7 uses of procedural generation that all developers should study

Richard Moss

In the right game — *Diablo*, *Rogue*, *Spelunky*, *Daggerfall*, *Elite*, *Spore*, even the likes of *Football Manager* — procedurally-generated content is magical. It elevates the design and highlights the elegance of the core system loops. It can save you time and earn you money. It's a key part of why games like *Skyrim* and *Minecraft* are able to attract and retain huge player bases that keep playing years after release. And it's the central pillar upon which Hello Games' much-anticipated *No Man's Sky* rests.

But procgen can be easily misunderstood or misused, and the idea of designing algorithms to create chunks of your game on the fly may not come naturally. So we've consulted with some smart developers and assembled this list of games that use the technique in interesting or distinct ways.

Each of the games below uses procgen to great effect, and studying them further might help you come up with ways to develop better games through generative algorithms.

Procedurally-generated content can add detail and depth to characters and systems just as successfully as it can to environment and level designs — where it's more often utilized. It's the very tool that lends Middle Ages strategy game *Crusader Kings II*'s computer-controlled characters personality. That personality manifests as various

traits, which under the surface are just some simple numbers (things like +1 intrigue and -10 greed) that alter the balance in a character's decision making. But because the systems revolve around relationships and behavior, these traits combine in fascinating, dramatic ways with the familial connections at the center of medieval power dynamics.

Kitfox Games creative director Tanya Short notes over email how the enduring popularity of the game and the deep forum discussions that grow around it show that players are completely willing to commit emotionally to the numbers and resultant behaviors (which are usually simple but when strung together can be read as complex). "For being a nigh-inaccessible hardcore game, it has that in common with *The Sims*," she says. "Players love coming up with narratives to explain the internal coherency of an individual, their politics, etc, especially when given an intrinsically interesting context like medieval Europe. Hence the evergreen popularity of 'out of context' Crusader Kings II forum threads." She cites this hilariously telling example.

TAKEAWAY: *Players will emotionally invest in procedurally-generated AI characters and relationships if they appear to behave in a manner that is consistent with their known traits — perhaps even more so if the underlying numbers are transparent.*

Tolkien-esque action RPG *Shadow of Mordor* also deviates from the procedurally-generated content norm of levels and maps. It leverages the concept to let players build a relationship with orcs. In short, every orc is unique — they're all algorithmically created by the game's Nemesis System, from name and appearance all the way down to the way they speak and their relationships with other orcs. The game endlessly combines and recombines character elements as if they were Lego blocks.

There's more to it. An orc gets promoted to a higher rank in Sauron's army if he kills the player's character. Then he remembers them next time they meet. And if he's killed or injured, he also returns — now sporting new talents, a big grudge to settle, and a procedurally-generated scar (such as a plate over his eye if he was stabbed there, or a bag over his head if he got decapitated).

This results in rivalries and feuds and leaves the player myriad opportunities to exploit orc social connections for their personal gain (by turning someone against their boss, for instance); it pushes you into emotional relationships with the bad guys, which is exceedingly rare despite bad-guy killing being a primary mode of interaction with most video games. *BioShock* director Ken Levine calls *Shadow of Mordor* "the first 'open narrative' game." It's an important step forward in creating traditional game narratives that genuinely adapt to a player's choices rather than merely branching off like a choose your own adventure book.

TAKEAWAY: *Procedurally generating enemy characteristics and post-fight scars can make players feel like they're battling specific and unique archrivals, not the same scripted sub-bosses that every other player will also encounter.*

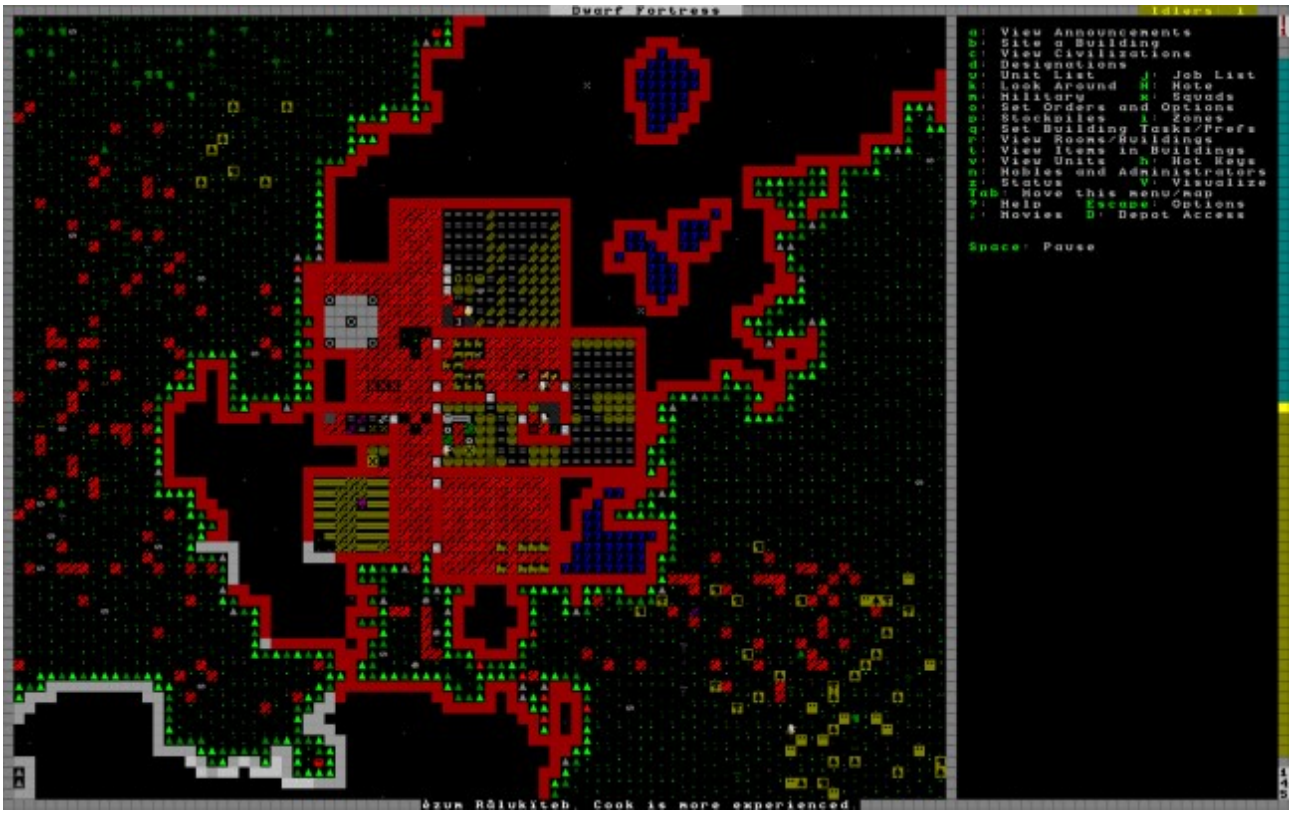
Perhaps only *Rogue* and *Diablo* are more synonymous with procedural generation than Derek Yu's *Spelunky*. Its brilliance as a platformer lies in the way it stresses mastery of mechanics and underlying systems rather than memorizing levels (which you can't do, as every level is generated just moments before you play it). New York University associate professor Julian Togelius (a researcher in the field) believes this fundamental shift in emphasis — from mastering a level to mastering a set of rules — makes it the best example of procgen used in a meaningful way. "It's absolutely crucial that you cannot predict which challenge is going to come next, and that's why the level generation (which is relatively simple but highly effective) is so crucial," he says.

Indeed, your typical *Spelunky* player has attempted thousands or even tens of thousands of levels. But only a small percentage has achieved the game's version of victory, which in its most basic form involves beating sixteen levels plus a final boss in a single run (there are also secret levels that lead to different endings).

It's important to note that levels are assembled according to a plan; there's a precise method underneath the pseudo-random placement of enemies, shopkeepers, items, cave walls, and other elements.

Countless imitators have sprung up since *Spelunky's* original 2008 release, but none have managed to capture the same blend of authorship, randomization, and elegance.

TAKEAWAY: *By generating levels at runtime according to a simple formula,* Spelunky *pushes the player to master the rules and systems rather than the geometry of the game.*



It's easy to get lost in the relentless ambition and complexity of *Dwarf Fortress*, a game so involved that it takes months to learn. At the beginning of every world generates a detailed dwarven lineage dating back thousands of years. I once had a co-worker who would every day regale me with stories of sprawling dynasties and mighty fortresses carved into the sides of mountains. He told me of love, loss, betrayal, disaster, adventure, famine, rampaging fires, murder, comedy, sentient chickens gone haywire, and so, so much more.

For Tanya Short, what's most inspiring about Tarn and Zach Adams' epic dwarven life simulator is the underlying simplicity of the dozens of interconnected algorithmic systems. "Most people see *Dwarf Fortress* as a creationist work of genius engineering, but it's less like a combustion engine and more like a delicate mille-feuille," she says. "Each component has its own rules and obeys everything else's rules, but by layering almost infinitely, complexity derives from their interactions — which is rather like our own universe. Less vision, more evolution."

TAKEAWAY: Dwarf Fortress *is not notable for its apparent complexity but rather for the way its many simple components are layered together to procedurally generate everything the player encounters, such that every session the player gets a new unique, complex, and unpredictable dwarven world.*

Roguelikes are a dime a dozen in today's crowded indie landscape, but few of them tie procedural generation to thematics as deeply as *RymdResa*. Like *Elite* and the upcoming *No Man's Sky*, it procedurally generates a vast universe filled with more planets and space junk than anyone could ever hope to visit. But it does so with a singular purpose: to make the player feel lonely.

The player drifts through hostile, dark nothingness pierced occasionally by across various floating artifacts, other ships, and celestial bodies or by a sudden turn for disaster when it becomes apparent to the player that they probably can't make it back to safety. The beauty of *RymdResa* is that, snippets of poetry from the astronaut inside the player's ship aside, the whole tone and feel — danger, loneliness, peace, vastness — emerges from the generative algorithms that produce the world. These aesthetic elements are normally the province of hand-crafted level design, but here they are enhanced by the alienness and huge scale of the procgen algorithms that create a new universe each time you begin the game.

TAKEAWAY: *Procedural generation's otherworldly tendencies can accentuate tonal and thematic qualities to set the player's mood or lend a sense of alienness or danger to level design.*

Try playing any game in the *Civilization* series using only fixed, hand-crafted maps and it soon gets boring. The series' trademark focus on the Sid Meier adage of good design being about a series of interesting situational choices just doesn't work as well when you know who and what lies where. As Togelius notes, "it's absolutely crucial for the gameplay experience that you play a new map each time, which you will explore during the first half of a full game not knowing the layout of the world or which civilizations you will meet."

The key here is to appreciate why a game like *Civilization* — or *Age of Empires*, to use a real-time strategy correlate — needs procedural generation to succeed while many other games with similar systems of interaction thrive only with hand-crafted maps. Procedurally-generated maps promote exploration, but they are very difficult to make balanced for different play styles and strategies. Togelius was part of a team that tested procedurally-generated maps with Blizzard's long-lasting multiplayer-focused RTS *StarCraft* in 2013. "We were met with scorn by the gamer community because the maps were not 100 percent symmetric," he says.

TAKEAWAY: *Procedural generation can produce a new map every time players log in to ensure that they'll explore and experiment, though you'll get better multiplayer balance without it if discovery isn't a core gameplay loop.*

Some of the most forward-thinking and innovative work in procedurally-generated content comes from academic research rather than for-profit game development. A recent Georgia Tech project produced an automatic level generator for Super Mario Bros, with an algorithm that learned by watching YouTube videos. Another Georgia Tech project generates simple interactive narratives.

And a game-creating AI called ANGELINA procedurally generates entire games — all the way down to the mechanics — from natural language input (like a theme word). Its designer Michael Cook also runs the Procedural Generation Jam, which in 2014 produced a procedural murder mystery game, among other more intriguing things.

TAKEAWAY: *The academic and experimental indie development scenes are worth keeping tabs on, as they often offer novel generative solutions to hard content problems. They're particularly useful at the moment for help in developing systems that adapt a game's design to suit different play styles.*

As you study these games and others that use procedural generation, it's crucial to realize that it's a tool to solve problems. You can turn to procgen to give you lots of levels and boundless environments, or to add a random element to enemy designs so that they don't all look and act the same way — all things that are hard and time-consuming to develop by hand. But writing and testing the algorithms is a huge time investment in itself, especially if they interact with other systems.

Never lose sight of the fact that more variety isn't always going to result in a better game.

Procedural generation is not just a content-creation tool, either. Most of the games mentioned above — particularly *RymdResa* — use procgen for aesthetic purposes, to add a sense of alienness or the unknown or to add tonal color to the design.

Procedurally-generated content can fit whatever purpose you need — be that *Spore*-

like adaptive animation systems, *Diablo* and *Rogue*-esque dungeon layouts, infinite levels, evolving cities, alien landscapes, infinite replayability, grokkable systems, adaptive music, or just about anything else.

Using procgen all comes down to a simple question: What elements of your game could a well-defined algorithm or set of algorithms do better than you?