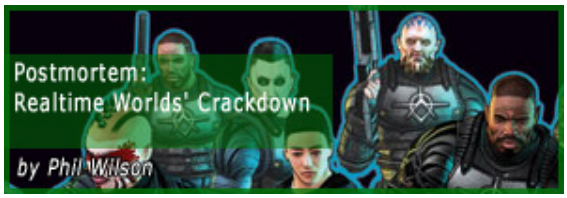
Postmortem: Realtime Worlds' *Crackdown*

By Phil Wilson



[In this Gamasutra bonus feature, published online for the first time in honor of Realtime Worlds' upcoming APB and spinoff studio Ruffian's Crackdown 2, we reprint Game Developer magazine's October 2007 postmortem of RTW's first project, the Xbox 360-exclusive open world action game Crackdown.]

Crackdown is the first title released by Realtime Worlds, an independent developer based in Scotland. Over the course of the game's turbulent development, the company grew from a small team of former DMA Design staff headed by Dave Jones, to an award-winning studio of 170 employees housed in a 30,000 square foot office. In addition to this main studio in the heart of Dundee, Scotland, two more have been set up in Seoul and Colorado.

I was producer of *Crackdown* from almost-but-not-quite the very beginning. Like many projects, the development cycle was a rollercoaster of highs and lows, but despite some truly gut-wrenching sensations, it was a fantastic experience.

What Went Right

1. Art Vision

The most immediately striking element of *Crackdown* is certainly its visuals. The project was blessed from the outset with a highly focused and creative art director who defined the perfect style to frame a dialed-up world of intense superhero action from a very early stage. His goal was to create a rich color palette, unique and stylized ambient shadowing, crisp and strong real-time shadows, exaggerated assets, and bold outlines on all the geometry.

We used various reference materials to collectively encapsulate this vision, but the most significant one was a none-too-mainstream Manga title: *Blood: The Last Vampire*. The Microsoft marketing team was initially quite nervous about the visual direction. It wasn't "safe" and was sure to consume valuable PR cycles explaining the game's style rather than substance.

Microsoft at least wanted a catchy handle to hang it all on, but we didn't take them seriously enough, belligerently referring to the concepts as having a graphic novel style. We eventually paid the price when even the specialist press heathenishly branded our lovingly crafted form as cel shaded!

Though we established the visual direction early, creating it was a far more arduous process. The few graphics programmers we had were preoccupied with relatively rudimentary rendering requirements for far too long.

By the time we reached the production phase, our vision was still a fragmented series of tech prototypes. The actual game, as the then-Microsoft art lead delicately put it, "still looked like ass." Everyone, not just the publisher, became worried that we wouldn't hit that all-important visual bar.

Then came the X05 event in Amsterdam. Microsoft wanted to tease second-wave titles at its pre-launch event. We weren't ready, and everyone knew it. The run up to the show was highly charged at all levels and ultimately, despite producing a relatively solid and promising demo, the wrong decision was made in taking the game to the event.

Not surprisingly, *Crackdown* was announced to lukewarm reception; but then it was on the media's radar, and from that point onward, we were subject to repeated requests for screenshots that served only to hang around on the net like a bad smell.

Behind the scenes though, the rendering tech was at last on track. The unsung hero feature was ambient occlusion, an ambient shadowing system embracing the principles of radiosity lighting in a proprietary 3DS Max tool. The ability to sample millions upon millions of photons in the scene at a fraction of standard radiosity calculation time resulted in an unprecedented level of environmental solidity, with darkness forced into corners for increased dramatic effect.

We made some last minute sacrifices in the pursuit of performance, such as heavily simplifying water reflections, but the result was still stunning and the massive sea change in opinion for the finished product was everything we had hoped for.

In fact, shortly after *Crackdown* shipped, Microsoft conducted a thorough consumer survey that finally vindicated everyone's efforts with one simple fact: Graphics were rated as the number one aspect of the game.

---

2. Marketplace Demo

"The *Crackdown* demo is like crack!" This comment came from the Microsoft user test lead, and no statement could motivate a team more.

Whatever you might think of the general quality of content, Xbox 360 Marketplace is a great piece of design. For a game like *Crackdown*, which we knew most gamers would enjoy if they'd just pick it up, the opportunity to freely distribute the demo to everyone with a broadband connection was a golden opportunity.

We began working on the *Crackdown* demo roughly five months before the game's completion. This early work mostly consisted of creating an infrastructure to support an alternative build and data configuration, meaning creating a special demo-only code branch could be deferred for as long as necessary.

We heatedly debated how much of the *Crackdown* experience we should present for free in the demo. Until we heard some reassuring user

test feedback, many of us were concerned that Realtime Worlds might only be remembered as those crazy guys who gave away the farm.

In this case "the farm" consisted of roughly a quarter of the total game environment, one-third of the game missions, and a generous 30 minute time limit that only kicked in when the player reached a certain level of progression.

Conversely, the decision to include hugely accelerated skill levelling was unanimous because we all accepted that the biggest hook came from just a taste of a fully evolved agent's capabilities.

Crackdown consists of only one level, but unfortunately, it eats up more than 4Gb. Not wishing our demo download's girth to scare away potential players, we cited our maximum as that of previous demo heavyweight -- *Project Gotham Racing 3*, which clocked in at an impressive 1.4Gb.

By cutting all the audio, video, vehicles, and high LOD environment blocks that we knew couldn't be triggered without leaving the demo district (and then painstakingly replacing those we were wrong about) we eventually hit comfortably below this target.

To the knee-jerk outrage at the news of *Halo 3*'s Beta attachment to *Crackdown*, the demo was the perfect antidote. Instantly well received and exploding to top honors in the 'most downloaded and played 360 demos ever' list, it was clear that *Crackdown* was going to enjoy the success we all reckoned it deserved. The gravy came when the team's aspirations for 'Warthog' style fan videos were also realized as hundreds of spectacular sandboxing stunts vied for space on YouTube.

3. Trust

As you'll see in "What Went Wrong,"*Crackdown* was in many ways dealt a poor hand and was forced to play badly until at the last moment, it threw down a Royal Flush. In other words, there was an enormous strain on the publisher-developer relationship until just at the eleventh hour, belatedly, we delivered the game in full and on the initial promise.

At the same time, it would be misleading of me not to say that the relationship between Microsoft Game Studios and Realtime Worlds was simultaneously and paradoxically strong. There were certainly some individuals within Microsoft's production team who were passionate, driven, and great to work with.

Additionally, the wider first-party family (traditionally our competitors) were only too happy to help out whenever the need arose. But without one key element, *Crackdown* might not even have been conceived -- and that's trust, albeit the anxious sort, like the trust a parent gives a child when handing over the car keys for the first time.

I've learned the hard way that projects become exponentially less predictable the more unique, groundbreaking features they take onboard. Despite enormous potential for *Crackdown* to be a shipwreck, senior management on either side of the Atlantic imbued their teams with enough freedom for the project to push back the boundaries of an urban action environment, creating a home to an incredibly engaging sandbox experience. Without belief at all levels, games like *Crackdown* simply don't wind up on a store shelf.

---

4. Passion

A good development team is much more than a group of talented people. There needs to be a palpable sense of drive, healthy competition, and synergy such that the sum effect is greater than the individual parts could ever achieve.

The *Crackdown* team had a reassuring buzz about it. People regularly congregated to see what recent progress was causing a stir before dispersing with an increased sense of combined purpose and accomplishment.

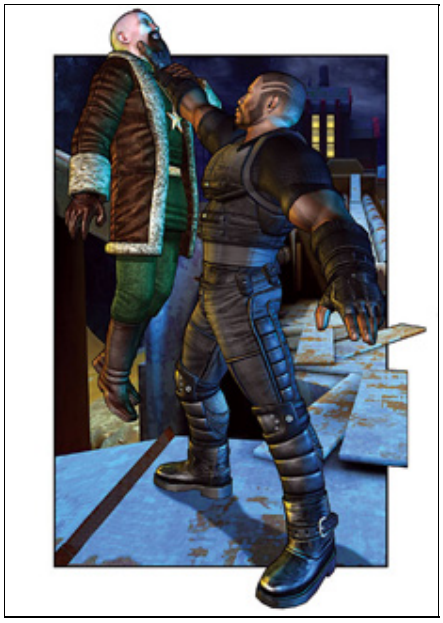
Even during the inevitable crunch, when particularly pooped and still facing an overambitious product scope, our indefatigable heroes pressed on, concealing content from the axe man by providing refuge in special extra-curricular projects (strictly speaking not always to be encouraged).

If there were a single source of fuel for this teamwide thrust, it was the knowledge that the initial prototype (and subsequent first-playable demo) was a belter.

Video games are creative art, and the best developers are inherently passionate about crafting them. However, there's a difference between developers who willingly give overtime, and managers who demand it.

I admit that at key points in the project, I did strongly urge the team to invest more than their contracted hours. Under the circumstances, and with predominantly the best interests of the project at heart, there was no other option. Unfortunately, this kind of demand subtly degrades the team dynamic and causes resentment.

Crackdown's total crunch period varies depending on the definition, but is widely agreed to have lasted far too long. The situation flew directly in the face of Realtime Worlds' first commandment -- Thou Shalt Not Abuse Thine Most Valuable Assets -- which is actually a doctrine we're better positioned to adhere to more religiously since *Crackdown*'s success.



Toward the end of the main project, the crunches became steadily more pronounced and, though we reached the mess hall just in time for Christmas brandy and cigars, it also allowed everyone to forget the commitment, professionalism and, above all, passion that came before.

5. Downloadable Content

We tried to plan the additional downloadable content before we finished the main game, but ultimately the pressure to focus on the project at hand meant it never progressed beyond a few conceptual discussions.

Even in the final release phase, where managers could do little more than buy pizza and mop brows, the key architects of a solid plan for new content were lost to an intense program of promotional video production.

In early February, after extensive and undeniably well deserved holidays, and with a thorough plan in hand, work finally began on the new content with roughly half the original team. Just three months later, the package was submitted to certification. In the shadow of a monumental four-year project, 12 weeks sounds almost inconsequential.

The reality, though, was that we finally had a stable technology base, and we all had experience working both with it and each other. Not only that, but a sudden boost in efficiency reminded us that check-in logjams were just one reason why we had regularly been pining for the halcyon days of smaller teams.

The greatest key to success for the downloadable content was that the team drove the scope. Naturally, there was some input and guidance from the stakeholders, but the targeted features and content were ultimately derived by the only people who really knew what was possible and worthwhile within the timeframe.

What Went Wrong

1. Scope and Change Control

We got a couple of milestones past our first playable demo (which went down really well but was held together with paperclips and sticking plasters) before feeling the sinking sensation that the project was out of control.

In the face of a great deal of pressure to continue making incremental progress, the production team agreed to immediately derail the entire team from chasing short-term disconnected goals and instead embark on a full-scale project audit.

Much of the team was tasked with breaking down the complete design with implementation information so that it could be processed and converted into a simple and solid plan.

A multi-disciplinary panel was charged with reviewing the extensive detail of each component in turn so there could be no more complaints that, for example, audio wasn't being given due and timely consideration.

With a pass to add development estimates, resource groups, and stakeholder priorities, the resulting project scope spreadsheet was incredibly cumbersome. The team at large resented it bitterly. Unwieldy as it was, it was still an invaluable tool for at last conducting a meaningful triage of work against available time and resources.

Another shocking reality was that the number of coders yet to be hired was expressed in the order of tens. In a pragmatic development

environment there would have been three choices: 1) increase the budget, 2) lower our ambitions, or 3) pull the plug. Unfortunately, the stakeholders were far from pragmatic, and rather than moving to jettison anything that wasn't in direct support of the clearly defined "project pillars," they used the scope discussions as a forum to add yet more ideas.

Several deep cuts were eventually approved (some creeping back in over the remainder of the project), but it still wasn't enough. At least we finally had a handle on the stakeholders' definition of "minimum content."

2. Renderware Graphics and Changing Hardware

Changing the target platform caused serious problems for *Crackdown*. Over the course of its four-year development, *Crackdown* moved from PC (where it was prototyped), to Xbox (where it was initially intended to stay), back to PC (in preparation for move to Xbox 360), to Xenon Alpha, then Xenon Beta, and at last to Xenon/360 Final.

Even on the final hardware, we continued to take hits from significant system software updates every few months. When at last the platform stabilized during the last year of development (post hardware launch), development efficiency increased massively.

With a relatively small development team to begin with and difficulty in recruiting experienced staff, the policy was to bring in middleware solutions wherever possible in an effort to reduce development time. By far the most significant of these was Criterion's Renderware Graphics and Studio.

However, we hit the wall full speed when the project shifted from Xbox to Xenon. Criterion opted not to support Renderware Graphics 3.7 in the transition, forcing us to move to an early beta version of 4.0. Not only was version 4 an unfinished product, but some features that we had come to rely on were not (and would not be) present at all (though in some cases Criterion did provide special code).

Since the middleware was suddenly a work in progress, each update came with a whole new set of bugs, transforming us overnight into hapless beta testers. Technical documentation was insufficient and in some cases inaccurate.

In hindsight, we realize that we simply did not adequately investigate the suitability and potential pitfalls of this new version of Renderware. Had we done so, we would have known that the correct decision would have been to back out and replace it all with our own technology. Yet with mounting pressure from the publisher to get on Xenon Alpha hardware (in itself a mistake given its tenuous relationship to the Xbox 360 proper), I'm not convinced the politics would have allowed it anyway.

Crackdown was already in development hell when it was blindsided by Electronic Arts' acquisition of Criterion. Initially, the EA takeover was transparent, but before long the well of excellent onsite support dried up, and we had no option other than to become the world's leading experts in the middleware.

Throughout the Xenon development phase, we had to take new versions of Renderware 4 in support of latest system software releases, which were never backward compatible. Eventually, we stopped taking new versions and rebuilt it ourselves. At long last we had a stable platform.

---

3. Renderware Studio and Data Pipeline

Not to dance too lively a jig on Renderware's grave (if only because the people at Criterion were nothing but professional and courteous at all times), but the *Crackdown* asset pipeline issues began and ended with Renderware Studio.

The initial evaluation of Renderware Studio's capabilities via the available demos was very promising. Unfortunately, we learned too late that these did not scale nearly so well to major next-gen game development. A lot of data that might otherwise have been part of the source code or simple text files was stored in the XML database.

As the project grew larger, the task of synchronizing Renderware Studio data with source code became more difficult, with changes to source code often requiring corresponding changes to Renderware Studio data.

Worse, a dependency between the structure of the XML data and the contents of the source code header files meant that code changes would regularly break the development environment for the artists and designers.

We were forced to introduce a more onerous check-in system that attempted to reduce the risk of one person single-handedly screwing over what became a very large team, again radically impacting efficiency.

The start-up time and general performance of Renderware Studio suffered heavily as the volume of game data steadily increased. Data sorting and searching was also inadequate so, as the structure became increasingly more complex, changes became correspondingly more dangerous. The inherent design of the system made it impossible to fix this.

When we started porting the game to Renderware 4, it emerged that the matching new version of Studio would not be ready in time. We opted to stay with the old version of Studio by means of some judicious hacking. We were then stuck with old unsupported software and a halfway-house build pipeline. We wasted a great deal of time making two Criterion products work together.

Renderware dominated our tool chain. The testing of a single asset could take upward of an hour, directly impacting productivity and

indirectly impacting quality since it naturally discouraged regular testing. One ongoing frustration was that the 3ds Max exporter source was never made available to us, preventing any modification and ultimately leaving us to write an alternative exporter.

Internally, we made various pipeline improvements, largely focusing on the automation of batch asset exports and subsequent error checking. Despite this, our final workflow was massively slower than the comparable environment of Unreal Engine 3. The required pipeline improvements were well understood, but the scope of overhaul and inevitable teamwide disruption made them completely non viable mid-project. We had no choice but to limp on to the finish line.

4. Cooperative Mode

Co-op mode certainly wasn't something that we did wrong, far from it. The implementation was spot on and the result was phenomenal. Unfortunately co-op mode was something that, during development, went wrong, repeatedly.

Crackdown was always designed to be a multiplayer experience. The plan was to make arena games running client/server for 8 to 16 players. The first playable demo delivered on Xbox featured a crude four-player version of Stockpile (a competitive game mode that was eventually shipped in the downloadable content pack). Though the game was great fun, the number of NPCs and interactive objects within even a limited arena were sub par for the intended *Crackdown* experience.

Then came a big push from Microsoft to include co-op in every title. After some fervent head scratching, we concluded that a traditional lock-step solution would be ideal for *Crackdown*, provided the number of players did not exceed two. Transmitting only player data meant that the bandwidth requirement would be low and the potential for maintaining an experience that felt pretty close to the solo game was very appealing.

The only potential stumbling block was that the game had to be tolerant of roundtrip latencies of up to 200ms. In this worst case scenario, the game's response to control input would be lagged for that same amount of time. The situation was mocked up and although noticeable, was acceptable.

Central to this method was a fully deterministic implementation of all game code. Initially progress was good with only the occasional bug, but soon the mounting pressure caused a knock on of programmer errors (un-initialized data, linking game logic to render data, random number issues).

All sync bugs were Severity 1 but needed to be tracked sequentially with no way of knowing how many lay in wait. For a long time, the soul-crushing task of tracking these down fell to just one person whom nobody envied.

Mass scale sync-locked development will always be difficult, but we should have planned to share the burden of identifying the bugs, if not to improve understanding of the key contributing issues, then at least to maintain coder sanity!

---

5. Unrealistic Targets

Initially, Xbox *Crackdown* was slated for spring 2005, and this always looked tight. In early 2004, when it was first proposed that we aim to get

the game on the next Xbox, we reasoned that our team size would be about 15 to 20 people short.

The concern from the Microsoft people with whom we were dealing was that the game did not yet demonstrate adequate progress to warrant an increase; in fact it risked being canned as a result.

Thus we soldiered on toward a revised spring 2006 deadline that was determined by virtue of the fact that "adding another platform would surely take roughly another year."

During 2004, the project audit took place, and then we ran headlong into the issues of destabilizing our core middleware.

It was certainly a bleak time, and if anything we were drifting away from the kind of demonstrable progress that would leverage the budget to staff up and minimize what was already a guaranteed slip.

In 2005, we at last transitioned to Xenon, and by March, when a new and more bullish Microsoft producer was brought in, we were already starting to make real progress. The new external producer was still a blessing though because he finally managed to kick open the door to new resources, albeit predominantly via contractors.

Under the circumstances, new recruits came on board pretty quickly, but it still takes time to find quality employees (and we refused to lower our standards to fill vacancies faster). As the hiring wore on, we compensated for the shortfall by attempting to find more people and ultimately suffered from having too many cooks in the kitchen and saw diminishing returns.

After the X05 event, the project was public and thus far less susceptible to being knocked down by a business decision. The game's profile was subsequently raised within Microsoft, and additional resources became far easier to appropriate. However, Realtime Worlds could not put itself in the position of having too many employees after the game was released and so refused to staff up any more than was planned at that time.

Microsoft has a policy of reviewing all its titles one year out (OYO). *Crackdown*'s OYO was in October 2005. In our view, progress and control was sufficient to agree that we were indeed looking at a completion date approximately one year later. Microsoft, on the other hand, "preferred" that the title hit before financial year end: We were faced with the unenviable task of driving toward a fairly hopeless May 2006 deadline.

As May approached, the spotlight moved to August, and when that evaporated, the sights were set on a tight but realistic October deadline. Eventually we slipped past our original OYO estimate by two months.

This Little Piggy Went to Market

Even after so many fires, *Crackdown* enjoyed the critical and commercial success that we all hoped it would. We like to think that a recent Develop Industry Excellence award for Innovation cements the job as being well done.

The finished product is cast firmly in the sandbox mould. There's no denying it lacked a little in directed content, but where it excelled was in handing over the 'Keys to the City', or, to use Dave Jones' favorite analogy, "providing a giant chemistry set."

Game Data

Developer: Realtime Worlds

Publisher: Microsoft Game Studios

Platform: Xbox 360

Team Size: 71 at peak

Development Time: 4 years

Outsourcing:

Character LODs: Nikitova, Kiev; 60% of gang vehicles: Valkyrie, Seattle; 40% of props: Ketsujin, Kiev; Cut scenes: Realtime U.K., Blackpool; Motion capture: House of Moves, Los Angeles

Release Dates:

Feb. 20, 2007: box copy

May 11, 2007: downloadable content

Return to the full version of this article