Automatically Categorizing Procedurally Generated Content for Collecting Games

In: *Proceedings of the Workshop on Procedural Content Generation in Games (PCG) at the 9th International Conference on the Foundations of Digital Games (FDG-2014).* New York, NY: ACM

Sebastian Risi
Center for Computer Games Research
IT University of Copenhagen
sebr@itu.dk

Joel Lehman
Department of Computer Sciences
The University of Texas at Austin
joel@cs.utexas.edu

David B. D'Ambrosio
ddambro@gmail.com

Kenneth O. Stanley
Department of EECS
University of Central Florida
kstanley@eecs.ucf.edu

ABSTRACT

A potentially promising application for procedural content generation (PCG) is *collecting games*, i.e. games in which the player strives to collect as many classes of possible artifacts as possible from a diverse set. However, the challenge for PCG in collecting games is that procedurally generated content on its own does not fall into a predefined set of classes, leaving no concrete quantifiable measure of progress for players to follow. The main idea in this paper is to remedy this shortcoming by feeding a sample of such content into a self-organizing map (SOM) that then in effect generates as many categories as there are nodes in the SOM. Once thereby organized, any new content discovered by a player can be categorized simply by identifying the node most activate after its presentation. This approach is tested in this paper in the Petalz video game, where 80 categories for user-bred flowers are generated by a SOM, allowing players to track their progress in discovering all the "species" that are now explicitly identified. The hope is that this idea will inspire more researchers in PCG to investigate applications to collecting games in the future.

1. INTRODUCTION

In recent years, games in which players accumulate a collection of diverse artifacts have proven popular. Examples of such *collecting games* include Pocket Frogs[1], where players try to collect a diversity of frog breeds with different names, and the Pokémon series of games[2], where players collect diverse species of characters. The appeal of such games is exemplified by the over 276,000 ratings that Pocket Frogs has attracted in the iTunes store, averaging to over four stars overall. The insight that players enjoy collecting artifacts suggests a potentially exciting opportunity for procedural content generation (PCG) [19], which offers the possibility of automatically generating the diversity of artifacts that players collect. Not only could these artifacts be virtually unlimited, but they could also in effect be optimized to appeal to players based on the observation of past player behavior in the game, as was demonstrated possible e.g. in Galactic Arms Race [5].

However, a significant obstacle to realizing this vision is that procedurally generated content does not fall into inherent categories that could facilitate crediting players with increasingly comprehensive collections. That is, to motivate players to continue playing such a game, some measure of *progress* is necessary to show that their collecting is successful. For example, in Pocket Frogs the aim is to collect all

Figure 1: Petalz Balcony View. Players in Petalz can breed their own unique flowers and share them with other players through a marketplace. This picture shows an example balcony that a player has decorated with various available flower pots and player-bred flowers. Petalz is designed as a casual and social Facebook game that is accessible to a large demographic.

[1]Pocket Frogs is produced by NimbleBit, LLC.
[2]Both Game Freak and Creatures Inc. have released games in the Pokémon series.

the different breeds of frogs. Therefore, to capitalize on the potential for PCG to contribute to collecting games, such content ideally would be categorized automatically by the game itself.

This paper presents a first such attempt to categorize procedurally generated content for the purpose of encouraging and rewarding players for collecting it. The proposed approach is to feed a sample of such content into a self-organizing map (SOM) [8] that subsequently separates it into as many nodes as are in the map. Once such a map is thereby organized, any new content can be classified by identifying the node in the map that responds most intensely to its presentation. In this way, each node in the map becomes a category in the classification scheme, and players (who do not need to know that the underlying technology is a SOM) can track how many such categories they have collected, providing a sense of progress.

This idea is demonstrated in this paper in the Petalz video game [10] (Figure 1; available on Facebook and through http://finchbeak.com/index.php/petalz-game/), where players breed novel flowers that they exhibit on their own personal balcony. By feeding a sample of such flowers into a SOM of predefined size, a set of categories is generated that helps to classify future flowers even though in principle the number of possible flowers is unlimited. In this way, the SOM turns an open-ended space of content into a concrete set of categories for collection, transforming Petalz into a bona fide collecting game. Other PCG-driven collecting games in the future could also potentially benefit from this SOM-based approach.

2. BACKGROUND

This section discusses games based on collecting, as well as work on PCG in games, and the PCG algorithm in the Petalz game.

2.1 Collecting Games

Video games nearly always feature some form of collection, from the coins and stars collected in Super Mario Brothers[3] to the weapons and achievements collected in modern first person shooters. However, a popular recent trend is to bring collection to the forefront and make it the focus of the game. Pocket Frogs, in which the player's goal is to collect and breed as many different types of frogs as possible, is one such game. Pocket Frogs has been successful in part because it taps into the natural human desire to find new items and set attainable goals [9]. Other popular games heavily integrate the idea of collecting, such as Rage of Bahamut[4] and other virtual collectible card games.

Such games also incentivize social interaction among their players, which can help them complete their respective collections. For example, games in the Pokémon series restrict the distribution of creatures such that some creatures are only found in certain versions of the game. Thus players must trade with others owning other Pokémon versions if they want to "catch 'em all," as the game's slogan suggests. This approach has been extended to online social games such as Outernauts[5], which also presents players with a large array of monsters to collect, trade, and battle.

[3]Copyright Nintendo, http://www.nintendo.com
[4]Copyright Mobage
[5]Copyright Insomniac Games

However, a major cost in designing games with a central goal of collection is that the artifacts to be collected are often individually designed by humans to be visually appealing, distinguishable, or otherwise separate and unique. There must also be a large number of such artifacts, or players will too quickly exhaust the game's content. This paper presents an alternative approach, wherein the items to be collected are created and categorized by the game itself, without direct human design.

2.2 Procedural Content Generation

PCG algorithms allow parts of the game (e.g. maps, textures, items, quests, etc.) to be generated algorithmically rather than by direct human design [6, 19]. This approach can reduce design costs and also benefit players by presenting them with unique experiences each time they play. For example, the popular Diablo series[6] generates dungeons with PCG algorithms that players explore. Many PCG approaches rely on a fixed set of parameters or random number generators, but recently focus has shifted towards applying artificial intelligence approaches to enhancing PCG.

Of particular note are evolutionary computation and other search-based approaches [19] that can further save on development costs and may be able to produce unique content beyond what human designers can create. One popular technique is interactive evolutionary computation (IEC [17]), in which the user guides evolution. One such example is NeuroEvolving Robotic Operatives (NERO [16]), in which players guide the evolution of a team of fighting robots. In another, Galatic Arms Race (GAR [4, 5]), weapons are evolved automatically based on user behavior. Other examples include Avery et al. [1], who evolved several aspects of a tower defense game, Shaker et al. [13] who evolved levels for the platform game Super Mario Bros, and Togelius and Schmidhuber [18], who experimented with evolving the rules of the game itself.

2.3 The Petalz Video Game

The game discussed in this paper, Petalz [10], creates virtual flowers through a PCG algorithm. The central game mechanic for players is to maintain and breed a collection of unique flowers. All players possess a *balcony* (Figure 1), which they can decorate with various items and flower pots that are purchased in the game's market. This market is also where flowers bred by other players can be bought and sold. Players can also interact by visiting each other's balconies and watering or liking the flowers there.

New flowers are discovered through an IEC process in which players breed offspring by selecting a single flower for mutation (i.e. slight variation), or by mating two flowers through cross-pollination. Figure 2 shows an example of the results of such mating.

2.3.1 Generating Flower Images and Shapes

This section provides context by explaining the encoding behind the flowers in Petalz. However, it is important to note that the idea for classifying procedurally generated content through a SOM in this paper in principle is independent of the specific encoding.

Flowers in Petalz are encoded by a special kind of compositional pattern producing network (CPPN [14]). CPPNs are a variation of artificial neural networks (ANNs) that

[6]Copyright Blizzard Entertainment, http://blizzard.com/

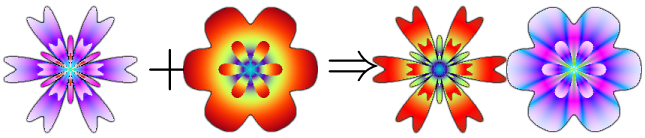

Figure 2: Flower Cross-pollination. By mating two flowers, children that exhibit the traits of both parents can be created. This approach gives players a powerful and intuitive means to explore the space of flowers.

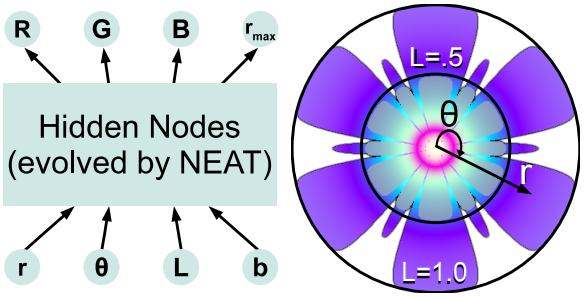

Figure 3: CPPN Flower Encoding. The CPPN that encodes flowers in Petalz takes polar coordinates (r and θ) as well as layer (L) and bias (b) values. The outputs are an RGB color value for that coordinate. The value r_{max} is also output, but only checked when $r = 0$ to determine the maximum radius for a given θ. The number and topology of hidden nodes is evolved by a standard CPPN-NEAT implementation [14].

differ in their set of activation functions and how they are applied [14]. While ANNs often contain only sigmoid or Gaussian activation functions, CPPNs can include both such functions and many others. The choice of CPPN functions can be biased toward specific patterns or regularities. Additionally, unlike typical ANNs, CPPNs are usually applied across a broad space of possible inputs so that they can represent a complete image or pattern. Because they are compositions of functions, CPPNs in effect encode patterns at infinite resolution and can be sampled at whatever resolution is desired. Other successful CPPN-based applications include Picbreeder [12], MaestroGenesis [7], EndlessForms [3], the Galactic Arms Race (GAR) video game [4], folded wire robots [11], and virtual soft-body robots [2].

The general idea behind the flower encoding in Petalz is to deform a circle such that the resulting shape resembles a flower. Because this approach focuses on a radial pattern, polar coordinates $\{\theta, r\}$ are input into the CPPN (Figure 3). For each value of θ, the deformed radius of the circle at that point (r_{max}) is queried by inputting $\{\theta, 0\}$ into the CPPN. Next, to fill in the colors of the flower's surface, each polar coordinate between 0 and r_{max} is queried with the same CPPN for a RGB color value. This approach produces a deformed, colored circle, but still may not produce flower-like images. Most natural flowers demonstrate basic radial symmetry in the form of their petals; this property is exploited in the encoding by inputting $sin(P\theta)$ into the CPPN *instead* of the raw θ value, which creates a repeating pattern of deformation and coloration. The optional P parameter allows control over the period of the sine function and thus the maximum number of repetitions around the circle.

Finally, to further improve the aesthetic of the flowers, the concept of *layers* is implemented to reflect that flowers generally have internal and external portions. To encode this property, a new flower is queried through the same CPPN for each layer L. Each such layer is scaled based on its depth and drawn on top of the previous layer. Thus the inputs to the CPPN are $\{\theta, 0, L\}$ and the outputs are $\{R, G, B, r_{max}\}$ to determine the shape of each layer (Figure 3). The internal coloring is then determined by querying the CPPN as with the outermost layer. Interestingly, because the layers are queried by the same CPPN, they are mathematically related, giving a natural look. Further optimizations on this general algorithm are employed [10] to achieve the fast rendering necessary for an online game.

The flower-encoding CPPNs in Petalz are evolved with the NEAT algorithm [15]. NEAT begins with a population of simple neural networks or CPPNs and then *adds complexity* over generations by adding new nodes and connections through mutations. By evolving networks in this way, the topology of the network does not need to be known a priori; NEAT searches through increasingly complex networks to find a suitable level of complexity. For a complete overview of NEAT see Stanley and Miikkulainen [15]. Most importantly, such complexification, which resembles how genes are added over the course of natural evolution, allows NEAT to establish high-level features early in evolution and then later elaborate on them. For evolving content, complexification means that content (e.g. flowers in the case of Petalz) can become more elaborate and intricate over generations.

3. AUGMENTING PCG GAMES WITH COLLECTION MECHANICS

While collection game mechanics have worked well in engaging players in a variety of different games like Pocket Frogs or the Pokémon series, how PCG-based games can be augmented which such mechanics is an open research question. Although PCG approaches can benefit players by presenting them with unique experiences and content each time they play, in PCG games like Petalz [10] or GAR [4], there are no a priori categories of flowers or weapons for the player to collect.

Furthermore, though some players thrive in such free-form games without explicit goals, many other players enjoy goal-directed game mechanics where concrete purpose is directly provided (e.g. collect all 80 flower species, collect ten flowers of a specific species). Therefore a promising idea is to automatically organize the procedurally generated content into specific categories, thereby providing a means to augment *any* PCG game with collection mechanics.

To automatically organize procedurally generated content requires a classifier that can automatically determine the category of newly generated content based on a training set of data. Importantly, this classification has to be meaningful to the player. For example, if the algorithm assigned the same class to two very differently looking flowers, the collection mechanic would not add meaning and structure to the player experience.

In this paper the procedurally generated content is classified based on the self-organizing map (SOM) algorithm [8]. While many other clustering algorithms could be applied, the SOM algorithm offers some unique advantages that give

it particular appeal in a game context, which will be explained in the next section.

3.1 Self-Organizing Map

The SOM [8] is an artificial neural network that performs an unsupervised mapping from a high-dimensional input space R^D with input input patterns $\mathbf{X} = \{x_i : i = 1, ..., D\}$ onto a two-dimensional grid of neurons. Each neuron j in the computation layer is connected to each input neuron i with weight vector $\mathbf{w_j} = \{w_{ij} : i = 1, ..., D\}$.

The basic training process works by iteratively presenting an input vector $\mathbf{x}$ to the SOM and then comparing the weight vectors of the neurons on the grid to that input vector. The neuron c with the most similar weight vector is called the *best matching unit* (BMU) and is defined by the condition:

$$||x(t) - w_c(t)|| \leq ||x(t) - w_i(t)|| \forall i. \quad (1)$$

Once the BMU is determined, its weights and the weights of all the neurons within a fixed distance to the BMU are updated:

$$w_i(t+1) = w_i(t) + h_{c(x),i}(x(t) - w_i(t)), \quad (2)$$

where t is the index of the training step and h is called the neighborhood function, which is commonly a Gaussian function:

$$h_{c(x),i} = \alpha(t) e^{-\frac{||r_i - r_c||^2}{2\sigma^2(t)}}, \quad (3)$$

where $0 < \alpha(t) < 1$ is the learning-rate, which decreases monotonically with the training step t. In addition, r_i and r_c are the locations on the SOM map and $\sigma(t)$ corresponds to the width of the neighborhood function. The width of the neighborhood function also decreases monotonically with the training steps.

The SOM can help with classification [8, 20] by assigning a class to each neuron and determining the class of a new sample based on the class of its BMU. In other words, by feeding a sample of procedurally generated content into a SOM of predefined size, a set of categories is generated that helps to classify future content even though in principle the number of possible content is unlimited. Figure 4 gives an overview of this approach to categorizing procedurally generated content.

Two properties of the SOM give it particular appeal in categorizing procedurally generated content for the purpose of encouraging players to collect it. First, after training, the SOM forms a *topographic map* of the input space, in which content that is visually similar is also nearby in space. In this way the SOM can provide an intuitive visualization of the space of content for the player. Second, the prototype weight vectors of the neurons on the grid can be visualized to give the player an idea of categories not yet discovered.

The next section describes how the SOM algorithm is applied to automatically categorize flowers into different species in the Petalz video game.

4. EXPERIMENT

In this paper the automatic categorization of procedurally generated content is tested in the Petalz video game [10]. For this purpose the game is augmented with a SOM-based collection game mechanic that allows the player to track their progress in discovering all pre-classified flower categories.

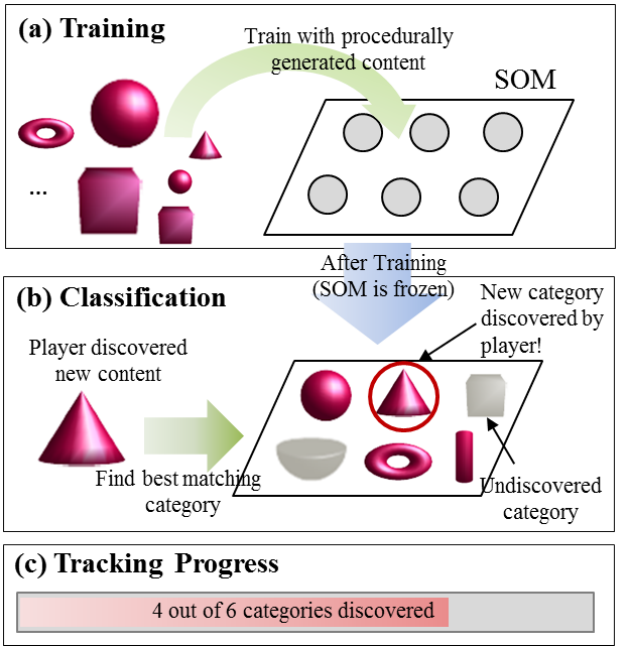

Figure 4: PCG Collection Game Mechanic. The SOM is first trained with samples of procedurally generated content (a). Once the SOM is fully trained, new content discovered by the player can automatically be categorized (b). Tracking the collection progress (c) can then add meaning and structure to the player experience.

To classify different flowers with a SOM the question of how to best represent these flowers for training becomes important. Two different methods for presenting flowers to the SOM are tested to determine which produces the most meaningful flower categories. The SOMs for all approaches have the same size of 5×16 neurons, resulting in 80 different categories for user-bred flowers. Each SOM is trained with a selection of 2,000 diverse flowers that were evolved by players during the game.

In the **phenotypic clustering** approach the images of the training flower are scaled from 200 × 200 pixels to a 50 × 50 pixel black-and-white version. Because the procedurally generated flowers in Petalz are all symmetric along the x and y axis it is only necessary to feed the top-left square (25×25 pixels) into the SOM for training and also classification during the game. Therefore each flower is described by a feature vector of 625 integers.

In the **genotypic clustering** approach each flower is described by its pair-wise genotypic distance to the 2,000 other flowers in the training set. Following Stanley and Miikkulainen [15] the distance δ between two CPPN encodings can be measured as a linear combination of the number of excess (E) and disjoint (D) genes, as well as the average weight differences of matching genes ($\overline{W}$): $\delta = E + D + \overline{W}$.

5. RESULTS

Figure 5 shows examples of the phenotypic clustering by the SOM. The flowers belonging to the same species show a clear resemblance to one another, suggesting that the SOM can produce a meaningful clustering of the procedurally generated flowers.

Figure 5: Phenotypic Flower Clustering. The picture depicts the 80 distinct species in Petalz. Examples of flowers clustered into three different species are depicted to the right. The main result is that the phenotypic clustering of flowers is meaningful and thus demonstrates the feasibility of augmenting PCG games with a collection game mechanic. Note that the SOM is toroidal, which means that the flower prototypes to the far right of the flower chart (left panel) are also bordering the prototype vectors to the far left. The same holds true for the flowers at the bottom and the top of the flower chart.

Every time the player discovers a new species, a flower chart is displayed (Figure 5, left) and the newly bred flower is highlighted. In addition to the discovered flowers, the prototype vectors of the remaining flower species are also shown. That way, the flower chart gives the player direct feedback about his progress in the game and about flower categories that are still undiscovered. Note that the 80 displayed flower categories reflect the topology of the SOM (16×5 neurons).

Additionally, because the SOM produces a topographic map of the training data, flowers that are phenotypically more similar than other flowers are also grouped closer together in space. As Figure 5 (left) shows, the result is a visually appealing smooth gradient of different flower prototype weights.

The genotypic clustering (based on the genotypic distances between the flowers) is shown in Figure 6. While some classifications are satisfying (e.g. *Species 1*), some flowers that do not show a clear phenotypic resemblance also group together (e.g. *Species 25*, *Species 69*). These inconsistencies in the clustering happen because small changes in the flower's genotype can sometimes produce more significant changes in the decoded phenotypes.

The main result is that procedurally generated content can automatically be clustered into different categories (like flower species), thereby adding more meaning and structure to player experience. Additionally, this paper showed that – at least in Petalz – a clustering based on phenotypic characteristics produces more intuitive classifications than one solely based on genotypic distances.

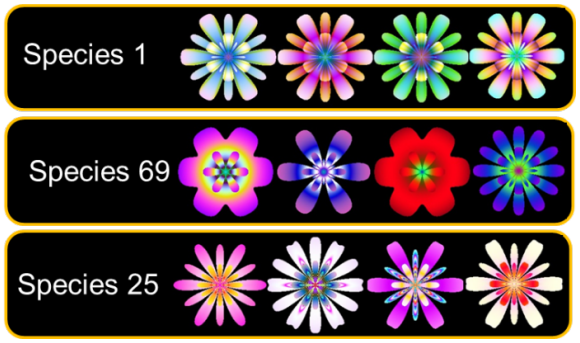

Figure 6: Genotypic Flower Clustering. This figure shows a SOM clustering of the flowers based on their pairwise genotypic distances. While such genotypic clustering sometimes produces meaningful clusters (e.g. *Species 1*), often flowers are grouped together that look significantly different (e.g. *Species 25*, *Species 69*).

6. DISCUSSION AND FUTURE WORK

The results represent a proof of concept for a general method to augment PCG games with a collection game mechanic. Importantly, such a game mechanic can add meaning and structure to a game player's experience, which otherwise might be missing when PCG is the game's central focus. That is, while user exploration of PCG can provide the potential for unlimited novelty in game content, it still may not inherently provide explicit *purpose* to motivate a user's search. For example, without the objective of collecting a flower from all 80 SOM-determined species in Petalz, a user has little purpose beyond indulging their own aesthetic preferences when breeding flowers. Although the success of open world games demonstrates that not all players require explicit goals, many players do enjoy the concrete purpose they provide. Thus the clear benefit for extending PCG games with a collection game mechanic is the potential for broader appeal, by providing enjoyable opportunities both for self-directed and goal-directed players.

An interesting side-effect of the collection game dynamic is that it grants inherent value to evolved artifacts in species that are hard to discover. Players may not only feel a sense of accomplishment in discovering a new species, but may assign greater personal value to such flowers themselves because they were hard to find. If players feel proud or connected to the content they create, an interesting possibility is that they may desire for such evolved content to be visible outside the context of a computer game. For example, three-dimensional printouts of a player's collected content

Figure 7: Printed Artificially-Evolved Flower. This example is a flower that was evolved by users in Petalz, manually converted into a three-dimensional representation, and then printed in three dimensions.

can potentially bring evolved game content to the real world (figure 7 shows an example printed from Petalz).

Finally, automatically clustering procedurally generated content may open up applications beyond the singular collection game mechanic presented in this paper. For example, many video games segment levels into higher-level groups that are often called worlds.

Typically, the levels in each world share a common theme, thereby providing a coherent experience for users as they progress through the game. For instance, in Super Mario Brothers 3[7], world four ("Big Island") features levels with oversize enemies and terrain elements, while levels in world seven ("Pipe Maze") are largely composed of networks of pipes. However, a game based on procedurally-generated levels may lack this sense of coherence unless such levels are organized in a noticeably structured way. Clustering procedurally-generated levels may provide a means for such organization, and may facilitate automatically generating worlds with a coherent theme. More broadly, clustering content provides a mechanism for automatic categorization of evolved artifacts, which can potentially be exploited in other ways beyond theming or collection.

An interesting direction for future work is to explore the generality of the automatic collection game mechanic by applying it to other games centered on PCG like Galactic Arms Race video game [5].

7. CONCLUSION

This paper proposed an approach to incorporating PCG effectively into collecting games. While procedurally generated content on its own does not automatically fit into predefined categories, the proposed approach is to feed the content into a SOM that is trained to classify it into as many categories as there are nodes in the SOM. That way, even if the space of content is effectively unlimited, players can still experience a sense of quantifiable progress as they discover an increasing proportion of the SOM-based category space. This approach was tested in the Petalz video game, where a SOM computed 80 categories (called "species" in Petalz) for players to discover. The main conclusion is that PCG can be adapted to support collecting games when complemented with an effective method for content categorization.

[7]Copyright Nintendo, http://www.nintendo.com

References

[1] P. Avery, J. Togelius, E. Alistar, and R. van Leeuwen. Computational intelligence and tower defence games. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1084–1091. IEEE, 2011.

[2] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*, New York, NY, 2013. ACM Press.

[3] J. Clune and H. Lipson. Evolving 3d objects with a generative encoding inspired by developmental biology. In *Proceedings of the European Conference on Artificial Life (Alife-2011)*, volume 5, pages 2–12, New York, NY, USA, Nov. 2011. ACM.

[4] E. Hastings, R. Guha, and K. O. Stanley. Evolving content in the galactic arms race video game. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG-09)*, Piscataway, NJ, 2009. IEEE Press.

[5] E. J. Hastings, R. K. Guha, and K. O. Stanley. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245–263, 2009.

[6] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(1):1, 2013.

[7] A. K. Hoover, P. A. Szerlip, and K. O. Stanley. Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. In M. L. Maher, K. Hammond, A. Pease, R. P. Y. Perez, D. Ventura, and G. Wiggins, editors, *Proceedings of the 3rd International Conference on Computational Creativity (ICCC-2012)*, 2012.

[8] T. Kohonen. *Self-organizing maps*, volume 30. Springer, 2001.

[9] W. D. McIntosh and B. Schmeichel. Collectors and collecting: a social psychological perspective. *Leisure Sciences*, 26(1):85–97, 2004.

[10] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley. Combining search-based procedural content generation and social gaming in the petalz video game. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012)*, Menlo Park, CA, 2012. AAAI Press.

[11] S. Risi, D. Cellucci, and H. Lipson. Ribosomal robots: Evolved designs inspired by protein folding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*, New York, NY, 2013. ACM.

[12] J. Secretan, N. Beato, D. D'Ambrosio, A. Rodriguez, A. Campbell, J. Folsom-Kovarik, and K. Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):373–403, 2011.

[13] N. Shaker, G. N. Yannakakis, J. Togelius, M. Nicolau, and M. O'Neill. Evolving personalized content for super mario bros using grammatical evolution. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012)*, Menlo Park, CA, 2012. AAAI Press.

[14] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[15] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.

[16] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, 9(6):653–668, December 2005.

[17] H. Takagi. Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[18] J. Togelius and J. Schmidhuber. An experiment in automatic game design. In *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, pages 111–118. IEEE, 2008.

[19] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186, 2011.

[20] A. Ultsch. Self-organizing neural networks for visualisation and classification. In *Information and classification*, pages 307–313. Springer, 1993.