Death Orb Wars

Postmortem

Eddie
PSYODRONIX

What Went Right:

The Game Was Finished

I have to start off with the biggest positive. While the game was an unabated failure by every metric, it still has the success of being the first game my team had completed start to finish. It didn't live up to our hopes and we hit many problems due to inexperience along the way, but it was a very unique experience. Even though the feeling of dread and failure were palpable in the closing moments of the game's development, it would have been far easier (and far less internally rewarding) to scrap the game and make something else.

All in all, the game sold around 30 copies, and was unusually popular in Japan (relative to other regions).

What Went Wrong:

Hardware Failures and Time

I originally purchased developer access to the Xbox Indie Game Marketplace around September 2011. I was working on a few small games that I found mildly amusing, but nothing with a team. I came up with an idea for a small game that could be "quickly" done with a team. This game was pretty simple in concept, but it was also going to be the first game I truly worked on.

The game's original version began development in December 2011. It was originally intended to be a multiplayer 2D bullet-hell survival game. The original plan was for 4 players to be able to play locally or over the internet. Unfortunately, during a storm, my power was knocked out, and my hard drive was bricked. Prior to that, I had never had a hard drive fail suddenly like that, and I hadn't created a backup.

Game scrapped. Time to restart.

Once all of the necessary software was installed on the new hard drive, development of the game began anew, but so did the problems. On the plus side, I was able to rectify some of the mistakes made in the previous version. On the down side, I was effectively restarting a project, and classes were back in full swing. By the time I got back to where I was originally, it was around April 2012. My poor backup skills had set us back nearly 4 months at this point.

Before the game could be completed, a new problem presented itself. My credit card billing statement had a $100 charge on it from Xbox Live. I had not purchased anything and I promptly contacted Microsoft. After spending multiple hours on the phone with them and without the ability to figure out which account (whether my Xbox Live account or the business account) was the one with the security breach, both accounts were locked. With that lock, I was unable to test the game on my Xbox until later notice.

Skip forward…

Both of the Xbox accounts were eventually unlocked. It was now July 2012. After an issue with the multiplayer in the game, multiplayer ended up being wholly scrapped in August 2012. We attempted to push the game live in August 2012 with a plan to update the game later in the month. First push? Game denied. Someone found a hard crash.

We were now approaching a 2-week window for our game to launch before the one-year subscription ran out. I made the call to try to push the game live again. We knew of one major bug, but once we had thrown the executable up on the Xbox Live Marketplace, we began working on improvements to the game. We had started working on our first patch within minutes of uploading the game.

A few days later, the game passes and it goes live. The major bug had been fixed at this stage, and I realized I had made a grave error. The update process had a 1-week lockout even after successful submissions. This meant that we likely only had one shot to get the update live before the monthly subscription ran out.

Fast forward to the update push

The main menu had been upgraded. The controls were spruced up. There was a menu that explained to everyone what every enemy did. There was a tutorial level where they could see everything in action.

Unfortunately, the update did not drum up enough attention on the Xbox Indie Developers forums to get enough people to review the game. Time came and went and the update didn't have enough votes. It failed. The one-year Xbox Indie Game Marketplace subscription ran out before the update could go live.

As a result, the game, to this day, is plagued with a horrendous FPS loss that is fixable by pausing the game while playing. The game did not run garbage cleaning on the Xbox 360 as regularly as it did on the PC. In fact, unless the game was paused, the Xbox 360 version never ran the garbage cleaner.

Poor Multiplayer Planning

The game was playable on the PC for the most part, but multiplayer ran into a new issue. In the game's core while loop, the game would iterate over all active components on the screen to instruct them to update. During the original planning for the game, I did not see a reason to need to directly reference anything besides the current actor and the player(s). The AI acted deterministically, so they would not need to be directly referenced at any point.

Then came the Dance Orb, an enemy type that was conceptualized later in development. The Dance Orb was an orb that, upon spawning, instantly changed trajectory in a random direction and a random speed. After 0.2 seconds, it did it again. Then it did it after 0.4 seconds, 0.8, 1.6, 3.2, 6.4, 12.8, and it would eventually fly off the screen into the kill zone.

The Dance Orb ended up requiring a different update system. The server ended up sending a packet that told the client, "At gametime W, Dance Orb X was at position Y, and it changed velocity to Z". The goal was to update the Dance Orb X's position to be where it should be given the time passed between velocity change and now. As a result, the Dance Orb was reasonably accurate across the network, although at times it would appear very janky.

That turned out to not be true. As the game moved along, enemies would spawn more regularly. Every time an enemy was spawned, the server would inform the client that “unit X spawned at position Y moving at velocity Z”. Unfortunately, this wasn’t guaranteed to create orbs in the exact same order as the server.

This ended up creating complete desync between client and server. Once a dance orb entered the screen, it was anybody’s guess as to how long until the game ended because one of the players died on one of the player’s screens. The server may determine that the dance orb [55] has changed its velocity to <5.2, 1.8> and it would inform the client(s) of such. As a result, the client(s) would update orb 55 to a velocity of <5.2, 1.8>. It wasn’t uncommon for orb 55 in this situation to not be a dance orb, though. So now came figuring out which orb 55 actually was.

As mentioned in the previous section, this is taking place in August 2012. We are facing the deadline. The only solution for this current problem is to redo the actor storage system from a simple list to a hash table so that an accurate reference could be passed regardless of lag. As could be guessed, I was worried about not making the September 2012 deadline, and multiplayer was put on the list of things that would be added if we decided to renew another one-year developer license.