Proceedings of the Fourteenth Artificial Intelligence and
Interactive Digital Entertainment Conference (AIIDE 2018)

Expanding Expressive Range: Evaluation Methodologies for Procedural Content Generation

Adam Summerville
California State Polytechnic University, Pomona
asummerville@cpp.edu

Abstract

Procedural Content Generation (PCG) has been a part of video games for the majority of their existence and have been an area of active research over the past decade. However, despite the interest in PCG there is no commonly accepted methodology for assessing and analyzing a generator. Furthermore, the recent trend towards machine learned PCG techniques commonly state the goal of learning the design within the original content, but there has been little assessment of whether these techniques actually achieve this goal.

This paper presents a number of techniques for the assessment and analysis of PCG systems, allowing practitioners and researchers better insight into the strengths and weaknesses of these systems, allowing for better comparison of systems, and reducing the reliance on ad-hoc, cherry-picking-prone techniques.

Introduction

Procedural Content Generation (PCG) has been a part of video games for the majority of their existence. *Beneath Apple Manor* (Worth 1986), the first known game with PCG, was released in 1978. In the last decade, PCG has become a focus area for research (Smith and Mateas 2011; Smith, Whitehead, and Mateas 2011; Togelius et al. 2010), with a goal being the understanding of different techniques, the properties they afford, and the ability to generate levels with certain properties or style.

While the foremost goal of most procedural generation systems is that of producing high quality, novel content there is still no agreed upon methodology for assessing whether a generator has achieved that goal. Furthermore, recently there has been a focus on PCG via Machine Learning (PGCML) (Summerville et al. 2017) wherein the generator is trained on content with the goal of learning the design latent within that content – but methods to assess whether these techniques succeed in that goal are still in their infancy.

This paper introduces a number of techniques focused on analyzing the capabilities of a generator. Broadly, these fall into two categories – the analysis of the generative space of a generator and the selection of pieces of content for showcasing. Each is important in its own right – these generators are capable of producing a large (if not infinite) amount of content, meaning that it is impossible to assess the generator solely on the basis of individual pieces of content, which is why generative space analysis is important, while on the other hand all that matters is the generated content, hence why assessing individual pieces is important.

The contributions of this paper are three-fold:

1. A method for visualizing the expressive range of a set of content that works across a large number of dimensions and in low sample size situations
2. A method for testing how well a generator has learned the latent distribution of its original dataset
3. Methods for the selection of content that are robust to cherry-picking and help illuminate properties of the generator that produced the content

Related Work

In the domain of procedural generation of game level content, expressive range has been the dominant qualitative exploration tool. Expressive range is an idea put forth by Smith and Whitehead (Smith and Whitehead 2010; Smith et al. 2011) whereby two metrics are chosen (e.g., for platformers Smith and Whitehead settled on *linearity*, a measure of how closely the ground of a level follows a line, and *leniency*, a proxy for how difficult a level is) as the axes for a density plot, showing which areas of the generative space a generator tends to explore. While the metrics that make up the expressive range are usually quantitative in nature, the act of the analysis is qualitative, in that it relies on the subjective analysis of the reader.

Danesh, a tool from Cook et al. (Cook, Gow, and Colton 2016), is designed to help bridge the gap between a generator and its expressive range. For many generators, it can be difficult to determine which set of parameters will achieve the desired outcome. *Danesh* displays the expressive range of a generator and allows users to target areas of the expressive range they find desirable, and then tries to find parameterizations that target that region of the expressive range.

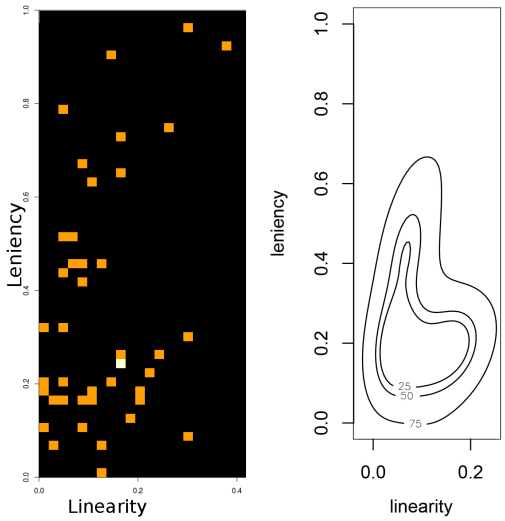
In Smith's thesis (Smith 2012) she discusses the expressive range of *Launchpad* (Smith et al. 2011), stating "There is good coverage of the generative space for both linearity and leniency." However, this claim goes unexplored. This question is tangentially addressed by Teng (Teng and

Bidarra 2017). Exploring the expressive range of his generator, he finds a region of the expressive range of two metrics (the connectivity and density of road networks) goes uncovered by his generator (e.g., the generator can not produce road networks of high density and low connectivity, as these properties are entangled); however, this is not a failure of the generator, but rather a sign that the generator is producing plausible road networks – a victory for the approach. As such, it is incorrect to say that a large expressive range is unequivocally a good quality for a generator.

A practice that is common in the field of procedural content generation is to pick some subset of metrics, and, instead of displaying the expressive range density plots, list summary statistics for the generator (Dahlskog, Togelius, and Nelson 2014; Shaker et al. 2012; Stephenson and Renz 2016; Mariño and Lelis 2016). While the pragmatics are understandable, as authors have limited space to report results and large scale comparisons require a table to report all combinations; however, for generator evaluation, this is an unsatisfactory approach. The shape and density of the sampling distributions is important, and simple measures such as mean and standard deviation (and by extension, tests such as Student's t-test which rely on said measures) tell a very incomplete story.

Anscombe's Quartet (Anscombe 1973) is quartet of datasets, each demonstrating pairs of $< x, y >$ coordinates that all have the same mean, variance, correlation coefficient, maximum likelihood linear regression, and coefficient of determination for the linear regression. However, upon visual inspection, it is quite easy to see that the the data-generating distribution for the four examples is quite different. This speaks to the danger of only reporting statistical properties, showcasing the need for the visual inspection of the sampling distributions and qualitative evaluation of both generated artifacts. However, while summary statistics are not sufficient, it is desirable to have statistics that allow for the comparison of generators' distributions, but these statistics must be holistic, so as to actually capture the shape and density of the distributions and can not just be univariate measures.

As generative models have become more popular in the image generation domain, the desire to find metrics that determine whether a generator is achieving its goal have begun to be researched by that community. One such measure that has become the de facto standard for Generative Adversarial Network based approaches is that of Inception Score (IS) (Salimans et al. 2016). IS utilizes a pre-existing image classification model, the Inception v3 Network (Szegedy et al.) pre-trained on the ImageNet corpus (Deng et al. 2009). However, while the approach is appealing (and has been shown to correlate well with human perceptual ratings (Salimans et al. 2016)), it has two limitations that make it unappealing for procedural game content generators. First, it relies on a pre-existing vetted classification system. While image classification has been an area of research for decades, there is no such community (or even an analogue for a similar problem domain) in the field of games. Any such classifier would, almost by necessity, have to be trained in a bespoke manner for every new game, and would lack the external validation that a state-of-the-art network like the Inception v3 comes ready with. However, while this practical consideration likely precludes using a measure such as the Inception Score (or similar Inception based measures such as the Fréchet Inception Distance (Heusel et al. 2017)), there are fundamental flaws in the Inception Score as discussed by Baratt and Sharma (Barratt and Sharma 2018).



(a) Expressive range plot of the above and below ground rooms of *Super Mario Bros.* Note that only one cell has more than one data point (the white data cell).

(b) Density estimate for *Super Mario Bros.* The contours represent the $25^{th}, 50^{th}$, and 75^{th} percentiles of the estimate. The KDE makes it much easier to see the pear shaped relationship between linearity and leniency in *Super Mario Bros.*

Figure 1

Analysis of the Generative Space

Expressive range analysis is a qualitative visual assessment of the generative space of a generator. While this is a good practice in general, it is especially important for a machine learned generator, but only in so far as a comparison tool between the metric space of the generator and that of the original corpus. The goal of a PCGML system is to match the metric properties of the original corpus – the design latent within. While this qualitative analysis is insufficient for rigorous comparisons between generators, it allows for exploratory understanding of the type of content producible by a generator – and ways in which the generator is biased away from the type of content found in the original corpus.

The analyses presented here will be focused on *Super Mario Bros.* with the data being shown coming from the set of above and below ground levels (there are no dungeons or underwater levels) and generated levels coming from Snodgrass and Ontañón (Snodgrass and Ontañón 2015), Guz-

dial and Riedl (Guzdial and Riedl 2015), and Summerville and Mateas (Summerville and Mateas 2016) – with most analysis having been performed on the most easily accessible – https://www.dropbox.com/sh/3mwie9yg3oznve0/AAAC9iMYgXuwyuTMrNGy20y_athe publicly available levels of Summerville and Mateas.

The classic two dimensional histogram used by Smith and Whitehead is poorly suited to the evaluation of the metric space of the rooms of most games. Smith and Whitehead focused on the analysis of generators that could easily create thousands of pieces of content, for which the histogram acted as a good estimate of density. However, most games tend to have a relatively small number of rooms – on the order of a few dozen – (although a few reach into the thousands, e.g. *N*++), which leads to extremely sparse histograms, hence the need for density estimation. The histogram for the above and below ground rooms from *Super Mario Bros.* with the same number of bins as used by Smith (Smith 2012) can be seen in figure 1a. Note that only one cell has more than one data point.

Kernel Density Estimation (KDE) (Rosenblatt 1956) is a statistical method for estimating the probability density function of a distribution from a set of sampled data. The heat density plots of Smith and Whitehead were an attempt to analyze these distributions; however, KDE seeks to find analytic functions that best describe the sampled results. The result is a cleaner system to determine the actual generative space of a PCG system, and a way to more accurately compare two generators.

Given $(x_1, x_2, ..., x_n)$ samples, then the kernel density estimator of the true probability density function, f, is:

$\hat{f_H}(x) = \frac{1}{n} \sum_{i=1}^{n} K_H(x - x_i)$

where H is a positive definite symmetric bandwidth matrix of the KDE, and K is the multi-variate kernel. The bandwidth is a hyper-parameter of the KDE that acts to smooth the supplied data points. As $H \rightarrow \infty$ the KDE become smoother and smoother, loosing all shape. Conversely, as $H \rightarrow 0$ the KDE becomes closer and closer to the supplied data. The smoothness of the density estimate allows for much easier understanding of the density function of the data as can be seen in figure 1b.

While this two dimensional comparison is useful, as shown in the previous chapter, there are a large number of metrics that one might wish to explore. While *Danesh* allows for the switching of what metrics are viewed, this interaction pattern makes it difficult to discern higher dimensional structure in the generative space. While a three dimensional plot can be navigable in an interactive setting (Ryan et al.), it is difficult to discern the the structure via a static image. Corner plots (Foreman-Mackey 2016) are a visualization technique that allows for an arbitrary number of dimensions to be viewed holistically, and have been used previously (Summerville and Mateas 2016; Summerville et al. 2016). Figure 2 shows a comparison between the generated levels of Summerville and Mateas compared to those from the original game across a larger set of metrics. The metrics shown here are:

- e – The frequency of the room taken up by empty space

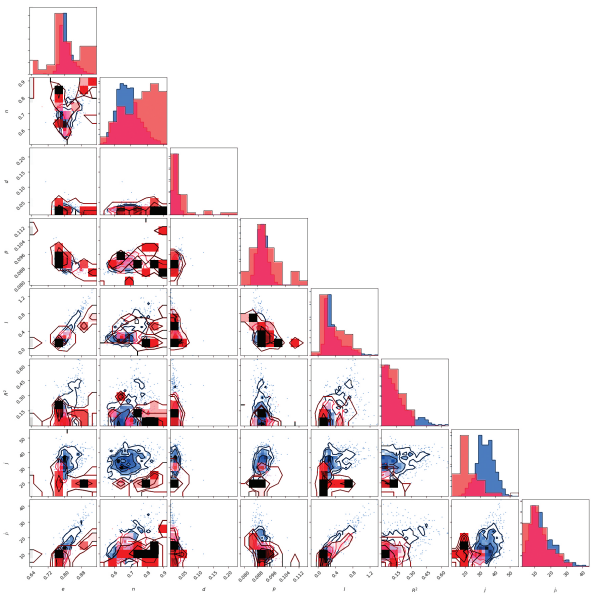


Figure 2: Corner plot for *Super Mario Bros.* (Red) and the Summerville and Mateas generator (Blue)

- n – The negative space of the room, i.e., the percentage of empty space that is actually reachable by the player
- d – The frequency of the room taken up by "interesting" tiles, i.e., tiles that are not simply solid or empty
- p – The frequency of the room taken up by the optimal path through the room
- l – The leniency of the room, which is defined as the number of enemies plus the number of gaps minus the number of rewards
- R^2 – The linearity of the room, i.e., how close the room can be fit to a line
- j – The number of jumps in the room, i.e., the number of times the optimal path jumped
- j_i – The number of meaningful jumps in the room. A meaningful jump is a jump that was induced either via the presence of an enemy or the presence of a gap.

While a broader expansion of expressive range is useful as a design paradigm, it is only one way in which the generative space of a generator can be analyzed. Snodgrass et al. Expressive volume represents a natural, quantitative follow on to the, now standard, expressive range analysis of procedural generators. In particular, it allows one to understand the "width" of the generative space via a metric that is relatively robust. However, as discussed, it is insufficient as the sole evaluative criterion, as a desirable criterion should assess not just the relative sizes of the metric spaces of generators, but also the location and shapes of these metric spaces. In the following section, I will discuss a robust multi-dimensional method for the comparison of generative spaces.

Where expressive volume has failings in terms of not capturing the locality of the density function, many traditional

Data Set	e-distance	p-value
Summerville and Mateas	8.2941	**0.4419**
Snodgrass and Ontañón MdMC	560.85	0.0233
Snodgrass and Ontañón MdMC , Playability Constrained	44.592	0.04651
Guzdial and Riedl	497.15	0.0233
Original Levels 10-fold Bootstrapping	12.039	**0.1395**

Table 1: Comparison using e-distance for a linearity, leniency, number of jumps, and negative space. Lower e-distance represents a smaller distance between the distributions, leading to a higher probability that null hypothesis of equal distributions is not rejected.

one-dimensional tests succeed. There are a number of unidimensional tests for whether two samples came from the same distribution. In the case of evaluation for a PCGML, we know a priori that the samples came from different distributions, but the goal is still the same. If two samples (e.g., a sample of levels from a generator and the original levels) are believed to have come from the same distribution, then the generator has achieved its goal.

The most common of these tests is that of Student's t-test, a common test for whether two samples came from the same distribution. However, a key assumption of the t-test is that the underlying distributions are Gaussian Normal distributions. From visual inspection, it is easy to see that most of the metrics considered are not normally distributed. Of the metrics considered in figure 2 only path length, p, and number of jumps, j, might plausibly be normally distributed. Thus, the t-test is unsuitable for the general case of comparing two distributions. The Mann-Whitney test does not have the same normality assumption, but the test relies on the assumption that the two distributions will both have identical shape and scale for the test to be valid, again an unlikely assumption. This holds true for the Kruskal–Wallis one-way analysis of variance, as it is based on the same ranking and summing as the Mann-Whitney test.

e-distance by Székely and Rizzo (Székely and Rizzo 2013) is a statistical comparison meant to alleviate these concerns. e-distance (e for energy, as the metric takes inspiration from Newton's gravitation potential energy) is rotationally invariant in a multidimensional space, meaning that it is not subject to the distance scaling concerns of the Wasserstein distance.

p-values are given via a bootstrap procedure where the populations are repeatedly resampled B times, the $T_{n_1 n_2, 0<b<B}$ is calculated for each resampling, and the hypothesis is rejected if the observed $T_{n_1 n_2}$ is greater than $100(1-\alpha)\%$ of the resampled values (for confidence level α).

This statistic is exactly what is desired for the comparison of two generative spaces.

- It makes no assumptions about the shape or scale of the distributions
- It allows for a multi-dimensional comparison of metric spaces
- It is rotation invariant and scale equivariant
- It provides a distance metric in addition to a statistical test, allowing for the comparison of how close a generative space is to another even if a statistically significant difference is found.

Table 1 shows the results of an experiment comparing multiple generators' multivariate expressive range distributions against that of the original rooms from *Super Mario Bros.*. The experiment compares the distributions using e-distance for a linearity, leniency, number of jumps, and negative space. Lower e-distance represents a smaller distance between the distributions, leading to a higher probability that null hypothesis of equal distributions is not rejected. Included for comparison is a 10-fold bootstrapping whereby the original levels are repeatedly split into different groups and compared against each other – shown is the average e-distance and associated p-value.

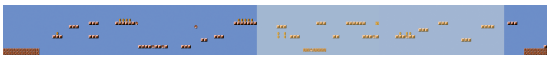
Included in this experiment are the levels generated by the generator of Guzdial and Riedl, two generators from Snodgrass and Ontañón, and the Snaking-Depth-Path generator of Summerville and Mateas (chosen as it had the best reported playability percentage). The levels from Snodgrass and Ontañón include a baseline Multi-dimensional Markov Chain (MdMC) generator discussed in (Snodgrass and Ontañón 2015) and a MdMC generator with playability constraints as discussed in (Snodgrass and Ontañón 2016).

We see that both of the two MdMC methods and the generator of Guzdial and Riedl are found to be statistically different from the original rooms. However, we see that the generator of Summerville and Mateas is not found to be statistically significantly different from the original levels. In fact, it is closer to the distribution of original rooms than a repeated sampling of the original rooms against themselves. This is in part due to the wide variability in the original rooms and, certainly, some subsets of the original rooms are much closer (e.g., e-distance 3.5136, p-value 0.9302) than others (e.g., e-distance 18.015, p-value 0.04651). Of course, a caveat of this approach is that a generator that simply copied from the original dataset would have a very low e-distance from the originals. This can be remedied by detecting plagiarism (as discussed below).

Analysis of Individual Pieces of Content

The previous section discussed methods for the large-scale analysis and comparison of the generative space covered by a generator. While this is certainly a useful practice for PCG researchers and practitioners, at some level, all that matters is the actual individual pieces of content themselves. However, the practice of showing content for a generator is a

(a) Random seed 692 from Summerville and Mateas.



(b) World 1-3 from *Super Mario Bros. 2 (JP)*.

Figure 3: The room generated by the Summerville and Mateas generator with the highest amount of plagiarism from the original rooms (top) and the room it takes from (bottom). While the portion is of decent size (45% of the original room), the generated room is very different, using the copied piece as a minor segment of the total room.

(a) Random seed 100 from Summerville and Mateas.

(b) Random seed 478 from Summerville and Mateas.

Figure 4: The two rooms generated by the Summerville and Mateas generator with the highest amount of self plagiarism. The two rooms start off identically (the first 20% of the rooms are identical), but they quickly diverge. This amount of plagiarism is not too terribly different than that found between the original rooms.

fraught one, as so far there has been little principle in actually assessing the quality of a generator via the individual content.

The PCG community as a whole has not adopted any methodology towards addressing these concerns and up to now have taken an ad-hoc approach toward choosing content. Snodgrass and Ontañón (Snodgrass and Ontañón 2014; Snodgrass and Ontañón 2015; Snodgrass and Ontañón 2016) show content based on parameterizations of their models. Hoover et al. chose a famous piece of content to build off of (Hoover, Togelius, and Yannakis 2015). Guzdial and Riedl choose "representative" content (Guzdial and Riedl 2015), but make no distinction as to what that means. Summerville and Mateas (Summerville and Mateas 2016) show "randomly" selected content in addition to making a wider set available.

The following sections will discuss two ways of selecting from a pool of content, such that the worst possible selections are presented (for various definitions of worst), providing a lower-bound on the capabilities of a generator.

Plagiarism As Selection Criterion

A major goal for PCGML systems is to generalize from a – usually small – pool of input, so as to be able to generate new content that is similar, but not identical (i.e., it could conceivably have come from the same designer). While it is undesirable for a PCGML system to copy wholesale from the input corpus, it is something of a philosophical question as to what the dividing line is that delineates between a piece of content that is copying from the input corpus and that which has learned from and generalized. Certainly, all of the individual words found in this document can be found in other documents. The same is most likely true for a large percentage of the pairs of words. So on for triplets of words. While these base atoms of content (words) are found in other documents, it would be absurd to claim this as plagiarism. That being said, there is certainly some point at which a certain string of words being found together would be plagiarism. 10? Maybe, if it were a very important 10 words. 100? It would have to be a very extenuating set of circumstances for this to not read as plagiarism. 1000? Most certainly. Sidestepping this, the goal of this is to not to provide that line in the sand, but merely to draw it to the attention of the reader who can make their own judgment.

For a PCGML system, determining the amount of plagiarism is relatively straight-forward. The author selects an original piece of content, according to some criterion, and then shows it alongside the generated content that has the most amount of plagiarism from that piece of content.

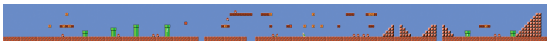
Potential criteria for selecting the exemplar content:

- The pair that has the most plagiarism – For each piece of content in the input corpus, find the piece of generated content with the most plagiarism. Present the pair with the highest amount of plagiarism (absolute or as percentage of the content pieces – for variable sized content)
- Pick a particularly important piece of original content – Perhaps there is a particularly iconic piece of content (e.g., *Super Mario Bros.* World 1-1). Find the piece of generated content that has the highest amount of plagiarism from the iconic content.
- The most self-plagiarized original content – For all pairs of content from the input corpus, find the pair that plagiarize the most from each other.

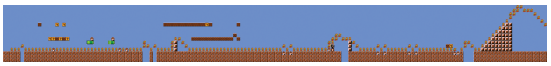
This last criterion gives us insight into providing a guiding principle for when plagiarism is an issue. By assessing the amount of plagiarism found in the original dataset, we can determine how much plagiarism might be expected from a similar piece of content. While, in a vacuum, it might seem bad for a room to copy 25% of its geometry from a piece of the original content, if the average amount of self-plagiarism is 35%, then this is to be expected. E.g., in *Super Mario Bros.* the highest amount of self-plagiarism accounts for 16% of the levels. By presenting the piece of generated content that takes the most amount of content, a reader can inspect the content to determine the degree to which the generator is memorizing from the original dataset. Figure 3 shows the room generated by the Summerville and Mateas generator which plagiarizes the most from the original *Super Mario Bros.* Furthermore, while the first two criteria are not suitable to classic PCG systems, the amount of self-plagiarism is useful to understand how likely the generator is to generate content that will be deemed sameish.

Metric Distance as Selection Criterion

Given the importance placed on metrics as evaluative criteria for understanding the generative space of a generator, it seems important to map this understanding back to the individual pieces of content. While acting as a principled



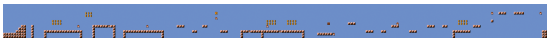
(a) World 1-1 from *Super Mario Bros.*



(b) Random seed 447 from Summerville and Mateas.
World 1-1 and the Summerville and Mateas generated room closest to it in the metrics of negative space, linearity, leniency, and jumps. The generated room has a few more gaps but is a relatively straight forward experience, like 1-1.

(c) World 4-2 from *Super Mario Bros. 2* (JP)

(d) Random seed 186 from Summerville and Mateas.
The room generated by Summerville and Mateas (bottom), closer to any of the original content than any other piece of generated content, and its closest touchstone, World 4-2 (top) in the metrics of negative space, linearity, leniency, and jumps.



(e) World 6-3 from *Super Mario Bros. 2* (JP)

(f) Random seed 177 from Summerville and Mateas.
The room generated by Summerville and Mateas (bottom), further away from the original content than any other, and its closest touchstone, World 6-3 (top)in the metrics of negative space, linearity, leniency, and jumps. The generated room has large gaps and progresses from a relatively straight forward ground based room to a floating platform based room at the final third, as in World 6-3.

Figure 5: Three examples of choosing based on metric distance. (a) and (b) showcase picking based on distance to an *Exemplar* piece of content. (c) and (d) showcase the *Closest* pair of levels. (e) and (f) showcase the *Furthest* pair of levels found in the original and generated sets.

methodology for selecting which content to showcase, it also acts as a check on the metrics themselves.

The *Exemplar* methodology is quite simply:

1. Choose an exemplar piece of content – say a particularly important piece of content from the input corpus
2. Using the metrics used elsewhere in the analysis of the generator, find the generated content that is closest to that piece of content or furthest to that piece of content

The *Closest* methodology, to find the piece of content most like any of the original content:

1. For all pieces of original content find the generated content that is closest to that piece of content

Or the *Furthest* methodolgy to find the generated content furthest from the all of the pieces of original content:

1. For all pieces of generated content find the original content with which it is closest.
2. Find the generated content that is further from its closest content than any other piece of generated content

These three methods each answer different questions about the generative space of the generator.

- *Exemplar* — The most subjective of the methodologies, but it can help to showcase particular strengths and weaknesses of the generator by choosing exemplars that have certain unique properties (e.g. choosing a room in the original corpus that is singular and showing how the generator either did or did not learn to match the properties of that content.)
- *Closest* — This acts as both a check on plagiarism (although at a more global scale) and on the metrics themselves. If two pieces are found to be very similar in metric space but do not seem so in visual inspection, then, obviously, there is some aspect of the metrics that is poorly related to the actual qualities that are being attempted to be measured.
- *Furthest* — This acts as both a check on the generalization of the generator and as a sanity check on the generator. If the furthest piece of generated content still appears very similar to all of the original content, then the generator is not learning to generalize beyond the bounds of the original content. On the other hand, if the furthest piece of content is incomprehensible, then the generator is not doing a sufficient job of learning the latent design.

The distance is determined via the Euclidean distance of the whitened metrics, a process that centers all metrics at 0 with a variance of 0. Whitening is a common practice in machine learning techniques that rely on distance (e.g. k-means, k nearest neighbors, etc.), which makes the dimensional scaling less of an issue. Figure 5 shows the three techniques applied.

Conclusion and Future Work

This paper has presented techniques that will hopefully move the field of PCG research forward. While Expressive Range has been a crucial tool for PCG researchers, it has flaws that make it unsuitable in certain cases. This work presents extensions to Expressive Range that expand its uses, hopefully becoming a useful tool for the field of PCG research. Furthermore, while visual assessment is important, a technique for the quantitative assessment of generators is presented which is particularly useful for machine learned generation systems. Also presented are a set of techniques for the selection of content for showcasing – as in a research paper. These techniques show a number of different fail-cases for the generators – memorization and copying (plagiarized content) and mode collapse and repetition (self-plagiarized content) – in addition to showing how well a generator has learned the design (proximity to examples and proximity to closest content) as well as learned to generalize (proximity to furthest content).

This work is not the end-point for assessing PCG systems, just Expressive Range analysis was not the end-point. It is intended as another part of the conversation, hopefully seeing adoption by the community and allowing for the better

understanding of generated content. Future work will depend on the assessed gaps in the coverage of these techniques.

References

Metanet Software. 2015. N++.

Anscombe, F. 1973. Graphs in statistical analysis. *The American Statistician* 27(1):17–21.

Barratt, S., and Sharma, R. 2018. A note on the inception score. *arXiv preprint arXiv:1801.01973*.

Cook, M.; Gow, J.; and Colton, S. 2016. Danesh: Helping bridge the gap between procedural generators and their output.

Dahlskog, S.; Togelius, J.; and Nelson, M. J. 2014. Linear levels through n-grams. In *Proceedings of the 18th International Academic MindTrek Conference*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Foreman-Mackey, D. 2016. corner.py: Scatterplot matrices in python. *The Journal of Open Source Software* 24.

Guzdial, M., and Riedl, M. O. 2015. Toward game level generation from gameplay videos. In *Proceedings of the PCG Workshop*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Klambauer, G.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a nash equilibrium. *arXiv preprint arXiv:1706.08500*.

Hoover, A. K.; Togelius, J.; and Yannakis, G. N. 2015. Composing video game levels with music metaphors through functional scaffolding. *Comp. Creativity & Games at ICCC*.

Mariño, J. R., and Lelis, L. H. 2016. A computational model based on symmetry for generating visually pleasing maps of platform games.

Rosenblatt, M. 1956. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics* 832–837.

Ryan, J.; Kaltman, E.; Fisher, A. M.; Owen-Milner, T.; Mateas, M.; and Wardrip-Fruin, N. Gamespace: An explorable visualization of the videogame medium.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2234–2242.

Shaker, N.; Nicolau, M.; Yannakakis, G. N.; Togelius, J.; and O'Neill, M. 2012. Evolving levels for super mario bros using grammatical evolution. In *2012 IEEE CIG*, 304–311.

Smith, A. M., and Mateas, M. 2011. Answer set programming for procedural content generation: A design space approach. *IEEE TCIAIG* 3(3).

Smith, G., and Whitehead, J. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 4. ACM.

Smith, G.; Whitehead, J.; Mateas, M.; Treanor, M.; March, J.; and Cha, M. 2011. Launchpad: A rhythm-based level generator for 2-d platformers. *IEEE TCIAIG* 3(1).

Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE TCIAIG* (99).

Smith, G. M. 2012. *Expressive design tools: Procedural content generation for game designers*. Ph.D. Dissertation, University of California, Santa Cruz.

Snodgrass, S., and Ontañón, S. 2014. A hierarchical approach to generating maps using markov chains. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Snodgrass, S., and Ontañón, S. 2016. Learning to generate video game maps using Markov models. *TCIAIG*.

Snodgrass, S., and Ontañón, S. 2015. A hierarchical MdMC approach to 2D video game map generation. *AIIDE*.

Snodgrass, S., and Ontañón, S. 2016. Controllable procedural content generation via constrained multi-dimensional Markov chain sampling. In *25th International Joint Conference on Artificial Intelligence*.

Stephenson, M., and Renz, J. 2016. Procedural generation of levels for angry birds style physics games.

Summerville, A., and Mateas, M. 2016. Super Mario as a string: Platformer level generation via LSTMs. *DiGRA/FDG*.

Summerville, A.; Guzdial, M.; Mateas, M.; and Riedl, M. 2016. Learning player tailored content from observation: Platformer level generation from video traces using LSTMs. In *AIIDE*.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A. K.; Isaksen, A.; Nealen, A.; and Togelius, J. 2017. Procedural content generation via machine learning (pcgml). *ToG*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A.; et al. Going deeper with convolutions.

Székely, G. J., and Rizzo, M. L. 2013. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference* 143(8):1249–1272.

Teng, E., and Bidarra, R. 2017. A semantic approach to patch-based procedural generation of urban road networks. In *Proceedings of the 12th FDG*, 71. ACM.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2010. Search-Based Procedural Content Generation. In *Applications of Evolutionary Computation*. Springer. 141–150.

Worth, D. 1986. Beneath Apple Manor.