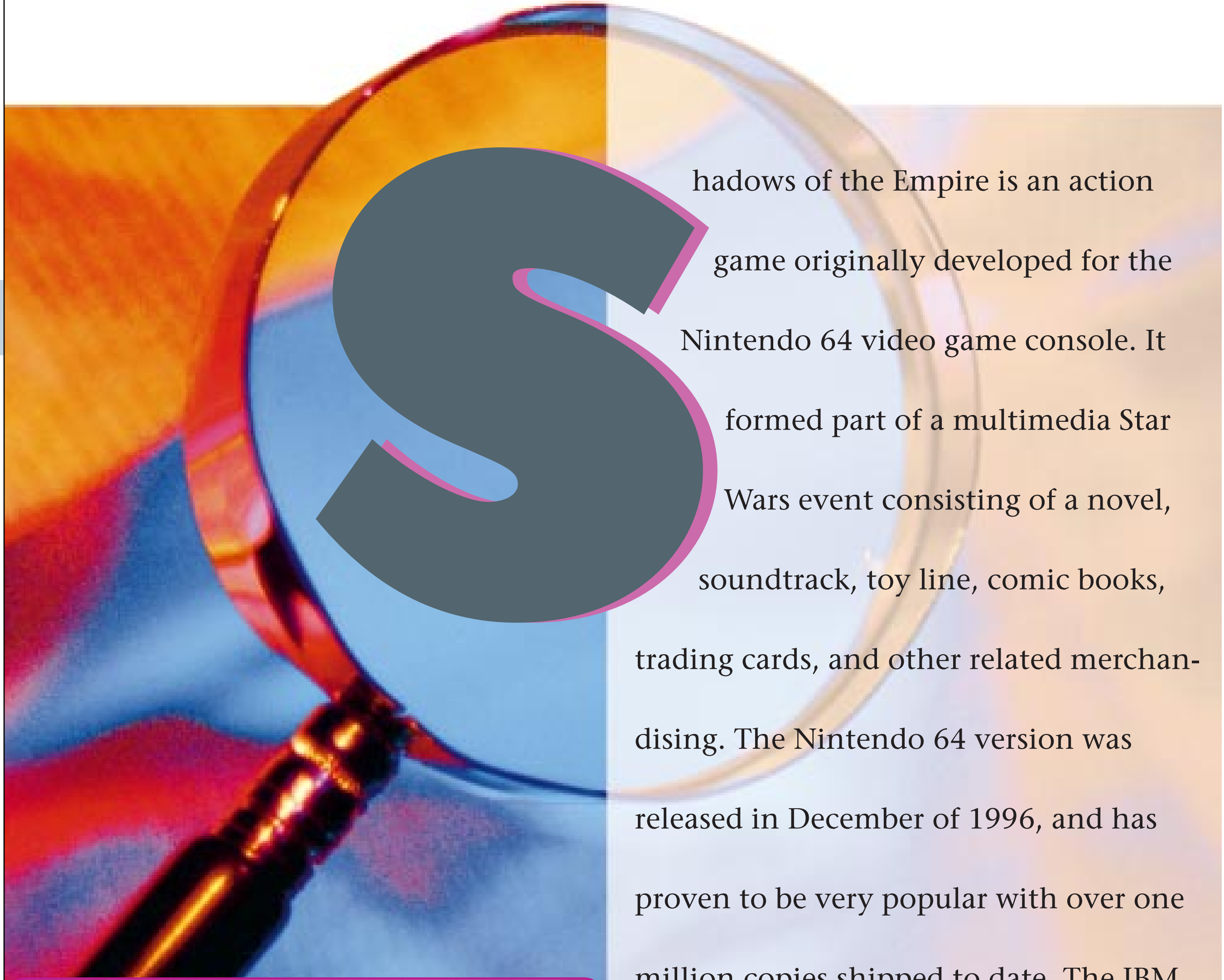
Back

# Star Wars: Shadows of the Empire

by Mark Haigh-Hutchinson



Shadows of the Empire is an action game originally developed for the Nintendo 64 video game console. It formed part of a multimedia Star Wars event consisting of a novel, soundtrack, toy line, comic books, trading cards, and other related merchandising. The Nintendo 64 version was released in December of 1996, and has proven to be very popular with over one million copies shipped to date. The IBM PC version was released in early September of 1997, and has enhanced cut scenes, Red Book audio (both music and voice), and high-resolution graphics. It requires the use of a 3D accelerator card.

*Dinner in Kyoto, Japan, August 1996. (Left to right: Don James, Hiro Yamada, Mark Haigh-Hutchinson, Shigeru Miyamoto, Kenji Miki.)*

Images courtesy of LucasArts Entertainment Company LLC, © 1997.

## Why Shadows?

Back in the summer of 1994, LucasArts was exploring the possibility of developing a new 3D title for one of the emerging "next-generation" platforms. After some discussion, the Nintendo 64 was decided upon as the platform of choice, even though there was no hardware available at the time. Due to our close relationship with Lucasfilm, we were aware that Lucasfilm Licensing was planning the Shadows of the Empire event. Jon Knoles, the lead artist and designer on the Nintendo 64 game, took an active part in deciding the timeline of Shadows. He suggested that it take place between *The Empire Strikes Back* and *Return of the Jedi*.

The Shadows story line deals mainly with the criminal underworld of the Galaxy, and the new period allowed us to explore some of the things that weren't explained in *Return of the Jedi*. It also opened up some new characters that were not bound to the original story, which gave us more creative freedom than using established figures. A bonus was that it allowed us to make use of everyone's favorite bounty hunter, Boba Fett.

Since we were developing one of the premier titles for an entirely new game machine, there was a conscious decision to attempt to stretch out and cover a number of different game-play styles. We wanted to ensure that the player would have as much variety as possible, yet still enjoy a satisfying experience.

## A Reality Engine for $200?

By early September 1994, we had received our Silicon Graphics workstations and the core team was working. Initially the three programmers were using Indigo 2 Extremes, with 200mhz CPUs, 64MB of RAM, and 24-bit graphics. Eventually, we would have to change our programmers' computers to INDYs (still powerful machines) to install the Nintendo 64 development systems.

In addition, we were fortunate that LucasArts allowed us to obtain a Silicon Graphics ONYX supercomputer. This impressive and somewhat expensive refrigerator-sized computer boasted Reality Engine 2 graphics hardware, four R4000 CPUs, and 256MB of RAM. It became an essential part of our development equipment, as it was the only hardware available that could possibly emulate how the final Nintendo 64 hardware would perform. Indeed, Nintendo and SGI supplied us with software that emulated most of the features that the real hardware would support.

In late September, the programmers took a trip down to Silicon Graphics to discuss the Nintendo 64 hardware design with its chief architect, Tim Van Hook. The SGI engineers were rightly proud of their design, and promised that they would deliver hardware matching the ambitious specifications. Nine months later, we learned that they had indeed met those specifications.

By Christmas of 1994, we had the basis of the first level of the game, The Battle of Hoth, running quite nicely on the ONYX — "quite nicely" being in high resolution (1280×1024), 32-bit color, and at 60 frames a second. By this point, we had also received a very early prototype of the Nintendo 64 controller. This consisted of a modified Super Nintendo controller with a primitive analogue joystick and Z trigger. Due to our strict nondisclosure agreement, we were unable to discuss the hardware or the project with anyone outside the core team. Consequently, we would furtively hide the prototype controller in a cardboard box while we used it. In answer to the inevitable questions about what we were doing, we replied jokingly that it was a new type of controller — a bowl of liquid that absorbed your thoughts through your fingertips. Of course, you had to think in Japanese....

In July of 1995, we received our first actual hardware as a plug-in board for

the INDY. This later became known as the Revision 1 board, but on inspection it was extremely "clean" — no wire wraps or other temporary items in sight. Within three days, technical lead Eric Johnston and second programmer Mark Blattel had ported the game to the actual hardware. It was an awe-inspiring moment when we first saw the Battle of Hoth running on the "real" machine. The first revision of the hardware was very close to the original specifications supplied by SGI. Other than the RCP (Reality CoProcessor) not running at quite the final speed, and one of the special video "dither modes" not being available, it performed extremely well.

Over the next few weeks, we would receive an additional two boards, so that all the programmers were developing in a similar fashion. Three months later, we would receive Revision 2 boards, which brought the RCP up to full speed as well as fixing a few minor bugs. Another pleasant surprise was the doubling of the amount of RAM to 4MB.

A further development was the hardware "dither modes" that perform several different kinds of functions at the video back end — mostly to reduce the effect of Mach banding, which is common when using 16-bit color.

## Technology

Since Eric Johnston and Mark Blattel had extensive experience with the SGI platform, we undertook to prototype the game using the Performer 3D API. This is an OpenGL-based system that is very flexible. Eventually, we would write our own

*Mark Haigh-Hutchinson is a Project Leader and Senior Programmer at LucasArts Entertainment Company. He has been developing computer and video games as a hobby since 1979, and professionally since 1984. Cutting his teeth on numerous 8-bit computers such as the Sinclair ZX Spectrum, he has contributed to 32 published games, 16 as sole programmer. He may be reached at mhh@lucasarts.com.*

*Jon Knoles directs actor Amos Glick who recoils from an imaginary shot. Mark Haigh-Hutchinson wrangles the cables and provides a supporting hand.*

subset of Performer's functionality on the Nintendo 64. This allowed us to move the game from a $140,000 SGI ONYX to a $200 Nintendo 64 in a matter of just three days.

Level designers used the tool set from DARK FORCES to construct the first-person levels for the game. This allowed a crude form of preview using the actual DARK FORCES engine on an IBM PC. This worked fairly well, although later in the project we were able to have a single SGI for dedicated use by the level designers. The PC solution, however, was also useful because the level designers were already familiar with the processes involved. Unfortunately, since the game engine wasn't running on the PC at that point, the development cycle was somewhat slow.

Additionally, the ONYX calculated the preculling visibility tree for each of these levels. The way it works is quite elegant, thanks to Eric and Mark. The world is subdivided into "sectors" — that is, polygonal regions defined by either geometry or some other criteria. These sectors control collision detection, have properties relating to game play, and perform several other related functions. The visibility program traverses the world rendering the scene from the center of every sector in a 360-degree arc as well as three elevations. For every polygon to be rendered in the scene from a particular sector, an identifying 32-bit value, rather than texture information, fills the appropriate pixels in the frame buffer. It's then a simple matter of reading the frame buffer to determine which sectors are visible from that location. This process became known as "pastelization" because the identifiers written into the frame buffer (effectively as RGBA values) caused the scene to appear as purely pastel colors.

## Motion Capture

In the spring of 1995, we decided to experiment with the use of motion capture to control the animations of the main character as well as enemies such as Stormtroopers. Fortunately for us, our sister company, Industrial Light & Magic (ILM), had a capture system available for use. It was a tethered system, using a magnetic field to determine the position of each of the sensors. The sensors were attached to the actor at 11 locations using a combination of a climbing harness, sports joint supports, bandages, and Velcro strips.

The nature of the system presented several problems. First, the actor had to perform on a raised wooden platform, since the metal construction supports in the concrete floor would affect the capture system. Secondly, since the actor was on a platform as well as tethered, we couldn't obtain a "clean" run cycle. Some of our more ambitious motions also proved problematic. On the positive side, once the system was calibrated, we were able to capture over 100 motions in a single day, each with two or three different "takes." We viewed the motions in real time on a SGI Indigo 2 Extreme computer running Alias PowerAnimator. This allowed us to quickly ensure that every capture was "clean" before continuing with the next action.

Unfortunately, we were to discover that after analysis, the motion data proved to be unusable. This was mainly because the angle information for the joints wasn't consistent on its representation of the direction around each axis. Consequently, all the animation for the characters was redone by hand, a somewhat time-consuming task.

## MIDI Music

Our initial approach to music for the game was similar to that taken on some of our PC titles — namely, a MIDI-based solution. However, the first problem that we came across was hardware incompatibilities between the MIDI keyboards used by our musicians and the Silicon Graphics computers used to develop the game. The theory was that the compositions could be previewed directly on the Nintendo 64 hardware as a musician played them on a keyboard. Naturally, this would provide the best possible feedback to the musician. Unfortunately, for some unknown reason(s), note on/off pairs were lost, causing chords to sound as one note. Additionally, note releases were sometimes missed completely. Before long, other unwelcome behaviors surfaced. We worked around these mysteries by having the musicians capture the sample set and play it solely on their keyboards.

After some experimentation, though, we felt that the MIDI music was good, but didn't capture the essence of the John Williams orchestral soundtrack that is so closely associated with Star Wars. Furthermore, each additional instrument channel would require more CPU time than we wanted to allocate.

At this point, we tried an experiment using uncompressed digital samples of the Star Wars main theme. The quality was extremely good, even after subsequent compression with the ADPCM encoder provided by Nintendo. After a little persuasion, Nintendo generously agreed to increase the amount of cartridge space from 8MB to 12MB. This allowed us to include approximately 15 minutes of 16-bit, 11khz, mono music that sounded surprisingly good. Considering that most users would listen to the music through their televisions (rather than a sophisticated audio system), the results were close to that of an audio CD, thereby justifying the extra cartridge space required.

## Art Path

A continuing problem throughout the development of SHADOWS was the inability to import and export data between the various 3D packages we were using. Eventually, we managed to circumvent these problems with a number of translation utilities as well as by using Alias Power Animator as our central "hub" format. However, there were still issues with scale, model hierarchies, and animation data. It was sometimes difficult for the artists to see what their artwork really looked like until it had been through the hands of our polygon wrangler (thanks Tom!). Initially, it was difficult for our texture artist to visualize the restrictions on texture size required by the hardware, as well as color reduction issues.

## New Hardware

There were a number of other issues that we had to deal with in developing the game, not the least of which was that for the first nine months of the project, we didn't have any real hardware on which to run the game. This deficiency wasn't insurmountable by any means, but it restricted our choices in certain ways, especially in level design. We were forced to make some assumptions, especially regarding to performance. Fortunately, this wasn't quite the bugbear that we anticipated. Still, as is well known, those on the bleeding edge of technology are often sacrificed upon it.

## Other Issues

There was considerable pressure to finish the game in time for the Christmas 1996 deadline. This reality meant many, many late nights, with some team members regularly working over 100 hours every week for the best part of a year. Hopefully, this sort of workload can be avoided in future projects. Time pressure is, of course, a common thing in the computer games industry — and we were certainly no strangers to the phenomenon. However, since we had to release our game shortly after launch of the machine, we were under more pressure than might usually have been encountered. Game testing also became an issue because there were very few machines with which to actually test the game.

## Game Play Variety

We were able to include a very wide variety of game play styles in SHADOWS. In retrospect, this meant that we couldn't tune each type of game play as much as we would have liked. It also meant an almost Herculean programming task in trying to write and debug what amounted to five different game engines. These consisted of low flight over terrain, gunnery action in space, first/third person on foot or with jet pack (including a moving train sequence), high-speed chases on a speeder bike, and full 360-degree space flight. Nonetheless, the result was that most players' experiences with the game were always interesting, at the expense of displeasing some of the more hard-core game players. A variety of game play was important for a game that, for many players, would be one of their first experiences in a fully 3D environment.

## Hardware Performance

As mentioned before, for the first nine months of SHADOWS, we had no real hardware with which to gauge the performance of the game — other than a rather nice Silicon Graphics

## The Core Team

**The core team developing SHADOWS from inception to completion consisted of mainly six people, although twenty people contributed to the game for varying lengths of time, and to varying degrees. Nonetheless, everyone played a vital role in the production of the game.**

Game Designer/Lead Artist - Jon Knoles
Project Leader/Senior Programmer - Mark Haigh-Hutchinson
Technical Lead - Eric Johnston
Programmer/Lycanthrope - Mark Blattel
Polygon Wrangler - Tom Harper
Level Designer - Jim Current
Level Designers - Matthew Tateishi and Ingar Shu
3D Artists - Paul Zinnes, Andrew Holdun, and Garry M. Gaber
3D Animator - Eric Ingerson
Texture Artist - Chris Hockabout
3D/Background Artist - Bill Stoneham
Storyboard Artist - Paul Topolos
Music Editor - Peter McConnell
Sound Designers - Larry the O and Clint Bajakian
Lead Tester - Darren Johnson.
Production Manager - Brett Tosti

Extra thanks go to Don James, Henry Sterchi, Hiro Yamada, Kensuke Tanabe, and Shigeru Miyamoto. Special thanks as always go to the staff at LucasArts, and particularly to George Lucas for his gift of the Star Wars universe.

*Back Row (left to right): Steve Dauterman, Peter McConnell, Jon Knoles, Andrew Holdun, Paul Topolos, Mr. B. Fett; Middle Row (left to right): Jim Current, Matthew Tateishi, Bill Stoneham, Brett Tosti, Ingar Shu, Tom Harper, Chris Hockabout; Front Row (left to right): Garry Gaber, Mark Blattel, Eric Johnston, Mark Haigh-Hutchinson; Not shown: Paul Zinnes, Larry the O, Clint Bajakian, Eric Ingerson, and Darren Johnson.*

ONYX. Nonetheless, when we finally received the real hardware, we were pleased to find that the performance estimates given to us by SGI proved to be very accurate. In fact, in large part due to the parallel nature of the graphics hardware, we were able to use floating-point mathematics throughout SHADOWS with no significant impact upon performance.

Additionally, SHADOWS was programmed entirely using the C language — it wasn't necessary for us to use assembler (a first as far as I was concerned, and a pleasant surprise even though I'm a long-time hardcore assembler fan). Since our scene complexity was relatively high (usually kept to around 3,000 polygons or so, but variable according to the level type and design), the graphics task took longer to execute than the program code (that is, we were graphics-bound). Consequently, optimizations to the program code didn't significantly improve overall performance.

## NTSC to PAL Conversion

After completing the American and Japanese versions of the game, it was my task to convert the game so that it could run on the European PAL television standard. Being British, I had a vested interest in making sure that the conversion was a good one. This meant two things: first, that the game used the whole of the vertical resolution of the PAL display (625 lines vs. 525 lines of NTSC); second, I wanted to ensure that the speed of the PAL game was the same as the NTSC one, even though the PAL refresh rate is 50hz rather than 60hz.

Fortunately, when we started work on SHADOWS, we realized that one of the most important things to consider was that it had to be a time-based game, rather than a frame-based one. This would allow for update rates that could vary considerably depending upon scene complexity, as well as the simple fact that we didn't have any real hardware from which to measure performance characteristics. Essentially, the program keeps track of the absolute time between each update of the game. This value, which we called delta time, became a multiplicand for any movement or other time-based quantity. By this method, the game runs independent of the video refresh rate, with all objects moving and responding at the correct frequency.

The other issue had to do with the "letterbox" effect that is common to many NTSC to PAL conversions. In most cases, there is no extra rendering or increase in the vertical frame buffer size, leaving unsightly black bands above and below the visible game area. Since the vertical resolution is now greater than the original NTSC display, the aspect ratio will also change, causing the graphics to appear stretched horizontally.

While I wasn't willing to accept this, I had presumed that I couldn't afford the extra CPU time necessary to render a larger frame buffer, even with the extra time available due to the 50hz video refresh rate. There was also a question of the additional RAM usage required by our triple buffering of the frame buffer. My first attempt, therefore, was simply to change both the field of view and aspect ratios of the 3D engine. This simple fix solved the "stretching" problem quite nicely, although the display remained letter-boxed, of course. Unfortunately, it also meant that any 2D-overlay status information remained "stretched." There was the potential that game play could be affected because the field of view, by definition, would affect the player's perception of the 3D world.

Again, this just wasn't good enough. What I needed was a solution that didn't require extra rendering, yet would fix the aspect ratio problems. After a little bit of research, I realized that I had discovered earlier that it was possible to change the size of the final visible display area on the output stage of the display hardware. In reality, it's possible to shrink or enlarge the display both horizontally and vertically. To compensate for the letterboxing, all I had to do was change the vertical display size by a factor of 625/525 or 1.19. Once I did this, I immediately had a full-screen PAL version. Or so I thought….

One of things about SHADOWS is that

we had to compress everything in the game to fit it into the cartridge space available. This included the thin operating system that SGI provides as part of the development system. Therefore, upon machine reset, it's necessary to decompress this OS to run the game. To perform this decompression, we wrote a small bootstrap program, which introduced a small amount of time between the hardware being initialized and the OS starting. This lag introduced a one-time glitch on the screen as the video hardware started. Not very noticeable, except to me. After many late nights, I discovered a way to remove the glitch by directly accessing the Nintendo 64 video hardware registers.

## Bad Idea

We then discovered that because we had accessed the hardware directly, it caused an infrequent bug. Rarely (1 out of 50 times) the Nintendo 64 would crash if the reset button were pressed at a particular point in the game. Not only that, I couldn't repeat the bug on my hardware (I hate it when that happens).

After a number of very late nights (over the Christmas holiday), with the help of Nintendo of America's technical staff (thanks Mark and Jim), we finally resolved the problem: first, by removing the code that directly accessed the video registers, and second, by restoring the registers controlling the scaling of the output in the vertical axis upon reset. Sometimes, the simplest solution is the best.

## Support from SGI and Nintendo

We were very lucky to receive excellent support from both SGI and Nintendo during the production of the game. The SGI engineers (thanks in particular to "Acorn") were very helpful and would normally have an answer to our questions within a day, sometimes within the hour. I would like to thank Nintendo for their assistance in the production of the game. Nintendo of America's technical support and QA departments also proved invaluable. In addition, three of Nintendo of Japan's staff spent some time working directly with us at our offices.

I was also fortunate enough to visit Nintendo's head quarters in Kyoto, Japan, to discuss SHADOWS with Shigeru Miyamoto, creator of MARIO 64. His insights were both fascinating and extremely relevant. He is simply a genius with an instinctive understanding of video games.

## Of Wampas and Men

When developing a project on the scale of SHADOWS, there will always be some things that didn't progress as smoothly as they could have…

1) The motion capture process proved to be a red herring for us. While originally promising a much more realistic animation solution, in our case the data proved unusable. However, I still believe that it has great potential and deserves further investigation, even though we didn't get to the point of dealing with the potential problems matching the motions to the character's environment and so forth. Caveat emptor.
2) Attempting to use a MIDI-based music solution also proved incorrect for this game. While it promised to be an efficient solution in terms of memory (an important consideration for a cartridge-based game), it simply wasn't suitable for an orchestral soundtrack such as Star Wars.
3) When we started work on SHADOWS, a major problem (that continued throughout the duration of the project) was the inability of various 3D packages to import and export data. Although we were able, for the most part, to write our own conversion utilities, it still proved to be a stumbling block and prevented us from having an efficient art path. Fortunately, the companies supplying these tools now recognize the need for importing and exporting data to other packages, and are taking steps to remedy the situation — VRML, for example, is proving to be a useful format.
4) Time was the biggest enemy of all in producing the game. This is nothing new, but was exacerbated by the fact that we were working on a non-existent machine for nine months. Nonetheless, even though this was, for the most part, out of our control, we were still able to produce a quality game.
5) With hindsight, probably the most important lesson to be learned from the game's development is that of focus. Do one or two things and do them extremely well. Although our ambitions were well placed in trying to provide the player with as much variety as possible, we effectively had to write five different game engines. Additionally, we could have also used a fourth programmer dedicated to all aspects of the front-end of the game; that is, level selection, controller options, and so forth. This would have taken some of the pressure away from the main programmers towards the end of the project.

## Out of the Shadows...

Thanks to the talent, dedication, and experience of the SHADOWS team, many things went well during the development process.

1) By using the powerful SGI computers (fairly uncommon in the games industry in 1994) to prototype, combined with our programmers' knowledge of 3D technology, we were able to develop the game rapidly, yet remain flexible in terms of performance requirements.
2) Our ability to reuse tools from our earlier DARK FORCES title saved us time and resources because we didn't have to build all new tools, although a large number of data conversion utilities were necessary. In addition, by reusing familiar tools, our level designers could be more productive earlier in the project than otherwise might have been expected.
3) Our decision to use digitized music proved to be a crucial one. Because most users would listen to the music through their televisions, the quality approximated that of an audio CD as far as many customers were concerned. This alone justified the extra cartridge space required and surprised many players who didn't expect that level of quality from a cartridge game.
4) The conversion of the game for the PAL television standard went extremely well and was much appreciated by customers in those countries. It would be fair to say that SHADOWS has set the standard in that it runs both full screen and full speed. There is no reason why all games from this point on shouldn't run just as well on PAL systems as they do on NTSC.
5) Given that we were working on completely new hardware and for the most part had to discover everything that we needed to know by ourselves, the support from both SGI and Nintendo was invaluable to us throughout the project.

## Varying Shadows

Even though we were not able to spend as much time as we would have liked tuning the game, SHADOWS does succeed in supplying the player with a variety of game-play styles. Its popularity is a testament to the creativity and talent of the team of which I was fortunate enough to be a part. ■