GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: Overhaul Games' *Baldur's Gate: Enhanced Edition*

By Trent Oster

How did a PC classic get reborn as a multiplatform "enhanced edition"? Founding BioWare developer Trent Oster takes us behind the scenes of the development of the Baldur's Gate remake, and all of its complications.

I'm Trent Oster, President of Beamdog / Overhaul Games. We're here to talk about the development of *Baldur's Gate: Enhanced Edition*. *BG: EE* (as we like to call it) launched on PC on November 28th and on iPad on December 8th to great fan interest, rocketing to the number 2 spot on iTunes in the U.S. and number 1 in many other App Stores around the world.

Our initial plans were to launch all platforms as close to each other as possible; in retrospect, this was very naive, given the size of our team and the volume of work required. We're working steadily now to build versions for the remaining platforms and to roll out concurrent versions so cross-platform multiplayer can be a reality. We've had great success on the PC and iPad, with great sales and positive feedback from the fans. We're very anxious to roll the remaining platforms out as fast as we can.

From the trenches, the development of *Baldur's Gate: Enhanced Edition* was an interesting journey. We had early moments of exuberance as we played the first tablet version and proved our theory that *Baldur's Gate* did indeed kick ass on tablets. We had great moments of insight when we brought people from the modding community in and shared work-in-progress versions with them. We had moments of despair, such as the lost source art and subsequent sacrifices to salvage the deal. We had moments of great distress, such as when we negotiated the September-to-November contract extension for the game. Every day that we had to wait for approval of the new deal terms so we could tell the fans was like torture. We worked incredibly hard to come to terms quickly, but until we had sign-off we couldn't announce the schedule slip.

We had some great moments as a team, such as the first time we saw the new UI scheme in the game and the first time Sam's new music played in an area. It has been a great journey for us and we've formed a strong core team of thirteen people who are capable of some great works. Along the way we had the opportunity to work with some great talents such as Mark Meer, John Gallagher, and Sam Hulick. We've had the opportunity to leverage all the great work from the original title as we try to build something better, something familiar, yet improved. In short, *Baldur's Gate: Enhanced Edition* was a challenge, but we've all come out of the development with some sanity remaining and a great understanding of the game, the engine technology and what makes *Baldur's Gate* the legend it is.

What Went Right

1. Fans and Community

Early on we knew one of the key elements to succeeding with the*Enhanced Edition* of the game was to bring in as many people from the modding and fan communities as possible. By working with them and listening to them, we could make the right changes and try to improve the game to better meet their needs. Our first few attempts were crude -- an invite-only Reddit, some direct e-mails -- but things rapidly turned around with the launch of the official message boards where we could speak directly with the modders and fans and listen to their concerns.

We shared new content with them as early as possible and acted on their feedback to make the game a better experience for users and a better platform for modding. We did this through hosting a long-running private beta. We would push a new build out to the beta group, have them poke it and respond with suggestions and feedback.

Our core community firmed up quite quickly, with a number of key people stepping forward with well thought-out criticism and thoughtful solutions. We read all the feedback, but we prioritized responding to the in-depth feedback and the community quickly oriented around that model, understanding we give the attention to those who put in the effort. The private beta was composed of hand-recruited community members from this active thoughtful group. We will be forever grateful to all the people on the forums for all the effort they contributed on our journey to make a better *Baldur's Gate*.

---

2. Previous Working Relationships

Once we had surmounted the initial hurdles in the deal and convinced the involved parties that*Baldur's Gate* was indeed worth revisiting, we were able to bring all the involved parties together and sort out the core terms.

The deal was a bear simply because of the number of parties involved: Atari, Hasbro, Wizards of the Coast, BioWare, EA and us (Beamdog). We managed to get the terms sorted quickly through great former relationships. It actually took longer for me to get through the wall at Atari and talk to the right people than it took to put the core deal together. I remember meeting the fellow who had shut me down for a few months at GDC once we had the deal underway. He offered, "You must hate me," when we met in person, to which I responded, "Not anymore."

I'm very happy to have had some great friends such as Richard Iwaniuk, Greg Zeschuk, and Ray Muzyka, who were exceptionally helpful in getting terms agreed to by the involved parties and signed off in a reasonable timeline. Richard was able to line up the key people on that end and quarterback the requirements. Derek French was also a great asset at BioWare, as he assisted us in our spelunking for code and art assets throughout the course of the development. (Key lesson here is without someone on the ground running a contract in each group, it just doesn't happen.)

I'm very happy to have worked with many great people in my former life at BioWare. During my 15-odd years there, I started as one of the original founders, worked on design, programming and art on *Shattered Stee*", led *Neverwinter Nights* from a one sentence description of "everything in the D&D Box" through seven years of development and two expansion packs. I served a tour as director of technology and returned to my first love, directing video game projects, before leaving in 2009. One of the great things about the game industry is the amazing people involved in it and I look forward to "paying it forward" whenever and however I can.

3. The iPad and the Touch Interface

We proved you could produce an epic, massive RPG in the tablet space, and the market was starving for such a project. The success has been awesome and the feedback from the fine fellows at respected media outlets like Touch Arcade and Pocket Tactics has been a real positive.

Beamdog co-founder and veteran developer, Cameron Tofer, went crazy on the tablet interface early on. We had an iPad version running and semi-functional almost a year before we shipped. Every new build brought radical changes as we explored the logical means to perform actions in *Baldur's Gate* using a touch design paradigm.

In the end, Cameron's approach broke down to favoring common actions and giving them the least complex control mechanism. In*BG*, the most common action is to scroll the screen, so we used the photo app UI standard of a touch drag for scrolling. The second most common is moving/interacting, which we mapped to a single touch. We initially favored group selection, but after a number of clumsy iterations we realized the vast majority of actions in *BG* are performed with either a single character selected or the entire group, and as such, group selection could be moved to a modal button. When you play the tablet version of *BG: EE*, it is the constant iteration you feel.

Click for larger version

We've added to this level of polish recently with a further enhancement we call smart radius. Smart radius allows the user to have the precision of a mouse with the less accurate nature of a touch interface, making it easier to enter buildings, pick items from the ground and so on.

We'll continue to iterate as we go, making the game better as we move along. From an overall UI design, we wanted to keep the original feel, but re-base around the new resolution of 1024x768. My priority in UI was to make the user portraits as large as possible. We listed all the required UI elements and slowly re-constructed all the art and panels around the new portrait sizes, we also tried to embrace some UI guidelines from Apple for the iPad, trying to keep buttons over 44 pixels in height (we only failed in a few spots like class selection). The overall process was challenging, as we tried to balance re-writing the entire UI code while keeping the existing complex (and overly integrated -- damn you, Scott Greig!) systems working.

---

4. Beta Testing

We beta tested the game for over six months. Throughout this time, the feedback on how to improve the game was tremendous. We found and fixed bugs at a rapid pace, and the experienced modders in the beta were able to suggest fixes which cut our development time.

In our mind, the beta was a core element to the success of the project. The feedback on our new content was excellent, as we found bizarre edge cases that caused the quests to break in unexpected ways. Due to the complexity of *Baldur's Gate*, we needed a lot of testing to find all the possible ways a quest could be broken and act quickly to fix the problem and re-test.

By sticking with the original game structures and asset formats, we were able to leverage the knowledge of a community of developers and not just the direct efforts of our team. As mentioned above, this beta testing group was a volunteer group recruited from our forums. We chose people with strong skills and a good background with the original game and as such, the feedback was fast and on-target. The beta team had access to our bug database and beta bugs went straight into the system.

We also hired an external testing company, iBeta, to help us final the iPad versions and to ensure testing coverage. They were able to run us through a "pre-certification" style testing to catch the differences between the different iPad hardware devices.

5. Our Own Digital Distribution Service

Beamdog is two parts; an online digital distribution service, and our game development team, which we call Overhaul Games. We developed the Beamdog Store with the goal of selling our games directly to our users -- no middlemen, no confusion on support, just a developer and a customer. The direct-sales option and a preorder program allowed us to receive payment many months in advance of a typical royalty-advance deal, which was a great benefit for a small, self-funded developer.

The other great benefit was how our digital distribution service allowed us to push out beta builds of the game six months in advance and update testers to new versions without a hitch. We were able to push PC fixes to the game in record time, doing four major updates in under two weeks as we found issues and corrected them. Sure, we had a few hiccups with the client software and out-of-date SSL root certificates on user systems, but the system performed exceptionally well.

We were able to add servers in Germany, Singapore, and on the East and West coast of the U.S. without a hitch as we approached launch, and we handled a huge surge of demand as we pushed hundreds of terabytes of data to our users. We use Beamdog daily internally for all our build distribution, and without it we couldn't have been nearly as agile in our response, effective at version control, or as fast at build distribution. We even use our service to send around Mac OSX, iPad, and Android development builds of the game. We've been very happy with our service to date and we'll be expanding on what it can do going forward.

What Went Wrong

1. Epic Code Complexity

We began work on the project almost two years from ship. At the start we planned to "ninja in, make some minor changes, replace the graphics system and ninja out." As we began to work with the code base, we found severe performance and stability issues with the engine. We traced some issues back to the core threading model of the Infinity engine.

Infinity was multi-threaded before there were processors capable of parallel code execution. As such, it was designed around a concept of threading that never really emerged. The main issue was that all of the threads shared the same set of data. The problem is all these threads were created and they all hit the same memory and as such, blocked each other, stalling all threads until the current one completed.

The code was also heavily laced with critical sections, which caused (in some cases) 70 percent of execution time spent in no-ops. Our long-term solution was to machete in and remove the threading, removing a couple hundred thousand lines of code from the game. The end result was more stable and had fewer performance irregularities. The "ninja-in" approach was tried in other areas, as well, and every time the end result was consistent. We always found a ton of complexity solving an era-specific problem that no longer applied. We continue to find roadblocks in the code and we're improving it as we go.

2. Intricate Code-Data Dependency

I have a development joke I often drop: "When a programmer believes he/she is being clever is when they create the greatest atrocities." The Infinity Engine is chock full of clever. For example: to render a character to the screen, the correct frame and orientation of a character sprite is first loaded out of a resource file using a very heavy resource-management system. The sprite is then color-mapped using a 256-color palette swap to enable player colors. Following the re-mapping, any number of further palette-manipulation code (stoneskin, anyone?) can step in and further change the actual data. Then the sprite is rendered against whatever potentially covering elements are nearby.

The final result is sent to the screen in 64x64 pixel tiles to be rendered. The entire system runs under a dynamic update system that flags 64x64 tiles as updated and renders them or, with no change, leaves the tile from the previous buffer. The volume of clever shifts and tweaks along the way make it nearly impossible to track down all the ways in which a simple sprite can be manipulated. In some cases, the data can reference many different .2da data files on how it can be manipulated, from equipment-changing animation frames to a data redirection to

render a dwarf sprite instead of the default human. Complexity is par for the course in an RPG of this magnitude, but the intricate linking of assets and code really limited our ability to make the architectural improvements we wanted to make.

3. The Lost Source Art

We had drafted the original deal in the context of a *Baldur's Gate: HD*. Our plan was simple: grab the original artwork, clean it up, re-render it at higher resolutions and with better materials, thus creating stunning versions of the areas everyone remembers. We planned to take the character models and re-render them with many more frames of animation and add new orientations to the movement to make the game smoother. We nailed down the core terms, got everyone on the same page and we got our first drop of the assets from BioWare.

A few days later we noticed a large hole where the source art should be -- stuff like 3DS Max files and texture images. "No problem," I said; I contacted Derek French over at BioWare and he dug further and sent us more data. We again dug through and failed to find the source art. I made arrangements to visit BioWare with a removable drive and work with Derek and the IS department to find the assets.

After two days of searching we came to the horrible realization that the source artwork was stored on a departmental drive and not a project drive, and as such was not frequently backed up. We dug through tape backups to no avail. The source art was lost.

At this point we brought the information back to Atari and the deal was dead in the water. Cameron and I spent a few weeks re-thinking the concept and we re-pitched Atari with an "Enhanced Edition" as opposed to an "HD" version. We discussed the implications of a non-HD version and we had to renegotiate royalty terms to get the deal back on track. We basically had to completely discard our plans and start anew after almost a year of negotiation.

Click for larger version

The end result is a little less graphical flash for launch than we had initially planned, but we actually had a bit more time to make more widespread improvements as a result, so the project is different; in some ways larger, but still pretty awesome. The key learning here is: don't panic and always have a towel, because you never know when you are going to be surprised... and towels can be very useful.

---

4. OpenGL Support on Intel Integrated Chipsets

We went into development on *BG: EE* with a clear goal to target the OpenGL 2.0 specification (from 2004) for all functionality, to ensure support on all devices. We did some quick testing on PC, on Mac, on iPad, on Android, and passed on all counts. Raspberry Pi, check. Raspberry-freaking-Pi OpenGL 2.0 support, check.

We expanded our beta testing group (and quite possibly due to the enthusiast nature of the participants in our beta) we had very few performance issues reported. (In retrospect, it makes perfect sense that gamers would elect for enthusiast video cards and, as such, have solid OpenGL support.) Close to ship we saw our first few reports of slow performance on Intel integrated hardware, specifically in older laptop systems. We did some digging and found many of the laptop vendors had not updated drivers in many years.

During this timeline, we shipped the game on PC. We started getting more reports of poor performance and graphic problems. We managed to find a few workarounds --such as force-updating to the latest Intel drivers (despite vendors not certifying them). In the end of our research, we came to a horrible conclusion. Many of the older Intel integrated video cards did not have viable OpenGL 2.0 support. In fact, many did not have viable OpenGL 1.0 support.

As a developer, we hit a hard situation. We've spent our development budget (and then some) and we have a portion of our user base who can not enjoy the game they paid us money to play due to a driver issue. We have a solution in mind and we will be spending further,

significant, development resources to re-write the entire rendering system for *BG: EE* around the missing functionality in the problematic drivers.

By talking with the Intel fellows, we've learned where we fall off the good path in their implementation and we have a new plan on how to re-build the entire system around those issues. This re-write is large in scope and is going to take us some time to accomplish, so we are relying on our fans for patience as we attempt to resolve the issue.

In retrospect, the big lesson learned is even if you have a lot of testers, you will not get good coverage of hardware. You might get good coverage of enthusiast hardware, but not all hardware. What we initially thought of as a few odd issues on very old hardware turned out to be much more prevalent than we thought was possible. As well, just because a specification is from 2004, and understood as an industry standard, does not mean there is good driver support.

We should have tested wider on hardware earlier. The diversity of the PC platform is a key strength, as it allows many vendors to create components and it creates competition, keeping prices low. This diversity also creates nightmares for developers, as there are always surprises when certain hardware configurations do not meet your assumed performance standard. I've been doing PC development since 1994 (yes, I am really old) and on every project there has been at least one major PC hardware issue. I look forward to the next PC issues as I would look forward to finding a cobra in my bed.

5. iOS 6

Late in development we started testing on the upcoming iOS 6 beta build. We're not 100 percent sure what was changed in the core iOS graphics code, but we took a massive hit in framerate on all iPads. The iPad 1 and iPad 3 became completely unplayable, less than a month from our contractually-amended ship date.

We quickly diverted most of our key programming effort to fixing the performance concerns. We fought a war between hurting the visual upscaling quality and improving the framerate. In the end we lost a lot of key developer time to a completely unplanned task. The end result was that a great deal of our bug-fixing time was spent not fixing the bugs we wanted, and the quality of the shipping build suffered. The lesson learned here is to build in some slack in terms of performance for when you need it due to an unplanned change, and to budget some extra time for periodic performance testing on current and beta OS releases.

Conclusion

In summary, *BG: EE* was defined by the product it is built upon and the hard, respectful work of a small and caring team. We could have done faster development work and we could have had fewer bugs by making sweeping changes, but early on we decided to adopt the role of curator. We didn't want to change *Baldur's Gate* just to show off our development skills and leave our mark; we wanted to make a great game even better and in the process, bring it to new platforms. With the great commercial and critical success so far, we're very anxious to continue to improve the game even more and bring our newly leveled-up skills to bear on *Baldur's Gate 2: Enhanced Edition*.

Return to the full version of this article