



Tim Sweeney Archive - Interviews

<http://www.team5150.com/~andrew/sweeney>

April 3, 2008

Contents

1	Beyond Unreal	3
2	An Interview with Epic's Tim Sweeney	8
3	Next gen engines: Tim Sweeney interview	21
4	Tim Sweeney & CliffyB Interview	26
4.1	The Game	27
4.2	The Engine	30
4.3	The Community	33
4.4	The Future	35
5	Tim Sweeney 64-bit Interview	37
6	Tim Sweeney of EpicGames	42
7	Interview with Epic's Tim Sweeney on UnrealEngine3	51
8	Talking nasty with Tim Sweeney	59
9	Q&A: Epic Games' Tim Sweeney on Ageia PhysX	63

10 Tim Sweeney/Unreal Engine 3 Interview	67
11 Tim Sweeney: the man, the mystery, the legend.	71
12 DirectX 10 Preview: The Future of PC Graphics and Gaming	77
13 Exclusive Interview with Epic's Tim Sweeney on Unreal Engine 3 and UT 3	82
14 EVGA Gaming Q&A with Tim Sweeney	87
15 Unreal creator Tim Sweeney: "PCs are good for anything, just not games"	92
15.1 PCs are good for anything, just not games	93
15.2 Intel's integrated graphics just don't work. I don't think they will ever work	95
15.3 DirectX 10 is the last relevant graphics API	98
15.4 Running Linux on a GPU is not a pipe-dream	101
15.5 Unreal Engine 4.0 aims at next-gen console war	104
15.6 The development of Unreal Engine 4 has begun	106

Chapter 1

Beyond Unreal

This interview was conducted by Dave "Fargo" Kosak for GameSpy on Jan 25, 2000.

<http://archive.gamespy.com/legacy/interviews/sweeney.shtm>

Interview

Dave "Fargo" Kosak: The Unreal Engine has been out for over a year and a half – how far do you think it's come since its release? What do you think are the biggest changes as it's evolved? Any surprises?

Tim Sweeney: There are four areas of improvement that really stand out in Unreal Tournament:

Robustness: Since Unreal 1 shipped, we've looked at literally thousands of bug reports and requests from users and improved the engine's stability a huge amount. With Unreal 1, we made a huge leap in technology, but the end result was very rough in many areas, including network code, performance on low-end machines, and ease of use. The 18 months of testing and fine-tuning we've been doing during Unreal Tournament's development has led to tremendous improvements here.

AI: While Unreal 1's "botmatch" AI was the first such feature to ship in a

3D game, Steve Polge wasn't satisfied with just that, and has taken it many steps further with Unreal Tournament's AI: now the bots know about team coordination; they're much more human-like in their fighting and strategic planning; and they're good enough now that you can start a 10-player CTF game, with you and four bots versus a team of five bots, and have a great teamplay experience, "no humans required".

User interface: Unreal 1's interface was derivative of Quake, which was derivative of DOOM, which was derivative of Wolfenstein 3D. In the years from 1991 to 1998, really no substantial improvements occurred in 3D game user interfaces – sure, everyone added polish, but it was the same archaic keyboard-driven menu system. For Unreal Tournament, Jack Porter ripped out all that code and replaced it with a very modern, mouse-driven user interface, sort of a "windows on steroids". This has enabled us to expose a gigantic amount more functionality to users which wouldn't have been possible with an old-style interface. Now, picking maps, selecting game options, choosing mutators, and so on – it opens up a whole new world of possibilities.

Modifiability: Though Unreal 1 was arguably the most "open" game of its time (since we shipped the editor around 50,000 lines of scripts – basically all the game code – with the product), it was still pretty hard to create and use mods, because of the overall complexity of the system, and forcing users to unzip files to subdirectories. With UT, the guys invented a new type of mod called a "mutator", which lets you create new game variations with very minimal programming. On top of that, we expose the complete user-interface and HUD framework to mods so they can be quite elaborate graphically; and mods are now managed by a "mod installer" which takes care of all the installation details, so users are only a few mouse clicks away from downloading and using a mod.

Dave "Fargo" Kosak: How do you feel about all the different licensees for the Unreal Engine? (Especially the non-game related projects?) What do you think of the different directions they've taken it?

Tim Sweeney: We're really happy with the different commercial and R&D directions our partners have taken with the engine. On the game side, there are a few leading-edge action games (Unreal, UT, Duke Forever), some crossover titles incorporating major RPG elements (Deus Ex, Wheel

of Time, X-Com Alliance), non-violent kids games (Nerf Arena Blast and Dr. Brain), and even a hunting game (TNN Outdoors Pro Hunter). On the R&D side, the Virtual Notre Dame project and the Unrealty ("real estate walkthroughs" of course), are pushing the technology in new directions.

When we started building the Unreal engine, a major goal was to make the technology applicable to other projects besides our own, but we had no idea how many widespread uses other teams would find for it. That has been a very pleasant surprise, and something we'll be even more focused on in the future.

Dave "Fargo" Kosak: What kind of LOD support does Unreal Tourney have for characters and models? What kind of support are you looking at in the future?

Tim Sweeney: Erik de Neve wrote the engine's continuous LOD code. It uses a topologically weighted edge-collapsing scheme which scales polygon counts down almost linearly with distance. It was inspired by the research on continuous LOD by many others, including Hugues Hoppe (<http://research.microsoft.com//hoppe/>).

Dave "Fargo" Kosak: Will you add bump-mapping support?

Tim Sweeney: This will be a feature of the next-generation engine. Regarding bump mapping, it's one of those features that you really need to plan for from day one in order to support it thoroughly.

Unfortunately, 3D card companies have been claiming to support bump mapping for years in their marketing hype, based on the fact that their current-generation chips support a grossly inaccurate approximation of the actual math...so most gamers don't appreciate the visual impact *real* bump-mapping will have when early 2001 hardware begins to support it. Stay tuned.

Dave "Fargo" Kosak: What are the plans for building curved surface support into the game and editor?

Tim Sweeney: Next generation engine! We could have hacked curved surface support into the current engine, but our philosophy towards those kinds of features (curved surfaces and bump-mapping) is to only support it when we can do so in a fully general way. Quake 3 Arena shows

that curved surfaces can look great, but there are many limitations on that approximation. For example, I expect to be able to build levels using CSG (constructive solid geometry) operations on everything; I want to have correct collision checking with all geometric primitives; I want to be able to apply decals to all geometry; I want all primitives to support dynamic, continuous level-of-detail. Tessellating bezier curves is easy; it's solving those other problems that is hard, and that will be one of the major focuses of the next engine.

Dave "Fargo" Kosak: Any plans to do an engine with real-time light sourcing?

Tim Sweeney: We've had it since 1997. :-) If you're asking about real-time shadowing, that's something we'll be exploring in great depth.

Dave "Fargo" Kosak: Skeletal animation?

Tim Sweeney: We already have skeletal animation up and running here in Epic MegaLabs(tm), though Unreal Tournament doesn't exploit it due to our development timing. We'll have the skeletal code in licensees' hands over the next few months, and it will be exploited in Unreal 2, being developed by Legend Entertainment.

Dave "Fargo" Kosak: Do you ever think you'll tinker with a voxel engine, or combining a voxel and a polygon engine?

Tim Sweeney: I don't think voxels are going to be applicable for a while. My thinking on the evolution of realtime computer graphics is as follows:

1999: Large triangles as rendering primitives, software T&L.

2000: Large triangles, with widespread use software-tesselated curved surfaces, limited hardware T&L.

2001: Small triangles, with hardware continuous tessellation of displacement-mapped surfaces, massive hardware T&L.

2002-3: Tiny triangles, full hardware tessellation of curved and displacement-mapped surfaces, limited hardware pixel shaders a la RenderMan.

2004-5: Hardware tessellation of everything down to anti-aliased sub-pixel triangles, fully general hardware pixel shaders. Though the per-

formance will be staggering, the pipeline is still fairly traditional at this point, with straightforward extensions for displacement map tessellation and pixel shading, which fit into the OpenGL/Direct3D schema in a clean and modular way.

2006-7: CPU's become so fast and powerful that 3D hardware will be only marginally beneficial for rendering relative to the limits of the human visual system, therefore 3D chips will likely be deemed a waste of silicon (and more expensive bus plumbing), so the world will transition back to software-driven rendering. And, at this point, there will be a new renaissance in non-traditional architectures such as voxel rendering and REYES-style microfacets, enabled by the generality of CPU's driving the rendering process. If this is a case, then the 3D hardware revolution sparked by 3dfx in 1997 will prove to only be a 10-year hiatus from the natural evolution of CPU-driven rendering.

Dave "Fargo" Kosak: Have you ever considered, or are you considering, *gasp!* putting the Unreal Engine aside and starting fresh with a new next-generation engine from scratch?

Tim Sweeney: We will do that to a certain extent with the next-generation engine. Anything that's not up to our standards of what a year 2001 game should look like, we'll rip out and replace. 100% of the rendering code will be new. Most of the scripting will be new. Other parts of the code, like the networking and core, are now in really good shape and won't be replaced unless our R&D reveals a significantly better approach.

Chapter 2

An Interview with Epic's Tim Sweeney

This interview was conducted by John McLean-Foreman for Gamasutra on Apr 06, 2001.

http://www.gamasutra.com/features/20010406/foreman_01.htm

Interview

Prologue

Gamasutra recently caught up with Epic's Tim Sweeney at the Game Developers Conference. Here, John McLean-Foreman talks with Sweeney about Unreal Tournament, his experience at GDC, and his favorite games.

Questions

John McLean-Foreman: Give us a quick background.

Tim Sweeney: I started at Epic about nine years ago with a little shareware game called Jill of the Jungle: a little platform game. For the next few years, I was mostly managing Epic and being the producer on projects, which wasn't much fun because I'm a programmer at heart. Around 1996, I started working on Unreal with James Schmalz and Cliff Bleszinski. That was in development for a long time; that was a three-and-a-half year project. We all worked on our first major project for the first time, and learned all the ropes on how to make a 3D game. We also learned how not to make a 3D game. (laughs) But in the end it all came together, and Unreal went on to sell really well. After Unreal, we did Unreal tournament, which shipped in 1999. We did the Unreal Tournament Playstation 2 version, and now we've been working on the next generation of the Unreal technology, and started to work on our next game.

John McLean-Foreman: How well do you think that Unreal has ported over to the PS2?

Tim Sweeney: To PS2 it did okay. With Unreal Tournament, we were aiming to make a launch title. Really, just get the game up and running and tweaked for the console, but not really take total advantage of everything that what possible, given the amount of time there was. So, it ended up being a good game, it's, I think, the number eight Playstation 2 game right now. But it's not the ultimate console first person shooter type of game. With our next project, we're really focusing on the console aspect of it from the very beginning, trying to make a good game. You know, a good PC game, a good Xbox game, as good a version on the Playstation 2. I think that kind of approach will, in the long term, work out a lot better because you can only do so much when you take a game totally designed for the PC and try to port it over. When you design it from the ground up, you have a lot more possibilities.

John McLean-Foreman: Will your new engine have any basis in the Unreal engine, or will it be something that we haven't seen before?

Tim Sweeney: It uses the existing Unreal technology as a starting point; we've pretty much redone all of the rendering. We have an all-new polygon pipeline for rendering extremely high detail, high polygon environments. So, we've been focusing a lot on, and taking advantage of the new hardware TnL (Transform and Lighting) cards, and the NV20 [GeForce3],

and all the new pixel shader and vertex shader effects that are possible. So, from a rendering point of view it's going to look all new and improved, but a lot of the tools are just incremental improvements. We spent a lot of time improving the Unreal Editor, but it's the same basic design. It's the same with the Unreal scripting language. Some parts of the engine are still very up to date and timely, so we're not going to spend another year redoing them when we don't have to.

John McLean-Foreman: One of the things that you've been quoted saying is: "One thing that people lose sight of when making games is that they forget to put the fun quality into it."

Tim Sweeney: Yeah, I guess that's pretty typical. That's a bad thing to do when you're trying to make a game whose whole point is to be fun.

John McLean-Foreman: Especially when you're programming code, how do you think in terms of fun?

Tim Sweeney: Well, it depends what you're working on. When you're programming graphics code you're not thinking of fun, you're thinking of fast frame rate and beautiful visuals. When you're writing gameplay code, when you're building levels, when you're doing all of that, you're thinking about the fun factor. So, different people on the team have different jobs and different focuses. With the Unreal technology, I've always been focused on the technology side of things. I mean, I play the game a lot to make sure I'm enjoying it and everything, but the graphics algorithms and design don't have much to do with fun.

John McLean-Foreman: How much does the Unreal community affect your decisions on how to alter the game?

Tim Sweeney: That's actually been a really huge factor with Unreal Tournament. With Unreal One, we started out with the design in a vacuum, and then to our surprise, all these websites started popping up: Unreal fan sites - interviewing us, showing screen shots, and message boards talking about the game. We started following them then, but it was really Unreal Tournament where we made the community our number one focus. So, we go on the message boards constantly, see what people are saying about adjusting games, seeing what people were thinking about our screenshots. We'd invite a lot of webmasters and random gamers from

all over the place over to our office to check out the game and give us some feedback. Starting with Unreal Tournament, the community has really been an integral part of our development process. If the community doesn't love the game, then we're screwed.

John McLean-Foreman: Do you do specific things that get them more involved in the whole process?

Tim Sweeney: Well, we're really a facilitator. We don't run Unreal websites, we don't go out and try to start things or anything, but as game developers, we put a lot of influences into creating good level editing tools, a good scripting language, and really making those things approachable by users. We make good documentation for Unreal websites, we make sure it's out there. Our job there is really just to get the tools into people's hands. If they don't pick up on it on their own then there's really nothing we can do to force them.

John McLean-Foreman: Have you ever thought of making an official tutorial for the Unreal Editor?

Tim Sweeney: We do that kind of stuff now and then, but really, the community has taken over that. There's hundreds of Unreal editing and scripting tutorials, and a few of them are really, really good and really detailed. We even point a lot of our new licensees to these community run web pages where they have tutorials on how to use the editor, for example. When we write documentation it's usually the pretty hardcore stuff, an overview of the Unreal Networking Subsystem for example, rather than tutorials. We do a lot of in person visits with Unreal technology licensees, showing them tutorials and stuff.

Everybody on the team has spent his time developing their stuff, whether they're a level designer or a programmer or whatever, part of that job includes documentation - but we don't have a full time writer or anything.

John McLean-Foreman: Do you foresee that you will get back to the storytelling aspects of Unreal or are you happy with the way that it's going with the multiplayer community?

Tim Sweeney: Well, I can't say a lot about our next game, but we're really trying to mind the best of both worlds. What we got right with Un-

real Tournament was keeping the single player and the multiplayer in the same style of gameplay. Okay, one has bots, the other has human opponents, but it's the same game. What we found extremely hard with Unreal One - we were trying to create a big story driven single player game with a completely different multiplayer. You know, with deathmatch maps and everything. We basically created two games, and that took as much time as creating any other two games. That kind of detracted from the whole project. It wasn't focused on one type of game or the other, it was doing both. We got out of that with Unreal Tournament. Unreal Tournament had very little story other than the one we made up at the last minute. In the future, what we want to do is keep the idea that the single player and multiplayer experience are the same, but really be able to integrate the detailed story with it, and give the player objectives rather than just: go shoot everybody on this level. You know, create a more in-depth type of game. Real-time strategy games have set a very good example there. You're getting the player involved with the story and not breaking the idea that single player and multiplayer are fundamentally the same kind of gameplay. We're not going to make a real-time strategy type of game, but we're going to learn that lesson from them.

John McLean-Foreman: There are a lot of games that have used your engine, some to greater effect than others. There is a fine line between what's going to make a story immersive and what is just going to be too much and/or boring. Have you learned that yet from watching people who have used your engine?

Tim Sweeney: We always err on the side of getting the player straight into the gameplay and not bothering too much with cinematics and things. We'll keep that kind of philosophy. One thing that I've never liked is in-game cinematics where you cut out of the game and go in to a movie, and the movie is prerendered, and it looks a lot better than the gameplay itself. That jars players and makes them realize: "Gee, the game's rendering just isn't that good." What we're starting to be able to do now, especially with the next generation technologies, is do any kind of cinematics in the game, not just stop gameplay and go to a cinematic, but integrate it into the story and into the flow of the game. Half-Life was the first game that really started down that path, but it can be taken a lot further. That's my view of the ideal: to not rip the player out of the

experience, but to make the story part of the gameplay.

John McLean-Foreman: Deus Ex uses the cutscenes within the Unreal engine, but there seems to be a random quality to it. If you went back to watch the same cutscene over again, it would be somewhat different than the first time.

Tim Sweeney: They did an amazing job with it. The game has a lot of different plot twists that have a significant effect on the end result. I don't know if anybody's seen the whole game because every time you play it you can do things differently and get different results. They did a great job with it. So, it depends on the type of game. You really want that with a roleplaying game, but some games, you tend to have more linear gameplay going through a set of levels. You know, whatever's appropriate.

John McLean-Foreman: When you are programming, what is the one thing that you try to keep foremost in your mind?

Tim Sweeney: (Laughs) The one thing? It's more of a balancing act between all of the different tradeoffs. There's always the performance aspect of your code: you've got to get it running fast enough for the gameplay to work well. There's also the simplicity issue. If you can write something in 10 lines of code, or 20 lines of code and do it slightly better, it's usually best to do it in 10 lines and just take the slight penalty for it because you end up with an enormous amount of code that you have to deal with every day. Productivity becomes the limiting factor in game development. No game doesn't ship because the artists were late finishing their work.

John McLean-Foreman: How do you best find the bugs in your code?

Tim Sweeney: You want to divide the code up into modules. There's a lot of reasons for that. One is it's easier to isolate problems. Another is it's easier to work on a project with multiple people. You can't have a bunch of programmers editing the same file - it would become a big mess where nobody remembers exactly what's happening. Our philosophy: we've always tried to keep the components of the engine clean and simple.

John McLean-Foreman: Does anyone help you program the core engine and the scripting language? Do you find it difficult to delegate?

Tim Sweeney: Well, with Unreal One, I was pretty hardcore about keeping all that to myself just because, at that point, we only had one other awesome programmer at Epic: Steve Polge doing all the AI and gameplay code, me doing all the engine stuff. That worked out okay for Unreal One, but with UT, we wanted to have a lot more features and do a lot more stuff. We didn't want to spend three-and-a-half years developing a game. So, since then we've really learned to work well with multiple programmers. Nowadays, we have six programmers working on the engine, and everybody has their piece of the engine. A lot of other people are helping out with various aspects of the rendering code and the gameplay code. Everything is really well divided up. It's not like a master programmer and a bunch of assistants. Everybody, in their own section of the code, they're the architect, the programmer and the bug fixer.

John McLean-Foreman: Are you taking any tutorials at the GDC this year?

Tim Sweeney: (Laughs) I haven't had any time to go into the tutorials. Mostly I'm here, meeting with guys, showing the technology, talking about stuff, doing the occasional interview and seminar here and there.

John McLean-Foreman: Would you find anything helpful by going to the seminars?

Tim Sweeney: Yeah, there are a number of seminars that I'd really be interested in going to. There are some incredibly smart people here, but there's also a lot of noise in proportion to that signal. There's definitely a lot of good stuff to learn here. If you go down to a presentation, like, there was one on rendering multiple resolution meshes, you know, level of detail - Stan Melax spent the last few years working on that. That's way more focused than any of us could afford to put into any particular task. That kind of knowledge sharing is very useful. Then there's some marketing guys talking about "how to make your game sell a million copies," and that's a waste of time. (laughs) Like any marketing guy knows how to do that.

John McLean-Foreman: Do you think that there is any accuracy to the predictions that the PC is going to die out?

Tim Sweeney: Well, I see the PC games business increasing and not de-

creasing nowadays. You used to have this fact of life in the PC market, probably 95% of games don't make a profit - they lose money, and probably 80% of them just completely bomb, and are gigantic losses for the publishers. Because you see a lot of crappy games coming out, a lot of clones, where somebody identifies a great game like Command and Conquer, and then tries to make a crappy clone with an inexperienced team on a quarter of the budget. Of course it sucks and nobody buys it. Then those publishers are the same guys who are saying, "Oh! PC gaming is dead!" If you look at the guys who are creating the really good games, you can reliably ship good games and have them sell well as long as you're smart about it, and you're realistic about your strengths and limitations.

John McLean-Foreman: Apart from the fact that an inexperienced team worked on them, why do you think that the majority of the clones are terrible? Do you think that there are other aspects to it that makes them bad?

Tim Sweeney: Cloning games fundamentally works sometimes. If you look at any major genre, there's always at least two games - you could even say that Unreal One was an attempt to make a Quake clone. The game was very successful on its own because it wasn't just a clone, it added a lot of new features to the genre, some really nice new visual effects, a lot of new editing tools that the community picked up on. It was adding a lot of things. You see a lot of clone games come out now that don't add anything to the genre, and just subtract by not having as good artwork, and not having as good gameplay, and being rushed out to market. The PC market has always been sick like that.

If every publisher was required by law to drop 80% of their games and only focus on the remaining 20%, I think that you'd see a lot more games coming out. I think that you'd see the industry sales increase overall even with far fewer games coming out because those games that do get completed would be a lot better. But, you know, publishers don't do it that way. They're always out for the quick buck. Every publisher has at least one major success, and they say, "Oh wow! Look at all this money we made from this one game. Now, if we make five more like it, we'll be making five times more money!" That's the big thing about the game business: it's not scalable at all. Development teams aren't scalable. A team that's created one good game isn't going to be able to split into two teams

and create two good games. They're probably going to be able to create two mediocre games, or something like that. It's this lack of scalability that nobody seems to realize and recognize formally. It's very weird because companies like Id Software have always been focused on staying small, and making a good game, and have been able to do well repeatedly, game after game after game. It seems like a lot of companies don't look around and notice that. They just think: "We can be more profitable by becoming bigger." Just the opposite happens.

John McLean-Foreman: How do you put the right teams together? How do you find the people that you really want?

Tim Sweeney: We hire people very slowly. Once in a while we look for somebody for a specific position, but often it takes six months before we find the right guy. Most of the people we hire aren't "Okay, we need a level designer. Let's post 'help wanted' ads, and look at the level designer resumes that we get." Mostly it's waiting around for six months, or eight months, and looking at all the level designers releasing stuff on the Internet, and picking the best one. When we see someone who is clearly way better than everybody else, we go after them really aggressively. I wouldn't say it's easy, but it's possible to hire extremely high quality people as long as you don't try to grow really fast, and as long as you're a fun, attractive company to work at. At Epic, we can pay people really well now, but even when we didn't have a successful game under our belt, and we weren't paying huge salaries or anything, we were able to attract really good people. We have a good team to work with, really well balanced developers, and the game is developed by developers - it's not controlled or dictated by marketing people or business people who are out of touch. Developers like that.

John McLean-Foreman: Do you think that the best games are always going to come from the smaller companies that aren't controlled by the publishers?

Tim Sweeney: At least in the United States those are really the only teams who are able to create great games time after time again. It's either by having a small team that's focused and does a great job, or by having a huge team and just investing an enormous amount of money into development. If you have an 80 person team to develop a game and it's

not a success, then you're guaranteed to lose money. Whereas with a 20 person team, if our game sells half of what we expect, then we're still not losing money on it, we're just not doing as well as we expected.

John McLean-Foreman: By making your engine so accessible to the public, do you feel that that you're increasing the amount of people who play your game or are you creating your own competition?

Tim Sweeney: If we're creating our own competition Great! We can only develop one game at a time, and that takes at least 18 months. Whereas when we work with other partners who are doing great stuff with our engine, you know, creating games like Deus Ex or Undying, those games we can really be proud of. Even though we don't develop them, we contributed to them, and we're making some money from those projects. I think that's a good way for a company like us, you know, a small focused team, to be able to do more stuff without growing big. There's always the ever-present threat, some new team that nobody's ever heard of, who's going to come along and make a game that blows us away. I think we always live in fear of that kind of scenario. We've been able to keep up so far, but we work really hard. It's not like we're resting on our laurels. We're all working the same 60-80 hours a week that we were working during Unreal One.

John McLean-Foreman: How do you come up with something new and innovative when so many people are creating first person shooters?

Tim Sweeney: It's kind of a random thing. You always have to keep watch on what's possible with the technology, and graphics cards, and CPUs and everything. Every once in a while there is a major, fundamental change where something that was completely impractical before becomes possible. One was the switch to hardware rendering. That was a pretty obvious thing back then because everybody had so much notice. Now there's other changes too. With pixel shaders and vertex shaders, you can start doing completely accurate lining on all of your objects, and you have a whole bunch of new tradeoffs there. Most of that comes from just being flexible and being able to reinvent all of your assumptions about how you develop games every once in a while. Nowadays we're creating levels that are more than 100 times the polygon counts of Unreal Tournament levels. Our whole way of building objects and using tools has

completely changed. You can no longer go through and build every little wall and floor in your entire game from scratch. You have to build big archives of different pieces of architecture, you know, doorframes, doorways, and spires, and things like that - build libraries and share them.

We're always looking at our productivity and seeing where are our bottlenecks and how can we fix them? So many companies are the master of one generation of technology or consoles or whatever. When the next generation comes along they just don't bother learning it, and they think that they can keep doing things the way they always did. It's easy to be the rebel when you're small because we can completely change direction without any big ramifications. Whenever we're ready, if we want to do something different, we can. If we wanted to make a game that's totally not a first person action game, no problem, we can do that. Six years ago, we were doing Jazz Jackrabbit, and Epic Pinball, and these 2D games. Now we're doing Unreal. What will we be doing six years from now? Well, who knows? Maybe it will be more Unreal style games, or maybe it will be completely different. I don't think that we've pigeonholed ourselves as one particular kind of game developer.

John McLean-Foreman: If you could make your perfect game, what would it be?

Tim Sweeney: It's hard to say nowadays. I think that we're doing a pretty good job taking advantage of what the technology makes possible. Long term, there are some cool possibilities. There's this big trend towards massive multiplayer games. Of course now there's 60 teams developing massive multiplayer games, and three of them will be successful, you know, the cloning effect. The big problem with them is they look a lot like America Online, and Compuserve, and Prodigy looked like six years ago. They're totally closed systems run by some, you know sorry, the "coolest" managers. They don't go in, and play the game, and talk to customers, and make sure they're keeping them happy. What seems really neat is to kind of take the Internet approach to that. You know, we're not going to have one central bureaucracy in charge of the thing, it's just going to be randomly distributed, and it's not going to work perfectly, but it's going to make up for it by giving everybody control to do their own thing and innovate with it. So, do a distributed massive multiplayer game where anybody can run a server, and anybody can add content to the game. I have

no idea how to make that work from a gameplay point of view, but that kind of stuff sounds interesting. I don't think that closed system model that you see going with Everquest and Ultima Online, I don't think that's sustainable. I don't think those games will exist ten years from now. How are we going to make the transition? Who knows? Games, and programs, and information want to be inherently open and modifiable by users. So, you've got to get to a different system. It's hard to do also because, even first person shooters, there's not a big incentive to cheat because you just get some more frags maybe. Whereas with a massive multiplayer game, there's huge incentive to cheat. If you could cheat, you could be the most powerful person in the game, and have these huge advantages. So, you have to deal with these issues like "how to avoid cheating", and things like that. I think that we'll get there at some point. If you've seen the Freenet Project, basically, it's like Napster but with everything encrypted to the point where servers don't know what kind of information that they're sending back and forth through the network. Everybody has end to end communication, but with complete secrecy and complete lack of ability to hack the system. That kind of stuff could be applied to gaming at some point. That would be cool. I'd love to do something like that someday, but the chances are 20% that it will happen or 80% something else will happen in its place. Definitely games will get better.

John McLean-Foreman: What games do you play apart from Unreal?

Tim Sweeney: I haven't played a lot recently; I played Undying most recently. That was very cool: very scary. Last game that really scared me when I was playing it, you know, go into your room, turn out all the lights, play the game, that was Doom, and that was very long ago. I guess some people say Unreal is scary, but when you're developing it and playing it every day, you just don't experience that. Undying really got that point across with the realistic shadows and the scary sequences.

John McLean-Foreman: It must be very gratifying to have put such hard work into something and have it become such a phenomenally popular product.

Tim Sweeney: I guess. Some of the team members feel that way. Everything that I've done in the past, it doesn't matter because I'm working on something different now, and it's either going to be a big success or a big

failure, and I don't want to screw it up. I really never have looked back. I was playing Unreal Tournament really heavily online for the first month of it shipping, and then I stopped. A lot of the guys still play it, but especially as the technology guy, I always have to be looking way ahead of the current thing towards the next thing.

John McLean-Foreman: Where do you get your inspiration?

Tim Sweeney: It's a lot of random influences. If you go for a walk in the woods, and you look around, and you see the trees and the amount of detail there, you realize that we're still a factor of a hundred short of being able to render those kinds of scenes in real-time. So, looking at things that way, looking at real life and saying: "Okay, how far away are games from this kind of thing in terms of rendering, and behaviour, and realistic scenes, realistic lighting?" That's really interesting, to just look at that and figure out: "Okay, how can I get a rough approximation of this into the game that runs fast enough?" We play a lot of other games; most of the guys on the team are really hardcore gamers. I play other games, but I'm not that hardcore. You're much more likely to find me in the visual C++.

With Unreal One, I was in Canada, it was freezing cold, and I was walking along, and there was a heavy haze everywhere. There were these street-lights with big globes of fog illuminated around them. I was saying, "Hey! I could program that!" So, we had diametric fog as one of the big features of the Unreal One engine.

Chapter 3

Next gen engines: Tim Sweeney interview

This interview was conducted by Tolstiy for Tolstiy on Nov 21, 2001.

<http://web.archive.org/web/20011121083307/http://www.tolstiy.ag.ru/what/sweeney.htm>

Interview

Tolstiy: Hi Tim! Please, tell us a bit about yourself and your work at Epic Games.

Tim Sweeney: I've been programming since I was 13. I founded Epic Games in 1991, and wrote several small shareware games, ZZT and Jill of the Jungle, before starting on the Unreal engine in 1996.

Tolstiy: Lets start from the past. First of all, tell us about your first engine you've ever created? And, don't forget to share some details with us, such as how did it pop up in your mind to create an engine, how did you do that, did anyone help you etc?

Tim Sweeney: Every game I've written has been an engine to a certain extent. My first shareware game was a simple text-mode action game,

but it included a built-in editor and encouraged users to create their own levels. I released ZZT in 1991. It was very low-tech, and wasn't a big commercial success, but thousands of people have created levels for it and released them on the Internet. For example, see www.zzt.org – some kids are STILL building levels for it!

The Unreal engine inherited a lot more ideas from that game than you might think. The editor, the focus on making the entire game user-modifiable, the object-oriented design, these are all things I was doing 10 years ago. What's new is the technology, the 3D graphics, Internet play, and so on.

Tolstiy: When did you understand that the time to move to full 3D has come? What were your first thoughts on this subject?

Tim Sweeney: The first time I saw DOOM, I believe it was a pirate alpha version, around 1993, it was a huge leap over all past 3D graphics. Before that point, 3D games were fun but unrealistic. They were mostly flat-shaded, or boxy, and poorly lit. DOOM made it clear that 3D was the future. Of course, going from realizing "this is the future", and learning to program something competitive, took a long time.

Tolstiy: As I understand, you were the one who worked tightly with Unreal engine, you wrote it (was writing it for a long time, but the result was excellent - no doubt!). What were those corner stones that you have stumbled across, as a young developer who is only beginning his long way into full 3D? This question may be considered as a manual for those who are going to write their own engines for their games.

Tim Sweeney: With Unreal, the hardest aspect of the engine was the renderer. It's the component I rewrote 7 times over 18 months before coming up with something I was happy with. Less glamorous is the overall framework of the engine, the code that holds everything together. The hardest thing for any new programmer is to build a large piece of software without having it collapse because of its size and complexity. Unreal has held up pretty well, but I'd been programming for 10 years by the time I started writing Unreal.

Tolstiy: When you were working on the UT engine, what were you primary aims - what did you see in the end of your work?

Tim Sweeney: Our goal with Unreal Tournament was to focus on the gameplay, to make a better and more polished game. So, the technology didn't change much compared to the earlier game Unreal 1. The network code was better; level-of-detail support was added to the mesh rendering code. Most of the effort went into making the game better.

Tolstiy: Any technology growth leads to more troubles while working with tools built for these or that technology. As the result, this may cause many independent gamers who like to make mods and custom maps to give up saying - "Too hard to master!" Can you comment somehow on this one?

Tim Sweeney: I think mod makers will find the next-generation Unreal engine easier to work with than any past 3D engine, Unreal, Quake, or the original Half-Life engine included. The editor is easier and more stable, but also more powerful; the UnrealScript game code is cleaner; and we have far more documentation for the next-generation engine than ever in the past. I expect mod makers will find it very easy to create levels and simple mods.

However, with very high-quality mods like CounterStrike (Half-Life) and TacOps (Unreal Tournament) available, it's becoming harder and harder to make a competitive mods. The best mods have dedicated teams of 10-20 people contributing. So, even if it's relatively easier to make mods with the new engine, there's now a lot more competition, and users' expectations are higher.

Tolstiy: By the way, your engines are usually used in shooters. And what other games you'd like to see basing on Epic's technology? May be strategies and quests? Is there any sense in such statement?

Tim Sweeney: Well, the Unreal engine was used for Deus Ex, an award-winning role-playing game from Ion Storm Austin, and for Digital Extremes' Adventure Pinball, a 3D pinball game. So, I'm not sure I'd say it's a shooter engine! Especially now that the next-generation engine supports 100X higher polygon counts than Unreal Tournament, and has much more powerful tools for getting 3D models and content into the engine, I think you'll see lots of non-shooters coming out. You could definitely use the new Unreal engine as a starting point for a realtime strategy game, an offroad racing game, a space combat game, or a massive

multiplayer game.

Tolstiy: What is your opinion about the companies who license technologies and don't write from scratch their own engines? How good or bad is that for the final game look, in your opinion?

Tim Sweeney: Licensing an engine enables these companies to focus 100% on game design and gameplay, instead of reinventing the technology. In the hands of a talented team, it's a very powerful tool. But, it's just a tool. If a team isn't very good with gameplay or game design, then licensing an engine isn't necessarily going to make their game any better.

Tolstiy: How do you appraise the technology growth we all can see - is the pure Good or maybe there is something Evil in this?

Tim Sweeney: Moore's law is a powerful concept. Computers are perhaps a thousand times more powerful than they were 15 years ago. Eventually they'll be smarter than we are. What then?

Tolstiy: A question also connected with technology growth. How do you think, is it possible that some day such characteristic of video cards as 3D will disappear, like it was with 2D characteristic of video cards several years ago? Maybe developers will choose to make games, which will have software render, like it was with Unreal - is it really possible? What may help to achieve that?

Tim Sweeney: This is a very good point. You can really think of a graphics card as being a massively parallel but only somewhat programmable CPU, having around a 20X performance advantage for 3D rendering, but a significant disadvantage for computation. In the long-term, it's reasonable to expect that CPU's will become more parallel (using techniques like symmetric multithreading, or just brute-force single-chip multiprocessing), and GPU's will become more programmable. Thus they will become more similar over time.

At some point in the future, we'll begin to question the value of having two separate and expensive chips in computers, the CPU and the graphics chip. Then what, will NVidia be trying to extend their pixel shaders and vertex shaders to achieve x86 compatibility? Will Intel be encouraging developers to go back to software rendering? These two industries

are in for an interesting collision course in 5-10 years.

Tolstiy: Are you happy with the niche that EpicGames now has, looking from the standpoint of companies, which create great technologies and games?

Tim Sweeney: Yes, very happy with it. By creating one major game at a time, while working with a small number of partners who use our technology, we're able to remain a small, focused, and profitable company.

Tolstiy: How do you see the future of your company - are you going to stay true to PC games market, creating stunning games or you may switch to consoles, like many developers do these days (I can't imagine how one can play an FPS game with a joypad!!!)

Chapter 4

Tim Sweeney & CliffyB Interview

This interview was conducted by PlanetUnreal for PlanetUnreal on Oct 22, 2002.

<http://www.planetunreal.com/features/ut2interview/>

Interview

Prologue

Now that the hype surrounding the release of Unreal Tournament 2003 has subsided a little bit, we recently decided to ask Epic Games Lead Programmer Tim Sweeney and Lead Designer Cliff "CliffyB" Bleszinski a bunch of questions. Thankfully, Tim and Cliff graciously provided a bunch of answers. And here they are!

Questions

4.1 The Game

PlanetUnreal: UT2003 began life as Unreal Championship at Digital Extremes, right? Is it correct to say that UT2003 is almost a port of Unreal Championship, rather than the other way around?

Tim Sweeney: Tim Sweeney: Way back in 1998, we started on a multiplayer bonus pack for Unreal 1, which was initially going to be a free release, then a level pack, and then it turned into Unreal Tournament, a stand-alone game that way exceeded our expectations in terms of sales and the cool stuff the community did with it. So changing course during development is something we're quite accustomed to.

As we moved to the PC, we and Digital Extremes redesigned the pieces of the game that didn't make sense in a competitive PC first-person shooter. The basic weapon designs, much of the art, and some of the levels were carried forward, but updated and tweaked. On the PC, the player movement speed is faster because the mouse gives you quicker and more precise control. On the Xbox, there is some weapon auto-aiming help. There are hundreds of little details like this where UT2003 was tweaked as a PC game, and UC is being tweaked as an Xbox game.

PlanetUnreal: Can you briefly explain why UT2003 shipped nearly half a year later than originally planned? What caused all the delays?

Tim Sweeney: That's how much longer it took to get the game to the point where we were really happy with it. :-)

CliffyB: We're really, really poor at predicting dates. This is something that we're continuing to work on and improve upon. It is important for a publisher to have a solid street date for a game in order to ramp up marketing and press awareness.

PlanetUnreal: What was the most difficult part of the UT2003 development process?

Tim Sweeney: A while back, we realized that developing both UC and

UT2003 simultaneously wasn't optimal for either game and split the projects apart, with Digital Extremes focusing 100% on UC, and Epic focusing 100% on UT2003. That was a big change for us here at Epic, switching gears from early development work on an unannounced project, and going into crunch mode to bring UT2003 together. That was tough, but it was definitely the right thing to do, and both games are better for it.

CliffyB: Aw, this is like asking a mother to choose her favorite children!

I'd have to say making the transition from simple 200 triangle environments to environments nearly 100 times the detail was a pretty Herculean task. Finding the balance between artistic clutter and smooth gameflow is an ongoing thing that we, and many other developers, are struggling with.

PlanetUnreal: What was the most valuable piece of feedback you received from the demo?

Tim Sweeney: Cliff and Steve Polge made a number of gameplay tweaks based on demo feedback, but that wasn't as clear-cut as you might expect. Everyone went through many thousands of message board comments, and tried to distill some consensus from them. There were perhaps 100 posts saying the minigun was too strong, and another 100 saying it was too weak, for example. The joke is that you're probably hitting the sweet spot when half the community complains that the weapon is too strong, and the other half complains that it's too weak! So we zeroed in on the feedback where a clear pattern emerged, for one example the requests to make the shock rifle alt-fire faster.

On some topics, we actually decided to go against the consensus when the team felt that peoples' initial impression of the demo would change as they gained experience. The classic example is the rocket launcher, which tons of people felt was too powerful, but as more people learn to double-jump, wall-dodge, and so on, it will become less powerful. This remains controversial; we'll be watching how the feedback on these issues changes over time and see whether this turned out to be the right decision. One thing we've learned is to never expect to see much of a community-wide consensus – everyone has their favorite weapons and their despised opposing tactics.

The technical feedback and bug reports were quite clear and did a lot of good. UT2003 is the first game we've shipped with the new bug reporting code Jack Porter wrote: whenever the game crashes, you have the option of submitting an automatically-generated crash report giving us information on what went wrong. This enabled us to track down a lot of bugs that would have been very difficult or impossible if the user were just describing the symptoms in an email.

The one area that still needs work is hardware compatibility. Like other games that have really pushed the hardware more than past games, if you have a motherboard problem, bad RAM, a heating problem, an over-clocked PC, or an older graphics or sound driver, UT2003 is much more likely to surface the problem than previous games like UT or Quake 3. So one of the big things Dan Vogel has been doing lately is talking with users who have these problems, and compiling a list of fixes, such as downloading new drivers, or in some cases even increasing the cooling of your computer case, working with guys like ATI and VIA to track down various compatibility and stability issues, and working around issues in our code where possible.

CliffyB: There were some good weapon tweaks that the community wanted that we acted on quickly. It is hard balancing this, however, because a weapon that you may be good with in the first, second, and third week of playing a game may not be the weapon that you ultimately wind up best with. For instance, your average gamer would pick up the game and gravitate towards the rocket launcher. However, once aim improves, the gamer may find his tastes shifting more towards, say, the lightning gun.

The other issue that we wrestle with in regards to weapon balance is that some weapons are more useful at lower pings whereas others are more useful at higher pings. There have been some grumblings about the power of the mingun, but our stance on that weapon has always been that, in the hands of a really skilled player that gun is a good, solid weapon. In the hands of your average player it is more of a "finishing" weapon. Tag someone once with splash damage from a rocket, or hit their body with the lightning gun and then quickly switch to the mini to finish them off.

PlanetUnreal: What aspect of UT2003 are you most proud of?

CliffyB: I'm proud of the fact that all of the game types in this game are enjoyable both offline and online. In the original UT, as cool as Assault and Domination were they never seemed to catch on. I really hope people give Bombing Run and Domination 2.0 a good, honest chance. They won't be let down. Digital Extremes really made great up-front choices with the game modes in the game.

PlanetUnreal: What UT2003 map is your personal favorite?

CliffyB: I'd have to say it is a tossup between BR-Skyline and DM-Antalus. The ways that skilled players move around in Antalus is nothing short of jaw dropping!

PlanetUnreal: Favorite mutator?

Tim Sweeney: Quad jump. It doesn't have as much impact on the gameplay as the others, but I really like things that give me more control over movement. The whole multiple-jump thing is one of those addictive features that, when you go back and play the original UT or Quake 3, you get frustrated that you can't do it, just like you long for mouselooking when you go back and play DOOM 1.

CliffyB: Quadjump baby! Once you try it, you'll never go back.

4.2 The Engine

PlanetUnreal: If you had to pick one thing the UT2K3 Engine does better than any other engine out there, what would it be?

Tim Sweeney: The Unreal engine has always been just about the opposite of a "does one thing better than the others" engine. Nowadays we have support for very high polygon-count geometry in the form of meshes and terrain; very realistic physics using the Karma physics engine for ragdoll player collision, vehicles (see the little vehicletest map in UT2003), and arbitrary rigid body dynamics; the Matinee cinematic editing tool (used in creating the UT2003 single-player intro), particle systems, improved networking, export plug-ins for 3D Studio Max and Maya, great mod support, a huge documentation site (see <http://udn.epicgames.com>),

...

The modern expectation of an engine is that it provides nearly all of the technology that relevant games will need, so in business terms I don't think there's much of a market for an engine that does one single thing much better than everyone else. It's much more a matter of providing a good, consistent, and complete framework.

PlanetUnreal: How different is editing for UT2003 compared to editing for UT?

Tim Sweeney: Well, everything you could do in UT, you can also do in UT2003, so the community should be able to get started with it very quickly. However, taking advantage of most of the cool improvements that have come since the original UT requires a significant change in workflow.

In UT, the only primitive you could build real world geometry out of was the BSP brush. In UT2003, there are also static meshes – which are rendered with far higher triangle throughput, and are instanced so they can be far more detailed, and terrain – for creating large outdoor areas. Creating great looking static meshes requires a modelling program like Maya or Max – which is why we prioritized shipping Maya Personal Learning Edition with UT2003 so highly. Some level designers will be good at this, but for those who aren't, we provide a huge set of prefabs, all of the static meshes in the shipping UT2003 levels, which you can use as building blocks.

These new features make the ideal pipeline for building mods in UT2003 quite different from that in UT. You benefit from more of a division of labor, where the modellers who are the most artistic create lots of static meshes, and those who are more gameplay-focused build levels using those static meshes as building blocks, as well as terrain and BSP. With this approach, the end result can be far better than the original UT visually, but doing highly custom-looking levels requires more effort than in the past.

PlanetUnreal: Epic has released several editing utilities in addition to UnrealED (including UPaint, the Unreal Development Environment, and others). Are these tools that Epic used in-house, or were they created

expressly for the user community?

Tim Sweeney: These are provided by our partners and are geared towards the community. For example, Maya costs several thousand dollars, and that's actually a great value if you're a pro developer, but way out of the price range of most enthusiasts. So Maya Personal Learning Edition ships on the CD for free, and provides nearly all the functionality of the full version, but only exports to UT2003.

PlanetUnreal: What modeling/animation packages are used at Epic?

Tim Sweeney: We use both Max and Maya. Digital Extremes uses those and Lightwave as well. Various guys have other modelling programs and utilities, but those are the major ones.

PlanetUnreal: How difficult is it to integrate static-mesh prefabs into the map?

Tim Sweeney: It's pretty straightforward once you're familiar with the static meshes that are available, and learn how they can fit together with each other and with nearby BSP geometry.

CliffyB: It is certainly a different way of thinking. For the bonus pack we're trying some variations on this approach, for instance, the other day I did just a simple BSP layout in a map and handed it over to an artist to fill in "the pretty." Still, making sure there's enough elbow room so that when additional detail comes in is a tricky thing. Curse 3 sure was tight. :-)

You have several ways of going about this if you're a designer. You can build the BSP first and then retro-fit meshes in the level, either by using existing ones or by building your own in max or maya. (there's a map t3d importer for max floating around somewhere...) Another way to build the level is to compose the static meshes first and to then build the simple BSP around them.

Even then, if you've got lots of time on your hands you can move towards building large sections custom in max/maya and splitting them up before import, importing, and tweaking from there.

PlanetUnreal: It appears that many of the UT2K3 maps are comprised

almost entirely of prefabs (such as DM-Gael), with BSP brushes used only to hollow out the basic shape of the level. Other levels (like BR-TwinTombs) use a great deal more BSP surfaces. Are there times when one form of architectural creation is preferable to the other, or is it more left up to designer discretion and preference?

Tim Sweeney: It's up to the designer. On one extreme like CTF-Magma, Shane Caudle created the entire map from scratch, with almost everything being custom. The benefit of this approach is that you can get the absolute best visual quality possible, but the drawback is that it's very time-consuming and doesn't result in many prefabs that other level designers can reuse. The majority of the maps are a combination of unique BSP and terrain, with prefabs placed throughout.

CliffyB: I'd have to say this is up to the designer, his preference, and his pipeline. If you're just a guy in your basement building stuff then build whatever works for you. If you work at a game company and you're a licensee you may have a pipeline that you have to adhere to.

4.3 The Community

PlanetUnreal: Epic has already shown a dedication to supporting the mod community with tools and tutorials. Are there future plans for continued support, such as continued additions to the UDN, or further tool creation/releases?

Tim Sweeney: Oh yes! Expanding the tutorials on UDN is the top priority. Besides that, we'll be tweaking the tools with each patch. The current tool set is remarkably complete (UnrealEd for level design, now also useful for cinematic creation, UnrealScript for programming, Maya PLE for high-poly modelling and characters, UPaint for art and skinning, KAT for physics setup), so in the next year or so the focus will be on polish rather than shaking things up too severely.

PlanetUnreal: What types of user-created mods would you like to see?

Tim Sweeney: I won't mention the obvious well-known categories like tactical mods, and focus on unique things I'd love to see:

Vehicle mods. The Karma physics support in the engine now is incredibly complete and powerful, but all we had time to put in was the little vehicletest map with a hastily-constructed vehicle. With Karma, you can do arbitrarily complex vehicles with independent suspensions like tanks, buggies, tractor-trailers, go-karts, bulldozers, etc.

An RTS mod. You could have an above-the-world-looking-down view zoomed out similarly to recent 3D RTS games, having both vehicles and foot soldiers. Gameplay could give the player control over how much detail he wants to get into...you could either give orders to your troops, or go into first-person view and micromanage them.

Also, there is going to be a very, very big mod contest for UT2003, which won't be announced for another few weeks, but the scope is unprecedented, and there will be lots of opportunities for community recognition, for projects of all sizes and complexities.

CliffyB: Be creative and move the camera already! I want more top down/isometric mods, if you don't have to worry about the horizon line you have less to worry about in regards to overdraw and triangle count!

PlanetUnreal: What is the number one mistake you think amateur mod developers make?

Tim Sweeney: To make a great mod, you need a small team of really talented core contributors who do the bulk of the work, and work together closely, and perhaps a few others who make valuable but less frequent contributions. You don't need to recruit a ton of people. Many mod teams seem to be unrealistically large, with the majority of members not being core contributors, making it hard to focus. Stay lean and don't be afraid to get rid of dead weight.

Also, it would benefit the best mod teams to be more aggressive about recruiting the most talented guys from other teams and the community in general (not just the Unreal community; there are a lot of mod teams still working with Quake 3 and Half-Life 1, which as 1999 games aren't growing any younger). There are various teams out there with incredibly talented members, who just aren't shipping very good mods, due to whatever internal management problems. For productive and efficient teams to thrive, they need to go after these kinds of talented but under-

utilized individuals.

That may all sound Machiavellian, but many mod teams will have the opportunity to transform into pro game development studios over the coming few years, and it makes sense to run the more serious mod teams like businesses, even before they're earning money at it or pursuing business deals.

CliffyB: In mod groups, I'd say the biggest mistake is hooking up with the wrong people.

PlanetUnreal: Do you think the Unreal fan community sometimes takes things too seriously?

CliffyB: Nah. They're fans of our games and while it may seem frustrating sometimes we understand that they're passionate and their hearts are always in the right place.

4.4 The Future

PlanetUnreal: What do you see as the future of the competitive first-person genre?

Tim Sweeney: I don't buy that these are a genre anymore. Among first-person shooters, there are now single-player focused games like Unreal 2 and (I assume) Half-Life 2, tactical-focused games like Counterstrike and Battlefield 1942, and the more traditional Deathmatch/CTF style genre pioneered by Quake 3 and Unreal Tournament. This looks like at least three genres, with the lines between them still somewhat blurred, such as with BF1942 or Medal of Honor.

With UT2003, we made a conscious effort to steer the game a bit in the direction of a sport, with DE's new Bombing Run game type, which is inspired somewhat by American football, and with other things like the UT2003 single-player intro cinematic, which is somewhat WWF-inspired, and the new worldwide stats tracking (<http://ut2003stats.epicgames.com>). In some areas, this probably went a bit too far. Ok, I've admitted it, so can everyone please stop sending us flames regarding the announcer voice?

:-) In other areas, like Bombing Run, the new style of play is really catching on.

PlanetUnreal: What kind of content can we expect from the bonus packs?

Tim Sweeney: More of it! :-)

CliffyB: "More." This bonus pack currently has over 170 megs of new art content. We're compiling much of the test artwork that was done to figure out our pipeline over the past year and giving it out to everyone. People are going to go crazy with this content, and the maps we ship with the pack will be top notch!

PlanetUnreal: It seems as though UT2003 is a step in the direction of successful sports franchises such as the Madden series, which indicates yearly installments. Will there be a UT2004?

CliffyB: Undecided at this point.

PlanetUnreal: Okay... spill it: What's the deal with Unreal Warfare? ;)

CliffyB: I can honestly say that there is no "Unreal Warfare" at this time. :-)

Chapter 5

Tim Sweeney 64-bit Interview

This interview was conducted by Brandon "Sandman" Bell for Firing Squad on Apr 22, 2003.

http://www.firingsquad.com/features/sweeney_interview/

Interview

Prologue

Epics Unreal and Unreal Tournament game engines are used extensively in todays latest games. Therefore, when AMD <http://firingsquad.gamers.com/news/newsarticle.asp?searchid=4569> running on its Athlon 64 processor at Comdex last year, the entire gaming world took notice. For further insight into Epics 64-bit plans and their thoughts on AMDs 64-bit approach, we turned to Epic lead programmer Tim Sweeney

Questions

Brandon "Sandman" Bell: Can you describe the process involved in migrating to AMD's 64-bit architecture? Has the transition been a difficult one?

Tim Sweeney: Since our code is pure C++ and already ran on 32-bit Windows and Linux, the only work required was to make the code 64-bit safe. No Hammer-specific work was necessary to get the port up and running; what we did for Hammer is the same thing that would be needed to run on 64-bit PowerPC or 64-bit Itanium.

In the case of the Unreal codebase, about 99.9% of the code was already 64-bit safe and didn't need touching. Of course, with a million-line codebase, the remaining 0.1% left a hundred or so places in the code that needed updating because of assumptions we made years ago before we'd thought about 64-bit. It was a relatively straightforward process, and took Ryan Gordon about 10 days of hard work.

Brandon "Sandman" Bell: Will you be adding any special features to 64-bit Unreal Tournament 2003 in particular?

Tim Sweeney: No. Our goal in porting UT2003 to 64-bit was to show that it can be done without much work, that the platform is stable, and that it's ready for gaming. We're not doing anything that really takes advantage of over 4 gigs of RAM or the large virtual address space.

The next generation of the Unreal engine is where we'll be taking major advantage of 64-bit in a very visible way, in the 2005 timeframe. We expect to support 32-bit and 64-bit clients and servers for gameplay, but might require 64-bit for content creation, because of the significant requirements of our new content development tools.

Because of our new content tools, we're already feeling a very strong need for 64-bit internally right now, and by year's end I expect we'll look at 64-bit as something that we couldn't possibly do our jobs without. We expect this sentiment to carry over to other game developers in the next 12 months, to high-end consumers over the next 24 months, and the wide mainstream all the way down to the lowest end of the market within 36

months.

Brandon "Sandman" Bell: What kind of performance benefits are you seeing with the 64-bit codebase and AMD's 64-bit processor in general? Are there any additional advantages to 64-bit?

Tim Sweeney: The extra registers can give you a significant performance gain in loops which currently have to spill over into memory. The on-die memory controller reduces the path for uncached reads significantly, which is a huge win for applications like Unreal which are memory-limited. On the other hand, pointers grow from 4 bytes to 8, so traversal of linked data structures takes an additional bandwidth hit. I think you'll probably see a clock-for-clock improvement over Athlon XP of around 30% in applications like Unreal that do a significant amount of varied computational work over a large dataset.

Brandon "Sandman" Bell: You've said in the past that: "On a daily basis we're running into the Windows 2GB barrier with our next-generation content development and preprocessing tools." How important a role has the additional memory capacity played in your development efforts and can you provide any specific examples?

Tim Sweeney: In Unreal Tournament 2003, we built 2000-polygon meshes, texture map them, and use them in-game with diffuse lighting. That was a simple process, which didn't require any memory beyond that taken up by the mesh and texture maps.

In our next-generation technology, we are building 2,000,000-polygon meshes, and running them through a preprocessing program that analyzes the geometry and self-shadowing potential of the mesh based on thousands of incident lighting direction using per-pixel floating point math, and compresses all of this data down to texture maps, bump maps, and 16-component spherical harmonic maps at as high a resolution as possible.

This process uses many gigabytes of memory, and implementing it on 32-bit CPU's places a lot of constraints on the size of meshes we can preprocess and the resolution of maps we can generate. With onerous programmer gymnastics, this kind of algorithm could be made disk-based or Address Windowing Extensions aware, but these approaches require an

order of magnitude more development effort, and aren't practical given the frequency with which we change and improve our algorithms.

So, overall, we've found 32-bit adequate for prototyping new content, but for serious development will only be possible with 64-bit.

Brandon "Sandman" Bell: Which operating systems will the 64-bit Unreal Tournament 2003 port support and when will they be made available?

Tim Sweeney: We've been up and running on 64-bit Linux for months, and will release it publically at the consumer Opteron launch. Stay tuned for news regarding a 64-bit Windows version.

Brandon "Sandman" Bell: What are your thoughts on the Athlon 64 and Opteron micro-architecture? How important is the integrated memory controller and 64-bit architecture and which feature do you feel is the most significant?

Tim Sweeney: Doubling the number of registers is a big and unquestionable win.

I see the onboard memory controller as a big win architectually, because it makes it possible to reduce system memory latency in half. Most people don't realize it, but system memory latency has only improved about 20X since the Apple][. Relative to CPU speed, it has worsened by 200X, so that now it takes 350 CPU cycles to read an uncached memory address.

Of course, having the memory controller on-die means that AMD had better be able to update the core frequently enough to track speed boosts in available memory. If AMD's memory controller is stuck at 333 MHz when 533 MHz memory is readily available and supported on Intel motherboards, that's going to put it at a disadvantage.

Brandon "Sandman" Bell: How do you feel AMD's 64-bit efforts compare to Intel's 64-bit Itanium family?

Tim Sweeney: Hammer follows the PC CPU pricing model. It's going to be very reasonably priced for the moderate high-end at launch, and over the next year will go down in price so that Hammer can ship in high-end, mid-range, and low-end PC's in all existing pricing segments, consumer,

workstation, and server, desktop and mobile. It runs all existing 32-bit software and OS's extremely well, better than existing AMD processors, and will run future 64-bit software and OS's extremely well.

Itanium isn't anything like that. You might as well be comparing Hammer to PA-RISC or SPARC. These are CPU's from a world alien to PC users, where you buy a \$10,000 workstation containing a pair of \$4000 CPU's and you only run the one or two CAD programs you bought your workstation for, because you can't run existing software at any reasonable level of performance. It's an interesting architecture, but it doesn't have anything to do with me or you.

Brandon "Sandman" Bell: How important a role will the 64-bit instructions play in your next generation engine? Will you be adding exclusive features for 64-bit users?

Tim Sweeney: There's a good chance 64-bit will likely be mandatory for content development. Since we release the Unreal level editor and scripting framework to users, this affects gamers and not just us internally.

For playing the game, we'll support both 32-bit and 64-bit. Depending on how much content we end up with, there's a good chance that we'll expose high-detail modes that will require 64-bit, giving you higher texture detail, for example. But there won't be any divergence in the gameplay itself.

Brandon "Sandman" Bell: What kind of development support have you received from AMD so far? Do you feel AMD is actively supporting the game development community with its 64-bit launch?

Tim Sweeney: Yeah, AMD is really standing behind the platform, not just providing early hardware, but also assuring that all of the development tools and OS components are in place and available to developers.

Chapter 6

Tim Sweeney of EpicGames

This interview was conducted by Anthony 'Reverend' Tan for Beyond3d on Feb 24, 2004.

<http://www.beyond3d.com/interviews/sweeney04/>

Interview

Prologue

It is unlikely that there are seasoned gamers that haven't already heard of Unreal. Despite seriously divided opinions about the game as a whole, its graphics certainly earned almost universal applause and was one of the major reasons that 3D accelerators took off in a big way (think Unreal and you'll likely think of the misguided-and-demised 3dfx, where Unreal's use of 3dfx's proprietary Glide API was a huge reason for owning a 3D accelerator back then).

Epic Games (then known as Epic MegaGames) then quickly took advantage of the online multiplayer first-person-shooter craze (given a big kick almost single-handedly by id Software's efferverscent Quake) and gave us Unreal Tournament and Unreal Tournament 2003, with the former offer-

ing many innovative gameplay modes and is still highly regarded by the multiplayer community.

One of the founders of Epic, Tim Sweeney became a very public figure due to Unreal. Chiefly responsible for the engine (originally called the Unreal Engine) powering all of Epic's games, Tim has been continually refining the engine (now into its UnrealEngine3 iteration), taking advantages offered by the advancing technologies of both the CPU as well as the 3D hardware industry. The Unreal Engine has to be labelled a very successful engine by virtue of the number of non-Epic-developed games that uses it. With a number of licensees of the engine, probably most notably by Ubisoft with games such as Splinter Cell and most recently XIII, Tim Sweeney plays an important and influential part in the balance sheet results of Epic - beyond just creating an engine for games to be developed by Epic, licensees of the Unreal engine pay considerable sums of money to Epic.

This interviewer, having interviewed Tim three times in the past but never before on this site (which, when this realization dawned on this interviewer, was surprising), felt that it would be good to shoot some wide-ranging questions Tim's way. While the nature and focus of this site naturally should lead us to assume that an interview with Tim would be mostly about next-generation hardware or future technology, such an interview wouldn't be possible without having Tim hinting (or at least having the public think he's talking) about Epic's future technology (or, most immediately, what Tim is working on at the moment). The questions are therefore rather general in nature. Finding difficulty in providing us with a recent and decent picture of himself, we'll have to be content with reading Tim's thoughts.

Questions

Anthony 'Reverend' Tan: Can you tell us how, and in what ways, Epic has grown from the days when it was developing the original Unreal to Unreal Tournament 2003? Of the three games, which was the hardest to create and produce?

Tim Sweeney: When James Schmalz, Cliff Bleszinski and I began working on the game that became Unreal back in 1994-95, Epic was a shareware developer and publisher. We had released a number of games such as Jill of the Jungle, Jazz Jackrabbit, and Epic Pinball, but these were all relatively small projects with 1-3 person teams, with everyone working remotely. Unreal was our effort to grow into a major developer.

The original Unreal was the most difficult project we've ever undertaken, largely because it was our first large-scale project and the first experience with 3D programming and content creation for many of us. Development took 3.5 years and it was a great and trying learning experience for all of us. The game ended up being a big hit back in 1998 when it was released, selling around 1.5 million copies.

But Unreal 1's multiplayer didn't achieve everything that we wanted, so we set out to create Unreal Tournament. It was an easier and more rewarding project, because we started out with working technology from the very beginning and were able to focus most of our efforts on gameplay and polish.

Unreal Tournament 2003 was about half-way between the two original Unreal games in complexity. We had upgraded the tech so significantly that a lot of new learning was required before we mastered the new pipeline.

On the other hand, UT2004 started out with a 100% working game to begin with, and now we have been able to focus all of our efforts on improving the core gameplay with new game types like Assault and Domination, and the team is having a great time with it.

Anthony 'Reverend' Tan: You are often credited as "the Unreal engine creator". Can you be more specific about your duties? What roles do your colleagues like Daniel Vogel and Andrew Scheidecker play at Epic? That is to say, do different parts/aspects of the engine have dedicated personnel working on it?

Tim Sweeney: For the original Unreal, Steve Polge wrote the AI and gameplay systems, Erik de Neve wrote the procedural texture effects and helped optimize the renderer, and James Schmalz had written some of the weapon and inventory code.

Most everything else, I wrote : the core engine and resource management, the editor, the renderer, the networking code, the scripting system, the collision detection and physics. It was a fun and daunting experience. I think the one-man-band approach works well for the first year and a half of developing entirely new technology – if multiple programmers were working on the project at that point we would have just gotten in each others' way and made a mess of things.

By early 1997, we had hired Steve Polge after seeing his Reaper Bot mod for Quake, and he had written his first pass of the AI and gameplay code. That was my first experience collaborating and sharing responsibility for a large amount of code with a great programmer, and it was a very positive experience that paved the way for the next generation of Epic programmers.

Nowadays, we're developing the third generation Unreal engine, and it's a very collaborative effort between a quite larger programming team than the original Unreal. Andrew Scheidecker has written the third generation Unreal Engine's new DirectX9 renderer with its scene visibility solution, new material shader system, new terrain system featuring displacement maps and foliage, and has improved many of the engine's internals. He joined us during development of UT PS2, and he absorbs knowledge at an amazing rate. Dan Vogel has responsibility for the particle system, sound system, new background loading code, and lots more. James Golding is heading up our physics system, while Warren Marshall is responsible for UnrealEd. In total, there are now 11 of us programmers at Epic, each responsible for certain parts of the engine.

Anthony 'Reverend' Tan: Which aspect of a game engine is the most difficult to program and which is the most important, assuming "most difficult" and "most important" are not the same?

Tim Sweeney: I don't think it's wise to single out any subsystem as "the most important", because games are such an across-the-board experience. For a successful game, you almost always have to have fun gameplay (which usually implies good AI, good network code, and good physics), impressive visuals (implying a modern and stable renderer), good sound, sufficiently powerful content creation tools to empower the artists to work their magic, and a sufficiently powerful and stable engine framework (the

unglorious things like resource management and loading) to enable the programmers to work productively. A game or engine seriously lacking in any of these areas is very likely to turn into a disaster.

On the other hand, it's easy to point to the hardest part of engine development: putting together a coherent set of subsystems that interact well together. It tends to be much easier to program one cool feature in isolation than to make it integrate well with lots of other features. A modern game engine has a very complex set of internal interactions between rendering, networking, physics, gameplay, and editing tools.

For example, good gameplay networking performance depends on having a predictable physics model, and physics depends on collision, which depends on editing tools, which require a renderer for visualization, and that renderer isn't of much use if you don't have an editor so you can get content in. There are a great many circular dependencies, and as Dijkstra said, "The hard part of programming is how to not to make a mess of it".

Anthony'Reverend'Tan: You have mentioned that UnrealEngine3 - what you're currently working on - is :

" the name for the next major step in Unreal technology, aimed at DirectX9 hardware as the absolute minimum spec, and scaling (way) upwards from there."

Could you briefly clarify (perhaps with a summarized list of DirectX9 features) what is meant by "DirectX9 hardware as the absolute minimum spec" since there appears to be different interpretations of "minimum DirectX9 spec" from a hardware perspective?

Tim Sweeney: By DirectX9 minimum spec, we mean we're going to make a game that brings today's GeForce FX's and Radeon 9700+'s to their knees at 640x480! :-) We are targetting next-generation consoles and the kinds of PC's that will be typical on the market in 2006, and today's high end graphics cards are going to be somewhat low end then, similar to a GeForce4MX or a Radeon 7500 for today's games.

Specifically, all pixel rendering is done via high level shading language, and these shaders are quite complex and are often dynamically and programmatically constructed as artists visually put together their own cus-

tom materials. All lighting and shadowing are per-pixel, and we support lighting and shadowing techniques that go one or two generations beyond what the major upcoming 2004 games are doing.

Anthony 'Reverend' Tan: In your opinion, what is the currently the biggest hurdle to overcome in 3D hardware technology? We've seen that shaders go beyond simple texturing and lighting and we're witnessing faster bus technology what specifics would you like to see addressed?

Tim Sweeney: Now is a great time because 3D hardware is coming out of the dark ages of being toy game acceleration technology, and morphing into highly-parallel general computing technology in its own right. The last hurdle is that the GPU vendors need to get out of the mindset of "how many shader instructions should we limit out card to?" and aim to create true Turing-complete computing devices.

We're already almost there. You just need to stop treating your 1024 shader instruction limit as a hardcoded limit, and redefine it as a 1024-instruction cache of instructions stored in main memory. Then my 1023-instruction shaders will run at full performance, and my 5000-instruction shaders might run much more slowly, but at least they will run and not give you an error or corrupt rendering data. You need to stop looking at video memory as a fixed-size resource, and integrate it seamlessly into the virtual memory page hierarchy that's existed in the computing world for more than 30 years. The GPU vendors need to overcome some hard technical problems and also some mental blocks.

In the long run, what will define a GPU – as distinct from a CPU – is its ability to process a large number of independent data streams (be they pixels or vertices or something completely arbitrary) in parallel, given guarantees that all input data (such as textures or vertex streams) are constant for the duration of their processing, and thus free of the kind of data hazards that force CPU algorithms to single-thread. There will also be a very different set of assumptions about GPU performance – that floating point is probably much faster than a CPU, that mispredicted branches are probably much slower, and that cache misses are probably much more expensive.

Anthony 'Reverend' Tan: When can we expect to see games featuring the UnrealEngine3? Have there been any interests in this engine from

potential licensees?

Tim Sweeney: We're not going to be announcing anything along these lines publically for a while, because UnrealEngine3 projects are quite early in development and are largely tied to platforms and launch timeframes that haven't been announced. But we will be showing UnrealEngine3 behind closed doors at GDC to select development teams.

Anthony 'Reverend' Tan: Is UnrealEngine3 being created with Longhorn (the codename for Microsoft's next Operating System) in mind?

Tim Sweeney: We expect to ship 32-bit and 64-bit executables on-disc, likely with the highest level of graphical detail that our game supports on PC only be available on 64-bit CPU's running the codename Longhorn OS. We certainly won't require it to run the game, but there are a lot of things we can do based on its key architectural improvements including address space, but not only that.

Anthony 'Reverend' Tan: With consoles being more attractive than the PC in terms of profits, how does this affect the way game engines are being designed that simultaneously target both platforms?

Tim Sweeney: Next generation console weighs verily heavily on our minds for the third generation Unreal Engine, and is going to be a major focus of ours both from a game point of view and an engine point of view. Can't say more yet, though.

Anthony 'Reverend' Tan: Your thoughts on PCI-Express in terms of its potential effect on game engine design decisions, as expressed by Ubisoft's Dany Lepage on his own developer page at our site?

Tim Sweeney: Fundamental architectural improvements due to PCI-Express will likely have to wait for the next major Microsoft OS release to be widely utilized by games and by video drivers. But in the meantime, it's a great platform improvement, allowing another decade of significant performance scaling just as AGP reaches its limits. I'm also looking forward to it as a geek toy, being able to buy high-end PC's with two PCI-Express graphics slots and plug in two high-end video cards.

Anthony 'Reverend' Tan: On the 3D Independent Hardware Vendor (IHV) front, the past year has seen a very competitive battle between NVIDIA

and ATI. This is not only in terms of outright performance differences but also in terms of DirectX9 API features, with each IHV attempting to one-up each other in the latter category. As an Independent Software Vendor (ISV), do these differences (different performance, different features among different IHV parts) affect the way games are developed?

Tim Sweeney: We've been very happy with both NVidia's and ATI's efforts in the past few years. Though they compete aggressively on features, the architectures and common-denominator feature sets are sufficient that we're not held back. The only thing more we could wish for is for the badly underpowered integrated graphics chips from Intel and others to go away or improve enough that they aren't such unfortunate handicaps for game developers.

Anthony 'Reverend' Tan: There have been criticisms of some DirectX9 games not looking much better than a game like Unreal Tournament 2003. Tomb Raider : Angel of Darkness is an example, despite featuring a host of DirectX9 features like floating point textures and different 2.0 Pixel Shaders for certain effects. In your opinion, what and how do the general gaming public determine what is "great looking"? Do you think it wouldn't be too far wrong to say that it is still a matter of detailed and colorful textures, instead of what DirectX9 proposes to provide ("realism")? Are some of the DirectX9 features too "subtle" to notice in its displayed form?

Tim Sweeney: The first DX9 games will mostly use it to add some visual improvements on top of a rendering pipeline designed over a multi-year development cycle and targetted primarily at DirectX7 hardware. That approach doesn't give you anywhere near the full benefit of DirectX9, but it's the commercially reasonable way to go if you're shipping a product in 2004, because there are at least 10X more DirectX7 class cards in gamers' hands than DirectX9 cards.

The really interesting things in shader land will start to happen in the late 2005 to early 2006 timeframe. That's when there will be a good business case for shipping DirectX9-focused games.

Anthony 'Reverend' Tan: Finally, where do you think 3D hardware and CPU technology should be headed? Do you think we are likely see 3D hardware taking over some of the functions of the CPU, going beyond

rendering?

Tim Sweeney: I think CPU's and GPU's are actually going to converge 10 years or so down the road. On the GPU side, you're seeing a slow march towards computational completeness. Once they achieve that, you'll see certain CPU algorithms that are amicable to highly parallel operations on largely constant datasets move to the GPU. On the other hand, the trend in CPU's is towards SMT/Hyperthreading and multi-core. The real difference then isn't in their capabilities, but their performance characteristics.

When a typical consumer CPU can run a large number of threads simultaneously, and a GPU can perform general computing work, will you really need both? A day will come when GPU's can compile and run C code, and CPU's can compile and run HLSL code – though perhaps with significant performance disadvantages in each case. At that point, both the CPU guys and the GPU guys will need to do some soul searching!

Chapter 7

Interview with Epic's Tim Sweeney on UnrealEngine3

This interview was conducted by Anthony 'Reverend' Tan for Beyond3d on Jun 18, 2004.

<http://www.beyond3d.com/interviews/sweeneyue3/>

Interview

Prologue

Recently at the Game Developers Conference as well as at NVIDIA's launch of the GeForce 6/NV40, one of the highlights of both events was the demonstration of Epic Games' next-generation engine called UnrealEngine3 (UE3 for short). Although the engine is still about 2 years away from completion, its public showing in its current state left almost everyone in awe of the technologies showcased and as explained in snippets by Epic's co-founder Tim Sweeney.

We took the opportunity afforded by Tim, having recently informed this interviewer that further discussions of the game engine is possible after

both events, by asking him in a little more about UnrealEngine3.

Questions

Anthony 'Reverend' Tan: You have said that UE3 will target DX9-based video cards as the absolute lowest denominator, even to the point of saying DX9 cards such as the GeForceFXs and Radeon 9x00s will be brought to their knees in terms of performance. Using Valve's yet-to-be-available Half Life 2 as an example, will UE3 be at all scalable starting from lesser-than-DX9-level video cards (by making available, perhaps, DX7 and/or DX8 paths) or did you really mean what you said, i.e. nothing but a DX9 video card will be able to run UE3-based games? If the latter is the case, you are saying you are not worried at all by gamers that, by the time UE3-games are made available, still own pre-DX9 (like the GeForce3 or Radeon 8500) video cards?

Tim Sweeney: DirectX9 is our minimum hardware target for Unreal Engine 3. A great deal of generalization, improvement, and even simplification has been made possible by eliminating legacy code paths and formulating all rendering around fully-general pixel shader programs.

If we were going to ship a PC game on UE3 this year, we certainly would have aimed lower, since those GeForce4 MX's are very prevalent on very low-end PC's today. But we're looking further ahead, and as a result Unreal Engine 3 has become a great engine for next-generation console games and PC games aimed at the mass market in early 2006.

Anthony 'Reverend' Tan: NVIDIA's latest GeForce 6 series of video cards has support for Shader Model 3.0 which is exposed, but not useable, in the publicly available version of DirectX9.0b. With an anticipated release of a revision of DX9 that allows the use of SM 3.0, could you tell us what are the more interesting and useable advantages SM3.0 offers over the current SM 2.0 model? Referencing the two public demos of UE3, what kind of pixel and vertex shader 3.0 were used?

Tim Sweeney: PS 3.0 utilizes a wide range of optimizations, from 64-bit frame-buffer blending to looping and dynamic conditionals for render-

ing multiple light interactions in a single pass without requiring a combinatorical explosion of precompiled shaders.

Our pixel shaders in the Unreal Engine 3 demos are typically 50-200 instructions in length, and are composed from a wide range of artist-controlled components and procedural algorithms.

Our vertex shaders are quite simple nowadays, and just perform skeletal blending and linear interpolant setup on behalf of the pixel shaders. All of the heavy lifting is now on the pixel shader side – all lighting is per-pixel, all shadowing is per-pixel, and all material effects are per-pixel.

Once you have the hardware power to do everything per-pixel, it becomes undesirable to implement rendering or lighting effects at the vertex level; such effects are tessellation-dependent and difficult to integrate seamlessly with pixel effects.

Anthony 'Reverend' Tan: More on shader models. With the GeForce 6 (and presumably other future SM 3.0 parts from other Independent Hardware Vendors) being made available, do you prefer Pixel Shader 3.0 and writing "fallbacks" Pixel Shader 2.0, or do you prefer 2.0 and write extra codes for 3.0 usage? What about Vertex Shader 3.0? Will it be used for effects that can't be done on Vertex Shader 2.0 hardware? What examples would those effects be?

Tim Sweeney: A major design goal of Unreal Engine 3 is that designers should never, ever have to think about "fallback" shaders, as Unreal Engine 2 and past mixed-generation DirectX6/7/8/9 engines relied on. We support everything everywhere, and use new hardware features like PS3.0 to implement optimizations: reducing the number of rendering passes to implement an effect, to reduce the number of SetRenderTarget operations needed by performing blending in-place, and so on. Artists create an effect, and it's up to the engine and runtime to figure out how to most efficiently render it faithfully on a given hardware architecture.

Anthony 'Reverend' Tan: There are currently two floating point implementations, 32-bit and 24-bit. You have said that UE3 requires 32-bit for accuracy. Can you provide us with specific (to UE3 and not in general) examples where 24-bit floating point hardware will either not run UE3-based games at all or will present artifacts? Essentially, what are the spe-

cific precision issues with 24-bit FP vis-a-vis 32-bit FP in UE3?

Tim Sweeney: Unreal Engine 3 works best with 32-bit floating point, but supports 24-bit floating point. Currently there are a few cases where minor artifacts are visible with 24-bit floating point. As shaders grow more complex, these will likely be amplified. But even with 24-bit precision, the scene quality looks far higher than, for example, Unreal Engine 2 scenes rendered with 8-bit-per-component precision.

Anthony 'Reverend' Tan: Can you explain the shadow algorithm of UE3? How is the "soft shadow" effect achieved, taking into account any aliasing artifact? What are the advantages and disadvantages of this dynamic soft shadow implementation?

Tim Sweeney: All characters in UE3 are capable of casting dynamic soft shadows, both for self-shadowing and environment shadowing, and the performance of this is quite reasonable on today's high-end DirectX9 hardware – which will be in very mainstream consumer PC's by mid-2005. We won't be publicly disclosing implementation details until the first UE3 games ship, however.

Anthony 'Reverend' Tan: With regards to "virtual displacement mapping" or what is known as offset-mapping, other than the normally expected nice bumpy edges of corner walls, will this software technology be used for other things like, maybe, bullet-holes on walls or enemies? Currently, what is virtual displacement mapping utilized for in UE3?

Tim Sweeney: We're using virtual displacement mapping on surfaces where large-scale tessellation is too costly to be practical as a way of increasing surface detail. Mostly, this means world geometry – walls, floors, and other surfaces that tend to cover large areas with thousands of polygons, which would balloon up to millions of polygons with real displacement mapping. On the other hand, our in-game characters have enough polygons that we can build sufficient detail into the actual geometry that virtual displacement mapping is unnecessary. You would need to view a character's polygons from centimeters away to see the parallax

Anthony 'Reverend' Tan: You had informed us, during the demo of UE3 in NVIDIA's GeForce 6 launch event, that in-game models consists of approximately 6000-7000 polygons but that this is actually "scaled down"

from an original size of millions of polygons. How is "scaling down" achieved? Is there a tool for this or would artists have to create their own low-polygon cage?

Tim Sweeney: We provide a preprocessing tool with Unreal Engine 3 that analyzes the two meshes and generates normal maps based on the differences. This can be very CPU-intensive, so it can run as a distributed-computing application, taking advantage of the whole team's idle CPU cycles to generate the data.

Anthony 'Reverend' Tan: Are there different and separate lighting systems for the indoors and outdoors world of UE3? Is there usage of a kind of global illumination system or is it just normal mapping ala Half Life 2?

Tim Sweeney: The lighting and shadowing pipeline is 100% uniform – the same algorithms are supported indoor and outdoor. Various kinds of lights (omni, directional, spot) are provided for achieving different kinds of artistic effects.

All lighting is supported 100% dynamically. There is no static global illumination pass generating lightmaps, because those techniques don't scale well to per-pixel shadowed diffuse and specular lighting.

Anthony 'Reverend' Tan: Can you tell us how you would be making the editor for UnrealEngine3 (let's call it UnrealEd3) easier for artists when it comes to writing shaders or level development?

Tim Sweeney: In UE2, the editor was based on two primary editing components: The realtime 3D viewports and the script-driven hierarchical property editor. UE2 adds two more fundamental tools: A visual component editor and a timeline-based cinematic editor. The visual component editor powers the visual shader editor, sound editor, and gameplay scripting framework.

In UE3, shaders are created by artists in a visual environment, by hooking together shader components. Programmers can supply and extend shader components, which take a set of inputs (from textures or other shader components), perform some processing work, and output colors or texture coordinates.

A huge focus in UE3 has been empowering artists to do things which pre-

viously required programmer intervention: creating complex shaders, scripting gameplay scenarios, and setting up complex cinematics. We're building an engine to be suitable for small teams and very large teams alike, and in the past, programmers have been a bottleneck in both situations. The more power we put in the hands of content creators, the less custom programming work is required.

Anthony 'Reverend' Tan: Will UnrealEngine3 be taking advantage of PCI-E and use the GPU/VPU for physics calculations?

Tim Sweeney: All DirectX9 applications will automatically take advantage of PCI-Express as soon as it's available.

We're using the GPU for visual effects processing, which includes both the obvious task of rendering visible pixels, and the invisible supporting work of, for example, generating shadow buffers and performing image convolution.

However, given the limitations of the DirectX9 and HLSL programming model, I don't take seriously the idea of running non-visual processes like physics or sound on the GPU in the DirectX9 timeframe. For serious programming work, you need modern data structure features such as data pointers, linked lists, function pointers, random access memory reads and writes, and so on.

GPU's will inevitably evolve to the point where you can simply compile C code to them. At that point a GPU would be suitable for any high-performance parallel computing task.

Anthony 'Reverend' Tan: In our last interview, you have said that 64-bit CPUs will afford the highest level of graphics detail with the next Operating System (code-named Longhorn) from Microsoft running. Can you elaborate on that? What would be the specific examples where 32-bit CPUs will lose out to 64-bit CPUs graphically?

Tim Sweeney: Off-the-shelf 32-bit Windows can only tractably access 2GB of user RAM per process. UT2003, which shipped in 2002, installed more than 2GB of data for the game, though at that time it was never all loaded into memory at once. It doesn't exactly take a leap of faith to see scenarios in 2005-2006 where a single game level or visible scene will

require > 2GB RAM at full detail.

Anthony 'Reverend' Tan: Also in our last interview, you stated that consoles weigh heavily in the design of UE3. It is presumed that you're including the various next-generation consoles. Can you expand on this? For example, "threading" is presumed to be an important aspect of next-generation consoles - would UE3 be very "multi-threaded"? Generally, without pre-announcing next-generation console configurations/architecture, can you indicate what aspects of the design of UE3 are being influenced by consoles?

Tim Sweeney: We can only discuss these things with developers who are under appropriate NDA's.

Anthony 'Reverend' Tan: The next-generation console from Sony (Playstation3) has the "cell" technology, utilizing and taking advantage of multiple processors. id Software's John Carmack has voiced his disagreement (at the recent GDC) to the concept of multiple processors on both the console and PC industry. Can you comment on the concept of multi-processors in both industries, whether if it is the right way to go, or if it is avoidable because it is the wrong way to go, or...?

Tim Sweeney: In general, more computing power is always a good thing when it's exposed in a programmer-friendly way.

Anthony 'Reverend' Tan: With more consumers buying HD-TV's in America at least, will UE3 allow for HD-TV resolutions in the console market and take advantage of the 16:9 aspect ratio? Have you done a side-by-side comparison of images at 1080i vs. 720p to see which looks better? Any thoughts on the technology differences of 1080i and 720p? Will games made by Epic target 60fps at the highest TV resolution and scale back from that?

Tim Sweeney: Let's instead talk about something that's not under NDA. One can reasonably extrapolate from Moore's Law that by the time Unreal Engine 3 games begin shipping on PC, the higher-end consumer GPU's will be capable of running UE3 games at 1920x1200 resolution at frame rates that are considered good and competitive in each particular genre.

Anthony 'Reverend' Tan: "Virtual Displacement Mapping" is one of the

cool new technology incorporated in UE3, one that nobody initially anticipated with DX9. Can you provide some other examples of graphics technologies in UE3 that you're excited about or wish to talk about? Perhaps stuff like interaction with objects (shoot holes in a bucketful of water and have the water come out the bullet holes, i.e. object physics), advanced artificial intelligence, deformable terrain everywhere, advanced physics, advanced audio, networking, collision detection, etc.

Tim Sweeney: See <http://www.unrealtechnology.com> for the features we've announced so far.

Chapter 8

Talking nasty with Tim Sweeney

This interview was conducted by Billy "Wicked" Wilson for Gaming Groove on Dec 14, 2004.

<http://www.gaminggroove.com/article.php?id=27>

Interview

Prologue

Tim 'The Man' Sweeney should require no introduction around here, but in case you didn't know and were afraid to ask, Tim is of course our favorite Founder and President of Epic Games, and he's been known to toss the old C++ compiler around for a spin, every now and then :) Today Tim goes deep into language and 3D technology territory:

Questions

Billy "Wicked" Wilson: With hardware rendering speed growing at a rapid pace, how much of an extra strain is this putting on the art developers to keep up? Are there any unique custom tools that the latest Unreal Engine comes with that you are especially proud of?

Tim Sweeney: Art requirements are going up dramatically with the coming generation. The content we're building in Unreal Engine 3 looks perhaps 10 times better than in the previous generation, but it's taking 3-4 times longer to model such high-polygon objects. Our art requirements are approaching what you see in movies; they are perhaps a factor of 4 lower right now, but in the two previous generations our detail was tens and hundreds of times lower!

Most development studios already have a 2-to-1 ratio of artists to programmers. I expect this ratio to continue to grow.

Billy "Wicked" Wilson: Do you see these new demands on artists pushing a lot of smaller gaming companies out of business?

Tim Sweeney: The great thing about game development is that there are a broad spectrum of opportunities available for developers, from 1-3 person teams creating Nintendo GBA games, 5-10 person teams developing budget PC games, 10-20 person teams creating smaller PC and console games, and then the big leagues of 25-60 person teams creating triple-A games for PC and console. Arguably, the massive multiplayer games are a step above that. In addition, there are mod teams aspiring to "go commercial", and numerous startups looking for initial opportunities like add-on packs and contract development while building their teams.

The key to survival now is for developers to either do the right kind of project for their size and experience. If you're frustrated that you can't find a publisher to fund a 40-person startup you're trying to assemble, you're not being very realistic.

But from what we've seen among the early-adopter teams we've been working with as early Unreal Engine 3 licensees and evaluators, it's clear

that a broad set of opportunities are out there for teams that are eager and realistic in their expectations.

Billy "Wicked" Wilson: Are there any good "3D capture" technologies available, e.g. a camera that can take a photo and triangulate it and upload it to 3D studio, for example? Do you envision such products ever (next decade or so) make a major impact in the game development world?

Tim Sweeney: We looked into several off-the-shelf 3D capture solutions, and didn't find anything that was worth using. The polygon data they output is so rough that it typically requires more effort to clean-up the data than to create similar data from scratch using a modeling program. However, we just looked at the \$5K-\$20K solutions. There are \$100K 3D laser scanners that might be better, but that's not necessarily a better solution than a modeling program.

Here, I ought to mention ZBrush, a modeling program that is truly a god-send in the new era of high-polygon modeling and normal-mapping. If that didn't exist, we might bite the bullet and work more with the 3D capture tools.

Billy "Wicked" Wilson: The same thing for software in general. Software complexity has been growing exponentially over the past decade or so but we are still using essentially the same programming language to develop most performance software. Do you see an end in sight or are there any new specialized languages or language features (other than DX9 obviously) which will dramatically reduce the code complexity of future projects?

Tim Sweeney: Most major software today is developed in C++, with some tools often built in Java and C#. These languages have fairly poor complexity-scaling curves, and don't do anything to ease the use of multithreading, which will be critical shortly as Intel and AMD release dual-core CPU's in 2005.

While C++ certainly remains viable – and, is realistically, the "least bad" solution available today – I feel that we're approaching the same sort of diminishing returns that caused early developers to switch from Assembly Language to C, and then from C to C++. For the coming genera-

tion, we can and well get by just fine with C++, but I feel a consensus will emerge around the end of the decade that current development approaches are broken and need to be rethought.

Unfortunately, there isn't a clear successor to C++ lined up. Java and C# had their chance and it became clear that, while they simplify some aspects of development, they don't put us on the wholly new effort/reward curve the industry needs to be on, especially as relates to graphics and other forms of parallel processing, and multithreading. So the next transition could be very interesting.

Billy "Wicked" Wilson: Pixel Shaders seem to be the all the rage at the moment. What do you see as the next "big leap" in the 3D hardware world? Do you see CPUs becoming so powerful in the next 10 - 15 years that specialized 3D hardware will no longer become necessary?

Tim Sweeney: Unreal Engine 1's software renderer is the programming project I'm the most proud of, so I have a special appreciation for this question. In an interview in 1999 (<http://archive.gamespy.com/legacy/interviews/sweeney.shtm>), I predicted that CPU's and 3D accelerators were on a collision course that would be apparent by 2007.

But that was before programmable shaders, high-level shading languages, and floating-point pixel processing existed! So, I don't think many people would take that prediction seriously today. But from time to time, developers do need to evaluate the question of whether to implement a given algorithm on either a CPU or GPU. Because as GPU's increase in generality, they are more capable of doing things beyond ordinary texture mapping, while CPU's have unparalleled performance for random-access, branch-intensive operations.

Billy "Wicked" Wilson: Do you think the differences in the approaches ATI and NVIDIA use for AA and anisotropic filtering are of any consequence in actual gameplay?

Chapter 9

Q&A: Epic Games' Tim Sweeney on Ageia PhysX

This interview was conducted by James Yu for GameSpot on Mar 08, 2005.

http://www.gamespot.com/news/2005/03/08/news_6119896.html

Interview

Prologue

If semiconductor company Ageia has its way, the PPU (physics processing unit) will soon join the CPU and GPU as another hardware component that has a major effect on gaming performance. Current CPUs are only powerful enough to support 30 to 40 physical objects that you can interact with onscreen, but systems with a PPU add-in card will be able to support more than 30,000 objects on screen.

Ageia promises to bring a new level of physics to games with its <http://hardware.gamespot.com/Story-ST-x-1610-x-x-x> processor, and the company has the game developer support to do it. Epic Games, the creator of the hugely popular Unreal first-person shooter series, has chosen Ageia's

NovodeX physics middleware solution for Unreal Engine 3.

We caught up with Epic Games CEO and founder Tim Sweeney to find out what he thinks the dedicated hardware physics processing will do for gaming.

Questions

James Yu: How did you react when you first saw the Ageia PPU? Would you say that the PhysX chip will bring on a new physics revolution in gaming similar to the 3D revolution sparked by 3dfx in the 1990s?

Tim Sweeney: It's very clearly an idea whose time has come. In games, software-based rigid body dynamics physics has been in use for about five years. It also took about five years from the release of the first realistic 3D games (Wolfenstein 3D and Doom) to the first truly great 3D accelerator, the 3dfx Voodoo1. This is about the time that an industry-changing idea takes from first implementation to industrywide change including hardware adoption. Physics and graphics are both areas where dedicated hardware can exploit the problem domain's underlying parallelism to deliver far more performance than a sequential CPU.

James Yu: Can you give examples of how a game might be able to increase realism with the help of a PPU?

Tim Sweeney: When people talk about physics in recent games, they mostly think of Unreal Tournament 2004's vehicles or Half-Life 2's dynamic objects. There, you have 10 or perhaps 100 big objects interacting physically in an otherwise static environment. Knocking chairs and tables around is fun, but that's hardly the apex of physics simulation.

The next steps are realistic dynamic environments, fluid simulation, large-scale particle simulation, and other very large-scale physical phenomenon. If you look at a modern action or sci-fi movie, and what's possible with the non-real-time computer graphics effects there, it's clear that major new physics innovations will be introduced into gaming as hardware performance increases 10x, 100x, and more.

James Yu: Do you see any limitations in what the PPU can accomplish due to overhead issues associated with offloading work and transferring it across the system bus?

Tim Sweeney: The computations driving physics simulation and collision detection make use of a large amount of static data that needs to be uploaded to the hardware once, and a smaller amount of dynamic data that needs to be transferred per frame. This is the same usage pattern as a modern GPU, where huge textures and vertex buffers change infrequently, while the smaller rendering commands need to be sent each frame. The PPU or GPU then expends an enormous amount of parallel computing power in computing the result, but the result itself is fairly compact. A GPU's frame buffer is a few megabytes, and a PPU's result matrices will be similarly compact. So, the PCI Express or even PCI bus will be plenty fast to accommodate the required traffic.

James Yu: How long do you think it will take for developers to release games that will look and play significantly better on a PPU-enabled PC system?

Tim Sweeney: As with any new technology, there will be early games available that add hardware physics support into a mostly finished game design. That's the first stage, and it will give gamers the first hardware-accelerated physics support right away.

The later revolution will be in next-generation games designed with large-scale physics from the very beginning. PhysX will make that possible on the PC, while other innovations will make large-scale physics possible on next-generation game consoles. There is a great deal of synergy there, with Ageia's physics engine providing a great hardware-accelerated solution on PC (with a software physics fallback for reduced detail) and also addressing the needs of the future consoles.

James Yu: How will Epic's Unreal Engine 3 incorporate Ageia's technology?

Tim Sweeney: We've been collaborating with Ageia since their inception, and Unreal Engine 3 thoroughly exploits the Novodex physics API; when the first Unreal Engine 3-based games begin shipping in early 2006, they will really up gamers' expectations. The combination of next-generation

graphics, next-generation physics, and content-rich games goes way beyond current games, both qualitatively and quantitatively.

Chapter 10

Tim Sweeney/Unreal Engine 3 Interview

This interview was conducted by Jacob Freeman for nVNews on May 20, 2005.

<http://www.nvnews.net/vbulletin/showthread.php?t=50895>

Interview

Jacob Freeman: First of all, do you plan on supporting 64 bit CPUs in a 64 bit environment?

Tim Sweeney: Yeah, that's going to be a big feature of Unreal Engine 3, that we support both 32-bit and 64-bit out of the box, this'll be big in PC games.

Jacob Freeman: Do you see any benefits of running 64-bit CPUs? Is there any performance benefit?

Tim Sweeney: Yeah, compiled code runs faster because you have twice as many registers to work with, but also with Unreal Engine 3 we're really pushing the content... we'll be able to use high-resolution textures with more detail in the environment and that'll be a great thing on PC, which

is a really scalable platform.

Jacob Freeman: Yeah, also, will it support dual CPUs?

Tim Sweeney: Yeah, yeah, absolutely, so Unreal Engine 3 is broadly targeting multi-core CPUs, Sony has something like that with the Cell architecture, and Microsoft has that with the three CPUs in the Xbox 360 and, you know, Intel and AMD are already out with dual-core CPUs in the PC market.

Jacob Freeman: And do you also see a performance benefit running dual CPUs? Because most games now, they don't support dual processors, so you don't really see a performance benefit.

Tim Sweeney: Well, Unreal Engine 2 just runs single-threaded so you don't get a significant benefit from it but in Unreal Engine 3 you'll be able to do rendering and animation updates and physics in multiple threads so, I wouldn't say it would double, but it'll increase performance significantly.

Jacob Freeman: Okay, and will the PC version of Unreal 2007 differ from the Xbox 360 version?

Tim Sweeney: Well sure, PCs...

Jacob Freeman: [interrupting] I mean, talking graphics-wise.

Tim Sweeney: Well, the PC is a more scalable platform, so it'll run on \$500 PCs and it'll scale down to the Xbox 360 and it'll run on a \$3000 SLI machine with dual kickass Nvidia cards in it and it'll run with even more detail. So we'll be all over the place.

Jacob Freeman: And are the shaders in Unreal Engine 3, are they being developed with HLSL [High Level Shader Language]?

Tim Sweeney: Well, Unreal Engine 3 has a high-level material system, and in ours you connect your different material effects together in a graphical way, so in the end it does translate to HLSL on Microsoft platforms, and CG on the Sony and Nvidia platforms. But it's not really what you see when you're working on the engine, you see the really high-level artist-friendly shader system.

Jacob Freeman: And does the DirectX 9.0 shader compiler do as good at producing optimized code compared to, like, hand-writing the shaders?

Tim Sweeney: Yeah, well, we look at all the generated code from the HLSL compiler and it's within, you know, 2-5% of hand-optimized assembly code, so, it's at the point where it's good enough where we'd never ever touch assembly again.

Jacob Freeman: Sounds good to me. Will Unreal Engine 3 support Pixel Shader 2.0?

Tim Sweeney: Yeah, so we'll ship on any reasonable DirectX 9 hardware; the Geforce 6200 at the low end and the ATI Radeon 9800.

Jacob Freeman: How about Pixel Shader 1.1, like Geforce 4?

Tim Sweeney: We're not planning on it, but we might decide at the end of the project to do a crappy backwards-compatible hack for the really old hardware but we look at that like we look at software rendering in UT2004: it's not beautiful but it works.

Jacob Freeman: I think you added software rendering later in the Unreal Engine 2004.

Tim Sweeney: [nods] Right.

Jacob Freeman: And will a 6800 Ultra be able to run Unreal Engine 3 with all the options maxed at 1024 by 768?

Tim Sweeney: Yeah, 1024 by 768 should be perfect for an Ultra, of course by the time Unreal Tournament 2007 ships at the middle or the end of next year, you'll have even higher-end cards than that; you'll have four times the performance, so you'll be able to run 1600 by 1200 on those.

Jacob Freeman: And do you have any support for real-time soft shadows?

Tim Sweeney: Yeah, we have support for stencil shadows which are hard-edged, we support real-time soft shadows, for soft shadowing of characters and characters casting shadows in the environment, and we support pre-computed shadows. So our objective with Unreal Engine 3 is to give artists this big toolkit of shadowing effects that they can select from so they can make the tradeoffs between performance and visual quality.

Jacob Freeman: First of all how do you pronounce [Ageia]? Is it ah-geea?, or ah-jeea?

Tim Sweeney: It's 'ah-jeea'.

Jacob Freeman: Okay. How will that affect the performance of Unreal Engine 3 running on a hardware physics engine as opposed to a software physics engine?

Tim Sweeney: Well, the big thing there is how we'll be able to put far, far more physical effects, with things like particle systems, and fluid effects, where without the Ageia system, we'll have a particle system with only a few hundred particles, and with the system, we could have tens of thousands of particles there. And it's really nice, because it mirrors the kind of non-traditional processing power that's available on the Playstation 3 with the Cell architecture, so it's a factor of ten times more computing power, but it's very special-purpose.

Jacob Freeman: And what's your opinion of dual-core GPUs? Does a dual-core GPU, is it more efficient than an SLI system?

Tim Sweeney: It seems like it should be about the same comparing a dual-core GPU to an SLI system, maybe a bit faster because the on-chip communication is faster than your PCI-Express bus.

Jacob Freeman: Okay, one more question: R520 or G70?

Tim Sweeney: [laughs] Oh, G70 for sure.

Chapter 11

Tim Sweeney: the man, the mystery, the legend.

This interview was conducted by Jacob Freeman for nVNews on May 12, 2006.

<http://www.nvnews.net/vbulletin/showthread.php?t=70056>

Interview

Jacob Freeman: How will UT2007 use Ageia enhanced physics effects? Increased object count? Fluid effects?

Tim Sweeney: Anywhere from explosions to have physically interacting particles... we are also looking at fluid effects to see where we can use those, gee, blood spurts sound like they might be a good candidate! A lot of other special effects like that, where they dont affect the core game play, so that players with the physics hardware and players without the physics hardware can all play together without any restrictions.

Jacob Freeman: There was a lot of controversy with the recently released Ghost Recon, where some players got lower performance when enabling Ageia effects because the video card has to render more objects. Is that

something that should be expected or should frame rate be the same?

Tim Sweeney: For the record, acceleration hardware is supposed to accelerate your frame rate, not decrease it! [laughs] That seems like its just a messy tradeoff that they made there. You certainly want your physics hardware to improve your frame rate. That means that the physics hardware might in some cases be able to update more objects so you can actually render another frame, so you need to have some sort of rendering LOD scheme for that to manage the object counts, and obviously you don't want to take this ultra fast physics card and plug it into a machine with a crummy video card. You really want to have a great video card to match up with your physics hardware and also a decent CPU to have your system in balance to really be able to take advantage of the full thing.

Jacob Freeman: How about Ageia effects over a network? Is that supported or is it Client side? I imagine trying to push that amount of physics data through the network, there might be a bottleneck.

Tim Sweeney: There are a number of networking solutions for physics, what we are doing in UT2007 is using the physics hardware only for accelerating special effects objects, things where the server tells the client, Spawn this special effect here! The client responds with an explosion with thousands of particles, and each of those operates as a separate physics object but it doesn't effect the game play... its just purely a visual effect there. Thats the easiest and most common solution.

Jacob Freeman: Havok recently announced the ability to accelerate physics on the GPU. Is that necessarily a bad idea?

Tim Sweeney: Thats a good approach, they have some good technology there. Havok has a physics system that runs largely on the GPU to accelerate the computations there. It seems to be a lower precision physics than you have for the rest of the game which is problematic. You really want all the physics in the world to be drawing with a constant level of precision, so you don't have to make weird trade-offs there. I guess there is also the trade-off with that, if your GPU is doing both physics and graphics, then you are not getting the full utilization out of the system.

Jacob Freeman: Have you guys ever considered the possibility of maybe allowing console players to play against PC players in UT2007? Or is it

too difficult to balance the different players?

Tim Sweeney: The question of PC players playing against console players, or even console players on PS3 playing against Xbox360 is really more of a game play question than a technical one, because right now we do support running a PC server having PC clients join and play alongside PS3 and XB360 clients in our games networking framework. That basic approach works, but on XB360 side Microsoft has chosen to keep the network completely closed so you won't have PC players playing on Xbox Live. Kind of an unfortunate decision, but it allows them to secure their network and control it carefully which you kinda want in a console environment. On PS3, we might actually enable that. When we get down to balancing the game to really play well on a controller, if it turns out we don't have to change the game play significantly, then we will enable PC players and PS3 players to play together, and we are really looking forward to that. We really like the idea a lot, and it looks like Sony will be very supportive of the open network approach.

Jacob Freeman: It seems like both Nvidia and ATI are looking at unified shader architectures, instead of separate vertex and pixel do you think that is beneficial to gaming on a graphical level?

Tim Sweeney: Having one unified shader architecture enables you to do dynamic load balancing. Some scenes have an enormous number of pixels with a small number of triangles or vertices some have huge amounts of geometry with simple pixel shaders. You really want to have all the computing power in the chip utilized all the time and that means being able to shift the resources around between any potential use pixel, geometry, vertices, or just general computation that you are happening to do on the GPU. So I think, this is just the very beginning of a long term trend in where GPU's will become more and more generalized and eventually turn into full CPU like computing devices. Obviously the problem that GPU's are optimized to solve will always be a very different problem than CPU's, but they will certainly become more and more general purposed, to the point where you can write a C program and compile it for your GPU in the future.

Jacob Freeman: This isn't really a UT2007 question this is more a UE3 question; Do you guys have any plans to add DX10 features to UE3?

Tim Sweeney: Oh yea absolutely, we will have full support for DX10, we will use their geometry shader stuff to accelerate shadow generation and other techniques in the engine, we will be using virtual texturing. With both UT2K7 and Gears of War we are offering our textures at extremely high resolution like 2000x2000 resolution which is a higher resolution than you can effectively use on a console because of the limited memory, but it is something that certainly will be appropriate for PC's with virtualized texturing in the future, so we will whole-heartedly be supporting DX10. Its hard to say what the timeframe will be on that because Vista could ship this year, or next year, or whatever. But, we will certainly be supporting it really well when it comes along.

Jacob Freeman: And ten years from now do you vision that we will see... GPU's handling graphics, and PPU's handling physics, CPU doing A.I. and that kind of thing or do you think we will see some kind of blend of the 3 technologies or maybe 2 of them?

Tim Sweeney: Looking at the long term future, the next 10 years or so, my hope and expectation is that there will be a real convergence between the CPU, GPU and non traditional architectures like the PhysX chip from Ageia, the Cell technology from Sony. You really want all those to evolve in the way of a large scale multicore CPU that has a lot of non traditional computing power as a GPU has now. A GPU processes a huge number of pixels in parallel using relatively simply control flow, CPU's are extremely good at random access logic, lots of branching, handling cache and things like that. I think really, essential, graphics and computing need to evolve together to the point where the future renderers I hope and expect will look a lot more like a software renderer from previous generations than a fixed function rasterizer pipeline and the stuff we have currently. I think GPU's will ultimately end up being... you know when we look at this 10 years from now, we will look back at GPU's being kinda a temporary fixed function hardware solution, to a problem that ultimately was, just general computing.

Jacob Freeman: And this isn't really related to UT2007 or UE3, but I'm sure you have heard John Carmack talking a lot about megatexturing and how he uses it. I was just wondering what your thoughts on it was?

Tim Sweeney: So, megatexturing is this idea of applying absolutely unique

textures to every object in your environment everywhere. Computationally it looks kinda difficult because our resolutions are always going up at a steady rate, the amount of detail on our environment is increasing and the sizes of our environments are increasing. This, to me implies that you want to reuse your content very frequently. You want to be able to build a mesh in one place and reuse it hundreds of places of the environment just with minor modifications here and there. So, if your going to move in a mega texturing direction, I think you really have to look at that in the context of a larger material system that lets you instance objects and share assets and not have an explosion in the amount of content creation work thats required because if an artist has to sit down and paint every little detail in every object in the world, that an uneconomical approach to game development. So in order for Mega texturing to work on a large scale, I think you need excellent tools, for being able to reuse, instance and reuse all this data so you save the artists time and they dont have to rebuild custom things all over the place.

Jacob Freeman: Just a few more questions here, kinda of in tradition of last year.... R600 or G80?

Tim Sweeney: Ha-ha, well at Epic we are using mainly the Geforce 7800 and 7900, we have a few ATI cards, they perform really well and we are really happy with the solutions from both companies.

Jacob Freeman: One more question, kinda in the same fashion, Conroe or AM2?

Tim Sweeney: Haha, well thats hard to say, I don't know much about AM2, but Conroe is a really fantastic chip. The funny thing is very few people in the industry have been willing to come out and say that the Pentium 4 architecture sucks. It sucked all along. Even at the height of it's sucking, when it was running at 3.6GHz and not performing as well as a 2GHz AMD64... People were reluctant to say it sucked... so IT SUCKS! But Conroe really makes up for that and I am really happy to see that, that Intel is back on this track of extremely high computing performance at reasonable clock rates. Not having to scale up to these extremely hot running systems that require a lot of cooling. I think they are on a good track there and if they scale the Conroe architecture up to four and eight cores in the future, then that will put the industry on a really good track for im-

proving performance dramatically at a faster rate than we have seen in the past.

Chapter 12

DirectX 10 Preview: The Future of PC Graphics and Gaming

This interview was conducted by FiringSquad for Firing Squad on Oct 27, 2006.

http://www.firingsquad.com/hardware/directx_10_graphics_preview/page8.asp

Interview

FiringSquad: As a game developer who is used to working on the cutting edge, which new features in DirectX 10 excite you the most?

Tim Sweeney: I see DirectX 10's support for virtualized video memory and multitasking as the most exciting and forward-looking features. Though they're under-the-covers improvements, they'll help a great deal to bring graphics into the mainstream and increase the visual detail available in future games.

FiringSquad: Is there anything in DirectX 10 that you couldn't do in DirectX 9.0?

Tim Sweeney: Realistically, DirectX 10 doesn't introduce fundamentally

new capabilities, but brings many new features that will enable developers to optimize games more thoroughly and thus deliver incrementally better visuals and better frame rates.

If you look at the long-term graphics roadmap, there have only been a few points where we've gained fundamentally new capabilities. The most visible was the move from DirectX 6, 7 and 8, which in practice were fixed-function, 8-bit rendering APIs, to DirectX 9 with programmable shaders and support for high-precision arithmetic. Most of the in-between steps have brought welcome but incremental improvements, and DirectX 10 falls into that category.

From here on, there is really only one major step remaining in the evolution of graphics hardware, and that's the eventual unification of CPU and GPU architectures into uniform hardware capable of supporting both efficiently. After that, the next 20 years of evolution in computing will just bring additional performance.

FiringSquad: A lot has been made about the speed boost DirectX 10 will bring over DX9. In part due to the new driver model and in part due to other efficiencies. In your position you get to work with the latest hardware; can you tell us without violating any NDAs if these speedups are realistic or not? Will we really see a 6X increase in games or is this all theoretical?

Tim Sweeney: We don't have hard data yet, but it looks like there's potential to reduce the CPU cost of submitting rendering by a factor of 2-4. Since DirectX9 games are often CPU-limited, this should lead to significant visible improvements in frame rate.

More important, this lower overhead will enable us to render more objects per frame and increase the visual complexity of scenes in a more organic way than simply adding more polygons to existing objects.

FiringSquad: Based on what you've seen with DirectX 10, do you think it will be easier for game developers to program for than DirectX 9 was? If yes, which features really stand out?

Tim Sweeney: You can't really use the word "easier" in conjunction with supporting DirectX 10. Because it's only available on Windows Vista and

not XP, all developers who support it will have to continue supporting DirectX9, and henceforth maintain two versions of the rendering code in their engine. It's worth doing this, and we're doing it for Unreal Engine 3. But, far from making our lives easier, it brings a considerable amount of additional development cost and overhead to PC game development.

FiringSquad: With games using higher resolution textures and screen resolutions also going up, memory bandwidth is sucked up quickly, particularly on lower-end cards with slower graphics memory. How big of a problem is this and should hardware developers be focusing more of their time on solving this problem than on adding more functions to the GPU such as physics?

Tim Sweeney: PC games deal with bandwidth differences between the high-end and low-end quite effectively by scaling our video resolutions. Today's games generally support resolutions from 640x480 up to 2560x1600, which means we can easily scale by a factor of 13 in frame buffer bandwidth.

Talk of "adding physics features to GPUs" and so on misses the larger trend, that the past 12 years of dedicated GPU hardware will end abruptly at some point, and thereafter all interesting features – graphics, physics, sound, AI – will be software problems exclusively.

The big thing that CPU and GPU makers should be worrying about is this convergence, and how to go about developing, shipping, marketing, and evolving a single architecture for computing and graphics. This upcoming step is going to change the nature of both computing and graphics in a fundamental way, creating great opportunities for the PC market, console markets, and almost all areas of computing.

FiringSquad: We know that Unreal Engine 3 was largely developed with shader model 3.0 in mind, but do you plan on adding any DirectX 10 aspects into Unreal Engine 3 and ultimately Unreal Tournament 2007 or is that coming in UE4?

Tim Sweeney: Unreal Engine 3 will make full use of DirectX 10, and many of our and our partners' games will ship in 2007 with full support for DirectX 10 and Windows Vista. But, despite the marketing hype, DirectX 10 isn't all that different from DirectX 9, so you'll mainly see performance

benefits on DirectX 10 rather than striking visual differences.

FiringSquad: What are some of the things you would have liked to have seen Microsoft add to DirectX 10 that aren't in there currently?

Tim Sweeney: Microsoft made the right key decisions in developing DirectX 10. They invested heavily in a couple of bold operating-system-wide initiatives, including video memory virtualization and support for preemption, and introduced many welcome incremental improvements.

Ultimately, the DirectX 10 feature set resulted from about 7 years of discussion with key game developers. A lot of major ideas were proposed, including a multi-year effort by John Carmack to lobby for video memory virtualization. The features that didn't make it into DirectX 10 either weren't particularly beneficial, or clearly weren't practical for this time-frame.

FiringSquad: We know that the first games that are capable of taking advantage of some of DX10's features will ship next year. But how long do you think it will take before games require DirectX 10? When should gamers really care about this new API, when will it really begin to affect them?

Tim Sweeney: Requiring DirectX 10 is tantamount to requiring Windows Vista, and we have a lot of historical data we can use as a guide to such transitions. 2006 is the first year where it became economical for developers to ship games that don't support Windows 98 and Windows ME, which implies that an operating system has a 6-year lifespan.

Vista will ship in 2007, so mainstream games that require it should start appearing in 2012 or 2013. So much can happen in that kind of time period that we ought not even consider it.

DirectX 10 is a good and solid step forward for graphics, but it's very much an evolutionary thing, and for a game shipping holiday 2007, DirectX10 will represent maybe 10% of a typical game's customer base, say 35% Xbox 360, 35% PC, 30% PS3 (which will still be ramping up then), with one-third of the PC owners having new computers running Windows Vista with DirectX10 GPUs, and the other two-thirds either running XP or running Vista on DirectX9 hardware. I want to point this out in ad-

vance, since the marketing around DirectX 10 exceeds the (good but not revolutionary) reality.

Chapter 13

Exclusive Interview with Epic's Tim Sweeney on Unreal Engine 3 and UT 3

This interview was conducted by PCGH for PC Games Hardware on May 31, 2007.

http://www.pcgameshardware.de/?article_id=602522

Interview

Prologue

The original printed version was first published in issue 06/2007 of the german PC Games Hardware magazine. Here you'll find the unabridged interview of PCGH with Tim Sweeney. The Interview was conducted via email by Frank Stower on behalf of PC Games Hardware, all rights reserved. (2007)

Questions

PCGH: You rescheduled the game to Q3 in 2007. Was one of the reasons behind this that you had to do some technical changes with the engine or that you found ways to improve your game technology?

Tim Sweeney: We didn't reschedule anything. UT3 has always been about being done when it's done. We're taking our time with the game because the UT franchise is very important to us and we want to get it right for this, the third generation in the series.

PCGH: You are using deferred shading. Will there be any chance to get FSAA working with DX9-Level-cards? What about DX10-cards?

Tim Sweeney: Unreal Engine 3 uses deferred shading to accelerate dynamic lighting and shadowing. Integrating this feature with multisampling requires lower-level control over FSAA than the DirectX9 API provides. In Gears of War on Xbox 360, we implemented multisampling using the platform's extended MSAA features. On PC, the solution is to support multisampling only on DirectX 10.

PCGH: Can you at this point of development tell our readers what kind of hardware will be required to play the game with all detail in 1024x768 (No FSAA/AF) and 1.600x1.200 (with 4xFSAA - if available - and 8:1 AF)? Will there be any fallback modes for gamers with older hardware like Shader 2.0 cards?

Tim Sweeney: Optimization is still ongoing, so these numbers change on a daily basis. In general, our Unreal Engine 3 games run quite well on DirectX9 class hardware that NVidia and ATI released in 2006 and later, and amazingly well on the high-end cards including DirectX 10 cards. We also support Shader Model 2.0 hardware with minimal visual difference.

PCGH: Can player speed up performance remarkably by buying a second card for a SLI- or Crossfire system? Have you already measured/experienced differences between those two systems?

Tim Sweeney: We test on SLI configurations on a regular basis. There impact at higher resolutions is significant so if you want to experience the full beauty at high resolutions this is a great way to preserve perfor-

mance while doing so. We haven't had a chance to run on Crossfire yet, but would expect similar results.

PCGH: Could you in a couple of sentences sum up the technical as well as the visual highlights of the Unreal Engine 3 and especially UT 3 that will make your product superior to other competitors like Crytek (Cry Engine 2) id (DooM 3 Engine with megatexture technique)?

Tim Sweeney: We let our games speak for themselves.

PCGH: When did you get your first next-gen-hardware (DX 10-cards) to fiddle apart from console stuff?

Tim Sweeney: Our early access to hardware is generally covered by non-disclosure agreement.

PCGH: What is your experience with Nvidia's and Ati's next generation graphics hardware? Could you already make a statement which card will be better for UT 3, the 8800 GTX or the Radeon 2900 XTX?

Tim Sweeney: The relative performance scores between NVidia's and ATI's best cards vary from day to day as we implement new optimizations. But, for the past year, NVidia hardware has been ahead fairly consistently, and a few months ago we standardized on Dell XPS machines with GeForce 8800 GTX's for all of our development machines at Epic.

PCGH: Are there any plans at Epic to upgrade the engine for DX 10? Have you already made experience with Microsoft's new API?

Tim Sweeney: Yes, we'll ship Unreal Tournament 3 with full DirectX 10 support. Support for multisampling is the most visible benefit. We're also able to use video memory more efficiently on Windows Vista under DirectX 10, enabling a given machine to use higher-detail texture settings than are possible in Windows Vista under DirectX 9. Most of Unreal Engine 3's effects are bound by fill-rate rather than by higher-level features like hardware geometry processing, so the real impact of DirectX 10 is incrementally better performance rather than entirely new features.

PCGH: Do UT 3 gamers profit from a 64 Bit environment? Will there be a 64 Bit version? What are the advantages of the 64 Bit version? Are there any differences as far as visuals or performance is concerned?

Tim Sweeney: We're testing Unreal Tournament 3 with Windows Vista 64-bit to assure compatibility, but we're planning to wait for the OS and application base to mature before doing anything further to really exploit 64-bit.

We were the first developer to port to the 64-bit environment back in 2004, with 64-bit Windows XP and Unreal Tournament 2004. We were very eager to embrace Windows Vista 64-bit also, and hoped to have moved all of our development machines over to it by now. Unfortunately, the software and driver compatibility still isn't quite there. The base OS is very stable, and it's a joy to work with in isolation. But, then, you need to print something, or run Max or Maya along with a collection of third-party plug-ins, and it all unravels. I'm sure the issues will be worked out with service packs and app updates within the Windows Vista generation, as machines with 4-8GB are finally becoming economical, and could be mainstream in the next 12 months.

PCGH: How important will main memory be for the overall performance? How much memory would you recommend?

Tim Sweeney: We require at least 512MB, and you'll want at least 2 gigabytes for optimal performance and detail. Unreal Engine 3 is very scalable in terms of memory usage, so it runs well on low-memory machines at the low texture-detail setting.

PCGH: It is well known that your engine supports multi core CPUs. What is the maximum number of threads the engine can calculate? What is the performance gain when you play UT 3 with a quad core CPU? Will the engine even support future CPU with more than four cores?

Tim Sweeney: Unreal Engine 3's threading support is quite scalable. We run a primary thread for gameplay, and a secondary thread for rendering. On machines with more than two cores, we run additional threads to accelerate various computing tasks, including physics and data decompression. There are clear performance benefits to quad-core, and though we haven't looked beyond that yet, I expect further gains beyond quad-core in future games within the lifetime of Unreal Engine 3.

PCGH: Can UT 3 be played with full detail on a single core machine?

Tim Sweeney: You can play UT3 at any detail level on any machine; the dependent variable is the frame rate! If you have a fast GPU (and thus aren't GPU-bound), then you'll notice significant performance gains going from a single-core PC to a dual-core PC, and incremental improvements in going to quad-core, at a constant clock rate.

PCGH: Are there any things you learned while developing Gears of War for next gen consoles that you can now benefit from when finalizing UT 3 for the PC?

Chapter 14

EVGA Gaming Q&A with Tim Sweeney

This interview was conducted by EVGA for EVGA Gaming on Jul 17, 2007.

http://www.evga.com/gaming/gaming_news/gn_100.asp

Interview

Prologue

EVGA Gaming recently sat down with Tim Sweeney of Epic Games, for some technical Q&A on their upcoming release, Unreal Tournament 3, which uses the revolutionary Unreal Engine 3 technology.

Questions

EVGA: Recently you spoke of Unreal Tournament 3 (UT3) using DirectX 10 rendering features. How will this differ from the DirectX 9.0 graphical

effects, is this just for a speed increase? How is the geometry shader being used with DirectX 10 hardware and UT3?

Tim Sweeney: We're primarily using DirectX 10 to improve rendering performance on Windows Vista. The most significant benefit is that, on Vista, DirectX 10 enables us to use video memory more efficiently than DirectX 9 and thus use higher-res textures.

The most visible DirectX 10-exclusive feature is support for MSAA on high-end video cards. Once you max out the resolution your monitor supports natively, antialiasing becomes the key to achieving higher quality visuals.

EVGA: Does Unreal Tournament 3 take advantage of over 512MB of video memory?

Tim Sweeney: We're going to significant lengths to take advantage of 512MB video cards and PCs with lots of memory. On PC, we're shipping lots of 2048x2048 textures, a higher resolution than we can support pervasively on console platforms. However, PCs running 32-bit Windows XP or Vista run up against a glass ceiling pretty quickly above 512MB of video memory. Because of the way the OS maps video memory into the 32-bit address space, going beyond 512MB doesn't really increase the overall usable memory. Thus, we're not focusing on exploiting more than 512MB video cards.

That situation will change over the next few years as Windows Vista 64-bit adoption takes off, because the 64-bit OS eliminates the address space bottlenecks and enables video cards to scale way up in usable video memory. But, if you look at the latest data on what gamers are actually using, e.g. with Valve's excellent survey of gamer hardware (<http://www.steampowered.com/status/s>) it's clear that 64-bit Vista isn't taking off yet. Give the industry a couple years, and this will change.

EVGA: Which next generation graphical features do you think will make the biggest impact on gaming?

Tim Sweeney: I feel like the evolution of gaming and graphics could almost be decoupled at this point. Today's engines and rendering feature sets are sufficient for building just about any type of game, just as to-

day's movie cameras as sufficient for filming any kind of movie. This isn't to say that we're done! Graphics quality will continue to improve at an impressive and scary rate, and I feel that we're still 10,000X short of having "enough" computing power for rendering 100% realistic scenes. But I think the days when "rendering feature X enables gaming innovation Y" are mostly over. This is why the use of middleware engines has increased so significantly - because one tech codebase can indeed satisfy the needs of maybe 80% of the types of games being built.

On the rendering side, I'm looking forward to the convergence of CPU and GPU programming models around next-generation uniform computing chips that are capable of both tasks. Such a convergence will enable a long string of software innovations that have thus far been held up by the Microsoft/NVidia/ATI obsession with adding new fixed-function features to the rendering pipeline. That's my wild prediction for the next decade! I believe many of the things that were hinted at in the late 1990's will finally start to happen - very efficient deferred software renderers, sub-pixel triangle rendering, analytic antialiasing, and novel new rendering schemes around voxels, frequency-space volumetric rendering, pseudo-realtime radiosity solvers, and so on.

EVGA: How will Unreal Tournament 3 use multiple cores on a CPU? Does it take advantage of Quad Core CPU's? If so, how/what task is assigned to each core?

Tim Sweeney: Unreal Engine 3 is a transitional multithreaded architecture. It runs two heavyweight threads, and a pool of helper threads.

The primary thread is responsible for running UnrealScript AI and gameplay logic and networking. The secondary thread is responsible for all rendering work. The pool of helper threads accelerate additional modular tasks such as physics, data decompression, and streaming.

Thus UE3 runs significantly faster on CPUs which support two or more high-performance threads. This includes dual-core Intel and AMD PC CPUs, the Xbox 360 (which sports 3 CPU cores and 2 hardware threads per core), and PlayStation 3 (with 1 CPU core running 2 high-performance hardware threads per core.)

Beyond two cores or hardware threads, UE3 performance continues to

scale up, as the additional threads accelerate physics and decompression work. However, not all scenes are performance-bound by such things, so there are diminishing returns as you go beyond 4 cores. By the time CPUs with large numbers of cores are available - thinking 16-core and beyond - we'll be on the start of a new engine generation, with some significant changes in software architecture to enable greater scaling.

EVGA: Does Unreal Tournament 3 use the "Subsurface Scattering" lighting effect?

Tim Sweeney: It's all about approximations! Unreal Engine 3 approximates of subsurface scattering using artist-created transmission masks, specifying the amount of light transmission through the back of a surface, and a color modifier. This enables us to achieve the most visible effects of scattering, such as the fine details of a character's face under complex lighting and shadowing details.

EVGA: What are your feelings on displacement mapping?

Tim Sweeney: Much ado is made about this feature, but ultimately displacement mapping is a form of geometry compression, and to realistically assess its benefits and drawbacks we must compare it to other geometry-compression schemes. In that regard, it's a pretty crappy form of geometry compression! It requires a parameterization of the underlying surface (which itself imposes significant burdens on artists to create an artifact-minimizing mapping), and to hide the seams, and has a directional bias often unrelated to the underlying geometry.

Indeed, there will someday be a revolution in fine tessellation of objects with sub-pixel triangle rendering, but displacement mapping won't be the magical feature that empowers it. More realistically, GPU makers talk about displacement mapping because it's a thing they know how to easily fit into their existing pipeline. Much of the modern graphics pipeline is derived from such expedience rather than a thorough analysis of how we might maximize rendering detail with the minimum hardware cost.

EVGA: Last year you mentioned that there possibly may be blood/water effects with physical properties assigned to it. Is this still planned to be implemented? Does Unreal Tournament 3 have any special effects for water/liquid effects?

Tim Sweeney: Stay tuned to upcoming press demos of UT3 for news.

EVGA: Does Unreal Tournament 3 support EAX? Does it use OpenAL like UT2004?

Tim Sweeney: On PC, Unreal Engine 3 uses OpenAL for sound, and takes advantage of the major 3D sound features available in hardware, or in the OpenAL software driver.

EVGA: Does Unreal Tournament 3 support HDR rendering with Anti-aliasing?

Tim Sweeney: Yes, on Windows Vista. On all PC platforms, we support running with 16-bit-per-component frame buffer (64 bits total). MSAA anti-aliasing support is only enabled on DirectX 10, because the deferred rendering techniques used by the engine require some capabilities not included in DirectX 9.

EVGA: Any plans to support cross-platform play in Unreal Tournament 3?

Tim Sweeney: Support for cross-platform play in Unreal Tournament 3 is a gameplay decision and a business decision that the development team is still considering. However, the underlying engine fully supports this capability to the extent that the underlying console platform allows communication with PC-based servers.

Chapter 15

Unreal creator Tim Sweeney: "PCs are good for anything, just not games"

This interview was conducted by Theo Valich for TG Daily on Mar 10, 2008.

<http://www.tgdaily.com/content/view/36390/118/1/1/>

Interview

Prologue

We got a chance to sit down with one of the sparkling celebrities of the IT industry during the the Game Developers Conference 2008: Tim Sweeney is founder and CEO of Epic Games, creator of the famous Unreal game engines.

TG Daily editor Theo Valich spoke with Sweeney about the future of the PC as a game platform, the role of the next-generation of game consoles, the next Unreal engine as well as the future of Epic.

We have known Sweeney for several years and are always looking forward to his view on the state of the gaming industry, which he is not afraid to discuss openly. In this first part of our three-part interview, Sweeney takes on the PC, which he believes is in trouble and can't keep up with game consoles, mistakes in Windows Vista and the integrated graphics dilemma.

Questions

15.1 PCs are good for anything, just not games

Theo Valich: Tim, Unreal has grown into a big success much because of the PC as a great gaming platform. We have heard about a new gaming PC alliance that wants to promote gaming on the PC versus gaming on the console. What is your view on that, especially on those stunningly expensive gaming rigs?

Tim Sweeney: There are many overpriced computers out there. It's like sports cars. They are everywhere, everybody writes about them, but there are only a few who can afford them. There isn't a great amount of people that will spend large amounts of money on that. In the case of PCs, they mostly don't deliver that amount of performance that you would expect to justify that cost. You pay twice as much money for 30% more performance... That is just not right.

Theo Valich: What about those high-end features? Do you think that industry is actually sending the wrong message when it comes to gaming? Do you feel that the hardware industry went with wrong message when it started to talk about 3-Way SLI and other high-end things, while they did not work on expanding the PC gaming message to masses?

Tim Sweeney: Absolutely. That was a terrible mistake. Marketing people believe that there is a small number of people who are gamers and who can afford spending good amount of money on buying high end hardware.

Theo Valich: You have to admit, the margin is obviously there.

Tim Sweeney: Agreed. But it is very important not to leave the masses behind. This is unfortunate, because PCs are more popular than ever. Everyone has a PC. Even those who did not have a PC in the past are now able to afford one and they use it for Facebook, MySpace, pirating music or whatever. Yesterday's PCs were for people that were working and later playing games. Even if those games were lower-end ones, there will always be a market for casual games and online games like World of Warcraft. World of Warcraft has DirectX 7-class graphics and can run on any computer. But at the end of the day, consoles have definitely left PC games behind.

Theo Valich: But we mostly talk about conventional retail sales. Do you see an increasing divide between the Pc and consoles?

Tim Sweeney: Retail stores like Best Buy are selling PC games and PCs with integrated graphics at the same time and they are not talking about the difference [to more capable gaming PCs]. Those machines are good for e-mail, web browsing, watching video. But as far as games go, those machines are just not adequate. It is no surprise that retail PC sales suffer from that. Online is different, because people who go and buy games online already have PCs that can play games. The biggest problem in this space right now is that you cannot go and design a game for a high end PC and downscale it to mainstream PCs. The performance difference between high-end and low-end PC is something like 100x.

Theo Valich: In other words: Too big?

Tim Sweeney: Yes, that is huge difference. If we go back 10 years ago, the difference between the high end and the lowest end may have been a factor of 10. We could have scaled games between those two. For example, with the first version of Unreal, a resolution of 320x200 was good for software rendering and we were able to scale that up to 1024x768, if you had the GPU power. There is no way we can scale down a game down by a factor of 100, we would just have to design two completely different games. One for low-end and one for high-end. That is actually happening on PCs: You have really low-end games with little hardware requirements, like Maple Story. That is a \$100 million-a-year business. Kids are addicted to those games, they pay real money to buy [virtual]

items within the game and the game.

Theo Valich: Broken down, that means today's mainstream PCs aren't suitable for gaming?

Tim Sweeney: Exactly. PCs are good for anything, just not games.

15.2 Intel's integrated graphics just don't work. I don't think they will ever work

Theo Valich: Well, we do have a fancy new operating system on the PC, which is actually heavily promoted as a gaming platform. What are your thoughts about Windows Vista?

Tim Sweeney: I really don't know why they kept the 32-bit version of Vista. I was surprised when they decided to keep the 32-bit version, I expected that they would push the 64-bit version exclusively. It would have been the perfect time for that.

Theo Valich: Considering that almost all the computers that can run Vista in fact support the x86-64 extensions, that choice belongs to The Twilight Zone of the IT industry.

Tim Sweeney: Let's be clear with it. The switch to exclusively 64-bit would clean up all the legacy viruses and spyware programs that have been plaguing us for years. The requirement for much more system memory cannot be an excuse, because most owners of 64-bit processors have at least 1 GB of system memory installed.

Theo Valich: It would have been a soft switch when we compare it to the Mac, right? Almost all of our 32-bit software for Windows would continue to perform as it used to?

Tim Sweeney: Yes, we would have liked something like that to happen. In terms of Apple, there's a new PC in your future. In the case of Vista that would have gone 64-bit only, you would have ended up with five year old computers that still would have been able to run the 64-bit operating system.

Theo Valich: Let's go back to the gaming PC. What would you think if everyone would pursue a sort of an ease-of-use approach? For instance, in last two years, there were efforts to bring external graphics to life. It was supposed to be a compact box that would have a powerful discrete card inside. But in the end, it turned out that Vista's driver mode (LDDM) was incompatible with that.

Tim Sweeney: External graphics?

Theo Valich: A year ago, the PCI-SIG certified the PCI External standard, which enabled the conventional PCI slot to extend through several different cables. There were several Taiwanese companies such as Asus and MSI that demonstrated products based on different cards. In the end, you simply needed to plug the external box into a notebook or a desktop. Prototypes were using the ExpressCard interface.

Tim Sweeney: Oh... that's cool. Actually, this would be a really good idea. We always joked that there will come a day when you won't be plugging a graphics card into a computer, but you would connect the computer into an Nvidia box, because they were quite loud and using a lot of power. But this idea would be really good. I didn't know there was actually a development in that area. Sadly, this would not solve a problem that we have today, and that is the fact that every PC should have a decent graphics card. A PC should be an out-of-the-box workable gaming platform.

Theo Valich: What about notebooks?

Tim Sweeney: For notebooks this could be a really good solution. There is no room to put a fast GPU into that compact form.

Theo Valich: With that much background and knowledge about what PCs make sense and which do not, I'd be interested to learn what PC you are using.

Tim Sweeney: My work computers are Dell workstations. Currently, I have a dual-CPU setup, dual-quad cores for a total of eight cores, and 16 GB of memory. We at Epic tend to go to the high-end of things. Until recently, we used to buy high-end consumer PCs, simply because they tend to deliver the best performance. However, as time goes by, we constantly run into stability problems with CPUs and graphics, so we decided to

switch to workstations. We just need very, very stable computers and they perform very well.

Theo Valich: So, you aren't really after the highest benchmark numbers obviously.

Tim Sweeney: Part of the problem we see with these systems is that that they are ultra-fast, but often we see our PCs running under full load for 16 hours a day on various projects. We are constantly loading the systems, for instance using Radiosity. These computing tasks are extremely hardware extensive. Most of the high-end systems we worked on are just not engineered to support that.

Theo Valich: What are your thoughts on the future of the PC as a gaming platform? Is scalability the future - we hear AMD talking about Spider and Nvidia is selling Triple SLI that will keep us upgrading over the next several years. Or did the industry lose its focus?

Tim Sweeney: PC gaming is in a weird position right now. Now, 60% of PCs on the market don't have a workable graphics processor at all. All the Intel integrated graphics are still incapable of running any modern games. So you really have to buy a PC knowing that you're going to play games in order to avoid being stuck with integrated graphics. This is unfortunate, and this is one of main reasons behind the decline of the PC as a gaming platform. That really has endangered high-end PC game sales. In the past, if you bought a game, it would at least work. It might not have been a great experience, but it would always work.

Theo Valich: Can that scenario change?

Tim Sweeney: Yes, actually it might. If you look into the past, CPU makers are learning more and more how to take advantage of GPU-like architectures. Internally, they accept larger data and they have wider vector units: CPUs went from a single-threaded product to multiple cores. And who knows, we might find the way to get the software rendering back into fashion. Then, every PC, even the lowest performing ones will have excellent CPUs. If we could get software rendering going again, that might be just the solution we all need. Intel's integrated graphics just don't work. I don't think they will ever work.

Theo Valich: These are harsh words. It looks like Intel has a lot of things coming down the pipe.

Tim Sweeney: They always say "Oh, we know it has never worked before, but the next generation ..." It has always been the next generation. They go from one generation to the next one and to the next one. They're not faster now than they have been at any time in the past.

15.3 DirectX 10 is the last relevant graphics API

Prologue

In the second part of our interview Tim Sweeney discusses the challenges and dramatic changes that are just ahead for game developers and gamers: Graphics rendering may change completely and Microsoft's DirectX interface may become less important. The successors of the Xbox 360 and Playstation 3, due in 2012, could be running entirely on software pipelines.

The idea of extremely powerful graphics processors being used for general purpose applications is a much debated topic in the games industry as well - and Sweeney believes that the GPU and CPU will be heading into a battle for the dominant position in a computer - and either one could be pushed out of the market.

Questions

Theo Valich: In the first part of our interview you implied that software rendering might be coming back. Daniel Pohl, who rewrote Quake 3 and Quake 4 using ray-tracing [and is now working as Intel's research scientist] recently showed ray-tracing on a Sony UMPC, an ultraportable device equipped with a single-core processor. True, the resolution was much lower than on PCs of today, but it looked impressive. What are your thoughts on ray-tracing? How will 3D develop in the next months

and years?

Tim Sweeney: Ray-tracing is a cool direction for future rendering techniques. Also, there is rendering and there is the ray scheme of dividing the scene into micro-polygons and voxels. There are around five to ten different techniques and they are all very interesting for the next-generation of rendering.

Rendering can be done on the CPU. As soon as we have enough CPU cores and better vector support, these schemes might get more practical for games. And: As GPUs become more general, you will have the possibility of writing a rendering engine that runs directly on the GPU and bypasses DirectX as well as the graphics pipeline. For example, you can write a render in CUDA and run it on Nvidia hardware, bypassing all of their rasterization and everything else.

All a software renderer really does is input some scene data, your position of objects, texture maps and things like that - while the output is just a rectangular grid of pixels. You can use different techniques to generate this grid. You don't have to use the GPU rasterizer to achieve this goal.

Theo Valich: What kind of advantage can be gained from avoiding the API? Most developers just utilize DirectX or OpenGL and that's about it. How does the Unreal Engine differ from the conventional approach?

Tim Sweeney: There are significant advantages in doing it yourself, avoiding all the graphics API calling and overhead. With a direct approach, we can use techniques that require wider frame buffer, things that DirectX just doesn't support. At Epic, we're using the GPU for general computation with pixel shaders. There is a lot we can do there, just by bypassing the graphics pipeline completely.

Theo Valich: What is the role of DirectX these days? DirectX 10 and the Vista-everything model promised things like more effects and direct hardware approach, claiming that lots of new built-in technologies would enable a console-like experience. DirectX 10.0 has been on the market for some time and the arrival of DirectX 10.1 is just ahead. What went right, what went wrong?

Tim Sweeney: I don't think anything unusual happened there. DirectX

10 is a fine API. When Vista first shipped, DirectX 10 applications tended to be slower than DirectX 9, but that was to be expected. That was simply the case because the hardware guys were given many years and hundreds of man-years to optimize their DirectX 9 drivers. With DirectX 10, they had to start from scratch. In the past weeks and months, we have seen DX10 drivers catching up to DX9 in terms of performance and they're starting to surpass them.

I think that the roadmap was sound, but DirectX 10 was just a small incremental improvement over DX9. The big news items with DirectX 9 were pixel and vertex shaders: You could write arbitrary code and DX10 just takes that to a new level, offering geometry shaders and numerous features and modes. It doesn't change graphics in any way at all, unlike DX9. That was a giant step ahead of DirectX 7 and DirectX 8.

Theo Valich: Since you are a member of Microsoft's advisory board for DirectX, you probably have a good idea what we will see next in DirectX. What can we expect and do you see a potential for a segmentation of APIs - all over again?

Tim Sweeney: I think Microsoft is doing the right thing for the graphics API. There are many developers who always want to program through the API - either through DirectX these days or a software renderer in the past. That will always be the right solution for them. It makes things easier to get stuff being rendered on-screen. If you know your resource allocation, you'll be just fine. But realistically, I think that DirectX 10 is the last DirectX graphics API that is truly relevant to developers. In the future, developers will tend to write their own renderers that will use both the CPU and the GPU - using graphics processor programming language rather than DirectX. I think we're going to get there pretty quickly.

I expect that by the time of the release of the next generation of consoles, around 2012 when Microsoft comes out with the successor of the Xbox 360 and Sony comes out with the successor of the PlayStation 3, games will be running 100% on based software pipelines. Yes, some developers will still use DirectX, but at some point, DirectX just becomes a software library on top of ... you know.

Theo Valich: Hardware?

Tim Sweeney: GPU hardware. And you can implement DirectX entirely in software, on the CPU. DirectX software rendering always has been there.

Microsoft writes the reference rasterizer, which is a factor of 100 slower than what you really need. But it is there and shows that you can run an entire graphics pipeline in software. I think we're only few years away from that approach being faster than the conventional API approach - and we will be able to ship games that way. Just think about the Pixomatic software rendering.

Theo Valich: That technique was awesome.

Tim Sweeney: Yes, up to DirectX 8, we were actually able to use Pixomatic software rendering. In Unreal Tournament 2003, you could even play the game completely in software, running off the CPU. It was completely independent from whatever graphics hardware you had. It is only a matter of time before that level of performance is there in new variants of DirectX. On a quad-core CPU, you should be able to do that sort of thing again - completely software based DirectX rendering. Over time, I think that the whole graphics API will become less relevant, just like any other Microsoft API. There are hundreds of them in Windows, file handling, user interface and things like that. It is just a layer for people who don't want direct access to hardware.

15.4 Running Linux on a GPU is not a pipe-dream

Theo Valich: If your vision comes true, it looks like graphics will take the best from the CPU and the GPU, with graphics hardware continuing its evolution from a fixed function pipeline into what are basically arrays of mini-processors that support almost the same data formats as floating point units on the CPU today.

Tim Sweeney: It is hard to say at what point we are going to see graphics hardware being able to understand C++ code. But data will be processed right on the GPU. Then, you are going to get the GPU's computational functionality to a point where you can - not that this is useful, but it will

be a very important experiment - recompile the kernel for a GPU and actually run the Linux kernel off the GPU - running entirely by itself. Then, the boundary between the CPU and the GPU will become just a matter of performance trade-offs.

Theo Valich: General purpose GPUs competing with CPUs? Do you already have any idea who might win this battle?

Tim Sweeney: Hard to say at this time. Both can run any kind of code, GPUs are just much better optimized for highly parallel vector computing. CPUs are better for authorized out-of-order, branching, and operating system type of things. Once they both have a complete feature-set, things will get very interesting there. We could see the CPU pushing GPUs out of the market entirely, if the integration of highly parallel computing influences future designs. Or, we could see GPU vendors start pushing actual CPUs out of the market: Windows or any other operating system could run directly on a GPU. There are lots of possibilities.

Theo Valich: In some way, this is already happening in the handheld world. STMicro recently launched a chip that integrates the ATI z460 GPU a.k.a. mini-Xenos [a trimmed-down version of the Xbox 360 GPU - ed]. Nvidia launched the APX2500, a system-on-a-chip product that uses an ARM11 core for CPU-type computing and an ULP GeForce for the rest of the system. Intel is talking about such SoCs for the consumer electronics segment. Will we see something similar on the desktop?

Tim Sweeney: It is unclear what these products actually are. As they become actual silicon, we will be able to see how far the miniaturization can go.

Theo Valich: There are signs that a whole new market segment might reveal itself, enabling 3D performance with CPU-type computing on handheld-type devices, which so far provided pathetic a 3D GUI experience for users.

Tim Sweeney: Well, if we look at the iPhone, we can see that these low-power devices can actually be very important part of our lives. Now you can really browse the Web on a handheld - I mean you can actually browse the web in a decent fashion. Look at my Blackberry [8700]. There is a crappy little web browser thrown in to provide the simple functionality

and it just sucks. It is a horrible web experience. Apple's version is really good, it is usable. The video player and the YouTube integration are excellent. I definitely see those devices becoming a much more important part of our lives. For that reason, we need more and more compute power inside the same size package.

Theo Valich: But realistically, will it ever be possible to run a high-end game on a handheld platform?

Tim Sweeney: The way people go online and do things replaced a lot of things we used to do on our PCs, but not everything, of course. You don't use your handheld and write a document in word processing software. You don't spend hours playing a game on a handheld because the battery won't last. And these are tiny screens. Why would you play on a handheld, if you can play on large screens and enjoy the full experience of a game?

These [small] devices are important and I feel they will grow over time, as processing capabilities increase. I believe that the next generation of mobile gaming will be quite impressive. I think we're only few years away from really good user experience [in all segments]. If you look at the PlayStation Portable or Nintendo's handhelds, they are so low-end and so low-performance that they are just not interesting to mainstream game developers. But with another generation or two, they will have enough power to run a scaled down version of a high-end game console or PC game. Reduce the level of detail, lower the resolution and you will get the same game running on these devices.

Theo Valich: Then, it would not be a far-fetched call to see games based on next-gen engines such as Unreal Engine 4, even 4.5 or 5 or something like that, running on a device that fits in your pocket?

Tim Sweeney: That is the great thing about this scalable factor. On a 320x200 pixel screen, you have 30 times less pixels than on the highest-end PC monitor that is currently available. When you look at the performance figures, the actual scale is within reach. It should be possible to create a compelling next-gen experience on consoles, PC and handhelds.

Prologue

In the final part of our our interview, we wrap up with and outlook into the future of video gaming. What will be the next generation of game consoles bring? What can we expect from the next Unreal game engine? What about all those fancy brain-computer-interface devices and future game controllers that will allow you to control your character in a game with your body, rather than with a controller in your hand? Join us listening to Sweeney's answers to those questions and get insight in how video games will change over the course of the next four, five years.

Questions

15.5 Unreal Engine 4.0 aims at next-gen console war

Theo Valich: Throughout GDC, there have been several companies that were presenting game controllers that are so-called brain-computer interfaces. A while ago, I used OCZ's NIA device in Unreal Tournament 2004 and was blown away by the usability of that device. How do you see the interface between us, humans, and the computer evolving, given the fact that Nintendo has seen such a huge success with its Wii-mote?

Tim Sweeney: I think the key challenge here is to look at all the cool things engineers are developing and identify which ones are just gimmicks, which ones are cool ideas that might benefit one part of the market, but aren't fundamental changes and which ones are things that really change the way we work with computing interfaces. I still think that motion controllers, such as the Wii controller, have a limited purpose, sort of a gimmicky thing. Standing there and holding a bunch of devices and moving them around wildly is great for party games, but I don't think that will fundamentally change the way people interact with computers. Humans are tactile beings, so things such as touchscreens fundamentally improve the user interface.

Theo Valich: That brings us back to the iPhone, which we talked about earlier. Apple appears to have made a lot of progress in this area.

Tim Sweeney: I agree. You are not just bringing up the map on the screen, but you move it with your fingers, you zoom in and zoom out. It's incredible that nobody thought of that earlier. With 3D editing tools, the touchscreen approach would be an excellent thing. You can grab vertices and drag them in different directions. A touchscreen could really improve and change things. I think that we might see that technology migrating to Maya. It is hard to tell how exactly that will pan out, but I see that as a very big improvement in computing versus the motion stuff. These are just neat toys.

The other big direction is head tracing - cameras built into consoles. They watch you and detect, for example, your arm movement. It is just more natural, because it is somewhat annoying to hold a bunch of wired plastic do-adds, wireless things you have to pick up and recharge them every once in a while. To me, it's more compelling to just use free-form movement and have computers recognize your gestures.

Theo Valich: You mean behavior analysis?

Tim Sweeney: Yes, but I do not know how that will work out. We humans don't have great motion control, when it comes to moving arms around in free space, while we have astonishing control over fine movement - when we touch an object, for example. If you look at people, what we are optimized for is manipulating objects. That is what separates us from animals: We can manipulate with toys and bulletins and interact with complicated objects. Anything that is useful in that space is going to give computers precise tactile feedback and enable us to be in touch with objects. That said, it is very hard to say how the user interface will evolve. I am not an expert in those areas.

Theo Valich: At the end of the day, we all go back to basics at some point. According to one of those industry legends, Bill Gates and Microsoft engineers were roaming inside Apple, at a time when these two companies were friends. Gates asked Mac engineers how they managed to develop hardware to control the mouse movement. It turned out that Apple's engineers wrote software to control it. When we take a look at input devices of today, it seems that we are repeating the same thing what happened

early 1980s.

Tim Sweeney: Five years into development of personal computers we exhausted all the major ideas such as keyboard, mouse, joystick and gamepad. But then you see something like Apple's multi-touch, or you see that YouTube video on a big screen based interface where people walk around and just start manipulating objects that are projected there. That is new stuff, that's entirely new. No one really has done that before and it is clear that there are still a lot of major ideas that haven't surfaced yet. Yet. As the technology improves, one thing is certain: As you increase complexity of the user interface, you need more processing power.

15.6 The development of Unreal Engine 4 has begun

Theo Valich: Let's talk about your game visions for the future and the next Unreal Engine? Where is EPIC going with the Unreal Engine 3.5 and 4.0?

Tim Sweeney: The Unreal engine is really tied to a console cycle. We will continue to improve Unreal Engine 3 and add significant new features through the end of this console cycle. So, it is normal to expect that we will add new stuff in 2011 and 2012. We're shipping Gears of War now; we're just showing the next bunch of major tech upgrades such as soft-body physics, destructible environments and crowds. There is a long life ahead for Unreal Engine 3. Version 4 will exclusively target the next console generation, Microsoft's successor for the Xbox 360, Sony's successor for the Playstation 3 - and if Nintendo ships a machine with similar hardware specs, then that also. PCs will follow after that.

Also, we continuously work on transitions, when we go through large portions of the engine. We completely throw out parts and create large subsystems from the ground up, while we are reusing some things that are still valid.

Theo Valich: Like ...?

Tim Sweeney: The Internet bandwidth. In five years, the bandwidth isn't going to be more than 5-6 times higher than it is today. So the network code we have in the engine now will stay the same. Our tools are still valid, but we will rewrite large sections of the engine around it, as the new hardware develops.

Theo Valich: What part of the engine will need a completely new development?

Tim Sweeney: Our biggest challenge will be scaling to lots and lots of cores. UE3 uses functional subdivision and paths, so we have the rendering thread that handles all in-game rendering. We have the gameplay thread that handles all game-plays and uses AI. We have some hopper threads for physics. We scale very well from dual-core to quad-core, and actually you can see a significant performance increase when you run UT3 on a quad-core when compared to a dual-core system.

Down the road, we will have tens of processing cores to deal with and we need much, much finer grain task-parallelism in order to avoid being burdened by single-threaded code. That, of course, requires us to rewrite very large portions of the engine. We are replacing our scripting system with something completely new, a highly-threadable system. We're also replacing the rendering engine with something that can scale to much smaller rendering tasks, in- and out-of-order threads. There is a lot of work to do.

Theo Valich: You already have started working on Unreal Engine 4.0?

Tim Sweeney: We have a small Research & Development effort dedicated to the Unreal Engine 4. Basically, it is just me, but that team will be ramping up to three to four engineers by the end of this year - and even more one year after that. In some way, we resemble a hardware company with our generational development of technology. We are going to have a team developing Unreal Engine 3 for years to come and a team ramping up on Unreal Engine 4. And then, as the next-gen transition begins, we will be moving everybody to that. We actually are doing parallel development for multiple generations concurrently.

Theo Valich: Stepping back, what do you see as the most significant technology trends these days?

Tim Sweeney: When it comes to the PC, Intel will implement lots of extensions into the CPU and Nvidia will integrate many extensions into the GPU by the time next-gen consoles begin to surface. We are going to see some CPU cores that will deal with gameplay logic, some GPU stuff that will run general computing... and two different compilers. One for the GPU and one for the CPU. The result will be a reduction of our dependence on bloated middleware that slows things down, shielding the real functionality of the devices.

It would be great to be able to write code for one massively multi-core device that does both general and graphics computation in the system. One programming language, one set of tools, one development environment - just one paradigm for the whole thing: Large scale multi-core computing. If you extract Moore's Law, you see that with the number of cores that Microsoft put in Xbox 360, it is clear that around 2010 - at the beginning of the next decade - you can put tens of CPU cores on one processor chip and you will have a perfectly usable uniform computing environment. That time will be interesting for graphics as well.

At that time, we will have a physics engine that runs on a computing device, we will have a software renderer that will be able to do far more features that you can do in DirectX as a result of having general computation functionality. I think that will really change the world. That can happen as soon as next console transition begins, and it brings a lot of economic benefits there, especially if you look at the world of consoles or the world of handhelds. You have one non-commodity computing chip; it is hooked up directly to memory. We have an opportunity to economize the system and provide entirely new levels of computing performance and capabilities.

Theo Valich: Let's close the circle and return to the beginning of our interview: What does that mean for the PC?

Tim Sweeney: Well, that trend could also migrate to the PC, of course. I can definitely see that at the beginning of next decade: PCs will ship with a usable level of graphics functionality without having any sort of graphics hardware in the system. Your graphics hardware will be a VGA and a HDMI-out connector and that's about it. The same thing happened with sound. All the time, you had these high-end, ultra-expensive sound

cards and different levels of sound acceleration. And then look at what happened when Vista arrived. Poof! It is 100% software based and with one operating system, all that hardware acceleration was gone. We now have software sound and all that hardware is now used for digital to analog conversion. That is a great approach, because now there is a lot more flexibility: Now you have sound algorithms, treble control, you we got rid of all the hardware incompatibility issues that were the result of complicated fixed-function hardware and poorly written drivers. Things are much cleaner now and much more economical.

Simplifying the development process and making more efficient computer hardware is going to be the next big step for us all.

Theo Valich: Thank you for your time.