

Video: How Command & Conquer: Tiberian Sun punished the computers of the day

reader comments

89 with 56 posters participating, including story author

Share this story

[embedded content]

Video directed by Sean Dacanay, edited by Paul Isackson. [Click here for transcript.](#)

When I was working at Babbage's in the mid-'90s, I recall there being three specific PC games that sat in the "coming soon" column seemingly forever—like, for years—and generated ridiculous amounts of pre-orders and buzz: *Mechwarrior 2*, the original *Diablo*, and the original *Command & Conquer*. As fate would have it, I worked the launches of all three of those games, and although they all were special, *Command & Conquer* was to me the most surprising to play.

I wasn't really big into the nascent real-time strategy genre at the time—perhaps unsurprising, since the "genre" prior to C&C's release consisted basically of *Dune 2* and *Warcraft*, but C&C blew me away. I was never any *good* at it, but I was fascinated by it—the strategy game genre was undergoing somewhat of a renaissance in the early-to-mid-'90s, and adding real-time decision-making into the mix was a wild twist on what had become an established formula.

The original C&C was successful, but the sequels established a bona fide gaming dynasty. For this episode of War Stories,

we've arranged a nicely technical chat with Westwood co-founder Louis Castle (who also worked on the studio's noir *Blade Runner* adventure) to dish on the challenges and issues the studio faced with developing *Command & Conquer: Tiberian Sun*, the direct sequel to *C&C* and one of the most well-regarded games in the entire series.

A path! A path!

And challenges there were. One of the goals with "*TibSun*" (as one calls the game when one is cool and *in-the-know*) was to have no limit or cap on the number of units players were allowed to build and control, and that goal had a huge string attached to it. More units on the screen meant more units for the computer to keep track of and control, and that meant lots and lots of *pathfinding*—that is, figuring out how to get those units from point A to point B. In spite of how simple it might appear to the human eye ("Just go from here to there!") pathfinding is actually a murderously complex programming challenge—one that, depending on the path, might be NP-complete.

The "eureka!" moment came when Castle and team realized they didn't have to build perfect—or even really great—pathfinding. Rather, they had to build pathfinding that *didn't look stupid* to the player. This meant that instead of having to solve an effectively unsolvable NP-complete problem, they could instead apply several layers of edge-case mitigation and modifications over their basic algorithm. Eventually, after much iteration, they reached a point where units acted (mostly) sane and did (mostly) what the player expected (most of the time).

The most interesting thing, at least to me, is that pathfinding is *still* a difficult problem even for modern games, and simply throwing more CPU power at the problem doesn't really help that much—such is the curse of NP-complete problems. The fixes developed by the Westwood devs are still

applicable today—and will continue to be applicable, right up until the point when someone proves $P=NP$.

And if *that* ever happens... well, pathfinding algorithms will be the least of our worries.

<https://arstechnica.com/?p=1464473>