Gamasutra - Postmortem: DoubleBear's Dead State

Dead State (and DoubleBear Productions) started as a part-time project for two members of DoubleBear and two members of Iron Tower Studios, way back in late 2009. We announced the title officially in mid-2010, and picked up a few other contributors to help out with the art. Over time, it became pretty clear that working on the game part-time was a less than efficient way of getting an RPG finished, and we weighed the decision to either cancel the game or seek enough funding to start full-time production.

Ultimately, we chose to continue the game's production, and mounted a Kickstarter campaign in June 2012; it was successful, earning us about $300,000 in funding to hire new contributors, bump part-time staff up to full-time, and finish the game. By the end of the project, we had about ten team members and a few regular contributors for sound, music composition, and testing/forum moderation. The majority of the team worked out of their homes, with staff members spread out over multiple countries worldwide. For many on the team, *Dead State* was their first released game.

The goal of *Dead State* was to create a zombie apocalypse story that focused on the human element and survival gameplay, inspired by movies like the original Dawn of the Dead and the real-world aftermath of natural disasters such as hurricanes Andrew or Katrina.

We modeled gameplay elements of *Dead State* after the original *Fallout* and *X-Com*, as well as *Jagged Alliance* and *Suikoden*. In addition to these inspirations, we utilized the genre-savviness of the team, and the RPG design experience the founders of the company had learned during our time at companies such as Black Isle, Troika, and Obsidian.

In the time between its initial concept and completion, zombie and survival (and zombie-survival) games became commonplace in the market. However, we continued to emphasize the elements that we thought made *Dead State* stand out -- namely the exploration and Shelter mechanics, reactive dialogue and characters, and real-world RPG setting.

Despite picking up additional team members to finish the game, *Dead State* slipped a year from our original December 2013 release date, appearing in Steam Early Access in February of 2014, and releasing in full on December 4th, 2014. *Dead State* was released to mostly fair-to-positive reviews praising the story elements and Shelter management, but was dinged for bugs, lack of feature polish, and reduced combat complexity. We have continued to support the game with fixes, balance, and content patches since release.

What Went Right

1. Kickstarter and Fan Support

If there's one thing *Dead State* had that many games don't ever get, it's a passionate fan base. Our early announcement of the game allowed for a community to form around it, and we did our best to interact with the fans, share ideas about the design, and fill them in on our progress. In a lot of ways, when things weren't looking great for the production, it was the fan support that kept the development team going.

Without the fans making people aware of the project -- in communities, forums, and press -- our Kickstarter campaign likely would not have succeeded. While Kickstarter does add more responsibility to the dev team to communicate progress and show work frequently, ultimately we would never have been able to finish the game without Kickstarter backing and with the continued support and involvement of our community. When you're in the most painful and stressful months of development, it is immensely helpful to know that there are people cheering for you.

2. Project Management Tools (Documentation, Assembla)

Early on in the process, we started documenting everything about *Dead State*'s design in an internal wiki, and kept it up-to-date throughout development. If you're a company of any size, having a dedicated resource for design and scripting can't be recommended enough. It made it much easier for remote team members to look up info and reference images without having to wait for other team members to get back to them via email or messenger. Seriously, if you don't have a wiki for your project, spend one day setting it up and teaching people how to use it; of all the ways I've seen project info tracked, an up-to-date wiki is the absolute best and cheapest way for team members to access project information.

Assembla was the other tool that made development easier, once we started making more frequent builds. It not only allowed us to quickly upload and change assets, but allowed us to back up the build, track bugs, and assign tasks to individual team members. Once we had the team integrated into Assembla, it sped up project development greatly.

With these tools and messaging programs, it was much easier to keep track of progress and create a "virtual office" for remote team members.

3. True to Scope

Many games -- especially RPGs -- end up cutting multiple levels and features before they get released. *Dead State*'s design had three (very large) key components: The Shelter (for dialogue/story and ally management), reactive story and dialogue elements (15,000+ lines for dozens of characters), and turn-based combat within non-linear world exploration.

We also hit several stretch goals for our Kickstarter, including adding dog allies and other animals, additional allies and special areas, and a larger multi-level map set in a city -- a significantly large area that requires special assets and planning.

While the game ended up taking longer than promised to complete, we didn't cut any major features from *Dead State* -- in fact, we ended up putting in more levels and features than originally planned. We knew that we sold *Dead State* on being a very specific kind of game, and we wanted to make sure that we lived up to the promises we made during the Kickstarter, even if it did involve quite a lot of effort and more crunch than we would have liked.

4. Core Design Did Not Need Major Revisions

From the beginning of the game, our design revolved around a few key "identified risk" design concepts: Our "Noise" system, which attracted zombies or enemies to your party's position; the Shelter, which involved the management and job system; and the Crisis Event system, which dealt with systemic Morale and character events.

If the Noise system hadn't worked, our combat would have been a lot less exciting, and the zombie apocalypse feel of the game would have been lost. If Shelter management had been less appealing or if the GUI was too complex, it would have made a large portion of the game unappealing. If the Crisis Event system hadn't worked, it would have taken away one of our more interesting story systems and given the sub-leader charaters less presence in the Shelter.

Fortunately, aside from a few tweaks, we did not have to scrap or completely redesign any major systems in the game, and Early Access feedback on the core gameplay was about on par with what we were hoping. If the feedback had been mostly negative for these systems, we would have had to have done major course

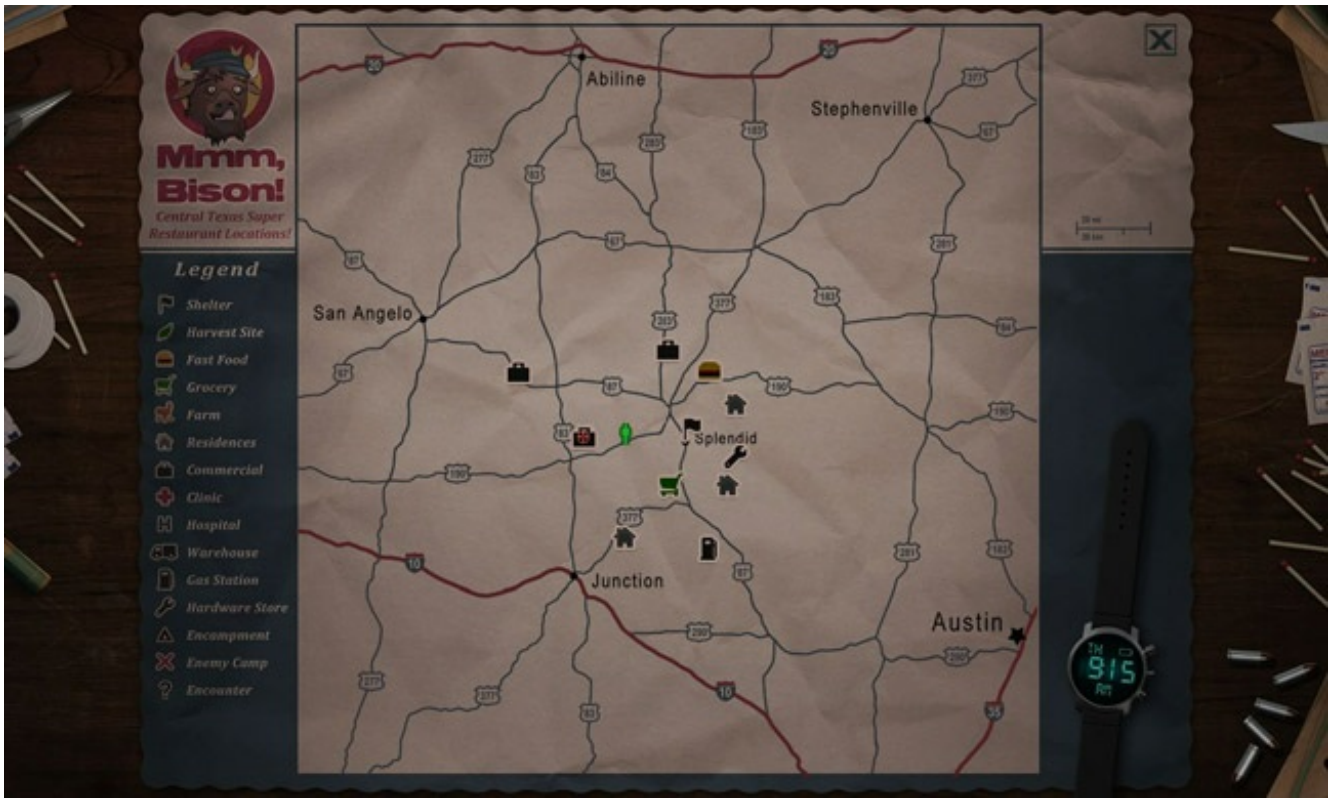
correction. Not having to gut and rework major systems in the game saved us a lot of time and effort. Being careful about iteration and implementation of risky design elements up-front was a good idea.

5. Level Creation Pipeline

Dead State ended up with over 75 levels of various sizes, and the timetable to get them done was incredibly short by industry standards. Our prototype level (Llano Commercial) had taken over a month and three leads to get it up and running, and since a lot of the game hinges on exploring and scavenging, we knew that we needed to refine our level development process.

We solved this by building level and building templates, creating a level tool that allowed us to easily generate or clone all the necessary level files instantly, and by establishing a system for handing off the level files to different specialists to get them to final implementation much quicker. Designers would do the basic layout, scripting, and combat with our editor, artists would polish the look of the map, a loot designer would do a loot pass, and then the project lead would hook them up into our world map for testing.

By the end of development, we got so good at this process that a random encounter could be put together in about an hour, and a larger map could be done in about a day. If we hadn't improved our level pipeline, the game would have had much less content, and the overall design would have suffered.



What Went Wrong

1. Lack of Mentoring

As mentioned previously, *Dead State* was the first released game for most of the DoubleBear development team. Usually, in a larger industry environment, juniors will be paired with seniors to mentor them and help them break bad habits that might be causing development issues. However, due to the budget, we did not have a lot of room to bring on seniors, and as a result, we ended up throwing a lot of our juniors into the deep end of development.

Fortunately, many of them rose to the occasion and were able to do a much more capable job than would be expected of many inexperienced developers. While I'm glad this mostly worked out for the project, I would have liked to have more training for the team and more experienced developer oversight for several systems in the

game. Not only would this help key systems get a more practiced eye and stop development issues before they arose, but it would improve the quality of life for the juniors involved by lowering their stress about production and raising their confidence by succeeding in less weighty tasks.

2. Budget Issues

Almost everything in game development comes down to money. Money buys additional team members or experienced team members, which in turn shaves time off the production and potentially helps create a better game. While we did well on Kickstarter and budgeted accordingly, it was only enough money to keep the team on the small side. Most RPGs I have worked on have had a minimum of about 25-30 people working on the game for most of production, while *Dead State* only had about 8-10 at any time.

Without the ability to bring on additional team members, it meant a lot of us were working extremely long hours and were wearing multiple hats throughout development. (For example, for most of the project I was the only full-time designer, the project lead, the business guy, the level lead, the lead writer, the systems and combat lead, and an additional scripter.)The lack of additional personnel slowed the development of key systems and created bottlenecks for both the creation/refinement of those features, and for getting feedback from key team members.

We planned the game around the Kickstarter budget, without making the mistake of counting on Early Access to prop us up if we got into trouble. If we had a budget closer to some of the other bigger RPGs released in 2014, it would have allowed us to hire on a few more seniors earlier in the process, and allowed for B-teams to help build out, test, and polish the game.

While we were able to achieve an impressive scope for the project, we definitely had to work with the base of what we could afford. Most reviewers called us out on the lack of activity in the Shelter or the turn-based combat not being interesting enough, and while we would have liked to have done more with them, we simply did not have the budget to make those features happen.

3. Tech and Tools Issues

Dead State was made using Torque 3D both because of the lead programmer's familiarity and because it was cheap/free – and unfortunately, like most engines, it does not natively support the tools and features needed to create RPGs.

While we originally planned to use Iron Tower's tools to speed up development, we ended up tweaking or creating new ones specifically for *Dead State*. While it was nice to have separate tools to do what we wanted, we had to take time not only to concept each one design-side, but build it programming-side, troubleshoot it QA-side, and start it all over again if we ran into any issues or new needs. Major features, such as character Perks and Traits, were often significantly delayed because the toolset needed to create them had run into a snag.

Torque is decent for making PC games, but is an older engine, and not exactly state of the art, or a great fit for Mac/Linux/console ports. Sticking with Torque 3D instead of switching to Unity early on was a big mistake, and the lack of ability to easily port the game impacts our ability to profit off of potential future ports.

When choosing your tech, make sure it's not only right for your game but for potential expansions or different systems down the road. In the future, we'll try to rely on out-of-box systems more, or make sure that we have a stronger custom infrastructure before attempting to create a significant amount of content with it.

4. Remote Team Communication Issues

Communication breakdown can happen even when everyone is under the same roof, but when you've got a team spread out over multiple time zones, the odds are pretty good that communication will break down.

And it did.

There was an inherent difficulty in tracking progress: Sometimes emails wouldn't be answered for 24 hours, builds would hit a major snag and need to be reuploaded, or critical bugs wouldn't be fixed for a day. Without the team in the same room, meetings went overlong, issues took more time to explain, and momentum/excitement about progress was less tangible. New builds would not always go out on a timetable that was helpful for the Seattle team, where the majority of DoubleBear team members were located.

A large amount of gatekeeping of certain systems and code exacerbated delays greatly, and frequently, team members would have to log on at early/late hours to discuss work or fix a bug. We improved communication over time, but many of our delays can be attributed to communication issues and not having a centralized team. If you're going to use remote team members, make communication a top priority and prepare for the issues that it can cause.

5. Too Many Characters

When we first developed *Dead State*, we wanted to make sure we had a large enough cast to make anyone expendable in the field. We shot high to make sure that players would have a large number of allies to present overlap in skillsets and to provide a lot of the dramatic conflicts that take place in the Shelter.

This meant that each ally not only had to have their normal Shelter dialogue, but their intro dialogue, dialogue based on their mood, conflict with other allies, "quest" dialogue, reactive event dialogue, random event dialogue, and so on. And even with all of this dialogue, the player may only see a fraction of it, depending on what they did with each ally.

There are over 45 allies in the game, with 6 to 30 branching dialogues for each ally. The amount of dialogue added up quickly and took a lot of time to write, script, and revise, especially when it was done by two designers that also had to oversee other systems.

The biggest problem stemmed from our allies being around the shelter for a period of up to 85+ days of game time. There was definitely a player (and reviewer) expectation that the allies should have had dialogue refreshed nearly every day, or that they should have as much to say as companions in other RPGs. However, most RPGs spread their companion dialogue and reactivity over 8-10 companions or non-playable characters, and update it in stages according to a set of linear plot points (another thing that *Dead State*, by and large, happened to avoid).

By the end, *Dead State* was already pushing the 15,000 line count with our massive cast, and that's not even counting combat/reactive barks. While I think we did a great job making the allies diverse and as interesting as we could for their screen time, I can honestly say I would never recommend anyone attempt to have 45+ companions in your game unless it's going to be a very short game.

Lessons We Learned

As we wrap up *Dead State* support, the DoubleBear team has already taken many of the lessons from the development and applied them to the development structure of our next game. These include:

1. Prototype all systems and scale scope quickly

Get the game design up and running as soon as possible. Stub in temp art menus. Use temp art to test concepts. If it's not fun with temp art, revamp it -- art is great, but it's not going to magically make design fun. If your systems/GUI demands are getting too complex, cut some. Figure out your time remaining and team size, then scale the rest of the game's scope to the production time you have remaining. These may sound like common sense ideas, but we definitely fell into the trap of trying to get the project looking like a full game before it was ready for it.

2. Get to Beta level on all systems much earlier

Dead State was still implementing some of the game's features very late in development -- although to be fair, many RPGs struggle with this because of their complexity. For future titles, we're developing tools to iterate quickly to get to Alpha earlier in the process, then using the back end of development to polish or enhance the existing content.

3. Structure game design and scope around budget and team size to avoid crunch

Dead State could have used twice the number of team members than we had for its development, which meant lots of time crunching to meet deadlines. Our future titles will be built on the team size we have the budget to support. For now on, we will cut or revise anything that needs major crunch to get implemented.

4. Better and more frequent communication

Dead State could have benefitted from more team interaction and lead meetings. For our next project, we have already adopted the process of meeting as a group and going over the design and builds together, with lots of opportunity for team feedback to be considered during the early stages. Even if you work remotely, as DoubleBear does, it is intensely helpful -- even vital -- to try and meet on a regular basis, even if you must use stand-ins like Skype if you cannot meet in person.

5. Announce close to release, save energy for release PR

Don't announce until you're much closer to release. You risk losing momentum for sales and patience from fans. And even though the final push of development can take a lot out of a small team, leave some dev time to promote the game. Some games are only going to get a press bump when they announce, so you need to capitalize on that hype closer to release or you may be forgotten or written off as vaporware if you announce too early.

Conclusion

For a new company, *Dead State* was quite a challenge and has definitely shaped the way we structure, advertise, and consider future projects. The budget was the bare minimum needed and the game has thus far sold on the lower end of our projections; it's difficult to justify making a sequel or another RPG in the near future considering the amount of time and team size needed to produce one that satisfies the expectations of hardcore fans and press.

Communication and tech issues were definitely a hindrance to getting the game finished in a more timely fashion. Press, for the most part, seemed to want to compare the features of the game to much higher-budget RPGs and strategy games, and dinged us for not having stylized/better graphics, not adding a cover mechanic

or interrupts (basically not recreating *XCOM*'s most recent iteration from 2012) to combat, and not having even more frequent dialogue interaction and animation in the Shelter portion of the game.

The general “zombie fatigue” of the press definitely did not help get us much in the way of release coverage, and we found it fairly difficult to get attention by traditional press outlets, despite the focus of the game being anything but another zombie-killing game.

Despite the limitation of our budget and staff size/experience, our team managed to put out a game that has garnered a strong fan base and much in the way of consistent word-of-mouth sales. It’s done well enough to allow us to make future titles, which is a pretty big win in the indie game business side of making games. We’ve already got some pretty exciting (and very fun) projects in the works, and at the very least, *Dead State* has given us a solid foundation to launch into the next phase of DoubleBear’s development as an indie company.