GAMASUTRA
The Art & Business of Making Games

PRINT

Indie Postmortem: *Gibbage*

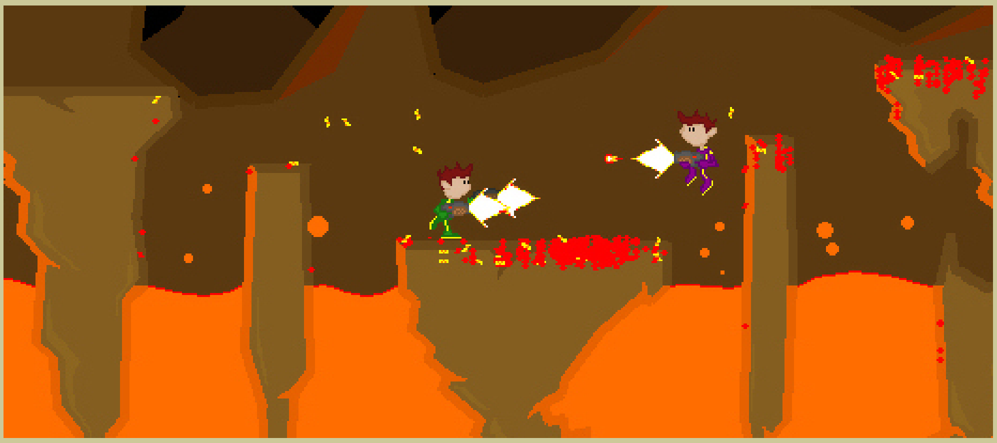
By Dan Marshall

The Origins of *Gibbage*

The tale of ultra-indie PC title Gibbage, which I developed myself over the period of around 2 years, is a fairly unique one. I was fed up with endlessly shooting men in the head in a narrow corridor. Big-budget blockbuster titles had begun to bore me; I was no longer challenged, interested or entertained by tapping the “quick save” key every eight seconds and picking up *another* machine gun.

Instead, I wanted a game that delivered short, sharp jabs of sublime gaming fun. Like in the olden days when I was a kid, and friends would come over to my house to play the likes of *Worms* or *Bomberman*. We'd laugh so hard at each other's ineptitude that it would*hurt*. Simple ‘fun' seemed to be an element sorely missing from the games I was playing - instead frustration, boredom and downright apathy had taken over.

When I took to writing *Gibbage*, I knew nothing about coding. I was reading*C++ for Idiots* on the train to work every morning, and studying online tutorials whenever my boss wasn't looking in the office. *Gibbage*'s first line of code went in shortly after, and I've been building on it ever since. After a short time, it became an obsession: every spare second I *wasn't* working on my game felt like a wasted second. Putting a new feature in that worked well was the most incredible high I'd ever experienced, and the sense of satisfaction I got from completing work on *my own game* far surpassed that of watching the end credits roll on someone else's.


Gibbage

What the Hell is it, though?

Gibbage is a cartoon Deathmatch game between two opposing teams. Each team spawns in a Power Booth with a Power count that is perpetually ticking down. Your goal is to run around collecting randomly-spawned Power Cubes and returning them to your base. This increases your Power count and reduces your opponent's. Once the other player's Power reaches zero, it's 'game over'.

Given that these Power Cubes are relatively rare, they're hotly contested which brings players together in a shower of bullets and physics-laden explosive fun. Players also have access to a wealth of wacky gun upgrades, power ups and special moves that add to the ‘unscripted mayhem' hilarity potential of the game.

What went Right

1. Gameplay. I love *Gibbage*, I think it's brilliant. Even though I've played countless games, I'm always eager to fire it up and have a quick blast through. I think it's testament to the game's core mechanic that, even after playing it so thoroughly, I'm still eager to come back for more. No two games of *Gibbage* are ever the same, and I couldn't be more chuffed at how the game actually*plays.*

From the amount of time it takes a player to respawn to the random appearance of the Power Cubes, everything works in synchronicity to create a unique mechanic that flows perfectly.

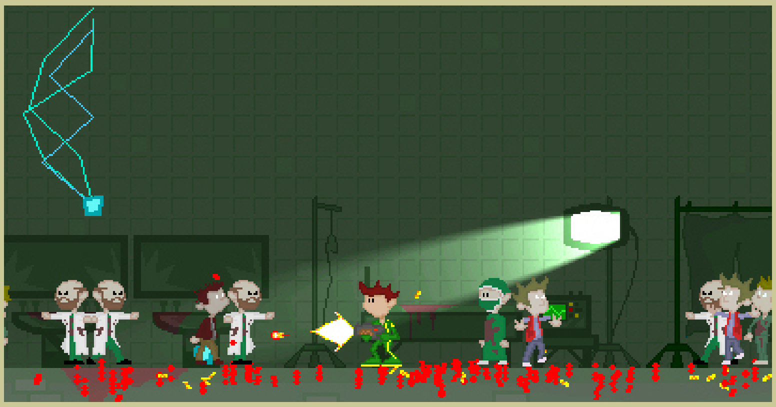
This didn't happen by chance: many weeks were spent with a ‘bare bones' engine tinkering with these numbers at the very start of the project, and playtesting extensively to ensure the balance was exactly right. It's important to remember that games and ideas *evolve* with time, and that you need to put the time and effort in to guide that evolution in the right direction from Day One.

2. Levels. From the start I wanted to make sure the majority of the maps in*Gibbage* had a 'hook'. There are a few 'vanilla' maps in which there are no unique elements, and the level's location is little more than fancy wallpaper. However, where *Gibbage* shines is in the variety of the levels. While struggling to win, you may have to deal with zombies, aliens, acts of God or a third player being thrown into the mix; or indeed fight in low gravity or underwater. It's this variety that keeps the gameplay fresh throughout each of its levels. I made a decision early on that I'd prefer to have 27 unique levels than hundreds that are essentially the same but with a slightly different layout, and I think this decision resulted in one of *Gibbage*'s best selling points.

3. Language. *Gibbage* was coded largely in C, with splashes of C++ here and there. However, I wouldn't have gotten nearly as far as I did without stumbling across the Allegro game-programming library. I discovered it very early on while pulling my hair out trying to set up a window in Windows, and have since come to the conclusion that libraries such as this (and SDL) are the saviors of indie developers everywhere.

4. Humor. There's something hilarious going on in*Gibbage*. It isn't funny in the same way that*Insult Swordfighting* is funny, but there's a sense of unscripted mayhem going on that can't fail but to raise a smile. It's moments like when you time a bomb to fling your opponent into a proximity-sensitive mine that blows up and causes their head to bounce around the screen that you realize there's something quite special going on at *Gibbage*'s heart.

5. Ethos. People have asked me to make*Gibbage* playable over a LAN or even online. However, that's going against what it's all about: it's *social interaction,* something so sorely missing in games these days *Gibbage* brings people together around the warm, inviting glow of the monitor. It's a socially acceptable waste of time in which tickling and nudging the keyboard away from your opponent are valid 'tactics'. If there's nothing on the TV, the first thing my girlfriend tends to suggest is that we head off into the bedroom and play a few bouts of *Gibbage*, which leads me to think that I've succeeded in capturing the spirit of games of yore.


Gibbage

What went Wrong

1. Inexperience. I hadn't a clue what I was doing when I wrote the first line of code. Beyond "Hello World",*Gibbage* is *pretty much the second computer program I've ever written*. If I was sensible, I would have written some smaller, less ambitious games first and started writing *Gibbage* when I was a relatively competent coder with a few notches on my cyber-bedpost.

Sadly, I'm not that intelligent, and as a result there are lines of code lurking at the back of *Gibbage* that are so amateurish and ghastly that I've got half a mind to delete all my source code so no one ever finds out. It's embarrassing, really, and it's a major faux-pas on my part.

2. Environments. The environments fell somewhat short of my original vision. I was always really keen on the idea of trashing the arenas- I love the way they're bloody, littered with corpses and *ruined* after a few minutes' play. Originally, I wanted to fill them with destructible objects; floors that gave way, bricks flying about under the force of explosions, cave-ins, that sort of thing. As far as I'm concerned, the more random elements you chuck into a game like *Gibbage*, the greater the potential for hilarious unscripted moments of fun. Unfortunately, when I started *Gibbage*, I knew that sort of coding was*way* beyond me, and I had to limit myself.

As it is, the gameworld itself is in fact entirely constructed using a rigid Tilemap. Freebie map-making utility "Mappy" was used to construct the levels and then pasted over with a single graphic wallpaper to try and reduce the repetitive patterns and ugly tiling traditionally inherent in tile-based environments. *Gibbage*'s major strength, replayability, suffers only in the inflexible nature of its gameworld.

3. AI. Originally, *Gibbage* was a two-player only game. This is because, like many similar titles (again like*Worms* or *Bomberman*), it's infinitely better when played against a real human being. It wasn't until some helpful soul on a game-dev forum made a comment about how *Gibbage* would 'flop' unless there was a single player option that I set about learning to code an AI opponent.

It turns out that A* pathfinding is quite complicated stuff, isn't it? So I gave up and set about defining my own set of fuzzy-logic rules for the bot. To be fair, he plays the game really *really* well, but in all honesty, his intelligence is pretty much smoke and mirrors. It would have been nice to have a genuinely smart AI opponent, but to do this I should have factored elements like this in from the beginning, instead of a third of the way through development. I've now learned the hard way how important it is to create a full design document from the beginning.

4. Graphics. Don't get me wrong: I love how *Gibbage* looks - honestly I do. But if I were to go back and start again, I'd most certainly make the game look more stylized and simplified. There was an over-reliance on using bitmap images where I feel working with straightforward graphics primitives instead would have enhanced the visual styling of the game.

It's difficult to stand out as an indie game, and to do so I think bold, stark colors with simple, cuter characters would have worked a lot better.

5. Life. I'm a hobbyist: during *Gibbage*'s crunch time, I was getting up at 6am to get a few hours coding in before heading off to do my day job, and working till midnight when I got home in the evening. I was physically and emotionally worn out for a large portion of *Gibbage*'s development. This state of heath doesn't lend itself well to the innovative, fresh-faced thinking typically associated with indie game development. As such, prolonged periods of mental blockage combined with a refusal on my part to switch off and stop thinking about things only served to slow the whole process down considerably. It's clear to me now just how detrimental to a project burnout can be, and that it's important to walk away from time to time if only to clear one's mind.

Conclusion

Gibbage is by no means a perfect game. I'm more than happy to admit that it has some rough edges here and there, as you'd probably expect from a game that was written in tandem with learning how to code.

However, the feedback I get from people who have played it is 95% extremely positive. *Gibbage* is blessed with being an extraordinarily simple concept that's easy to grasp hold of, but difficult to master in its entirety. That combination came about using the following magic formula: (hard work x luck).

If you can see past one or two crude faults that are largely the result of my own early inexperience, there's a brilliantly simple, charming, addictive, laugh-out-loud funny game shimmering underneath. It's my baby, and I'm exceedingly proud of it.

Game Data

Publisher: Dan Marshall
Developer: Dan Marshall
Number of full time developers: 1
Length of Development: Two years (including learning how to code)
Release Date: April 9th 2006