Story Generation in Butterscotch Shenanigans’ Games: Inside and Out

By Maximilian Levine

Butterscotch Shenanigans (BS) is an indie game development studio based out of St. Louis founded by the three, tight-knit Coster brothers: Adam, Sam, and Seth [1]. The games *Crashlands* and *Levelhead* were their first foray into larger, nonmobile apps. *Crashlands* is a survival crafting game in which the player crafts varying tiers of resources and completes quests to advance the game’s rich sci-fi story. *Levelhead* is a platformer game in which players build and upload their own levels for others to play. Each game has in common their use of content-generative systems. Despite being very different games, the stories within, the stories surrounding, and the stories generated by, these two games bind them together: there is the story of a fight with death, and of learning and growing skills; and in the games, there is a contiguous growing lore about a whimsical and absurd universe, novel systemic/emergent based experiences that the player discovers, and absurd stories to inspire joy.

It was only in the last quarter of development though that the brothers began adding dialogue and story to *Crashlands*. To do so, Adam developed a browser-based app called the Crashlands Creator. Through this, the brothers were able to add and edit dialogue, stories, quests, and structures into the game. Similarly, for *Crashlands 2* (which is currently in development), they are creating a tool called the Game Changer, which easily allows for many aspects of content—enemy types, interactions, stats, etc.—to be tweaked while the game is running. The purpose is to be able to tune the systems to produce the best emergent stories. Creating this tool was inspired by their previous game, *Levelhead*; it was incredibly easy to bug test and experiment, since the built-in editor easily let them set up and test scenarios [3].

In *Crashlands*, the map is infinite, using Perlin noise to delineate biomes and spawn resources, and spawning enemies randomly. *Levelhead*, on the other hand, uses user-generated content, as players build and upload levels for others to play, again offloading developer work and aiming to produce near infinite content. In this way, the procedurally generated content of *Crashlands* and the user-generated content of *Levelhead* had much the same result: cheap, theoretically endless content generation. BS was in a great position to make *Levelhead*, after Adam gained web and database experience, as well as creating editors and tools like the Crashlands Creator.

BS's games take place in the same fictional universe, and are centered around the Bureau of Shipping (which, ironically, like the name of the studio, shortens to "BS"). In *Crashlands*, the protagonist, Flux Dabes, is an intergalactic trucker for the BS trying to deliver packages when they crash land on planet Woanope, which is rich with a resource called juice, which the antagonist is trying to harvest. There are tensions between the antagonist, alien inhabitants, and trying to deliver packages on time. In *Levelhead*, the player takes on the role of an employee within the BS. The ingenious framing of *Levelhead* puts the player in charge of training robots to deliver packages in "every possible delivery scenario." Thus, building levels and playing them trains the delivery robot, called GR-18, to deliver packages across challenging terrain. This fictional layer adds to the meaning of playing and building levels.

The brothers also pulled from real life. The second area of *Crashlands*, called the Bawg, is entirely made of living flesh, and the final boss there, Toomah, is a cancerous tumor. This storyline was a reference to Sam's battle with cancer. BS started development of *Crashlands* when Sam got cancer, and asked the question, "What if this is the last game I ever make? I want it to be important." Sam, who was the artist, began drawing the game's many assets while in chemotherapy. The three brothers then united under the studio name to make *Crashlands* a reality [2].

One of the reasons BS may lean into absurdity is their parents, who limited their video gaming in their youth and encouraged them to make a positive difference in the world. Making the games funny is a way to brighten players' days, and their step-father says, as long as making games engages their brains, it can't be a bad thing [2]. And BS's game stories are absurd: *Extreme Sloth Cycling* was about riding a sloth as a motorcycle, and *Towelfight 2* was about shooting animals out of your face/monocle.

In addition to *Crashlands*, all of BS's past titles harnessed randomness to produce content. So it will be informative to discuss the state of the art in procedural generation, or proc-gen. One of the first examples of proc-gen is from 1740, where a mix and match booklet of Mozart measures allowed users to form unique waltzes never heard before [4]. Other examples include astrology, mosaics, crochet, and, more recently, stable diffusion and GPT-3 [5].

For terminology, the procedure/algorithm is called a generator, and the results are called artifacts. It is a misconception in the industry that proc-gen can cheaply generate endless content. *No Man's Sky*, for example, boasts in its marketing having 18 quintillion planets, but players quickly became bored with the "samey" planets. Dr. Kate Compton [4] calls this the "10,000 bowls of oatmeal problem": technically they are unique, but they all taste the same. This is not always bad, for creating texture like the many trees that populate a landscape, but it doesn't work well for forming core game features.

To study generators, we first define an ideal generator G as one that has terminability (it always outputs something), fixed input size (it accepts input of a set length), and injectivity (different inputs give different outputs). An ideal generator then has these qualities: The length $|G|$ is the compressed size of its source code. The possibility space size is the number of unique artifacts, denoted $P(G)$, as well as $p(G)$, which is $\log_2 P(G)$; this happens to correspond to how many bits would be needed to label each artifact. Finally, there is Kolmogorov complexity, which is a measure of how many natural language words would be needed to describe an artifact: $K(A)$ is the description size of artifact A, $K^*(G)$ is the most complex, and $\overline{K}(G)$ is the average across all artifacts. Low K-complexity is simple and repetitive, but too high K-complexity is noisy, with no discernable pattern. Using these metrics, we can form several inequalities constraining a generator: $K^*(G) \geq p(G)$ because a label would need to have more bits than the largest artifact, $|G| + p(G) \geq K^*(G)$ because the size of the source code plus labeling all artifacts must of course have higher complexity than just one of the artifacts, and finally $C_1 + P(G) * C_0 + \sum |P_I| \geq |G|$ puts an upper bound on $|G|$ by essentially redefining the generator such that it is a list structure that is queried (C_1 and C_0 are commas) [5].

Using these inequalities, we can constrain a generator into a finite, triangular space, where there is a tradeoff between cost and scale. Cost represents the human effort, or encoded knowledge, that the generator must contain. Rabii puts the expense of human effort in terms of coffee [5], and Compton calls imbuing a generator with domain knowledge as "making an artist in a box" [4]. Compton recommends finding an expert or at least reading their writing; for example, to make *Spore*, she interviewed Disney animator John Cimino about how to model creatures. Scale represents how many artifacts are possible; decreasing scale can increase the quality of artifacts, but at the cost of having less artifacts. In *Minecraft*, for example, in order to add villages, changing scale was not an option, since there are a set 2^{64} seeds [6], thus human effort and knowledge about villages was needed. Many interesting results can follow from these inequalities, such as estimating possible artifacts just from the size of a program [5]!

Compton recognizes many methods for creating proc-gen content. Tiles, square, hexagonal, or otherwise, can form a foundation. Grammars/recursion can generate content; Dr. Jorin Dormans did this to great effect in *Unexplored*, where a key is placed behind a door that you need another key to get to, etc. Another method is distribution, where items are placed randomly, or via procedures like barnacling (placing smaller next to bigger), footing (making adjacent elements interact), or greebling (adding texture). Other methods than distribution include parametric and interpretive. Subtractive methods can then be used to manage results: either through blacklisting seeds manually, or by enforcing an algorithmic quality assessment.

Procedural Level Generation via Machine Learning (PLGML) is a ML method relying on levels annotated by humans, and as such only works for levels that have been annotated (such as *Super Mario Bros.*). To extend the usefulness of this, Jadhav proposes a method working on 2D tilemap games, using autoencoders to transfer learning from annotated games to those that haven't been human-annotated [7].

BS aims for their games to be designed, fun experiences, and they recognized the limitations of *Crashlands*' proc-gen. One limitation imposed by Perlin noise was that objects generated could only be one tile wide. And overall, it led to an "oatmeally" [5] experience where there was no reason to keep track of landmarks—anywhere you walked would produce the same result. So, BS is building *Crashlands 2* with a hand-built map—a departure from its prequel. The flexible *Levelhead* editor may have inspired this [4].

Storytelling is an important aspect of BS's games, whether generated traditionally, emergently, or algorithmically. It's certain their next games will create or curate powerful storytelling, no matter which methods are used.

Works Cited

[1] "Butterscotch shenanigans," bscotch.net, https://www.bscotch.net/about. (accessed Nov. 27, 2023).

[2] J. Reichmuth, Director, A. Summerfield, Producer, *A Crashlands Story: Dev Diary*, Jan 21, 2017 [Video]. St. Louis, MO: Forever an Astronaut, 2017 (accessed Nov. 27, 2023).

[3] Adam, Sam, Seth Coster, Speakers, *Coffee With Butterscotch*, 2017-23. St. Louis, MO: Butterscotch Shenanigans. [Podcast]. Available: podcast.bscotch.net. (accessed Nov. 27, 2023).

[4] K. Compton, "Practical Procedural Generation for Everyone," in *Game Developers Conference*, San Francisco, CA, 2017. [Online]. Available: https://www.youtube.com/watch?v=WumyfLEa6bU. (accessed Nov. 27, 2023).

[5] Y. Rabii, "Why Oatmeal is Cheap," in *Strange Loop Conference*, St. Louis, MO, 2023. [Online]. Available: https://www.youtube.com/watch?v=nq5C82Nn7XM. (accessed Nov. 27, 2023)

[6] David, "How many minecraft seeds are there?," Apex Hosting, https://apexminecrafthosting.com/how-many-minecraft-seeds-are-there/. (accessed Nov. 27, 2023).

[7] M. Jadhav and M. Guzdial, "Tile Embedding: A General Representation for Level Generation", *AIIDE*, vol. 17, no. 1, pp. 34-41, Oct. 2021. Available: https://ojs.aaai.org/index.php/AIIDE/article/view/18888. (accessed Nov. 27, 2023).