GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: *Brothers in Arms: Double Time*

By Kurt Reiner,Kristin Price

[In this exclusive Wii postmortem, developer Demiurge discusses the ups and downs of bringing the Brothers In Arms WWII combat franchise -- in this case, a two-disc enhanced compilation of the first two titles in the Gearbox-created franchise -- to Nintendo's console.]

Brothers in Arms: Double Time is a first person shooter and strategy game that deviates from the standard run-and-gun, merging the action of a first-person shooter with unique tactical elements. When it comes to the Wii in 2008, *Double Time* is both the series' debut on the platform and the first Wii game for Demiurge Studios.

As with many first-time platform experiences, we had moments of genius, moments of folly, and moments of pure luck. A project that started as blue sky and nothing but possibilities quickly needed to become a compelling Wii experience that leveraged the platform without forfeiting gameplay. Even the seemingly simple ideas about "how to throw a grenade" on the Wii turned into intense design debates.

We faced technical obstacles, like keeping memory down, more than any other platform we had worked with before. For our studio, working on the Wii was like game development with a new and invigorating twist, and it made the project that much more fun and uniquely challenging to create.

When we started the project in late 2006, the Wii was still very new. Working on *Double Time* while the studio was enthralled by Wii games could not have synched up better -- we tackled the project by day and by night played every Wii game in existence.

From the onset, we sought to make *Double Time* feel natural and integrated with the platform. We wanted to take this story and reinvent it on the Wii. So we gathered a small, close-knit, versatile team and got to work. From technical snags to design challenges, the path to release was laden with successes and troubles.

What Went Right

1. Good platform.

The Wii proved itself to be an outstanding platform for a smaller studio. As the team size required to ship a title has grown, it's become more difficult for us independent developers to find a gig to call our own. With a history of successful co-development projects behind us, the Demiurge team decided it was time to stretch our legs a bit.

This project, and dozens of other Wii titles like it, proved to be a great way to stay in the traditional retail console space without having to staff up quickly. At a fraction of the cost of the 360 or PS3, the development equipment pricing also made getting started a lot easier. The innovation of the Wii interface also helped foster an innovative spirit at Demiurge -- and really in the game industry as a whole -- that we've been able to maintain moving forward on our 360, PS3, and PC projects.

2. User testing refined the control scheme.

We knew that the control scheme was a crucial deciding factor of whether *Brothers in Arms: Double Time* would succeed on this platform. The inclination is to use all the features of new technology, but we wanted to make sure the Wii controls folded well into the gameplay -- not the other way around. So the team sat down and brainstormed creative ways to leverage the abilities of the Wiimote and Nunchuk without overdoing it. After several iterations, we began running user tests to help settle on a final control scheme.

We performed two different kinds of tests. In the first, the producer administering the test did not speak, leaving the participant to rely on pure discovery to play the game. The second kind was a series of "tissue tests" in which we brought in people who had never seen the game before, from internal developers on other teams to folks outside the industry.

The tests were administered one-on-one with the producer and user tester in a private room. The tester was introduced to the game, given instruction on how to play, and asked to play a level twice, each time using different control schemes. The testers were asked to choose their favorite scheme and respond to a series of questions about their experience and preferences.

What came out of these user tests was hard data that helped us understand what players of all experiences liked and disliked about the various setups. We also discovered trends in behaviors between seasoned and new Wii players. This data eventually led to the ability to choose between two totally different control schemes in *Brothers in Arms: Double Time*. In the end, the two most popular schemes for testers made it into the game.

---

3. Created reusable Wii technology.

At the beginning of development, we identified the creation of reusable Wii technology as a long-term engineering goal of this project -- but we wanted to do so while still emphasizing the need to deliver a high quality *Brothers in Arms* experience for the player. Beyond the typical issues that arise from developing for a new platform, early in the development cycle we identified optimization as a major challenge in creating an engine with flexibility enough for a variety of game types.

Our goal was to ensure that not only would *Brothers in Arms* run well and look good, but any future projects that used our engine would not need to spend significant amounts of time on optimization.

By emphasizing the long-term goal of reusability and the short term goal of shipping a high quality product from the outset, we were able to focus our efforts in the right places. The result was an engine that delivers a visually compelling Wii title with technology flexible enough to be used by Demiurge in future Wii projects.

4. Scrum and constant planning saved time.

During development for *Brothers in Arms: Double Time*, the project management methodology of Scrum and Agile Development was beginning to take hold of the game industry. After researching it internally, we elected to give it a small trial run.

We began using Scrum toward the end of the project to systematically tackle the close-out of the project rather than solely in the development cycle. Members of the team would take on classes of bugs during a sprint. The goal was to have each system online and bug-free at the end of the sprint. This, coupled with a strong engineering team, allowed us to complete the project without crunching.

Since working with the Wii involved a lot of new technology challenges, we planned hard from the start of the project through zero-bugs, revising and planning even more aggressively as we entered the bug fixing and optimization periods of the project. At the end of the project, we posted a large bug fix chart that constantly reflected how we were closing down.

This kept everyone on the team aware of how we were tracking toward our goal and instilled some competition for how quickly we could optimize and fix issues. We also began tracking data as to how the team was progressing so that in future projects, the pre-production planning could be robust from the get-go.

We found Scrum to be very successful in allowing us to "plan" initial bug-fixing so that we had achievable goals that were within our timeline and owned by the team. It also allowed us to adequately think out all the problems before digging in and tackling them one by one. We've since used the Scrum methodology for all of our projects with increasing success.

5. Support and resources from Nintendo of America.

Double Time was the first Wii title to be developed at Demiurge Studios, so there was some ramp up time to get familiar with the platform. We were working on this title long enough and early enough in the platform lifecycle to see an evolution in the SDK, the profiling tools, and the lot check requirements.

In short, the learning never stopped. While Nintendo cannot do the work for you, we found the documentation and web resources very helpful. In particular, the newsgroups are an excellent source of knowledge. Someone somewhere has had the same problem you're having and they've asked about it on the newsgroup.

The response time to questions from Nintendo of America was usually within an hour or so. It seemed as though the family friendliness of the Wii console also existed in the developer support arena.

What Went Wrong

1. Build turnaround time too long. The number one development mistake we made was to create a build process that took longer than a few hours to complete. The build machines generated custom asset loading lists and memory maps for each level. The level was "played" on the NDEV by a bot while instrumented systems sent profile data back to the PC. The data was then analyzed and used to construct an optimal level package.

This means that customers will experience short load times, dense levels, and smooth gameplay -- but the resulting build time was nearly 16 hours for the game's 37 levels and five languages.

This was acceptable early in the project when we were delivering builds two or three times per week and wanted to focus on gameplay features. However, the turnaround time was a painful delay during the final stretch when we needed to respond quickly to minor requests from Nintendo or Gearbox.

We employed two build machines, but we would have come out ahead had we spent another week upfront to further parallelize and optimize the entire process.

2. Submitted too early.

The last few weeks of any project are difficult. The end is close and everyone wants to see the game on the shelves. We've seen a number of projects try to rush this stage and each one has been kicked to the curb by either the publisher or the console manufacturer. Being "bounced" invariably causes delays for the developers as everyone takes an interest in the issues and is distracted by the result.

It might create ill-will within the lot check trenches too, since receiving anything but someone's finished product is bound to be frustrating. But we ignored what we knew from experience when we heard that test turnaround times at Nintendo were getting longer due to the holiday rush.

So we submitted early and entered the bounce-house for a few cycles. Eventually we emerged on the other side with a stamp of approval. But it was more like the nod of consent you'd get from a border guard than the warm hug from your mom after school.

3. Shifted development team before game approved.

Having small development teams means everyone's time is in high demand. We had managed to keep the team intact up until the end of the project, when we made the mistake of ramping down too quickly. As soon as we submitted to Nintendo, we reduced the team size.

The result was that members of the team ended up bouncing between projects towards the end of *Brothers in Arms: Double Time* as the game came back from submission and required more work. We had to shift resources to make sure we had the coverage we needed, which added to the distraction of submitting and then regaining the staff to respond to results.

We took the gamble and lost here, and the lesson we learned is to give the project lifetime its proper due in all aspects, including submission and staffing.

---

4. No long range development plan at the start.

The overall development effort "went right," but from a scheduling and planning perspective, the development production staff got lucky. This project was fueled by excitement about bringing the *Brothers in Arms* property to the Wii. The pre-production technical due diligence was in the spirit of "relax, everything will be *juuuuust fiiiiiine*."

There was never an itemized list of the technical hurdles that the team would need to overcome. It's almost as if we were lulled into complacency by the cute white controller and that mustachioed muscleman in yellow at WarioWorld.com.

After a couple of months, when the engineering team was staffed up to full strength, the scope of what was required on the optimization front became clearer. At that point, rescheduling to accommodate optimization is more art than science.

A team can make huge strides in a short amount of time while that last little bit of performance to bring the game up from 27 to 30 FPS could require major changes. We managed to work through a host of platform specific and optimization complications: a byte-swapped platform, slow IO connection between the PC and NDEV, an elf that was 4x too large, excessive (and minor) memory allocations, a new and untested rendering pipeline, slow physics/ragdoll simulation, in-game IO hitches, and a seemingly endless stream of miscellaneous framerate hits.

In the end, the consensus was that we could have done more pre-production planning or risk assessment to better set expectations with our publisher.

5. Made design decisions too late in development.

The Wii controls have such flexibility that you have to be careful not to overdo it. You don't want the players to need to do acrobatics in order to succeed -- unless, of course, you're making an acrobat game.

From the first day of the project, we were brimming with ideas on how to use controls, the UI, and all the features that separate Wii from other platforms. In fact, we couldn't turn *off* the ideas. We didn't set a feature lockdown date, and as a result we were still tweaking the control scheme long after we should have. This left a lot of pieces feeling chaotic and unfinished right down to the wire.

The user tests ultimately helped us finalize; however, we started them too late in development, so all of the best data was arriving late to the party. User testing was not planned from the beginning of the project, but rather was a thankful realization that we needed clean, external input in order to settle on a well-informed decision.

The lesson there was to plan user testing from the start so that there are set periods earlier in the process where the team is prepared to show the game and react to the feedback.

Without setting a firm lockdown date, we allowed the user testing and subsequent tweaks to continue too long. The realization that a variety of play styles would be better captured for players in two separate control schemes was positive, but it came with a cost. We had to scramble at the end to get UI, balance, and all of the other items that hinge on the controls updated and finished.

Had we properly set a feature lockdown date, we would have been forced to conduct user tests earlier, secure the control scheme earlier, and then take the proper amount of time to get the UI and balancing worked out to each control scheme. Once again, we succeed because our talented team pulled through.

Conclusion

We were so excited to work on *Brothers in Arms: Double Time* that we stepped into the ring on a wing and a prayer. We managed to emerge successful because of a talented team that happened on the right decisions at the right times.

The project brought great things to our studio, like the Scrum development process and tech we can use for all of our future Wii titles. Beyond the studio growth that came with *Brothers in Arms: Double Time*, it was exciting for Demiurge to land on a new platform when the platform itself was still reasonably new.

Game Data

Developer: Gearbox / Demiurge

Publisher: Ubisoft

Platform: Wii

Release Date: September 2008

Development Time: 13 months

Peak Development Team Size: 15

Return to the full version of this article