GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Ritual Entertainment's *Heavy Metal: F.A.K.K. 2*

By Scott Alden

The decision to create a game based in the *Heavy Metal* universe started back in the hot Texas summer of 1997. Ritual Entertainment was in full-blown production of *Sin* when our concept artist's agent, Russell Binder, called up one day. Russell mentioned that his client, Kevin Eastman, was looking to make a videogame based on a new animated movie called *Heavy Metal: F.A.K.K. 2*. Kevin told us that the name of the movie was based on the name of the lead character in the movie and that character was being based on the image of his wife, Julie Strain. The name of the movie was later changed to *Heavy Metal 2000*, but we decided to keep the *F.A.K.K. 2* name, due to the fact that we already had two magazine covers and previews, and we didn't want to create product confusion.

**Julie's nemesis, Lord Tyler,
on a good day.**

Ritual jumped at the chance to create a game based in the *Heavy Metal* universe. Everyone at Ritual was familiar with the original *Heavy Metal* movie released in 1981, and a lot of us grew up reading the *Heavy Metal* illustrated magazine. A few business meetings later with Kevin and Russell brought the deal to a close.

Ritual then hired a new development team consisting of two programmers and two level designers. This small team worked on preproduction of *F.A.K.K. 2* during production of Sin in 1997 and on into the beginning of 1998.

Circumstances arose that caused the two programmers to leave Ritual, and the level designers were assigned to the Sin team. This caused *F.A.K.K. 2* production to slow down during Sin's final crunch, from March to October 1998. After Sin shipped, part of the team worked on SinCTF (a free multiplayer add-on to Sin) while the rest of the team moved over to work on *F.A.K.K. 2*. When SinCTF was released to the Internet, the *F.A.K.K. 2* team was fully realized. We started off with five programmers, five artists, five level designers, one support person, one sound engineer, and one project manager. However, when we finished the game the core team consisted of three programmers, three artists, three level designers, one sound engineer, and one support person. Art director Robert Atkins was the only team member that survived from the beginning of preproduction to the very end of the project when *F.A.K.K. 2* went gold on July 31, 2000.

In the midst of this turnover, many of the original design team left, and many elements of the original design laid out in preproduction had to be rewritten. *F.A.K.K. 2* production began in earnest in early March 1999 with the new design.

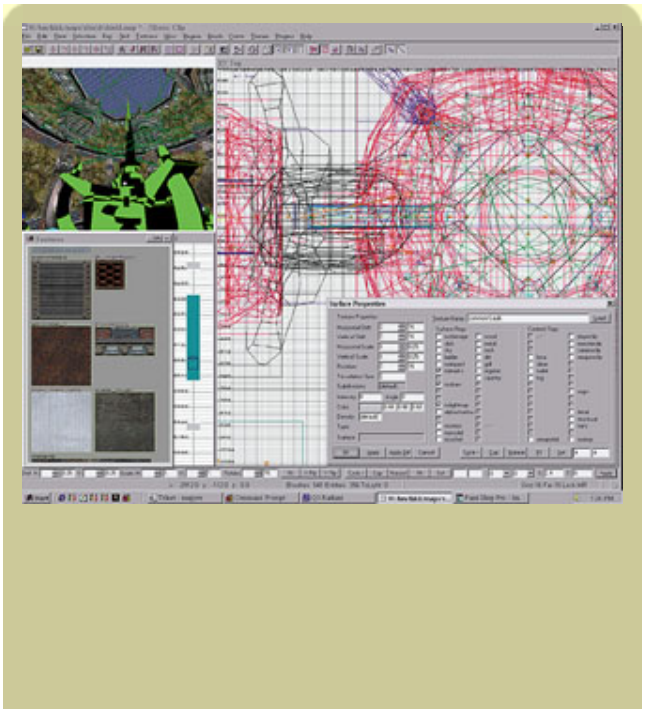
Concept sketch of Lord Tyler (final boss).

What Went Right

1. Team familiarity with the engine. We originally intended to use Ritual's UberEngine for *F.A.K.K. 2*, but it lacked some necessary components to complete the entire game. The UberEngine was basically all the components we designed and added to the Quake 2 engine for the development of *Sin*, plus a new renderer and networking layer. There was still a lot of work to be done on the UberEngine, though, and we chose to license a third-party engine for the project. The team was very familiar with the Quake 2 engine (thanks to Sin), and it was pretty high on our list of choices. The other engines we considered were Lithtech and Unreal. We came to the conclusion that we would have to do a lot of work to any engine (besides Quake) in order to do some of the things that we had designed in the preproduction of *F.A.K.K. 2*.

Luckily, an early version of the Quake 3 engine became available at the last moment before we had to make a final decision. We ultimately ended up with a limited Quake 3 license where we got a snapshot of the code in February 1999. Quake 3 had not been released yet, but it was in a workable state when we first got the code. This was the best decision we made during the entire project.

Ritual Entertainment is extremely familiar with the Quake family of engines, having started out with the Quake 1 engine on the *Quake Mission Pack #1: Scourge of Armagon*. Next, *Sin* was initially developed under a modified Quake 1 engine and was converted over to Quake 2 late in the project. The team was very relieved that we wouldn't have to add the extra learning time that it would take for us to become familiar with Lithtech or Unreal. The level designers would not have to learn a new editor, and since they were already familiar with the capabilities of the Quake engine, they could build prototype levels very quickly at the beginning of the project.



Q3Radiant level design tool showing the Shield Generator puzzle.
[Expand Image]

From the programmers' standpoint, it let us leverage all of the code that we wrote during our development of Sin. We were able to drop in the Sin game code and get a working version of the game very quickly, including our scripting language.

We received the Quake 3 code in late February 1999 and had a very impressive technology demo ready for E3 in early May. Since we had five programmers at the beginning of the project, in the early months of the project we were able to drop tons of new features into the engine and show them off at E3.

Concept sketch of the Soul Harvester (mini-boss).

2. In-game tools and engine modifications. The modifications that we made to the Quake 3 engine helped us put on the finishing touches and ultimately garnered *F.A.K.K. 2* such praises as "The Most Beautiful Game Ever." After hearing some of those comments, all the extra work required really was worth it. Almost all of the tools that we created on top of the Quake 3 engine allowed the level designers and artists to create content with a text file. When a system was designed, we always took into account how the team was going to use it. We used text files and simple English commands to allow the designer or artist to make game changes without having to recompile anything.

Another element that we added to the tool system was the ability to edit things in the game engine itself. This saved a lot of time for the artists and level designers. They got to see their changes immediately on the screen instead of having to exit out of the game and restart every time they changed something.

Skeletal animation system and LOD. One of the first modifications to the engine was the addition of a skeletal animation and level-of-detail (LOD) system. This allowed us to put in a ton of animations for the game's main character, Julie. We could also throw a lot of in-game models and monsters on the screen thanks to the LOD system. Our lead animator, Darrin Hart, hand-animated 11,000 frames of animation, of which about 6,500 frames were actually used in the game. This would not have been possible with a vertex-based animation system.

Morpheus scripting language. One of the best systems we came up with during the development of *Sin* was the scripting system. We gave it the internal nickname Morpheus, partly in tribute to *The Matrix*, and partly due to the fact that you could do just about anything you wanted to with it.

The scripting system is described in more detail in the *Sin* Postmortem I wrote in March 1999, but in a nutshell the scripting language gives the level designer complete control over any object or entity in the game. The functionality it provides ranges from the simple linear movement of objects around the level to the complex scripted sequences that exist throughout *F.A.K.K. 2*. We extended the functionality from *Sin*'s 500 or so commands to about 700 in *F.A.K.K. 2*. This added complexity put a more rigorous demand on the level designer, as they really had to put on a programmer's cap in order to use the scripting language in *F.A.K.K. 2*.

TIKI animation system. The DEF animation system from *Sin* was ported over in the early stages of *F.A.K.K. 2* development and renamed TIKI. Its main function was to allow the text file definition of models in the game. It also allows events to be synched with the animation on a per-frame basis. It was largely used for synching sounds and effects with the character animation.

In-game screenshot of the Ghost particle system menu.
[Expand Image]

Ghost particle system. The Ghost particle system was written completely from scratch and incorporated into the Morpheus scripting language and the TIKI animation system. Ghost allows artists and level designers to create user-defined particle systems and integrate them into the game via the animation system. As is the case with almost every Ritual-created engine modification, this is accomplished with a simple text file. There are about 50 commands used to modify the parameters of a particle system, and the systems can be combined together to create some complex-looking effects. For example, the firing of the Uzi in the game is a combination of five particle systems: smoke, shells, muzzle flash, tracer, and impact debris.

Cinematic camera system. *F.A.K.K. 2* is a very cinematic-intensive game. The story demanded some very complex scenes to advance the story, and we wanted to show as much detail as possible in these cinematic elements. We started with the spline-based camera system in Sin and ported it over to *F.A.K.K. 2*. We also added the ability to edit the camera paths in the game, actor tracking, and field-of-view control. This gave the cinematic designer complete control over the scene and allowed him to preview his changes immediately.

Sound system (Zound). Another in-game system is the Zound editor. This system let our sound engineer place sound and music triggers in the game without having to recompile the entire level. This allowed him to place music cues and music mood triggers around the level to create tension or humor when needed.

In a scene from Heavy Metal 2000 Julie convinces Germaine St. Germaine to help her out.

3. Third-party license and creative control. Kevin Eastman is the owner of the *Heavy Metal* magazine and co-creator of the Teenage Mutant Ninja Turtles. Together with Simon Bisley, they were the creative force behind the *Heavy Metal 2000* movie. When we discussed the possibilities of the game, we came up with idea of the game being a sequel to the movie instead of doing the standard movie-to-game conversion. He was thrilled to hear this, and fully supported our decision to make the game a sequel to the movie.

Very early in the production we received a crate full of *Heavy Metal* magazines -- every single issue. We spent days poring over the issues to see what kinds of styles *Heavy Metal* is known for, and to get inspiration for designing the characters.

Kevin also gave us complete control over the story and provided us with some inspirational concept sketches, which influenced the design of the game. Working with Kevin has been a blast and he's supported us completely with our decisions on what to put in the game. For instance, when we decided to resurrect Lord Tyler from the movie, Kevin wholeheartedly agreed.

The Ghost particle system in action.

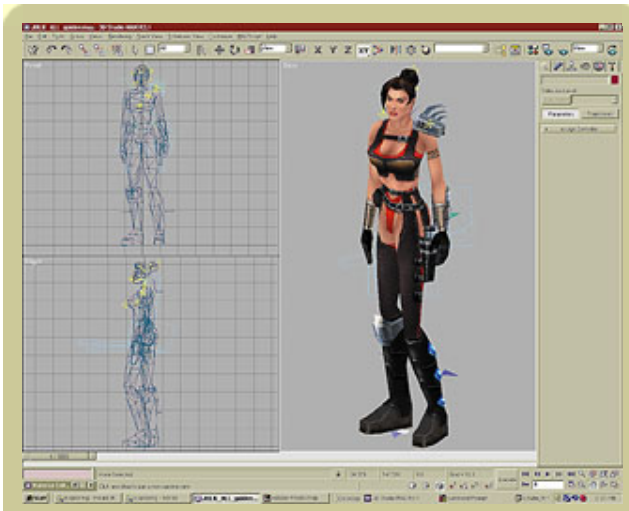
4. Focus on single-player. Early in the development of *F.A.K.K. 2*, we decided to focus solely on the single-player aspect of the game. This was a very tough decision for the team, as just about everyone at Ritual loves to play multiplayer games, and Quake 3 deathmatch is a frequent pastime around the office.

Since the team was small, we decided to focus on the single-player aspects of the game instead of trying to do multiplayer, which usually has an impact on what can be accomplished in the single-player version. We came up with a very tight game design and avoided repetitive gameplay. Almost every level in the game presents the player with new monsters and weapons.

All of the levels were sketched on paper before the level designers started working. They used these 2D maps to get a feel for how the level was going to be laid out and where all the action and encounters were going to take place. Another part of preproduction that helped out was the detailed description of the game's characters and monsters. The character sheets gave the artists the knowledge of how the creature was going to function in the game, which helped them create the necessary animations.

5. Strong focus on graphics. Right from the beginning of the project, we decided to focus strongly on graphics. We wanted the visual experience in the game to be something unparalleled in the game industry. Since the game is in third-person perspective, the main character had to look great -- if you spent the entire game looking at this character, it should be really nice to look at. We combined elements from the original *Heavy Metal* movie, the comics, and the new movie to create Julie. We went through three revisions of her character before finally settling on what you see in the game.

When we designed the world, we wanted it to be an interactive version of the *Heavy Metal* universe. We decided to use a rich color set with lots of red and yellows in the town, and we used vibrant greens and blues for some of the outdoor areas. We wanted to get away from the dingy, dark look that so many other shooters in the genre have. We put the Quake 3 shader system to great use and were very liberal in our use of weapon effects and cinematics. For our efforts, critics have proclaimed *F.A.K.K. 2* to be one of the best-looking games of 2000.



3D Studio Max showing Julie in the battlesuit with

3D Studio Max showing Julie in the battlesuit with various armor attachments.
[Expand Image]

What Went Wrong

1. No project management. During the development of *F.A.K.K. 2*, we had a project manager at the beginning of the project and a different project manager near the end. There were 12 months of development time in between, during which we had little to no management on the project. There were members of the team that took on this role, but only in a limited capacity, as they had tons of other work to do as well.

Not having a single person manage tasks, and letting just about every feature request get added to the list, was a major reason why we had to go into crunch mode in order to finish everything by our July deadline. This resulted in a crunch mode that lasted nearly five months, and one month of "super crunch," consisting of seven-day weeks and 12- to 16-hour days. I don't mind crunch mode every now and then, but for an extended period of time it really wears you out. The team morale during this crunch was very low.

We worked on never-ending task lists for months and months. Just when you thought you were close to being done with what you were assigned, another 30 tasks would appear on the list after a team meeting where we would flesh out the incomplete areas of the design. Having a good project manager allows you to have a majority of the design details fleshed out from the beginning, schedule the correct amount of time for tasks, and have the appropriate number of people on the team to finish a game. This seems to be a major problem in the gaming industry, as nearly everyone I talk to has just about the same story about project management and death-march crunch modes.

In-game model of Julie wearing her flightsuit.

2. High team turnover rate on a small team. At the beginning of this article, I mentioned that the team started off with 18 members, and we finished the game with 11. Of those 11 people, only one person was on the original *F.A.K.K. 2* team from the beginning. It was a weird project, because most of the team didn't have a handle on the design of the game. The key designers of gameplay had moved on, and weren't available to talk about the ideas that they had come up with. We ended up scrapping a lot of the original design document and starting over. This set us back pretty far in the gameplay area. Another area that suffered was models and animations. We lost several artists who worked on different animation packages, and when they left we had to redo the models they were responsible for.

An excerpt from the F.A.K.K. 2 comic by Joel Thompson that appeared in Heavy Metal magazine.
[Expand Image]

The gaming industry is a turbulent one; people join and leave companies on a regular basis, and it definitely has an impact no matter what anyone says. Unless a person didn't contribute anything to the game, losing personnel greatly impacts finishing a game on time. *F.A.K.K. 2* had a very small team that had to work extra hard to make up for the lost employees that weren't replaced. In the end though, we were very satisfied with the game's quality in spite of having lost so many people during the course of development.

3. No multiplayer; a short game for hardcore players. As I said, *F.A.K.K. 2* was designed as a single-player game from the start. We were going to try to put multiplayer into the final release if we had the time, but our July deadline came so quickly that we just didn't have the time to finish it. We also designed a very tight game that can be finished by hardcore game players in less than ten hours.

In-game screenshot of Julie's house.

This was our biggest complaint from reviewers and players alike. I do agree that the game is on the short side, but we didn't want to put in dozens of levels that repeated the same gameplay over and over, as so many other games do. Even though I am defending our decision in this article, I do acknowledge it as one of the problems that we had with the design. A short game with no multiplayer has a very limited lifetime in the gaming industry. (Note: As of the writing of this article, a multiplayer patch is in the works that will provide arena-style battles in various *F.A.K.K. 2* settings).

4. No demo before release. Our July deadline was fast approaching, and we had yet to release a playable demo to the Internet. This hurt us in two ways. First, we weren't able to build up any pre-game buzz by having a killer demo for our game, and when the game was released people seemed surprised to hear about it. Second, a lot of people will not buy a game unless they play a demo beforehand to see if they like it.

Fortunately, we were able to get a demo out the door within two weeks of our release, but the jury is still out on whether or not this is a good or bad thing.

In-game screenshot of Julie battling the Happy Mask Hourde creatures.

5. Complex systems. I mentioned this problem in the*Sin* Postmortem back in March 1999, and it looks like it hit us again. It's not surprising, since we used many of the same systems that we had in Sin, and we even extended them. Game design on the PC is becoming more and more complex as time goes on. Level designers aren't just creating levels anymore. They have to design puzzles, scripted events, cinematics, and on and on. In *F.A.K.K. 2* this is all done through the scripting language, so each level designer had to be something of a programmer as well. On the artists' side, they also got a taste of programming. The effects system is driven entirely by text files, and the artists needed to learn the system's intricacies in order to get the effects they wanted.

This complexity added more to the development time than we had anticipated. Even though everyone on the team was really familiar with the engine, getting used to the new tools and modifications took a lot of extra work. The scripts for *F.A.K.K. 2* total nearly 10,000 lines. I'm not really sure if this problem is ever going to change in the industry. As people demand ever more immersive game environments, the complexity goes up in order to attain these goals.

In-game model of Gith Recruiter.

In the End

Even though *F.A.K.K. 2* had its share of development problems, we are very proud of what we accomplished with the game. While the design of the game was not always as well defined as some of the team members would have liked, we still were able to create an incredibly fun third-person action game in just under 18 months. We feel that *F.A.K.K. 2* is our best game to date, and our future games will only benefit from the experiences we had during its development.

Game Data

Ritual Entertainment
Heavy Metal: F.A.K.K. 2

Publisher: Gathering of Developers

Project length: 18 months

Number of full-time developers: 11-18

Number of contractors: 1

Budget: Approx. $2 million

Release date: July 31, 2000

Platforms: Windows 95/98/NT

Development hardware used: 450MHz Pentium IIs with 256MB RAM, TNT2 Ultras, and 13GB hard drives.

Development software used: Visual C++ 6.0, Photoshop 5.0, 3D Studio Max 3.1, Deep Paint 3D

Notable technologies: Quake 3 engine from id Software, RAD Game Tools' Miles Sound System, InstallShield 6.2, DemoShield 6.51, Robocopy 1.96 (from the Windows NT Resource Kit)

Project size: 364,825 lines of code; 11,519 files