GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Eutechnyx' *Big Mutha Truckers*

By Paul Jobling

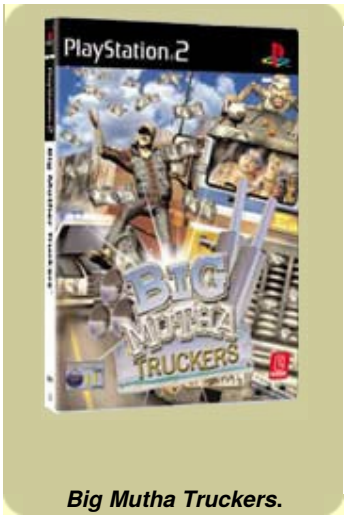
The origins of *Big Mutha Truckers* are very interesting. Eutechnyx has a reputation for developing highly accurate racing simulations based on lifelike physics and handling models, featuring licensed vehicles and accurate performance dynamics. We'd just completed another serious racing sim for the PS1, but when it came to our next title, we decided we wanted to develop something that didn't take itself as seriously and was not so focused on "pure" racing.

Also, as it was our first next-generation title, our goal was to produce a title that would make maximum use of the new systems' improved specifications and was much broader in scope, and not just do a graphical update of a previous game. We'd been working on developing streaming technology and so wanted to do a more "free roaming" type of game, as this would be an ideal showcase for this newly developed technology. This also meant that any titles we developed would be much less linear in their design and could give players something much more than "just racing."

We were thinking around a number of ideas on this front when our publisher at the time approached us with a simple concept: it wanted us to produce a driving game based around truck driving, but they didn't want a racing game or a coin-op style title, -- it wanted something with more depth and strategy. In other words, a free-roaming style of game.

During a three-day series of meetings with our publisher's designers in July 2001, we began to flesh out the ideas. We began with a simple question: why would a trucker drive from one place to another? We quickly hit upon the rather obvious motivation of "Trucks take things to places… don't they?" and so the "trucking and trading" model was born.

Any ideas of "fixed delivery challenges" were quickly pushed aside and instead we decided to let players purchase different "valuables" from different cities and then deliver them to a place of their choosing. This decision had the "free roaming gameplay" box on our checklist covered.



Big Mutha Truckers.

But the game still needed an ultimate objective, so we came up with the idea of "The Trial by Trucking", where the player chooses one of Momma's trucker kids as his or her persona and has 60 game days to make as much money as possible, thereby beating their siblings and inheriting the family business. Sure, it sounded like the plot to a chase movie, but that's exactly the mood we wanted to reflect. With this in mind, we began to focus on some key scenario elements, as we have found that the content we want to deliver affects our technology choices.

We came up with a short list of key elements:

- Setting it in the deep south of the United States (or to be more accurate -- the Hollywood perception of this region).
- Strong, interesting, and diverse characters
- A wide range of locations in the country to explore.

Because we believe that games by their very nature should make players feel good, one of the major motivators we put into the game was the ability to make wads of money. Not just a few grand--we were talking serious investment. A real "feel-good factor."

These thoughts immediately brought visions of Kris Kristofferson in Convoy, not to mention films like Smokey and the Bandit, and Every Which Way But Loose. It was probably because we'd seen too many Burt Reynolds movies that it just seemed natural that the game would be set in the "Deep South", where the good ol' boys are running moonshine, the sheriff's corrupt, and there's feudin' folk around every corner.

With our publisher in agreement regarding the basic concept and "plot drivers," we went away and started fleshing the game design out, beginning with the key characters. We started with the titular Mutha, who would be everyone's nightmare for a mother. She's cheated, swindled, and generally fiddled her way through life, and no matter how much her kids try to follow in her footsteps, Momma always would have gone a step further.

From Ma, there were her four dysfunctional offspring:

- Earl: An opinionated, overeating bigot
- Cletus: Dropped on his head one too many times as a child
- Rawkus: The cool guy, the brains of the operation
- Bobbie-Sue: We don't know where she came from but she's a lot prettier than Ma!


Earl.


Cletus.


Rawkus.


Bobbie-Sue.

The cast of rednecks from *Big Mutha Truckers*.

We're firm believers at Eutechnyx that the game's technology should be used as a means to deliver the entertainment content, not replace it. With this in mind, we were always keen to give as much thought on the story, location and character elements of the game as we did on our technology.

Taking the characters forward though, for real interaction there had to be conflict, and to achieve this we created a large number of non-player characters with fairly weird and wacky backgrounds that each player character would respond differently to. This was something we felt very strongly about, as we wanted to make our characters as lifelike as possible. It also meant - from a player's point of view - that playing as each character offered a different experience beyond simply having different handling. In effect, we wound up producing four games in one.

We then began populating our world with interesting and diverse characters. We didn't want to simply have "a mechanic" or "a barman," so we concentrated on developing characters with more depth. For example, there was "Red", the ex-con bartender, jailed for a series of major armed robberies and now recently released from sing-sing. Now he's trying to shake off his past and open up a franchise of bars, converted from former slaughterhouses. In total, there were over 30 non-player characters in the game, with each given as much thought and consideration as the player characters.

Of course, travelling between locations had to present the player with both entertainment and challenges, so we also worked on a number of events players could encounter on the way: police, biker gangs, natural disasters, and more.

The design team melded this mix of character conflict with interesting locations and driving challenges with the ability to make money and get a real financial reward, to create *Big Mutha Truckers*.

What Went Right

1. Streaming Technology. Eutechnyx has streaming capability on each of the major consoles, but since it was developed initially on PS2, that platform will be the focus of this discussion.

Eutechnyx developed streaming technology for its first outing into next generation platforms. The technology streams music, speech and the environment simultaneously during play. Although this technology is relatively commonplace now, at the time of development this technology was sought after, and was truly an attractive feature to publishers.

A benefit of the Eutechnyx streaming system is that it's easy to use for programmers. The streaming system was not reserved for music, speech or transitions between the front-end and game but was actually used for all data reading. From a high-level programming point of view, reading data was performed using exactly the same interface, whether reading files across the network when using a PS2 tool, from the HDD on the Xbox, from within a massive WAD-type file on DVD, or across a Firewire connection on a test kit. All the programmer needed to be aware of was that the "read" would take time, and to wait for the data to become ready.

Despite this flexibility, near-optimal read rates are regularly achieved and seek times are kept to an absolute minimum. There are four relatively simple ideas applied to the streaming system that enabled this to be possible. Importantly, these ideas are very interlinked and do not work well in isolation.

For memory management purposes, the game world was divided into segments at least 500 meters long (in the game world). Each block could contain unique textures, giving the game the ability to present a gradually changing landscape to the player.

The majority of available memory on the IOP is reserved for streaming buffers. Separate buffers are devoted to each data type. This means that although the read head can only be in one place at a time, multiple files can be read simultaneously and -- most importantly -- that the contents of these buffers are not invalidated when the head switches from stream to stream. When mastering, all files are concatenated into a single archive file and are grouped and sorted within the archive. Grouping is performed manually by scripting and in the first instance groups each data type into separate areas of the disc. Grouping is used sparingly as it takes precedence over the more powerful sorting method; grouping is reserved mainly for music, speech and videos. The sorting method is semi-automatic, and very simple. Near completion, all areas of the game are visited; a file list is written out as each file is accessed, this file is then fed back into the build process. The low-level streaming code also automatically reads ahead, resulting in many files being read "for free."

The game world needed to be divided up into land-blocks. As each block would fit into a pre-determined block of memory to prevent fragmentation, each land-block needed to be approximately the same size in memory, but could be any physical size. A lower limit was set on physical size of 500m in length. The lower limit was "calculated" using both qualitative judgments and mathematical equations. This limit was clearly based on loading speed, but also on manageability from an artist's perspective. The blocks became more like "zones," as each one can have its own unique textures and therefore its own look and feel.

In addition to art assets, AI data needed to be streamed too. AI data needed to be divided into global and local categories. Global data needed to be kept to a minimum, as this data would be in memory at all times. The global data consisted of waypoints and routes between them. This data was used so the AI players could be aware of and know how to locate specific places, even if they weren't currently loaded into memory. Local data was loaded with each land-block and contained information that was only needed when the camera was nearby - for example, more accurate path data, physics material properties, point sounds, and so on.

Streaming is a feature that is integral to our development and we are using it in our current projects as if it were second nature.

2. Extensive Economy Testing Prior to In-game Implementation. The economic model that powers *Big Mutha Truckers* is an in-depth system, based predominantly around supply-and-demand models. Although many other games use such a system, the *Big Mutha Truckers* model has a much more dynamic structure, since it had to be affected not only by the player's actions, but also by those of the non-player character rivals, who don't follow a fixed "trading script" and instead make decisions based upon the same market conditions as the player.

One of the designers was an economics graduate, so we called upon his expertise as an economist to ensure it behaved in a realistic and lifelike manner.

The result was that each game of *Big Mutha Truckers* will be different, but in a way that isn't based around simply generating random numbers or "cheating." Instead, there's a viable, fully functioning economic model based on real world behavior. Players can learn the system

and exploit it to make millions of dollars in the game, something we felt was key to motivating the player.

Because the economy formed the backbone of the game's challenge, it was vital that it functioned in a realistic manner -- and was fun to play. To ensure that the economy functioned in the desired manner, we produced a point and click "desktop toy" version of the game, which we then circulated around the office and encouraged people to play.

The trading system within *Big Mutha Truckers* is essentially turn based, although the turn order depends upon arrival times at different cities each day. The fact that the economy works in this way means that there are well-defined links between it and the driving aspects of the game. This allowed the economy to be fully developed, tuned and tested before the driving sections were implemented. Certain aspects of the driving section do affect the economy, such as fuel usage, truck damage, and police encounters -- and these features were implemented in the trading tool as simple mini-games, where players had a "PASS/FAIL" dexterity test (i.e. two "buttons" appear on screen and flash rapidly on and off, with players able to stop the flashing at the press of a key. Obviously, the aim was to stop on the "PASS" button.) The only additional aspects of the main game that needed to be tuned were related to damage and fuel usage.

Using the "desktop toy" we were able to ensure the economy couldn't be "broken" and this was tested extensively whilst the console technology and main game were being developed. The result was that we could immediately "plug in" the trading economy into the game knowing that it would work, thus saving valuable testing time later in the project. In the final game we also added parking and crash-combo bonuses to give the player extra cash, but aside from this the economy remained unchanged from the test tool to the final game.

3. The Asset Build Process. The final game-file archives are created from many hundreds of thousands of source files - textures, models, sound effects, music, scripts, and so on. We noted that it would be extremely beneficial to have a system where strict dependencies were maintained between all assets so that, for example, when a texture was changed, any models that referenced that texture were rebuilt. We created an extremely flexible build process tool with a GUI and a debugger that automatically calculated dependencies and only rebuilt assets when absolutely necessary. This system was written in Python, and the build scripts (lists of assets to be created) themselves are Python source files. This enables very complicated dependencies to be calculated at build time (for example, dependencies that depend on the contents of other assets). Although it initially took considerable effort to identify all dependencies, the fact that we now do not need to worry about whether or not an asset is up-to-date saves an enormous amount of time that used to be spent rebuilding all assets from scratch.

We also built a system that automatically kept assets up-to-date on all platforms, code-named "Wanderer", that allowed most programmers to just copy the latest files, avoiding the time previously spend rebuilding themselves.

For future projects, we are extending the build tool to a distributed build system that uses unused processor time on machines throughout the company.

4. Cross-Platform Development Philosophy. *Big Mutha Truckers* was Eutechnyx' first next-generation project and there was significant pressure to keep our existing engine and simply port it to PS2. However, the existing engine was optimized for the PS1 and not geared for cross-platform development. The decision was made to rewrite our game engine.

Having evaluated third-party rendering solutions, we opted for the flexibility of writing our own renderer. We knew that this decision would mean that we would be competing with second-generation titles in our first outing. We would it would put as at a disadvantage by having to develop both technology and a game simultaneously.

The cross-platform "common-interface" is set at a high level; this allows both general and platform-specific optimizations to take place. It also allows shared functionality across all platforms. The unique feature of the Eutechnyx common interface is that it has "width". For example, the sound-system has shared code (cross-platform) for managing active sounds; this then accesses the common interface to the platform-specific sound system. In this example I consider the common interface layer as "thick": it has a lot of multi-platform code below the common interface. In other cases (for example, in our optimized particle system), the layer is "thin".

While developing these interfaces, our philosophy was as follows:

1. Plan the interface thoroughly.
2. Get it functioning extremely quickly.
3. Spend as much time as necessary tidying and optimizing

Step 1 doesn't take much time if the correct people are involved. Stage 2, however, is the key. It can be reached quickly and allows features to be built, the game programmers to program, and the technologist to optimize. There may be early complaints of frame-rate problems, but that's nothing new and exactly what stage 3 addresses.

We would encourage getting as many areas as possible to stage 2 and worry about stage 3 later. This sounds like a "just in time" development philosophy, but that is exactly what is needed when developing technology simultaneously with a game.

As well as being cross-platform friendly, the Eutechnyx engine is cross-game friendly, with the core physics and driving code designed to run any style of driving game. This means we can develop multiple games on multiple platforms simultaneously. This helped Eutechnyx develop more cross-platform technology for subsequent development contracts.

5. Use of Detail Mapping To Improve Location Variation. Detail mapping was used mainly to enable us to have more variation on our

roads, while simultaneously using relatively little memory. It was also used on the walls of buildings, pavement, grass, and so on, but we found that its main benefit was on the road surfaces, where variations in lane layouts, chevrons, and junctions would have taken up valuable amounts of memory using standard texturing techniques.

Before Detail Mapping. Overall the general texturing looks good and pleasing to the eye.

After detail mapping. With the "detail" texture applied, the image is enhanced and the illusion of a bumpmap is created, adding contours to the hills and greater detail in the road.

The detail in the road surfaces tended to repeat, so it could be captured with tiled textures. On the other hand, the general road variation needed a texture with a much lower resolution, which allowed us to store many variations. This same effect could have been achieved using a repeating road texture with floating polygons for lane and road variations, but this required more polygons.

With detail mapping, the detail texture can be mip-mapped to the point where it has no effect, then not drawn at all in the distance. The trickiest part of the processes, therefore, is setting up the mip-mapping of the detail texture so that it doesn't shimmer at certain distances, while maintaining its clarity close up to the camera.

Using this technique, we were able to produce more varied environments for a minimum amount of effort by using detail mapping to customize the standard road/building textures within the game. This saved valuable development time without lowering the standard of the artwork or requiring more memory.

What Went Wrong

1. Over-Documentation. The game design document for *Big Mutha Truckers* was excessively large. It had to cover all areas of the game, such as the gameplay mechanics of the driving sections, the trading and background economic model of the world, win/lose scenarios, special mission scenarios, character profiles, and in-depth descriptions of all of the game's locations.

Due to the sheer volume of descriptive text required, the design was written much more like a screenplay or "game bible," and concentrated more on characters, the environments and the various scenarios within the game.

The design was also written as an in-house marketing tool, to help sell the concepts. As a result, instead of concentrating on the "hows" and "whys" of the game's production, it was instead focused on the "whos" and the "wheres." This, combined with the fact that the information relevant to a particular section of game was not always contained in one area, but often spread out over the entire document, meant that staff not only had to "extract" the tasks from the document, but also that, because of its size, they found it difficult to take the time to fully read the document.

The developers initially concentrated on the economic aspects of the game, but soon realized that players liked dealing with on-road events, like this hijacking attempt. Fortunately, Eutechnyx was able to add more events to the NTSC and later European releases.

We have since reorganized the way we produce designs, with each team given specific documentation relative to their area of production. A "standard" game design document is produced to detail the story, the visuals of the characters, and the locations and other content-related elements. That document is then broken down further to detail the specifics of each area for each team. The technical design document is also produced as a separate document, covering not only the technical side of the game (such as detailing how each element will function). but also a risk assessment of each element.

We also have created a collaborative intranet web site (a Wiki - see Jamie Fristrom's recent column for more information about Wikis) to aid sharing information. This is proving to be an invaluable way to keep important documents in one place and track changes within them. Any

changes are automatically flagged on the home page, so everyone can easily see when a relevant document has changed. While this is invaluable for internal development, the game design document is still essential and is available to all staff as part of our shared resource library.

2. Resources Focused on "Obscure" Content. One of the main problems with *Big Mutha Truckers* was the fact that our content -- and allocation of staff to its development -- wasn't as efficiently distributed as it could have been. There are many spectacular, one-time events within the game which took a lot of resources to implement, but they're not obvious and often difficult to find, so players don't often see them.

Our design intention was to include a lot of replay value, so we deliberately had some elements of gameplay that required the player to do a little more work to find them or replay the game and take a different "route" to uncover them.

For example, at one point players are given the opportunity to accept a mission that involves destroying a large suspension bridge. However, to qualify, they must be in a certain location on a certain date, having completed a certain number of missions prior to this. In other words, the criteria required to qualify to take part in this mission - and see the outcome - were far too complex. As a result, there are a number of events that most players will never encounter.

Not only is this bad from a gameplay point of view, but it also means that in-house resources were misspent on infrequent game events or content, when they could have been more focused and used on the "everyday" content. Rather than concentrate 50 percent of our output on content 10 percent of players might see, we should, instead, have catered to the majority of players. An example once more of Pareto in action.

We have since re-evaluated our approach to content within our games and our designs/game flow layouts work on the principal of allowing the player to encounter the special one-time events more readily and on using repeatable code, rather than squandering our efforts on elements players won't see.

3. Gameplay Balance. Throughout the game's development we discussed the primary focus of the game -- was it about trading or driving? How much of a challenge should there be in getting from city to city? Was finding the best price for your load the challenge, or did it come from on-road attacks? Were we aiming at driving gamers or strategy gamers?

Our intention was always to make the game's primary challenge come from the trading side of the game, with the on-road action being secondary. The idea was that players would be rewarded for making astute purchasing decisions and finding the best place to make a profit, as we felt this added an additional element of depth to the game and differentiated it from other driving games. Because the economy was dynamic, we felt this made the game more interesting and offered much greater replay value, as no two games would be identical, especially when compared to an arcade game where players can learn the "patterns" of the enemies.

However, when we demonstrated the game to people, they often would forget what they had just bought before leaving town and why getting $100 per unit more on their truckload of cattle was such a good thing. There were more interested in outrunning the police en route to their destination. Unfortunately, we planned a lot of on-road events (such as UFO abductions, redneck feuds and more) but didn't implement them, as we felt the trading to be the differentiating factor in *Big Mutha Truckers* and didn't want to dilute that. It seems that we were wrong and the game would probably have benefited from the occasional alien abduction.

With hindsight, more re-useable on-road activity would have made the game more immediately rewarding to players and accessible to a larger audience. This was addressed to some extent in the NTSC and later European releases, with the player being able to pick up cash bonuses by orchestrating crash combinations, but the initial European release received some criticism for the lack of hazards en route.

4. Underestimating the Animation Complexities. Although we'd never done it before, we thought that producing lip-synching would be a much easier task than it turned out to be, and as we couldn't get it working in time, it had to be dropped. We eventually cracked lip-synching for another project that was being developed concurrently with *Big Mutha Truckers*, but at that point in the development schedule it simply wasn't possible to implement and deliver it in the game.

The combination of a small animation staff and a large number of characters with speaking roles was

a challenge for the team. Eventually Eutechnyx abandoned the idea of implementing lip-synched dialog to keep the game on schedule.

Additionally we were working with a very small team of animators who were developing a lot of character models. A team of initially three people had to model, texture and animate over 30 characters in a short space of time. This meant the quality of the animations differed from character to character, with some being stronger than others. As there was a steep learning curve involved, some of the earlier characters weren't as well animated as the later ones, but we simply didn't have time to go back and modify them.

We also had problems with the way data is exported from 3ds max, which forced our technology team to constantly re-write our exporter. In max, characters would move perfectly and look great, but when they were exported, much of the data would be lost or mangled, so when the characters appeared in game, their movements looked unnatural. Vital development time was spent trying to fix this, and as a result, we were rushed toward the end of the project.

Seeing this problem, we assigned additional staff to our character team as the focus of the game shifted to a more character-driven title, and although the extra staff helped ease the workload and produce better results, we were still battling the deadline. Thankfully this situation has now been addressed by hiring animators and improving the abilities of our original staff, who were still "learning on the job" during the game's development.

Additionally, members of the technology teams have been assigned to work specifically with the character team - in other words, we've improved the lines of communication between departments - and the two interact on a daily basis to solve problems and test new systems and routines before going into production. This pre-production period has helped "iron out" critical path tasks, where previously this time -- particularly for animation technology -- was not available due to key programmers being tied up for the majority of the project developing game engine and rendering technologies.

5. Managing "Greenhorn" Staff. At Eutechnyx we often hire graduates straight from university. Although they rarely have any commercial programming experience, we do this because not only are they highly qualified academically, they also don't bring baggage with them from previous development studios and haven't picked up any bad habits.

Although this means we can shape a new programmer to fit our needs and work patterns, it also means that they require more help and supervision during the early stages of their tenure at Eutechnyx.

The start of the development of *Big Mutha Truckers* coincided with the start of the current generation of consoles and was also a time of company expansion, so Eutechnyx had just hired a number of people who were new to the industry.

Unfortunately, at the start of the project the experienced programmers were working on the technology that would be required to power *Big Mutha Truckers* - as it was our first PS2/Xbox title - and the less-experienced programmers were assigned the task of implementing this technology and adding content - in other words, coding the game content itself.

As mentioned previously, the economy was developed independently of the main game and this was an ideal mini-project to give to our new programmers, who did an excellent job. However, after the integration of the economy with the main game, we should have reorganized the teams.

In this instance the teams remained unchanged, with the majority of the game code being developed by new programmers and the technology by the veterans. Although each team size was relatively small, I believe if one programmer had been exchanged between the teams (each team consisted of about five programmers) the benefits would have been huge in terms of information exchange and accelerated learning.

There big benefit to integrating technologists with the game programmers is the transfer of knowledge. Interestingly, we found that this transfer is bi-directional: a technologist may be so bogged down with VU code that he becomes unaware of high-level utility functionality that is obvious to the game programmer.

Following this experience, Eutechnyx re-structured its teams and now has a technology department in addition to game teams. Many of the members of the technology department are now dedicated to a particular project, and our game programmers also perform technology tasks that benefit all projects. This crossover is working extremely well. When the next incarnation of game consoles arrive, I believe we will be much better prepared as a company to ensure that all our programmers will be busy producing useful code, while at the same time learning from each other.

It's A Good Sign When The "Wrongs" Are Hard To Find…

Overall, we were pleased with *Big Mutha Truckers*. Not to sound egotistical, but we actually had trouble coming up with five "What Went Wrong" items. We found it much easier to list the "What Went Rights." But perhaps that's because the game has a special place in our collective hearts as our first next-gen game.

It also appears that the game has struck a chord with gamers looking for something a little different. In a market dominated with licensed franchises and sequels, it's reassuring and pleasing to see that there's also a place for a quirky, original title like *Big Mutha Truckers*.

Game Data

Big Mutha Truckers
http://www.bigmuthatruckers.com

Publisher: Empire Interactive/THQ
Number of full-time developers: 40
Number of part-time developers: None (all staff were full time)
Contractors: Motion capture performers, soundtrack licensing, outsourced original music, voice actors, additional script writers.
Length of development: 2 years
Release Date: Christmas 2002 (Europe), Summer 2003 (USA)
Target Platform: Sony PS2, Microsoft Xbox, Nintendo Gamecube, PC.
Development Hardware: 800MHz+ PC , 512-102 MB ram, GeForce 2+, console devkits.
Development Software: Microsoft Visual Studio 6 (C++), Python, Photoshop, 3ds max, Mapper (in-house proprietary texturing/world building tool), LangTool (in-house text/localization tool), Tasker (in-house project management and bug reporting tool), Guibuild (custom build tool with full dependency checking and automatic building), Wanderer (custom asset version control)
Notable Technologies: Highly optimized cross platform engine, Streaming, Asset build system.

Key Staff:
Andrew Perella, Head of Programming
Kev Shaw, Lead Designer
Mark Barton, Creative Manager and Head of Studio
Peter Davies, Lead Programmer *Big Mutha Truckers*
Paul Jobling, Marketing Director

Return to the full version of this article