Chapter 12
Evaluating content generators

Noor Shaker, Gillian Smith, and Georgios N. Yannakakis

Abstract Evaluating your content generator is a very important task, but difficult to do well. Creating a game content generator in general is much easier than creating a good game content generator—but what is a "good" content generator? That depends very much on what you are trying to create and why. This chapter discusses the importance and the challenges of evaluating content generators, and more generally understanding a generator's strengths and weaknesses and suitability for your goals. In particular, we discuss two different approaches to evaluating content generators: visualizing the expressive range of generators, and using questionnaires to understand the impact of your generator on the player. These methods could broadly be called top-down and bottom-up methods for evaluating generators.

12.1 I created a generator, now what?

The entirety of this book thus far has been focused on how to create procedural content generators, using a variety of techniques and for many different purposes. We hope that, by now, you have gained an appreciation for the strengths and weaknesses of different approaches to PCG, and also the surprises that can come from writing a generative system. We imagine that you also have experienced some of the frustration that can come from debugging a generative system: "is the interesting level I created a fluke, a result of a bug, or a genuine result?"

Creating a generator is one thing; evaluating it is another. Regardless of the method followed, generators are evaluated on their ability to achieve the desired goals of the designer (or the computational designer). This chapter reviews methods for achieving that. Arguably, the generation of any content is trivial; the generation of *valuable* content for the task at hand, on the other hand, is a rather challenging procedure. Further, it is more challenging to generate content that is both *valuable* and *novel*.

What makes the evaluation of content (such as stories, levels, maps, etc.) difficult is the subjective nature of players, their large diversity and, on the other end of the design process, the designer's variant intents, styles, and goals [9]. Furthermore, content quality is affected by algorithmic stochasticity (such as metaheuristic search algorithms) and human stochasticity (such as unpredictable playing behaviour, style, and emotive responses) that affect content quality at large. All these factors are obviously hard to control in an empirical fashion.

In addition to factors that affect content quality, there are constraints (hard or soft ones) put forward by the designers, or imposed by other elements of game content that might conflict with the generated content (e.g. a generated level must be compatible with a puzzle). A PCG algorithm needs to be able to satisfy designer constraints as part of its quality evaluation. We have seen several types of such algorithms in this book, such as the answer-set programming approach in Chapter 8 and the feasible-infeasible two-population genetic algorithm used in Chapter 11. The generated results in these cases satisfy constraints, thereby they have a certain value for the designer (at least if the designer specified the correct constraints!). But *value* has varying degrees of success, and which constraints to choose are not always obvious, and that is where the methods and heuristics discussed in this chapter can help.

PCG can be viewed as a computational creator (either assisted or autonomous). One important aspect that has not been investigated in depth is the aesthetics and creativity of PCG within game design. How creative can an algorithm be? Is it deemed to have appreciation, skill, and imagination [4]? Evaluating creativity of current PCG algorithms, a case can be made that most of them possess skill but not creativity. Does the creator manage to explore novel combinations within a constrained space thereby resulting in *exploratory* game design creativity [1]; or, is on the other hand trying to break existing boundaries and constraints within game design to come up with entirely new designs, demonstrating *transformational* creativity [1]? If used in a mixed-initiative fashion, does it enhance the designer's creativity by boosting the possibility space for her? The appropriateness of evaluation methods for autonomous PCG creation or mixed-initiative co-creation [19] remains largely unexplored within both human and computational creativity research.

Content generators exhibit highly emergent behaviour, making it difficult to understand what the results of a particular generation algorithm might be when designing the system. When making a PCG system, we are also creating a large amount of content for players to experience, thus it is important to be able to evaluate how successful the generator is according to players who interact with the content. The next section highlights a number of factors that make evaluating content generators important.

12.2 Why is evaluation important?

There are several main reasons that we want to be able to evaluate procedural content generation systems:

1. To better understand their capabilities. It is very hard to understand what the capabilities of a content generator are solely by seeing individual instances of their output.
2. To confirm that we can make guarantees about generated content. If there are particular qualities of generated content that we want to be able to produce, it is important to be able to evaluate that those qualities are indeed present.
3. To more easily iterate upon the generator by seeing whether what it is capable of creating matches the programmer's intent. As with any creative endeavor, creating a procedural content generator involves reflection, iteration, and evaluation.
4. To be able to compare content generators to each other, despite different approaches. As the community of people creating procedural content generators continues to grow, it is important to be able to understand how we are making progress in relationship to the current state of the art.

This chapter describes strategies for evaluating content generators, both in terms of their capabilities as generative systems and in performing evaluations of the content that they create. The most important concept to remember when thinking of how to evaluate a generator is the following: make sure that the method you use to evaluate your generator is relevant to what it is you want to investigate and evaluate. If you want to be able to make the claim that your generator produces a wide variety of content, choose a method that explicitly examines qualities of the generator rather than individual pieces of content. If you want to be able to make the claim that players of a game that incorporates your generator find the experience more engaging, then it is more appropriate to evaluate the generator using a method that includes the player.

One of the ultimate goals of evaluating content generators is to check their ability to meet the goals they are intended to achieve while being designed. Looking at individual samples gives a very high-level overview of the capabilities of the generators but one would like for example to examine the frequency with which specific content is generated or the amount of variety in the designs produced by the system. It is therefore important to visualize the space of content covered by a generator. The effects of modifications made to the system can then be easily identified in the visualized content space as long as the dimensions according to which the content is plotted are carefully defined to reflect the goals intended when designing the system.

The remainder of the chapter covers two main approaches for evaluating content: the *top-down* approach using content generation statistics, in particular *expressivity measures* (see Section 12.3), and the *bottom-up* approach which associates content quality with user experience and direct or indirect content annotations (see Section 12.4).

12.3 Top-down evaluation via expressivity measures

A tempting way to evaluate the quality of a content generator is to simply view the content it creates and evaluate the artifacts subjectively and informally. But if a content generator is capable of creating thousands, even millions, of unique levels, it is not feasible to view all of the output to judge whether or not the generator is performing as desired. If you see five levels that are impressive, among 50 that you choose to ignore or re-generate, what does that say about the qualities of the content generator?

To solve this problem, it is possible to evaluate the expressive range of the level generator. *Expressive range* refers to the space of potential levels that the generator is capable of creating, including how biased it is towards creating particular kinds of content in that space [15]. This evaluation is performed by choosing metrics along which the content can be evaluated, and using those metrics as axes to define the space of possible content. A large number of pieces of content are then generated and evaluated according to the defined metrics and plotted in a heatmap. This heatmap can reveal biases in the generator, and comparisons of the heatmap across different sets of input parameters can show how controllable the generator is. Seeing how expressive range shifts according to input changes yields good insights on the controllability and the quality of your generator.

12.3.1 Visualizing expressive range

The expressive range of a content generator can be visualized as an N-dimensional space, where each dimension is a different quality of the generator that can be quantified. This allows us to imagine authoring level generators as creating these spaces of potential levels as a result of the emergent qualities of the system. By adding and removing rules from a rule-based generator, the shape of the generator's expressive range (also referred to as a generative space) can be altered.

For only two dimensions, the generative space can be visualized using a two-dimensional histogram. Higher dimensionality requires more sophisticated visualizations of generative spaces, which have not been deeply explored in PCG. This requires generating a representative sample of the content and ranking it according to the metrics; determining the amount of content to generate can be tricky. While it is simple for some systems to compute the total number of variations that can be generated, others may be able to create infinite variety. One method to ensure an acceptable sample size in the case of infinite content is to generate increasingly large amounts of content and visualize the expressive range, stopping when the graphs begin to look the same as for the previous, smaller amount of content. Expressive range charts are not intended to be perfect, mathematical proofs of variety; rather, they are a visualization that can help the creator of a generative system to understand its behaviour, and potential users of that system to understand its abilities.

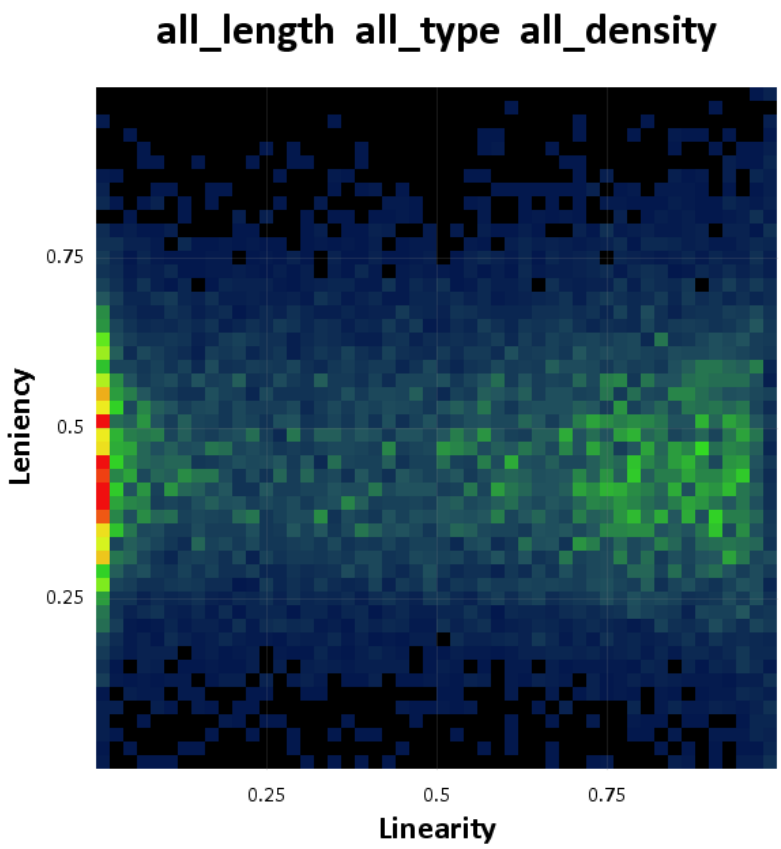

Fig. 12.1: The expressive range of the Launchpad level generator. Adapted from [16]

Figure 12.1 shows the expressive range of the Launchpad level generator [16]. Notice that there is one large hot-spot for creating medium-leniency, low-linearity levels, and another bias towards creating medium-leniency, high-linearity levels (more on these metrics in the next section). Understanding that the system is biased towards these areas forces the designer of the system to ask why such biases exist.

Figure 12.2 presents an alternative method for visualizing the expressive range of one of the content generators for *Infinite Mario Bros.* [13]. The figure shows different distributions of the levels according to three expressive measures defined: linearity, leniency, and density.

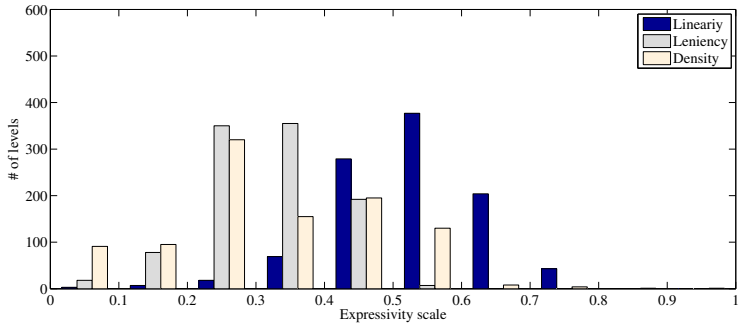

Fig. 12.2: The histograms of the linearity, leniency and density measures for one of the *Infinite Mario Bros.* generators. Adapted from [13]

12.3.2 Choosing appropriate metrics

The metrics used for any content generator are bound to vary based on the domain that content is being generated for. The "linearity" and "leniency" metrics used in the Launchpad generator mentioned above make sense in the context of 2D platforming levels, but perhaps not in the context of weapons for a space shooting game.

The important rule of thumb to remember when choosing metrics for your content generator is the following: strive to choose metrics that are as far as possible from the input parameters to the system. The goal of performing an expressive range evaluation is to understand the emergent properties of the generative system. Choosing a metric that is highly correlated to one that is used as an input parameter (e.g. if your generator accepts "difficulty" as an input and has "difficulty" as an expressive range metric) can only ever provide confirmatory results. If the system is specifically designed to create a particular kind of output, measuring for that output can only show that the algorithm operates as expected; it cannot deliver insight into unexpected behaviour or surprising output.

12.3.3 Understanding controllability

An important consideration in procedural content generation is understanding how well the generator can be controlled to produce different kinds of output, and especially how small changes in rule systems or priorities alter the expressivity of the system. If a designer requests that the system create shorter levels, will it still be capable of producing a broad range of content? Will adding a new rule to the grammar fundamentally alter the qualities of levels that can be produced?

Insight into these issues can be visualized by comparing expressive range graphs for different configurations of input parameters. That can be achieved through vi-

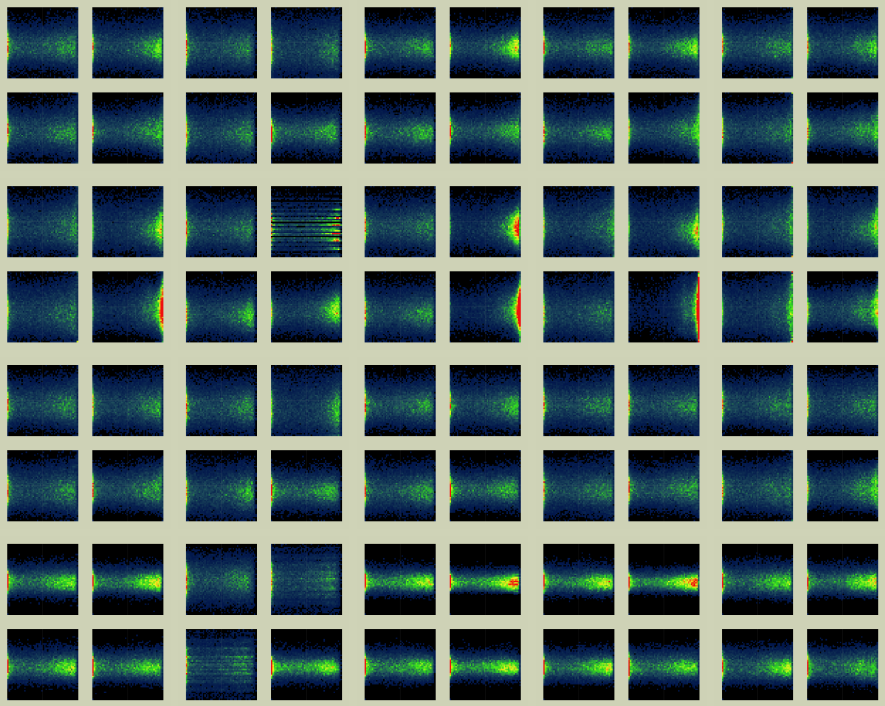

Fig. 12.3: Expressive ranges corresponding to different input parameters. Adapted from [16]

sual comparisons showing the expressive range of the Launchpad level generator when varying its rhythm input parameters (length of segment, pacing of segment, and type of rhythm; see Figure 12.3). Notice that, while several graphs look quite similar to each other, there are notable parameter configurations that lead to drastically different resulting spaces. Gaining insight into the problem is helpful not only after creating a system and wanting to evaluate it, but also during the development process itself as a debugging tool.

12.4 Bottom-up evaluation via players

Complementary to the top-down approaches for the evaluation of content generation, quantitative user studies can be of immense benefit for content quality assurance. The most obvious approach to evaluate the content experienced by players is to *explicitly* ask them about it. A game user study can involve a small number of dedicated players that will play through varying amounts of content or, alternatively, a crowd-sourced approach that can provide sufficient data to machine-learn content

evaluation functions (see [14, 8, 3] among others). It is important to note that any bottom-up content quality assessment can be complemented by content annotations of the designers of the game or other experts involved in content creation.

12.4.1 Which questionnaire should I use?

Estimates of content quality can be based on a player's preferences about the content retrieved through questionnaires during or after the gameplay. Content quality can be represented as a class, a scalar (or a vector of numbers)—such as continuous annotation throughout a level—or a relative strength (preference). Content quality can be characterized by a number of dimensions, such as novelty, feasibility, playability, and believability [2], depending on the content type evaluated and the scope of the evaluation.

There are several user protocol schemes and questionnaires one can adopt for the subjective annotation of content. The survey naturally asks players to self-report their experiences about content using directed questions which can vary from simple tick boxes to multiple choice items. Both the questions and the answers provided may vary from single words to sentences. Questionnaires can involve elements of the player experience (e.g. the Game Experience Questionnaire [7]), demographic data, or other factors that might impact the assessment of content quality (e.g. personality traits).

As a general recommendation for self-reporting it is suggested that subjects should be asked to *rank* the content they experience [21, 17]. Our recommendation is based on studies by Yannakakis and Martínez showing limitations when using the more common choice of ratings for subjective assessment, which provide empirical evidence for the superiority of rank-based questionnaire schemes for subjective annotation (in games and beyond) [20, 21, 10]. In particular, rating-based questionnaires generate lower inter-rater agreement and higher levels of inconsistency and order effects, and are dominated by a number of critical biases that make any post-hoc analysis questionable [21, 20, 18, 11].

12.4.2 Ways around the limitations of self-reporting

While self-reporting biases can be minimised by using rank-based questionnaires, not all disadvantages of self-reporting can be avoided this way [21]. Self-reports can be replaced by or triangulated with alternate measures of player experience such as physiological manifestations (of e.g. arousal, interest, and attention) [22] and/or behavioural playing patterns [5] that may map to content quality directly. For instance, a player being stuck at the same point of a map for several minutes might indicate player frustration and bad level design. Further, the rank-based approach can be enhanced if users are given the opportunity to view and annotate their video

playthroughs (e.g. as in [6]). Evidently such a survey protocol further eliminates reporting biases associated with memory and cognitive load which are present when users are asked to annotate the content in a post-experience manner.

Even though the data-driven approach for annotating content is user-centric and promotes the assessment of content quality directly by its end users, it has another core limitation which is associated with the potential treatment of subjects as random content evaluators. Thus, ideally, the generator should be coupled with selection mechanisms that will prune the available content (which arguably can be generated in massive amounts automatically) prior to it being presented to the players. On that basis, content can be evaluated via a sequence of logic operations without the need for player behavioural metrics or other input from players [12]. The satisfaction of constraints or logical operators can be coupled with simulations of AI agents that evaluate the quality of generated content (i.e. offline generate-and-test PCG). Further, the quality of content can be evaluated *implicitly* through player preferences during gameplay. It is natural to assume that the more content is selected, used, experienced, or altered during gameplay, the higher its value becomes. Studies in weapon generation, for instance, estimate the weapon's quality implicitly by its use during the game (see Section 11.3.2.1). and without explicitly asking the players about the quality of the weapons or other characteristics. Such an approach however relies heavily on the assumption that content use indicates content quality.

12.5 Summary

This chapter focused on the important but complex task of content evaluation, and discussed various methods to achieve it. In summary, content quality can either be assessed in a top-down fashion via well-designed content statistics (such as expressivity metrics) or in a bottom-up fashion by having players experience the content and assess its quality. Content evaluation via players may rely on both behavioural data and data for objective assessment such as physiology, such as the methods discussed in Chapter 10. This chapter, instead, focused on content evaluation via subjective user data as obtained from questionnaires. Arguably a hybrid approach involving both top-down and bottom-up methods can provide a more holistic approach to game content evaluation, and ultimately to *understanding* what the generator does, and whether it is suitable for the job the designer wants to use it for.

References

1. Boden, M.A.: What is creativity? In: M.A. Boden (ed.) Dimensions of Creativity, pp. 75–117. MIT Press (1994)
2. Camilleri, E., Yannakakis, G.N., Dingli, A.: Platformer level design for player believability. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (2016)

3. Chernova, S., Orkin, J., Breazeal, C.: Crowdsourcing HRI through online multiplayer games. In: AAAI Fall Symposium: Dialog with Robots, pp. 14–19 (2010)
4. Colton, S.: Creativity versus the perception of creativity in computational systems. In: Proceedings of the AAAI Spring Symposium on Creative Intelligent Systems, pp. 14–20 (2008)
5. Drachen, A., Thurau, C., Togelius, J., Yannakakis, G.N., Bauckhage, C.: Game data mining. In: M. Seif El-Nasr, A. Drachen, A. Canossa (eds.) Game Analytics, pp. 205–253. Springer (2013)
6. Holmgård, C., Yannakakis, G.N., Martínez, H.P., Karstoft, K.I.: To rank or to classify? Annotating stress for reliable PTSD profiling. In: Proceedings of the International Conference on Affective Computing and Intelligent Interaction, pp. 719–725 (2015)
7. IJsselsteijn, W., De Kort, Y., Poels, K., Jurgelionis, A., Bellotti, F.: Characterising and measuring user experiences in digital games. In: Proceedings of the ACE 2007 Workshop on Methods for Evaluating Games (2007)
8. Li, B., Lee-Urban, S., Appling, D.S., Riedl, M.O.: Crowdsourcing narrative intelligence. Advances in Cognitive Systems **2**, 25–42 (2012)
9. Liapis, A., Yannakakis, G.N., Togelius, J.: Towards a generic method of evaluating game levels. In: Proceedings of the Artificial Intelligence for Interactive Digital Entertainment Conference, pp. 30–36 (2013)
10. Martínez, H.P., Yannakakis, G.N., Hallam, J.: Don't classify ratings of affect; rank them! IEEE Transactions on Affective Computing **5**(3), 314–326 (2014)
11. Metallinou, A., Narayanan, S.: Annotation and processing of continuous emotional attributes: Challenges and opportunities. In: Proceedings of the IEEE Conference on Automatic Face and Gesture Recognition (2013)
12. Nelson, M.J.: Game metrics without players: Strategies for understanding game artifacts. In: Proceedings of the First AIIDE Workshop on AI in the Game-Design Process, pp. 14–18 (2011)
13. Shaker, N., Nicolau, M., Yannakakis, G.N., Togelius, J., ONeill, M.: Evolving levels for Super Mario Bros. using grammatical evolution. In: Proceedings of the IEEE Conference on Computational Intelligence and Games, pp. 304–311 (2012)
14. Shaker, N., Yannakakis, G., Togelius, J.: Crowd-sourcing the aesthetics of platform games. IEEE Transactions on Computational Intelligence and AI in Games **5**(3), 276–290 (2013)
15. Smith, G., Whitehead, J.: Analyzing the expressive range of a level generator. In: Proceedings of the First Workshop on Procedural Content Generation in Games (2010)
16. Smith, G., Whitehead, J., Mateas, M., Treanor, M., March, J., Cha, M.: Launchpad: A rhythm-based level generator for 2-d platformers. IEEE Transactions on Computational Intelligence and AI in Games **3**(1), 1–16 (2011)
17. Yannakakis, G.N.: Preference learning for affective modeling. In: Proceedings of the International Conference on Affective Computing and Intelligent Interaction (2009)
18. Yannakakis, G.N., Hallam, J.: Ranking vs. preference: A comparative study of self-reporting. In: Proceedings of the International Conference on Affective Computing and Intelligent Interaction, pp. 437–446 (2011)
19. Yannakakis, G.N., Liapis, A., Alexopoulos, C.: Mixed-initiative co-creativity. In: Proceedings of the 9th Conference on the Foundations of Digital Games (2014)
20. Yannakakis, G.N., Martínez, H.P.: Grounding truth via ordinal annotation. In: Proceedings of the International Conference on Affective Computing and Intelligent Interaction, pp. 574–580 (2015)
21. Yannakakis, G.N., Martínez, H.P.: Ratings are overrated! Frontiers in ICT **2**, 13 (2015)
22. Yannakakis, G.N., Martínez, H.P., Garbarino, M.: Psychophysiology in games. In: K. Karpouzis, G.N. Yannakakis (eds.) Emotion in Games: Theory and Praxis. Springer (2016)