Postmortem: Epic Games' *Unreal Tournament*

By Brandon Reinhart

Unreal Tournament, released in November 1999, was, in a way, an accident. After the original Unreal was completed, Epic wanted to follow up the project with some sort of add-on pack. Unreal multiplayer code was very poor, so the team felt that an expansion that improved multiplayer would be ideal. As feature lists grew and patches to *Unreal* were released, the add-on turned into a complete and independent game.

Unreal Tournament has certainly seen a very nontraditional development cycle, one that I feel would not have succeeded in any other genre. Ultimately, our decisions paid off, because the game earned more than five "Game of the Year" awards and is consistently rated in the top ninetieth percentile in reviews. The online community is producing excellent expansions and modifications to the game and we feel that *Unreal Tournament* will be around for a long time to come.

Early Development

>A proper look at the development of *Unreal Tournament* begins with the completion of *Unreal*. The *Unreal* engine was four years under development and the team was wearing down. When the game shipped, it met with a large amount of acclaim, but that positive image was tarnished over time as hardcore players began to complain about the terrible network support. The *Unreal* team was now faced with several more months of work on the game, essentially to bring it to the point it should have been at when it was put on shelves.

Early in the process, plans were discussed to work on an official Epic add-on to *Unreal*. The add-on would introduce much-improved network play, new maps, and probably some new game features. The original ideas for the add-on were never put on paper and it never had a name. I was hired by Epic in August 1998 to assist with patching *Unreal*. Eventually I started to write new code for the add-on with Steve Polge.

Initial work on the add-on in early summer 1998 was made difficult by the fact that Epic was a virtual company. The last year of *Unreal*'s development took place in Canada, with the U.S.-based Epic team flying back and forth to work with Digital Extremes in London, Ontario. When *Unreal* was finished, no one at Epic wanted to travel anywhere, but at the same time the team recognized that they needed to move to a central location. The team decided to relocate all of its employees to Raleigh, N.C.

By September 1998 everyone was together or had a travel plan. Work started to come together rapidly on the add-on project. Steve Polge had laid the groundwork for several new game types, including Capture the Flag and Domination. The level designers had five or six good maps ready for testing. Throughout sporadic but intense meetings, the team agreed to focus the add-on entirely on improving the multiplayer aspect of the game with new features and better net code.

A close-up shot of the Black Thunder skin on Shane Caudle's Male1 model. This was one of the first new skins developed for *Unreal Tournament.*

The amount of content grew and we soon realized we had a much larger project on our hands than we had originally thought. In November,

after meetings with our publisher GT Interactive, Mark Rein suggested we turn the add-on into a separate game. Initially, the team opposed the idea. We wanted to finish the project quickly and move on to something fresh. The promise of a much higher profit potential, coupled with our recognition of the state of the project finally led us to agree with GT. In December, the name *Unreal: Tournament Edition* was chosen, with "Edition" subsequently dropped from the title.

A Game Takes Shape

Epic's internal structure is extremely liberal, probably the most liberal in the entire gaming industry. Programmers work on the projects they want to work on, with major features being assigned to whoever steps forward to take on the task. Artists work with level designers but are given significant design freedom. Level designers work on the kinds of maps they think would be cool. This design philosophy pervaded *Unreal Tournament*'s development.

In December, I downloaded a sample of a new *Unreal* mod under development by an Australian named Jack Porter. The mod, UBrowser, was a server browser using a Windows-like GUI. It was impressive, so I showed it to James Schmalz, lead designer at Digital Extremes, who said, "We need that, we need to hire this guy." A few weeks later Jack was a part of the team, expanding his UWindow GUI and reworking *Unreal Tournament*'s menus to use the system. Jack fit into the team perfectly, bringing a complete solution for the interface and menus as well as his own independent programming initiative.

Weekly meetings infused order into our chaotic corporate structure. Everyone would debate and yell about what features were cool and what features sucked. The assignment of major features was largely automatic. Tim Sweeney worked on improving net code and engine fixes. Steve Polge wrote the original AI code and focused on adding player orders and other improvements (in addition to filling out the new game types). Jack had the windowing system and a lot of menus to work on. Programmer Erik de Neve was in Europe putting together level-of-detail code as well as experimenting with next-generation technology. I worked on the single-player game, game-play features, scoreboards, HUDs, special level actors, tutorials, and wrote a lot of the game's story and character background content.

The best features were added entirely by the initiative of individuals. Level designer Cedric "Inoxx" Fiorentino designed CTF-Face, an extremely popular Capture the Flag map. I added the Multi-Kill system after a short discussion with lead designer Cliff Bleszinski sparked the idea, and Jack implemented decals shortly before we shipped. It was this individual creativity that ultimately bound the team together. Each new feature infused everyone with the enthusiasm to add more.

Unreal Tournament's deathmatch maps were not constrained to any one particular theme or timeframe. Cliff Bleszinski's DM-Barricade, shown above, is a castle floating above a storm, while Pancho Eekels' DM-Galleon is a massive ship sailing the ocean

Once the first batch of new player models, weapon models, and maps was completed we realized we had a game quite different from Unreal. Feedback from the *Unreal* deathmatch community (including the highly vocal Quake community's complaints) also drove our designs. Subtle alterations to player movement and control changed the feel of the game completely. Some changes in game play - such as whether to enable weapon-stay in single player - were controversial, so we held polls on popular Unreal message boards.

Throughout the spring and summer of 1999, Epic was pursuing contract renegotiations with GT Interactive. Everyone believed the game could ship at any time, so development became stop-and-go. We would be in a code lockdown one week and adding major new features the next. The result of this jarring development cycle was good and bad. The periods of code lockdown allowed us more time to play-test and fix bugs, which contributed greatly to the game's overall polish. On the other hand, it prevented us from adding many features that would have otherwise been included and it was detrimental to the morale of the team. We liked working on *Unreal Tournament*, but it still felt like old technology to us. The world had seen the *Unreal* engine; we were ready to move on.

New Code, New Features

As it turned out, though, we had a lot of time to enhance the engine. *Unreal* was before its time and a lot of the content and code was rushed by the need to ship. With *Unreal Tournament*, the team had a lot of time to use previously unexplored engine features. Erik de Neve's level-of-detail code ended up really speeding the game up, giving us room for beefier characters and more map decorations.

Early on we experimented with using 16 256¥256 textures per player, but opted for three or four 256¥256 pieces out of memory considerations. This quadrupled the detail available to our skin artists for the player models. Reserving one of the 256¥256 textures for the

head alone allowed us to mix and match body skins with heads, yielding a massive amount of customization with only a small amount of work. Another one of the 256¥256 textures was reserved for team color bits, so that a player skin could encompass all five possible team colors (none, red, blue, yellow, green) without too much memory use.

Level design didn't stand still either. Changing from single-player to deathmatch-oriented design was refreshing for the designers, but not without its unique challenges. One issue was the task of balancing the number of "hardcore" maps with "theme" maps. A hardcore map focuses entirely on layout and game play, while the overall style of the map comes second. Theme maps, on the other hand, focus on a unifying idea or look and build from that. For example, the Koos Galleon, designed by Pancho Eekels of Digital Extremes, is a large sailing ship. It's a very beautiful level, but focuses on the theme of being a ship more than being a deathmatch map.

***Unreal Tournament* used from three to four 256x256 textures per model. This allowed us to focus a lot of detail in the head and face area. Within the game a player can choose a skin and then swap through several different faces. This means players on a team can wear identical armor and clothing but have unique faces.**

The *Unreal Tournament* team decided that mixing the two styles was the best approach. While most magazine reviews have expressed frustration at the theme-oriented maps, we didn't want to appeal to only the hardcore crowd. Including maps that were designed for their look and feel increases the game's interest to average players who aren't skilled enough at the game to benefit from hardcore designs. Realism through textures and architecture is one of the *Unreal* engine's strengths and it was critical that we exploit that strength. Ultimately, we shipped *Unreal Tournament* with somewhere around 45 to 50 maps, offering more than enough variety and replay value for everyone.

Another task we faced was choosing which of *Unreal*'s weapons to keep and which to ditch. *Unreal Tournament* has two firing modes which makes designing a weapon like designing two weapons in one. *Unreal*'s stinger and dispersion pistol were not needed in *Unreal Tournament*. Those weapons were good in *Unreal*, because a player needed to start with simple, weak weapons and build up. In *Unreal Tournament*, all the weapons had to be equally effective and carefully balanced. A player good with the minigun needed to be lethal with it. A player good with the pulse gun needed to be lethal with it. Eventually we settled on the current load-out, but made quite a few game-play changes to the weapons that stayed from *Unreal*. Each weapon was also given a much more beefed-up look and sound.

An interesting little anecdote: GT started doing promotion for *Unreal Tournament* before the new rocket launcher was finished. They produced a lot of marketing material with old screenshots showing the eightball launcher from Unreal. If you look at the gold trophy used in the print ads, you'll see the characters at the top are holding eightballs, a weapon that isn't in *Unreal Tournament*.

In the End, It All Worked Out

While the talents and devotion of individual team members created the content, the overall team spirit tied it together. *Unreal Tournament*'s design process was often reckless, but the game that resulted is nevertheless very polished and a hell of a lot of fun. The deathmatch-focused first-person shooter doesn't need a story, dialogue, or scripted sequences, which are all features that more or less require an organized design. Had we applied our hodgepodge design approach to a more focused genre, we probably would not have had such a successful game. *Unreal Tournament* should not be seen as a lesson in how to design a game, but as a lesson on how to organize a small team of developers.

What Went Right

1. Smart internal marketing team

At the front of Epic's public relations were Mark Rein and Jay Wilbur. Their job was particularly difficult during the development of *Unreal Tournament*. The media perceived us as impossible upstarts, taking an engine with terrible net-play and attempting to compete against id

Software, the industry multiplayer champion. Both Mark and Jay fought hard to win over supporters in the online and magazine press. Mark made sure that the team stayed professional and that everyone was saying what he needed to be saying. Jay hunted down potential engine licensees, and helped establish a level of curiosity among the community and media.

Unreal Tournament was able to garner significant magazine coverage because of the ongoing "*Quake* killer" debates. Mark and Jay worked to turn the initially negative public response into something positive. While we felt that our game would definitely stand on its own, we had to ensure that the positives were being clearly broadcast. Epic was very careful to avoid mentioning *Quake 3: Arena* whenever possible, keeping the focus solely on *Unreal Tournament*'s features and staying away from comparative previews. Most interviews and previews would ask us the inevitable "What about *Quake 3*?" question, to which we tried to answer with complete respect for id's project. Everyone knew that *Unreal Tournament* and *Quake 3* would be pitted against each other. Mark and Jay established very early on that the competition would be friendly.

The characters in *Unreal Tournament* were designed to be futuristic pit fighters. The selection of characters include ex-military specialists, criminals, and alien warriors such as the Necris Phayder Assassin pictured above.

2. Liberal internal structure, open design discussion

The laid-back environment that both Epic and Digital Extremes fostered greatly enhanced the quality of *Unreal Tournament*. Everyone was free to suggest or implement an idea.

Programmers had as much design freedom as anyone else on the team. Cliff Bleszinski (Epic) and James Schmalz (DE) were the lead designers of their respective companies and served as content filters. They worked towards focusing the ideas put forth in the meetings. In addition, both of them contributed significantly to the final game content. James designed two of the player models and created many skins and faces, while Cliff designed many of the game's best maps.

Team members were allowed to come into work when they wanted and stay however long they felt like being there. The only requirement was that every member attend a weekly design and focus meeting. This system worked because Epic was very careful to hire mature, dedicated employees and the core development team was kept small. The open hours often saw team members working a 24-hour day, sleeping on a couch for six hours, and then working another 24-hour day.

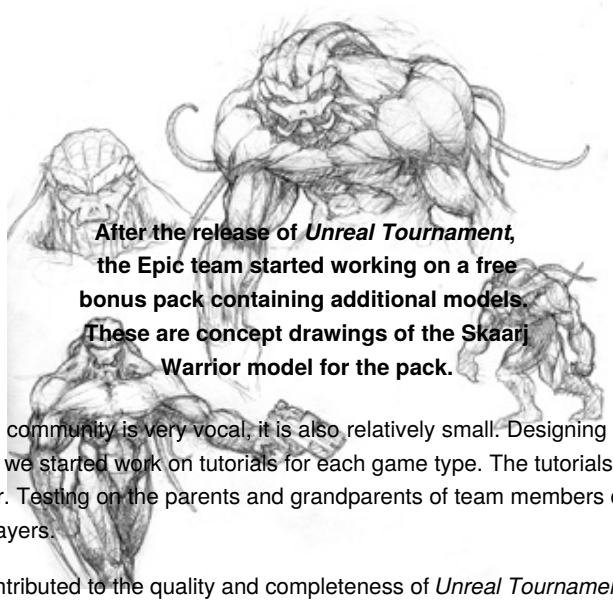
In addition to fostering a hardcore work ethic, the system created a sideways information flow. A programmer would go straight to the artist he needed something from, instead of through an art director. The fast communication allowed the programmers to stomp out bugs relatively quickly and the level designers to talk directly to the texture artists. An example of this was the single-player ladder system. Shane Caudle designed the art and I wrote the code. The fewer people we had to consult in order to complete the task meant a much faster turnaround. Everyone participated in giving the "coolness factor" thumbs-up or thumbs-down, but the actual development process was intentionally kept thin.

3. Direct communication with the gaming community

Nearly every Epic and Digital Extremes employee frequented message boards dedicated to the subject of *Unreal* and *Unreal Tournament*. The majority of Epic employees were drawn directly from the gaming community, either through mod projects or independent game work. Keeping in contact with the gaming community allowed Epic to focus on the target audience during the design process.

Beyond our direct communication with the *Unreal* community, we also trolled *Quake 3* message boards, reading the discussions of the fans of our lead competitor's game. Learning what people liked in a first-person shooter and why they liked it helped us change the marginal multiplayer experience in *Unreal* to the much faster paced game play in *Unreal Tournament*.

The gaming community can really help set the tone for your game. When Unreal was released, the online community became extremely vocal and angry about the state of the net play. While most magazines had reported positive experiences with *Unreal*'s single-player mode (reflected in positive reviews), the media eventually came to reflect the cries of the hardcore gaming community. This was in part because the net play was poor, but also due to the fact that many members of the gaming media are themselves hardcore game players and visit those same message boards and community outlets.



After the release of *Unreal Tournament*, the Epic team started working on a free bonus pack containing additional models. These are concept drawings of the Skaarj Warrior model for the pack.

We also learned that while the hardcore community is very vocal, it is also relatively small. Designing a game to appeal to that community alone is a critical mistake. Early in 1999 we started work on tutorials for each game type. The tutorials are far from definitive, but they did cover the basics of playing a 3D shooter. Testing on the parents and grandparents of team members demonstrated that the tutorials were useful for attracting and keeping new players.

This community-mindedness greatly contributed to the quality and completeness of *Unreal Tournament*. We had a very good idea of what players wanted. As I mentioned earlier, we often posted controversial design questions on public message boards to gauge public reaction. The results of these polls were taken into consideration when the feature in question was implemented.

4. Strongly object-oriented engine design

The *Unreal Tournament* engine's strong object-oriented design makes it extremely modular. This modularity allowed our programmers to make massive changes to parts of the game without affecting other features. Each subsystem is connected to other subsystems through a clearly defined interface, and platform-specific code is consigned to separate libraries. Creating the Linux port, for example, was simply a matter of rewriting an input and sound device and writing a Linux version of the platform-specific library behavior.

Throughout *Unreal Tournament*'s development, Tim Sweeney and Steve Polge worked on improving the networking code. The modularity of the engine meant that their work didn't disturb anyone else's work. Some features, such as Jack's decal system, were added very late in the project. The decal system added a lot of depth and feedback to the game, and took less than a week to get working and fully debugged. Erik de Neve's mesh level-of-detail code touched only a handful of source files.

Every weapon in *Unreal Tournament* has two distinct firing modes. This made designing and balancing each weapon twice as complex as a normal first-person shooter.

This ease of use is also reflected in the engine's scripting language, UnrealScript. Calling it a scripting language is a misnomer; it's actually a lot like Java. Weapons, pickups, level events, AI nodes, and other world actors are all independent objects. A weapon can be added to the game without touching any source files but the new object definition. This highly extensible language meant that each programmer could add extensive new game-play features with a very limited set of potential side effects. In the end, 90 percent of *Unreal Tournament*'s game-play code was written in UnrealScript.

The *Unreal Tournament* engine's modular package system coupled with UnrealEd makes the game a mod-creation system out of the box.

We designed a lot of our code with amateur extension in mind. Everyone at Epic recognized the value of the mod community and we wanted to make the game attractive to new artists and programmers. Constructing game code in this way made it much easier for us to prototype our own new features. Early UT weapons and pickups were child objects of Unreal. The two games can easily coexist even now.

5. Good Timing

As I said earlier, *Unreal Tournament* was developed in the same time period as id Software's *Quake 3: Arena*. The two games promised to be of the same genre and the two companies were known for a high level of competition. While we tried to avoid the "*Quake 3* vs. *UT*" comparisons, they ultimately worked in our favor. The high level of public interest in the new engine war greatly increased our visibility. Magazines and web sites often posted split previews instead of focusing on one game in particular. Interviews with id employees would always lead to *Unreal Tournament* questions and vice versa.

Unreal Tournament took almost exactly a year and a half to develop, giving the team a lot of time to pack in features. We didn't have to focus on writing an engine from scratch, so we were free to focus entirely on improvements. At this point, we've released three patches for *Unreal Tournament* that have solved a handful of relatively minor problems. The team has had a lot of time available to spend on adding even more features to the game since its release, instead of fixing outstanding issues. By the time this article hits the stands, we'll have released our first bonus pack: a free collection of new models, maps, and game-enhancing features.

What Went Wrong

1. Bad timing

Many aspects of the game's timing worked against us. While the *Quake 3* vs. *UT* hype increased our exposure, it also set a very hard deadline for completion. It was critical that we complete the game before *Quake 3* was released. The media advantage belonged to id and we believed that if *Unreal Tournament* launched after *Quake 3*, we would be forgotten in the storm. At the same time, however, we were caught up in grueling contract renegotiations with GT Interactive. We did not want to deliver the completed game until we knew the contract would work in our favor. Many times during the development of the game we were promised that a resolution to the contract issue was close at hand. The team would race to reach a point where the game could be shipped, only to have negotiations drag on.

The gold master was delivered to GT days after a final contract was agreed upon. Unfortunately, the game hit shelves in November, pushing us very close to *Quake 3*'s release date. While *Unreal Tournament* often performed better than *Quake 3* in reviews, we believe that sales would have been much higher still had we released in October. Word-of-mouth is a powerful force and the extra month would have given us time to build a larger community before Christmas.

2. No central design document

While I am a big supporter of open, cabal-style design, I have to stop and wonder how *Unreal Tournament* would have turned out had we a strong initial design. It's quite possible that the game's weaker elements would have been much stronger if we had put together some concept art and focus material. In reviews, we have been criticized for not having enough variation in characters. If *Unreal Tournament* had had a library of concept art to draw from, we might have had more interesting alien warriors. The story is more or less nonexistent in *Unreal Tournament*, but at times we considered having in-game cutscenes as rewards for a player's progress. The idea was dumped, but a design document might have made it easier to visualize those scenes.

The *Unreal Tournament* development team felt that several of *Unreal*'s weapons were a lot of fun. Here is a bot carrying the Shock Rifle, an updated version of *Unreal*'s ASMD.

I suppose this isn't really a "what went wrong." It's simply more of a "what we should have done." I think it's important to think about the game in that light. *Unreal Tournament* is a very fun game with a lot of features packed into a short amount of development time. Those features

were largely added through spur-of-the-moment decisions. A more unified approach to design would have allowed us to construct features that play on features, or even think of ideas we didn't have the perspective to realize. Epic will always be a very open, liberal company when it comes to the design process. If we develop a design document, we'll use it with the understanding that it can be modified at any time. That having been said, I think there is a definite positive argument for having some sort of central guide to everyone's ideas. Having the ability to sit down and look over the big picture is very valuable.

3. Co-development across two countries

Epic Games and Digital Extremes co-developed *Unreal Tournament*. The Digital Extremes team was located in Canada and Epic was located in the U.S. Epic supplied the programming team and a large group of content designers. Digital Extremes provided level designers, a sound guy, and texture artists. James Schmalz, the high-up man at Digital Extremes, contributed two of the game's player models and a lot of art. This co-development worked well for the most part, but near the end of the project it became very difficult.

During *Unreal*, Epic team members flew to Canada to work at Digital Extremes' offices. With *Unreal Tournament*, it became Digital Extremes's turn to do the traveling. Unfortunately, flying and driving back and forth every couple of weeks is a very draining experience. Many of the Digital Extremes team members spent several weeks away from their wives and girlfriends. Near the end of the project, they grew increasingly frustrated with the situation. To compound this problem further, Digital Extremes and Epic were attempting an expensive merger. As *Unreal Tournament* came to a close, it became clear that the merger would not happen. It was prohibitively expensive for a small company to move across the border. Many Digital Extremes team members already had apartments and plans for living in Raleigh, and the news of the terminated merger process was devastating.

In the Assault game type, players have to enter a heavily defended base and complete map-specific objectives to win. Assault was the most difficult *Unreal Tournament* game type to design, balance, and play-test.

Much to Digital Extremes's credit, the company quickly recovered and moved to its backup plan of developing its own game with the *Unreal Tournament* engine. Nonetheless, the process of co-developing the game had taken its toll on everyone. The ups and downs of the merger process had a negative effect on team morale. Had the co-development happened between companies more closely situated, it would not have been a problem.

4. Not enough artists

On the content side, *Unreal Tournament* was held back by the number of available artists. Epic's artist, Shane Caudle, is a supreme Jack-of-all-trades, creating skins, models, and levels of the highest quality. He spent most of his time working on new player models and skins for those models. Digital Extremes brought a few texture artists to the table, but not enough to create the huge libraries of new textures needed for the game. In order to supplement the skin and texture production, Epic turned to contract artist Steve Garofalo.

Steve Polge, our AI and game play programmer, made the bots understand the unique advantages and disadvantages of each weapon. Here a bot is moving in very close to use the powerful Flak Cannon.

Even with the additional help from external sources, the team was unable to produce enough new textures. Level designers who wanted custom textures for their maps had to make do with their own texturing ability. While the final texture and level count in *Unreal Tournament* is quite high, the levels would have been much more impressive had the team been able to act with full freedom. Since the completion of *Unreal Tournament*, Epic has hired both Steve Garofalo and John Mueller to strengthen the art team for future projects.

5. Visual Basic editor interface

The *Unreal Tournament* engine uses UnrealEd as its level design and content management tool. For several years, UnrealEd has used a windowing interface written in Visual Basic. The VB code is fragile and very old. Add to this the fact that nobody at Epic except Tim Sweeney knows or cares about VB, and you have a level design team that is stuck with a tool that's not easily updated.

Several interface bugs have plagued UnrealEd for some time and nobody on the team had the time or inclination to fix them. If we had a more easily extensible tool, the team would have been able to add extra features to the editor for level designers to use. As it stood, the editor was considered "off limits" for new features.

For our next few projects we will most likely use a new C++ editor that Legend Entertainment developed. For *Unreal Tournament*, however, we simply didn't have the time to work on a new editor. Fortunately, our time spent using UnrealEd taught us the dos and don'ts of tool design.

Where We Go from Here

The things that went wrong are, all in all, much less significant than what went right. *Unreal Tournament* could have benefited from a more focused initial design and a more solid ship date, but it turned out to be very polished and a lot of fun. Many of the factors that worked in our favor, like timing, also worked against us to some extent. "What went wrong" is a good way of looking at what we could have done to make *Unreal Tournament* even better.

Epic has developed some pretty clear plans of where we want to go from here. We've been working on free content to release to support *Unreal Tournament*. We are also looking into doing some kind of Playstation 2 version of the game. After that, we want to focus on an entirely new engine technology for the PC. In the short term, Jack Porter is working on his terrain system and Erik de Neve is putting the finishing touches on the skeletal animation system. Tim Sweeney has been developing an entirely new programming language to support the next engine, with some very powerful features such as parameterized functions.

Unreal Tournament served as a good learning tool for the team. We have a good idea of what processes we need to adopt to produce larger, more story-driven games in the future. We see *Unreal Tournament* as a good lesson in how to organize a team and produce a game in a short amount of time. The team has grown socially, and everyone is much more experienced in the process of game development. We feel very prepared to face the upcoming challenges and, hopefully, to continue to be seen as innovators in the industry.

Brandon "GreenMarine" Reinhart is a 21-year-old programmer formerly with Epic Games Inc. *Unreal Tournament* was his first game after being recruited by Epic from the *Unreal* and *Quake 2* mod community. He is obsessed with games, game programming, and game design. When he isn't playing games, he can be found reading Michael Moorcock, painting miniatures, or listening to the latest in Norwegian black metal. Blodu Ok Jarna!

Unreal Tournament

Epic Games Inc.
Raleigh, N.C.
(919) 854-0070
http://www.epicgames.com

Digital Extremes
London, Ontario, Canada
(519) 657-4260
http://www.digitalextremes.com

Release date: November 1999

Intended platform: Windows 95/98/NT, Linux

Project budget: $2 million

Project length: 18 months

Team size: approximately 16 developers

Code Length: 350,000 lines of C++ and UnrealScript

Critical development hardware: Pentium II 400s with 256MB RAM and Voodoo 2 or TNT-based cards

Critical development software: Microsoft Visual Studio, 3D Studio Max, UnrealEd

Return to the full version of this article