SPORE'S PLAYABLE PROCEDURAL CONTENT GENERATION

GILLIAN SMITH

Northeastern University
gi.smith@neu.edu

Spore (Maxis, 2008) was first conceived as *SimEverything*: a conceptual followup to the wildly successful *SimCity* (Maxis Software, 2003) and *The Sims* (Maxis, 2000). The game would operate at a galactic scale, with the player able to interact at multiple layers of abstraction, inspired by the concept of being able to zoom out on the universe by "powers of ten" (Johnson, 2013). The entire game would be enabled by a procedural simulation of the universe and procedural generation of planets and creatures. Will Wright, the father of the *Spore* concept, spoke excitedly about creatures and planets being represented as "DNA", which would enable the vastly reduced file sizes necessary to have a rich, shareable, procedural universe. The game spent years in development, breeding hype among reviewers and game enthusiasts. When *Spore* was finally released in 2008, it was met with mixed reviews—confusion and criticism over the shallow gameplay and poor model of evolution mingled with excitement over player creativity and user-created content.

Spore is a game that is broken into five core "stages". The cell stage has the player take on the role of a single-celled organism, fighting for survival and the right to evolve into a more complex form of life. The creature stage has the player take control of this fledgling lifeform as it interacts with other creatures in its world, guiding the creature's development and evolution to give it a competitive advantage. The tribal stage marks where the creature attains intelligence and the semblance of a society; in this stage, the player must gather food to help grow the tribe and socialize with other tribes on the planet. The civilization stage has the player grow their tribe into a larger civilization, competing with others on the planet for resources. Finally, in the space stage, the player has become the dominant civilization on the planet, and ventures into space to meet and conquer other planets.

In its gameplay, *Spore* initially seems to present itself as a single-player strategy game, inviting the player to build up a civilization that can compete on the galactic stage. However, the relatively simplistic design for the five "mini-game" phases it presents makes it a failure in this regard. Where the game shines is in its support for player creativity via a suite of design tools that allow the player to create professional-quality models and creatures and share them with other players. All content created by players is saved into the *Sporepedia* (Maxis, n.d.), a publicly-accessible, online repository.

The core tool to support player creativity sits at the transition from the cell stage and is then used throughout the creature stage. The creature creator lets the player design and "evolve" creatures using a library of existing creature parts—arms, legs, eyes, lips. The tool is so delightful to play with that it was initially released as a standalone toy before the full game was released. It includes extensive procedural support for creature creation; it will procedurally texture and animate any creature created within the tool without any assistance required from the player.

However, the creature creator also defines some of the controversy around *Spore*. Its failure to model evolution in any way turns *Spore* into a game about intelligent design, rather than a simulation of the universe.

To understand *Spore*—in both its successes and failures—is to understand procedural content generation (PCG), user-created content, and how the game fosters a relationship between them. The much-acclaimed design tools lean heavily on PCG to enable users to create content for the game, and its use to support player creativity fed both the excitement and controversy around *Spore.* However, the mismatch between how the player interfaces with this procedural support and the generator's actual design leads to a shallower model of evolution, resulting not only in criticism of the game's failed scientific underpinnings but also too much freedom for players to change their creatures to address gameplay challenges. This article explores the ways in which PCG is deeply integrated into *Spore*.

Nurturing Life from Cell to Creature

Let's begin our examination of *Spore* midway through the cell stage of the game. In this stage, the player navigates their cell around a vast procedurally generated ocean, filled with other cells at varying scales. The player has one main evolutionary decision to make in this stage of gameplay—whether their cell should be an herbivore or a carnivore. Herbivores seek out green plant life to eat, while carnivores chase down other, smaller cells for a snack. Both herbivores and carnivores are subject to attack from larger cells that are floating around in the primordial soup. When the "giants" of this playground attack each other, "DNA" is released for the player to pick up. These pieces of "DNA" correspond to components that can be added to the single-celled lifeform to alter how it moves, eats, attacks others, and defends itself.

The gameplay in this stage of the game feels almost meditative at times, and is heavily exploration-driven. The procedurally generated environment means that each time the player picks up this stage, they are experiencing vastly different content that makes it impossible to memorize paths. The simple rules for play and randomization of other content do not lend themselves to developing complex strategies for survival. Rather, the player is content to float around the pool, seeking out sustenance and occasionally breaking from this meditative state to either attack or defend oneself against other cells.

As the little cell eats and grows, it has several opportunities to seek out a mate, thus allowing the player to guide its evolutionary path by adding and removing cell parts. Entering a mating ritual launches a simplified version of the creature creator, where the player can manually alter the function of the organism using whatever cell parts are currently available, as well as changing its appearance through recoloring and retexturing it. The fiction for entering the creature creator—that a single-celled organism would reproduce sexually—is not based in science in the slightest. However, the transition is done playfully and the ability to make sweeping changes to even this tiny single-celled organism helps keep the player invested in their creation.

Figure 1 shows the use of the game's editing tools to create a more "evolved" cell, with flagella to help it move and turn more quickly and a stinger to defend itself against attacks from the rear. The player is given free reign for altering the cell, with the only limit being the number of cell parts that the player has found thus far.

Inside the creature creator, altering the cell's composition. The player drags components on and off of the main cell "body" to make changes, and is free to make as many changes as desired.

Once it has eaten enough from the ocean that it is as large as it can get, our little cell is ready to turn into a complex, multi-celled organism. In a shockingly vast evolutionary leap, the cell enters the creature creator, the player gives it legs, and the newborn creature toddles onto land to start its new life and push the player into a new stage of gameplay (Figures 2-3).

Figure 2. The cell becomes a creature with the addition of legs.

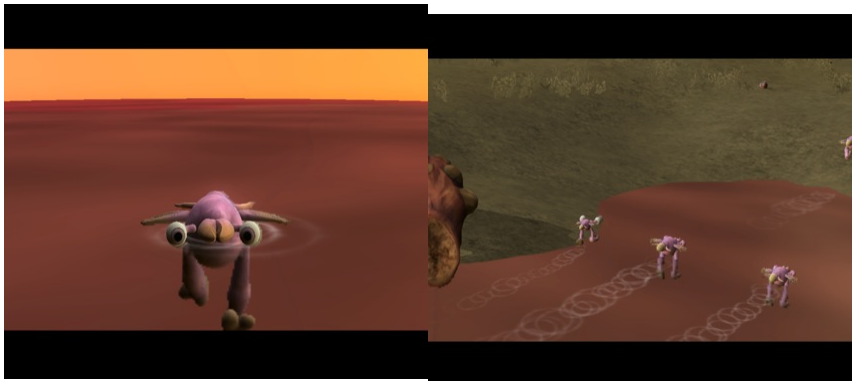

Figure 3. The new creature walks onto land.

In the cell stage, PCG was used to procedurally texture and color the creatures; however, this transition phase marks the first time that the game uses PCG to support the player in creating *functional* creatures. A procedural animation system determines how the cell should walk around space based on where the creature's legs are placed.

As in the cell stage, the creature stage involves the player collecting fragments of DNA that correspond to creature

components. When the player is ready to "evolve" their creature, they click the "mating call" button to find a mate for the creature. An elaborate (procedurally animated) mating dance occurs, the creature lays an egg1, and the creature creator loads so that the player can design the next generation to be born (Figures 4 – 5).

Figure 4. Having defeated her first enemy and found DNA, the creature goes back to the nest and engages in a mating ritual.

Figure 5. Inside the creature creator, it is possible to completely reconfigure the components that make up the creature, as well as its appearance. Our new creature bears little resemblance to the parent.

The creature creator really shines late in the creature stage, when the player has found a large number of creature components. The player can completely strip down their creature and reform it on each stage of its "evolution", if desired. This includes an ability to change the creature's spine length and shape, to swap out and add in functional creature components such as limbs and eyes, to provide the creature with decorative elements, and to alter its appearance. The procedural animation system reacts quickly, and the player will receive immediate feedback when altering limb placement. The creature immediately raises a newly added limb to stare at it admiringly and make a sound of approval.

Nurturing Creativity and a Community

After only a few moments of play with the creature creator, its broad appeal is obvious. The tool provides simple and casual play—one of the design motivations was to have the player feel like they are drawing with "magic crayons" (Gingold, 2003): simple tools that are natural and easy to create with, yet seamlessly imbued with artificial intelligence so that, as if by magic, the crayons create an amplification of what is actually drawn. The creature creator has the player interact as though they are creating a lifeless, static model, and the computer provides support to automatically turn that static model into a real character. The procedural animation and texturing systems underlying creature creator provide this "magic". The tools allow the player to feel creative agency, in that they can make meaningful decisions about the creature's appearance and some characteristics, while relegating the more technically challenging work of modeling, rigging, and animating to the computer. *Spore* kickstarted the growing trend in games to support user-created content. Games from the *Little Big Planet* (Media Molecule, 2008) series and, more recently, *Minecraft* (Persson, 2011) are built entirely around user-created content, but neither offer the kind

of procedural support that made *Spore*'s creation tools so simple and engaging.

In addition to the creation tools, *Spore* provides means for players to share what they have created with a broader audience, both within and outside the game world. Whenever the player saves a creature, it can be published to the *Sporepedia*, an online resource storing information about every creature, vehicle, and building that has ever been made for *Spore.* As of this writing (October 2014), the *Sporepedia* contains almost 186 million unique creations, with hundreds of new creatures made each day. Additionally, the game lets the player export video of their creatures that is uploaded to YouTube, offering an additional method for players to share their creative work with people outside of the game.

Within the game, other players' planets are randomly populated with creatures from *Sporepedia*, with the option of using creatures from a particular set of players by subscribing to individuals' "sporecasts". This is where *Spore* begins to blur the lines between procedural and user-created content, by using user-created content as a means to achieve a common goal of PCG: replayability. Each time the player begins a new game, the environment they play in is shaped by the creations of other players. Additionally, players can browse other creatures from *Sporepedia* within the game, download them to their own game and modify them further. This introduces a light social layer on top of the user-created content, and encourages a community of modding and sharing among players. This ability to modify "parent" creatures downloaded from *Sporepedia* again flirts with the idea of evolution without explicitly addressing it in game—there is a sense that a creature can have a parent, and that lineage is preserved in the metadata for the creature in *Sporepedia.* However, this concept of parentage for creatures is drastically removed from any basis in the actual science of evolution.

PCG Analysis

Spore is built heavily around the use of procedural content generation to enable user-created content. In order to understand how PCG is influencing the player's overall experience, see what makes the PCG in *Spore* successful, and unpack how it also contributes the game's controversy, it is necessary to dive deeper into how the PCG system is designed. This section will examine *Spore*'s use of PCG in comparison with other games, as well as how PCG interfaces with the game's mechanics, dynamics, and aesthetic goals (MDA) (Hunicke, LeBlanc, & Zubek, 2004).

Positioning Spore along Spectra

Broadly, there are three main spectra along which we can compare PCG systems, defined here by their endpoints: 1) data-intensive vs. process-intensive systems, 2) graphical vs. playable content, and 3) developer vs. player authoring. *Spore* manages to position itself at extremes along all these axes—the creature creator, as discussed later in this section, even sits at *both* ends of the data vs. process-intensive spectrum.

Many PCG algorithms are typically highly data-intensive, drawing from a rich library of human-authored content and recombining it using simple algorithms. This seems especially prevalent in commercial games where the use of data-intensive PCG means that the qualities of the content can be tightly controlled through crafting the palette of building blocks used by the system. Process-intensive systems, on the other hand, use more sophisticated algorithms with a small and limited set of building blocks; this is common in graphical PCG systems used to create smoke or water (Ebert, 2003) where the algorithm can dictate how individual particles flow to create emergent effects, and also in more recent advances in PCG, such as evolving weapons in *Galactic Arms Race* (Hastings, Guha, & Stanley, 2009).

Spore sits in an interesting position along this spectrum of PCG systems. As a design tool, the game provides players with a great deal of data—in the form of anatomical parts for the creatures—for the player to piece together using their own internal "algorithm" for deciding how the parts should fit together. However, the PCG system itself is highly process-intensive, using the raw 3D geometry created in the tool to determine how the creature should act in the world through a set of complex algorithms (Hecker, 2011).

Over time, PCG has moved from focusing on how to create graphical environments and effects to creating content that the player must deeply interact with, such as weapons and puzzles, to even creating entire game rulesets (Hendrikx, Meijer, Van der Velden, & Iosup, 2011). *Spore* sits closer to the "graphical" end of this spectrum; it uses sophisticated tools to support creating content that is only lightly interactive, from a design perspective. The environments that are generated procedurally and even the creatures do not need to support the player interacting with them deeply—the creature creator effectively creates the equivalent of canned animations that play when instructed by the player. The player does not need to manipulate these behaviors as part of gameplay.

Finally, it is interesting to look at PCG from the perspective of *whose* authoring is impacted by the system. The earliest PCG systems were created to allow a developer to create vast amounts of content, essentially as a form of data compression (Braben & Bell, 1984). In these systems, the developer is using the algorithms as a means for authoring non-varied content. Control over authoring is loosened in systems that use PCG for a variety of aesthetic reasons, supporting content that changes on each play and even adapts to player behavior. The next stage of evolution for this spectrum is supporting the player directly authoring content for games using procedural support. *Spore was*

one of the first games to use this form of PCG, letting the player shift difficult parts of their design burden onto the computer.

MDA Analysis of PCG in Spore

Understanding how *Spore*'s use of PCG works relative to other games that are PCG-enabled can help us see that the game is an exemplar of PCG in games. It firmly established PCG as a tool to support player authoring of rich, graphical content. To better understand this role that PCG plays in supporting player creativity, it is important to understand how the system fits in to the overall design of the game. The following analysis builds on a framework for understanding content generation in games that was previously published (Smith, 2014). The framework and vocabulary for describing content generation is based around Hunicke et al.'s MDA framework, and is intended to help understand the role that PCG plays towards a player experience and a game's design.

Aesthetically, Spore is a game about *discovery*—players discover new generated worlds and new creatures that they can create. These aesthetic goals are realized through the dynamics of *practicing the game mechanics* in different settings and *interacting with a community of players*. It's important to note that these dynamics do not arise directly from the use of PCG, but rather from its indirect use in supporting the user-created content. The use of PCG is *core* to the player's overall experience—without the content generator, there would be no user-created content, and without user-created content, there would be no *Spore*. Yet it is also acting firmly in a support role; it is user-created content that drives *Spore*'s replayability and makes the game appeal to players who want to flex their creative muscles.

Mechanically, the PCG in *Spore*'s creature creator plays an interesting role. It operates *online*, able to respond immediately to the player's actions when creating their creature. This design

decision has remarkable impact on the experience of using the creature creator. Instead of needing to wait until the creature has been fully fleshed out and placed into the world to see how it moves around, the player gets instantaneous feedback on the decisions they have made and how that impacts the creature's behavior.

The player has strong *compositional* control over the creatures they are creating—a rich library of creature parts, the ability to shape the creature's main body, and being allowed to place the parts anyone on that body means that there is an extremely wide variety of unique creatures that can be made using the tool. The player can create almost any creature they can imagine, and while all look stylistically similar in that they share a common art and animation style, the player can still take creative ownership over what they produced.

The other two facets of the mechanics of PCG in the creature creator are more interesting to examine, as the player's perception of the system they are interacting with is quite different from the reality. From the perspective of the player, they are *directly manipulating* the creature, and the underlying model for how the creature is assembled is as a combination of *experiential chunks* of geometry. Each chunk has a clear set of aesthetics and purpose for being added or removed from the creature. However, the underlying PCG system operates on a much different scale—the player is manipulating a creature that is then taken, as a whole, as input to the procedural animation system, and the knowledge representation used for the creature is of raw geometry and a skeleton. The procedural support in the creature creator does not understand the creature on the same scale as the player. The player understands semantic information about not just the appearance of the creature but also its function. The PCG system understands only a set of vertices that make up a creature to be textured and its underlying skeleton that must be animated. And it is here, at the layer of

understanding the mechanical systems to that make up the PCG of *Spore,* where the controversy surrounding *Spore's* treatment of evolution lies.

Mismatched Expectations

Spore is successful for exactly the same reason it has been considered a failure. PCG is used to support player creativity, and the system is expressive enough that it allows for the creation of millions of unique creatures. But the ways in which PCG is integrated into the tool also means it would never be able to support the a rich model of evolution in the way that many players had hoped, given the original marketing for the game.

The core issue is the mismatch between how players perceive their interaction with the tool vs. how the interaction is actually handled computationally. Players perceive that they are using a data-driven tool; a set of customized lego blocks where the player understands and communicates each blocks' "evolutionary" function for the creature. But this semantic information about the function of the creature is not at all considered in the procedural support for the tool, and it becomes the player's responsibility to maintain a model of evolution, if it is even desired. *Spore* is a game about creativity and player expression.

Spore's creature creator is not a tool intended to provide intelligent support for the science-based *design* of creatures. Rather, it is intended to help players realize a vision for *making* creatures. The input to the procedural system is merely a mesh and a skeleton, with no way for the creature creator to explicitly reason about evolution as part of its support to the player. Nor does this seem to be a goal of the system. The language of evolution is used only to lightly frame the game, not as a core mechanical component. The focus is on supporting players using an engaging, simple tool to realize their creative potential.

This is what makes the creature creator so powerful as a creative tool, and in turn what makes *Spore* a successful game. The player can complete forget that the underlying PCG system is present, and focus only on playing with the creativity toy. The ability for the system to rapidly produce an animation for any arbitrary geometry provides the player with the freedom to play with a wide variety of creature combinations and immediately see their creation come to life.

Conclusion

The successes and perceived failures of *Spore* are driven by its use of PCG to support player creativity and user-created content. Players can create a wide variety of polished, professional-quality creatures that are automatically textured and animated. Yet this user-created content lies at odds with some of the game's original stated goals of offering a rich simulation of the universe and evolution. The game touches upon evolutionary themes on occasion—allowing players to build creatures based on user-selected "parents" or producing newer and more capable creatures as a result of "mating"—but this theme is superficial. Evolution cannot be modeled in *Spore* because the PCG for its creature creator does not explicitly reason about it.

Despite failing to live up to some of its hype, *Spore* is a groundbreaking game. It was the first game to really carefully consider how to incorporate user-created content and provide rich support for player creativity. It provided intelligent, playful creativity tools that have not yet been paralleled in other games. The hype over *Spore* as an evolution simulator and a deep strategy game, and subsequent frustration over its inability to deliver on these promises, have obscured *Spore*'s actual contributions to game design: a deep focus on enabling player creativity with the appropriate supportive tools to make players feel like they are capable of creating vast worlds and sophisticated creatures with ease and enjoyment.

(1) It appears to always be the player-controlled creature that lays the egg. Either all playable creatures are biologically female, or the player is to assume that sex of individuals does not matter for sexual reproduction in *Spore*.

References

Braben, D., & Bell, I. (1984). *Elite (BBC Micro)*. Acornsoft.

Ebert, D. S. (2003). *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann.

Gingold, C. (2003, April). *Miniature Gardens and Magic Crayons: Games, Spaces, & Worlds* (Master of Science in Information, Design & Technology). Georgia Institute of Technology. Retrieved from http://levitylab.com/cog/writing/thesis/

Hastings, E. J., Guha, R. K., & Stanley, K. O. (2009). Automatic Content Generation in the Galactic Arms Race Video Game. *IEEE Transactions on Computational Intelligence and AI in Games*, *1*(4), 245–263. doi:10.1109/TCIAIG.2009.2038365

Hecker, C. (2011, February 25). My Liner Notes for Spore. Retrieved from http://chrishecker.com/My_liner_notes_for_spore

Hendrikx, M., Meijer, S., Van der Velden, J., & Iosup, A. (2011). Procedural Content Generation for Games: A Survey. *ACM Transactions on Multimedia Computing, Communications and Applications*.

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research. In *Proceedings of the 2004 AAAI Workshop on Challenges in Game Artificial Intelligence*. San Jose, California: AAAI Press.

Johnson, S. (2013, September 30). Spore: My View of the Elephant. Retrieved from http://www.designer-notes.com/?p=654

Maxis. (2000). *The Sims (PC Game)*. Electronic Arts.

Maxis. (2008). *Spore (PC Game)*. Electronic Arts.

Maxis. (n.d.). Sporepedia. Retrieved April 29, 2012, from http://www.spore.com/sporepedia

Maxis Software. (2003). *SimCity 4 (PC Game)*. Redwood City, CA: Electronic Arts.

Media Molecule. (2008). *Little Big Planet (Playstation 3)*. Sony Computer Entertainment.

Persson, M. (2011). *Minecraft (PC Game)*. Retrieved from http://www.mojang.com/notch/mario/

Smith, G. (2014). Understanding Procedural Content Generation: A Design-Centric Analysis of the Role of PCG in Games. In *Proceedings of the 2014 ACM Conference on Computer-Human Interaction*. Toronto, Canada.