GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Bungie's *Myth: The Fallen Lords*

By Jason Regier

As the team at Bungie Software put the finishing touches on the Marathon series of first-person action games, our thoughts drifted to bringing our 3D game experience to the real-time strategy game (RTSG) genre. We were inspired by movies such as *Braveheart*, with its close-up portrayal of bloody melees between large forces, and books such as Glen Cook's *The Black Company*, in which gruesome tales of battle contrast with engaging and intriguing characters. We envisioned a dark, amoral world where opposing sides are equally brutal and their unity is torn by power struggles within the ranks. We dreamed of game play that combined the realism and excitement of action games with the cunning and planning required by strategy games.

Our original design document, if you could call it that, was simply opposing lists of "Stuff that Rocks" and "Stuff that Sucks." Anything vaguely cliché, such as excessive references to Tolkien novels, Arthurian legend, or "little boys coming of age and saving the world," went in the "Sucks" category. The "Stuff that Rocks" list was filled with ideas that contributed to the visual realism of the game: a true 3D landscape, polygonal buildings, reflecting water, particle-based weather, "blood-spattered battlefields littered with limbs," explosions that send shock waves through the terrain, and "lightning frying guys and their friends."

Our goals for the product were lofty: simultaneous release on Windows 95 and Macintosh platforms, integrated Internet play, and a free online service to allow players from across the globe to battle one another. From this vision, *Myth: The Fallen Lords* was born.

The Making of a Legend, er, Myth

The project began in January 1996 with four programmers, two artists, and a product manager; midway through development, one programmer dropped out and an artist was added. Music, sound effects, and cut scenes were done out-of-house, and a few artists were contracted to help with interface artwork.

The roots of the *Myth* programming team were on the Macintosh, so most initial coding was done on the Mac with Metrowerks CodeWarrior. When PC builds were required, though, we used Microsoft Visual C/C++. Myth was written entirely in C.

In addition to creating the shipping product, we developed four tools to aid in the construction of the game. One utility, the Extractor, handled the importing of sprites and the sequencing of their animations. Another tool, dubbed Fear, dealt with importing polygonal models such as houses, pillars, and walls. The Tag Editor was responsible for editing the constants stored in cross-platform data files, which we called tags. And finally, Loathing, our map editor, handled the rest. Loathing was built around the *Myth* engine and allowed us to modify the landscape, apply lighting, set terrain types, script the AI, and place structures, scenery, and monsters.

The artists used Alias|Wavefront's PowerAnimator and StudioPaint on a single Silicon Graphics Indigo 2 to create polygonal models and render all the characters. At one point, the artists worked separate day and night shifts so that they could maximize their time on the SGI. Models were brought into the game using Fear, while the sprites were cleaned up in Adobe Photoshop and imported with the Extractor. To create the texture maps for the terrain, the artists used Photoshop to draw what looked like an aerial photo and applied it to a 3D landscape in Loathing.

If this sounds like a lot of work to you, you're right. Most maps took at least a week or two to create. We considered using fractal-generated landscapes, but we were worried that the inherent randomness of such terrain would make it extremely difficult to design good levels. As a result, all maps were painstakingly constructed by hand. As the artists put the finishing touches on the landscapes, the programmers, who doubled as level designers, scripted the AI for the levels.

Myth took approximately two years from start to finish. It began as a six-degree of freedom engine that allowed you to fly around a landscape. Soon, troops were added, heads started flying, blood was made to destructively alter the terrain's

color map, and the network game was born. Most of the first year was spent developing the initial network/multiplayer game play. Almost the entire second year was spent developing the single-player game, refining the levels, and testing bungie.net, our free online service.

Myth Character Design

What Went Right

1. Bringing carnage to the masses.
It's a real trick to create a simultaneous, identical-look-and-feel, cross-platform release. It's even harder to do so within the expected time frame with only three programmers. Our experience porting *Marathon*, our popular Macintosh-only action game, to Windows 95 was a valuable learning experience, and we vowed when starting *Myth* that, "This time, we're going to do it right."

Doing it "right" meant designing *Myth* from the ground up to be cross-platform compatible. Ninety percent of the code in the game is platform independent; the other ten percent is split evenly between routines that handle PC- and Macintosh-specific functionality. It was a programmer's dream come true -- we spent almost all our time implementing features and solving real problems, rather than wasting it fighting the OS.

All of the data for *Myth*, from animated cut scenes to the percentage of warriors who are left-handed, is stored in platform-independent files called tags. Tags are automatically byte swapped when necessary and are accessed via a cross-platform file manager.

One of our programmers worked in conjunction with Apple Computer Inc. to develop a cross-platform networking library code-named Über. One of the greatest things about Über is that it supports plug-in modules for network protocols. Thus, although *Myth* currently only allows games over TCP/IP, AppleTalk, and through TEN, it would be trivial to add support for new protocol modules. *Myth* must provide a user interface to set up the connection, but once Über establishes that connection, game play over a LAN is the same as over the Internet.

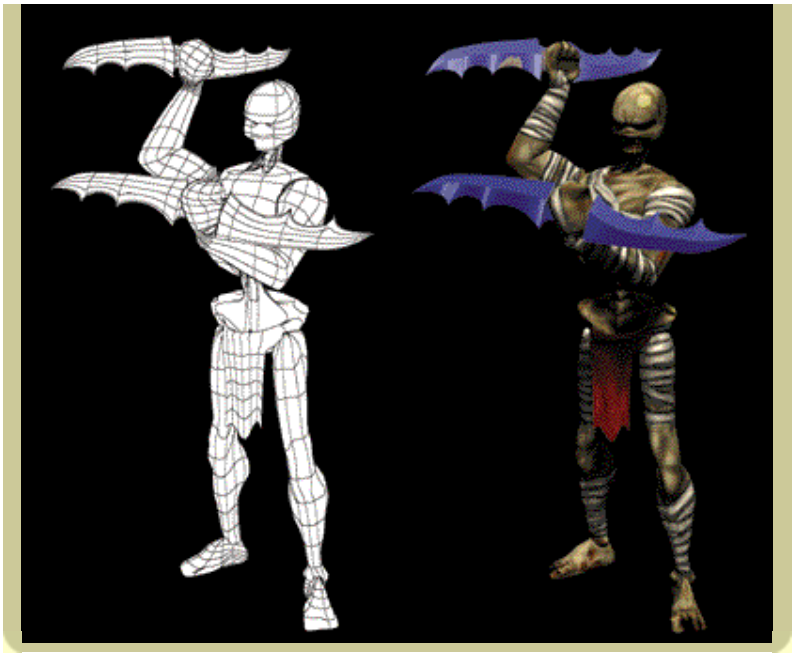
To keep the game's appearance identical across platforms, we implemented our own dialog and font managers. This allowed us (actually, it required us) to use custom graphics for all interface items. We designed our font manager so that it supported antialiased, two-byte fonts, as well as a variety of text-parsing formats. Thus, our overseas publishers Eidos and Pacific Software Publishing were able to localize relatively painlessly. The German version of *Myth* was finished only a couple of weeks after the English release, with Japanese and French versions close behind. The only game experience that is different for the two platforms is the installation, and two players on bungie.net have no idea whether their opponents are on Macintoshes or PCs.

2. bungie.net and beta testing.
Myth was also released with integrated support for our first online service, bungie.net. This service was designed specifically for *Myth* and was developed simultaneously. Similar to online services for other games, it allows players to connect via the Internet to game rooms, where they can chat or play against one another. The Linux-based server that runs bungie.net keeps track of player statistics and gives everyone a score in our ranking system. The service's web site (www.bungie.net) has access to this database and sports a leader board that lists the top players.

Our networking layer is based on a client/server model. Once you advertise a game on the network, you become a server, and other players join your game. Network traffic during a game is limited to the commands issued by the players. All copies of *Myth* in a network game run deterministically and merely interpret the commands that they receive. This makes cheating difficult; if you hack the game to perform something illegal, such as making all your units invincible, you'll go out of sync with other players. When portions of the game data are periodically checksummed and compared, a message will indicate that you're out of sync (and out of luck). So far, the only form of cheating we've encountered is users trying to exploit the bungie.net ranking system.

To rigorously test our server load capacity and the bungie.net code, we released a public beta of the network game. We were initially apprehensive because it was our first public beta test of a product, but it was an amazing success. When errors occur, *Myth* alerts the player, logs the error messages, and usually allows the user to save a replay of the problem. Testers submitted these detailed bug reports via e-mail and chatted about features and improvements to levels on internal newsgroups.

Best of all, the testers used bungie.net to give instant feedback to the developers. This interaction allowed us to gather even more useful information about bugs, and it made the testers really feel involved in the final product. By the end of the beta-testing cycle, we not only had a clean product, but also had a loyal following of users who sang our praises when the NDAs were lifted.



---

3. 3D graphics acceleration.
When the project started, 3D acceleration hardware was only just starting to become popular. Nevertheless, we tried to keep hardware acceleration in mind when designing our rendering pipeline. When the opportunity arose to add hardware acceleration, the implementation worked beautifully. We worked closely with people from 3Dfx and Rendition and added support for their chipsets in about a week. It's amazing how much these accelerators add to the smoothness of the terrain, the fluidity of camera movement, and the realism of the units and effects. These chips rock, and great on-site developer assistance made them easy to support.

4. Getting back to the people.
Once we had released *Myth*, we definitely did the right thing by waiting for player feedback and then releasing a patch to address their issues. Since our public beta test caught most of the bugs in the shipping product, nearly all our post-shipping efforts were directed towards adding user-requested features. We scoured the newsgroups, read e-mail, and talked to customers about their complaints. From these disparate sources, we compiled a list of improvements for our 1.1 patch.

All major user complaints were addressed in the patch. We added support for Rendition and Voodoo Rush cards. We extended the camera's maximum zoom for a better view of the battlefield. We made our easy difficulty levels even easier. And we improved the unit AI. By the time the early reviews came out, we'd already released a beta patch that addressed almost everything on the reviewers' lists of Myth's failings.

5. Doing more with less.
It doesn't take fifty people to create a major cross-platform software title. Period. Bungie Software has barely half that number of employees in the entire company, and we not only develop all our games, but publish and distribute them as well! Macintosh and PC versions of *Myth*, all our internal tools, and our online service were essentially developed by only six people, and everything shipped on time with no major glitches. There's no big quality assurance department here at Bungie; the public did our testing for us, and we listened to them as seriously as if they were coworkers on the project.

We didn't hire any game designers, writers, or level designers to come up with our game concept and story line. *Myth* truly is the combined vision of our team, and each of us feels that it was our game. We came to work each day excited about the project, and we're damn proud of what we managed to create.

What Went Wrong

1. Staffing problems.
On the flip side, it became clear very early in the project that we were understaffed for such an ambitious undertaking. Success or failure rested with a handful of people, and that was extremely stressful. Losing a programmer halfway through development added still more pressure during the final push to get the game out the door. Additional programming tasks had to be shouldered by the remaining developers, who were already also responsible for level design. To alleviate the problem somewhat, we even found it necessary to ask our busy network administrator to aid in AI scripting and level design.

We did hire a third artist near the end of the project, but it was almost too late. While his contributions to the final product were by no means

insignificant, it took a long time to get him up to speed. Similarly, when we dropped the services of our original sound guy late in the development cycle, a new sound team had to rush to redo all the work.

If you're looking for good anecdotes about how we blew off steam with wild weekend trips to Cancún, you won't get any. We all worked incredibly hard, and did so willingly because *Myth* represented a two-year labor of love. All the great previews and supportive feedback from beta testers kept us excited and made us realize that we really did have something special on our hands. Nobody wanted to slack off and allow competing products to beat us to the shelves. The moral of the story: staff up as early as possible and plan to weather the unexpected.

2. Scripting.

The biggest announced feature that didn't make it into the final version of *Myth* was a scripting language that would allow the player to modify elements of the game. We had hoped that user scripts could be written for extensible artificial intelligence, as well as custom formations, net game rules, and map behaviors.

We selected Java as a good basis for the *Myth* scripting language because of its gaining popularity, good information-hiding capabilities, and relatively simple byte code interpretation. After several months of work, early versions of the game loaded, compiled, and ran code from tag files. A few simple scripts worked for presentation purposes, including one that instructed a unit to search the battlefield for the heads of the enemy and collect them in a pile.

Unfortunately, when the programmer responsible for the scripting language parted ways with Bungie, we were left with a number of features to implement and no library of user-friendly interfaces with the game code. Given its incomplete state at such a late stage of development, there was little choice but to drop this functionality.

3. More frames of animation.

One of the complaints most often voiced by players is that the sprite-based units' animations are not fluid enough. At the start of the project, when we planned for the number of frames of animation per unit, there was a good deal of uncertainty regarding how much RAM would be consumed by large texture maps, sounds, and other resources. As things were, it was not uncommon for our landscape textures to reach 5MB in size, and certain animations already consumed close to 1MB -- our uncertainties were not unfounded. We erred on the conservative side. Though we implemented caching schemes that greatly reduced our memory requirements, there wasn't enough time to rerender the units.

4. Pathfinding.

Perfect pathfinding seems to have become the Holy Grail for games in the RTSG genre, and *Myth* is no exception. The game's terrain is a 3D polygonal mesh constructed from square cells, each of which is tessellated into two triangles. Cells have an associated terrain type that indicates their impassability, and they may contain any number of solid objects, including trees, fence posts, and units.

Ah! Square cells, you say? Having read previous Game Developer articles (Bryan Stout, "Smart Moves: Intelligent Pathfinding," *Game Developer*, October/November 1996; Swen Vincke, "Real-Time Pathfinding for Multiple Objects," *Game Developer*, June 1997), your first thought may be that the A* pathfinding should do the trick. The first problem with a pure A* approach for *Myth* is that impassable obstacles, such as troops and trees, may lie anywhere on the terrain. Penalizing the cells beneath impassable obstacles is a bad idea because the cells are fairly large and obstacles are not guaranteed to be aligned at the center of a cell. Furthermore, even if a tree did consume exactly one cell, the A* path to avoid it would make a unit walk up to the tree, turn, and continue around it. Units that bump into trees and walk between the centers of large cells appear extremely stupid; you really want your group of troops to avoid obstacles (including each other) ahead of time, and smoothly weave their way through a forest.

Myth Screenshot

To produce this effect, we created our own pathfinding algorithm. First, we ignore all obstacles and calculate the A* path based solely on the terrain impassability. For all intents and purposes, the terrain in *Myth* never changes, so this path can be calculated once and remembered. Now, we consider the arbitrarily placed obstacles and periodically refine our path using a vector-based scheme. If the planned path would cause us to hit an obstacle, we need to deviate our path. We recursively consider both left and right deviations, and choose the direction that causes us to deviate least from our A* path. Thus, we've considered terrain impassability information and we can avoid arbitrarily placed (or even moving) obstacles well before we bump into them.

For every game, pathfinding is a pretty complex and sensitive beast. This method worked well for 90 percent of our cases, but rigorous testing revealed certain cases that were not adequately handled. As the ship date drew near, we were forced to say "good enough" rather than handle these problem cases and risk introducing new bugs. Our current algorithm works pretty well and provides the effect we sought, but there's definitely room for improvement.

5. Features that missed the cut.

With a few exceptions, everything from our list of "Stuff that Rocks" made it into the final product. Those features that didn't make it came so close and were so exciting that they definitely deserve mention.

Near the end of the project, we started adding support for 3D fire, which would be ignited by explosions and flaming arrows. Our flames were sprite-based 3D particle effects, complete with translucent smoke. Fire could spread across the landscape and move at different rates over the various types of terrain. To our dismay, when a spark in the woods spread into a raging forest fire (as it should), all the translucent smoke sprites slowed even fast, 3Dfx-accelerated machines to a crawl. With little time to rectify the problem, we had to put out the fire, so to speak.

We had also planned for wildlife to scamper across the terrain and for birds to fly through the air, breathing life into our empty landscapes. Our attempt at ambient life started with a giant squirrel created by one of our artists. Unfortunately, due to time constraints, we didn't have a chance to create very interesting behaviors for it. Just about the only AI that we had a chance to code simply made the squirrels gravitate towards the player's units. We thought it best to drop ambient life rather than subject players to hordes of nuzzling squirrels.

Post-Release Reactions

With all the prerelease hype *Myth* had received, we were very anxious to see how the public would receive the final version. The reactions from beta testers were phenomenally positive, as were the comments from customers and reviewers. Our swiftness in correcting problems and adding several user-requested features with a 1.1 patch only earned us more kudos from the press and public.

But possibly the most satisfying result of the game is the degree to which it lessens the appeal of playing with a traditional isometric perspective. Working on *Myth* so consumed our time that we didn't get a chance to play anything else; we looked forward to playing some old favorites and the latest demos of our high-profile competition after we shipped. It was a real surprise to discover that once we were accustomed to *Myth*'s 3D camera and its associated freedom, playing isometric games was frustrating -- the action seemed distant and unrealistic, while the view of the world was annoyingly rigid. This sentiment was echoed in both player comments and reviews of the game. Since our Marathon products were derided by some as Doom rip-offs, it was especially satisfying to hear players say that *Myth* pushes the genre in a new direction, from which there's no looking back.

As of late 1997, *Myth: The Fallen Lords* had shipped 350,000 copies worldwide in four languages on two platforms. bungie.net currently boasts tens of thousands of registered users and is being expanded to keep up with the constantly increasing demand. As I write, it has just been declared Game of the Year by *Computer Games Strategy Plus* and Strategy Game of the Year by *Computer Gaming World*. It remains to be seen whether *Myth* will inspire other entries into the 3D real-time strategy game genre. But if nothing else, *Myth* is proof that a very small team with a strong product vision can still make a very big game.

Return to the full version of this article