gd

GAME DEVELOPER MAGAZINE

MAY 1998

UBM Tech

Is Our Silence Killing Us?

In an ideal world, games would be successes or failures based solely upon consumer tastes. That's the way the free market should work. I'm starting to see that this economic model isn't operating quite as I might hope. As I wrote about last September ("The Wal-Mart Effect"), nationwide retailers who carry game software are starting to push back at publishers. Games with objectionable content (sex, violence, language) are being silently passed over by stores, and in some cases, the chains are approaching publishers with suggestions for toning down certain content.

The publishers I've spoken to about this situation express growing unease about the situation. They fear a future where game design decisions are based largely upon what the largest retailers will carry, not what they think the public will enjoy. Unfortunately, these publishers are often unwilling to speak openly about the problem and point a finger at channel partners, for fear their supposed allies will retaliate economically. And who could really blame these publishers? Nationwide chains now represent a significant portion of game sales. So the problem slowly mounts because publishers remain silent about it, and retailers take heart in their ever-growing influence and try to sway our content even more.

I fear where this headed. Recently an acquaintance of mine at a major publisher asked me to look at the current crop of titles atop the PC Data charts. He pointed out an interesting fact that wasn't as apparent as recently as a year ago: as we go to press, six of the top twenty Windows game titles on the January 1998 chart use licensed content from outside the industry. Lego. Monopoly. Tonka. Barbie. You get the idea. Now, it's a given that Hasbro's a major reason we see so many familiar boardgames packaged as CD-ROMs, and their titles are generally well executed. But the toy licensing trend seems to be growing as other publishers join in.

Here's my fear: these games are selling better because the buyers for major chains favor games that leverage brands like Tonka, in part because the buyers recognize the brands and think that consumers will favor them, and because of these titles' lower wholesale prices. As a result, more licensed-property titles get picked up by the big chains, appear on more shelves around the country, and consequently sell more. The simplicity of these games also means fewer returns for the stores to deal with. In contrast, the games that go out on a limb, those that try to push the boundaries of genre, design, or pioneer new technologies are viewed by the large retailers as too risky or expensive (besides being too violent, sexually explicit, and so on). Often, they never see the same distribution.

This is a tough thesis to argue because it has some chicken-and-egg elements. Do some games sell well because they're good and therefore get wide distribution, or do games get wide distribution, consequently sell more, and then get pegged as good? You can argue it either way. Unfortunately, this arguing isn't taking place enough. A cone of silence has descended upon most publishers, who fear the consequences of rocking the boat with retailers.

It's time to talk about this phenomenon. When it comes to innovation, game developers get beaten up enough as it is by players and reviewers. If this trend is real, I fear game innovation will suffer even more at the hands of channel partners who would have us develop only what they consider safe.

New Column: Hard Targets

Omid Rahmat, a former analyst at Jon Peddie Associates who has since left the nest, begins his Hard Targets column this month. Hard Targets will look at trends in consumer hardware, presenting a snapshot each month of critical technologies (3D graphics controllers, next-generation consoles, 3D audio chips, CPUs, motherboard architectures) that are driving our target markets forward. Welcome, Omid. ■

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 Wesley Hall
whall@mfi.com

ART DIRECTOR Laura Pool
lpool@mfi.com

EDITOR-AT-LARGE Chris Hecker
checker@bix.com

CONTRIBUTING EDITORS Jeff Lander
jeffl@darwin3d.com
Josh White
josh@vectorg.com
Omid Rahmat
omid@compuserve.com

ADVISORY BOARD Hal Barwood
Noah Falstein
Brian Hook
Susan Lee-Merrow
Mark Miller
Josh White

COVER IMAGE Istvan Pely

PUBLISHER Cynthia A. Blair
cblair@mfi.com

WESTERN REGIONAL SALES MANAGER Alicia M. Langer
(415) 905-2156
alanger@mfi.com

SALES ASSOCIATE Ayrien Houchin
(415) 905-2788
ahouchin@mfi.com

MARKETING MANAGER Susan McDonald

AD. PRODUCTION COORDINATOR Dave Perrotti

DIRECTOR OF PRODUCTION Andrew A. Mickus

VICE PRESIDENT/CIRCULATION Jerry M. Okabe

ASST. CIRCULATION DIRECTOR Mike Poplardo

CIRCULATION MANAGER Stephanie Blake

CIRCULATION ASSISTANT Kausha Jackson-Craine

NEWSSTAND ANALYST Joyce Gorsuch

REPRINTS Stella Valdez
(916) 983-6971

Miller Freeman
A United News & Media publication

CEO-MILLER FREEMAN GLOBAL Tony Tillin

CHAIRMAN-MILLER FREEMAN INC. 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

VICE PRESIDENT/SD SHOW GROUP KoAnn Vikören

SENIOR VICE PRESIDENT/SYSTEMS AND SOFTWARE DIVISION Regina Starr Ridley

www.gdmag.com

INDUSTRY WATCH

by Alex Dunne

SUE? SUE? ANYONE WANT TO SUE? Five dissatisfied Ultima Online players decided to file a class action suit against Origin Systems and EA in San Diego County Superior Court in early March. The crux of the suit stems from the well-publicized problems that Origin has had supporting UO on a 24x7 basis, as well as gripes with the game's lag time, subscription payment policies, and the hardware needed to play UO. In short, they didn't like the game. The suit seeks unspecified punitive damages, and the attorney representing the five disgruntled role players told GameSpot in an interview that if the case is approved by the court, others could join the bandwagon.

SPEAKING OF CLASS-ACTION LAWSUITS, GT Interactive got its ninth one slapped on it since January. Like all of the others, this one was filed by investors who claim officers at GTI misled shareholders and then profited on the artificially inflated stock prices.

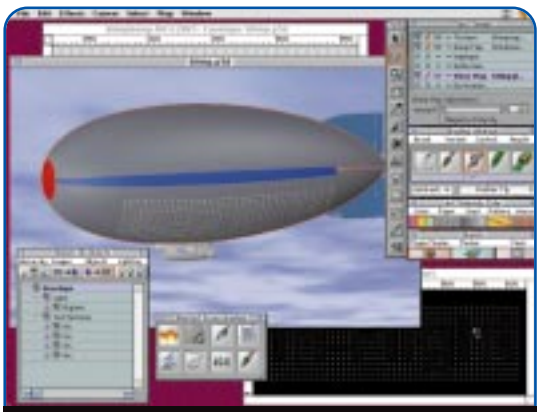
MORE CHANNEL WOES. Interplay announced that its upcoming title, OF LIGHT AND DARKNESS: THE PROPHECY, is being refused by several unnamed national retail chains on the grounds that the box art is too provocative and unrelated to the game itself. The game, developed by Interplay's Tribal Dreams division, features box art from artist Gil Bruvel and shows what looks like an angel in the fetal position. No word yet as to how Interplay will try to defuse the situation with its channel partners, if at all.

FOLKS AT CAPCOM MUST BE SMILING, as the company's recently released RESIDENT EVIL 2 blew away even the most optimistic sales projections. In its debut weekend in late January, the PlayStation title sold 380,000 units (valued at over $19 million), amounting to more than

Painter 3D

METACREATIONS just announced the Spring 1998 availability of Painter 3D, a painting tool for 3D models based on Fractal Design Detailer.

Painter 3D allows you to paint directly on computer generated models in real time, and features over 100 brushes and image-editing effects to help you create surface maps for 3D objects and characters. One of the key new features included with Painter 3D is simultaneous map painting, which allows you to create multiple maps — such as texture and bump — at the same time using the same strokes. Map sharing will allow you to attach one image map to multiple objects, so as you make changes, they will be applied to all objects. If, for example, you're working on a spider model, you'll be able to apply one map to all of the leg objects at once. Multiple object management will let 3D designers isolate individual items and view their hierarchy with image-to-object relationships. Additionally, new model-view controls and user interface enhancements will allow you to customize palettes to suit your working style.



MetaCreations's Painter 3D features easily customizable palettes.

Painter 3D will run on Windows 95/NT and Power Macintosh. It has a suggested retail price of $449 and an estimated average selling price of $299. MetaCreations will offer an upgrade path from Detailer to Painter 3D for $149. If you purchased Detailer after Feb 10, 1998, you're eligible for a free upgrade.

■ MetaCreations
Carpinteria, Calif.
(805) 566-6200
www.metacreations.com

Miles Sound System 4.0

RAD GAME TOOLS has unveiled the Miles Sound System version 4.0.

Two of the biggest new features in the Miles Sound System are full DLS-1 MIDI support and integrated ADPCM compression support. DLS-1 support. removes two of the biggest hassles when using MIDI — the limited General MIDI instrument set, and inconsistencies between different vendors' GM sets. With DLS, you ship your own instruments with your MIDI file — they sound exactly as they did when you authored them, and exactly the same on every sound card. ADPCM compression gives you 4-to-1 data compression. The Miles Sound System decompresses the ADPCM data on-the-fly in the digital sound mixer, so the data doesn't have to be decompressed before playing it. This feature will really reduce an application's audio RAM budget. The Miles Sound System includes an integrated software-synthe-

sizer, support for S3 hardware DLS-cards, tools to compress DLS sample files, tools to extract only the instruments used out of a generic DLS file, and tools to merge MIDI files with instruments into one song file. The Miles Sound System also includes the Sound Player and Sound Studio applications, both of which can be downloaded from the company web site.

The Miles Sound System sells for $3,000 per game, and $7,500 per site.

- **RAD Game Tools**
  Kirkland, Wash.
  (801) 322-4300
  www.radgametools.com

INSIDETRAK HP

POLHEMUS recently released INSIDETRAK HP (High Precision), a PC-insertable tracker, and an addition to its line of 3D position tracking, motion capture, and digitizing technology.

The INSIDETRAK HP eliminates the need for a chassis, power supply, and cabling because it will plug directly into the ISA slot on your PC — so it's relatively easy to use. The device computes the position and orientation of a tiny receiver as it moves through 3D space. The receiver's position and orientation are measured dynamically, which allows the data to be updated continuously, discretely, or incrementally. Further, the INSIDETRAK maintains a clear line-of-sight between transmitter and receiver by utilizing low-frequency magnetic transducing technology.

INSIDETRAK HP is PC-compatible, and is available for $2,495. This includes a receiver with a 20-foot cable, a transmitter frequency module, and a transmitter. An optional second receiver can be added for applications requiring extended capabilities.

- **Polhemus**
  Colchester, Vt.
  (802) 655-3159
  www.polhemus.com

The Incredible Comicshop

DIGIMATION has announced the latest addition to its MAXimizer series, The Incredible Comicshop.

Yet another plug-in enhancement for 3D Studio MAX, The Incredible Comicshop is a 2D cel-renderer that renders your 3D scenes so that they appear as 2D cartoons. Basing itself on traditional 2D-cel animation, Comicshop has three material types at its heart — Ink & Paint, Cel Paint, and Cel Ink. Using these tools, you can tweak the look of the surfaces and edges of models and create outlines that don't have the standard computer-generated look and feel. You can also easily manipulate variable line weights to lend a hand-drawn quality to your images. Further control comes from three levels of paint shading and full transparency and mapping abilities for both inks and paints. You can also mix and match 2D and 3D elements in in a single pass by changing paint materials. Comicshop ships with Shadow/Light from Blur Studios — this map type provides more control over the Ink and Paint blending process, and can help you produce anime-style images and animation.

The Incredible Comicshop works with both MAX 1.2 and 2.0. It retails for $695.

- **Digimation**
  St. Rose, La.
  (504) 468-7898
  www.digimation.com

60% of its initial production run. That beats out debuts by hits like SUPER MARIO 64 and FINAL FANTASY VII. To put some context to its opening weekend numbers, the game grossed more that weekend than nine of the top 10 movies.

HARDWARE-ACCELERATED IPO. nVidia, maker of the RIVA 3D accelerator found in Diamond and STB graphics boards, will soon be going public. The proceeds from the IPO are expected to be used for working capital and capital expenditures, and should put the company on better financial footing to compete against competitors like 3Dfx, Rendition, and PowerVR.

RIVEN @ 1,000,000. I know you're tired of hearing about RIVEN. So I'll make it short and sweet: it just surpassed the one-million-units-sold mark. In related news, Robyn Miller left Cyan to start his own film company called Land of Point, but we've been assured that there are no plans for a movie based on either MYST or RIVEN.

DVD-HMMM. Market research firm InfoTech projects that the installed base of DVD-ROM-equipped PCs will reach 7 million by the end of this year. Worldwide, the company says that DVD-ROM title revenue will rise from $3.5 million in 1997 to a whopping $567 million in 1998. InfoTech says most of the publishing revenue from DVD-ROM titles this year will come from royalties on bundling deals with upgrade kits and PCs.

MINDSCAPE SOLD. The Learning Company is acquiring Mindscape for a cool $150 million in cash and stock from its British parent, Pearson. Pearson has owned Mindscape since 1994, and took a $346 million loss on the sale.

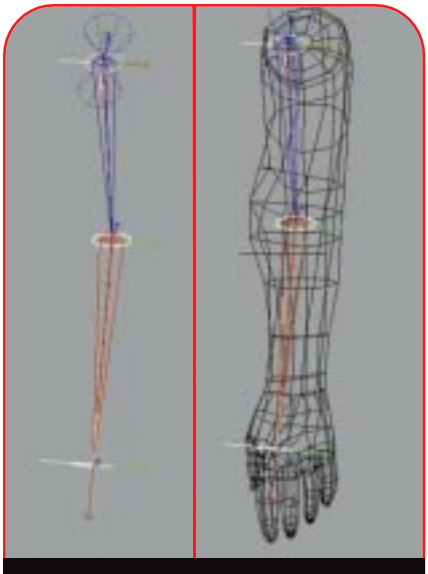
FIRST IT'S TIGER, NOW IT'S HULK. EA penned an exclusive license with World Championship Wrestling (WCW) to develop games based on WCW wrestlers like Hulk Hogan and "Macho Man" Randy Savage. The first WCW title From EA will appear in mid-1999, according to the company. Larry Probst, EA's CEO, was quoted in a press release saying "World Championship Wrestling is a huge spectator sport." I think EA forgot how to spell "spoof."

Skin Them Bones: Game Programming for the Web Generation

Well, it's true. The Internet has really changed things, and I don't mean in the way the news has been hyping it. Although it hasn't quite lived up to all of the media hoopla, the Internet has changed the way people communicate.

This is particularly true of accessing information. When I was initially learning 3D graphics for display on my lowly Apple II (and then on my Amiga), I really had to dig. I was fortunate to live near several major universities, and when SIGGRAPH was in Anaheim, it was right in my backyard. I would also comb through magazines and journals trying to figure out what the heck was going on. The books were never up-to-date on the latest techniques, and the people who were working on the coolest things were scattered all over the world. I could never afford to attend the seminars and symposiums where the professors met and compared notes, so I waited for the printed word to get back to me.

Today, all of this information is at your fingertips. Most journals and papers are now available directly online. All researchers post their papers on their own web pages before they're published in print. Even better, the people creating this work post their e-mail addresses on these pages. Now I can read up on the latest and greatest. If I have any questions, I just ask the author. Most impressive of all, the majority of these authors get right back to you and are flattered that you find their work interesting. Imagine being a kid in California and hearing about an Englishman named Newton. This guy has just come up with some interesting ideas about how things react when they bump into each other. So, because you're trying to make a pinball game, you fire off an e-mail about the problem. Newton fires back a quick explanation of his third law of motion, complete with animated .GIFs of things bouncing into each other. I can't wait to see what this generation of kids will come up with.



FIGURES 1 and 2. *First the bare arm skeleton, then with skin applied.*

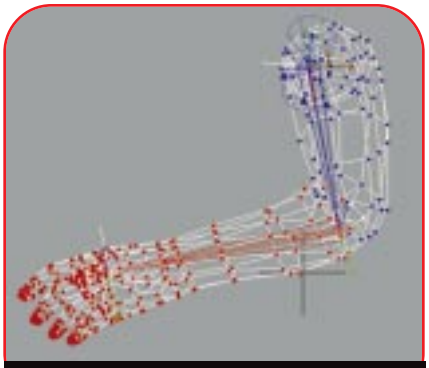


FIGURE 3. *The arm skeleton once it has been weighted and deformed.*

Bend without Breaking

So this brings me to my current problem. In my last column, I wanted to deform a skin mesh to a set of bones in a hierarchy ("Better 3D: The Writing Is on the Wall," April 1998). A real-time 3D character created from a single deformed mesh looks much better than one made up of separate objects. Every major 3D graphics animation package has a method of deforming a single mesh object. Most of them work by embedding some form of bone system inside the character, then using these bones to influence the shape of the mesh. This is the approach that I wanted to take for my character animation.

Not wanting to build a bridge where there was already a tunnel, I hit the books. Just by looking through the SIGGRAPH proceedings and hitting the

When not bending the bones of some strange alien creature, Jeff can be found hanging out at his studio at the beach. See if you can smack some sense into him by writing to jeffl@darwin3d.com.

Web, I came up with a whole bunch of stuff on character animation. These sources provided a good start, but weren't quite right for my purposes. I followed up on some references in those papers and still wasn't satisfied that I had found anything that directly applied. So, I fired off some e-mail to the different authors and some colleagues, asking if they knew any good sources for information. Amazingly, I received over an 80 percent return on those e-mails — including responses from some of the biggest names in computer graphics over the past decade. I don't know why I was ever intimidated by asking questions of the people best suited to answer them. Every one of them helped and encouraged me. Within a week, I was plowing through a pile of information and suggestions.I encourage everyone to ask questions, but keep in mind that you should be willing to reciprocate.

My basic approach was pretty sound. I really like the way Softimage handles skeletal deformation. It allows you to individually weight a vertex in a mesh to any bone in a skeleton. These weights represent the degree of influence each bone has on the final position of that vertex. This allows me a much greater degree of control than if I were working with a system that only had a sphere of influence with a falloff. My research convinced me that if I were to build a real-time system for displaying these weighted meshes, I could create quite compelling 3D characters. As a bonus, I could use the weighting interface from Softimage and preview how the animated character would look in the game.

For my sample mesh, I created a two-bone hierarchy to represent my arm (Figure 1). The blue bone represents the upper arm, and the red bone represents the lower arm. I then attached the mesh for the arm to this two-bone hierarchy (Figure 2). Applying the weights to each vertex and rotating the lower arm produced a deformed mesh (Figure 3). I took special care in weighting the vertices near the joint between the two bones. If I allow one bone to completely influence (weight 100 percent) the vertex position, then it's possible that in certain orientations, the mesh will fold in on itself. You'll achieve better results when each bone contributes to the final vertex position. Figure 4 shows the Softimage interface for editing vertex weights. This is an example of an individual vertex being weighted between two bones.

Once I'd the weighted mesh, I needed a way to perform the deformation. I created the prototype for the deformation engine in OpenGL. OpenGL can get this type of tool up and running very quickly. From there, you can easily port the routines over to the API or platform of your choice. As an added benefit, the resulting image in the tool is identical to the preview window in Softimage because Softimage uses OpenGL for its real-time display. When you're trying to develop and debug pathways, this eliminates one source of image problems.

DisplayLists has proved itself very effective for drawing static geometry. For my application, I wanted to display the bones in the user interface in the same way that Softimage draws them (as a diamond shape). The code to create the bone geometry DisplayList appears in Listing 1. The routine descends the hierarchy and creates the diamond-shaped display if a bone has a child. I used the y translation element of the child to determine how long the bone should be. To make it easy to access the DisplayList later, I used the ID for the bone as the list number. You can see the results of a two-boned arm hierarchy in the OpenGL application in Figure 5.

Matrix Fun

My process for deforming the mesh was simple. I calculated the position of each affected vertex as if it were completely under the influence of each bone. I then used the weighting values to interpolate between these positions. Let's look at that in different terms.

For each vertex
finalPosition =
(position[1] * weight[1]) +
(position[2] * weight[2]) + …

where each position[N] is the initial position of that vertex multiplied by the transformation matrix of bone N.

However, to efficiently calculate the position of each vertex as it would be transformed by each bone, I needed to know each transformation matrix in the hierarchy. I obviously didn't want to recalculate the matrix for each bone at every vertex. So during an initial pass, I stored the transformation matrix as it accumulated down the hierarchy. The OpenGL method of handling a hierarchy of transformations via a matrix stack is very efficient — as you may remember from my article on motion capture ("Working with Motion Capture File Formats," January 1998).

The call to get the current matrix is

```
glGetFloatV(GL_MODELVIEW_MATRIX, float
*matrix);
```

This returns the sixteen values that make up the current matrix. The values are laid out as follows:

$$M(v) = \begin{bmatrix} 00 & 04 & 08 & 12 \\ 01 & 05 & 09 & 13 \\ 02 & 06 & 10 & 14 \\ 03 & 07 & 11 & 15 \end{bmatrix}$$

Note that this representation (called column-major) is different from many matrix routines you may see (usually row-major). Because of this difference,

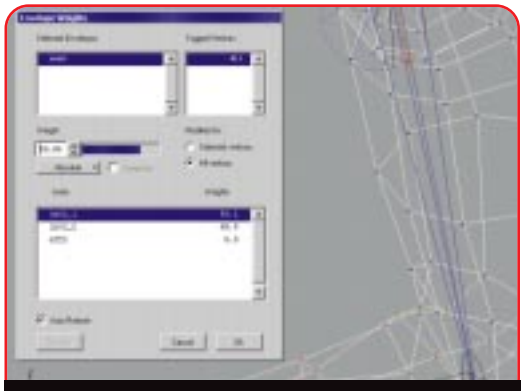


FIGURE 4. *Softimage interface for editing vertex weights.*

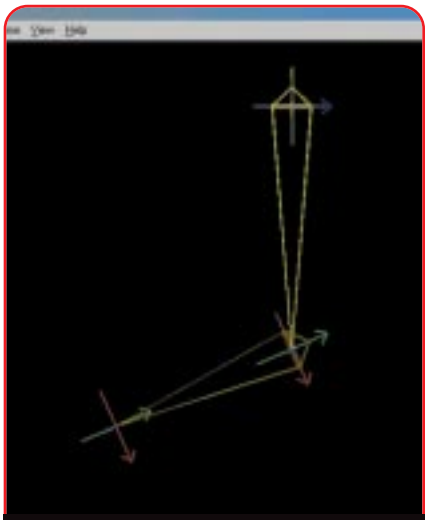


FIGURE 5. *Displaying the skeleton.*

if you declare the matrix structure to be **float matrix[4][4]** you'll get the wrong result when trying to access the data in C. It's much better to use **float matrix[16]** for your matrix storage. This is how OpenGL handles this procedure.

The matrices are stored in the bone structure for use when I actually calculate the positions. I have found this routine to be notoriously slow in OpenGL implementations. However, since the call to get matrix is only done once per frame for each bone in the system, it's not a big problem. For speed-critical applications, it may be wise to create your own matrix stack and matrix routines to speed up this process.

The code for saving the matrices is in Listing 2. It's a recursive call that will descend the hierarchy. At each node, it will draw an axis at the root of that bone. If that bone has a child, it will draw the bone geometry that I created earlier and highlight any selected bone. Note that the transformation operations are called in reverse order. OpenGL handles matrix operations this way. You may need to change this if you use a different API. This OpenGL feature causes a great deal of confusion for many people starting to work with the API.

At this point, I had to calculate the positions for each vertex. I could have called a **glLoadMatrix** for each bone. However, the call to **glLoadMatrix** is particularly slow (it would be called for each bone for each vertex). I wanted to avoid this one, because it would be called for each bone for each vertex. I could've avoided the issue by multiplying out all the vertex positions by each bone and storing all the intermediate results. However, that approach can be a huge memory issue for a mesh of significant size, so I decided against it. I chose instead to implement my own **MultVectorByMatrix** routine to calculate the intermediate positions. The drawback to this method is that you loose any benefit from 3D transformation hardware that you may have. This isn't an issue for consumer 3D hardware cards because they don't have hardware transformation acceleration. This may change in the future or in specific applications, so you'll have to evaluate the costs and benefits for yourself.

I multiplied each vertex by the matrix for every bone that vertex influences. I needed to subtract the root position of each bone from the vertex first in order to be sure that it was rotated about the bone's base. I calculated this distance back in during the matrix multiplication. I combined the results of all these calculations using the weight values to arrive at the final position for each vertex. I now have a mesh that is deformed in world space according to the settings of the bones controlling it. This mesh can be drawn as any other 3D mesh object. You can see the results in a standalone OpenGL application in Figure 6.

FIGURE 6. *OpenGL deformed mesh.*

Is it Worth it?

I now have a mesh object that can be deformed realistically in real time. This realism adds a lot of flexibility to your application. You can use it to create very compelling characters that react to their environment. But all this flexibility comes at a cost. Each vertex that is affected by more than one bone requires more calculations. The interpolation code and all the extra matrix handling add to the burden. I wouldn't even use these techniques on an enemy character whose entire motion sequence is finite and scripted. However, for a key character who can make unique moves that react to the people and environment around him, it's well worth it. The image quality generated by a weighted, deformed, single mesh is significantly

LISTING 1. *Displaylists for skeleton.*

```
////////////////////////////////////////////////////////////////////////////
// Function:        CreateBoneDLists
// Purpose:         Creates the Displaylists for the Bones in a Skeleton
// Arguments:       Pointer to a bone hierarchy
////////////////////////////////////////////////////////////////////////////
void COGLView::CreateBoneDLists(t_Bone *bone)
{
   // ONLY MAKE A BONE IF THERE IS A CHILD
   if (bone->childCnt > 0)
   {
      // CREATE THE DISPLAY LIST FOR A BONE
      glNewList(bone->id,GL_COMPILE);
         glBegin(GL_LINE_STRIP);
                  glVertex3f( 0.0f,  0.4f, 0.0f);                    // 0
                  glVertex3f(-0.4f,  0.0f,-0.4f);                    // 1
                  glVertex3f( 0.4f,  0.0f,-0.4f);                    // 2
                  glVertex3f( 0.0f,  bone->children->trans.y, 0.0f);          // Base
                  glVertex3f(-0.4f,  0.0f,-0.4f);                    // 1
                  glVertex3f(-0.4f,  0.0f, 0.4f);                    // 4
                  glVertex3f( 0.0f,  0.4f, 0.0f);                    // 0
                  glVertex3f( 0.4f,  0.0f,-0.4f);                    // 2
                  glVertex3f( 0.4f,  0.0f, 0.4f);                    // 3
                  glVertex3f( 0.0f,  0.4f, 0.0f);                    // 0
                  glVertex3f(-0.4f,  0.0f, 0.4f);                    // 4
                  glVertex3f( 0.0f,  bone->children->trans.y, 0.0f);          // Base
                  glVertex3f( 0.4f,  0.0f, 0.4f);                    // 3
                  glVertex3f(-0.4f,  0.0f, 0.4f);                    // 4
         glEnd();
      glEndList();
      // CHECK IF THIS BONE HAS CHILDREN, IF SO RECURSIVE CALL
      if (bone->childCnt > 0)
         CreateBoneDLists(bone->children);
   }
}
```

better than a character composed of separate objects, or whose joints are simply skinned over. Also, since you only need to store the orientations of the base skeleton, you can save a lot of memory on animation over straight, predeformed, single mesh characters.

The Application

The sample application that accompanies the article allows you to play with a deformable mesh. You can control the orientation of the bones as well as adjust the weighting on individual vertices. This will allow those who don't have access to Softimage to adjust the weighting on a deformable mesh and see the results. The arm itself is composed of an interleaved array of triangles. Each triangle is vertex colored to create a realistic shaded look.

I also used OpenGL's feedback mechanism to allow you to select vertices. I don't have space to cover that now.

Next issue I will address feedback as well as some other user interface issues.

Grab the source and the executable at the *Game Developer* web site at www.gdmag.com. ■

RESOURCES

I never found any one source that applied to the techniques I was using, but many resources were very helpful. If you're interested in learning more, check out these publications:

Badler, Norman, et al. *Simulating Humans: Computer Graphics Animation and Control*. Oxford: Oxford University Press, 1993.

Badler, Norman and M. A. Morris. "Modelling Flexible Articulated Objects." Computer Graphics, Proceedings of the Online (1982): pp. 305-314.

Magnenat-Thalmann, N. and D. Thalmann. *Interactive Computer*. Upper Saddle River, N.J.: Prentice Hall, 1996.

Parke, Frederic and Keith Waters. *Computer Facial Animation*. Wellesley, Mass.: A. K. Peters, 1996.

Terzopoulos, Demetri, et al. "Elastically Deformable Models." Computer Graphics, Vol. 21, No. 4 (SIGGRAPH 1987): pp. 205-14.

Acknowledgements

Thanks to the many people who have contributed to my knowledge of these techniques and methods over the past six months. Here are a few of them: Paul Atkinson, Norman Badler, Paul Douglas, Chris Hecker, Hexapod, Frederic Parke, Demetri Terzopoulos, and Nadia and Daniel Thalmann.

LISTING 2. *Grabbing the* `ModelViewMatrix`.

```
////////////////////////////////////////////////////////////////////////////
// Function:        drawSkeleton
// Purpose:         Actually draws the Skeleton it is recursive
// Arguments:       None
////////////////////////////////////////////////////////////////////////////
GLvoid COGLView::drawSkeleton(t_Bone *rootBone)
{
/// Local Variables ///////////////////////////////////////////////////////
   int loop;
   t_Bone *curBone;
////////////////////////////////////////////////////////////////////////////
   curBone = rootBone->children;
   for (loop = 0; loop < rootBone->childCnt; loop++)
   {
      glPushMatrix();

      // Set base orientation and position
      glTranslatef(curBone->trans.x, curBone->trans.y, curBone->trans.z);

      glRotatef(curBone->rot.z, 0.0f, 0.0f, 1.0f);
      glRotatef(curBone->rot.y, 0.0f, 1.0f, 0.0f);
      glRotatef(curBone->rot.x, 1.0f, 0.0f, 0.0f);

      // THE SCALE IS LOCAL SO I PUSH AND POP
      glPushMatrix();
      glScalef(curBone->scale.x, curBone->scale.y, curBone->scale.z);

      // DRAW THE AXIS OGL OBJECT
      glCallList(OGL_AXIS_DLIST);
      // DRAW THE ACTUAL BONE STRUCTURE
      // ONLY MAKE A BONE IF THERE IS A CHILD
      if (curBone->childCnt > 0)
      {
         if (curBone == m_SelectedBone)
                  glColor3f(1.0f, 1.0f, 0.0f);   // Selected bone is bright Yellow
         else
                  glColor3f(0.4f, 0.4f, 0.0f);   // Selected bone is dull Yellow
         // DRAW THE BONE STRUCTURE
         glCallList(curBone->id);
      }

      // GRAB THE MATRIX AT THIS POINT SO I CAN USE IT FOR THE DEFORMATION
      glGetFloatv(GL_MODELVIEW_MATRIX,curBone->matrix);

      glPopMatrix();          // THIS POP IS JUST FOR THE SCALE

      // CHECK IF THIS BONE HAS CHILDREN, IF SO RECURSIVE CALL
      if (curBone->childCnt > 0)
         drawSkeleton(curBone);

      glPopMatrix();          // THIS POPS THE WHOLE MATRIX

      curBone++;
   }
}
//// drawSkeleton ////////////////////////////////////////////////////////
```

Should Game Artists Go to Conferences?

I had just started working as an artist in the game industry. Ned Lerner (president of Multitude and a major name in game development) had hired me to create art for his driving game, CAR & DRIVER, and in his mellow "Sure, why not?" way, he suggested I come by and check out the 1991 CGDC. Back then, I was too poor to afford a pass (and I didn't know you could volunteer and get a pass in exchange, because I didn't know anyone in the industry), so I just cruised around looking for nothing. And that's what I found. People streamed by me, and I felt outside and unwelcome. I left after a couple of hours, wondering who would bother going to these things. It seemed so pointless — a bunch of closed-door meetings with coders and designers, not much for a technical artist doing this weird real-time artwork. So I just went home and built more 3D racetracks with my hacked-up art tools.

Without a definite agenda, it's easy for an inexperienced conference attendee to get lost in the crush of people.

This was really too bad, because I needed the connections at that show. My project wasn't the only real-time 3D project (though it was one of the few — this was during the Sega Genesis's heyday). I could've learned a lot by talking to other artists who were building low-polygon models. Like what? Well, better tools for starters. I was using AutoCAD to build terrains, and I'm sure there were better options even then.

I also needed to compare techniques with other people. I was struggling with building some of the first BSP-sorted terrains and other weird, messy, problematic 3D artwork and I didn't know any other artists dealing with these things. I'm sure other people had already solved the problems I was facing, but I didn't know them. I probably could have met some at that conference. For example, I could have tried to find the artists who made TEST DRIVE III (a driving game similar to mine) and talked with them.

The most important opportunity I missed was the chance to secure other job contracts. I was working for only one

Josh White runs Vector Graphics, a real-time 3D art production company. He wrote Designing 3D Graphics *(Wiley Computer Publishing, 1996), he has spoken at the CGDC, and he cofounded the CGA, an open association of computer game artists. You can reach him at column@vectorg.com.*

Conferences provide opportunities to learn about new tools, share knowledge, and establish new contacts within the industry.

company at the time and didn't need any more work right then. But when the project was finished, I was suddenly out of a job. Because I didn't know anyone in the industry, it was really hard to get more work. When I discovered later that talented contract artists can just walk up to art directors at conferences, I was amazed. It saved everyone the hassle of the normal job-searching strategies. It was like an old-boys' network, except any attendee could do it — they didn't care if you knew them already. If you could show that your work was good, you got a chance.

So, here's why artists should go to conferences:

- Meet other artists.
- Learn about new tools.
- Share knowledge and get recognition.
- Get industry gossip.
- See cool game art.
- Find art contracts.

How to Get Something Out of a Conference

You're an artist at a major trade show, wondering what's worthwhile. What should you do and see? How do you avoid wasting your time? I've listed my strategies here. (What are yours? E-mail me your favorites.)

I visit the show floor at least once, hang out at the bar, attend at least two classes/roundtables, try to find cool parties, and hang out with my friends.

EXPOSITION (SHOW FLOOR). Trade-show floors are totally exhausting, but they can be really great sources for new information amid the frothing hype. Few artists can stand browsing an entire floor and still feel bouncy afterwards. More often, artists just wander, snarling at salespeople and half-heartedly poking at the demos, then sulk out the exit in 15 minutes. Obviously, these visits aren't worth much.

So, how do you prevent burnout and get good information fast? Limit your exposure by seeking specific answers and cutting through the hype.

Set a goal. Before you walk onto the floor, dredge up some specific problems you're facing at work. In an ideal world, you would bring a list of work problems to the show, but let's be realistic — it's more likely that you'll wander in without any preparation. Seek an answer for that problem by visiting relevant booths. If your first few visits are way off-target, ask the salespeople for some guidance — they often know the show floor.

Copy others' notes. Ask everyone you meet, "What's cool on the floor?" and go see that. Of course, this means you'll miss the hidden gems in the small booths in the back.

Meet artists. If you're a shy type, this is one of the best opportunities you'll get to meet strangers at the show. Talk to other people visiting the booth. Offer opinions on what you're seeing. Ask them what cool stuff they've seen. Tell them about a party or where to get a free T-shirt. Ask them for cool events. And, if you're selling something, remember that it's obnoxious to visit competitors' booths and talk to people there about your product. Sounds obvious, but I've seen it happen.

Use the salespeople to get your information faster. Paradoxically, they're your quickest route through the hype. Instead of turning away from their gleaming smiles and welcoming overtures, ask them simple, clear questions to get information quickly. A couple of handy sentences are,

"So what does your company do?"

"Who are your competitors?"

"Are there any of your customers around that I can talk to?"

"I'm looking for art tools — 24-bit image animation editors."

Don't be patient with off-topic explanations. Gently but quickly interrupt if they're trying to sell you something you don't need. "Oh, I see, but I'm actually an artist, so I don't write sound drivers. Do you have any art tools?"

Of course, most artists who go to the show floor also spend time drooling over motion-capture suits and fantasizing about fancy art tools with five-figure prices — and that's excellent. It's a great place to flirt with the notion of adopting new tools. Even if you're not looking for any new tools, this is the your chance to give powerful feedback on your current tools. If you complain, you might learn that there's a plug-in or upgrade solution. Even if there isn't an immediate solution, those sales people often have a lot of control over the next generation of their tool. Don't you wish Photoshop's batch mode had a way to select certain files instead of a whole directory? This is your chance to tell Adobe, in person.

And finally, don't be a jerk out there. Many game developers seem to think that show salespeople are slimier than used-car dealers. In my experience, that's just not true. Most exhibitors aren't actually trying to get your money. No, they're trying to give the impression that their tools will solve all your problems forever, and that your company should pay them 10 gazillion dollars for their special solution. That means they want you to think they're the bomb. In other words, they usually don't want your credit card number —

they want your company's purchase order. They need your help to get that.

CLASSES. Game developers scorn classes, mostly on principle I think. I know that I definitely don't identify with the student role. Begging for crumbs of knowledge from the Master Lecturer makes me cranky. "Oh, please," I snort to myself. "As if I need help from those posers." Well, there are plenty of reasons to attend courses at conferences, despite my fat head.

- Develop an existing skill.
- Hear about a new skill.
- Catch industry trends.
- Understand your coworkers (go to a programming lecture to get the vibe, not for specific information).

There are two categories of official learning events at conferences. The usual kind involves somebody getting up on stage and yakking for an hour to an audience of 100 or so, then taking a few questions at the end — a lecture. There are a couple of variants. Seminar is a vague term that usually means that the question/answer (Q/A) session is longer, and sometimes also means more live demos. Panels put five people on-stage instead of one, and they try to argue about something. These are usually best when there's a good lively topic, and the panelists disagree with one another.

In the second type of learning events, the panel topic is turned inside out, and they call it a roundtable. It means that about 30 people walk into a room and talk about something. There's a moderator, who keeps the discussion on track and in control (prevents sales pitches and encourages everyone to speak).

One rung down the formality scale are Birds Of a Feather (BOF, for the acronym-obsessed) sessions. Though I personally detest the inane name, it's a very cool concept. These are unplanned sessions where someone at the conference wants more discussion and just puts up a sign: "BOF: 2D Artwork Techniques, room 999, Tues. 10AM." These are relatively unorganized and undirected, but usually have the spark of life.

PARTIES AND OTHER EVENTS. There are two kinds of conference parties: huge, sponsored affairs, and small, private events. I like small, invite-only events. I don't know why, but there's something magic about getting an invitation to an event — it makes attendees feel special, even though the parties are basically the same as the public ones.

Why bother with parties? This is how people personally connect (if the value of that is unclear, you need to make more friends). Aside from the sheer fun of meeting new and weird people, those personal connections can mean a lot when you're stuck with some problem and need somebody new to call and bounce ideas off.

Which Conferences Should You Attend?

There are lots of conferences to choose from, but my top five are:

CGDC. Computer Game Developer's Conference is the original. Bought by Miller Freeman a couple of years ago, it has about 4,000 attendees and is considered the key conference by most game developers. I'm very involved in it, so of course I think it's great.

E3. The Electronic Entertainment Expo is an unbelievable experience. It's mostly about the show floor, an overwhelming experience in which industry giants such as Sega and Nintendo duel it out with 30-foot projection screens and other such insanity. With huge attendance, it's a great place to meet people and see all the latest games, but it's not meant for learning how to make them as much as marketing and sales. I go when it's on the West Coast.

SIGGRAPH. SIGGRAPH is the granddaddy of computer graphics (CG, as it's known there) conferences. It's huge (40,000 people) and covers all kinds of CG: games, movies, scientific, education, research, simulation — you name it. It's successfully kept a very strong art focus despite being a technical CG Mecca. The papers presented there are legendary — it's the place to announce your new breakthrough in computer graphics. It also has a great trade show floor, and the parties and other events are wonderful.

IGDN DEVELOPER'S CONFERENCE. This conference is in March 1998 in Austin, and since I'm writing this in February, I think it's going to be great. There are great speakers, and I'm guessing it's going to be small and friendly compared to the larger conferences I've listed above.

DAVE MENCONI'S DEVCON. This conference is in April, and this is its first year. It's the first exclusive developers' conference I've heard of: you must be a game developer to attend. I have high hopes for this event too — I bet it'll be small and friendly and well worth attending.

Aside from these, you might want to check out other events during the year. There are many others that are worth considering attending: Comdex, NATE, AniFX '98, the 3D Design Conference, and so on. There's probably nothing wrong with these shows, but I can't go to every show there is.

Parties can provide the opportunity to meet other artists and see the latest in game art.

You Spoke!

Hey, my readers wrote me! I've gotten some great feedback at column@vectorg.com, some of which I think you'd find interesting.

Mike Kelleghan had this really cool addition to my article about communication triangles ("Communication Triangles," March 1998): "One trick that has helped me quite a bit whenever I've had a problem I'm trying to solve is to communicate to my team what the problem is, rather than my solution. In your article, Pam Publisher communicated the solution to her team (which I've done many times). Without knowledge of the problem she's trying to solve, the team can only guess at the reasons behind the communication. By allowing the team to participate in the problem-solving, you build a better team and usually get better solutions." Good point, Mike, and I agree. Thanks for telling us.

A while back, Tess Snider found my earlier feature article on artist/programmer interactions reposted on Gamasutra ("The Artist Synapse," August 1997). It was wonderfully long and detailed, so I must summarize. But she had some points worth repeating. "For the most part, I found the article fair and an interesting read. The only generalization that didn't seem to ring true in my mind was in the "Thinking Visually" section. Traditionally, the observations you made about programmer psychology have largely been true within the programming industry, but I think that this is something that is slowly changing. I think that we will see more intuitive programmers out there as the diversity of programmers increases (which will happen as there is more demand, and there is less of a social geek-stigma attached), as children begin to be encouraged to learn programming at a young age, and as teachers begin to learn that not everyone thinks like robots (which may never happen, but hey)." Though we cheerfully argued a few points, I really appreciate her writing. Thanks, Tess!

MAX 2 Review Addendum

And finally, I've gotten supporting feedback on my review of MAX 2 from many artists. Every artist I've talked to has supported my criticism of MAX 2's interface (I'm not the only one who doesn't like it). I was moved to write a plea to Kinetix about the UI problem. ■

An Interface Prayer to Kinetix

We unworthy users beseech you, our fearless and triumphant Market Leader: bless us with improved User Interfaces!

First, we beg you: full keyboard shortcuts for everything. Look unto Softimage for guidance, even if it hurts your well-earned pride.

Let your cool innovations, such as Strokes, continue to rain upon us — but first implement standard Windows UI features such as docking toolbars.

O Great Leader, Free us from the ironclad screen layout, and let us resize and dock our modeless dialogs. Help us preserve our screen real-estate with efficient white space and make everything on-screen optional. Make full use of the upcoming multimonitor support in Windows 98.

Release us from the demon mouse and its carpal-tunnel plague. Instead, allow us ten ways to access everything. For each command give us a keyboard shortcut, menu item, toolbar button, context-menu item, and MaxScript command. Allow us input via weird software (Strokes, voice input) and hardware (3D digitizers, mechanical monkeys, and MIDI knobs).

If this comes to pass, O Mighty Kinetix Developers, we will create more fantastic visions with joy in our hearts and your software at our fingertips. May your GPFs be banished, and your every assertion stand true for eternity!

FOR FURTHER INFO

3D Design Conference
www.3dshow.com

AniFX '98
www.natpe.org/anifx98

Comdex
www.comdex.com

Computer Game Developer's Conference
www.cgdc.com

Dave Menconi's DevCon
www.menconi.com/devcon.htm

IGDN Developer's Conference
www.igdn.org

NATE
www.etshows.com

SIGGRAPH
www.siggraph.org

3D Graphics Hardware

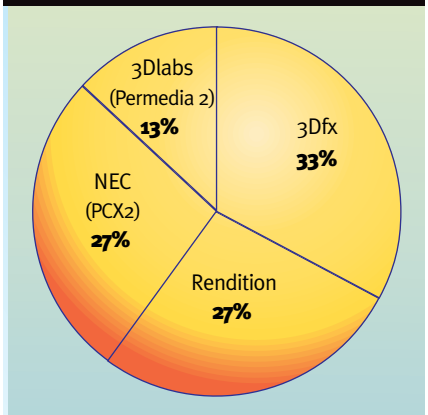
Trends in the hardware market impact the features and technologies that go into tomorrow's games, and as PC technologies leapfrog over each other at a dizzying pace, the effect can be bewildering to game developers. It can be difficult to know which features to try to take advantage of in the latest and best hardware, and just as hard to predict whether the target audience will choose to adopt (or in some cases, be able to adopt) new hardware.

This is the first of a series of columns that will address that problem. I'll explore, from an industry analyst's perspective, the fashions in consumer devices, components, and systems that affect game developers. It seems fitting, therefore, to kick off this first column by looking at the trendiest industry affecting the user base, 3D graphics acceleration.

The Players

At one time, it seemed as if the 3D graphics market would become a quagmire of competing companies with little differentiation in product, except in name, as companies strove to take advantage of opportunities arising out of DirectX and Direct3D. At present, a number of trends are indicating that 1998 will be a very interesting time for the graphics vendors and the people who follow them.

FIGURE 1. *Performance graphics chips: market share by company.*



Oppenheimer & Co. Inc. estimated that in the first half of 1997, based on unit shipments the following companies split the market for performance graphics chips (Figure 1).

3Dfx. With the release of Voodoo², and deals with Diamond and Creative Labs, 3Dfx has set itself up as the premier solution for gamers. The company has successfully built brand name recognition for itself and its architecture by assiduously courting game developers and players alike. The fact that 3Dfx has been purely focused on accelerating 3D games, has avoided the mass market graphics controller market, has proven itself to be the performance leader in both Direct3D and OpenGL games, and has used its own proprietary API, Glide, to build its own software support structure, have all helped to keep the company ahead of the pack.

nVidia. The RIVA 128 from nVidia is probably the best combination of 2D and 3D graphics performance on the market. nVidia is looking to go public this year, and will be very aggressive in maintaining its momentum in the market and increasing PC OEM support for its chipset. And even the company's announced support of OpenGL is targeted at the games enthusiast market as well. In some ways, this marks nVidia as a more interesting company to watch than 3Dfx, since nVidia is going after both the retail upgrade market, exemplified by PCI-based systems, and the new AGP-enabled machines of brand-name PC OEMs such as Gateway 2000. It will be more difficult for 3Dfx to evolve from a pure 3D company to a fully fledged graphics chip company — were they interested in doing this — but nVidia has the opportunity to come close to 3Dfx in terms of performance, and offers more flexibility in design and support to its OEM and end users.

NEC. NEC has both resources, as it has repeatedly shown, and a new architecture in its second version of PowerVR. The company is poised to release five new chips this year, targeting the console, PC, and arcade markets. Since taking its lumps from developers after the first generation of PowerVR chips, it has tried to turn perception around. It's now touting itself as an "invisible technology" for developers and emphasizing the lower price points for its hardware.

Rendition. Once seen as being in step with 3Dfx as far as the game player audience is concerned, Rendition has dropped out of the limelight. The company's Vérité 2200 did not hit its original production dates — the chip was slated for release in the middle of 1997, shipped late, and consequently hurt Rendition leading up to the Christmas buying season. As a result, Rendition had to fall back to a lower-class offering, the Vérité 2100, which supported a slower RAMDAC.

Intel. With Intel stepping into the mar-

Omid Rahmat works for Doodah Marketing as a copywriter, consultant, tea boy, and sole employee. He also writes regularly on the computer graphics and entertainment markets for online and print publications. Contact him at omid@compuserve.com.

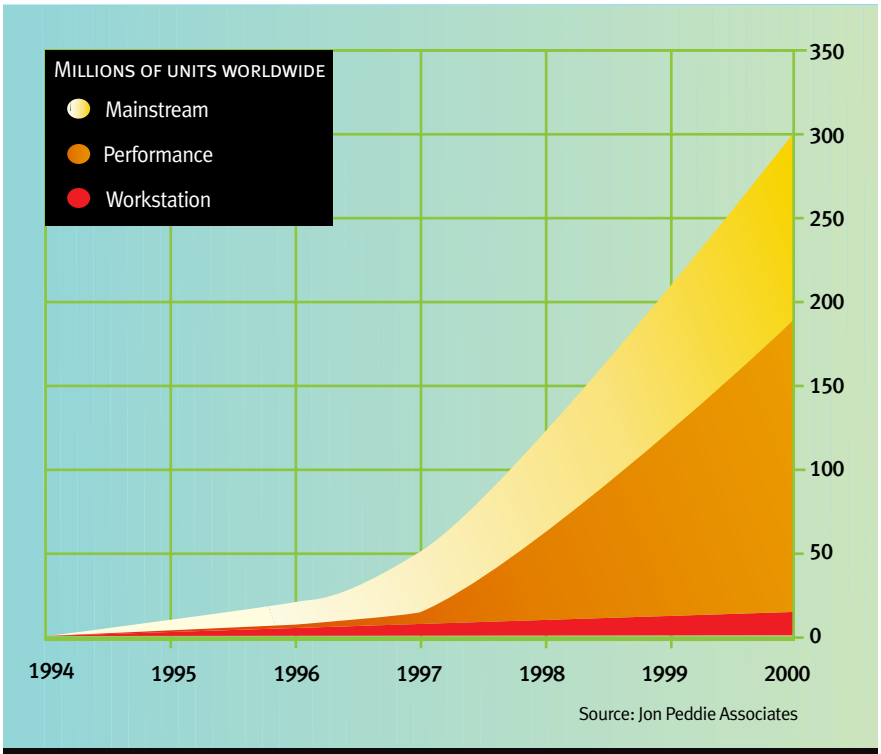


FIGURE 2. *Historic and forecast installed base of 3D graphics controllers by performance category.*

ket via the Intel740, companies may either feel pressure to increase their role in mainstream markets or to push into the higher end of the consumer PC market — the games enthusiast segment. In most cases, Intel's presence will be beneficial to game development on the PC platform. Intel's objective is to raise the performance bar for graphics in order to keep the industry abreast of its own rapid escalation of AGP and Pentium II adoption. The biggest question mark hanging over the PC industry revolves around the sub-$1,000 PC market and its impact on the consumer market. The most likely scenario is that lower-cost PCs will attract new PC users, and that existing buyers, who are predominantly game players, will be more interested in the higher end of the PC performance spectrum.

Growth Trends

Jon Peddie Associates, a market research firm in Tiburon, Calif., specializes in the PC graphics market and categorizes PC graphics chips into three classes:

1. Mainstream. By product, this category includes the major offerings of companies such as S3, ATI, Trident, and Matrox.
2. Performance. 3Dfx, nVidia, Rendition, PowerVR, Intel, Number Nine, and the Permedia 2 from 3Dlabs fall into this category, although the ATI Rage Pro is also said to fall into this category as an AGP solution.
3. Workstation. This category is reserved for the high-end, professional NT workstation market. It includes the offerings of 3Dlabs, Evans and Sutherland, Intergraph, and Dynamic Pictures.

Andy Fischer, vice president of Jon Peddie Associates, sums up the market growth influences on each segment. "In February 1998, after just seven months of production, nVidia and SGS-Thomson claimed to have shipped over 2.5 million RIVA 128 chips. Along with the extraordinary success of the ATI Rage Pro at the dawn of the AGP era, the adoption of 3Dlabs Permedia 2 in high-volume systems, and the strength of 3Dfx's Voodoo among game enthusiasts, we are seeing a quick ramp up of the installed base of very capable 3D controllers.

"Recent entries from Rendition and Chromatic Research, the just announced Intel740, and the anticipated next-generation PowerVR will all contribute to a sizable target market of approximately 50 million 'performance' 3D graphics controllers by the end of this year." (Figure 2)

Mr. Fischer has this to say to game developers, "Whereas in the recent past, developers were rightly cautious about targeting a PC platform that assumed the presence of capable 3D acceleration, this should no longer be an inhibition. The graphics battleground has moved to the third dimension, and system OEMs find themselves configuring even business desktop models with performance 3D accelerators. Graphics vendors who can't offer a competitive part to the RIVA, the Rage Pro, the Vérité, or the 740 simply don't get to play. Without the market demand generated by gamers, graphics vendors wouldn't have pushed 3D acceleration into the high-volume OEM segments."

Therefore, market statistics seem to bear out the notion that everyone is squeezing their leading products into the performance sector, and in doing so, higher-performance 3D graphics acceleration in hardware is fast becoming the norm, not the exception.

Performance Trends

Mercury Research of Scottsdale, Ariz., is another leading market research organization with expertise in both graphics and general PC components. The company regularly tests the latest graphics controllers as part of its subscription-only report, *PC Graphics Chip Sets and Technologies '98*. Mercury reveals that the PC graphics controller market topped the 100 million unit mark for the first time in 1997, shipping 102.1 million chips worth an estimated $1.4 billion. Of those chips, more than a third — or 36.2 million — included hardware 3D capabilities. For 1998, Mercury Research is forecasting shipments of 3D-enabled accelerators to more than double, to 75.3 million units.

Mike Feibus, a principal analyst at Mercury Research says, "Although 3D is grabbing all the headlines, systems makers continue to demand 2D-only accelerators as they scramble to find

ways to contain costs in the growing sub-$1,000 PC segment."

The graphics market is strongly driven by WinMark scores derived from the Ziff-Davis' WinBench graphics benchmark. (See Figures 3 and 4 for examples of Mercury Research's benchmarking tests.)* Dean McCarron, another principal analyst at Mercury Research says, "The main issue with WinBench is that it is very sensitive to a few key tests — especially the MIP-map tests, of which there are four. By simply adjusting the answers a graphics board driver gives during the MIP-map testing, you can significantly alter the test scores."

This has already created a great deal of controversy in the graphics business. Vendors are testing their systems in competitive situations, being very kind to the capabilities of their own system, and not so kind to those of their competitors.

Game enthusiasts have been highly critical of WinMark scores, and have chosen to look at frame-rate performance as a more objective measure of performance for their needs. However, there are some issues to deal with here as well.

McCarron explains, "The issue with frame-rate tests is that without some element of quality, which WinBench tries to address, you can have one controller that renders a frame with

FIGURE 3. *Mercury Research AGP-only tests.**

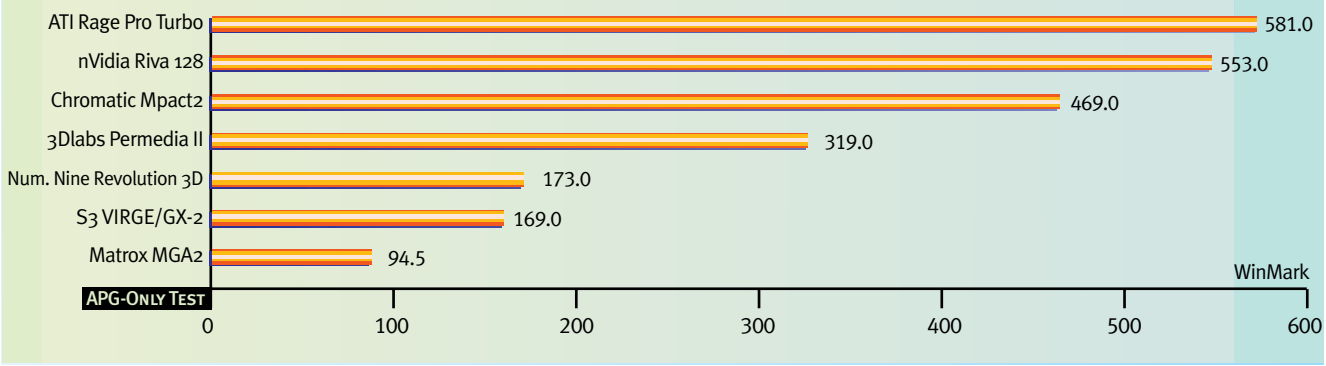
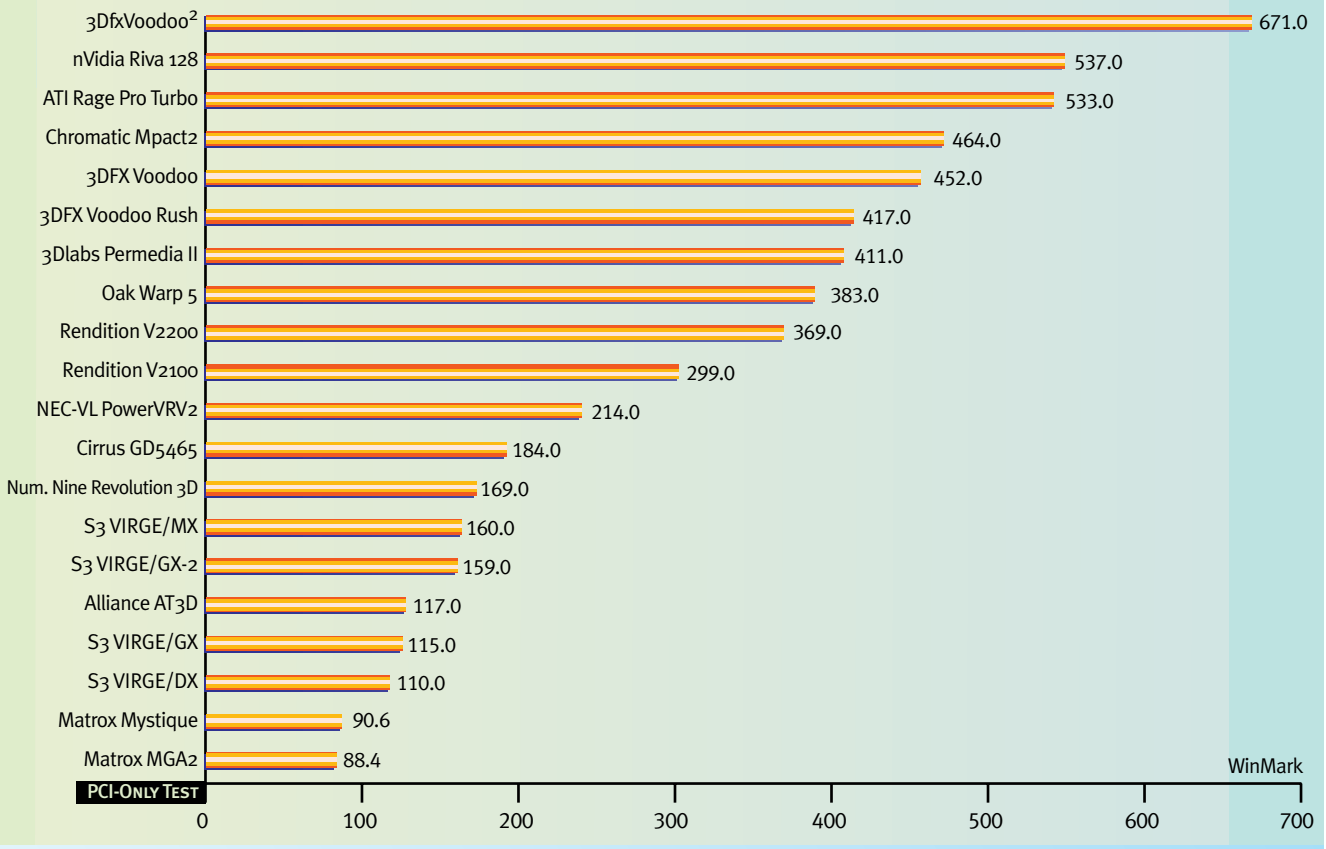


FIGURE 4. *Mercury Research PCI-only tests.**



all features and get a lower frame rate than some controllers that don't render all features."

In WinBench, the default is to render in software what cannot be done in hardware, which is meant to be a better way of handling the issue of quality. However, the feeling is that the graphics industry is going to go through yet one more round of benchmark wars, based on the tests Ziff-Davis puts out, and create all manner of confusion in the market.

Trends in the Channel

Ultimately, the sales channel is going to determine who gains the most in the 3D graphics market. One of the most recent dropouts from the 3D graphics accelerator market, Oak Technology, weighed the costs and risks associated with building a customer base, attracting premium software developers, and creating consumer demand, and decided that the market was too crowded for them.

At present, most of the sales activity by hard-core gamers happens through the retail channel — purchasing boards off the shelf. This is because many brand name PC makers have been reluctant to upgrade their multimedia consumer PCs with the addition of performance games graphics acceleration. The introduction of Pentium II and AGP systems in 1998 may shift 3D hardware sales towards OEM products, but it's too early to say how this will affect the installed base. ■

THE WINBENCH GRAPHICS BENCHMARK

The benchmarks contained in the report are based on Ziff-Davis' WinBench 98 version 1.0 and 3D WinBench 98 Version 1.0. When the term "Business WinMark" or "BizWinMark" is used, we are referring specifically to the version 1.0 Business WinMark 98 as performed by the WinBench 98 program. When the term 3D WinMark 98 score is used, we are referring specifically to the version 1.0 3D WinMark 98 as performed by the 3D WinBench 98 program.

The system configuration used as the Mercury Research Test Platform consists of a 300 MHz Intel Pentium-II with MMX Technology operating on the Asus P2L97 AGP motherboard with 64 MB of SDRAM, a Seagate ST36450A 6448MB hard disk, and IDE controller integrated into the Intel 440LX chip set with no hardware disk cache. An Ethernet card running 3COM's 10/100 PCI controller is present to supply device drivers to the system.

The graphics controller used for the testing changed with each test. Except where otherwise noted, all test were performed at a the default display refresh rates and resolutions as requested by the benchmark program; this is typically a 75Hz display refresh rate. The display resolution in all cases was 1024 x 768, 16-bit color for 2D tests and 640 x 480, 16-bit color for the 3D tests.

The CD-ROM environment consisted of the Mitsumi CRMC-FX600S 6-speed CD-ROM drive and used the existing IDE controller of the Intel motherboard previously described. The supplemental cache size for the CD-ROM driver under Windows 95 was set to large, resulting in a 1238K cache.

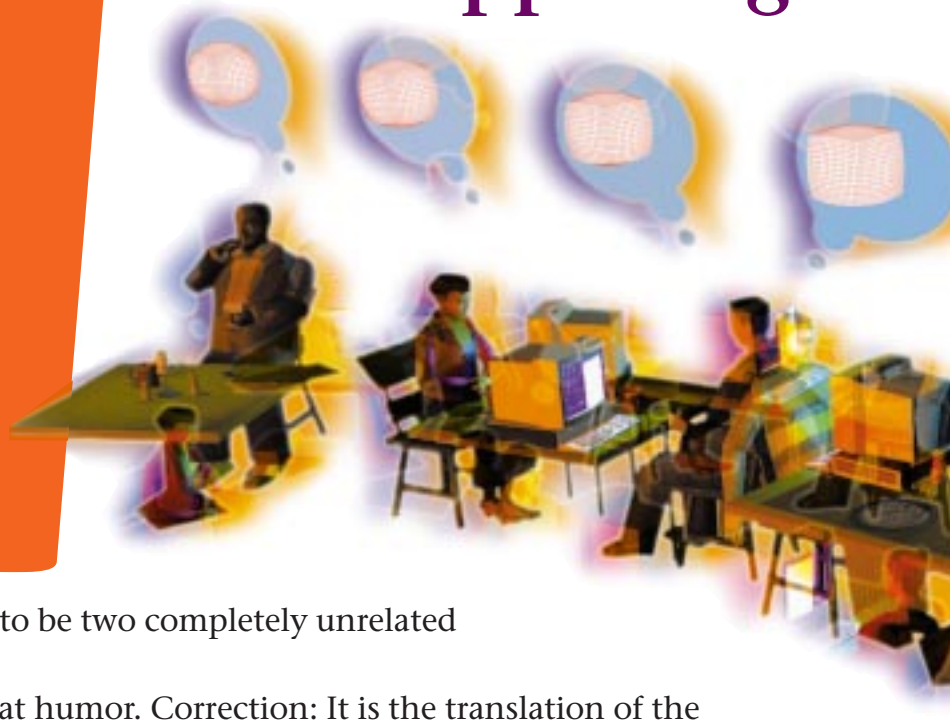
The operating system used for the test was Windows 95, version 4.00.950a (build 950) "OSR2," with a display resolution of 1024 x 768, 16-bit color. The file system is configured as a 32-bit system for a typical desktop, with maximum read-ahead optimization. All tests were performed under DirectX version 5, the most recent stable release at the time of testing.

All products tested were shipping versions at the time of the test, available to the PC OEM community and will be available to the general public after normal manufacturing delays or with sufficient demand. Most drivers were available to the general public via the internet as of January 20, 1998.

These tests were performed without independent verification of the score by Ziff-Davis and Ziff-Davis makes no representations or warranties of the results.

LOCALIZING BEYOND THE

"What's happening?"

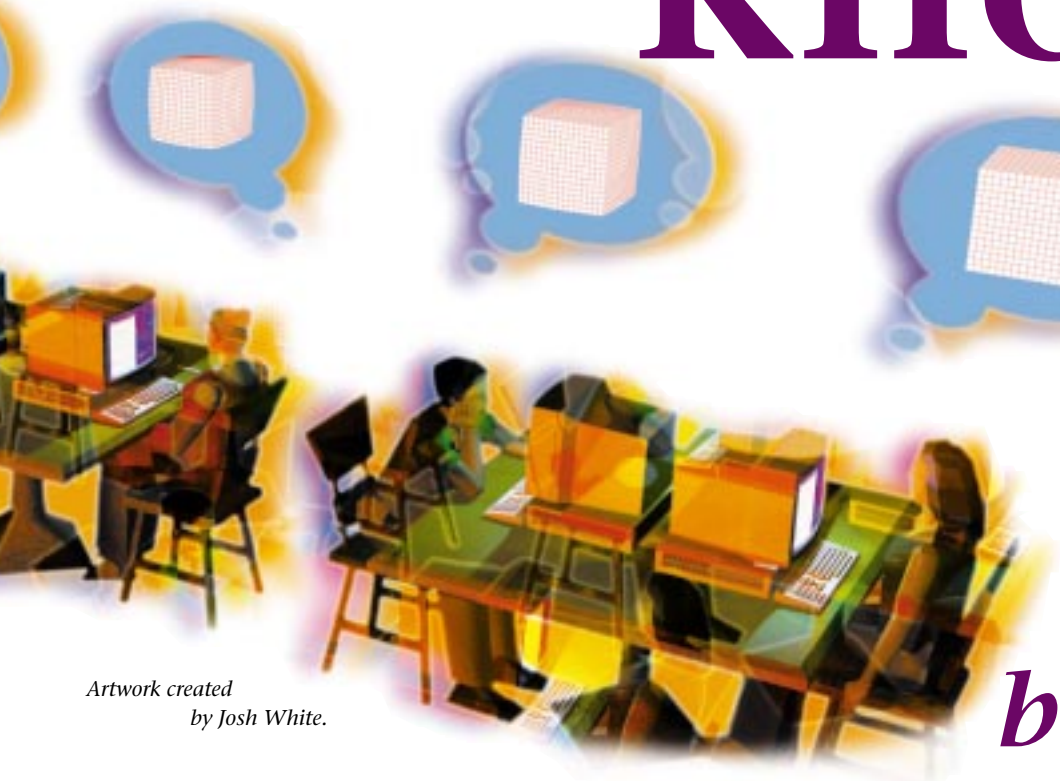


Huh? What may at first appear to be two completely unrelated sentences is, in fact, an attempt at humor. Correction: It is the translation of the translation of an attempt at humor. Either way, it doesn't make much sense. Sadly, this is an excerpt from the dialog of an adventure game, and as such, it's an example of a common misconception: localization equals translation. Thankfully, the situation is improving and blunders such as this are becoming rare; the process of localization is becoming

FOR LANDS
WILD FRONTIER

BY PATRICK DOWLING

"Don't know. The blanket?"



Artwork created by Josh White.

more and more streamlined as people gain experience. Unfortunately, most of that experience was gained through the trusted and proven method of trial and error.

FIGURE 1. *Spot the difference: unacceptable and acceptable.*

FIGURE 2. *Spot the difference: unacceptable and acceptable.*

The Four Steps to World Domination...

If you're planning to make your next project available for worldwide distribution, one of the first things you need to do before beginning production of a foreign version is to decide whether it's worth the effort. You may have the greatest and most realistic shoot-'em-up-kill-a-thon with hours of perfectly synchronized German voice-over, but it probably won't sell in any great numbers in Germany because the violence will make the game available only under the counter to those over 18. Submitting your game to a sales system with no advertising, no reviews, and no revenue is not exactly the best way get your money's worth. Whether or not such systems are necessary, useful, or just a pain isn't the issue; the fact is, it's "different strokes for different folks." In such a case, you have to evaluate your options. Changing the game to make it suitable for a certain market may be fairly simple, such as replacing red blood splatters with green, but can be more invasive, involving removing the gore and death completely. Take a look at the screenshots to see just how little can make the difference (Figures 1 and 2).

On the other hand, a game may be perfectly acceptable, but simply out of place — soccer games may sell like hot cakes in Europe, but soccer isn't the mass-market sport of choice in the U.S. Likewise, business simulations still sell well in Germany, but aren't really an international product worthy of localization. Be aware of these differences, and if in doubt, have someone check out the situation.

Once the decision has been made to go through with the foreign version, the fun really starts. Even if you haven't yet decided upon or brought up the question of localization, it's worthwhile to be prepared. Basically, a localization goes through four different stages: organization, translation, modification/integration, and testing.

During the organizational stage, you need to make an early decision as to when to localize: during development or after completing the game. Most often, the localization takes place once the original product is complete. This isn't due to the process itself, but rather to the insecurity of development and the additional costs that a localization entails. Unless you're certain that your game will sell well abroad, it's hardly worth risking the money in advance. Scheduling a near simultaneous release of all versions really is perfectly feasible — it's just a bit more involved, and can benefit from the close ties between the development and the localization staffs. Whatever the scenario, one person should be responsible for organizing the materials. Any others who will be needed should be informed early on of the decision to localize. Development schedules are tight at the best of times, and you don't want the programmers to find out at the last possible minute that they will be doing some "minor" modifications.

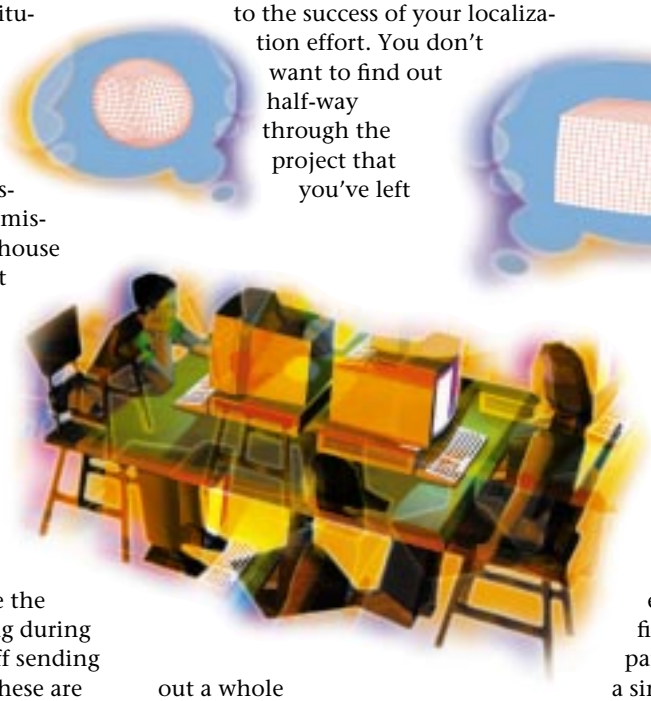
How the materials are organized is critical for the success of the localization. An approach that has worked well is the creation of a localization kit, which contains all of the game's material and some form of documentation. This documentation is often a table containing file names, types, and a brief explanation of each file's purpose, as well as any other notes (Table 1). Text files may need a bit more description, but we'll get into that later. Just use some common sense when preparing the kit. A dozen CD-ROMs with thousands of files documented in hundreds of tables is going to be as useless as a disk full of source code that contains the game's text somewhere in a couple of dozen .c files.

The size of the kit depends on another factor: will you use an outside consultant merely to translate the raw text, or will someone else be replacing the graphics and editing the audio and video as well? This is really a matter of agreement and preference. Some may prefer that an external source perform all localization-related tasks, while oth-

A Canadian living in self-imposed exile in Germany, Patrick Dowling has been described as "the guy who's too bilingual by far." After living in Switzerland, England and Germany, he graduated from university in Canada and worked on the German versions of various games. He is currently a freelance programmer and translator and can be reached at pld@gmx.net.

ers will want to do all the work themselves, using their own staff to perform the adaptation.

Given the sensitive and proprietary nature of games and game technology, it's perfectly understandable that a developer would want to perform localization in-house. The situation, however, does have certain pitfalls. Those making the changes will, in most cases, have little understanding of the material that they're processing, making it very easy for mistakes to slip in. Even for in-house localization, I would suggest having a complete and fully documented kit, using it to keep track of work as it progresses. Thus, if you do end up running out of time, it's fairly simple to off-load the work to someone else.



The kit can also be used to set costs and determine the time needed to translate the materials. If you're localizing during development, you're best off sending the materials in chunks as these are finalized. As soon as the graphics are finished, you can send along a graphics kit. As soon as the text is complete, you can have the text translated. Sending only final versions makes it easier to keep track of what is where and avoids the extra cost of retranslating elements that may have changed. You should provide as much information with the kit as you can: file formats, palette restrictions, compression, and any other information that may be needed to ensure that the materials returned are as complete as possible and useable right away. This completeness is vital to the success of your localization effort. You don't want to find out half-way through the project that you've left out a whole section of the game. Most likely, you'll have some internal documentation of assets anyway, which you can simply modify and re-use for localization.

Translation is, of course, the step most likely to be out-sourced, and is actually somewhat more involved than just translating the text. A translator must also move the text to the new cultural context; the process can take anywhere from a few days to weeks. Only then can you localize the rest of the materials.

Once the materials are complete, you can integrate them into the product. The modification phase that follows translation can, in some cases, be short to nonexistent, or in other cases, very involved. As I mentioned previously, modification may be as simple as replacing red blood with green; in other cases, you may have to remove or replace entire portions of the game. In one instance, a game involved a section with thousands of index cards, each with a graphic showing a few lines of text. Imagine the work necessary to replace all of these, individually. Given the game's estimated sales, the developer finally decided to leave out this part of the puzzle, and in its stead, a simpler puzzle was devised. This story illustrates why it's a good idea to consult the person responsible for localization efforts early in the development cycle; it will help you avoid some major problems.

Testing can be one of the most difficult phases of the project. The integration and testing work may take place

TABLE 1. *Sample from a localization kit.*

FILENAME	FILE TYPE	PURPOSE	TEXT TO BE TRANSLATED	NOTES
Intro.bmp	BMP, 8-bit, no compression	Splash screen at game start	Click to continue...	Palette as PAL.bmp
PAL.bmp	BMP, 8-bit, no compression	Palette for main screens	—	—
Normal.bmp	BMP, 8-bit, no compression	Main Menu buttons, normal state	Audio,Video Options, Quit	Palette as PAL.bmp Each item max. 8 chars
Highlite.bmp	BMP, 8-bit, no compression	Main menu buttons, highlighted state	Audio,Video Options, Quit	
Pressed.bmp	BMP, 8-bit, no compression	Main menu buttons, pressed state	Audio,Video Options, Quit	

many thousands of miles apart — even with telephone, fax, and e-mail, this is quite a distance. What usually happens is a series of to-ing and fro-ing as the local testers find errors, pass these back, and receive new versions in return. There is no real way to shorten this process, and it can only work well if the people doing the integration and modification work are motivated, much the same as during the primary development. It is at this stage where all the previous organizational work pays off. If your game's assets were all labeled and sorted before they went out, and they come back from the localization team in the same state, then they'll just slot right into place. On the other hand, if something does go wrong, is misplaced, or goes missing, then you're really going to have trouble trying to put together the correct bits and pieces.

Let's look at what's involved in localizing some of the different types of material that make up the average game. In spite of their seemingly common-sense nature, someone, somewhere, has ignored one or more of these points, and has thus caused confusion, delays, costs, and many late nights... or at the very least, a localized game with dialogues like the one at the beginning of this article. But first, here are a couple of general tips:

- The English text will, in general, be the most concise; the translated version may be up to twice as long. Normally, the translated version shouldn't be more that 1/4 to 1/3 longer than the original, but it's still something to consider. None of your game elements can escape this rule; graphics will need to leave more space for text, you'll need more space on the CD-ROM, and certainly the programmer will need to account for varying text lengths. While accounting for completely variable text length is a real pain, you shouldn't assume that English is a good measure — it usually isn't. Note any restrictions you make (such as, menu items: 16 characters; descriptions: 256 characters) and make certain the translator knows about them (by documenting them in the kit).
- Don't forget the simple things: make room for the localization people in the credits. Many people have only done this reluctantly, not wanting to see any other names on their product. A good localization often depends on people putting in a lot of effort. A properly localized game is as much their product as it is the developers'.

A Programmer's Guide to Foreign Languages

Depending upon the nature of the game, the programmer may not need to code with future localization efforts explicitly in mind — as long as that programmer is organized and keeps the data separate from the code. Storing text within the code is ill-advised, as it makes the text very difficult to collect. Your localization people could very easily miss whole portions. With all the strings in a string table — even the system provided by Windows — replacing all of your game's text is a simple matter. The task doesn't even require a programmer.

From a programmer's point of view, graphics and sound files can be in any language — besides, perhaps, the varying length of the text. The simplest way to cope with variations in the length of a line of spoken text is to have the speak animation run until the text file has ended. Alternatively, you can have the program read the file's length beforehand. Both approaches are easy to implement. One nameless and unreleased adventure game used the following system for spoken text: Once the script was complete, the script writer timed himself speaking the text (I swear it's true) and passed this information on to the artist and programmer, who then constructed an animation that would run for that length of time — for all 250 pages of script. Then the script changed. What was that bit about "common sense?"

That said, there are three instances in which the programmer needs to be careful when handling text: display, input, and manipulation.

TABLE 2. *International number formats.*

	US	OTHER
DATES	MM/DD/YY	DD/MM/YY
NUMBERS	1,000.25	1.000,25
TIME	1:00PM	13:00

DISPLAY. Some languages (many languages, actually) use more characters than English, not to mention languages that use completely different character sets. The German characters ä, ö, ü and ß can be replaced with ae, oe, ue and ss, but this really looks unprofessional. And you really can't replace the accents in French. In the age of Windows, these character discrepancies are less of a problem, but nonetheless something to keep in mind if you're using your own font and text-display routines. You should try to adhere to some standard for these characters, so the text files won't need conversion or extra work. It is worth noting at this point that number formats vary as well (Table 2).

INPUT. A similar problem arises when you need input from the user. One oversight has probably generated more frustrated users and calls to hotlines than anything else: keyboard layouts. I can think of at least one installation program in which you couldn't change the installation drive — not because the program didn't let you, but simply because you can't enter the [\] character on a German keyboard. Similarly, Yes and No don't always start with a Y or an N; the French prefer *Oui* and *Non*, the Germans *Ja* and *Nein*. At the very least, try to ensure that these simple functions work — for more complex key layouts, making certain that the localized manual lists the correct keys for that locale should be sufficient.

MANIPULATION. If you're trying to localize a game that involves some form of text manipulation that goes beyond simple retrieval, then you may have bitten off more than you can chew. Trying to build sentences from individual words is relatively simple with the familiar English grammar. The results of trying to transfer this familiar system directly to a foreign language range all the way from hilarious to bizarre to, well, completely embarrassing. Anyone who has ever tried to learn a foreign language will remember the difficulties: word order is often completely different, and even if the order is the same, the meaning may be completely different. One might assume that half-three means three-thirty, but *halb drei* in German is two-thirty. Not to mention those blasted articles and their declensions. Take

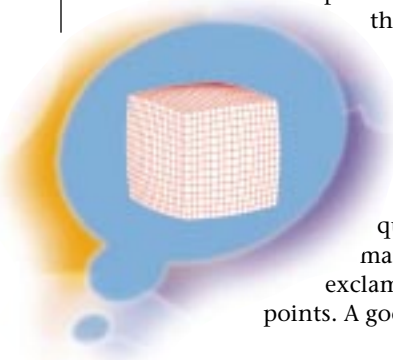
TABLE 3. *A complete audio script.*

Character Name	Flags	Filename	Text	Translation
Mike	R	Mike1	I say, chaps! Marcus really blew his gasket when he heard the news!	Meine Güte, Leute! Marcus ist total ausgeflippt, als er's hörte.
	TD		'Chaps' means 'men.' 'blew his gasket' means he became very angry.	
	AD		Mike is very surprised	

Legend: R: To be recorded TD: Translators direction AD: Actors direction

German, for example (not for extra credit): usage of the three articles *der*, *die*, and *das* depend on the gender of the object. Each article and noun has four cases (plus plural), and usage of the different cases depends on where the word is used in the sentence. Is it the direct object? The indirect object? The subject? What pronoun is being used with it?

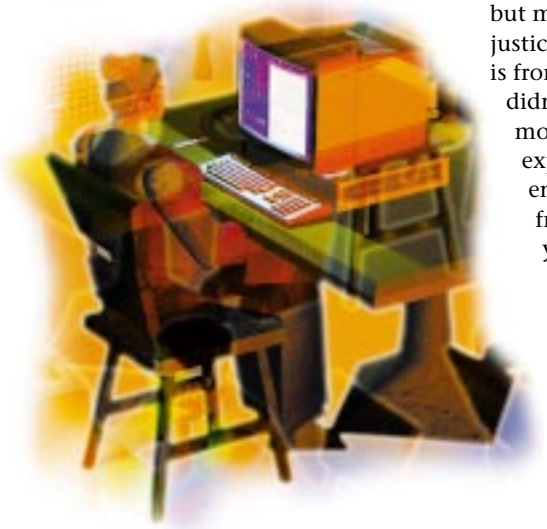
Punctuation also varies from language to language. Whereas in English the punctuation is connected to the previous word, the French expect a space before colons, semicolons, question marks, and exclamation points. A good translation company will worry about these details for you, but if you're doing your own translation, pay close attention to punctuation rules.



Many other languages can be even more complicated than German and French, and even if the language's rules are known and defined, generating the data required to translate all of your game's text will be a major undertaking. The point is if you're considering a system that involves text manipulation, you're going to have to talk to someone who is knowledgeable about each of the planned languages well in advance to try to find a system that is useable.

"T" Time: Text and Translation

The key to translating a computer game effectively (especially if it's a text-dependent game) is attention to detail. Giving the translation to a friend who "lived in Paris for a few years" may improve your friendship, but most likely won't do your game justice — unless, of course, your friend is from another area in France and just didn't like Paris. But I digress... The moral of the story is, bring on the experts. Translating any form of entertainment is quite different from other types of translation — you essentially have to rewrite the text to recreate the atmosphere within the cultural context of the target country. Real-world references in a game can add a lot to the dialog, but once those references have been translated and moved out of their original context, they actually do more harm than good. A good translator will replace (and often improve) those references with some that are more suitable. Similarly, some freedom should be left to change the nature of the characters. One of your game's characters may be mere filler in the original version, but given a local dialect, might transform into a real highlight. The new emphasis on this character might even compensate for a slightly uneventful dialog with the main character. Changing the a character's name may very well improve this "new" character further. The film and cartoon business are no different in this respect; take a peek at some of your favorite cartoon characters, such as Bugs Bunny.



In order to be able to recreate your game's atmosphere, the translator needs as much information as possible. An experienced game translator will require less coaching, and a well-commented script will go a long way; character sheets and sketches are helpful, too. I've seen scripts that go so far as to comment each line with a brief explanation of any slang that's used and a quick mention of the context (Table 3). The point is simple: assume that the translator knows nothing about the game. Even such a simple line as, "That's great" can be interpreted and translated in many ways — most of which are wrong — depending on the context in which it's used. Translators may have a near perfect grasp of the language from which they're translating, but some peculiarities are lost even on a native speaker. This is exactly the problem that leads to obscure and pointless dialogs, such as the introduction to this article.

You should also inform the translator, in advance, of any restrictions — such as text lengths and, more importantly, the in-game purpose of the text — that they may have to watch out for when translating. Translating text to be dubbed to video is somewhat more difficult than translating static text that is simply displayed. Ideally, you want the syllable count to be as close to the original as possible, so you don't end up with characters who speak without opening their mouths — this really makes things look cheap. Experienced voice talent can compensate for minor differences, but when there is only time for a simple "Yes," you won't be able to fit in, "Certainly, my dear," even though that may fit the dialog better. Note that in the example script (Table 3), the translation is inserted next to the original text; it does not replace it. This format serves two purposes: first, when you're looking for a particular file, you can just look for the English and find all the information in one place; and second, you can judge approximately how long the translated versions are relative to the original.

Any text that appears literally in the game should be explicitly marked to ensure that it isn't translated. The best example of this is a game that required the player to input a password. The password itself wasn't particularly difficult for players to find; sufficient clues were dropped in previous conversations. For most players, the difficult part was figuring out that the password had to be input in English — not the most logical conclusion considering that the rest of the game was completely in German. The same rule applies to any text that appears on graphics, such as place names and labels.

Often, you'll end up using several translators. You'll need to ensure that they are all aware of the previously translated work. Many words don't have unique or obvious translations, and the last thing you want is a menu that's translated differently in the game, the manual, and the help file. Usually, there are standard words for everything about the computer or console being used. Sony actually has a set of guidelines with translations for standard items that appear in PlayStation manuals, such as controller, memory card slot, and so on.

Visuals

The secret of successfully localizing your game's visual elements is to retain the raw materials for any graphics that may have to have text replaced. This is one area where layers are quite useful. All you have to do is replace the text layer, reposition it if needed, and you're done. It's that simple. Imagine the pain of having to retouch the background on 70 screens so that you can replace the text. While it can be done (believe me), it really is wasted time. If you don't have the original backgrounds and are planning many localizations, you should only have to restore the backgrounds once.

If your game has a lot of graphics that incorporate text, such as signs or labels, then you'll have to decide whether it's worth the time and effort to replace them. Place names can usually be left as is. If you have a panel that contains some instructions that are required for the rest of the game, however, you may have to replace them, unless the instructions can be built into the text elsewhere.

Just make sure that you make a note of these graphics. They are easy to overlook, and if their existence isn't noted until the later stages of localization, making changes will be all the more difficult.

Even if the original artist isn't going to be doing the changes, it's nonetheless useful to at least have them on hand in case someone less proficient ends up doing the work. That unique combination of 25 different filters may look good in the original, but once someone else has collaged the foreign text onto it, the image may have lost some of its luster. Even though a thorough explanation of the illustration methods may not yield the same results, it will at least ensure that the quality drop isn't too extreme.

Talkin' the Talk

Even though recording may be one of the more time-consuming steps, it's usually best left until late in the localization schedule. The reason for this is simple: you want to do all your recording at once. Studio time isn't cheap, and the last thing you want is to have to go back and re-record a voice-over because the testers have found

Who Does It?

In general, the localization and foreign distribution issues are handled by a publisher. They will either handle the localization themselves, or have contacts in the target country who will take care of it. Many large publishers either have their own foreign offices or have permanent contracts with a local publisher. LucasArts, for example, works closely with Funsoft in Germany and UbiSoft in France, and can thus ensure that the localized versions of their games are usually available though the local publisher at around the same time as the original, and in the same quality. Another common arrangement is for a distributor to bear/share the cost of localization in exchange for exclusive distribution rights in their territory. Bomico in Germany is one company that has managed to negotiate many such agreements and, in fact, have staff at hand to oversee the localization and coordinate the work of studios, freelance translators, and artists. Smaller companies will usually have contacts to external companies such as Polyglot, Polylang, SDL, and the SRC Group. Companies such as these have an increasing amount of experience with all forms of media and can usually handle almost all aspects of localization, often for more than one language. There are also, of course, smaller companies and individuals available locally who deal directly with the publishers and can offer the same quality and service.

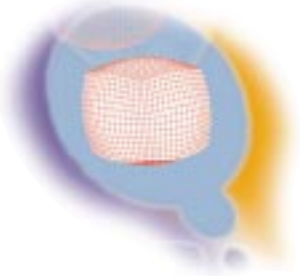
that several important clues are missing, or worse, misleading. Check the script, check it again, and before you check it, make sure you check it.

You really do want to use local talent for voiceovers; using someone who lives nearby and happened to live abroad for a few years might save you the cost of recording overseas, but it will cost you the game's atmosphere. Using a recognized personality as a voice-over actor can boost sales in your home country — why not do the same abroad? You may not be able to have Harrison Ford record the German voiceovers — or the original for that matter — but you can have the next best thing: his voice. How? Almost all famous actors have one actor who does all foreign-language synchronization for all their movies. In the eyes — I mean ears — of the foreign audience, that is the actor's voice. And while they may be more expensive than the normal voice talent, they are still cheaper than trying to hire the original. Imagine having your main character speak in the voice of Brad Pitt, and your heroine sound like Whoopi Goldberg. At the very least, it gives your marketing people something to brag about, the magazines something to write about, and may well give your game some additional publicity.

The basic rules apply here as well: keep the raw materials. Most audio and video software these days lets you put different elements onto different tracks in much the same way that you can layer images in paint programs. You also should have decided in advance who will be doing the editing. While your in-house audio whiz may have the time to edit the entire script, he does have a slight handicap: he (probably) can't understand what it is that he's editing. If you have the script prepared correctly in advance, most studios with some experience in the business will actually record right into the correct files. They'll even add the required sound effects and supply you with a complete set of files that you can use to replace the original files.

With a complete script, you might not even need someone in the studio who knows the game. I would still recommend it, however, to make sure that text is emphasized correctly and doesn't seem out of place, and that the pronunciation is correct.

Finally, don't forget to state the format requirements in advance. High-quality betacam tapes might be great, but if they're in NTSC format, someone in Europe who only has PAL playback won't see much color, if they can actually see anything at all. Format requirements represent only minor hurdles these days, but people experienced in directly dubbing AVI or QuickTime are still few and far between. And trying to organize an overnight videotape conversion isn't something anyone wants to do twice…



Wrapping It Up

Another often-neglected localization item is the supporting materials such as packaging and manuals. Packaging can be especially difficult to oversee from abroad. In some cases, standardized sizes are preferred or even necessary to get any shelf space. In others, the whole packaging design may be unsuitable for the market. Attention to detail is the key here, too. Emblazoning the packaging with reminders of the fact that this game comes from the creators of BIG, BAD, AND UGLY won't help sales if BIG, BAD, AND UGLY was never released in the target country, and review scores from U.S. magazines really don't mean much in Europe and Asia. People might think that BIG, BAD, AND UGLY was never reviewed in their country, or that you're afraid to publish the review scores. Given some of the ghastly localizations that people in other countries have had to put up with — even of high-profile and hyped products — the level of public skepticism is quite understandable, especially because they may be paying a lot more for your game than U.S. customers.

I'd like to offer a final word to the wise: just because a word or name makes no sense or has no connection to reality where you are doesn't mean it has the same status elsewhere. A good example is SECRETS OF RAMA. You might think that a harmless name. However, Rama is brand of margarine in Germany. That would only be a minor slip-up (and wasn't, if I remember correctly), but I'm sure no one has forgotten a large Japanese company's ill-fated Internet campaign featuring Woody Woodpecker as "Woody — The Internet Pecker." That's the kind of publicity we can all do without, and is an excellent illustration of why it's important to have local people involved in any localization project. If localizing a game seems to involve a lot of effort and details to keep in mind, well, it can at first. Once you have a few localized versions under your belt, the process becomes more familiar and can be smoothly incorporated into the development schedule. The costs may also seem high initially. Try thinking of it this way: for a small amount relative to the development costs, you are in effect producing a new product, for a new market, where it can then sell as well as or even better than the original. You've thereby halved the development costs for each product. And if that doesn't convince you, well, then I wonder: What's up? The ceiling, perhaps? ■

Acknowledgements

Thanks to Matt Saettler at Monolith and Volker Reick at Eidos for their help in obtaining the BLOOD and MYTH screenshots.

Psychological Research Methods for Game Design

by Brian J. Geiger

Far from uncovering your game's unresolved issues with its mother, psychological testing is a way of determining how you can make your game just what its players want it to be. Designing a game is not just an art, but a science as well. Psychological testing is a very good tool for the science aspect of game design.

What Is Psychological Testing?

The scientific method is a lot like good debugging. In order to find out what's going on, you hold almost everything static, changing only one thing at a time. Then you measure the output to determine how the item that you changed affects everything else in the system. Once you know how all the parts work separately, then you can examine how they work together.

Psychological testing, though arguably not as rigorous nor as accurate as experiments in the hard sciences (such as chemistry and physics, for example), tries to apply scientific principles and testing procedures to human thinking. Perhaps psychology isn't as well respected as certain other scientific fields because studying the human mind is somewhat ambiguous. A great amount of variation exists from one person to the next, so even the most widely proven laws of psychology will never apply to all people.

WHY PERFORM IT? Although there are a great many difficulties in performing psychological studies, the rewards are also great. One of the major problems with designing games is that it's so subjective. Even if a first-rate design team really loves the concept that it's working with, the possibility still exists that consumers won't like the product.

However, by performing fairly rigorous and scientific tests on the basics of the game, you can reduce the likelihood that everyone will hate what you're doing. That, or you can find out it's going to be a flop and dump it. Furthermore, psychological testing can help mitigate disputes among the designers, between the developer and the publisher, or even between the producer and the design team. The design itself can be the subject of heated debates. Usually, whoever has more control over the project will win a design argument — not necessarily the best outcome. If you can run a test, provide the results, and show that everyone hates idea A but loves idea B, you'll have a better chance of getting idea B into the game. And if you were arguing for idea A, you'll know it's probably better just to forget it.

HOW IT DIFFERS FROM FOCUS GROUP TESTING. My first exposure to testing in the game market was a focus group test. The people running the focus test for one of our games took the game (a puzzle game) and showed it to their target audience (puzzle gamers). The puzzle gamers loved the puzzle game. Go figure. What also helped our case was the fact that there were hardly any other puzzle games at the time for the host platform. So what we learned from the test was that the puzzle gamers who owned a system that has no other puzzle games to

Brian J. Geiger, in between his counseling sessions for socially maladjusted computers, works as a Producer for Gorilla Systems Corp. Aside from FLIPOUT!, his most recent works include TALK WITH ME BARBIE and ADVENTURES WITH BARBIE OCEAN DISCOVERY. He can be contacted at bgeiger@pobox.com for questions and comments.

speak of would buy our game. This focus group was not a significantly useful test, especially considering that the system was the Atari Jaguar and that there were few owners of the system, much less owners who were puzzle game addicts.

In a more scientific setting, a better method would have been to take a more representative sample, a favorite term of research psychologists. The object is to make certain the audience that you're testing matches closely with the audience that might buy your game. A better representative sample in our case would have been: a) gamers in general, or b) Jaguar owners.

Researching Games with Psychology

So, let's assume that you're hooked, and you want to run a test to be sure that your concept is not only as good as you think it is, but that it will work for someone other than, well, you. How do you compose the test, gather the data, run the trials, and analyze the results? Good question. (If you weren't thinking of that question, please humor me. It'll make your enjoyment of the rest of the article much more satisfying, and it causes the previous few sentences to make much more sense.)

COMPOSE THE TEST. First, you'll have to answer several questions; and not particularly easy questions at that. Ask yourself at the outset, "What, exactly, am I testing?" You must be careful when answering this question. If you're asking yourself, "What does everyone want," your scope is way too large. Although you could compose an experiment to answer the question, your budget wouldn't allow for it, and by the time you analyzed the data, it would no longer be relevant.

So, try to test for something very specific. For example, "Given similar experience levels, do players in an immersion game who use a mouse have an advantage over players who use the keyboard?" Another fine question is, "What shapes, when placed on a flipping tile, will immediately distinguish that tile from other, similarly colored tiles, on a 320×480-pixel^2 resolution television monitor?"

While I would like someone to test the first question (and send me the results), the second actually had relevance for us when we were developing FLIPOUT! for the Atari Jaguar. I'll use the FLIPOUT! example throughout this article.

Now, for all you aspiring research psychologists, if you want to get your results published, you'll have to go through the tedious process of accurately defining your terms. For example, what do you mean by "mouse," "play better," "immersion game," and "television?" If you really want to run your experiment that strictly, you're going to want to know much more — pay attention to the "Where to find more information" section later in the article. For the rest of you, merely try to be specific with your terms. Any ambiguity is going to either lessen the validity of your tests or increase the cost of the experiment.

After you've defined the scope of the experiment, you'll have to figure out how to test it. This is where the ambiguity becomes expensive. If, by television, you mean anything from a two-inch portable television to an eight-foot front projection screen, then you're going to have to run a version of the experiment you design on each of those types of sets. For our experiment, we chose the 13-inch Sonys that we had lying around for development. Most of the other factors, we decided, wouldn't affect the results enough to be noticeable.

In the accompanying figures, you can see how the game grid is laid out. In the simple form, you have a three by three grid of tiles that can be flipped in the air. The goal on the easier levels is to match the tile visually with the appropriate base. Our challenge in developing the game was to make the tiles easily distinguishable from each other.

Because of the nature of our data compression and the lack of space on the game's cartridge, the tiles had to be the same hue at different levels of brightness. However, there weren't enough distinct levels of brightness to allow players to distinguish between at least nine different tiles on the screen at the same time. So we had to come up with a different way of differentiating the tiles.

We had to consider several additional factors as well. First, we didn't want to make the tiles look too childish. Second, we were constrained by our compression scheme, with any pattern adding to the memory requirements. Third, we could colorize our tiles, so that we only had to have one tile animation in memory as long as the only thing that changed was the base color.

To satisfy all these demands, we considered the question, "What is the smallest number of tiles, differing by more than color, that a player can easily distinguish among while playing the game?"

Again, if this were a full-fledged scientific experiment, we would define terms such as "easily distinguish" with something similar to, "able to pick the item from a group of nine within .25 seconds." However, we didn't intend to publish our results in a scholarly journal, so we didn't bother.

GATHER THE DATA. First, we composed our test. We made three versions of FLIPOUT! that only differed in the way the tiles were displayed. We had to take great care in this step, because accidentally changing a different variable would throw off the results. For example, if one experimental group has a greater RAM footprint than the others, and the memory swapping routine causes the game to run 10 percent slower, then the players may find it easier or harder to play for a reason that has nothing to do with the tiles' appearance.

The first version of the game was the easiest on RAM. We made all the tiles the same pattern, but the color variations were more subtle in the hope that the players could tell the differences (Figure 1). The second version was RAM and ROM heavy. Each of the tiles had a different pattern rather than varying by only color (Figure 2). The third version was a compromise. The tiles varied by color, but we halved the number of colors needed by using one shape (we chose a circle), which we put on tiles as necessary (Figure 3).

Our choice of test subjects was relatively unscientific. This is a good way to save time and money at the expense of scientific accuracy. We used coworkers, friends, and neighbors as our test subjects, having them play through each of the games and telling us which they preferred. The purist in me would have changed this aspect of our experiment, but the pragmatist in me said,

FIGURE 1. *A version of FLIPOUT! in which the tiles are merely different colors.*

"No, fool, we have no time for such nonsense." So, instead, I'll tell you what I would have done differently.

Most importantly, I would have preferred a better way of determining which method of tile distinction was best. Probably the easiest way of measuring each version's user-friendliness (next to asking) would have been to examine players' scores or the highest level attained in the game. Calculating the average time to finish a level would have been a still more accurate measure.

I also would have preferred to base our test on a more representative sample of the population. Family and friends are great, but they don't make for the least biased test group. Unfortunately, finding test participants is difficult and usually involves a major investment in time and money. So how do you get participants quickly without spending too much money?

THE LOCAL UNIVERSITY IS A GOOD TESTING PLACE. If your local university has a psychology department, chances are good that the students already participate in a few psychological tests each semester. Usually, students can get extra credit in their psychology classes for helping out other students or teachers who are performing experiments.

Here is a potentially large group of people who are ready to help you with your test. If your product is at all interesting, the students will probably enjoy helping you more than they would matching colors to numbers or whatever the other tests might be. But how do you approach a school about performing experiments on their property with their students?

There are a couple ways to try to get the university's help. The first is to go to the head of the psychology department, explain what you're doing, ask for help, and possibly offer some money. It's likely that other businesses in the area are doing similar experiments for whatever purpose, so the university probably wouldn't consider your request to be out of line.

A better way would be to advertise for students who want help with their projects. Many students have a difficult time thinking up topics for their research projects. By offering students a ready-made topic, you're killing a veritable aviary of birds with a single pebble.

First, if any compensation is necessary, approaching students directly would be less expensive than working through the department of a university. Students are notoriously low on money, and universities are notoriously expensive.

FIGURE 2. *The RAM-heavy, different-pattern version of FLIPOUT!*

Second, you skip past many of the trickier issues of dealing with the university directly. All you would have to do is verify with the professor or, more likely, a graduate assistant, that it's all right for your company to propose the project's basic idea. Chances are good that the novelty of the experiment will be enough to get the idea accepted.

Third, a lot of the niggling questions that you would normally skip over due to lack of time and resources can be performed by the student. Design of the experiment, verification of the soundness of the theory, gathering of data, and analysis of the data would all be done for you by the student (and, of course, the professor or graduate assistant).

Naturally, however, there is a down side. You'll be entrusting your project to a student whose work habits are unknown. The student might procrastinate all semester before doing the paper, might do a poor job, or might completely miss the purpose of the experiment. Not to worry, however. With sufficient backup plans and proper delayed-payment incentives, most of your problems will be negligible.

ANALYZE THE RESULTS. Analyzing your test results is probably the trickiest part of the experiment, at least for those who don't remember their statistics courses from college. On the other hand, heavy-duty statistics might not even be necessary for your purposes. For example, in FLIPOUT!, our results were clear enough without our having to break out the HP-48. Most of our test subjects preferred the all-shapes version (Figure 2), but only a few more than the compromise version (Figure 3). The all-color version (Figure 1) brought up the rear. Since the difference between the compromise and the all-shapes RAM-eater was so small, we went with the compromise.

For those who can't get someone else to do their psychological testing and really do need the solid statistics, I recommend getting a book on the subject. There's just too much important information to go over before anything really useful is conveyed.

Let me wrap up with some quick advice. First, try to make sure your experimental group contains at least thirty subjects. Thirty is a magical number in behavioral statistics. Second, don't think that just because something seems right, that it is. If you're going through the trouble of performing the test, and the results are at all in doubt, analyze the results properly. Sometimes, statistical quirks will make what a relatively large difference appear minor, and vice-versa. Obtaining accurate results is especially important when the results are going to affect something expensive.

FIGURE 3. *A compromise version of FLIPOUT!, with tiles distinguished by different colors and a few patterns.*

Ethics

Ah, ethics. Ethics are particularly important if: a) you plan to have your work published; b) you're involving people you don't know; c) you're involving people you do know; and d) you are performing an experiment. Psychological tests have the potential to disturb people greatly. It's doubtful that anything you would test in developing a videogame is going to seriously damage someone's psyche, but try to be careful.

If you have any doubt that what you're doing is completely safe, don't do the experiment. Check with a psychologist to determine whether something might be wrong with your test. Allow your participants to stop the experiment at any time. Immediately. With videogames, warn the participants of possible nausea and the potential of inducing an epileptic seizure. Try to shy away from objectionable material in the test, unless that's the focus of the experiment. If the possibility that people might be offended exists, warn them before they start the experiment.

Where to Find More Information

As I've mentioned, the information I'm leaving out is enough to cover at least two undergraduate courses, and many more postgraduate courses. Check your local bookstore or, better yet, university bookstore for their Research Methods or Experimental Psychology books.

One of the books that I used in college was B. Michael Thorne's *Statistics for the Behavioral Sciences* (Mayfield Publishing Co., 1989). It's a decent enough book, and for those who are rusty on their math skills, worry not: the first chapter and section, respectively, are titled "Calming Your Fears" and "It's not as bad as you think." I am in no way making this up.

You can get the general information you want from *Research Methods, A Process of Inquiry* by Anthony M. Graziano and Michael L. Raulin (Harper Collins College Publishers, 1993).

The American Psychological Association has a set of ethical guidelines at www.apa.org/ethics/homepage.html. They also have a number of online sources of information, including previously published articles of other researchers' experiments. It's worth checking to see if anyone has done an experiment similar enough to yours before going through the trouble yourself.

Keeping It in Perspective

By following these guidelines and perhaps even doing some independent research on the topic, you can greatly improve the playability of your games and ease the pains of the design arguments. Remember, as with any tool, don't overuse it. Test only one thing at a time; keep the tests to the necessary topics, not every question you encounter; and remember: game design isn't just a science, it's an art as well. Trust your instincts, unless the data says otherwise. ■

Building A Little Performance Misery

by Robert Wyatt

Techniques used by today's microprocessors to achieve unprecedented performance have also resulted in unprecedented complexity in the way that assembly routines are optimized. Minimizing the number of instruction execution cycles no longer guarantees the fastest solution. As a developer, you have to consider additional factors, such as cache and translation lookaside buffer (TLB) states, as well as the state of pipelines and execution units. You also have to account for the architectural aspects of the processor, such as support for speculative instruction execution, reordering, and retirement. The alignment of code and data can also make a huge difference. Since the majority of games developed today are implemented almost entirely in C/C++, developers have to rely on the compiler to generate good instruction sequences. Often however, this isn't good enough, and you're forced to profile the game.

In order to fulfill the technical objectives mandated by our TRESPASSER project, our team had to identify, characterize, and resolve performance bottlenecks in a prioritized fashion. We needed a low-level performance analysis tool that would provide real-time information representative of actual execution without severely impacting the performance of the application. The fundamental problem with profiling software is that it's intrusive and therefore affects the execution environment; the profiler's results may be inaccurate.

Inaccuracies can pop up when the profiler itself affects the machine's performance. In other words, the act of measurement is itself affecting the results. This makes it important to be able to "hide" or "remove" the overhead introduced by the profiler. Profilers use some clever tricks to remove their impact on the profiled system, but they can still introduce potentially large inaccuracies.

Profilers have two options for how they gather required information: they can either be intrusive or nonintrusive. An intrusive profiler — such as the one I'll explain how to build — is compiled and linked with the application, and inserts measurement code within the application's instruction stream. This intrusive method of profiling rapidly samples your game's register set. (If you have a prebuilt application, this is usually the only method of getting profile information.) From this information, you can determine what's currently executing. By using the compiler debug information (if it's present), you can also obtain the source file and line number.

On the other hand, some compilers and tools (such as Visual C++ and Intel's VTune) use a nonintrusive technique. VTune, for example, uses time- and event-based sampling to monitor running software. It periodically interrupts the processor and collects samples of the instruction addresses, matches these addresses with an application or an operating system routine, and then populates a database with this information.

Both profiling methods have advantages and disadvantages. For example, to obtain the same accuracy as an intrusive profiler, a nonintrusive profiler has to sample very quickly, which reduces performance. Likewise, intrusive profilers affect the instruction stream and

Rob Wyatt is an engineer at DreamWorks Interactive, based in Los Angeles. He is currently working on the PC title JURASSIC PARK – TRESPASSER. Rob has been involved with game development on many platforms for the last 10 years, but specializes in the PC and Windows. Feedback can be addressed to gdmag@mfi.com

can introduce very hard-to-find bugs. So, how can accurate profiling be performed if the processor doesn't execute code to acquire execution information? Fortunately, the Pentium, Pentium Pro, and Pentium II all contain internal registers and features to help you profile.

This article, which will dive deep into the inner workings of Intel's processors, describes how to use these features to develop an accurate profiler and performance tool (intrusive by the above definition) that operates in real time within your application. The solution uses some privileged instructions and assumes that you're familiar with the architecture of the Pentium and Pentium Pro family of processors, as well as with Windows. The profiles obtained from the code within this article are more accurate than those obtained from either Visual C++ or Intel's VTune.

How to Accomplish Basic Profiling

You can do simple code profiling using the RDTSC (ReaD Time Stamp Counter) instruction. This instruction reads a 64-bit counter that is incremented every processor cycle. The result is returned in two 32-bit integer registers: the EAX contains the lower 32 bits and EDX register contains the upper 32 bits. (All instructions that return a 64-bit result use this approach to returning 64-bit values; Visual C++ also expects functions that return 64-bit values to use these registers.) Intel guarantees that, architecturally, this counter will not wrap around within 10 years after being reset to 0, but in reality, with current processor speeds of 300MHz, the wrap around period is about 2,000 years.

The inline assembler within Microsoft's Visual C++ (including version 5) won't assemble the RDTSC instruction, so you have to inline the opcode bytes **0F 31** like this:

```
__int64  i64_clock;

_asm _emit 0x0F // RDTSC Opcodes 0F 31
_asm _emit 0x31 // eax:edx contain the counter

_asm mov DWORD PTR [i64_clock],eax
_asm mov DWORD PTR [i64_clock+4],eax
  // variable i64_clock contains the counter
```

To make your code readable, I suggest defining this instruction as a macro, like this:

```
#define RDTSC(var) \
_asm _emit 0x0F \   // RDTSC
_asm _emit 0x31 \
_asm mov DWORD PTR var,eax \// Bottom 32 bits
_asm mov DWORD PTR var+4,edx  // Top 32 bits
```

Now, from within your C/C++ code, you can simply write

```
__int64 i64_clock;
RDTSC(i64_clock);// read clock counter into
                    i64_clock
```

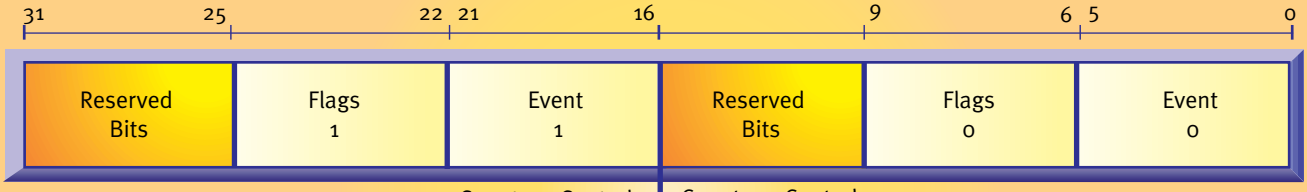
Placing an RDTSC instruction at the start of a section of code and another at the end will give you the number of cycles that the code actually took to execute, including any cache misses, processor stalls, and so on. Listing 1 (which, along with my other extensive code listings, is available on the *Game Developer* web site) shows an example of three different 64K copy operations for your comparison. When you compile and run them in debug and release mode, the output you see is the number of cycles taken for each of the copy operations. You can convert the clock count into a human-readable time by calculating the clock speed of the processor and dividing the clock count by the clock speed. Listing 2 contains a simple function that gives the processor's clock speed.

The RDTSC instruction measures absolute clock cycles, including the overhead caused by the Windows interrupt handlers and context switches. While this means that you can't profile your code by itself, the reality is that your application has to run under Windows, and this overhead is part of the user's experience.

If you profile a large section of code or a substantial loop (as in Listing 1), the handful of cycles for the profile overhead is insignificant. However, be careful when profiling very small sections of code. There's a slight overhead in getting the profiling results into C/C++ variables, and the RDTSC instruction itself takes up to 20 cycles. If you have to profile a very small section of code, you have many approaches from which to choose. First and most simply, you could execute two consecutive RDTSC instructions to calculate the profile overhead, then profile the small section of code and subtract the first profile timing result from the second, giving you the number of cycles taken by the section of code.

But beware. Taking the aforementioned approach is prone to errors. Intel doesn't specify the exact point at which the time stamp is actually read

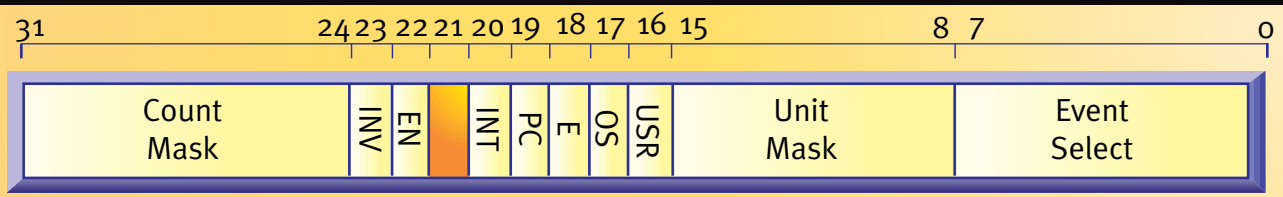
FIGURE 1. *The Pentium family's event select and control flags (MSR 0x11).*



Event 1	P5 Performance Event ID for counter 1
Flags 1 - bit 22:	Count selected event while in kernel modes (OS Bit)
Flags 1 - bit 23:	Count selected event while in user mode (USR Bit)
Flags 1 - bit 24:	Count clocks (ignore selcted event) while mode selected by bits 22 and 23
Flags 1 - bit 25:	PM1 Pin used by external hardware (set to zero)

Event 0	P5 Performance Event ID for counter 0
Flags 0 - bit 6:	Count selected event while in kernel modes (OS Bit)
Flags 0 - bit 7:	Count selected event while in user mode (USR Bit)
Flags 0 - bit 8:	Count clocks (ignore selected event) while mode selected by bits 6 and 7
Flags 0 - bit 9:	PM0 Pin used by external hardware (set to zero)

FIGURE 2. *The Pentium Pro Family's event select and control flags (MSRs 0x186 and 0x187).*



Event Select:	P6 Performance Event ID for the counter.
Unit Mask:	The subgroup of the above event (see App. B).
Counter Mask:	The processor counts the event if the number of events in a single cycle is greater than this mask. To count all events, set this mask to zero. For example, if you set the event to decode instructions and the counter mask was set to 3, the performance counter would increment on the cycles that decoded 3 instructions.
USR bit:	Set to count the selected event while in user mode.
OS bit:	Set to count the selected event while in kernel modes.
E bit:	This enables edge detection of events when set to 1.
PC bit:	This controls the external PMi pins and is not relevant to this article.
INT bit:	Set to generate an interrupt on the processor when the counter overflows. When not used, set to 0.
EN bit:	The enable bit. This bit is only present on Event Select 0 (MSR 0x186) and when set enables both counters.
INV bit:	This inverts the result of the counter mask comparison so that both "less than" and "greater than" comparisons can be made. (Not used by this article.)

during the execution sequence. More specifically, RDTSC isn't a serializing instruction, so it can be executed out of order. Therefore, the cycles that you measure are apt to change. On a Pentium, this won't make a significant difference. However, on a Pentium Pro or Pentium II (which implement out-of-order execution) your results may contain significant inaccuracies. My advice for profiling small sections of code is to try to execute the code multiple times in a loop and average the results, as in Listing 3.

Normally, the RDTSC instruction is unprotected and can be executed from code running at any privilege level. A secure operating system can disable RDTSC from executing at all privilege levels apart from kernel mode (Ring-0) by setting the TSD (Time Stamp Disable) bit (bit 2) in Control Register 4 (CR4). Windows NT and Windows 95 do not by default set this bit, so the RDTSC instruction is executed normally.

Measuring Hardware Performance Events

Undoubtedly, you'll have sections of code that don't run as fast as they should, even after optimization. In these cases, you need to know more than just the cycle counts — you need to dive into the world of performance event counters.

All Intel processors since the first Pentium contain extensive facilities for monitoring performance in the form of two 40-bit performance counters. These counters count "events" that occur within the processor; such events are "data has been read" or "an instruction has been executed." These counters are implemented and accessed as Model Specific Registers (MSRs), and they can monitor two performance events simultaneously (one on each counter).

Be forewarned that the results obtained from these counters require substantial knowledge of the processor to be of any value. This is especially true of the Pentium Pro and Pentium II processors. I highly recommend that you read the technical developer manuals and optimization guides on Intel's developer web site at http://developer.intel.com.

The event select register is an MSR that lets you select which event to monitor. The P5 processor family has a single event select register to control both counters (Figure 1). The P6 family has an event select register for each counter (Figure 2).

As you can see in the event select registers in Figure 2, you can select which processor mode the selected events will be counted within by setting one or both of the USR (user mode) or OS (kernel mode) bits. (If both bits are clear, the counter will stop.) Setting only the USR bit lets you profile just your application code by counting only events that occur while the processor is in USR mode. Interrupts, context switches, and some system calls are performed in kernel mode, and therefore won't be timed.

Other applications also run in user mode, and if a context switch occurs within the section of code being profiled, the application you're switching to will be profiled, producing incorrect results. Since context switches don't occur at regular intervals, they're hard to predict. But generally a thread will execute for a maximum of about 10ms before Windows switches to the next thread. Windows switches between threads based on their priority. The lower-priority threads will only execute after all higher-priority threads have executed. You can use this to your advantage by setting the priority of your application to the maximum level. There will thus be little chance of another thread executing, and your results will be more accurate. The following code will change the priority of your application to real time, which is the maximum allowed:

```
HANDLE h_process = GetCurrentProcess();
SetPriorityClass(h_process,
                 REALTIME_PRIORITY_CLASS);
```

To profile a different event on the P6 family, change the event select register for the required counter. The new event will then begin to be profiled with the next cycle. Unfortunately, changing the events that you're moni-

toring isn't so easy with a P5 processor. The P5 processor only has a single event select register. In order to change events, you first have to read the event select register, modify it, and then write back to it. There is one additional caveat: before selecting the new event, you have to clear all count mode bits to disable the counter, then clear the counter. The final sequence to change the event on a P5 processor follows:

1. Read the event select register.
2. Clear out either the top or bottom 16 bits, depending on which counter is being set. (It's just as easy to clear all 16 bits as it is to clear the mode bits.)
3. Write this new value back to the event select register to disable the counter.
4. Write **0** to the MSR for the required counter
   OR write the new event value and processor mode with the result of Step 2 (in the correct position for the required performance counter).
5. Write this value back to the event select register. Counting the new event will begin on the next cycle. The other counter that is unchanged is unaffected.

The P6 family of processors has a far more flexible performance monitoring system, including an effective 16-bit event select register (8 bits for the event select and 8 bits for the unit mask). This 16-bit register will prevent the P6 processors from running out of event bits, as happened to the P5 when the MMX events were added. (This is the reason that the MMX events on the P5 are attached to specific counters.)

Also useful on the P6 is the event that specifically monitors clock cycles (**P6_CLOCK**). This performance counter is exactly the same as the RDTSC counter if the event select register selects event counting in both kernel and user modes. Neither USR (user mode) nor OS (kernel mode) bits have to be set, so this performance event can be used as a more advanced RDTSC. The P5 doesn't have a performance event for clock cycles. Instead, the event select register on the P5 has two control bits to disable event counting and enable clock counting (Bit 8 for Timer 0 and Bit 24 for Timer 1 – see Figure 1). Using these control bits gives the P5 the same clock counting functionality as the P6. The ability to monitor clocks with the performance counters lets you to switch back to time-based profiling by changing the event select register. Without this capability, you would have to change the RDPMC instructions to RDTSC instructions and recompile.

While monitoring specific events may not give you much information, comparing this information with another event clears up the picture. For instance, look at the **P5_INSTV** event. This event counts the number of instructions executed in the V pipe. By itself, the counter isn't particularly useful. But when this event is counted on one of the performance counters and the **P5_INSTR** event (monitoring the total instructions exe-

FIGURE 3. *Functions within GDPERFORMANCE.C.*

```
BOOL bPSInit(void);
```

This function detects the operating system and opens the correct driver. It also detects the processor in the machine so it can verify you are using the correct events on the correct processor. Call this function within the startup code of your application.
On entry: No parameters.
On exit: Returns TRUE if the driver is ready to use, FALSE otherwise. Non-Intel processors will return FALSE.

```
BOOL bPSClose(void);
```

This function closes the device driver and puts things back to normal. Call it just before your application shuts down.
On entry: No parameters.
On exit: Returns TRUE if the device driver closed successfully, FALSE otherwise.

```
BOOL bPSWriteMSR(DWORD dw_msr, __int64 i64_value);
```

This function writes to a specific MSR. Writing an invalid value to a register or writing to an invalid register may cause a hard crash.
On entry: `dw_msr` The Model Specific Register to write.
`i64_value` The 64-bit value to write to the MSR.
On exit: Returns TRUE if the IOCTL call succeeded, not if the WRMSR succeeded. This function will return FALSE if the driver has not been started.

```
BOOL bPSReadMSR(DWORD dw_msr, __int64* pi64_value);
```

This function reads the current value of the specified MSR. Reading an invalid MSR may cause an exception.
On entry: `dw_msr` The Model Specfic Register to read.
`pi64_value` A pointer to a 64-bit integer that will contain the result of the RDMSR instruction.
On exit: Returns TRUE if the IOCTL call succeeded.

```
BOOL bPSSelectPerformanceEvent(DWORD dw_event, DWORD dw_cou
BOOL b_kernel);
```

This function uses the above MSR functions to select a performance event onto a counter.
On entry: `dw_event` A processor event from either Appendix A or B.
`dw_counter` The performance counter to be used to count the event.
`b_user` Set to TRUE to count the selected event in user mode.
`b_kernel` Set to TRUE to count the selected event in kernel mode.
On exit: Retruns TRUE if the performance event was successfully set. This function will return FALSE if the driver has not been started, the requested event is for the wrong processor, or if the requested event cannot be set on the specified counter.

```
int PSGetProcessorFamily()
```

Returns a value indicating the family of the processor in the machine.
On entry: No parameters.
On exit: 5 – Pentium processor (Pentium/Pentium with MMX).
6 – Pentium Pro processor (Pentium Pro/Pentium II).

TABLE 1. *Performance-Related Model Specific Registers (MSR).*

Pentium/Pentium MMX Performance MSRs

Register Name	MSR Number	Description
Event Select	0x11	Control and Event select for both performance counters.
Counter 0	0x12	Performance Counter 0
Counter 1	0x13	Performance Counter 1
Time Stamp	0x10	Time stamp counter as read by RDTSC

Pentium Pro/Pentium II Performance MSRs

Register Name	MSR Number	Description
Event Select 0	0x186	Control and Event Select for performance counter 0
Event Select 1	0x187	Control and Event Select for performance counter 1
Counter 0	0xC1	Performance Counter 0
Counter 1	0xC2	Performance Counter 1
Time Stamp	0x10	Time stamp counter as read by RDTSC

cuted) is counted on the other performance counter, you get the number of instructions in the U pipe (by calculating `P5_INSTR - P5_INSTV`). You can also calculate the pairing percentage ((U pipe instructions/V pipe instructions)*100), which gives you the percentage of clock cycles that executed two instructions — a far more useful statistic.

Now that we've seen what performance information is available to us, I'll show you how to use it. Table 1 provides a concise list of all the MSRs required for profiling. To access them, you use the RDMSR (ReaD Model Specific Register) and WRMSR (WRite Model Specific Register) instructions. At this point, Windows gets in the way. These instructions are protected and can only be executed from within kernel mode. A Win32 application running in user mode isn't allowed to execute privileged instructions, and any attempt will generate a privileged instruction violation. To get around this problem, you have to write a device driver to execute the instructions on your behalf. This device driver only requires two entry points: one to read an MSR and one to write an MSR. The entry points are in the form of Device IO Control (IOCTL) commands. IOCTL commands allow a Win32 application to make device driver calls using the `DeviceIOControl` API.

There is now just one problem to solve. Using `DeviceIOControl` is time consuming and can take a thousand cycles or more to get to the driver code. While this isn't a problem for setting up which events to monitor, it's a major problem when reading the counter values. Intel provides the answer in the form of the RDPMC (ReaD PerforMance Counter) instruction, which reads one of the performance event counters for you. The event counter to be read (0 or 1) is specified in the ECX register (Table 2). The RDPMC instruction, by default, is disabled from executing outside of privilege level 0 and needs to be enabled. Setting the PCE (Performance Count Enable) bit (bit 8) in Control Register 4 (CR4) enables the RDPMC instruction. Control registers cannot be changed from user mode, so our device driver will have to do it during initialization. To be completely safe, the TSD (Time Stamp Disable) bit is cleared. Note that early Pentium processors, specifically those without MMX, don't implement the RDPMC instruction. On these older processors, the RDPMC instruction will generate an unknown opcode violation; you have to use the device driver to read the performance counters by reading from the MSRs `0x12` and `0x13`. The Pentium with MMX, Pentium Pro, and Pentium II all support the RDPMC instruction.

Using a device driver and the RDPMC instruction, reading the performance events listed in Appendices A and B is as simple as reading the time stamp counter. Fortunately, I've done the device driver work for you.

Building and Installing the Performance Driver

Building Windows device drivers is no easy feat. If you're new to the world of device drivers, download my prebuilt drivers from the *Game Developer* web site. For Windows NT, use the GDPERF.SYS device driver; and for Windows 95, use the GDPERF.VXD device driver. The source for the drivers can be found in Listing 5 (for Windows NT) and Listing 6 (for Windows 95) on the web site. Both device drivers and the support library share a common header file (Listing 4), which contains the IOCTL codes we will use. Be sure to read the documentation that comes with the drivers on the web site. It contains directions for installing and using them correctly.

Using the Performance Driver

To use the driver posted on the *Game Developer* web site, link with the GDPERFORMANCE.C file in Listing 8. This source file contains all the functions that you'll require while monitoring the performance events. The associated header file in Listing 9, GDPERFORMANCE.H, contains all of the performance event IDs, instruction macros, and a set of function macros that map to all the functions within GDPERFORMANCE.C. These macros let you remove all the performance code by simply changing a single line in the header file. To enable performance profiling, use this line in the header file:

```
#define PERFORMANCE_PROFILE (TRUE)
```

To disable performance profiling and remove all profiling code, use this line in the header:

```
#define PERFORMANCE_PROFILE (FALSE)
```

The ability to turn profiling on and off is essential, lest you ship your game with traces of the performance code in the product, which will cause problems because RDPMC is protected. Figure 3 documents the functions in GDPERFORMANCE.C for capturing inline performance information.

The function implementations in GDPERFORMANCE.C are basic, but they're all that's required for a fully functional inline performance monitoring system. The header file implements two macros, `RDPMC0` and `RDPMC1`, which read performance counters `0` and `1`, respectively. These macros are implemented and used in the exact same manner as the `RDTSC` macro from the start of this article. A dummy event for the P5 family of processors, `P5_CLOCKS`, is included within the P5 performance events. This isn't a real event and it has an invalid event field for which the `bPSSelectPerformanceEvent` function checks. If it's detected, the function sets the clock count bits in the event select register as previously described.

In Listing 10 on the web site, you'll find a small and simple test application that demonstrates how to use the performance functions. The code starts by initializing the performance system and detecting the family of processor using the `PSGetProcessorFamily()` function, followed by setting two events that are specific to the local processor. The code then performs a very inefficient matrix multiply as an example of a code segment. On Pentium processors, timer `0` counts floating point operations (`P5_FLOPS`), and counter `1` counts data reads and writes. On Pentium Pro processors, similar events are monitored: timer `0` counts floating point operations retired (`P6_FPOPS`) and timer `1` counts data memory references (`P6_DMREF`). The `P6_FPOPS` must be attached to timer `0`, as it cannot be attached to timer `1` (see Appendix B). The example finishes by displaying the results and closing the performance system. Try building a debug and release version so you can be sure the compiler is optimizing.

Non-Intel Processors

All of the performance information contained within this article applies to Intel processors. Other processors may contain performance registers, but they're guaranteed to be different due to the architectural differences. The AMD K6 and the new AMD K6 3D both implement the RDTSC instruction as indicated by the CPUID instruction. The CPUID instruction also indicates that the K6 processor has model-specific registers, which it does. Unfortunately, none of them contain performance data. As such, the K6 also doesn't implement the RDPMC instruction. Any attempt to execute RDPMC on a K6 will generate an unknown opcode violation. Trying to set the PCE bit in CR4 will cause General Protection Fault (GPF).

Note that the driver included with this article doesn't check the processor and will generate a GPF when used on an AMD machine. The driver on an NT machine is installed and initialized at boot time, so the PCE bit isn't set until the driver is opened by an application. Setting the PCE bit within the initialize code would cause the machine to crash on boot up. The performance library support code in Listing 8 will check for an Intel processor; if the code doesn't find an Intel processor, the driver isn't loaded, and all performance library calls are disabled.

All of the x86 processors that are currently available may contain other valuable information in undocumented Model Specific Registers. If information becomes available, the driver in this article or a modified version of it will enable you to obtain the information.

Experiment With It!

All of the events monitored within the included sample application, no matter what your local processor, are set to count only user mode events. Generally, this is the mode you will use, unless you're profiling Windows calls (I definitely encourage you to profile Windows calls, especially DirectDraw and Direct3D calls). For example, try profiling the same call twice, once within user mode and once in kernel mode. You'll see what the call does within your game and what it's doing within the driver. Another eye opener is using the profiler to determine what configurations of Direct3D execute buffers give the highest driver throughput. The possibilities for profiling Windows are endless. Are functions faster on Windows NT or Windows 95? It's always useful to know how long your favorite Win32 calls take to execute. You may not look at Win32 in the same way ever again!

There you have it: an intrusive performance and profiling tool using the built-in features of the Intel processors. Generally, our profiler shouldn't adversely affect your game's performance, but it will if you read the performance counters or change the selected events too often. While developing TRESPASSER, we didn't permanently profile anything below the triangle level.

To profile TRESPASSER, we had a very useful C++ class that encapsulated all of our profiling needs. This class had a macro that was inserted at the start and end of a block of code. The macro read the performance counters, and the difference was passed back to the profile class, which counted the total time spent in a particular profile block and the number of times the block had been entered. At the end of the frame, the class displayed a hierarchical view of the profile blocks, along with information such as the number of times a block had been entered, total time spent in a block, and the average time per block.

Our profile class could display the hierarchical profile in terms of time or in terms of the new event. At the touch of a button, we could view the number of data references, pipeline stalls, or any other event for any section of the code. At the same time, we could change the rendering options and watch the effects, allowing us to pinpoint bottlenecks that would otherwise have been impossible to locate. We accomplished all of this without quitting the application or using an external tool. I encourage you to try out this profiler for yourself. ■

TABLE 2. *Overview of performance and profiling instructions*

Instruction	Opcode	Description
RDTSC	0F 31	Read Time Stamp Counter *Entry Registers:* *Nothing.* *Exit Registers:* *EAX : Bottom 32 bits of time stamp* *EDX: Top 32 bits of time stamp*
WRMSR	0F 30	Write Model Specific Register *Entry Registers:* *ECX: Model Specific Register Number* *EAX: Bottom 32 bits of MSR* *EDX: Top 32 bits of MSR* *Exit Registers:* *All registers preserved*
RDMSR	0F 32	Read Model Specific Register *Entry Registers:* *ECX: Model Specific Register Number* *Exit Registers:* *EAX: Bottom 32 bits of MSR* *EDX: Top 32 bits of MSR*
RDPMC	0F 33	Read Performance Counter *Entry Registers:* *ECX: Performance counter to read (0 or 1)* *Exit Registers:* *EAX: Bottom 32 bits of performance counter* *EDX: Top 8 bits of performance counter*

(Performance counters are only 40 bits, the top bits of EDX are filled with 0. A fault will occur if ECX is not 0 or 1.)

Appendix A. Pentium family performance events.

Symbolic Name	Event ID	Event Counters	Type	Description
P5_DTCRD	0x00	0 and 1	E	Data read
P5_DWRIT	0x01	0 and 1	E	Data write
P5_DTTLB	0x02	0 and 1	E	Data TLB miss
P5_DTRMS	0x03	0 and 1	E	Data cache read miss
P5_DWRMS	0x04	0 and 1	E	Data cache write miss
P5_WHLCL	0x05	0 and 1	E	Write (Hit) to M or E state line
P5_DCLWB	0x06	0 and 1	E	Data cache lines written back
P5_DCSNP	0x07	0 and 1	E	External snoops
P5_DCSHT	0x08	0 and 1	E	External data cache snoop hits
P5_MAIBP	0x09	0 and 1	E	Memory access in both pipes
P5_BANKS	0x0A	0 and 1	E	Bank conflicts
P5_MISAL	0x0B	0 and 1	E	Miss aligned memory reference or I/O
P5_COCRD	0x0C	0 and 1	E	Code read
P5_COTBL	0x0D	0 and 1	E	Code TLB miss
P5_COCMS	0x0E	0 and 1	E	Code cache misses
P5_ANYSG	0x0F	0 and 1	E	Segment register loaded
P5_BRANC	0x12	0 and 1	E	Branches
P5_BTBHT	0x13	0 and 1	E	BTB hits
P5_TBRAN	0x14	0 and 1	E	Taken branch or BTB hit
P5_PFLSH	0x15	0 and 1	E	Pipeline flushes
P5_INSTR	0x16	0 and 1	E	Instructions executed
P5_INSTV	0x17	0 and 1	E	Instruction executed in V pipe
P5_CLOCL	0x18	0 and 1	D	Bus active
P5_PSDWR	0x19	0 and 1	D	Full write buffers
P5_PSWDR	0x1A	0 and 1	D	Waiting for data memory read
P5_NCLSW	0x1B	0 and 1	D	Write to E or M state cache line
P5_LCKBS	0X1C	0 and 1	E	Locked bus cycle
P5_IORWC	0x1D	0 and 1	E	I/O read or write cycles
P5_NOCMR	0x1E	0 and 1	E	Non-cacheable memory read
P5_PSLDA	0x1F	0 and 1	D	AGI
P5_FLOPS	0x22	0 and 1	E	Floating point operations
P5_DBGR0	0x23	0 and 1	E	Breakpoint match on DR0
P5_DBGR1	0x24	0 and 1	E	Breakpoint match on DR1
P5_DBGR2	0x25	0 and 1	E	Breakpoint match on DR2
P5_DBGR3	0x26	0 and 1	E	Breakpoint match on DR3
P5_HWINT	0x27	0 and 1	E	Hardware interrupts
P5_DTRWR	0x28	0 and 1	E	Data read and write
P5_DTRWM	0x29	0 and 1	E	Data cache read or write miss
P5_BOLAT	0x2A	0	D	Bus ownership latency
P5_BOTFR	0x2A	1	E	Bus ownership transfer
P5_MMXA1	0x2B	0	E	MMX Instruction executed U pipe
P5_MMXA2	0x2B	1	E	MMX Instruction executed V pipe
P5_MMXMS	0x2C	0	E	Cache M state line sharing
P5_MMSLS	0x2C	1	E	Cache line sharing
P5_MMXB1	0x2D	0	E	EMMS instructions executed
P5_MMXB2	0x2D	1	E	Transitions from MMX to FP
P5_MMXBU	0x2E	0	D	Processor bus utilization
P5_NOCMW	0x2E	1	E	Non-cacheable memory write
P5_MMXC1	0x2F	0	E	Saturated MMX instructions executed
P5_MMXC2	0x2F	1	E	Saturation's performed
P5_MMXHS	0x30	0 and 1	D	Cycles not in HALT state
P5_MMXD2	0x31	0	E	MMX Data read
P5_MMXDM	0x31	1	E	MMX Data read miss
P5_MMXFP	0x32	0	D	Floating point stalls
P5_MMXTB	0x32	1	E	Taken branches
P5_MMXD0	0x33	0	E	D1 starvation and FIFO is empty
P5_MMXD1	0x33	1	E	D1 starvation and 1 instruction in FIFO
P5_MMXE1	0x34	0	E	MMX data writes
P5_MMXE2	0x34	1	E	MMX data cache write miss
P5_MMXWB	0x35	0	E	Pipeline flushes, wrong branch prediction
P5_MMXWP	0x35	1	E	Pipeline flushes, wrong branch prediction resolved in WB-stage
P5_MMXF1	0x36	0	E	Misaligned MMX memory reference
P5_MMXF2	0x36	1	D	Pipeline Stalled for MMX data read
P5_MMXRP	0x37	1	E	Returns predicted incorrectly or not predicted at all
P5_MMXRI	0x37	0	E	Returns predicted
P5_MMXG1	0x38	0	D	MMX multiply unit interlock
P5_MMXG2	0x38	1	D	MOVD/MOVQ store stall due to previous operation
P5_MMXRT	0x39	0	E	Returns
P5_MMXRB	0x39	1	E	Return Stack Buffer Overflows
P5_MMXBF	0x3A	0	E	BTB false entries
P5_MMXMP	0x3A	1	E	BTB miss prediction on a not taken branch
P5_PXDWR	0x3B	0	D	Full write buffers while executing MMX instructions
P5_PXZWR	0x3B	1	D	MMX write to E or M state cache line

Source: *The Pentium Processor Developer's Manual*. This manual, along with all Pentium and Pentium Pro reference manuals, can be downloaded in .PDF format from http://developer.intel.com.

Symbolic Name: The name in the header file of the event. These are defined within GDPERFORMANCE.H and are used to select the event to be monitored.

Event ID: The performance event ID numbers that identify the required function in the event select register (see Figure 1). These functions are specific to the P5 family of processors.

Event Counters: These state which counters an event can be monitored by.

Type: This returns an "E" for an event and a "D" for duration/cycles. When counting events, the counters are incremented each time the specified event takes place. When measuring duration, the counters count the number of processor clock cycles that occur while the specified event is true.

Description: A short text description of the event.

Columns in reverse type represent events that are only present on Pentium processors with MMX. These are invalid events on earlier Pentiums.

Appendix B. Pentium Pro and Pentium II family performance events

Symbolic Name	Event ID	Unit Mask	Event Counters	Subsystem	Description
P6_STRBB	0x03	0x00	0 and 1	MO	Store Buffer Block
P6_STBDC	0x04	0x00	0 and 1	MO	Store Buffer Drain Cycles
P6_MISMM	0x05	0x00	0 and 1	MO	Misaligned Data Memory Reference
P6_SEGLD	0x06	0x00	0 and 1	MISC	Segment register loads
P6_FPOPE	0x10	0x00	0	FPU	FP Computational Operations
P6_FPEOA	0x11	0x00	1	FPU	FP Microcode Exceptions
P6_FMULT	0x12	0x00	1	FPU	Multiplies
P6_FPDIV	0x13	0x00	1	FPU	Divides
P6_DBUSY	0x14	0x00	1	FPU	Cycles Divider Busy
P6_L2STR	0x21	0x00	0 and 1	L2$	L2 address strobes
P6_L2BBS	0x22	0x00	0 and 1	L2$	Cycles L2 Bus Busy
P6_L2BBT	0x23	0x00	0 and 1	L2$	Cycles L2 Bus Busy transferring data to CPU
P6_L2ALO	0x24	0x00	0 and 1	L2$	L2 Lines Allocated
P6_L2MAL	0x25	0x00	0 and 1	L2$	L2 M-state Lines Allocated
P6_L2CEV	0x26	0x00	0 and 1	L2$	L2 Lines Evicted
P6_L2MEV	0x27	0x00	0 and 1	L2$	L2 M-state Lines Evicted
P6_L2MCF	0x28	0x01	0 and 1	L2$	L2 Cache Instruction Fetch Misses
P6_L2FET	0x28	0x0F	0 and 1	L2$	L2 Cache Instruction Fetches
P6_L2DRM	0x29	0x01	0 and 1	L2$	L2 Cache Read Misses
P6_L2DMR	0x29	0x0F	0 and 1	L2$	L2 Cache Reads
P6_L2DWM	0x2A	0x01	0 and 1	L2$	L2 Cache Write Misses
P6_L2DMW	0x2A	0x0F	0 and 1	L2$	L2 Cache Writes
P6_L2CMS	0x2E	0x01	0 and 1	L2$	L2 Cache Request Misses
P6_L2DCR	0x2E	0x0F	0 and 1	L2$	L2 Cache Requests
P6_DMREF	0x43	0x00	0 and 1	DCU	Data Memory References
P6_DCALO	0x45	0x0F	0 and 1	DCU	L1 Lines Allocated
P6_DCMAL	0x46	0x00	0 and 1	DCU	L1 M-state Data Cache Lines Allocated
P6_DCMEV	0x47	0x00	0 and 1	DCU	L1 M-state Data Cache Lines Evicted
P6_DCOUT	0x48	0x00	0 and 1	DCU	L1 Misses outstanding
P6_TSMCD	0x52	0x00	0 and 1	MISC	Time Self-Modifiying Code Detected
P6_BRDCD	0x60	0x00	0 and 1	EBL	External Bus Request Outstanding
P6_BRBNR	0x61	0x00	0 and 1	EBL	External Bus Cycles While BNR Asserted
P6_BUSBS	0x62	0x00	0 and 1	EBL	External Bus Cycles - DRDY Asserted (busy)
P6_BLOCK	0x63	0x00	0 and 1	EBL	External Bus Cycles - LOCK signal asserted
P6_BBRCV	0x64	0x00	0 and 1	EBL	External Bus Cycles - Processor receiving data
P6_BURST	0x65	0x00	0 and 1	EBL	External Bus Burst Read Operations
P6_BRINV	0x66	0x00	0 and 1	EBL	External Bus Read for Ownership Transaction
P6_BMLEV	0x67	0x00	0 and 1	EBL	External Bus Writeback M-state Evicted
P6_BBIFT	0x68	0x00	0 and 1	EBL	External Bus Burst Instruction Fetches
P6_BINVL	0x69	0x00	0 and 1	EBL	External Bus Invalidate Transactions
P6_BPRBT	0x6A	0x00	0 and 1	EBL	External Bus Partial Read Transactions
P6_BPTMO	0x6B	0x00	0 and 1	EBL	External Bus Partial Memory Transactions
P6_BUSIO	0x6C	0x00	0 and 1	EBL	External Bus I/O Bus Transactions
P6_BUSDF	0x6D	0x00	0 and 1	EBL	External Bus Deferred Transactions
P6_BUSTB	0x6E	0x00	0 and 1	EBL	External Bus Burst Transactions
P6_BMALL	0x6F	0x00	0 and 1	EBL	External Bus Memory Transactions
P6_BSALL	0x70	0x00	0 and 1	EBL	External Bus Transactions
P6_CLOCK	0x79	0x00	0 and 1	MISC	Clockticks
P6_BRHIT	0x7A	0x00	0 and 1	EBL	External Bus Cycles While HIT Asserted
P6_BRHTM	0x7B	0x00	0 and 1	EBL	External Bus Cycles While HITM Asserted
P6_BRSST	0x7E	0x00	0 and 1	EBL	External Bus Cycles While Snoop Stalled
P6_CMREF	0x80	0x00	0 and 1	IF	Total Instruction Fetches
P6_TOIFM	0x81	0x00	0 and 1	IF	Total Instruction Fetch Misses
P6_INTLB	0x85	0x00	0 and 1	IF	Instructions TLB Misses
P6_CSFET	0x86	0x00	0 and 1	IF	Cycles Instruction Fetch Stalled
P6_FTSTL	0x87	0x00	0 and 1	IF	Cycles Instruction Fetch stalled due to pipeline
P6_RSTAL	0xA2	0x00	0 and 1	STL	Resource Related Stalls
P6_MMXIE	0xB0	0x00	0 and 1	MMX	MMX Instructions Executed
P6_SAISE	0xB1	0x00	0 and 1	MMX	Saturated Arithmetic Instructions Executed
P6_PORT0	0xB2	0x01	0 and 1	MMX	MMX micro-ops executed on Port 0
P6_PORT1	0xB2	0x02	0 and 1	MMX	MMX micro-ops executed on Port 1
P6_PORT2	0xB2	0x04	0 and 1	MMX	MMX micro-ops executed on Port 2
P6_PORT3	0xB2	0x08	0 and 1	MMX	MMX micro-ops executed on Port 3
P6_MMXPA	0xB3	0x00	0 and 1	MMX	MMX Packed Arithmetic
P6_MMXPM	0xB3	0x01	0 and 1	MMX	MMX Packed Multiply
P6_MMXPS	0xB3	0x02	0 and 1	MMX	MMX Packed Shift
P6_MMXPO	0xB3	0x04	0 and 1	MMX	MMX Packed Operations
P6_MMXUO	0xB3	0x08	0 and 1	MMX	MMX Unpacked Operations
P6_MMXPL	0xB3	0x10	0 and 1	MMX	MMX Packed Logical
P6_INSTR	0xC0	0x00	0 and 1	ID	Instructions Retired
P6_FPOPS	0xC1	0x00	0	FPU	FP operations retired

Symbolic Name	Event ID	Unit Mask	Event Counters	Subsystem	Description
P6_UOPSR	0xC2	0x00	0 and 1	ID	Micro-Ops Retired
P6_BRRET	0xC4	0x00	0 and 1	BRN	Branch Instructions Retired
P6_BRMSR	0xC5	0x00	0 and 1	BRN	Branch Mispredictions Retired
P6_MASKD	0xC6	0x00	0 and 1	INT	Clocks while interrupts masked
P6_MSKPN	0xC7	0x00	0 and 1	INT	Clocks while interrupt is pending
P6_HWINT	0xC8	0x00	0 and 1	INT	Hardware Interrupts Received
P6_BTAKR	0xC9	0x00	0 and 1	BRN	Taken Branch Retired
P6_BTAKM	0xCA	0x00	0 and 1	BRN	Taken Branch Mispredictions
P6_FPMMX	0xCC	0x01	0 and 1	MMX	Transitions from Floating Point to MMX
P6_MMXFP	0xCC	0x00	0 and 1	MMX	Transitions from MMX to Floating Point
P6_SIMDA	0xCD	0x00	0 and 1	MMX	EMMS Instructions Executed
P6_MMXIR	0xCE	0x00	0 and 1	MMX	MMX Instructions Retired
P6_SAISR	0xCF	0x00	0 and 1	MMX	Saturated Arithmetic Instructions Retired
P6_INSTD	0xD0	0x00	0 and 1	ID	Instructions Decoded
P6_NPRTL	0xD2	0x00	0 and 1	STL	Renaming Stalls
P6_SRSES	0xD4	0x01	0 and 1	STL	Segment Rename Stalls – ES
P6_SRSDS	0xD4	0x02	0 and 1	STL	Segment Rename Stalls – DS
P6_SRSFS	0xD4	0x04	0 and 1	STL	Segment Rename Stalls – FS
P6_SRSGS	0xD4	0x08	0 and 1	STL	Segment Rename Stalls – GS
P6_SRSXS	0xD4	0x0F	0 and 1	STL	Segment Rename Stalls - ES DS FS GS
P6_SRNES	0xD5	0x01	0 and 1	MISC	Segment Renames - ES
P6_SRNDS	0xD5	0x02	0 and 1	MISC	Segment Renames - DS
P6_SRNFS	0xD5	0x04	0 and 1	MISC	Segment Renames - FS
P6_SRNGS	0xD5	0x08	0 and 1	MISC	Segment Renames - GS
P6_SRNXS	0xD5	0x0F	0 and 1	MISC	Segment Renames - ES DS FS GS
P6_BRDEC	0xE0	0x00	0 and 1	BRN	Branch Instructions Decoded
P6_BTBMS	0xE2	0x01	0 and 1	BRN	BTB Misses
P6_RETDC	0xE4	0x00	0 and 1	BRN	Bogus Branches
P6_BACLR	0xE6	0x00	0 and 1	BRN	BACLEARS Asserted

Source: *The Pentium Pro Family Developer's Manual* and *The Intel Architecture Optimizations Manual*. These manuals along with all Pentium and Pentium Pro reference manuals can be downloaded in .PDF format from http://developer.intel.com.

This table lists the events that can be monitored on the P6 family processors (Pentium Pro, Pentium II).

Symbolic Name: The name in the header file of the event. These are defined within GDPERFORMANCE.H and are used to select the event to be monitored.

Event ID: The performance event ID numbers that identify the required function in the event select register. See Figure 2.

Event Counters: These state which of the counters an event can be measured with. The P6 family has more register space for performance counters, so unlike the Pentium, most events can be measured by either counter. Some events have identical ID numbers, and in these cases the Unit Mask field determines the function.

Unit Mask: This indicates the unit mask for further qualifying the selected event. The unit mask is concatenated along with the event ID to make a 16-bit value. This value is used in the Event Select Registers, of which there are two.

When monitoring cache events, the unit mask specifies a MESI state for the event. (MESI is an acronym for "Modified, Exclusive, Shared, Invalid." The MESI states are mutually independent and can all be set simultaneously.) The MESI state in the unit mask is used as a binary mask on the MESI state of the cache line that caused the event. If the logical 'AND' of the unit mask MESI state and the cache line MESI state is nonzero, then the event is counted.

The MESI state in the unit mask has the following layout:

Modified (Bit 3): Set to count cache events for cache lines that have been modified; as in, written to by the processor.
Exclusive (Bit 2): Set to count cache events for cache lines that are exclusive to a single processor.
Shared (Bit 1): Set to count cache events for cache lines that are shared by multiple processors.
Invalid (Bit 0) Set to count cache events for cache lines that are incorrect with respect to main memory.

To count all events irrespective of the state of the cache line, you set all the bits in the MESI unit mask (a value of 0x0F). Thus, the events that this applies to are marked 0x0F in the Unit Mask column.

For convenience, the most common MESI states are specified explicitly within the table. For example, P6_L2FET counts the number of level 2-cache-instruction fetch events with a MESI mask of 0x0F, whereas P6_L2MCF is the same event but uses a MESI mask of 0x01 (Invalid) and counts level 2 instruction cache misses. There is a similar pair of entries in the table for every level 2 event.

The performance support code included with the article (GDPERFORMANCE.C, Listing 8) does not allow you to arbitrarily set the unit mask/MESI state. To set a different unit mask for a given event, you must add a new entry to the header file. The bit layout of the IDs within the GDPERFORMANCE.H header file (Listing 9) is detailed in the header file.

Subsystem: This indicates which subsystem within the processor the specific performance event relates to. The various subsystems are:

DCU: Data Cache Unit
L2$: Level 2 Cache
IF: Instruction Fetch
EBL: External Bus Logic
FPU: Floating Point Unit
MO: Memory Ordering
ID: Instruction Decode
INT: Interrupts
BRN: Branch unit
STL: Stalls
MISC: Miscellaneous
MMX: MMX event
PentII: Pentium II processors only

Description: What the event is.

Events listed in reverse type are only available on Pentium II processors. These are invalid events on the Pentium Pro.

WaveConvert Pro 2.3 and BarbaBatch 2.1

by Mark Steven Miller

How many times has your producer said something like this: "I know we said that you would have 50MB of space for audio on the CD. Well, we had some trouble compressing the models. Could you get everything to fit in 25MB?" How many times have your 2,500 voice-over files just not sounded right when you used some strange compression codec? How many times have you had to make a couple of hundred sound effects a little louder and brighter?

If you work on audio for interactive media, then this is just your life, buster. Most of the time, you just don't have the budget to hire someone else to take care of these audio-processing tasks, which is especially painful when you have to run multiple DSP passes over each file, one at a time. If you're the person who gets stuck with these audio-processing chores, then thank your lucky stars for two Macintosh-based tools, WaveConvert Pro 2.3 and BarbaBatch 2.1 (Figures 1 and 2, respectively).

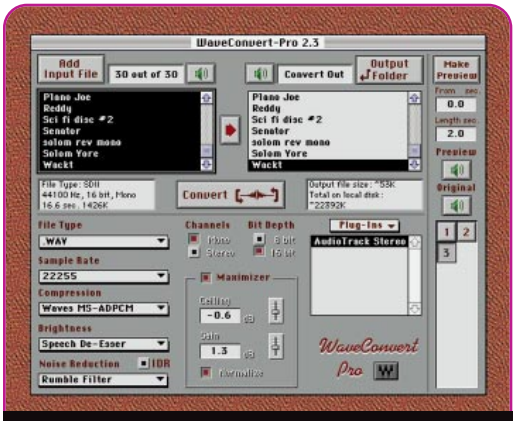


FIGURE 1. *WaveConvert Pro 2.3.*

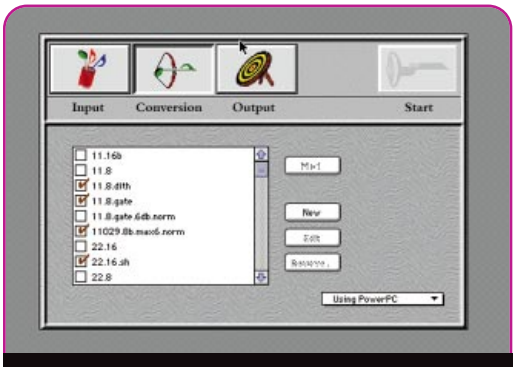


FIGURE 2. *BarbaBatch 2.1.*

For example, certain file types contain restrictions as to the bit and sample rates that may be used. Furthermore, some complex files, though supported, are

Tables 1 and 2 summarize the file types and formats supported by both tools at the input and output stages. There are a number of important things to consider when reading this table. First, not all of the combinations in Tables 1 and 2 are possible.

Mark Steven Miller is the senior producer at Harmonix Music Systems in Cambridge, Mass., where he is currently obsessed with producing animated Claymation musicians for upcoming Axe album releases. In past lives, he has served as the chairman of the Interactive Audio Special Interest Group, the audio and video manager for Crystal Dynamics, and the audio director for Sega of America. In addition to writing occasional articles, Mark is on the advisory board of the Computer Game Developer's Conference and the upcoming Music and Technology Exposition. Mark can be reached at neuroms@slip.net.

handled less than elegantly (QuickTime is one such format). Finally, while a tool may not directly handle some file types, it may be able to prepare files for conversion to that file type (for instance, WaveConvert Pro can prepare files for conversion to Shockwave and RealAudio formats).

Table 1 shows that both programs support most major formats to some degree. Note that BarbaBatch is missing support for the Raw and IMA-ADPCM file formats, but that it outputs MPEG1 Layer 1 and 2 files and RealAudio files directly. Version 2.2 of BarbaBatch, which will be available by the time you read this, will also support QuickTime, 24- and 32-bit Sound Designer, AIFF, WAVE and NeXT files. WaveConvert Pro is missing many of the less common file types, but has excellent support for QuickTime, Shockwave, RealAudio, and IMA and MS ADPCM files.

TABLE 1. *File formats supported by WaveConvert Pro (WCP) and BarbaBatch (BBB).*

File Format	WCP Input	WCP Output	BBB Input	BBB Output
AIFF	X	X	X	X
.WAV	X	X	X	X
QuickTime	X	X[1]		X
Mac Resource .snd	X	X		
Sound Designer	X	X	X	X
Sound Designer II	X	X	X	X
Raw	X	X		
IMA ADPCM	X	X		
MS ADPCM	X	X	X	X
Dialogic Vox			X	X
NeXT/Sun a-law .snd			X	X
Headerless a-law .au			X	X
AIFC			X	X
MPEG1 Layer 1			X	X
MPEG1 Layer 2			X	X
Shockwave		via presets		
RealAudio		via presets		X
Amiga IFF 8SVX			X	X
AVR			X	X
Waves NoLoss		X		

[1] WCP QuickTime support allows for the saving of interleaved (flattened) or not interleaved movies. In addition, WCP offers elegant handling of multiple audio tracks within a QuickTime movie. In a QuickTime movie with multiple audio tracks, each can be processed with its own setting or not at all, and the resultant movie will have all of the original tracks (processed if instructed to do so) in place when the process is complete.

File Input

In both programs, files can be selected either in a standard Macintosh file selector or by dragging and dropping folders or files onto the desktop icon or main application window. When adding files to a batch list, BarbaBatch stands out in that it lets you drag and drop folders containing nested directory structures. This is a huge advantage for projects in which you have a large number of source files that require some organizing directory structure (such as a league/team/player name directory in a sports game). This nested directory structure will be recreated during the output phase, with separate subdirectories created for each different conversion.

WaveConvert Pro simply ignores files that exist in nested directories during a drag-and-drop input, so you have to visit each subfolder separately to select all of the files that you need. For a large job, this can be quite time consuming. Once you've input all your files, however, WaveConvert Pro does allow you to save a text file called a Joblist, which, among other things, can be loaded again later if you need to reprocess a large batch of files.

Both programs have good features that allow you to audition individual files. In WaveConvert Pro, you can double-click on a group of files and they will play in order. BarbaBatch lets you batch audition files from a folder and even lets you insert an alert sound between files.

Loading files was easy in both programs. The only trouble I had was when I tried to import Raw audio files into WaveConvert Pro — the product had trouble recognizing Raw files that it had created itself, which was odd.

Processing Audio Files

Both tools support basic file conversion processes, such as mono-to-stereo and vice versa, sample-rate and bit-rate conversions, and file format conversions (see Table 1 for more detail.) BarbaBatch also can export split stereo files, which is very useful for people who work with a multitrack digital audio workstation, such as Digidesign's ProTools or Macromedia's Deck.

Beyond these basic functions, the products begin to diverge. BarbaBatch offers a minimal, but well targeted suite of processing functions. WaveConvert Pro, on the other hand, offers a much wider range of processing options and supports external DSP plug-ins. So it's almost more appropri-

TABLE 2. *Sample and Bit Rate Chart: (Sample and Bit rates may be restricted by file format)*

Format	WCP Input	WCP Output	BBB Input	BBB Output
4 Bit		X	X	X
8 Bit	X	X	X	X
16 Bit	X	X	X	X
24 Bit	X		X	X
Sample Rates	3Khz-48Khz	3Khz-48Khz	1Khz-100Khz	1Khz-100Khz

ate to think of WaveConvert Pro as a platform rather than a single application. These plug-ins, also from Waves, represent some of the highest quality audio signal processing available on any platform, anywhere. While considerably less expensive than comparable hardware devices, these external plug-ins are not cheap (each costs up to twice as much as WaveConvert Pro itself). Because of this price difference, suffice it to say that WaveConvert Pro's power comes from its extensibility.

Sample Rate Conversion

While BarbaBatch supports a wider range of sample rates, both applications cover most of the important ones. To complement its sample rate conversion capabilities, WaveConvert Pro also offers eleven brightness settings: four for Real Audio, four for Shockwave audio, a "hard" setting for more intelligible speech, a "soft" setting for music, and the new speech De-esser. (De-essers do just that — they remove extra "esses" from speech). The "hard" and "soft" brightness settings work very well, adding back high end and presence into files converted to lower sample rates, while not making the files too harsh or brittle.

The RealAudio and Shockwave audio settings produce much better results than if you do a straight conversion using the tools supplied with either of those codecs, and also surpass the quality of BarbaBatch's RealAudio output files. While WaveConvert Pro can't save out these file formats, both Real Networks and Macromedia supply their own simple batch-processing tools for this purpose.

Compression

Both programs can generate files in the most common audio compression format, 4-bit ADPCM (Adaptive Pulse Code Modulation). WaveConvert Pro outputs both Microsoft and International Multimedia Association (IMA) ADPCM. Though BarbaBatch inputs and outputs only the Microsoft flavor of ADPCM, it also supports AIFC files (the Macromedia Director format).

In addition to standard ADPCM encoders, WaveConvert Pro offers its own ADPCM compressors for both Microsoft and IMA compression. Both compressors offer definite improvements over the standard versions when played back within WaveConvert Pro, but results can vary when the files are played back using other implementations of ADPCM decompression. The Waves NoLoss encoder utilizes Waves' own lossless compression format specifically designed for audio files. NoLoss is useful for archiving or transferring files, but I didn't review it specifically.

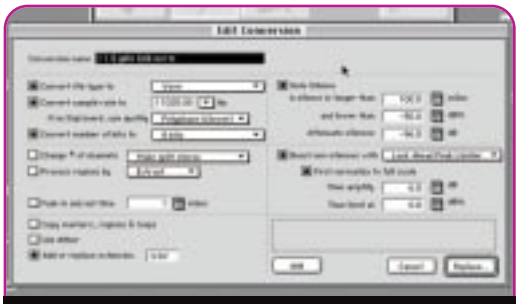


FIGURE 3. *The "Define New Conversion" dialog box in Barbabatch.*

Dynamics and Bit-Rate Conversion

Even more so than low-sample–rate files, low-bit–rate files have long been the bane of multimedia and game audio developers. Simply put, most 16-bit to 8-bit file conversions sound awful. In addition to the loss of 48 decibels of signal to noise ratio, the reduction in amplitude resolution almost always introduces quantization noise. This introduced noise is most noticeable in quiet passages of music or in the spaces between words of speech files. Both programs have many useful tools for dealing with these problems.

Both tools offer great flexibility, allowing you to limit peaks without normalizing and vice versa. For instance, suppose you have a dialog between two characters that contains both whispered and shouted lines. For clarity and ease of use, you've cut up the dialog into separate files, all of which need to be converted into 8-bit. In this situation, you'd add some gain and increase the signal-to-noise ratio prior to conversion. However, if you normalized these files, the whispers (not to mention background ambiance) would end up disproportionately loud, and the flow of the dialog would be lost. Instead, you'd want to maximize the files by adding gain and using the peak limiter to prevent clipping, but leaving the normalization off to retain the relative volumes between the lines. This approach will boost the overall perceived volume of the dialog, and thus it's post-conversion signal-to-noise ratio, while leaving the dynamics intact.

Normalization and maximization do not, however, address the issue of quantization noise in quiet passages. Fortunately, both programs have a noise gate to deal with this problem. A noise gate looks for signals that fall below a certain volume (as quantization noise typically does) and then processes these low-level signals. WaveConvert Pro provides a noise gate with two settings, hard and soft, which silence quantization noise quickly (with hard) or more gradually (with soft). WaveConvert Pro also supplies a rumble filter for removing very low-frequency noise, such as HVAC noise, from recordings. For higher sample rate files (down to 22KHz), WaveConvert Pro offers a noise shaping and dithering scheme called IDR (Increase Digital Resolution). Although I can't remember ever using 44KHz 8-bit files for anything, 22KHz 8-bit files are still a common format. BarbaBatch offers only noise gating and simple dithering features, but the noise gate has more adjustable parameters (Figure 3). This allows for greater flexibility in dealing with low-level signals that can't be altogether removed, such as room tone.

Extensibility

As I mentioned, only WaveConvert Pro can be extended with DSP plug-ins. These external plug-ins include L1 (a high-end look-ahead peak limiter), C1 (a compressor/gate), Q10 (a multiband "para-graphic" equalizer), TruVerb (a digital reverberation and room simulation processor), S1 (a stereo enhancer) and others.

WaveConvert Pro allows you to enter multiple plug-ins and even multiple instances of the same plug-in into the batch process. All settings and the order of the plug-ins are saved in both the settings files and the Joblist files.

WaveConvert Pro includes the full version of the AudioTrack plug-in (Figure 4), which contains a four-band para-graphic equalizer combined with a compressor and noise gate section. AudioTrack adds a tremendous amount of depth and control when used in conjunction with WaveConvert Pro.

File Output

Both programs are very flexible when it comes to organizing the output of a batch process. To process a file in WaveConvert Pro, you need to set all of the settings, including specifying the output folder, and then move the file into the output list for processing. Different files in the output list can have different settings, including different output folders. File names can be changed automatically (extension only or automatic 8.3 conversion) or by double-clicking in the output list and typing in a new name. All of these front-panel settings (including plug-in settings) can be saved as a settings file, which is a long overdue capability in the WaveConvert family.

In addition to the settings, the Joblist saves everything from a session. This includes the path to all of the files to be processed, all of the settings, and the paths for all of the output files. As a text file, it can be edited and then reloaded so that you can run a modified batch without reloading all of the files. WaveConvert Pro even auto-saves a back up Joblist in case you quit (or crash) with out finishing a batch. This back-up Joblist can be reloaded and run at a later time. This is a great feature and a big time saver, but I do have one major problem with the way that it's been implemented: if one of the input files or plug-ins that you use no longer exists along the previously specified path, the Joblist may only partially load. If any of the output folders specified in the Joblist don't exist, it will still load, but may not be able to perform the conversion. This isn't an elegant solution to the problem and seems quite out of place considering the thoughtfulness and care that went into to the rest of the user interface. I take issue with this discrepancy because, in this business, it's common practice to process a large batch of files and burn them (source and output) to a CD for archival purposes. It's great that you can save a Joblist to burn with your audio files, but if I have to restore these files and redo the batch, it's likely that the path for the source files and the output directories won't be the same. Editing the Joblist batch file would be fine for a small batch, but not worthwhile for a large one (such as 2,500 files).

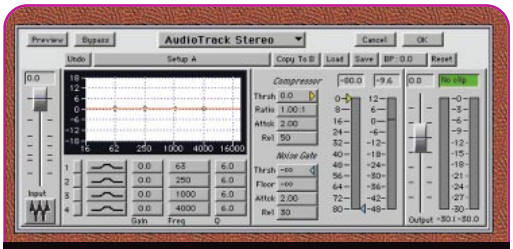


FIGURE 4. *AudioTrack's main control surface.*

On the subject of interface problems, I have one other issue with WaveConvert Pro. Once you've selected a folder for your output, that folder and any files or folders contained therein no longer appear in the "Add Input File" selection box. This is a problem if you have an existing nested file structure and you need to output your processed files to a higher level in your filing system than where your files to be processed reside. Waves has told me that this is to prevent people from writing over their source files, which is understandable, but a dialog box warning would be a more elegant solution.

Further establishing itself as the more feature-rich application of the two products, only WaveConvert Pro has a preview function (and a very nice one at that), so you can hear how certain processes will affect a file. In BarbaBatch, you must fully process the file to hear what it will sound like.

Overall, file output is handled very nicely. WaveConvert Pro gives you an estimate of how much space the output file will take up on a hard disk or CD-ROM, taking the minimal disk physical block size.

BarbaBatch has a much simpler method for handling file output. It works by creating conversions. A conversion consists of a group of settings and a name, similar to the WaveConvert Pro settings files. The names appear in the conversion window. Once you load up your input files, you just check off all of the conversions that you want to apply. You select an output folder and hit Start. While processing, BarbaBatch creates a new directory with the name of each conversion and stores all of the files processed by that conversion inside. This is useful, if not completely flexible, especially when combined with the ability to input and automatically recreate nested folder structures. BarbaBatch also outputs a comprehensive log file for each batch that it processes.

In the end, WaveConvert Pro has a potentially more powerful tool in the Joblist. There are a few things that you can do with a Joblist that you can't do with BarbaBatch, but for the most common applications, BarbaBatch's methodology is simple and well thought out as to the tasks at hand. As I said at the outset, this one will come down personal taste.

Performance

I did a number of conversions on my older yet still functional PowerMac 7100/66 with a Digidesign ProTools II audio card installed (with the 442 interface) to test the products' performance. In general, WaveConvert Pro was somewhat faster than BarbaBatch in most circumstances. For example, one minute of stereo, 44.1KHz, 16-bit digital audio converted to 8-bit and a slightly nonstandard rate of 110,029Hz with normalization and peak limiting took WaveConvert Pro one minute and forty seconds to process; BarbaBatch performed the same conversion in two minutes and thirty seconds.

How Does it Sound?

Perhaps the most important metric of all is how the results sound. Again, I did many listening tests on a variety of files converted in many different ways. I won't present

the full details, but I'll instead give my overall conclusions. To do my listening test, I chose to forgo the rarefied environment of the recording studio and opted instead for a typical high-end multimedia PC listening environment. Using a Creative Labs AWE64 Gold card and a Cambridge SoundWorks PCWorks subwoofer/ satellite-amplified speaker system, I turned the volume up fairly loud to compensate for the ambient noise of the machine. The files I processed were 16-bit, 44.1KHz music and voice-over files from the last game I worked on. All of the files were produced on mid-level professional equipment, such as a Mackie 8*Bus board, AT 4033 microphones, Tascam DAT machines, and K2500 and Samplecell samplers, and were reasonably free from noise and distortion. I also tested some really poor-quality files from an outside contractor just to see what kind of repair work was possible.

BarbaBatch presents a very nice, natural, open sound for all of its conversions. The BarbaBatch sample-rate conversion algorithm is very effective. Music files converted with BarbaBatch retain the most natural, least processed sound. WaveConvert Pro music files tend to have a more processed, filtered sound, even with minimal conversions (such as 44.1KHz to 22KHz only). WaveConvert Pro, on the other hand, did the best job with 8-bit conversions. Its wider variety of signal processing options and excellent set-up libraries yielded much cleaner, quieter low–bit-rate files. WaveConvert Pro was also much more useful in cleaning up my poor-quality audio files.

I also tested the files on an inexpensive set of Labtec speakers. In this case, the WaveConvert Pro-processed files sounded clearer, and artifacts from the sound processing were much less noticeable than on the higher-end consumer audio hardware.

Picking a Winner

Both applications should serve the needs of most game developers very well. In fact, I think that they both represent large steps in terms of sound quality for real-world applications. Allow me to couch my recommendation in practical terms. If you work mostly with well-produced source material to be delivered at reasonably high resolutions (22KHz 16-bit files, for example) for high-end PC CD-ROM titles, then I would say BarbaBatch is for you. On the other hand, if you work in space-limited or unusual environments (such as cartridge games or the Internet), you have a specialized delivery platform (such as a kiosk), or you often need to perform substantial clean-up on files, then WaveConvert Pro is the program for you.

In terms of value for the dollar, my recommendation is slightly different. When I began writing this piece, both programs were priced at around $500. Recent price drops, however, have placed WaveConvert Pro at $300 and BarbaBatch at $399. Given this, WaveConvert Pro is now clearly the better buy of the two products. Right out of the box, WaveConvert Pro is the more feature-rich program, and once you consider the fact that it comes with WaveConvert for the PC, as well as the Audiotrack plug-in, there's really no comparison. BarbaBatch's feature set, while modest in comparison, is nevertheless well designed and covers almost all of the basic needs. BarbaBatch does come with some utilities, but they're mostly for file moving and renaming and don't compare in value to Audiotrack.

Despite some grammatical errors, the documentation of both programs is excellent. Both show how to use the products and explain the underlying technical and aesthetic issues involved. I recommend a thorough reading of both of these manuals to anyone who creates audio for interactive media just for the background information included. ■

BarbaBatch 2.1 for Macintosh

Rating (out of five stars): ✪✪✪✪

Developed by **Audio Ease**, the Netherlands
Distributed by MacSourcery
Escondido, Calif.
760-747-5995
www.macsourcery.com

Price: $395

Software Requirements: System 7.1 or later

Hardware Requirements: Power Macintosh, or a Macintosh with the series 68881 Floating Point Unit installed. (SoftFPU will also work)

Pros:

1. Great sounding sample-rate conversion.
2. A simple, elegant user interface.
3. Reads and writes a wide variety of file types.

Cons:

1. Bit-rate conversion and quantization noise tools could be better.
2. It has somewhat limited DSP options.
3. More expensive than WaveConvert Pro on a feature-by-feature basis.

WaveConvert Pro 2.3 for Macintosh

Rating (out of five stars): ✪✪✪✪✪

Waves Ltd.
Knoxville, Tenn.
800-264-0109
www.waves.com

Price: $300 (includes AudioTrack and WaveConvert for Windows)

Software Requirements: Sound Manager 3.1 or higher, and QuickTime 2.5 or higher (the latter two are installed by the Waves installer). The PowerMac version also requires the ObjectSupportLib, which is also installed by the application. Also available for Windows.

Hardware Requirements: Power Macintosh, or a Macintosh with the series 68881 Floating Point Unit installed.

Pros:

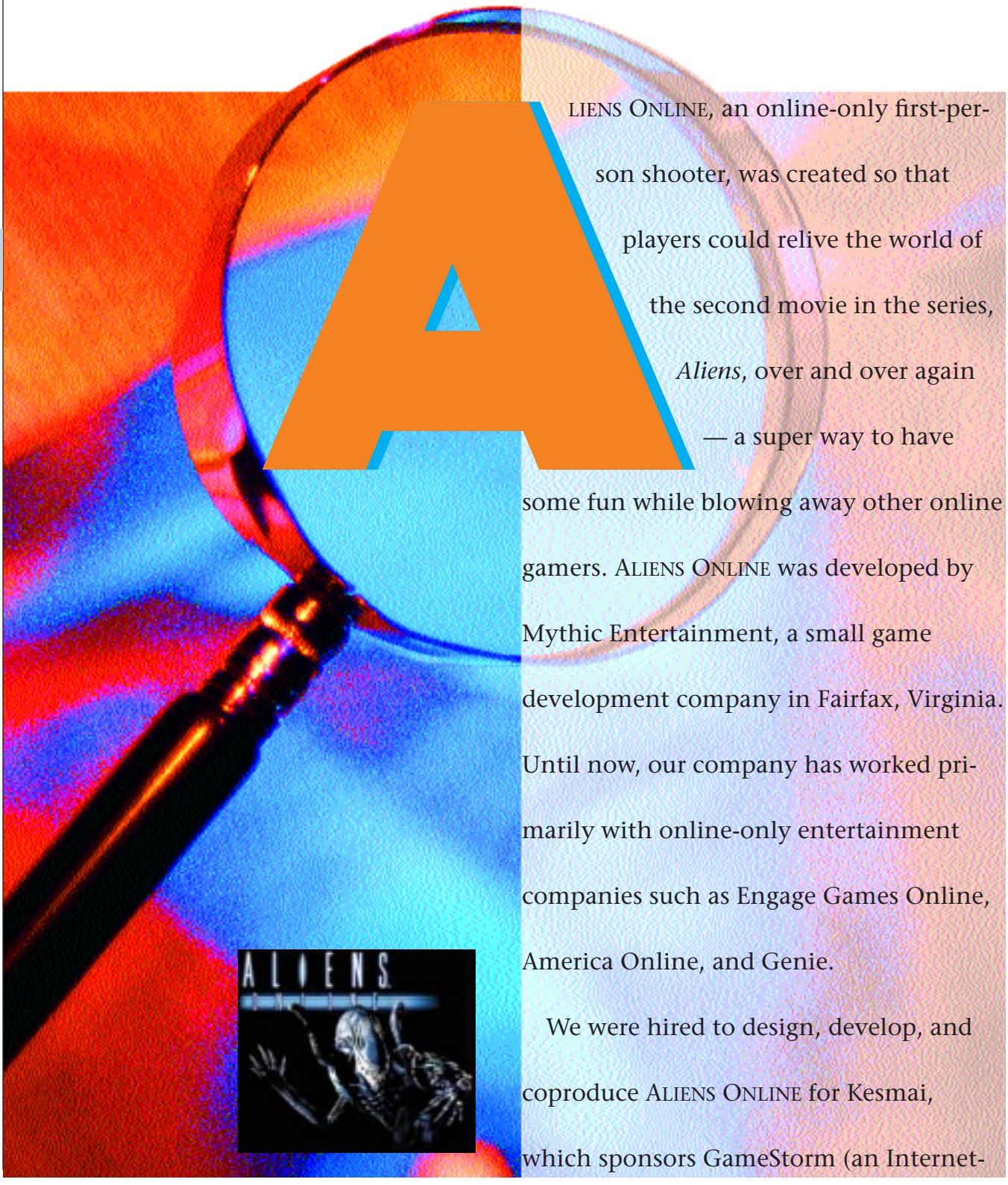
1. Extensive signal processing options and extensible architecture.
2. With the inclusion of AudioTrack and WaveConvert for the PC, it provides big bang for your buck.
3. The Joblist feature saves time when reprocessing large batches of files.

Cons:

1. Sample-rate conversion could be better.
2. A few, minor UI irritations.
3. The Joblist doesn't respond well when the paths of files have changed.

ALIENS ONLINE

by Matt Firor



ALIENS ONLINE, an online-only first-person shooter, was created so that players could relive the world of the second movie in the series, *Aliens*, over and over again — a super way to have some fun while blowing away other online gamers. ALIENS ONLINE was developed by Mythic Entertainment, a small game development company in Fairfax, Virginia. Until now, our company has worked primarily with online-only entertainment companies such as Engage Games Online, America Online, and Genie.

We were hired to design, develop, and coproduce ALIENS ONLINE for Kesmai, which sponsors GameStorm (an Internet-

based gaming network) and provides games to America Online. Our liaison for the project was Jason Bell, the game's executive producer at Kesmai. Jason repeatedly proved that managing a third-party project can be accomplished efficiently and with a sense of teamwork.

Kesmai is owned by News Corp., which also owns Fox, which explains how Kesmai acquired the *Aliens* license. In fact, Kesmai handled all aspects of the license for us — we never had to raise our heads out of the code, design, and art during the lifetime of the project. All the intercompany politics and intrigue were kept away from us, which kept us focused on our task.

Kesmai provided a wonderful marketing and advertising campaign for the game, a key feature lacking in our other titles. In fact, I can't think of a major game magazine in the last three months that has not run an ALIENS ONLINE advertisement. Kesmai also did a West Coast press tour to promote ALIENS ONLINE and the launch of GameStorm, which raised even more interest in the game industry press.

The Joys of Working with a Great License

Most games based on movie licenses don't live up to their names. I don't have to recite the long, sad litany of glittery but shallow and just plain pathetic games based on Hollywood box-office smashes. Their pale memory lives on in the clutter of unfinished game CD-ROMs and boxes that litter every game developer's desk and can be found shortly after release in the bargain bin at the local software store.

For many reasons — usually impossibly short development cycles — these Hollywood-licensed games just never get off the ground. Another excuse bandied about in the game industry, probably by disgruntled producers, is that licenses are too restrictive to make good games. Notable exceptions to this rule are the *Star Wars*-based games from LucasArts.

Because *Aliens* is Fox Interactive's premier license, and because we at Mythic, an out-of-house developer, would be designing and developing the game, Kesmai had to put a lot of effort into getting the license from Fox. So not only did we start off with a great license, we also knew we had the full support of Kesmai's management because of their work in securing the license for us.

The Design

ALIENS ONLINE was designed in a little over a month by Mark Jacobs, the president and lead designer at Mythic Entertainment. The design called for a relatively simplistic game in which players can join one of two sides (colonial marines or aliens) and attempt to annihilate the other. The overarching goal of the design was to retain the creepy, gloomy, dark look and feel of the movie.

The original design called for colonial marines, drone aliens, queen aliens, and face huggers (proto-aliens that scurry about implanting hapless humans with deadly alien parasites). Additionally, artificial persons (androids, for the dozen people reading this who haven't seen the movie yet) and colonists were designed, but had to be dropped from the game during the production process, due to memory constraints.

The employees of Mythic Entertainment, squinting into unfamiliar sunlight. The author is at far right.

During game play, the colonial marines try to hunt down and kill all the aliens using an array of weapons straight out of the movie. The aliens get a claw and bite attack. They also get a tremendous speed advantage over the marines, as well as a leaping ability that lets them jump into air ducts in the ceiling to hide from colonial marine gunfire.

Our two game play goals were to make the game team-oriented and to make it fun to play both sides while giving neither a clear advantage. The first goal was accomplished by organizing colonial marine players into fire teams, a concept right out of modern-day Marine Corps tactics. A fire team is a group of four colonial marines that work together, scouting and covering each other as they make their way through the infected colony. Fire team-work is helped by perhaps the best feature of the game: miniature 3D views showing fire team members their teammates' views (a concept also used in the movie to track the whereabouts of each person).

We accomplished our second game play design goal by making sure that the alien's speed and leaping ability compensated for the marine's devastating firepower. Eventually, we gave the aliens the ability to see all of the colonial marines in the mission on their map, making it easier for them to hide and sneak up on them.

The design had to take into account the game's extremely short development cycle — only ten months. This is ludicrously short by most standards, so the design reflected the fact that we would be using as much existing technology as possible.

Aliens Art

Even before the ink was dry on the ALIENS ONLINE contract, Missy Castro, lead artist on the game, bought a copy of the movie. She made extensive use of her Snappy video capture board to make textures for the game's different arenas. As much as possible, all textures inside the game came from the movie itself, lending an air of instant familiarity to any player who had seen the movie.

Matt Firor is the producer of ALIENS ONLINE. He lives in Arlington, Virginia with his guitar, cat, and Mr. Coffee. He can be reached at mattf@mythicgames.com.

The colonial marine character models were created using 3D Studio MAX and then touched up in Photoshop. The resulting model was rendered into the twelve different views used by the game. Each figure required several different animations: walking, crouching, death, and so forth.

The first big problem that we overcame was the game's art budget. Each figure, essentially a series of large bitmaps, took up a lot of game memory, and we quickly realized that we didn't have enough RAM to make many different types of male and female avatars. Rob Denton, the game's lead programmer, solved this problem by devising a scheme whereby the user could select a combination of different heads, bodies, and legs, sort of like a paper doll. During game play, the engine assembled these different pieces based on the player's specification.

Rob also came up with another great method to reduce the game's memory footprint. In the game, players can choose from many different heads, five different torsos, and five legs. To save resources, Rob used just one torso and switched the object's color palette depending upon the color selected by the player. Each object in the game is stored in an 8-bit palletized format that is dynamically rendered in high color via a light table. So the five torsos and five heads — each for male and female — ended up taking up very little memory because there was essentially only one copy of each.

Alas, even with the palette swapping, we just didn't have enough room for the many different types of human figures originally designed for the game. Thus, artificial persons and colonists were removed from the design, making the game solely a colonial marines vs. aliens game.

Client/Server Architecture

ALIENS ONLINE is an online-only game. You can't go into a store and purchase it; you download it from the commercial game service and play it online. As such, the game uses a client/server model. The game's client holds all the art and sounds and displays them appropriately. The server — running on a HP-UNIX machine — holds all the positional information and keeps each client updated on where each player is.

Keeping the amount of information passed between the client and server small and concise was a technical goal for the game's programmers. The more streamlined this communication, the more smooth and efficient the server's propagation.

Graphics Engine

ALIENS ONLINE was developed using Mythic Entertainment's RAZE engine, which was also used in our other games, ROLEMASTER: MAGESTORM and SPLATTERBALL. The short development cycle for ALIENS ONLINE meant that we had to use as much existing technology as possible. RAZE is a DirectX-based engine that was developed using Microsoft Visual C++ with some assembly routines. The RAZE engine is essentially a raycasting engine with flat bitmap character and object graphics. We knew when we started the project that ALIENS ONLINE would probably be the last game developed using the RAZE engine; as such, the engine was tweaked and modified in order to extract as much technical mileage from it as possible.

Even though today's hardware-accelerated 3D engines make the RAZE engine seem dated, RAZE can display high-color graphics without requiring hardware acceleration — somewhat revolutionary in its day. This is accomplished by having 32 different palettes to display object, texture, and character images. We stretched this palletization even farther by splitting the game into alien palettes and colonial marine palettes. This way, the alien view of an object or texture can be substantially different from the marine view. In the game, the alien view has a reddish, insect-like look, which is a result of the alien palettes having more of a reddish tint than the marine ones.

The RAZE engine also supports "areas on top of other areas," a concept that isn't supported by many other (if any) raycasting engines. While RAZE is more limited than a true polygonal engine, these stackable areas are used to great effect in the ALIENS ONLINE missions in which aliens can sit above a colonial marine, drop down though vents in the ceiling (much like in the movie), and jump back up. RAZE also supports sloped floors and voxels, which makes the engine able to display non–right-angle architecture.

Things That Went Right

1. **SERVER INTEGRATION.** On the server communication side, we integrated our existing RAZE server with ARIES, Kesmai's server API. ARIES has been around for over ten years, and was a joy to work with. We had no trouble integrating it into our existing server base. Since ARIES has been around for so long, it's well documented, stable, and runs smoothly. ARIES had most functions that we needed built-in, and didn't require much, if any, additional work on our side.

   Integrating with ARIES gave us instant compatibility with Kesmai's existing hardware server structure. In order to ensure that our code was compatible with their machines, all we had to do was make sure the server was ARIES-compliant.

2. **THE GAME'S SOUND.** Jason Bell insisted from the beginning that the game be as aurally advanced as possible. His goal was to make the game's sound as suspenseful as that of the movie. In order to do this, we were allowed to use Kesmai's proprietary sound system, KSound.

   Right off the bat, we realized that KSound's biggest advantage is that it supports a compressed sound format — essential to keeping downloads to a manageable size. To trim the size down even farther, more focus was placed on

ambient noise and orchestral stabs rather than a full-blown score.

By using KSound, we also got to experiment with BEDLAM (Bodacious Event-Driven Language for Audio Manipulation), Kesmai's sound-scripting system. BEDLAM gave us a random sound generator to add ambient noises, bullet ricochets, different player noises based on their health, and a host of other aural benefits.

Using BEDLAM was a risk, though. ALIENS ONLINE would prove to be the first game that extensively used the system, and we spent many hours during development listening to misplaced sounds, wondering whether it was our fault or BEDLAM's. In the end, though, all the problems were ironed out, and we got the benefit of compressed sounds, and a good, solid sound-scripting system.

3. WHAT A GREAT LICENSE! We lobbied hard with Kesmai to get the ALIENS ONLINE deal, and when I heard we were going to get it, my first thought was, "What a great license." Indeed, *Aliens*, the second movie in the series, adapts itself quite easily to a game format.

Another benefit of working on a great license such as *Aliens* is that it generates instant curiosity and media attention. We had a difficult time getting the world to care about our first RAZE engine game, ROLEMASTER: MAGESTORM, but as soon as the press release announcing ALIENS ONLINE went out, our web site was bombarded with hits and questions about it. Kesmai and their marketing blitz did a great job of riding this wave of initial curiosity and answering press queries. So far, ALIENS ONLINE has generated more press than all of our other games put together.

Our plan was to get our foot into the movie-licensed–game door with ALIENS ONLINE and then branch out from there. So far, we've parlayed our work on ALIENS ONLINE into two other licenses, those for STARSHIP TROOPERS: BATTLESPACE and GODZILLA ONLINE, both based on the movies of the same names.

4. SPLITTING THE ART BETWEEN COMPANIES. As soon as the design phase of the project was over, it was obvious that we needed more art than our resources would allow. So, we teamed with Bob Frizzell, Rich Sisson, Greg Grimsby, Benjamin Tanaka, and Frank Williamson of Kesmai's art department to create some of the game's static cut screens and other pieces that didn't involve integration with the game's graphic engine.

Our usual process was to create an initial screen as a placeholder and ship the game as a whole to Kesmai with an explanation of which screens needed replacing. Their artists went ahead and replaced these screens and e-mailed them back to me, so I could drop them into my copy of the game. If I liked the art — and I usually did — I put Kesmai's version of the screen into the release directory of the game, and it was automatically included in the next release.

By dividing the art workload, we were able to create a game with an extremely high production value. We would never have been able to create as much quality artwork by ourselves — the art produced by the Kesmai Studios artists adds the critical look and feel of the movie to the game.

5. USING EXISTING ENGINE TECHNOLOGY. Using our RAZE engine helped the project tremendously. Because we'd already developed two games based on the engine, everyone on our team knew their roles and how to perform them. The arena designer, Colin Hicks, was already proficient in the use of the arena editor, and was able to start making arenas long before the game was even close to completion.

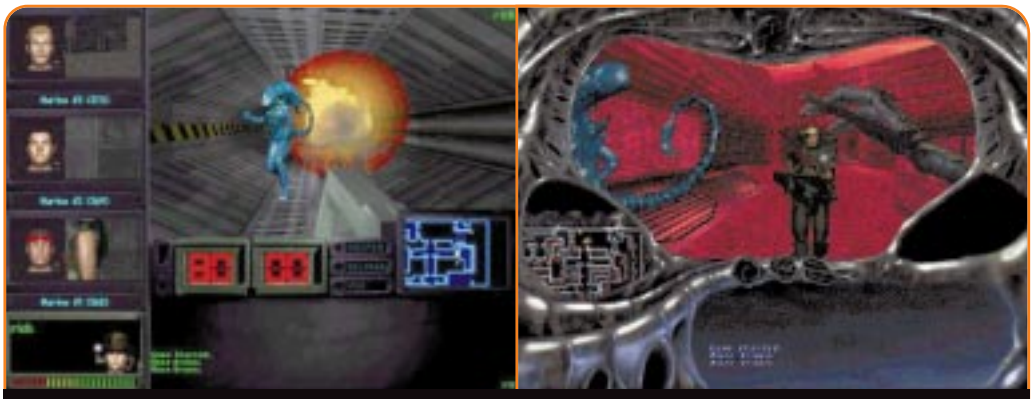
Because two other RAZE-based games had been running on America Online for about six months before ALIENS ONLINE development started, we had a good idea of the performance that we could obtain, and we designed the game around that. More often than not, baselining of ALIENS ONLINE tasks to our previous RAZE projects worked well.

Finally, our other RAZE-based games provided a wonderful test site for ALIENS ONLINE. If a bug popped up in one of them, we knew with relative certainty that the same bug would probably appear in ALIENS ONLINE eventually, and we acted appropriately.

We Should Have Known Better...

1. REMOTE DEVELOPMENT OF IN-GAME MODELS. Working with the Kesmai art department to create additional art was one of the high points of the development process. When given projects that entailed static, nonanimating graphics, they did a wonderful job. However, they also requested the task of creating the animated alien models — that of the drone, the face hugger, and the queen.

Like any other first-person game engine, a close relationship exists between figure animations and the code that drives them. Unfortunately, having



Colonial marine and alien players have different user interfaces that use different color palettes.

Textures were often created from video stills captured directly from the Aliens *movie.*

the artist who created the animations a hundred miles away hindered the process. ALIENS ONLINE was the first RAZE game that required employees of other companies to be involved in the development. We couldn't explain the idiosyncrasies of the engine — and there were a fair number — well enough over the phone.

As a result, the art staff at Kesmai had trouble figuring out our somewhat arcane art implementation process. Because of this, the first versions of the animated alien had too many frames and skated like Michelle Kwan around the arena. We eventually got the animation keyed to the engine correctly, but it took many months of phone conversations, tweaking, and heartache.

2. SERVER, NOT CLIENT, OPTIMIZATION. About five months before the project was scheduled to be completed, we noticed that the game's server was taking up too much CPU time, which limited the number of people that could play the game simultaneously on one server. This wasn't a problem with our earlier RAZE-based games, as we never had the demand to warrant it. However, the imminent launch of GameStorm, which is extremely inexpensive for players, meant that we had to ensure that thousands of online gamers could play ALIENS ONLINE simultaneously.

We had to spend almost three weeks going through the server code line by line, profiling each task in an attempt to figure out what was taking so much CPU time. At the time, we knew that server optimization would be done at the expense of client optimization, and that the game's frame rate would suffer.

We were successful at getting the server CPU numbers down to acceptable levels by the game's launch. However, during the first month of the game's open beta and live period, we were hammered by users for poor client engine performance. Almost all the users had great things to say about the game, but it simply didn't run well on anything under a Pentium 166MHz. We've released several patches since the game went live, and each gives an incremental streamlining of the client frame rate. In hindsight, maybe we should have bit the bullet, limited the number of players on one server, and taken the time to make each player happy by making the game run better on the client side.

3. FRONT END HACKING. One of the least-publicized, but quite common problems in online gaming today is dealing with hackers who swarm around any new online game and attempt to figure out ways to cheat. All online games, in one way or another, are susceptible to hack attempts. Online games are more popular to hack than box games, as online games are inherently head-to-head, and an advantage gained by cheating translates to more kills and status.

We knew going into the project that we could store no player data whatsoever on the client's hard drive. All data, including character stats, hit points, propagation, and so on was designed to come from the server. However, we left one large gaping hole that hackers immediately discovered: editing the computer's memory while the game was running. The client never stored data on the client's hard drive, but it did download data from the server into a section of memory. Intrepid hackers found the memory area and quickly developed a program that let them hack their experience and other character attributes.

As soon as we realized this, we corrected. The game code now double-checks the client value against the host value each time an update is issued. If the two aren't in sync, the client shuts down with a memory violation error.

4. DESIGN OUT OF SCOPE. When working with a great license, it's easy to fall into the trap of trying to create a game that completely covers all aspects of that license — without taking into consideration the time it takes to implement all the fantastic features. *Aliens* is full of features that could be used in a game — androids, a fully-functional alien life-cycle complete with chest-bursting larvae, and armored personnel vehicles. Simply put, there was a lot to draw from, and we drew most of it for the game's design.

Of course, the project had a firm, fixed, finite release date that could not change, due to the fact that it would be launching around the same time as the GameStorm network itself. So, instead of lengthening the project, we had to cull out some of the great features from the design.

It was a plus — as I mentioned previously — that our relationship with Kesmai allowed us to alter the design. Still, explaining our design decisions took a lot of meetings and wasted time that wouldn't have been necessary if we had worked within realistic parameters to begin with.

5. PUSHING RAZE TO ITS LIMITS. Colin Hicks was given a mandate at the beginning of the project to make the game's missions as intricate and eye-popping as possible. The results of his efforts are missions that are stunning, immersive, and really convey the feel of the movie.

The arena architectures were by far the most complex of any RAZE engine-based game. As such, the raycast scenes often were too tough for the engine to handle — it just crashed. As a result, much of the first couple of months of the project was spent ensuring that the engine was robust enough to handle the arenas.

We made the mistake of assuming that because we had developed other games with the RAZE engine, we wouldn't find many raycasting problems. Stretching the engine to its limit with complex architecture showed us how many problems were lurking just under the surface.

New Nifty Features

As I write this, we are working on an extension to ALIENS ONLINE that will add many new nifty features. Part of the fun of an online game is that you can work on it for as long as people want to play it. Updates are only a patch download away… ■

by Dave Perry

Patents Are Like Safe Sex

Software patents are yet another way for attorneys to turn us all against each other. That said, they are able to describe the concept rather well.

"It's like a giant condom. Because David, we want you to be safe."

"It protects you, and you make money from it…"

"We write the patent and we file it for you. It couldn't be easier."

Sounds too good to be true.

In the past, I spent/suffered through hundreds of hours with attorneys to learn in-depth the licensing, merchandising, and negotiation side of video games. It's such a tight community — political, aggressive, backstabbing, and rewarding. For game developers, we did well, getting an Earthworm Jim TV show, toy line, comics, fast food, underpants — you name it.

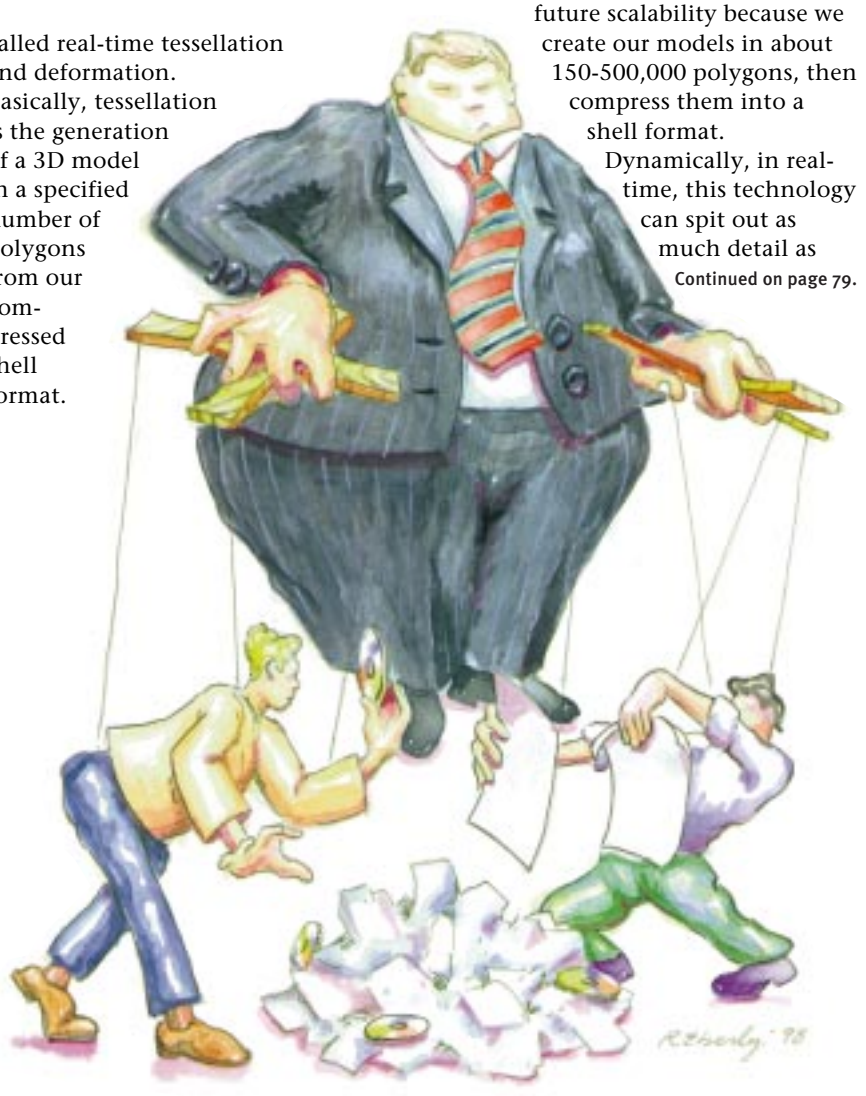
So when my attorneys knocked on my door, and patents was the subject, I listened and learned. Then I realized that they were at it again.

We all understand that if you write some bloody amazing compression program, it can take considerable effort, time, and money. Then, when every copycat hack in every country steals your work, you're screwed. Legally, you can stand on a copyright soapbox and make noise. But in practice, the laws vary from country to country, it will take too long to examine code, and you've lost control of your work.

Patents are more universal and more respected than copyrights. Nobody (unless they're stupid) is going to take clearly marked, world-patented code and base another product around it. Now, as there are a lot of stupid people, this happens all the time. That's why these attorneys have such a great business (and such impressive offices).

The technology we have at Shiny is called real-time tessellation and deformation. Basically, tessellation is the generation of a 3D model in a specified number of polygons from our compressed shell format. It allows insane levels of detail for free, and actually enables the game to speed itself up by intelligently deciding which important polygons to retain and which less important ones to discard. Tessellation also allows easy interpolation. It also allows future scalability because we create our models in about 150-500,000 polygons, then compress them into a shell format. Dynamically, in real-time, this technology can spit out as much detail as

Continued on page 79.



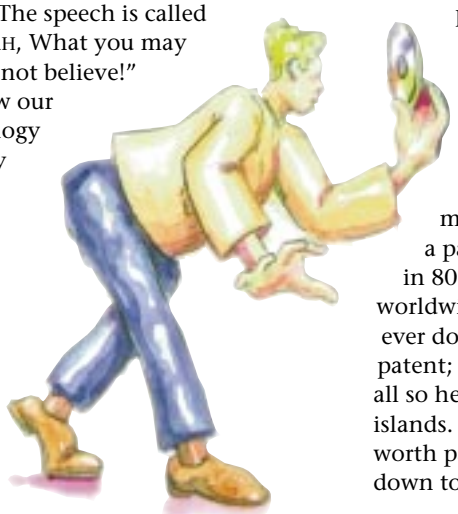
David Perry began his career in the game development industry thirteen years ago as a young guy in County Antrim in Northern Ireland. His assets at the time were a ZX81, a bad haircut, and a passion for videogames. Since then he's had 37 games published. These days he's President of Shiny Entertainment where he produces hits such as EARTHWORM JIM and MDK.

Continued from page 80.
any processor/video card combination can promise in the next three years. In short, we store more detail than we can see today.

Deformation means that every bone in the character's body can influence any part of any body surface in any way during any frame. Skin stretches, muscles bulge, cloth pulls. Because we algorithmically regenerate every polygon every frame, we can distort geometry to create subtle nuances of difference between characters (two similar people might have different heights or head sizes). Combined with motion capture, deformation really breathes life into 3D. It looks fantastic.

So here's the problem: tessellation as a concept has been around for ages. Deformation as a concept has been around for ages. But when I said publicly that we were thinking of patenting our Compression, Tessellation, and Deformation Decompression for Video Games technology, I got flamed. Cocky programmers told me that I was every color of slime, and that they had been "doing it for ages." Wrong, I replied. Shiny was the first company to ever display real-time tessellation, deformation, and volumetric lighting in a full 3D game engine, and we showed it to the press at the E3 show 1997. The press agreed. To this day people are talking the talk, but I have seen no competitive demos. Then I was told that I was a liar and that it was impossible.

So, to stop the arguing, I'm giving a speech at the Computer Game Developers Conference in Long Beach with Saxs, the MESSIAH team leader. The speech is called "MESSIAH, What you may or may not believe!" I'll show our technology publicly for the first time. I'll also post video to www.shiny.com for those who can't attend. I feel that with the time and effort we have poured into this technology, it should be an inspiration to other programmers. It's a fresh new way to think, and it's the future. Any leading successful company will have to do something similar in the future. John Carmack himself is on record stating that fact.



In id's case, some uninformed programmers think they give away their code for free. Hmmm… They actually sell some through Michael Abrash's books, and the complete engines cost major bucks to license (just ask John Romero). Needless to say, Shiny will also be licensing the MESSIAH technology.

The problem is that when you think you have something, no matter what it is, the attorneys will con you. They'll tell you that you can patent just about anything. They make you feel like a star, as if everything you've done is patentable. This is the funny part! They charge you while you laboriously explain your code line by line, then they charge you to search it, then they charge you for the letter that says "Oops there is something similar out there, but if we change our application, we could try again," then they charge you to make the re-application. This can go on forever.

Now before you turn the page and think, "Sheesh, forget patenting; glad I never wasted my time on that," let me tell you about a flight I had in a private plane over amazing islands in Fiji. Now, the island I was going to was bought from the air by a guy pointing, "I'll have this one, that one, oh, my staff can have that one." He 's a multi-multi-multi-millionaire who has a patent on a fiberglass gun used in 80 percent of boat-building worldwide. It's the only thing he has ever done. Design the gun; file the patent; let the attorneys take care of it all so he has time to go and pick out islands. So how do you decide what's worth patenting so you can head on down to Fiji? Some tips:

1. Try to find a patent attorney who has handled technical software cases before. Nothing is more annoying than trying to describe in low-level detail your latest tessellation routine when this guy is an expert in flowerpots.
2. Don't expect your programmer to deal with the attorneys. They'll abuse his time and he'll be bored to death. Find somebody else technical that can handle most of the laborious paperwork.
3. Act early; you only have one year from the first public showing to get your patent on file.
4. Document everything as you go along. This saves tons of time when you don't have to start from scratch.
5. Search everything yourself. It gives you a much better understanding of what the attorney will stumble across, and you'll have already adjusted your pitch. Here's how to search. Go to the following sites:
   - http://patents.uspto.gov/ US Patent & Trademark Office
   - http://patent.womplex.ibm.com/ IBM Patent Server (my favorite)
   - www.spi.org/freesuseragr.htm Software Patent Institute

   Search for key words in your idea. Try searching for "Rubik" to see how it works.
6. Get a bid early from the attorney in writing. One attorney said $10,000 to file, then a few days later said $25,000. They just pick numbers from the air. The game they play is "Until we see exactly what the patent is, we can't give accurate figures." So you detail it out more; so they jack up the price. That is where (4) and (5) can save the big bucks.

So, to wrap up: Companies such as Microsoft, Apple, IBM, Sega, Nintendo, Atari, and Sony have tens of thousands of patents. They patent everything they can. It improves the value of a company vastly to have a really great number of patents. If you're ever planning to sell your company, you would be stupid not to at least consider it. But that's a whole different article. ■