Postmortem: Double Fine's Brutal Legend

PRINT

By Caroline Esmurdoc

[In this postmortem taken from the December 2009 issue of Game Developer magazine, iconoclastic developer Double Fine Productions outlines the complex and at times daunting process of developing its action/strategy rock god epic Brütal Legend, taking in everything from legal troubles with Activision to tool development and everything in between.]

If the adage "go big or go home" applies to any software development effort, it applies to the making of *Brütal Legend*. As we did previously with *Psychonauts*, Double Fine once again bet it all on innovating -- this time on a game borne from the Full Throttle side of Tim Schafer's mind. *Brütal Legend* is a molten, balls-forward, third person, open world, strategic action-adventure interactive ride into the very soul of heavy metal.

The development story started out simply enough. After shipping *Psychonauts*, Double Fine created a collection of concept work, a pitch document and a game trailer intended to capture the spirit of *Brütal Legend*. Most publishers we spoke to were interested in the game concept, but their questions commonly indicated that they didn't understand where we were headed with it.

Questions were posed to us, such as "Why heavy metal? How about rock, or country, or hip hop instead? Why would you want to play as a roadie? How about playing as a rock god?"

One publisher, Vivendi Universal Games, did not ask these questions in the pitch meetings -- or in any meetings. They understood the game for what it was, and signed it for what they knew it could become.

We started development by focusing first on the multiplayer mode of the game -- our thinking was that since we'd never made a multiplayer game before, figuring that out would be our top priority. It took 16 months to do so.

In Rocktober 2006, we delivered a fully playable Ironheade vs. Tainted Coil skirmish to our publisher. At Vivendi's request, we then focused on the single player campaign, expanding its scope well beyond the initial design.

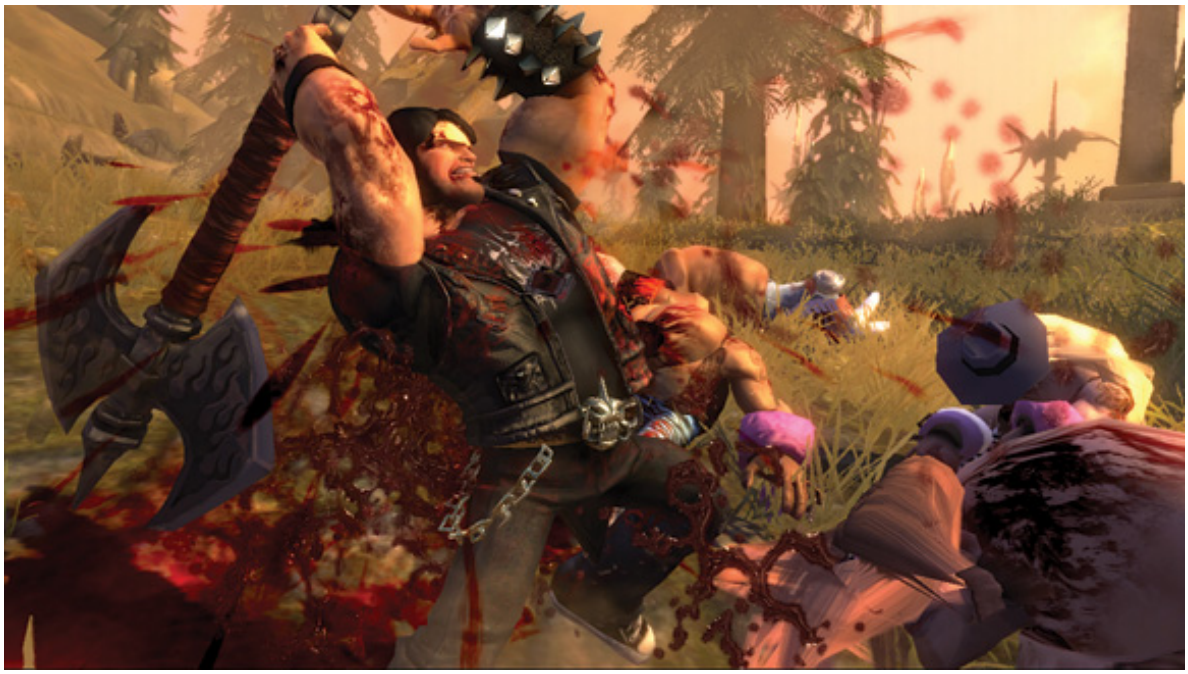
This is the period in which we added the voices of Jack Black and a host of other celebrity talent to the game, as well as other enhancements that solidified the vision for the campaign experience that we ultimately shipped. In June 2007, we delivered the first meaningful portion of the single player game and also promptly admitted that all of the changes we had made to the game content put us way behind schedule.

Our first schedule revision extended the project by ten months, the second by another seven. Initially scheduled to be released in May 2008 under the Vivendi Universal Games/Sierra banner, *Brütal Legend* finally shipped on October 13, 2009, published by Electronic Arts.

Double Fine encourages innovation, but that drive also means we can't always rely on previous experience to predict how a feature or an approach will turn out. On *Brütal Legend*, the practice of continuous iteration and concept refinement led to a number of prototyped ideas, many of which survived to ship in the final game, but just as many of

which were left to digitally fossilize in the annals of Perforce.

Whittling down to ten the list of things that went right and wrong during the development of *Brütal Legend* presented a considerable challenge. Here are some lessons that were the most surprising or impactful.



What Went Right

1. Pushing Creative Limits

Brütal Legend was to be the interactive amalgamation of the over-the-top ridiculous (yet deadly serious) world of heavy metal. We were reverent fans of the genre and felt it would be an honor to bring that world to life. *Brütal Legend* began with a simple list -- a game that embodied everything that could be found on a heavy metal album cover: chrome rivers, pools of blood, volcanoes, caves, fire-breathing metal beasts, laser panthers, bladehenge and beerhenge, dominatrices, latex and chains, disembodied undead heads...

To that we added the core gameplay. We wanted to make a brawler adventure game, where the player was a heavy metal roadie who evolves into a rock god over the course of the game. *Brütal Legend* on the Xbox 360 and the PlayStation 3 would have the brawn of an action game and the elegance of an RTS.

We learned early on in our relationship with Vivendi that RTS was a naughty word in the console space, so we stopped calling it by that name and, by extension, so did Electronic Arts -- positioning the game largely as an action title in the marketplace.

We wanted our RTS to exploit the consoles' advantage; putting the player in the center of the action. We wanted to give the player intuitive control of a character that could perform a variety of badass movements and abilities and also allow the player to personally bond with that character. And we wanted that character to command dozens of masterfully-dialogued troops simultaneously.

One of our biggest challenges in solving the RTS accessibility issue was squad orders. It took numerous attempts and countless focus tests, but we ultimately decided on a simple unification of the orders interface, wherein the AI behaved as the player would want them to on the battlefield.

The player would have four orders:

1) "Follow," where Eddie gives his troops commands to move to a specific location and possibly attack, allowing the order to be given only within "shouting range," a relatively large distance around Eddie that did not encompass the entire map -- solving the forced (ignore path enemies/obstacles) and non-forced (engage enemies along the way) issue by making the "follow" order non-forced when Eddie was near his troops and forced when Eddie was far away.

2) "Defend," where Eddie could command his army to stop moving and hold position, aligning them in the most advantageous manner (melee up front, ranged behind, support in the rear) facing the camera.

3) "Move," where the migration would be forced until the army is close to its destination, at which point it would engage nearby foes.

4) "Charge," a non-forced move to the enemy that is closest to the average position of all nearby enemies if your army was not attacking, and a forced move to the attack position even if that meant disengaging from their current activity if your army was attacking.

The Double Fine incarnation of a console RTS occurred to us not in an early pre-production meeting, but over years of painstaking iteration and reinvention and rework. We tested our progress in periodic "Mandatory Hour of Fun" sessions, where the entire team played the latest build and then met as a group to discuss what was working well and what was frustrating or could be made better. This open forum for the exchange of ideas as well as the momentum for continuous iteration fueled profound changes to the core game mechanic over the course of development.

---

2. Scrum

Prior to starting work on *Brütal Legend*, the Double Fine team had spent the previous five years developing *Psychonauts* -- the last two years of which consisted of a giant, grueling crunch wherein the company lost its initial publisher and nearly shut its doors before ultimately releasing the game.

When the euphoria of having shipped our first title wore off, it was apparent to all of us that Double Fine did not develop games the way other studios did, and that a different system of product development needed to be put in place.

The main cause of *Psychonauts*' horrifying crunch was due to our continued development of the game features even after the levels were built. With each improvement to the game mechanics came a corresponding rework of all of the levels. Lather, rinse, repeat.

Double Fine, and notably Tim, needed to play the game, live it, breathe it, let it steep over time, and iterate continuously on what makes the game fun and funny.

After research into methodologies, we were drawn to the advantages of agile software development and decided to adopt Scrum. Within the first few months of *Brütal Legend* development, the team was practicing Scrum, and the initial payoffs were impressive.

Scrum's emphasis on features over systems, on rapid prototyping and iteration, on cross-disciplinary teams, on people over process, and on the creation of a potentially shippable piece of software every sprint/milestone made the game playable at a very early stage in development: by month one we had a renderer, terrain, and a playable character (Eddie Riggs), by month two Eddie could drive his hot rod (the Druid Plow) around the terrain, and by month three Eddie could run over endless numbers of headbangers with his Druid Plow around a terrain height field. Hilarity ensued.

We applied Scrum not only to meta-game creation, but to micro-projects as well. At the very start of the development process, we had no idea how to make an RTS, and had no suitable engine with which to make one. We solved both problems by creating prototypes with an off-the-shelf PC engine and with which a number of our team members had some familiarity -- Unreal 2.5.

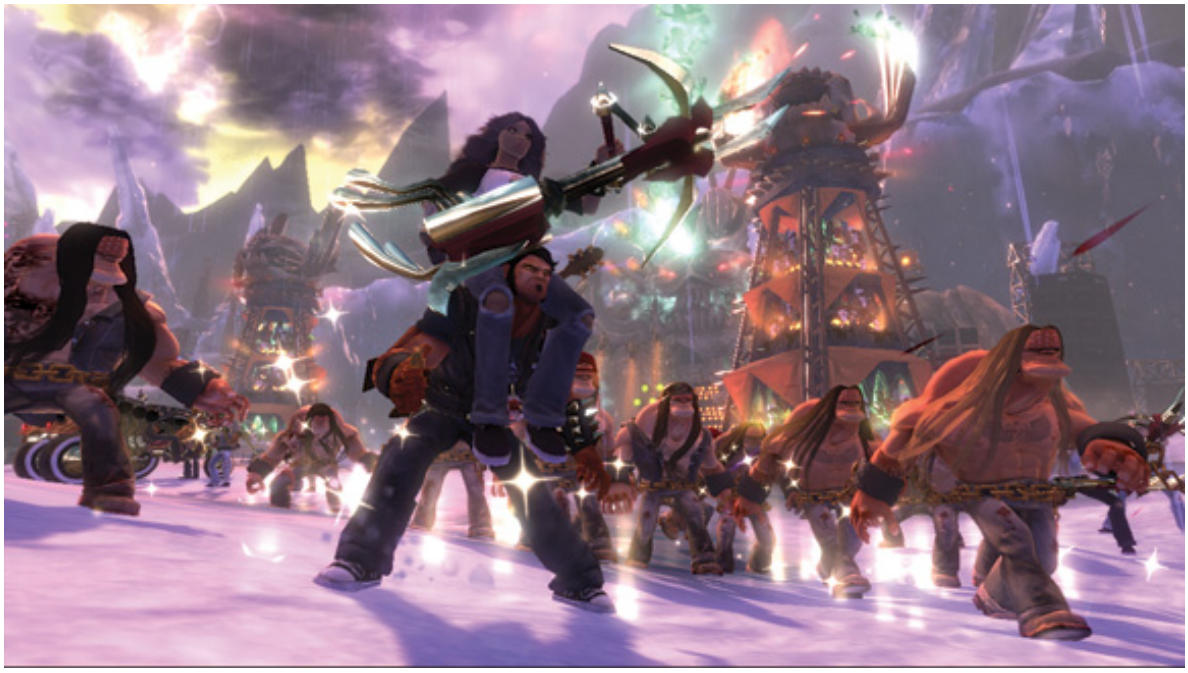
The design demands of *Brütal Legend* were such that trying to develop the game using an existing FPS engine would have proven difficult, but having the initial access to the flexibility of UnrealScript meant we could test some of our early RTS ideas right on our development PCs.

This approach allowed our designers and gameplay programmers to be immensely productive right away, while the programming team went to work building our new engine. This very early glimpse at the design challenges we would face during development, and the opportunity to iterate on something quickly with UnrealScript, gave us invaluable direction into how to architect our new engine and critical insight into the mechanics that would come to define *Brütal Legend*.

There was a notable downside to Scrum that bears mention in spite of our success with it. Our implementation of Scrum encouraged a pre-production mindset far too long into production. Scrum neither encouraged defensive programming, nor the practice of designing systems that scaled well.

We took a number of systems to 80 percent, enough to prove an idea, or extend and refine it in a future deliverable. But we found in the last few months of development that the remaining 20 percent was another 80 percent of the effort -- leading to some unexpectedly crunchy milestones for the hardest-hit team members. Once the project was in production, because of the reliability and lower risk of art creation, a waterfall approach was more optimal than Scrum.

Even with these caveats, Scrum allowed the team to quickly test wild theories and not only keep the best ones, but also spit-shine them with continuous iteration over the entire course of development. It was a significant process improvement for the studio.



3. Bert's Bots

Double Fine employs a full-time testing army of two. We relied on our two testers to smoke the daily builds of the game, as well as to pound on any new features we intended to get into the next deliverable. It was obvious very early on that two testers alone could not keep up with the stability demands of the game, nor was it financially possible to hire a full time test staff for the duration of the project. Our solution was to develop an automated testing system, which we affectionately named RoBert, after Bert, our software test engineer.

The automated tests began as an experiment, and the initial system took about a month to put in place. The first simple scripts summoned every character and object in the game. They proved immediately useful in finding warnings and crashes.

Running the tests on a regular basis allowed these crashes to be found in a timely manner, narrowing down the causes of the crashes to the more recent changes made to the codebase. Programmers started running automated tests locally so that they could test risky code changes before checking them in. Tests would also be run to help reproduce a bug.

One particularly crafty programmer came up with the idea to borrow idle Xbox 360s and PlayStation 3s to run automated tests. Team members could always end a test and take their machine back, but it was useful and efficient to use idle machines to run tests 24/7. We estimate that automated tests in the bot farm ran for a combined total of 147,000 hours.

The automated test system was so successful that we extended it to include tests with two armies battling it out and balance tests to determine how powerful each unit was. Bot functionality was then added where input could be simulated.

This allowed a test to perform moves exactly as a player would in an actual game. Bots were used to perform attack combos, and a variety of other moves with every squad, as well as to find stuck-spots in the world. They were also used in multiplayer tests, which were invaluable in finding desyncs in our peer-to-peer lockstep networking system.

As long as there were available machines, 1v1 and team multiplayer tests could run on a regular basis. Finally, the bot system was expanded to play through the campaign. Test settings were added for playing through the secondary missions, exercising the Motorforge, upgrading equipment, or failing each mission before completion. Memory reporting was soon added to the campaign tests so programmers could track memory usage and leaks.

Automated tests could be run remotely and crash reports emailed to interested and responsible parties. Doing so significantly simplified running multiple tests simultaneously and tests were often run using differently-configured builds. Early on, most tests ran using a debug build so programmers could attach to a crashed machine and more easily debug the problem.

Late in the project, tests ran a special release build to find release-only crashes. There were some limitations to the system, such as a crashed machine in the bot farm being rebooted by the machine's owner before a programmer could debug the extant crash. And the bot never learned to path or drive, nor could tests be run in the pause or front end menus. We will be making these improvements and additional expansions to RoBert for future projects.

---

4. Tools Superpowers

Double Fine wrote a number of tools that, when added together, made for a streamlined development pipeline. It's hard to single out any one tool and express how it enabled efficient development. A couple of our weapons are described here.

The MUE (Multi-User-Editor) is our collaborative open world building tool which forms the backbone of our world production process. Its primary function is to allow designers and artists to work simultaneously on our large world.

The *Brütal Legend* world is divided into different "tiles", and the MUE allows for individual tiles to be checked out and edited by different individuals. In addition, any or all of the thousands of entities populating the world can be checked out and edited by different individuals.

The MUE also offers a propriety set of tools for editing height map terrain -- a bit like ZBrush lite, but specifically optimized for editing height map terrain.

In addition to the expected suite of sculpting and smoothing tools, the MUE has an innovative sculpting tool known as "terrain stamping" which allows artists to import pieces of mesh, arrange them on the terrain and then click a button to "stamp" that shape into the height map terrain.

It was an especially useful technique when creating critical gameplay areas that had specific height, angle, or width requirements for the terrain. In addition to sculpting the terrain, the MUE allows artists to paint down blended terrain materials and ambient mesh (grass, bushes, and so forth) in real time.

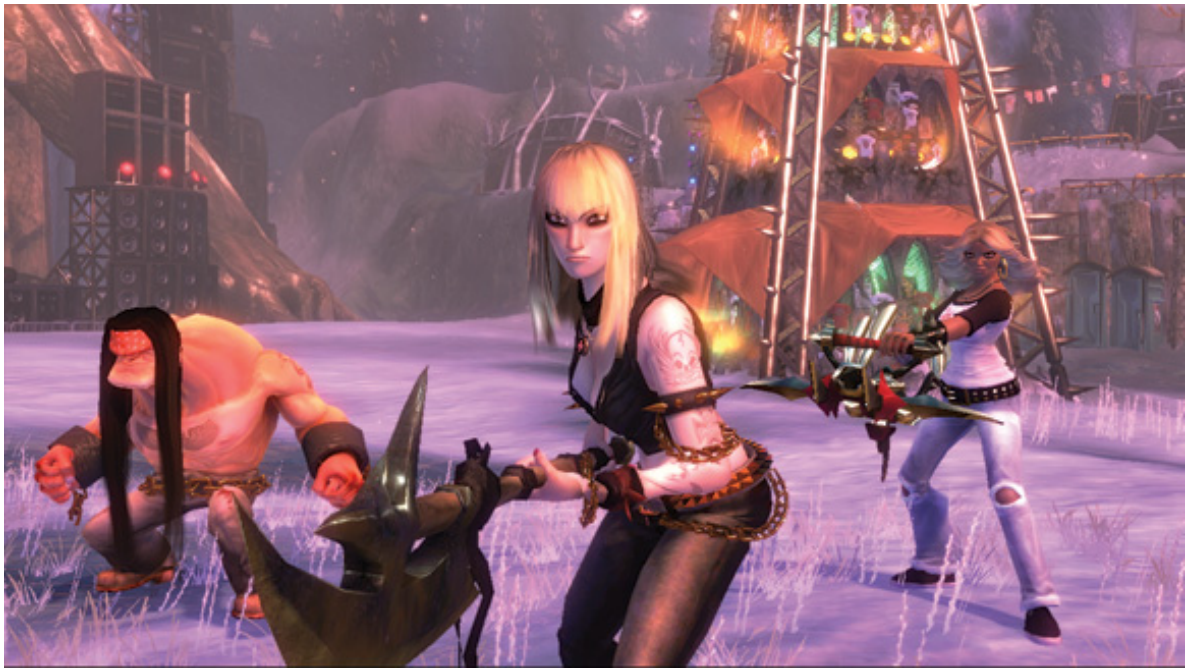
The MUE runs inside a Maya window, but under the hood lies a powerful SQL database. While having a database back-end may not sound exciting, it served as a powerful safeguard against data loss and enabled rapid worldwide changes. If an artist wanted to change all the poster trees into tire trees, he would run a one-line search-and-replace python script, which also handled the checking in and out of the data.

A second tool (called the Rigerator) was created to support our rigging and animation tools pipeline. Faced with the daunting task of creating and animating around 150 unique characters, some with upward of 40 facial shapes, as well as around 300 cinematic scenes, it was critical to automate the character rigging and animation pipeline as much as possible. Due to the high volume of art revisions made internally, we also had to enable making changes easily.

To achieve this we started by using Maya referencing and naming conventions assigned automatically by the tools to minimize user input, minimizing inconsistencies between users and files.

To author and edit character rigs, we created an automated process wherein a user could match a simple skeletal layout to their character's proportions and simply press a button to create the complex animation controls instantly with proper placement and naming based on whatever unique size or shape that character had. Prior to the development of this tool, the process would take one to two days to complete manually, but animators were now able to rig their characters instantaneously.

Additionally, since unique characters require many unique animations, we developed a tool that managed a library in which a character's unique poses and animations could be easily saved and loaded onto other characters in the game. While speeding up the animation process, this Anim Toolbox also kept all character animation sets creatively cohesive regardless of author, and it applied to both in-game animations as well as the cinematic scenes for the game. Without enabling this workflow we could not have created the vast amount of animation in the necessary time.



5. Big Names

Few games, if any, can boast the number of licensed music tracks or the number of signed celebrity actors/musicians that *Brütal Legend* can. It was a monstrous and painstaking endeavor to select each of the artists and tracks to include in the game, but it was worth the effort.

For licensed music, Tim and music director Emily Ridgway chose songs that had substance and credibility. Each song needed to both fit its use in the game and also be embraced by metal fans. For *Brütal Legend* to truly honor the genre, it had to be rich with heavy metal presence, since the music, lyrics, and imagery served as the game's creative direction.

Once songs were selected, it was then a Herculean effort to seek out, negotiate, budget, sign, and shepherd through the licensing process over 100 music tracks, each with a master and a synch agreement, many across multiple rights holders, and some from bands that had long since split up.

We sought permission for each song individually, and in some cases, Double Fine's personal outreach to bands on behalf of *Brütal Legend* resulted in the creation of original music for the game by the artist, such as Lita Ford's "Betrayal." The final soundtrack to *Brütal Legend* is epic, and the oft-unheralded licensing process that took the excitement of identifying and selecting tracks and translated it into permission to use is something we're proud of.

While our music content dreams were lofty from the start, we had not initially planned to cast Jack Black as the voice of Eddie Riggs. We approached him with the prospect after learning he was a fan of *Psychonauts*, and, as a metal fan himself, he signed on to do the voice acting for the game.

In addition to Jack Black, a number of other high profile musicians were identified and sought to provide the voices for the characters that would personify them in the game and honor their contributions to heavy metal. Each required outreach, negotiation, scheduling, and directing. Personality and humor always define the characters in a Double Fine production, and *Brütal Legend* is another example of casting and voice direction done right -- except this time around, the cast is a little more, well, famous.

Surprisingly for us, the beefy licensed music soundtrack, and the addition of a celebrity cast to voice the characters, especially the voice of Jack Black as Eddie, helped to inspire Tim's writing, motivate the team, and excite publishers.

What Went Wrong

1. Middleware

The prospect of building new next-generation tech on multiple consoles at the start of the *Brütal Legend* development cycle was a daunting endeavor for us to consider, and neither the practice of Scrum nor our own ambition would allow us to write every system necessary for the game.

We recognized the advantages of middleware use early in the game's development. Selecting the right middleware to integrate allowed us to get the game off the ground early and facilitated the rapid testing of new ideas.

This advantage allowed us to make strong demos at the start of our development cycle, which helped communicate our game concepts to potential publishers when pitching the game, and also to present strong milestones during pre-production when publisher expectations for deliverables were more forgiving.

Double Fine used middleware for audio (FMOD), physics (Havok), user interface (Scaleform), lip sync (Annosoft -- tools and offline), and video (Bink).

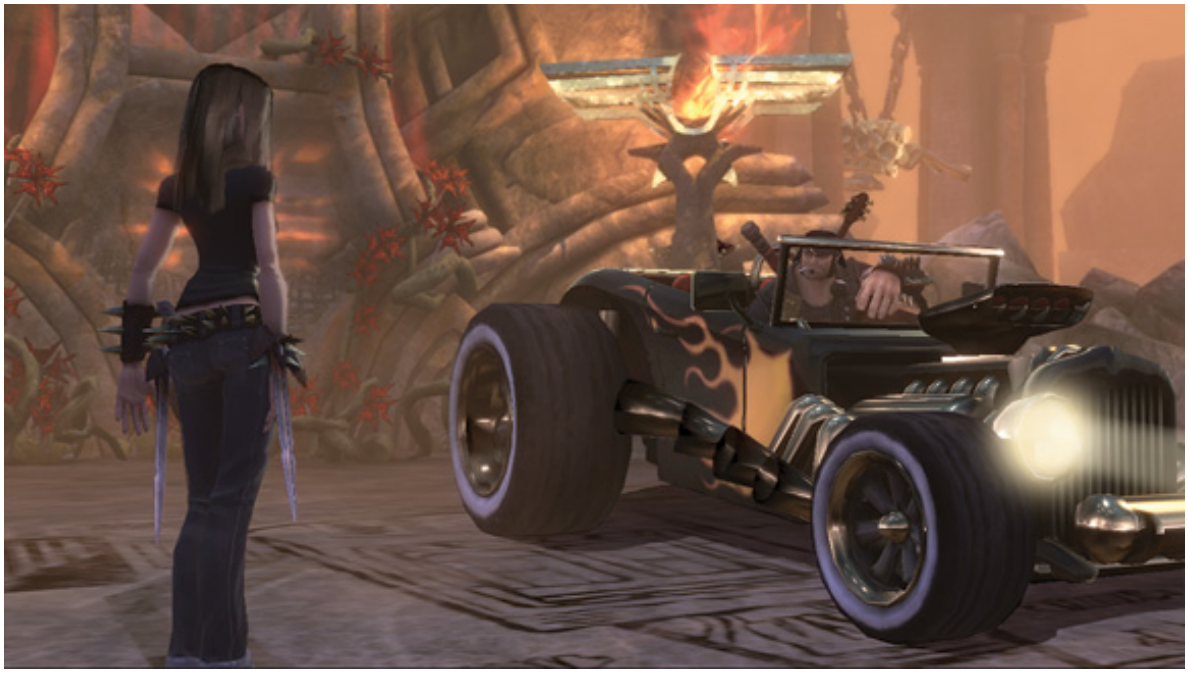
Though there were some false starts with other middleware packages, the decision to distribute the development risk across commercially available software packages was sound, and the call to integrate those packages early in development paid dividends throughout the development life cycle.

As beneficial as it was to rely on commercial software for various systems, Double Fine's batting average for selecting appropriate middleware was .500. We ended up selecting and integrating a number of middleware software

packages, and then later discarding them because they no longer served our needs as the project evolved. The middleware with which we ultimately shipped still demanded a tremendous investment in its integration, requiring far more staff than we had to handle the emerging stability and performance issues.

We were informed by multiple middleware companies that we were uncovering bugs and failures in the software that had not been found previously due to our placing new demands on their systems. That news was both flattering and unnerving, and ultimately costly to manage during stressful development periods. It was a tremendous relief to have the kind of tech support that we did from the individual middleware companies -- without their timely attention and desire to collaborate on our title, we would not have shipped the game at all.

While we already know we plan to continue to use certain middleware packages in future projects, we will be committing to the use of others much later on in the development cycle or writing our own software that specifically meets our needs. The wasted time spent integrating inappropriate or unusable middleware into our codebase, then reintegrating others or rolling out our own solutions, absorbed a tremendous amount of engineering time and effort, as did supporting and debugging the middleware we ultimately did elect to use through *Brütal Legend*'s release.



2. 11th Hour Tools

From the start of development, we recognized that good tools needed to be a priority to ensure faster iteration times for our team. A mix of legacy team structure, intrinsic complexity, insufficient input from content creation leadership, poor planning, and bad luck all contributed to a significant delay in the delivery of several critical tools to the team -- delays which jeopardized milestone deliveries, reduced artist productivity, and required unplanned emergency intervention at the expense of other engineering efforts.

Unfortunately, we seriously underestimated what it would take to get from our brightly colored Xbox *Psychonauts* graphics to the rich, highly-detailed lushness of our Xbox 360 and PlayStation 3 *Brütal Legend* graphics.

First, in an attempt to address the lack of dedicated tools programmers we experienced on *Psychonauts*, we created a tools group within *Brütal Legend*'s engineering team, and staffed it with four engineers. While this was an improvement over *Psychonauts*, the problem with this approach was that it facilitated the separation of tools work from the rest of the engineering effort, isolating the accessibility of the tools to the user from the process of delivering features for the game.

Often, this separation created inefficiencies in implementation, or a mismatch between the tools design and the

runtime feature it supported -- in several cases, this disconnect even led to sprints where a runtime team would complete an artist-facing feature without any tools support for it at all, making that feature (often a milestone deliverable to demonstrate) unusable.

A second critical mistake involved effective tools prioritization. Without strong guidance from content creation, the tools team focused initially on back end services such as tools deployment and build infrastructure, with the intent that these services would serve as a productivity multiplier on future tools development. Unfortunately, without any tools to distribute or code to build, these initial investments significantly delayed the development of content/game-facing tools.

Later on, we struggled to find the right balance between meeting short term needs and building tools that could scale to a full production environment. A prime example of the effect of this ineffective prioritization was the one year deployment delay of our MUE (Multi-User-Editor). An MUE would be a massive undertaking under any circumstances, but in our case the effort was made even more difficult by a late start and a rocky path from the quick and dirty short-term to the production-ready real version of the tool.

Then there was some bad luck. We decided to be an early adopter of a COLLADA-based Maya exporter pipeline. Initially, we underestimated how long it would take to implement a COLLADA 1.3-compliant parser. Then, when COLLADA 1.4 required a massive change to our schema, we had to significantly rewrite our parser.

Once it was finally functional, we were faced with ongoing maintenance or unexpected limitations, and ultimately failed to recognize any of the expected upsides, like easy interoperability or enhanced leverage of a tool or pipeline utility. On a lighter note, as a result the word "COLLADA" became a swear word and elicitor of groans at Double Fine Scrum meetings. In retrospect, at least it provided us with humor.

In hindsight, we should have created a programming team structure that enabled the creation of good tools alongside the development of runtime features, perhaps even allocating a larger percentage of the engineering team to tools early in development.

Moving forward, we intend to build our tools starting with the most client-facing work first, then iteratively improve them to meet the needs of the content creators. We also intend to de-emphasize technology investments that promise to yield large theoretical future benefits in favor of investments that speak to present needs.

---

3. Content Avalanche

Brütal Legend is not a small game. Fortunately, we thought we knew what we were facing and invested heavily in data/build infrastructure. What went horribly awry was that we both underestimated the total content push and, more importantly, didn't anticipate the huge content spike at the very end of production. From the start of the game through the end of 2008, both our rate of data churn and data growth were fairly steady and corresponded roughly to increases in staffing and team productivity. This was expected and planned and supported by the technology.

But then, in January 2009, everything exploded. All at once. After three years of development we had accumulated about 2.5 GB of optimized/packed game data. Less than four months later, we'd jumped to over 9 GB.

The central cause of this was a very large increase in asset delivery from a number of teams simultaneously. For example, we went from 0 localized files to about 100,000 in a matter of weeks.

We received the high resolution video assets for the Jack Black intro and all our main menus in one heap. We made a late decision to contract additional audio work, and new ambiences and sound effects were quickly added to the game. And so on.

This simultaneous significant increase across a number of types of content put a massive burden on our entire infrastructure, in particular our build machine, Perforce server, and network backbone. To exacerbate matters, we

started to see cascade effects -- where a massive hit to one system (such as a check in of 10,000 .wav files) would bog down Perforce, causing a bottleneck in all of the dependent systems (like our build server and individual check ins) and these bottlenecks would then cause other bottlenecks.

It's a credit to our pipeline and build infrastructure that things never failed, but we experienced a number of severe performance degradations, many of which required emergency interventions from the engineering team. This unexpected firefighting caused lost productivity, invariably at the worst possible times (like during preparation for a demo or a milestone).

These large content dumps also put significant strain on our runtime systems. The per-line memory overhead in the voice system was not prepared to handle tens of thousands of lines, causing us to panic about our ability to even fit or a dual layer DVD.

Across the board, these unexpected increases in content caused ripple effects throughout our IO, memory, and processing profile. And because the rate of increase was both high and unexpected, the engineers responsible for wrangling these systems were pulled from their assigned work and redirected to emergency fire fighting.

Moving forward, we will be much more cognizant about working with content creators to proactively estimate the total amount of data that they plan to create and to factor these numbers into our technical designs to ensure that we meet the final needs of the product. Additionally, we plan to invest more in scalable data infrastructure in the hopes that we can be better positioned to bring new capacity online quickly should it prove necessary. With those improvements and a little luck, hopefully content avalanche handling will be something we brag about in our future projects.

4. Facilitators vs. Implementers

As *Brütal Legend* moved into production, it became clear that the team was understaffed in key implementation positions. To meet this need, we reallocated positions set aside for facilitation hires (such as design and production) to staff up more implementers (such as animators and programmers).

While it was necessary to staff up on implementers, we failed to recognize until we were deep into production that our overall efficiency was reduced by this personnel trade, due to the statistical increase in designs that had to be reworked, an up-trend in management oversights, and general miscommunication.

The price of understaffing the design department meant that we were often implementing ideas that had only been

loosely discussed, before the feature had been fully planned. Serious flaws were sometimes discovered only after a feature had been partially or fully implemented.

This also meant that there was always a hefty backlog of decisions and specifications, even before adding in the rework required after false starts. Since it is often at the top of the dependency chain, bottlenecking design had a deleterious impact on the rest of the team, especially considering our broad game scope.

The same understaffing symptoms could be found in the production department. Producers are responsible for communication, for resolving dependencies, for ensuring the team works reasonable hours, and for keeping the project on schedule. They ensure that the road ahead is paved well ahead of the team's arrival. With insufficient staffing, some of the production responsibilities fell to the leads, or, absent that safety net, inefficiencies were handled with overtime and stress.

In retrospect, staffing facilitation and implementation positions should have been equally prioritized. Scoping the game to meet our capacity to create it would have almost certainly meant a smaller game, but one more likely more polished and smoothly managed.

---

5. Double Fine gets served

In June 2009, Activision Entertainment Holdings, Inc. filed suit against Double Fine, claiming breach of contract and seeking a preliminary injunction to stop the release of the game by Electronic Arts on Rocktober 13. Less than 2 months later, the case settled out of court. I can't talk about any of that in this article, or any article really. I bring up getting sued as something that went wrong because of the impact the transition between publishers and subsequent lawsuit had on the development of *Brütal Legend.*

Let's go back a little bit. We had been working collaboratively and successfully with various groups at Vivendi for two years until Vivendi merged with Activision and we lost touch with both publishers while a lawsuit percolated.

The merger announcement and subsequent diminution in publisher contact with Vivendi personnel, especially after such a previously harmonious relationship, caused internal unrest and morale dips among the team.

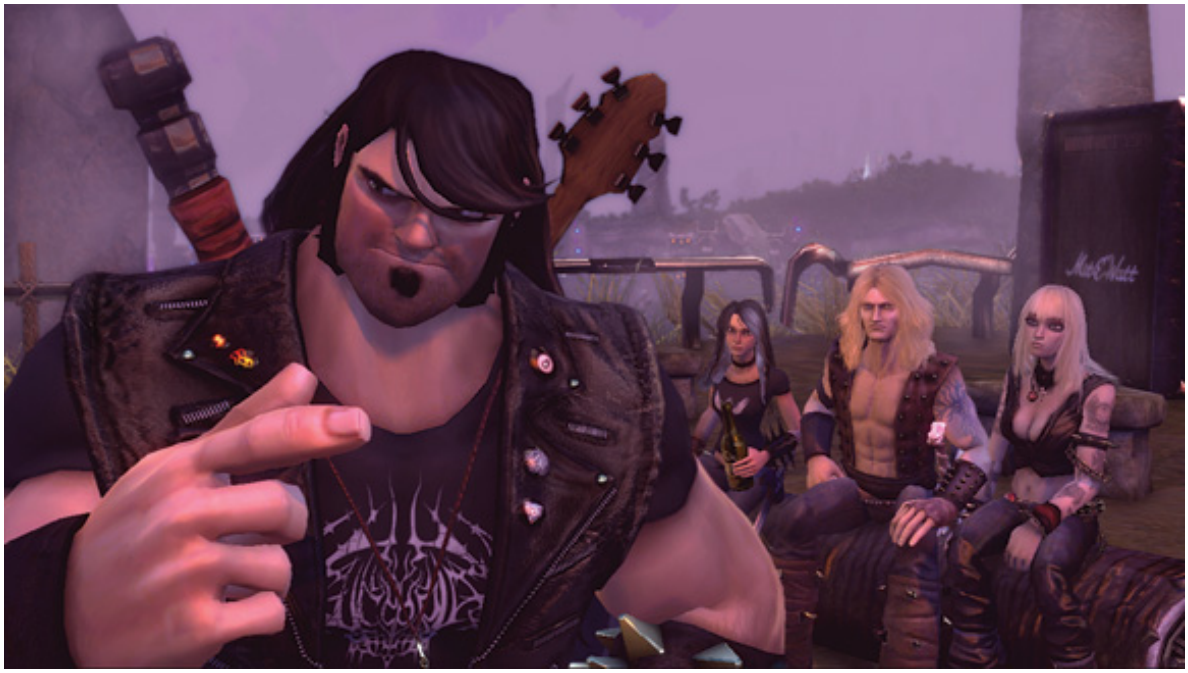
Company meetings often included frustrating discussions about what little we knew about the current situation at our publisher, and what the various possible outcomes would mean for Double Fine.

This demoralizing uncertainty lingered for months, during which time the leads continued to motivate the team to hit their scheduled milestones while watching our coffers run dry in the absence of any publisher payments.

We learned Activision was not going to be publishing *Brütal Legend* through an official press announcement issued by Activision that listed the games they would be shipping, with ours conspicuously absent. Again, the team was abuzz with anxiety -- and the official hunt for a new publisher began, distracting Tim, myself, and various team leads during an already intense development period.

Even after the game was re-signed with Electronic Arts, we enjoyed only a brief reprieve before the legal communications began among Double Fine and Activision and Electronic Arts. Most of the team was shielded from the drama that unfolded between December 2008 when Electronic Arts announced that they had picked up the game for publication and July 2009 when the lawsuit settled. But Double Fine's leadership was not, and the distraction and stress took its toll on individuals and on our deliverables.

The lawsuit was filed just as the game went Alpha, with a stipulation that it be heard prior to Gold Master being submitted -- relegating Tim and myself and a cadre of team leaders to the unenviable job of information gathering, declaration writing, lawsuit reading, witness interviewing and all around non-game-making during the crunchiest, most critical time of development. The lawsuit took its toll on the team, on the company, on our product and on our optimism. Wrong, any way you slice it.



Double Fine, for Metal

Everything in the game -- from the lore, to the locations, to the life, to the linguistics -- has been individually handcrafted by a Double Finer. 107 licensed heavy metal tracks from 75 different bands and countless heavy metal album covers inspire the game's creative direction.

The voices of celebrities Jack Black, Rob Halford, Lemmy Kilmister, Lita Ford, Ozzy Osbourne, Tim Curry, Kyle Gass, David Cross, Brian Posehn, and Wil Wheaton alongside the stellar voice acting work of veteran video game actors such as Kath Soucie, Zach Hanks, and Jennifer Hale bring the more than 150 original characters to life through 40,000 lines of dialog that were written, recorded, edited, and hand-integrated into the game.

Glenn Tipton and K.K. Downing personally wrote and recorded Eddie's, Ophelia's, and Doviculus' original guitar riffs. It takes the following words -- third-person melee, single player, multiplayer, open world, driving, RTS, adventure -- to describe how Eddie uses one axe, one guitar, and one hot rod to harness the power of metal to command armies and defeat evil.

In the end, 83 developers, propped up by the patience and tolerance of their families and friends as well as by the profound support of Electronic Arts, exemplified the utmost devotion to their craft and a fierce tenacity in the face of uncertainty and disappointment -- surmounting publisher plate-shifting, turbulent contract negotiations, lawsuits, misdirection and redirection, focus test surprises, and extreme excesses of ambition.

Armed now with a proprietary game engine, a robust tools pipeline, a talented and experienced staff, and the creative freedom and corporate mandate to innovate, Double Fine is well positioned to set forth on its next epic journey.

Game Data

Developer: Double Fine Productions, Inc.

Publisher: Electronic Arts

Platforms: Xbox 360, PlayStation 3

Release Date: Rocktober 13, 2009

Number of In-House Developers: 74

Number of External Contractors: 9

Budget: $24 million

Development Time: 4.5 Years

Typical Workstation: Dell Quad Core Xeon, 4GB RAM, 150GB, NVidia GeForce7800, Windows XP

Software Used: MS Visual Studio 2005, Autodesk Maya 2008, ZBrush, Adobe Photoshop, Adobe After Effects, ProTools, Sound Forge, Perforce, Microsoft Xbox360 XDK, Sony PlayStation 3 SDK, Internal suite of development tools

Total Lines of Code:
Game (C++, excluding middleware): 554,736
Game (Lua): 42,745
Tools (C++ & C#): 52,263
Tools (MEL): 85,294
Tools (Python): 34,079

Total Number of Perforce Checkins: 164,863

Return to the full version of this article