE, and JOHN MADDEN FOOTBALL.

JOE CATAUDELLA, TRONIX MULTIMEDIA. Cataudella, who has spent over 10 years in the game retail industry, is the owner of Tronix Multimedia, a web-based mail-order and retail store in New York City that caters to hardcore gamers around the world. Previously, he was the manager of a top NYC software dealer. His store carries titles across the board for PC, Sega, Sony, and Nintendo. He's attended E3 since its inception and CES for years prior to that.

THOSE INVOLVED IN INTERACTIVE ENTERTAINMENT WILL HAVE THEIR EYES FOCUSED ON ATLANTA THIS SUMMER, AS INDUSTRY HEAVIES SHOW OFF THEIR LATEST AND GREATEST WARES AT E3 1997.

The Questions

? E3 should stand for "Evaluate Evaluate Evaluate." What do you do — or see others do — at E3 to evaluate the industry, the competition, and your own position against the rest of the field?

GORDON: The most valuable feedback we get at E3 is not about software titles at all, but about hardware prospects. The "buzz" in each hardware booth defines retail and software community support for the rest of the year. Since most of our titles are on a 12-15 month cycle for fall release, we will use this E3 to prioritize our hardware support for Christmas 1998.

Retailers don't order software at E3, as a rule, they order shelf position. Publishers with a good "buzz" at E3 get allocated more of a retailer's open shelves for fall. It's a lot easier for them to predict company market share than delivery date and chart position for a single title.

CATAUDELLA: Basically, I have a ground rule: Hit every booth to see what's coming up and figure out which titles are near finished, which ones are beta, alpha, and so on. I scan each booth during the latter two days to make sure nothing new has popped up, because sometimes something new might be shown on day two or three.

As a retailer, I look for stuff that stands out to me — I've been around this industry long enough to spot a hit. I watch the crowds, too — that's important because I will go back and place higher orders to avoid being caught without enough stock for hot titles. I also make a list of dogs to avoid. How much a company tells me they're going to push a game through a channel is important, too. Good games can get buried without good support.

? What are you expecting at this year's E3? Is there anything special you're looking for or expecting to happen?

GORDON: It appears that Sony and Nintendo are committed to matching each other's mass market price

E3 PROVIDES A VENUE FOR PUBLISHERS, PRESS, AND THE RETAIL CHANNEL TO ESTABLISH TRENDS AND TO COMMUNICATE AREAS THAT NEED TO BE IMPROVED.

Game magazine editors do "order" software at E3. They all come away from the show with their "Top 10 Best of Show" lists, and allocate feature stories accordingly. SOVIET STRIKE garnered a lot of feature coverage because of its E3 splash last year.

GILES: E3 provides a venue for publishers, press, and the retail channel to establish trends and to communicate areas that need to be improved. Maxis uses E3 to meet with clients, analysts, and press to launch new releases and reinforce relationships.

CRANE: There certainly are a lot of vendors going around looking at competitors for things like who's got a flashier renderer or a better physics model. It's actually a lot easier at E3 to evaluate technology than a specific product — you just don't have the time to play a game thoroughly enough. You're focusing on what's new. I can sort of breeze through the show in about two hours to do this sort of evaluation.

Specifically, I'm looking for any advances in the state of the art in 3D graphics, AI, physics simulation, and network multiplayer.

changes. So I'm very curious to see what the software line-ups are going to be on each machine this holiday season. This will also be a "make or break" show for many of the publishers who missed the market last Christmas and haven't been profitable. Which ones will show the creativity and gumption to turn themselves around?

GILES: I think that in the past, the Electronic Entertainment Expo was focused on multimedia and console products. I'm anxious to see better graphics and a fresh approach to online gaming.

CRANE: Frankly, I'm not expecting anything in particular. Things we saw this year that had the biggest impact, such as TOMB RAIDER, weren't even seen at last year's E3.

I'm expecting more character-based 3D games. This is important for younger users now entering the console market. I expect multiplayer online games will be all over the place, but overall, I'm not expecting anything wildly new.

CATAUDELLA: I'm anticipating seeing more Nintendo software, which everyone is asking about. I'm hoping to see the DD Drive for the Nintendo 64, and I'm very excited about the force-feedback peripherals. I'm very curious what Sony's going to do next — is this platform maxed out technologically? I want to see if they're going to push out quality titles, because the dazzle factor for PlayStation is over. I'm also interested in seeing what new hardware rumors creep up, such as new game machine platforms.

doing their part to port products to Windows 95, but Microsoft has not been aggressive enough in upgrading consumers from Windows 3.1 to Windows 95. Our hope is that Windows 95 will become the gaming platform of choice.

CRANE: Clearly, Microsoft has something to prove. No one's taking them seriously at this point, but no one's ignoring them either. Eidos will be hot. They've published a couple of interesting titles and made some interesting deals. Nintendo is still an issue — all the titles for that machine have fallen well short of Mario 64. What's going to happen with the next-generation of third-party games? Can anybody come up with something new on the PlayStation to compete with the capabilities offered by the Nintendo? Multiplayer has a need to begin moving into the limelight. I'll be curious to see if CUC can start to pull all its acquisitions together into some sort of whole.

CATAUDELLA: Sony is still riding high — we're over the look-at-what-we-can-do hump now. Sony's got some strong titles like FINAL FANTASY VII now. The RPGs are going to be big on that machine. I think Nintendo is going to stay strong.

Sega has a lot to prove. Can they pull the Saturn through? U.S. Sega is not happening. The Segasoft line is not doing it — it's only because of the arcade titles.

? Where is the game industry overall as it heads to E3, and where is it after E3 has had its effect?

GORDON: This is clearly the breakthrough sales year for advanced game systems. I'd like to see E3 kick off a higher level of interest from news and entertainment press about the phenomenon of interactive entertainment.

GILES: I think E3 provides a common ground where the industry will come

BASICALLY, I HAVE A GROUND RULE: HIT EVERY BOOTH TO SEE WHAT'S COMING UP AND FIGURE OUT WHICH TITLES ARE NEAR FINISHED, WHICH ONES ARE BETA, ALPHA, AND SO ON.

? Who's riding high into this year's E3, and who's got something to prove?

GORDON: Sony and Nintendo are driving the US videogame business. And any publisher with a Top 10 title or franchise is in pretty good shape, if they're not spending their gross profits on corporate jets and jobs for relatives.

I'd like to see the PC business more clearly presented at E3, as a major entertainment software platform. Perhaps Intel could play this role. I hope that Andy Grove's keynote speech is very well received.

GILES: I think Nintendo is better positioned than ever with their Nintendo 64 system. I believe Microsoft has to improve penetration of Windows 95 this year. Publishers are

trying some more creative things. Force-feedback will be big. So will multiplayer. There are a lot of classics coming back, which seems to be a backlash against some of the poor innovation overall.

By next E3, I think we'll see or hear hard facts about Sony's next PlayStation. We might also see if the Sony Yarouze machine (the consumer development version of the PlayStation) has managed to produce a sneak hit product for them.

? Tell me about the session you'll be participating in at E3. Who should come to it and why, and what do you expect to talk about specifically?

GORDON: EA has a very deliberate process for prioritizing among platform opportunities, and a reasonably good flow of information about global trends and intentions. This has served us pretty well over the years, helping us to stake early leadership positions on PlayStation, Genesis, Amiga, and Commodore 64. Perhaps even more importantly, we avoided another 75 systems, mostly failures.

People can use this session to learn about EA's biases and secrets, and make

I'D LIKE TO SEE E3 KICK OFF A HIGHER LEVEL OF INTEREST FROM NEWS AND ENTERTAINMENT PRESS ABOUT THE PHENOMENON OF INTERACTIVE ENTERTAINMENT.

together to establish viable directions that will continue to perpetuate our industry as exciting and growing.

CRANE: In terms of genres, it seems that there is a swing back toward RPGs/Action-RPGs. Real-time strategy is exploding, and I think you'll see significant enhancements in this area. I'm expecting, in terms of multiplayer, that you're going to see a move to this whole world of "Persistent Worlds," where the world and story goes on after you log off. I'll certainly be over looking at ULTIMA ONLINE as a future trend.

In terms of the games industry in general, I see basic steady growth after E3. It's particularly good on the console side — even on the PC-side. There is a good level of innovation, even amidst a lot of me-too stuff. This is one of those mini-golden ages we've seen before.

In terms of next year, I'm not really sure where new hardware might be — M2 or Sony. I think Sony is going to wait until 1999 to show off new hardware.

CATAUDELLA: I think a lot of companies are going to be exploring new areas. I think you'll see a lot of people breaking away from these me-too titles and up their own minds about whether EA can still affect the success or failure of any new hardware platform.

GILES: The session that I will be participating in is "Working with Marketing." The objective of this session is to assist developers and product development individuals in establishing better communication with marketing departments and free-lance consultants. The contribution I hope to make is to share my experience with establishing marketing material standards at Maxis. As Director of Channel Marketing, it has been my

challenge to identify the materials required to gain shelf-space within the timeframe of the retail channel. I have worked very closely with producers, directors, and product managers and affiliates to provide materials that will accelerate the adoption of our products in the retail channel and electronic channels.

CRANE: I haven't really sketched out a plan, but I assume I'll be talking about Internet games — what's happening and the opportunities they raise for people. I'll be discussing the idea of "Persistent Worlds," where a server can be compiling worlds and providing new opportunities. I'll probably be talking about how boring 3D rendering is getting.

CATAUDELLA: I won't be at a session. I'll be on the show floor the entire time, evaluating titles, talking to other attendees, and making sure I've gotten to every game once and made a list of what's hot and what's not.

everybody's ideas — anything I've heard about three or four times, I have to see. After you've talked to 40 or 50 people, you've got a decent idea.

CATAUDELLA: Retailers going to E3 for the first time need to collect as much info as possible. Don't miss anything. E3 is a lot different from getting sales kits in the mail. Get a good feel for the product yourself. Don't rely on the sales people since they all love their own products.

? Any other basic thoughts you'd like to offer regarding E3?

GORDON: E3 is a lot more exciting than the old Consumer Electronics Show that it replaced. Maybe it's just me, but I think entertainment software is a lot more interesting than radar detectors and VCRs.

GILES: I am looking forward to gaining exposure to new directions that will assist our industry in addressing the challenges that we face in the coming year with a retail channel that is struggling and a consumer that is more sophisticated.

CRANE: One thing I do is run through and do a quick quantitative count on how the genres are breaking down. How many RPGs? How many sport games? How many fighting games? And so on. This way I get an idea what the overall trends are.

CATAUDELLA: Companies can really help retailers with their pre-sell at E3. They can really help a retailer educate customers by presenting titles and information that retail can pass on first hand to customers. I can sell a game to people when I add the weight that I've played it behind what I tell them otherwise. I wish companies would bring more developers to E3 because they are much better to talk to than sales reps. I can talk to the sales reps anytime. ■

E3 CAN BE OVERWHELMING FOR THE NOVICE! IT IS FILLED WITH EXCITEMENT, AND IT HAS GROWN TREMENDOUSLY OVER THE LAST THREE YEARS.

? What advice do you have for people in the industry who are attending E3 for the first time?

GORDON: Wear comfortable shoes or insoles.

GILES: E3 can be overwhelming for the novice! It is filled with excitement, and it has grown tremendously over the last three years. What I like to do is plan my three days ahead of time, identify the sessions I am going to attend, make lunch dates with those people I never get to see, and map out the booths that I just have to visit. If you can stay all three days, use Saturday to visit booths because it can be really quiet on the floor.

CRANE: Usually at E3, there are about 10 titles that you need to see. So when I get there the first thing I do is talk to various people I know and get early info on what those 10 are. I write down

Dawn of the 3D Pixel Sprite

Game artists involved in real-time 3D projects typically are faced with a paradoxical task: to create eye-catching, detailed graphics that can be displayed without a great commitment of end-user system resources. The twin objectives are for the game to be both great-looking and smoothly interactive. However — and here's the paradox — either goal can generally be advanced only at the cost of the other. In the last Artist's View ("Different Perspectives on 3D Sprites," April/May 1997), I discussed two common approaches to this dilemma: 3D sprites of either the bitmapped or polygonal variety. Each of these methods involves significant compromise and a delicate balancing act between insufficient detail on the one hand and unsatisfactory run-time performance on the other. Even the most skillfully constructed and carefully texture-mapped low-resolution polygonal models cannot avoid a somewhat crude, blocky look, while the exponential memory requirements of a full series of bitmapped sprites force limitations on animation smoothness.

The ideal solution calls for a figure that occupies the 3D environment as convincingly as a polygonal model, provides the fine detail of a bitmap, yet can be more smoothly animated than either of these quasi-solutions. That's a tall order, but AnimaTek International claims that its Caviar Technology for creating "3D pixel characters" fits the bill. A growing list of game developers is looking to prove the company right this coming holiday season, when their Caviar-powered titles hit the market.

AnimaTek is familiar to many computer artists for its 3D landscape generation software, World Builder, and for Bones Pro, the archetypal skeletal deformation plug-in for 3D Studio. Recently, the company's Moscow-based programming staff turned its considerable talents to developing a technology that would enable highly realistic, real-time, online environments peopled by fully interactive avatars. Polygonal models were quickly ruled out.

"Polygons are fine for depicting flat walls," observes AnimaTek president Vladimir Pokhilko, "but for a complicated object like a person you want to be using polygons little bigger than a pixel on the screen. Yet each polygon requires three vertices, normals, texture-map coordinates.... It's a huge overhead. You just can't achieve great quality using polygons in real-time."

AnimaTek's quest for a nonpolygonal solution started with voxels (volume pixels). However, while valuable for medical imaging, the volume information inherent in true voxels — like the baggage attached to polygons — was unnecessary overhead for a real-time entertainment application. Animatek wanted to depict the object's surface only and to do so as efficiently as possible.

If the combination of high-detail true 3D sprites and great run-time performance sounds fishy to you, take a look at AnimaTek's new Caviar Technology for creating 3D pixel characters, and taste the difference voxelization makes!

The solution — dubbed Caviar Technology — involves two elements. The first is a converter, the so-called Caviator, that voxelizes or "caviates" a polygonal model by covering its surface with a chain of 3D pixels, which then stand in for object geometry. You can think of this as something like a mummy: the original polygonal model serves as the mummy's body, which is then completely wrapped in a strand of 3D pixels — thereafter, only this wrapping is used to represent the character.

The Caviator works as a plug-in for 3D Studio (R.4 or MAX) or as a standalone that accepts file formats such as Alias and Softimage. This means characters are created with the artist's chosen tools. The second element of Caviar Technology is a run-time library that developers can incorporate into their applications to manipulate characters and their component parts and to control lighting and shadows in real time.

In keeping with their genesis as online avatars, Caviar (.CVR) files can be made streamable. AnimaTek has already made a Netscape plug-in available on their web site and is working with Netscape toward an even more integrated solution for the very near future. While the use of Caviar characters as online avatars holds great promise for the quality of tomorrow's virtual environments, the potential for using them as game sprites is already being realized.

All in a Roe

The Caviar system places remarkably few strictures on the artist while providing several significant perks. The biggest advantage is simply that a Caviar character offers the best of both worlds when compared to other 3D sprites. The characters are both rich in surface detail like a bitmap and authentically dimensional like a polygonal model, all within an economical file size that compares well to these other methods. As with bitmapped sprites, the complexity of the 3D character created has no effect on run-time performance: polygon count and texture maps serve only to impart detail during the "caviation" process. These resource-intensive elements aren't brought into the game, so no "optimizing" is called for as would be the case with true in-game geometry. Actually, as the Caviator ignores 3D Studio's "smoothing groups," a highly tesselated object helps prevent a faceted look.

Another big advantage provided by the Caviator is the ability to create LODs (level-of-detail models) automatically. Rather than calling upon the artist to fashion separate models with a different number of polygons appropriate for viewing at close, medium, and long ranges, the user simply indicates how many LODs are desired. The Caviator then creates the appropriate versions, increasing resolution by 150% for each subsequent LOD. Resolution with a Caviar object is actually a matter of scale: the coverage of 3D pixels per "world unit." "Higher resolution" corresponds to smaller scale (that is, denser coverage of 3D pixels). The multiple LODs reside within the same .CVR file, and the most suitable size is automatically selected by the Caviar rendering library at run-time. This feature can provide a great savings in development time, as a single model can be used for cinematic sequences and any number of in-game LODs.

The polygonal objects that you start the process with can use any texture maps compatible with your 3D software, but the Caviator will convert them to 3D pixels using an indexed 256 color palette (Adobe Photoshop .ACT format), which must be prepared beforehand. All or part of the palette can be used during the caviation process: Start Color and End Color spinners set the range of valid colors from the specified palette, which allows you to reserve color ranges for other purposes such as system colors, backgrounds, or on-screen controls. High-color mode is also supported with a 16-bit gradient option.

Rendering speed is identical for both flat and texture-mapped materials. With 3D Studio materials, the Caviator uses ambient, diffuse, specular, shine strength, shininess, and self illumination parameters for flat materials and only the diffuse parameter for textures. Scene lights are ignored, as real-time lighting and shadows are defined from the Caviar rendering library. To avoid adversely affecting performance, shadows are cast only onto a flat ground plane but not onto walls or other intervening objects.

Any sort of polygonal object run through the converter can then be viewed from all angles as a 3D pixel object, which looks essentially identical to the geometry on which it is based. Beyond this, though, the .CVR file can also include animation data for segmented characters (those made of linked body parts rather than a seamless skin). Animation data actually adds very little to the file size: the 3D pixel chain for each body part is saved just once, and a transform matrix is then created for each body part throughout the movement routine. Each frame of animation adds only about 60 bytes to the .CVR file size, so numerous frames can be used for each movement without blowing your memory budget wide open. However, only position, rotation, and scale keys can be processed by the Caviator. Animation involving morph keys, skeletal deformation, or changes in materials are not yet possible with Caviar Technology.

Though it's unfortunate that Caviar cannot yet handle these advanced forms of animation, the system does offer a number of advantages to using segmented characters. One is that body parts can be used by multiple figures, a practice that can improve performance by cutting down on the amount of data that must be loaded into memory at run-time. For example, two characters could use the same body — torso, arms, legs — topped with a different head to distinguish between them. Animation routines can also be saved separately from geometry, so the two identical bodies could each be given a distinct walk.

Caviar in Action

Several other advantages of using segmented models with the Caviar system are being exploited by artists and programmers at TecMagik, where they're using Caviar Technology in the creation of the company's next title,

Caviar files can represent extremely complex geometry in high detail, yet are viewable in real time as true 3D objects.

Given careful attention to modeling and an understanding of the range of movement called for, great-looking animated characters can be achieved even with the segmented models called for by Caviar.

DEADLY HONOR. It's a familiar-sounding action-packed exploration game, except that it features the towering form and glowering mug of aikido master Steven Segal and so, fittingly, mixes the usual corridor-tramping, shoot-'em-up action with the sort of hands-on mayhem you'd expect from a fighting game.

"When you're trying to take advantage of the popularity of a license like Steven Segal, being able to create a character that looks as much like him as possible is a huge plus," points out project producer Robert Burnett. "We're doing a cyberscan to get a very lifelike digital representation of his face and a motion-capture shoot to get things like gait and fighting stance — movements that would be less accurate if animated by hand."

Caviar Technology lets TecMagik take advantage of these authentic details while maintaining a high level of performance. One trick is to employ Caviar's ability to mix and match voxelized body parts. A high-resolution head can be used with a lower-resolution body, for example, to retain the desired detail in the model's face, while requiring less memory than would a complete high-resolution character.

Caviar's flexibility also has facilitated the integration of motion-capture data essential to the authentic aikido action the game demands. Though the .CVR file can contain the full segmented model and animation data together, it also can consist of just voxelized geometry or just the transform matrix for movement. TecMagik has chosen to caviate its models piecemeal — one body part or connected group of body parts at a time — then use Direct3D to associate the pieces with motion-capture data.

"Using the Caviar engine in combination with motion-capture data allows us to create fluid movement that looks really cool, and Direct3D lets us put it all together very efficiently," explains Burnett.

To get the segmented models to look like a unified figure rather than a collection of body parts is a challenge welcomed by TecMagik's senior 3D motion artist Bruce Gill. "The main trick is to have them not wearing trenchcoats," he points out with a laugh. Not only do you want all geometry to be closely associated with the figure's form, he explains, it's also important to understand how the figure moves: how it moves in general, but more specifically, how it will be called upon to move within the confines of the game. Correctly setting pivot points and planning ahead to build geometry that works throughout the full range of rotation called for by the game are crucial steps in the creation of a successful segmented model.

"This game calls for some very extreme moves, so we've done some interesting segmentation of the model," Gill explains. "We know a lot of flex is going to take place, a lot of over-rotation of the joints, so I've broken up the model in nontraditional ways. Essentially, a single-piece torso is most commonly used, but we've broken it up into multiple parts to accommodate the sort of extreme rotation that will be taking place there. We'll also have over-rotation of body parts where they may just snap, which is a typical Segal thing to do. We can just swap in different segments for the broken limb. Also, though we can't do morphs, we can swap out fast enough that we get a nice transition from a neutral to an exaggerated facial expression."

Caviar's rendering library — with its ability to handle the model in discrete body parts — lets TecMagik's artists and programmers spice up DEADLY HONOR with these sorts of special effects that add visual spice and fun to the game.

A former modeler for Viewpoint and Zygote, Gill appreciates the freedom to craft a character without relying on texture maps for detail. "A lot of what texture maps do is add detail that's deficient in low-rez geometry. With Caviar, they're unnecessary. From a modeler's standpoint, I'm very

Animation is handled with transform matrices costing only a few bytes per frame, delivering remarkably fluid movement in exchange for very little memory.

Real-time lighting and shadows add even more character to Caviar's 3D pixel characters. Note how highlights relate to the placement of lights in each instance: this isn't just a palette trick.

impressed with the ability of Caviar to allow the geometry to do most of the talking as far as providing detail in the face and clothes. Because we're using high resolution, highly detailed geometry, the use of a texture map would only convolute that detail.

"When people see stills of what we're working on, they can't believe it's actual game-play graphics that they're looking at." Those same people will be even more impressed when they see the fluid animation, real-time lighting and shadows, and memory-saving special effects provided by the rendering library. It may sound fishy to those who haven't tried it, but Caviar is a real treat.

Taste It and See

To see AnimaTek's Caviar Technology in action, you can visit its web site (www.animatek.com) and download a Netscape plug-in that will let you to view a number of animated samples online. You can also download a standalone Caviar player or request the free demo CD, which includes several more sample characters. If you want to try your hand at creating your own Caviar characters, a one-month, noncommercial evaluation package, which includes the 3D Studio IPAS, Caviar API, and full documentation, is available for $1,000. Commercial licenses for single product use are available for a $10,000 advance against a 50 cents/copy royalty or for a flat fee of $20,000. Multiple product licenses are $40,000 in advance against a 25 cents/copy royalty or a flat fee of $50,000. Hey, you didn't expect something called Caviar to come cheap, did you? ■

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

Caviar Technology

AnimaTek International, Inc.
812 S. Fremont St.
San Mateo, CA 94402
800 471-1233 or 415 638-2177
www.animatek.com

SOAPBOX

by Chris Crawford

Weeds and Oaks

Fourteen years ago, I moved into a new house in the hills and commenced planting trees on a few acres of open ground. I was urged to plant Monterey Pine, eucalyptus, and other fast-growing species. I did plant a few of those, but I also planted lots of oaks. I groused acorns from underneath good-looking trees and planted them on my land. Some of my friends shook their heads in good-natured dismay at my naiveté. I'd be dead before those trees were mature, they said.

They were right — but who ruled that the account books on a man's life close when he dies? I can derive just as much satisfaction from the expectation of a long-term achievement as from instant gratification. If the mind's eye can see into the future clearly, the fruits of the future are just as sweet as those of the present.

The computer games industry seems to take the opposite approach. They like to plant weeds, not oaks.

Consider, for example, the clonitosis that is endemic in the industry. Everybody's rushing to make COMMAND & CONQUER clones. A few years ago, DOOM and MYST were being frantically imitated. Yet COMMAND & CONQUER is little more than a remixing of design concepts that we've seen hundreds of times in previous games. DOOM is just a souped-up version of WOLFENSTEIN 3D, which in turn was based on an Apple II game called CASTLE WOLFENSTEIN. MYST is an utterly conventional adventure game, in design terms no different from the original ADVENTURE computer game, only souped up with '90s graphics. It seems that we are dizzily cloning the clones of old clones. Wouldn't it be better in the long run to take the time to design something original once in a while?

Then there's the emphasis on the latest techie-gee-whiz stuff. The industry spends lots of time and money sweating the newest technical developments, with each developer trying to one-up everybody else with some new software gizmo. First it was "2½D" displays; then it was 3D displays; then it was 30fps 3D displays. Fortunately, the universe stops at three dimensions, or we'd surely be seeing claims of "3½D" or even "4D" ("and 5D is just around the corner!") Wouldn't it be better to invest some of that money in the occasional creative fling?

What nobody seems to notice is that today's cutting-edge technology is tomorrow's silly fad. Remember lava lamps? They were high-tech in the '70s — now they're gauche. My game EASTERN FRONT (1941) attracted lots of attention in 1981 because it had "smooth scrolling" — nifty-keen! Fortunately it was also a decent game. Even so, it's laughable by current standards.

How'd we get ourselves into this hole? I think that it's primarily due to the Silicon Valley get-rich-quick mentality. Slap that company together, get that IPO out, then cash in — what happens after the IPO doesn't matter. Such a mentality prefers flash and glitter to substance, and it seeps into every sinew of our community. We ship boxes with fabulous exteriors and mostly air inside. Our advertising takes hype to heights that would make Madison Avenue blush. Even our companies combine snazzy corporate offices with high-turnover workforces.

My only emotional response to this is a sense of sadness for the futility of a community that measures success by current income. All those weeds crowd and shove each other, fighting for sunlight/money, and when one weed manages to grab a bigger portion of sunlight/money, all the little weeds gaze on approvingly and whisper, "If he's rich, he must be right!" Then winter comes and they all die.

Still, weeds have their place in the ecosystem, and we'll always have computer games companies making their living on the latest fads. Indeed, I suspect that the entire industry is permanently wedded to weed-think.

Which is fine by me. My weed-loving friends smirk at my little oak seedlings. I smile at their ribbing. Someday, my seedlings will be mighty oaks towering far over the heads of the weeds. ■

Chris Crawford began designing games for Atari in 1979. There, he created the best-selling EASTERN FRONT (1941). In 1984, he wrote BALANCE OF POWER for the Macintosh, his most successful game. He is currently working on a technology for interactive storytelling. He also founded the Computer Game Developers' Conference and has lectured on game design in eight countries and at many universities.