GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Outrage's Descent 3

By Craig Derrick,Jason Leighton

Descent 3's creation was a long and arduous task that was both a joy and a pain. Exciting technology coupled with inconsistent design and almost nonexistent management had all members of the team exhausted by the end of our 31-month development cycle. When the game finally did ship however, we knew we had a winner.

Developed by Parallax Software and published by Interplay Productions, *Descent* was released in 1995 and took the world by storm with the first first-person shooter offering 360 degrees of movement. No longer were players constrained just to walking around a 2D world — now they had complete freedom of movement in a true 3D space.

Players were able to fly their Pyro ship in any disorienting direction — up, down, and everywhere — and top became bottom, bottom became up, as players plummeted down never-ending tunnels blasting mechanical robots and saving imprisoned miners.

This innovation in action gaming was immediately successful and garnered The Academy of Interactive Arts and Sciences Best Title of 1995 and Best Computer Game of 1995. And PC Gamer voted it Best Action Game in the World. One year later, the sequel *Descent II* was released, which added another 30 levels to the mix and improved the game's AI, thanks to the addition of the Guide-Bot and Thief-Bot. The game ended with a cliffhanger cinematic that almost certainly guaranteed another sequel. That's where the *Descent 3* project began.

Who the Hell Is Outrage?

Descent was developed by a small group of programmers and artists at Parallax Software in Champaign, Ill., headed by Mike Kulas and Matt Toschlog. After the successful completion of *Descent*, Toschlog moved to Ann Arbor, Mich., and established a second office for Parallax Software. He took three designers with him and hired two additional programmers. Both offices then began to work simultaneously on *Descent II*. While *Descent II* was deemed a successful project, the process of trying to get teams located in two distant offices to work effectively together took a heavy toll on both teams. It was at that time that Matt and Mike decided that each office should work on separate titles and eventually become separate companies. Thus, Outrage Entertainment and Volition Inc. were born.

It was another eight months before the production of *Descent 3* began. After the release of *Descent II*, Outrage immediately began work on *The Infinite Abyss* (a Windows 95 version of *Descent II*, along with a new level add-on pack entitled The Vertigo Series). Meanwhile, Volition concentrated on development of *FreeSpace*. It wasn't until after the release of *The Infinite Abyss* and *Descent* Maximum (the Playstation version of *Descent II*) that the developers here at Outrage began focusing all our energy on the design and development of *Descent 3*.

Another Sequel And Why *Descent 3*?

Descent I and *II* had the makings of a franchise for Interplay, and with any franchise, successful or otherwise, sequels are sure to follow. Technology had taken a big leap in the year and a half that *Descent* had come out: notably Windows 95 and hardware-accelerated 3D. It was easy to see how *Descent 3* could be dramatically improved over its predecessors.

By the fall of 1996, we began to compile a list of features that we would like to see in *Descent 3*. By November we had created a design document detailing the new features that would be implemented for *Descent 3* and submitted it to Interplay for approval.

The initial design and programming work on *Descent 3* began in December 1996. Some of the team had just completed work on *Descent II – The Infinite Abyss* and *Descent Maximum* for Playstation, while others were involved with research and development for *Descent 3*, where they learned about new tools and technologies. We were excited about taking the *Descent* franchise to the next level and eager to begin, but little did we know that our over-eagerness would impair the game's development.

Initial conceptual drawing of the Phoenix Interceptor.

About six months after starting development, we stepped back and took a long hard look at what we had and where we were going. Originally, it was deemed that *Descent 3* would have both a software and hardware renderer. After checking out the competition, it was apparent that, if we wanted to be visually stunning (and maintain interactive frame rates), we would either have to scale back our technology design or go with a hardware renderer only. We chose the latter. In retrospect this was a good decision, but it was unfortunate that we had to make it six months into the development, since many of the tools and software rendering technology were already developed.

At this point, not only did we decide to go with a hardware-only renderer, we decided to scrap the engine that we had been developing. The engine we had going was an enhanced segment engine — a portal engine that used six-sided deformed cubes to represent geometry. Essentially, it was the same technology used in *Descent II*, with some additional features thrown in for improved geometry modeling. If we had stuck with this engine until the game shipped, we would have been way behind the technology curve with respect to our competition. Instead, we went with a "room"-based engine, which allows designers to create just about any geometrical area within a 3D modeling program, such as 3D Studio Max or Lightwave.

Shortly thereafter, the terrain engine was developed. We were seriously considering the idea of creating outdoor areas for *Descent 3*, but we worried about the high polygon count associated with such a large rendering distance. Fortunately, we used a good level-of-detail (LOD) algorithm to combat the frame-rate problems. Unfortunately, the decision to include terrain would adversely affect the overall design in ways that we couldn't possibly have foreseen, such as the scale of the terrain dictating how fast the ship appeared to move while outside.

For the next 18 months, work continued on *Descent 3* at a frantic pace. We were learning how to use our custom in-house tool, D3Edit, so in the process we ended up creating and then throwing out an incredible amount of our content — what looked cool one month looked dated the next. This was largely due to the fact that we were developing a cutting-edge engine at the same time we were trying to design the game itself — a pitfall many developers have fallen victim to. Unfortunately, throwing out so much work also cost our team a lot in terms of our morale. What we should have done is freeze the design of the engine about a year before the product shipped and then worked on the game. Unfortunately, in our lust for sexy technology, we just couldn't do that.

When we finally did ship, we were exhausted in a variety of ways. Working on the same game for two and a half years is emotionally depleting, to say the least. Although we knew the game was cool, we didn't know how the public (or reviewers) would receive it. Thankfully, it turned out that our fears were unfounded — *Descent 3* has received incredibly good scores from a variety of sources, including print magazines and gaming web sites.

What Went Right

1. **Working on a sequel.**

   Working on any sequel, whether a game or movie, has its ups and downs, and *Descent 3* was no exception. Much of the joy of working on a sequel comes from the fact that you can improve on an already established title, and in many cases, add features that were previously impossible to do.

   Knowing your target market is essential to making a successful sequel. You must not deliver a product that completely turns away your core audience, and yet enough of it must be new in order to persuade them to purchase it. The four long years between *Descent II* and *Descent 3* convinced us that new technology alone warranted a new product, but we didn't want just to copy the previous version. *Descent 3* had to retain all of the elements that made its predecessors big hits and yet, creatively, be different. And therein lies the paradox of making a sequel.

The new and improved thief.

   Starting with our old games' design document and features list, we cataloged elements that the team liked and disliked about the previous games. Every item was scrutinized and broken down into specific lists, such as art, weapons, sounds, user interface, and so on, to determine what part of the feature needed to be changed and what needed to be left alone. Developers sometimes do this with their competition's products when beginning development of a similar game, but nothing beats being able to do it with your own game because you have an intimacy with the product that outsiders are not privy to.

   Using a hardware-accelerated 3D engine.

   When development of *Descent 3* began in November 1996, hardware accelerators (specifically 3dfx's Voodoo 1) had just come out. Our initial design document called for *Descent 3* to ship with a hardware and software renderer, but as our aspirations for the graphics engine grew, so did the need for hardware acceleration. Complex geometric rooms, robot enemies having nearly twice the amount of polygons and animation states compared to our previous games, complex outdoor terrain, and the whiz-bang effects expected in today's games all necessitated hardware acceleration. With all these features in the game and running at abysmal frame rates with our software renderer, it was decided, reluctantly, that we would release as a hardware-only game.

   As development wore on, technology advanced and accelerators became faster, cheaper, and more popular with each passing year. It seemed that games were coming out every week that sported a hardware accelerator mode or patch, but up to this point, nothing had come out that was hardware-only. We knew just by looking at our progress on the game under acceleration that we had a beautiful looking game with all the latest technologies — but would anyone actually be able to play it?

   Our Christmas deadline came and went, but in retrospect I feel it was for the better. If you didn't have a hardware accelerator before Christmas 1998, chances are you did have it later. With the implementation of Direct3D, OpenGL, and Glide, *Descent 3* was capable of running on just about every video card available when it was released. Because we took a chance on technology, believed in our product, and slipped a bit, *Descent 3* looks, feels, and plays like a next-generation *Descent*. And that's all we really wanted.

3. **Out with the old engine, in with the new.**

   As mentioned earlier, the initial plans for *Descent 3*'s graphics engine were to include both a software and hardware renderer. The engine itself was to be a heavily modified version of the segment (cube-based) engine used for *Descent I* and *II*, meaning that all geometry had to be sculpted by connecting one deformable cube to another. While this engine supported some interesting geometry, it just couldn't handle the complexity we had in mind for *Descent 3*'s levels.

   So, six months into development we started over on the engine, and this time we aimed higher. We set our sights on creating what is now the Fusion Engine. This engine would actually be two separate engines — one for internal settings and one for outdoor terrains — that would work together seamlessly. The internal "room" engine allows designers to model almost any type of complex geometry in a program such as 3D Studio Max and import it directly into our game editor. Once imported, designers can modify the geometry, texture it, place objects, and light it. Designers could then take the individual rooms and join them together, creating a portalized world for the player to fly through.

   The terrain engine actually began as a prototype for another game that Jason was interested in developing. Unfortunately, Bungie's *Myth* beat us to the idea, but the terrain technology was solid enough to be incorporated into *Descent 3*. It was based on a great paper by Peter Lindstrom and colleagues entitled Real-Time, Continuous Level of Detail Rendering of Height Fields (from Siggraph 1996 Computer Graphics Proceedings, Addison Wesley, 1996). Of course, it was bastardized heavily to fit the needs of *Descent 3*, but the overall concept was the same — create more polygonal detail as you get closer to the ground and take away polygons when you are farther away. After implementing the real-time LOD technology, our frame rates quadrupled.

   The outdoor engine gave designers the ability to create an internal structure and its outside shell (an external room with the normals flipped), and place it anywhere on the terrain. This let us create seamless transitions between a structure within the level to an outdoor section, with absolutely no load times whatsoever. For the first time in *Descent*, players could actually leave the mine. When players cross the portal that leads from inside to outside, the game code would switch collision detection, rendering, and so on, to use the terrain engine.

4. **Incredible technology.**

   One of our biggest goals in developing *Descent 3* was to bring the game engine up to date. This included graphics, AI, sound, and multiplayer. I think we hit the mark. *Descent 3* includes just about every whiz-bang graphical feature there is, the AI is very smart for an action game, and the multiplayer plays pretty well even over lagged connections. Even though we're not in the first-person-shooter genre — a genre that is judged by its graphics and networking technology (some say at the cost of game play) — we compete favorably with the offerings in that arena.

5. **Great multiplayer.**

   We knew that *Descent 3* had to have the best multiplayer right out of the box, or people would be disappointed and scream for our heads. Sadly, in this age of release-once, patch-many, there are a lot of games that come out that haven't fully tested their multiplayer aspects, and these systems are full of bugs. Fortunately, *Descent 3* did not suffer from this. We spent a lot of time testing *Descent 3* networked games over a variety of conditions, both lagged and unlagged. It was a tiresome process, but in the end, I'm happy we did it.

Another thing we did that showcased our attention to multiplayer was to give a whole slew of options to the player. We had support for IPX, TCP, DirectPlay Modem, and DirectPlay Serial. These options allowed players to connect to games using the protocol best suited for their situation, instead of just offering TCP as a lot of other games do. There were also three network architecture types: D3 client/server, peer-to-peer, and permissible client/server. We did this because we knew we had to support the *Descent I* I fans (peer-to-peer), the *Quake* and *Unreal* fans (permissible client/server), and try to forge our own path with a new network technology (D3 client/server). We guessed that permissible client/server would be the most popular model, simply because that was what players were used to. To our surprise, it turned out that D3 client/server was the most frequently used architecture. This architecture changed the way lag was perceived. In games such as *Quake* and *Unreal*, there is a noticeable delay between firing the weapon and when the weapon appears on your screen. The reason is that the client asks the server to fire and the server gives the client permission to fire (hence permissible client/server). The D3 client/server is different in that it allows you to fire right away — when you press the trigger, the laser appears immediately. The downside is that you have to lead your opponent by your ping to the server, and sometimes when the laser would hit your opponent on your screen, it wouldn't really hit them on the server. We found this to be less frustrating that the permissible client/server way (which personally drove me crazy), and thankfully our fans agreed.

While providing players with so many options might have cost us development time, I'm confident that we made up for it in the satisfaction that our customers had by being able to customize the game to their liking.

Overall, the development team at Outrage was very energetic to work on *Descent 3* and their dedication paid off more than once during development. Working on a game as long as we did is always tough going. Because we, too, are game players, it was sometimes tough to be working on something for so long, never quite knowing whether or not the work you were doing would be

accepted by your peers within the industry, or more importantly, by consumers and fans of the product. This all changed for us during 1998's E3 convention.

We didn't get confirmation from Interplay that we would be showing the game at E3 until about a month before the show. When the word finally did come, we shifted gears from our production and went full steam towards making the best E3 demo that we could. This would be the time that the game and we wanted to come out a winner. Fortunately, everything turned out all right. The fans who mpressed and the comments from the press were overwhelmingly positive.

gh was the amount of pride that each person had in their own particular domain. The level designers evels ever, the programmers wanted their particular systems to stand out, and the artists wanted the art style of the game to be unique. So this pushed people to do their very best — the atmosphere was almost competitive. There were a couple of times when things got out of hand and egos had to be held in check, but for the most part, the individual team members were allowed to shine without stepping on the toes of a fellow colleague.

What Went Wrong

1. **Lack of direction or vision.**

   The biggest problem on *Descent 3* was the weak management structure. *Descent* and *Descent II* were developed by small groups that worked closely together (often in the same room), and as we grew to a team of almost twenty people, we didn't introduce enough management to control the process. Even though people had designated titles, there was no real authority attached to those titles. No code reviews, no art reviews, no way of saying, "This is bad and we should be going in a different direction." What we needed was more of a hierarchy and a systematic way to get things done. People would have disagreements that were never settled, and that led to bad feelings among team members. It would have been much better if some sort of hierarchy had been in place to mediate disputes, but unfortunately this didn't materialize during the entire development cycle of the game. For example, if a couple of people had a problem with the way a particular level played, there was no recourse to get that level changed. Bringing up the faults in an individual's level, system, or art would start arguments, and that got us nowhere. Next time we'll know better. An anarchistic development environment is great in theory (total freedom), but you really do need a hierarchy of some sort, a chain of command where people know whom to go to with problems. It was a painful lesson that could have been avoided if we had set up our team to be more like a company and less like a garage band.

2. **No standardized level design tools.**

   Another major problem on *Descent 3* was the tools the level designers used to make their levels. Some people used 3D Studio Max, some used Lightwave, and one designer even wrote his own custom modeler from scratch. Due to our somewhat anarchistic development environment, this was seen as O.K., and not too many people complained. However, the different tools led to inconsistent quality across our game levels. One designer would make great geometry that had bad texturing, while another designer would create the opposite, especially if his tool of choice made texturing or geometric modeling difficult. What we should have done is standardized using one tool (probably 3D Studio Max) and made the designers learn how to use it whether or not they were comfortable with it. This would have allowed us to make use of certain features that 3D Studio Max has (such as parametric surfaces) without worrying about whether Lightwave or the custom modeler supported that feature.

Watch out for the dual Fusion attack of the Thresher.

After assembling their geometry, the designers took their rooms and models, imported them into our custom editor (D3Edit), and tried to glue everything together. Unfortunately, our editor was written by programmers who didn't give enough thought to the user interface — they often designed interfaces that were intuitive to a programmer, but not to a designer. Conversely, a designer would ask for a feature that might take a programmer a long time to code, but then the designer wouldn't use the feature very much. This led to feelings that the designers didn't know what they wanted, and the programmers didn't care enough to make things easier for the designers. It was a vicious circle that didn't get cleared up until the latter third of the project. Even in the shipped game you can tell

which levels were made early on and which were made near the end of the production cycle. The later levels are much better looking, have better frame rates, and generally have better scripts.

3. **Programmers working on dedicated systems.**

   Many systems, such as the graphics, AI, and scripting, were written exclusively by one person. This fostered a feeling of pride in a particular system, but it also caught us with our guard down when we found that one system was falling behind schedule. Our AI, for example, was very much behind schedule for the duration of the project because the programmer just had too much to do. This caused a ripple effect that was felt throughout the design of the game. After all, you can't tell how a level is going to play if the robots that you're fighting aren't behaving at all as they should. We couldn't simply add another programmer to that particular system to help out, because by the time the new programmer became familiar with the new system, then his system would fall behind. It might sound as if this was an understaffing problem, but it wasn't. It was more of a case of "design as you go," where someone would suggest a great feature and then the programmers would implement it. Unfortunately, all these excellent features started adding up and taking a tremendous amount of time to implement. It would have been better if there were more programmers on a particular system, or at least ones who were familiar with it, so that if there were concerns with slippage, other people could have been brought aboard to help.

4. **Working on a sequel.**

   With sequels come high expectations that can almost never be achieved completely. (Just look at *Star Wars: The Phantom Menace* to see what I'm talking about.) While we aimed high for the entire game, many times trying to meet the varying expectations of our team (more in-level scripting and rendered cinematics), publisher (using more AI and levels), and fans (more of everything) resulted in half-implemented features, such as our in-game cinematics or items that went in untested at the last minute (such as some Guide-Bot functionality).

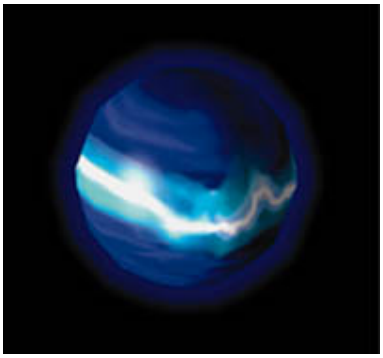
   With a game so big and widely anticipated, I'm not sure if we would have ever been able to manage everyone's expectations. And, one of our biggest problems was that those expectations were controlling the design of the game. As developers, we should have been more confident in our abilities to produce this type of game and should have approached new ideas with a bit of skepticism. While I don't dismiss the valuable role that the fans and our publisher played in the development of *Descent 3*, we were too apt to deviate from our design document when others wanted us to add features to the game.

   It's very important to stay true to your design document and know which systems are valuable. Every new feature should be evaluated not only on its value to the game, but also from a production standpoint. When we decided to create in-game cinematics, it was simply to introduce the boss robots. The effect was so impressive that we decided to use the system to establish the player's location in the beginning and ending of each level. This worked out so well (do you see where this is going?) that we used the system to help establish when puzzle elements were completed. This one-time only system was utilized in ways that hadn't been thought of before its implementation, which caused numerous problems.

5. **Fast company growth and green employees.**

   When we started work on *Descent 3*, Outrage had just eight employees on staff. Since some of the *Descent II* team left Ann Arbor to work at Volition during the project, we literally had to build the team and company at the same time we started production on the game. This resulted in a mixed bag of results ranging from pushing back larger projects or features until later in the project to adding multiple tasks to someone's already full schedule.

One of the results of being understaffed was the decision to contract out most of our 3D animated door modeling to an outside company, Vector Graphics, for completion. Our schedule showed that one animated door took anywhere between three to five person-days to complete. With a design document that dictated more than 30 doors in the game, we would have run out of time. We made the decision to round up all our sketches for doors and hand them over to the very capable hands of Vector Graphics. This allowed our designers to concentrate on their scheduled tasks and then add the doors later as we received them.



The *Descent* orb, now realized in full 3D.

What we didn't account for were the problems that came up because our teams worked in different locations. (Somehow we forgot that our company was split for this very reason.) Assembling our development environment and building an editor for Vector Graphics was troublesome, and once we gave the editor to them, they really didn't know how to use it. Our lead designer worked with them almost daily to bring them up to speed and fix any file incompatibilities caused by our constantly evolving editor. This caused our lead designer to fall behind in his schedule, and we had to shuffle around due dates for his specs and level deliverables. As we hired more people, he caught up.

In the end, we hired an additional nine employees, only two of whom actually had any professional game development experience. And while everyone on the team gave the game their full attention, we definitely had issues come up that were the result of inexperience.

A loss of professionalism, maturity, and a standard of conduct was apparent during the development of *Descent 3*. Some of this was due to young employees who came to us directly from school with no professional work experience, while others had trouble dealing with the long hours that come with the job. This, coupled with the lack of a strong management hierarchy, resulted in clashes over direction, seniority, and leadership on the project. More times than not, a person would not follow the direction of the lead simply because of their personal issues with that person. This resulted in the lack of respect for that person and the very responsibilities that came from being the lead.

But our biggest problem was definitely with art direction. Without a dedicated art director on staff, we often second-guessed everyone else's work. Each artist and designer had specialized tasks, and without an art director to make a final decision, art was simply added to the game without any formal approval. In the beginning, we had show-and-tell meetings to show off the latest work from people, but this almost always turned into a forum for criticisms and led to animosity between team members. An art director leading us would have kept our artwork consistent and would have been the final authority on all artwork-related matters.

end game *Descent 3* was an incredibly challenging project, to say the least. The design-as-you-go and design-by-committee aspects were a large part of the problem, and any future Outrage projects will be more diligent in their initial design. We'll make sure that the designers have absolutely the best tools at their disposal and we'll have better management to mediate design disputes. Without these problems during the development of *Descent 3*, I'm fairly confident the whole project would have been a little less painful.

Jason Leighton likes to kill time by complaining about the horrid Michigan weather cycles. When it is actually nice outside, he complains anyway. To hear his weather woes, or if you just want to talk about game programming, write to jason@outrage.com. Your e-mail is important to him, but may be monitored for quality assurance purposes.

When Craig Derrick isn't off buying DVDs, he's often writing to Jason Leighton about his weather woes and trying to convince him that all he really needs to do is to stop programming and go outside. If you need a motivational speaker or if you have a problem and no one else can help, write to craig@outrage.com.

Descent 3
Outrage Entertainment
Ann Arbor, Mich.
(734) 663-9120

Release date: June 1999

Intended platform: Windows 95/NT/98

Project budget: $2 million

Project length: 31 months

Team size: 19

Critical development hardware: Intel Pentiums and AMD K-6 II processors. Each machine was equipped with a hardware accelerator and at least 64MB of RAM.

Critical development software: Photoshop, 3D Studio Max, Lightwave, Microsoft Visual C++ 4/5/6; rendering APIs used include Sourcesafe, OpenGL, Direct 3D, and Glide; Direct Sound, Aureal, and EAX were used for the game audio.

Return to the full version of this article