This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

Petalz: Search-based Procedural Content Generation for the Casual Gamer

Sebastian Risi, Joel Lehman, David B. D'Ambrosio, Ryan Hall, and Kenneth O. Stanley

***Abstract*—The impact of game content on the player experience is potentially more critical in casual games than in competitive games because of the diminished role of strategic or tactical diversions. Interestingly, until now *procedural content generation* (PCG) has nevertheless been investigated almost exclusively in the context of competitive, skills-based gaming. This paper therefore opens a new direction for PCG by placing it at the center of an entirely casual flower-breeding game platform called *Petalz*. That way, the behavior of players and their reactions to different game mechanics in a casual environment driven by PCG can be investigated. In particular, players in Petalz can (1) trade their discoveries in a global marketplace, (2) respond to an incentive system that awards diversity, and (3) generate real-world three-dimensional replicas of their evolved flowers. With over 1,900 registered online users and 38,646 unique evolved flowers, Petalz showcases the potential for PCG to enable these kinds of casual game mechanics, thus paving the way for continued innovation with PCG in casual gaming.**

***Index Terms*—Procedural Content Generation, Collection Mechanics, 3D printing, CPPNs.**

I. Introduction

While the potential of search-based procedural content generation (search-based PCG; [27, 28]) in competitive gaming is now well-established [7, 15, 20], its benefits are less examined within the context of social and casual games. Yet casual gaming, with its focus often on activities like collecting[1] or decorating[2], is among the most promising venues for realizing the potential of procedurally generated content. After all, these kinds of casual activities are fueled by *content* that therefore must continually be refreshed and updated. A casual game that could continue to generate new content in concert with the actions of the players would open up new avenues for entertainment. Moreover, such a system might even produce *value* by yielding products that players genuinely desire to collect and trade.

This paper provides a comprehensive overview of the Petalz video game, which is designed to demonstrate that such generated content can indeed integrate effectively into a modern casual Facebook game. It unifies and expands specific results reported previously in conference proceedings [16, 18] with newer experiments on the conversion of generated content to real three-dimensional printed artifacts. The main contribution is thus the first investigation of a complete PCG-based casual game that has been deployed on Facebook and tested by hundreds of players.

The main game mechanic in Petalz is flower breeding. Each player is given a balcony (which can later be expanded to multiple balconies) where the player can arrange pots in which they breed unique custom flowers through a specialized genetic encoding designed for evolving flower-like forms. The uniqueness of each flower bred in Petalz facilitates a bond between the player and his or her flowers, which are thereby genuine reflections of each player's effort and personality. Social interactions like visiting the balconies of friends and watering their flowers further help to build community and offer opportunities for recognition.

The key issue investigated is how the generated content can naturally enhance the game experience. The opportunities explored for such benefits include (1) a marketplace for buying and selling evolved content, (2) a content collection mechanic that recognizes and rewards players for acquiring diversity, and (3) the ability to generate printable models of the evolved content. All three of these mechanics become possible for the very reason that the flowers are unique generated artifacts, and each is accordingly demonstrated in depth in this paper. In this way the hope is to inspire continued research and innovation in the application of techniques in PCG to casual gaming.

The paper begins in the next section by reviewing related prior work and the particular search-based technologies behind Petalz. Section III then details the game's implementation and mechanics. Next, Section IV explains how the player-bred flowers in Petalz become part of a global marketplace. The generated content is further leveraged in Section V to turn Petalz into a collecting game, and additional value for the generated content is demonstrated in Section VI through 3D printing. The paper finally concludes with discussion and future work (Section VII), as well as a conclusion (Section VIII).

II. Background

This section first discusses social and casual games, as well as existing work combining PCG with traditional video games. The section concludes by reviewing the technical building blocks of the PCG algorithm employed in the Petalz game.

A. Social and Casual Games

Social and casual games represent a rapidly expanding market in the video game industry [2]. Typically, these games distinguish themselves from traditional video games by (1) not requiring high levels of skill from their players and (2) facilitating and rewarding social interactions among their players [11]. Combining these two factors results in games

SR is with the IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen, Denmark. JL is with the University of Texas at Austin. KOS is with University of Central Florida, 4000 Central Florida Blvd. Orlando, Florida 32816. Emails: sebr@itu.dk, joel@cs.utexas.edu, ryanhallcs@gmail.com, ddambro@gmail.com, kstanley@eecs.ucf.edu

[1]Pokémon © Nintendo, Pocket Frogs © NimbleBit, Dragon Collection © Konami

[2]FarmVille © Zynga, The Sims © EA, Animal Crossing © Nintendo

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

with a much wider audience than "traditional" video games, in part because feedback loops created by both game content and social pressure keep players engaged.

Some of these games, such as Candy Crush Saga[3], rely on simple rules to build increasingly complex and competitive challenges. Other games, like Pocket Frogs[4] task the player with finding and collecting a diversity of artifacts (e.g. diverse frogs in Pocket Frogs). These types of games have been successful in part because they tap into the human desire to find novelty and set attainable goals [14]. By exploiting our curious and competitive nature, as well as many other human factors [10], many social and casual games exhibit the powerful ability to keep players coming back again and again.

Many of these games are free for users to play, and are monetized by advertising or in-game purchases. Thus there is a disincentive for game developers to provide a definitive ending point that might break player engagement. As a result, a challenge in these games is that keeping users engaged requires a stream of new content (e.g. puzzles, levels, items to collect). Problematically, this new content can be expensive to produce because it requires continual creativity to design content sufficiently outside the players' current in-game experience; otherwise if content is only superficially novel existing players may get bored and stop playing. In contrast to more traditional video games, social and casual games are thus often continually in development, requiring the expense of artists, programmers, and designers to continually generate new content. To ameliorate this added expense, this paper presents an alternative approach wherein this new content can instead be created and categorized algorithmically through PCG, which is reviewed next.

B. Procedural Content Generation

When applied to games, PCG allows game elements (e.g. maps, textures, items, quests, etc.) to be generated algorithmically rather than through direct human design [8, 27]. This approach can reduce design costs and can also benefit players by providing them unique experiences every time they play. For example, the popular Diablo series[5] features procedurally generated dungeons that players explore as a central focus of the game. Like Diablo, many other PCG approaches similarly rely on a fixed set of parameters and randomness to generate content within a heavily constrained space of possibilities. However, a recent focus is to apply artificial intelligence approaches to enable more open-ended generation of PCG.

In particular, evolutionary computation and other search-based approaches [27] can limit the need for hand-designed rules, and may thus further save on PCG development costs. More interestingly, it also enables design of new content outside the scope of a fixed space of rules. One popular technique is interactive evolutionary computation (IEC [25]), in which the user in effect guides an evolutionary algorithm.

An example of IEC applied to video games is provided by NeuroEvolving Robotic Operatives (NERO [24]), in which players guide the evolution of a team of fighting robots. In another example, Galatic Arms Race (GAR [7]), weapons are evolved automatically based on user behavior. Further examples include Avery et al. [1], who evolved several aspects of a tower defense game, Shaker et al. [21] who evolved levels for the platform game Super Mario Bros, and Togelius and Schmidhuber [26], who experimented with evolving the rules of the game itself. While not strictly a video game, Electric Sheep [5] is an example of IEC applied to a crowd-sourced internet application. Electric Sheep enables users to evolve fractal animations (called sheep) in form of an interactive screen-saver. Electric Sheep shows how crowd-sourced and collaborative IEC can allow users to produce interesting visual digital artifacts, providing precedent for the approach in Petalz.

The next section reviews a particular evolutionary representation that has previously proven promising for generating content, and is thus applied in the Petalz game.

C. Compositional Pattern Producing Networks

The flowers in Petalz are generated by a variation of artificial neural networks (ANNs), called *compositional pattern producing networks* (CPPNs [22]), which differ in their set of activation functions and how they are applied. 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 queried across a space of possible input patterns to 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 [19], MaestroGenesis [9], EndlessForms [4], the Galactic Arms Race (GAR) video game [7], folded wire robots [17], and virtual soft-body robots [3]. Note that the modifications to the general CPPN representation to produce flower-like images are described in Section III-A.

D. Neuroevolution of Augmenting Topologies (NEAT)

The flower-encoding CPPNs in Petalz are evolved with the NEAT algorithm [23], which is the standard neuroevolution algorithm for such purposes [6, 19, 22]. 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 [23]. 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.

III. The Petalz Video Game

This section introduces the Petalz video game (available publicly at http://apps.facebook.com/petalzgame/), in which

[3]Copyright King
[4]Copyright NimbleBit
[5]Copyright Blizzard Entertainment, http://blizzard.com/

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

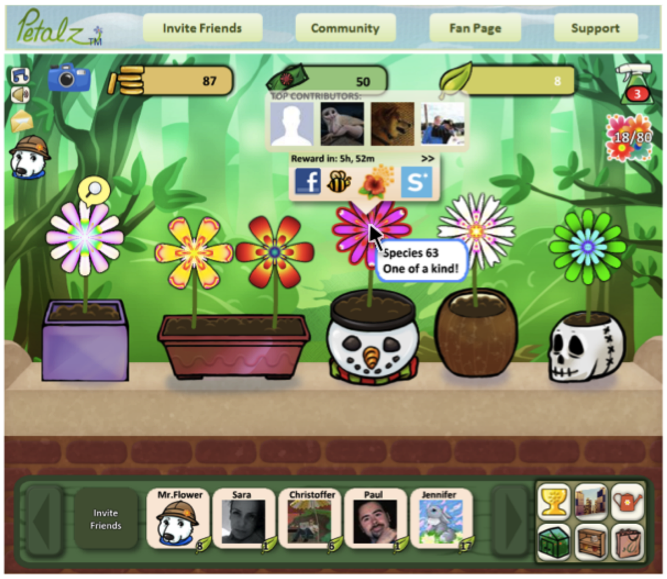


Figure 1: **Petalz Balcony View and Flower Interface.** This picture shows a user's balcony that was decorated with various available flower pots and player-bred flowers. Through a flower context menu players can breed flowers, print them in three dimensions, share them with their friends on Facebook and visit players that contributed to a flower's lineage. Petalz is available online at: http://apps.facebook.com/petalzgame/

players create 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.

Petalz is the first game to (1) combine social gaming and search-based PCG, (2) implement a market for user-evolved content, (3) extend a PCG-based game with collection mechanics and (4) allow 3D printing of in-game evolved content. The next section describes the PCG algorithm to generate the flowers in Petalz and the underlying flower encoding.

A. Generating Flower Images and Shapes

The purpose of an encoding is to define a *space* of possible content that can be explored. Although CPPNs have previously been applied to generating two-dimensional images [19] and three-dimensional structures [4] the same generic approach is not possible to apply when a *specific* class of image or structure is desired. Thus this section discusses the modifications made to the general CPPN representation to produce flower-like images. In short, the general idea behind the flower encoding in Petalz is to deform a circle such that the resulting shape resembles a flower.

The process begins by deforming the circle. Because this approaches focuses on a radial pattern, polar coordinates $\{\theta, r\}$ are input into the CPPN (Figure 2). 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

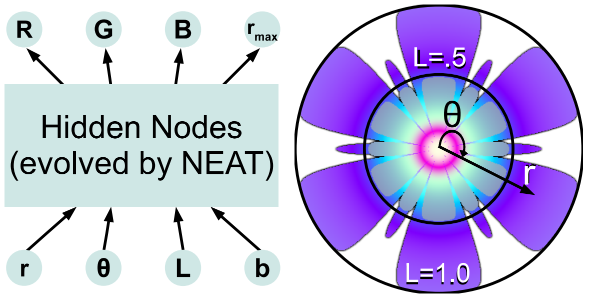


Figure 2: **CPPN Flower Encoding.** The CPPN that encodes flowers in Petalz takes polar coordinates (r and θ) as well as layer (L) and bias (b) values. The outputs are an RGB color value for that coordinate. The value r_{max} is also output, but only checked when $r = 0$ to determine the maximum radius for a given θ. The number and topology of hidden nodes is evolved by a standard CPPN-NEAT implementation [22].

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. Many natural flowers that humans find pleasing 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 2). 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.

It is important to note that the aim of the algorithm is not to produce flowers that necessarily look biologically realistic. While the common number of petals of real flowers is one of the smaller Fibonacci numbers (e.g. 2, 3, 5, etc.), the majority of Petalz flowers has a number of petals that is a multiple of six. This is due to the current setting of parameter P=6, which was chosen because it produced flowers that are most visually appealing at the current flower size. Also importantly, the produced flowers align with the cartoon theme of the overall artistic style and the chosen setting makes it possible to render only a quarter of the flower and copy it over to the other quadrants. Non-even values would break that optimization, which is necessary for the fast rendering required in a game context. The interested reader is referred to the following textbook by Mabberley [13], which gives a comprehensive overview of different plants and their classifications.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

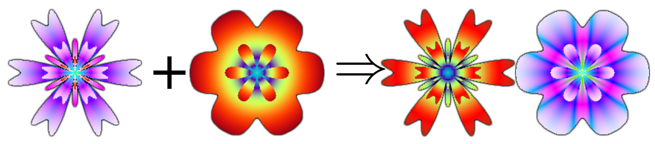


Figure 3: **Flower Cross-pollination.** By mating the CPPNs of two parents, children that exhibit the traits of both parents can be created. This approach gives players a powerful but intuitive way to explore the space of flowers.

B. Game Mechanics Overview

The primary interface through which players interact with flower evolution is a context menu that the player can open by hovering over a particular flower (Figure 1). The menu allows the player to (1) pollinate the flower, (2) cross-pollinate the flower with another flower, (3) post the flower to a friend's Facebook wall, (4) clone the flower, (5) inspect the flower's family tree, (6) sell the flower on the marketplace (described in the next section), (7) store the flower in the player's inventory, (8) or send the flower to Shapeways to print it in three-dimensions (described in Section VI). Actions like pollination (i.e. mutation of a single flower genome), cross-pollination (i.e. crossover between two flower genomes; Figure 3) and cloning produce up to five little *flower seedlings*, which are miniature version of what the adult flower would look like. These seeds can then be planted by the user after which they take up to several minutes to grow into their adult form. Additionally, because users in Petalz can collaborate to create and share their flower creations, the top four collaborators for a particular flower are also shown. These collaborators can be visited through the context menu, thus creating a social link between different users through collaboratively and procedurally evolved content.

Every flower that is not a clone is unique, which, as the results in this paper will show, allows players to continually find novel flowers with characteristics evolved from flowers they preferred in the past.

C. Development

Petalz was developed as an online social game that currently runs through Adobe Flash on the Facebook platform. The game was designed to be casual, featuring an intuitive and easily accessible UI. The development of the game (which is still ongoing) has been underway for over three years by a six-member team (which includes one artist). The Petalz client is composed of 71,241 source lines of actionscript code (SLOC), while the Petalz server is 3,972 SLOC of PHP.

The development of Petalz can be divided into three major milestones that will be explained in detail in the following sections. First, a **flower marketplace** (Section IV) was introduced that allows the players to sell and share their flowers with other Petalz players. Transactions in which one player buys seeds from another were designed to create a new social element that links players through an in-game economy of evolved content. Second, because the procedurally generated flowers on their own belong to no predefined set of classes, for collectors there is no inherent concrete quantifiable measure of progress. Thus

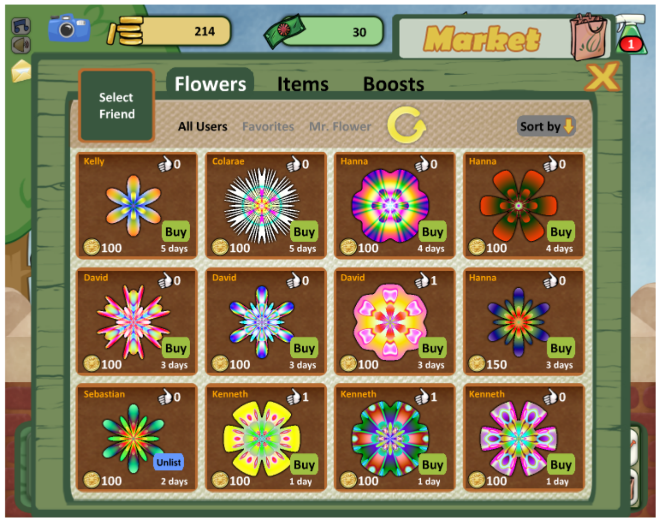


Figure 4: **Petalz Marketplace.** The marketplace enables players to purchase flowers from other players and to sell their own unique creations. Players can chose to view the flowers listed by all players or by only a specific player, and can sort flowers based on their price, listing date or number of likes.

collection mechanics (Section V) were added to incentivize collecting and thereby make the game more engaging. Finally, to further extend the impact of a PCG-oriented game, Petalz was augmented to allow the automated conversion of flowers into physical reality through integration with the Shapeways **3D printing** marketplace (Section VI).

IV. The Petalz Marketplace

The fact that each flower is a unique and lasting discovery opens up a number of intriguing possibilities. One is that players can sell their flowers in a global market to other players (Figure 4). The user can list a flower in the market either directly from her balcony or inventory. The listing price in the virtual "coin" currency, which can be earned by growing new flowers or performing other actions in the game, can be freely chosen by the seller in a range from 10 to 9,999 coins. Users have to pay 10% of the price as a fee to list their flower for a maximum duration of six days. The listing fee is intended to discourage spamming the market with low-quality flowers.

Because flowers are genuinely novel artifacts, their seeds have potential economic value that can allow skillful breeders to be rewarded for their efforts. Furthermore, the transaction in which one player buys seeds from another creates a new kind of social interaction that links the players in the transaction. This interaction goes beyond simply enjoying the product of another player: Once seeds are purchased, the buyer can now breed new flowers from the purchased seeds, generating a whole new lineage. The market in Petalz also has a section for pre-evolved *starter flowers*, from which the player can choose an aesthetically-pleasing flower to begin the game.

Beyond user-evolved flowers players can also purchase new backgrounds, decorations or flower pots, allowing them to customize their balcony to their unique taste. A variety of balconies from different players are shown in Figure 5.

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

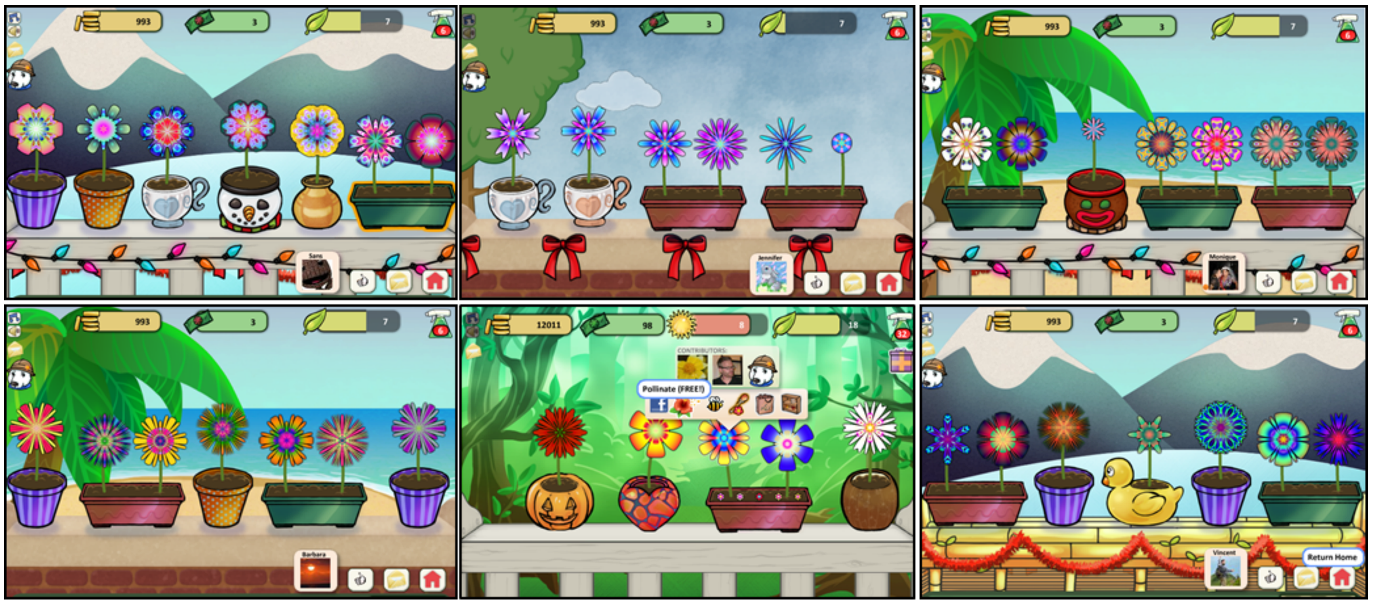


Figure 5: **Example Player Balconies.** The ability to breed unique flowers combined with the option to purchase a variety of backgrounds, flower pots and decorations, allows players to create unique balconies.

User Data	
Total Testers	22
Max # of Flowers Evolved	1,206
Avg # of Flowers Evolved	155.14
Avg # Flowers Listed	37.54
Max # Flowers Listed	157
Avg # Flowers Bought/Sold	10.95
Max # Flowers Sold	36
Max # Flowers Bought	71

Table 1. **User Data During Testing Period.** A summary of a snapshot of user data records taken during the four month testing period. Users tended to evolve a significant number of flowers and interact often with the market.

Flower Evolution Data	
Total Evolved Flowers	3,703
Total Liked Flowers	130
Max Generation	288
Avg Generation of Balcony Flowers	103.19
Avg Likes of All Evolved Flowers	0.04
Avg Likes of Balcony Flowers	0.5
Avg # of Users Contributing to Balcony Flower	7.27

Table 2. **Flower Evolution Data During Testing Period.** Users collaborated indirectly to evolve many flowers and tended to display those found to be more aesthetically pleasing on their balconies.

A. Market Analysis

To investigate the implications of a market place for evolved content, Petalz was initially tested by 22 players (including the five authors) over a period of four months. A variety of statistics intended to illuminate the impact of the evolved flower marketplace on the Petalz game were derived from a data snapshot retrieved at the end of this testing period. It is important to note that none of the extrema data, i.e. minima or maxima in the tables in this paper, result from the authors.

Table 1 summarizes general information about the behavior of the players during the testing period. It is evident that players together yielded both significant evolutionary and market activity: The average player evolved no fewer than 155.14 flowers and listed on average 40 on the market.

Table 2 provides further insight into the dynamics of evolution in the game. First, over three thousand flowers were evolved by the 22 players during the testing period suggesting that breeding is a significant attraction in the game. Furthermore, many flowers resulted from long evolutionary lineages for interactive evolution (103.19 generations on average separate flowers displayed on users' balconies from the flowers that seeded the game). In addition, data from "liked" flowers suggests that there is measurable variety in the aesthetic appeal of flowers and that users are more likely to display more appealing flowers on their balconies. Thus taken as a whole the data from Table 2 suggests that evolution within Petalz is driving force behind player behavior.

Given the significance of breeding within Petalz, an important question is what impact the evolved marketplace has on the game. To help investigate this question, each transaction in the Petalz marketplace was recorded during the testing period. This recorded data is summarized in Table 3. Some of these statistics establish that the marketplace is working in an intuitive manner. For example, cheaper flowers sold better than more expensive flowers, and not all listed flowers were purchased. Furthermore, the flowers that were bought tended to have more likes, indicating that flowers that appealed more aesthetically to users were also more highly *valued* by them. More significantly, the statistics also support that the marketplace is integral for both facilitating collaboration and evolving more desirable flowers. Importantly, at the time the data snapshot was taken, the majority of flowers displayed on users' balconies were close descendants of flowers bought from *other users* on the marketplace, although users can choose to ignore the marketplace completely or buy not from other users but from a permanent set of pre-evolved starter flowers that are always available on the marketplace.

Furthermore, marketplace flowers themselves were on average the product of many users collaborating, indicating that flowers bought from the marketplace were often further

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

Flower Market Data	
Total Market Listings	826
Total Market Sales	241
Avg Listing Price of Flowers	216.97
Avg Sales Price of Flowers	164.12
Avg Likes for Listed Flowers	0.36
Avg Likes for Sold Flowers	0.64
Avg # of Users Contributing to Market Flower	7.46
Proportion of Balcony Flowers Descended from Other Users' Flowers'	0.73
Avg Generations from Market to Balcony Flower	6.71
Total Currency Flow on Market	39,555

Table 3. **Flower Market Data During Testing Period.** Flowers that were cheaper or more aesthetically pleasing tended to sell better. Importantly, the market also facilitated collaboration between users.

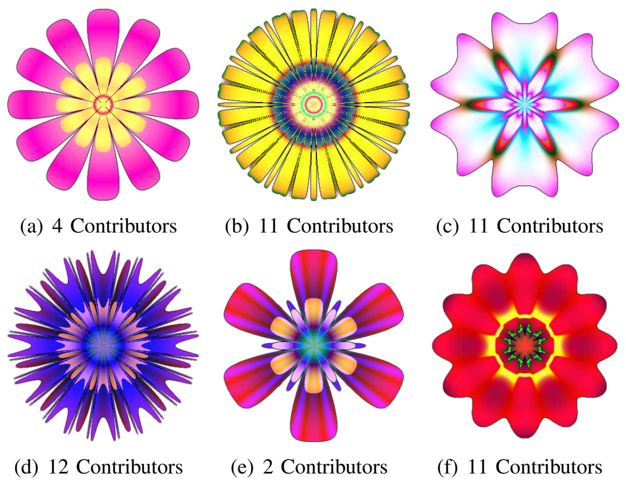

(a) 4 Contributors (b) 11 Contributors (c) 11 Contributors
(d) 12 Contributors (e) 2 Contributors (f) 11 Contributors

Figure 6: **Evolved Flower Examples.** The CPPN-based encoding allows the discovery of a great variety of aesthetically-pleasing flowers, which show varying degrees of complexity. The number of contributing users for each flower shows that the Petalz flower market facilitates meaningful collaboration.

evolved and later sold to other users. Figures 6 and 7 present examples of this process. While Figure 6 shows a selection of user-evolved flowers, Figure 7 shows a phylogeny, collaboratively evolved by 13 different users. The tree reveals that Petalz together with its marketplace allows users to collaborate on a wide variety of flowers. While some mutations produce a minor change in the flower's phenotype (Figure 7a), other mutations produce a greater variety of different flowers that nevertheless share common features (Figure 7c). Cross-pollination allows players to breed flowers that exhibit a mixture of the traits of both parents (Figure 7b).

However, while Petalz allows the player to evolve an unlimited variety of different flowers and the marketplace enables users to easily collaborate with each other, it became apparent that some measure of progress would become necessary to motivate players to continue playing such a game. Therefore we introduced the second Petalz milestone, a method for augmenting PCG-games with collection mechanics, which is described next.

V. Petalz Collection Mechanics

Collection game mechanics have worked well in engaging players in a variety of different games like Pocket Frogs or the Pokémon series. However, how PCG-based games can be augmented with 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 or GAR [6], 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 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 Petalz the procedurally generated content is classified based on the self-organizing map (SOM) algorithm [12]. While many other clustering algorithms could be applied, the SOM offers some unique advantages that give it particular appeal in a game context, which will be explained in the next section.

A. Self-Organizing Map

The SOM [12] is an artificial neural network that performs an unsupervised mapping from a high-dimensional input space R^D with 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: $||x(t) - w_c(t)|| \leq ||x(t) - w_i(t)|| \forall i$.

Once the BMU is determined, its weights and the weights of all the neurons within a fixed distance to the BMU are updated: $w_i(t+1) = w_i(t) + h_{c(x),i}(x(t) - w_i(t))$, where t is the index of the training step and h is called the neighborhood function, which is commonly a Gaussian function:

$$h_{c(x),i} = \alpha(t) e^{-\frac{||r_i - r_c||^2}{2\sigma^2(t)}}, \tag{1}$$

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.

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

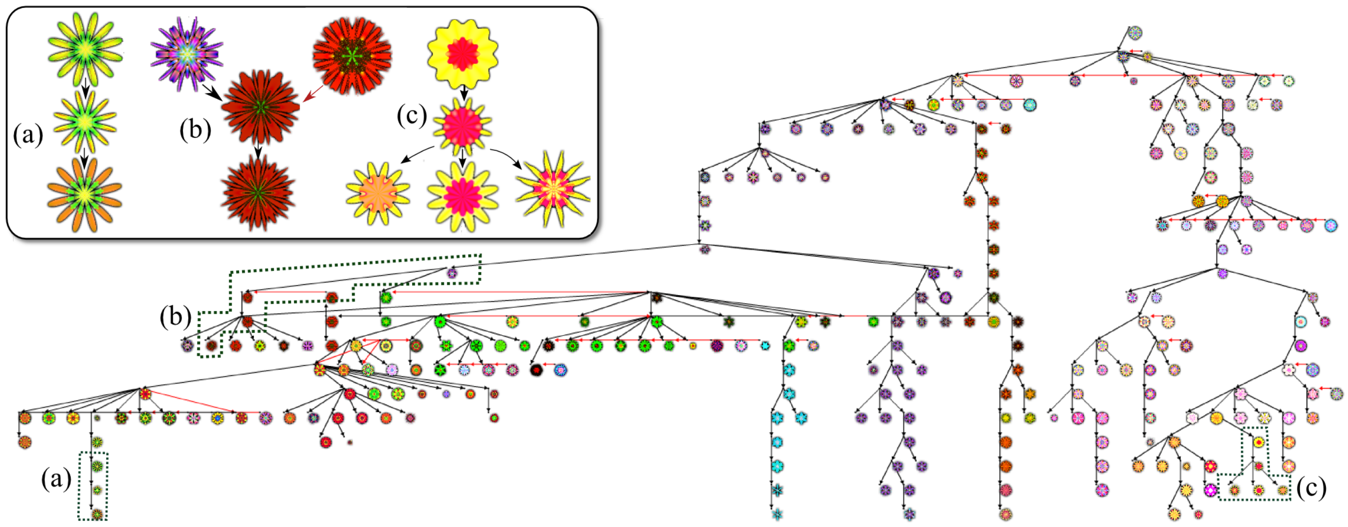


Figure 7: **A Phylogeny of Flowers.** This tree depicts the efforts of 13 different users. Each parent and child are separated by one generation (e.g. one pollination action). If a flower is the result of cross-pollination, the line connecting the second parent is shown in red. The inset shows modest phenotypic changes (a), a cross-pollination example (b), and more significant phenotypic changes (c) that nevertheless share common features.

The SOM can help with classification [12] 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 8 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.

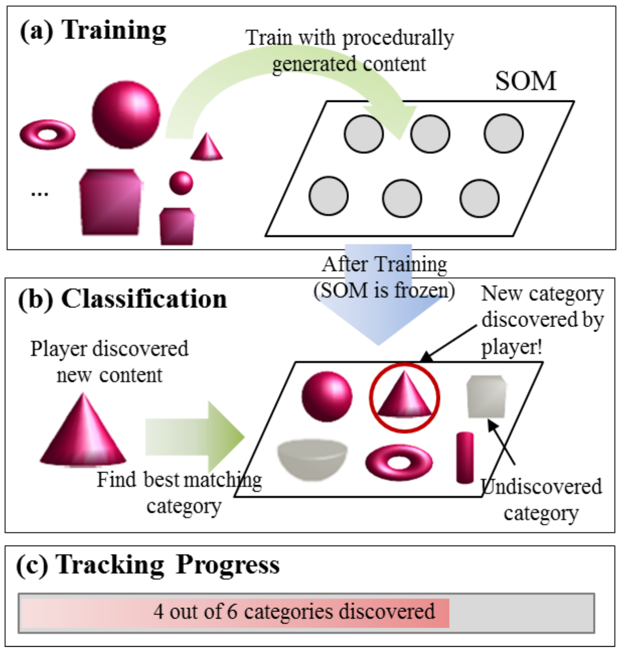

(a) Training
Train with procedurally generated content
SOM
...
After Training (SOM is frozen)
(b) Classification
Player discovered new content
New category discovered by player!
Find best matching category
Undiscovered category
(c) Tracking Progress
4 out of 6 categories discovered

Figure 8: **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.

B. Petalz Classification Experiment

To test the automatic categorization of procedurally generated in Petalz, 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.

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 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

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

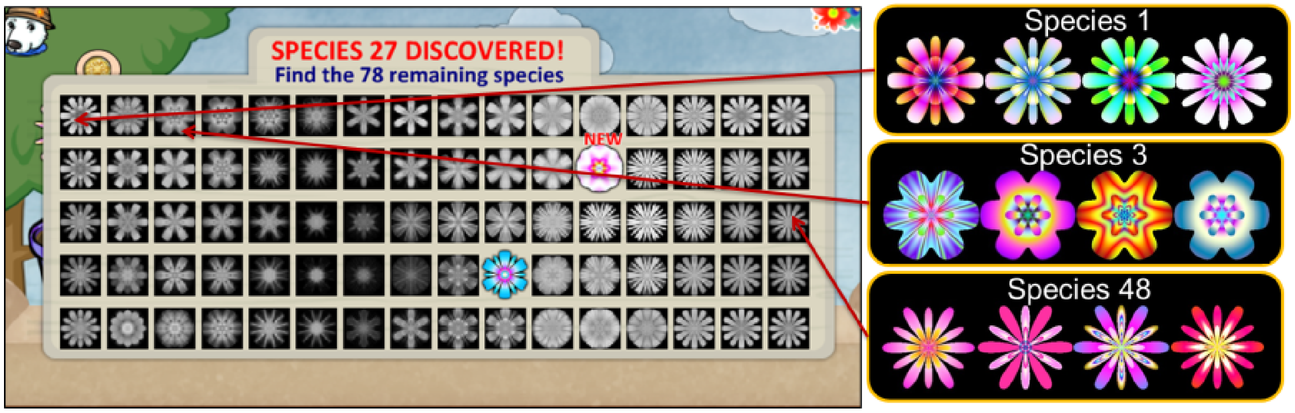


Figure 9: **Phenotypic Flower Clustering**. The picture depicts the 80 distinct species in Petalz. Examples of flowers clustered into three different species are depicted to the right. The main result is that the phenotypic clustering of flowers is meaningful and thus demonstrates the feasibility of 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.

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 [23] 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 ($\overline{W}$): $\delta = E + D + \overline{W}$.

C. Classification Results

Figure 9 shows the phenotypic clustering by the SOM algorithm in Petalz. 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.

Every time the player discovers a new species, a flower chart is displayed (Figure 9, 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 9 (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 10. 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.

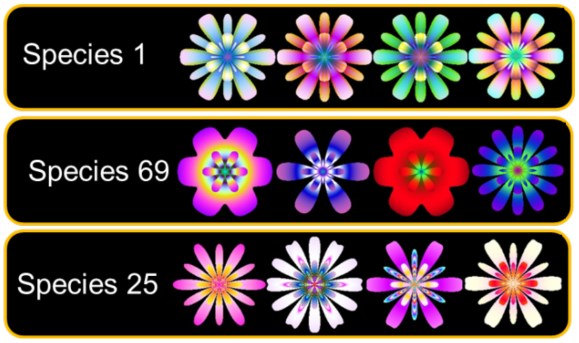


Figure 10: **Genotypic Flower Clustering**. This figure shows a SOM clustering of the flowers based on their pairwise genotypic distances. While such genotypic clustering sometimes produces meaningful clusters (e.g. *Species 1*), often flowers are grouped together that look significantly different (e.g. *Species 25*, *Species 69*).

The main classification 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, the results suggest that – at least in Petalz – a clustering based on phenotypic characteristics produces more intuitive classifications than one solely based on genotypic distances.

So far we have explored how a game with PCG offers opportunities for a global marketplace of evolved content and a collection mechanism that rewards players for acquiring diversity. Another interesting possibility for extending the impact of a PCG-focused game is to enable user-generated content to be physically embodied in the real world. An approach enabling such embodiment for the Petalz game is described in the next section.

VI. Automated 3D Printing of Evolved Flowers

While physically generating content originally evolved in simulation has been previously explored in collaborative IEC

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

(e.g. through printing evolved images on consumer items [19] or 3D printing an evolved shape [4]), it has not previously been applied to enhance PCG-focused *games*. Importantly, computer games may be a particularly promising domain for transferring simulated content into reality because games can provide mechanisms for encouraging emotional connection with in-game artifacts, which could motivate players to purchase embodied versions of such artifacts. For example, in an analogy to the real world, flowers in Petalz grow over time from small initial buds; the resulting uncertainty in how such flowers will eventually appear may stimulate anticipation of such flowers' first blossoming. In this way, players may begin to care about the game and the artifacts within the game that they have created, which might motivate them to purchase real-world versions of such artifacts. Furthermore, PCG in games can borrow existing social convention and meaning from the real world. For example, because real flowers are often given as gifts, it may be possible that once embodied, user-evolved flowers in Petalz could also be given as physical gifts.

For these reasons it may be promising to physically instantiate content evolved within PCG-focused games. This section focuses on exploring such an idea using Petalz as a case study. One such approach for the Petalz game is algorithmically to generate three-dimensional models from two-dimensional flower images generated within the game, such that the resulting three-dimensional models can then be 3D printed. In particular, the approach implemented in Petalz is to extrude the outline of a particular flower's bitmap to produce a three-dimensional model textured with the original bitmap, which is then integrated with the Shapeways 3D printing service. The entire process is automated and integrated with the game to enable players to easily bring simulated content into the real world. The next sections describe this process and its results in more detail.

A. Three-dimensional Flower Conversion Algorithm

There are two main stages in the process that converts two-dimensional evolved flowers in Petalz to three-dimensional flower models that can be successfully rendered by a 3D printer: (1) smoothing the two-dimensional flower to remove artifacts that cannot be 3D printed, and (2) generating a three-dimensional model from the smoothed two-dimensional flower. In the first stage the outline of the two-dimensional flower is analyzed for outcroppings that are too narrow to be successfully printed.

Rather than smoothing the generated flower bitmap itself, which might require fragile or complicated image processing algorithms, the implemented approach instead smooths the *CPPN-generated shape*. First the CPPN is queried for radii over all angles (as detailed in Section III-A) to create an array that defines the flower's outline. Then the smoothing algorithm iterates over this array repeatedly to detect and smooth away small outcroppings (see Algorithm 1). Note that the smoothing algorithm also enforces that each radius value is at least equal to a minimal threshold to ensure that each flower has a printable central core. Finally, using the smoothed radii (instead of the original queried radii), the CPPN is queried

Algorithm 1 Flower-smoothing Algorithm

```
 1: function SMOOTHFLOWER(radii)
 2:     outcropping_found ← True
 3:     radii_length ← length of radii
 4:     while outcropping_found do
 5:         outcropping_found ← False
 6:         for $i$ ← 0,radii_length-1 do
 7:             if OUTCROPPING_DETECTED(radii,i) then
 8:                 outcropping_found ← True
 9:                 radii ← MOVING_AVERAGE(radii)
10:     for $i$ ← 0,radii_length-1 do
11:         if radii[$i$] < min_radius then
12:             radii[$i$] ← min_radius
13:     return radii
```

to fill in the flower's color using the algorithm described in Section III-A.

In the second stage of the three-dimensional conversion algorithm the smoothed flower bitmap resulting from the first stage is converted into a three-dimensional model. The basic process is to extrude the outline of the flower to approximate the appearance of a real flower, texture the resulting shape with the flower bitmap to match the original flower's coloring, and finally to attach a green 3D printed base.

First the outline of the flower is extruded, which requires calculating a depth offset and thickness for each planar coordinate of the two-dimensional flower outline. The depth offset can be viewed as bending the flower's outline into the third dimension, while variable thickness across the flower's outline creates a recognizable flower shape. Three main features of real flowers were exploited to create natural three-dimensional flower models: natural flowers tend to have (1) petals that curve upwards as a function of distance from the flower's center, (2) petals that taper at their edges, and (3) a center differentiated from the flower's petals. To realize (1) upward curving petals, the entire two-dimensional flower outline was bent by the following curve: $z = \frac{{d_c}^{0.65}}{2.5}$, where d_c is the distance from the center of the flower and z is the magnitude that the flower outline is offset into the third dimension (i.e. bent upwards). To enable (2) tapered petals, thickness is *decreased* the nearer a point on the flower is to any edge of the flower's shape and the further a point is from the flower's center; the resulting petals thus taper towards their edges and as they project outwards. Finally, to (3) differentiate the center of the flower from its petals, the thickness of the flower's center was increased to create a slight spherical bump in the middle of the flower, which resembles a similar feature in some natural flowers. The particular implementation of (2) and (3) is the following calculation: $\text{thickness}(x, y) = \frac{3 \cdot d_e}{5 \cdot d_c} + \text{bump}(x, y)$, where d_e is the distance to the nearest edge of the flower and d_c is the distance to the flower's center. The bump term is specified by:

$$\text{bump}(x, y) = \begin{cases} 0 & \text{if } d_c > 0.25 \\ 3 \cdot (0.25 - d_c) & \text{if } d_c \leq 0.25 \end{cases}. \tag{2}$$

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

After the outline is extruded to create a three-dimensional model of the center of the flower and its petals, the next step is to texture map the model with the smoothed flower bitmap from the first stage of the conversion algorithm. This mapping is achieved simply through maintaining the unmodified coordinates of the bitmap throughout the extrusion process. In other words, before the flower outline is bent and extruded, it is merely a flat sheet whose planar coordinates directly correspond to the desired texture coordinates of the flower bitmap. The final step is to attach a short green cylindrical base to the bottom of the flower to enhance its similarity to a real flower. A hand-designed textured cylinder model is attached at the same set of coordinates to each processed flower. This attached cylinder model is hollow and can thus fit an optional separately-printed stem that can help to display the flower.

The end result of this process is a three-dimensional model that is realizable by a 3D printer, that bears significant similarity to the two-dimensional flower from which it is converted, and that has aesthetic features inspired by natural three-dimensional flowers. Figure 11.1 shows an example of a flower in Petalz and the resulting three-dimensional model created by this process.

B. Integration with Petalz and a 3D-Printing Service

While the previous section provides a means for generating three-dimensional models, to provide a seamless experience for players to 3D print their flowers additionally requires facilitating access to 3D printers and providing an in-game interface such that users can initiate model generation and purchase the 3D-printed result. Because 3D printers, especially those capable of color printing, are not yet household items, the model generation process was integrated with the API of a popular 3D-printing service called Shapeways (https://www.shapeways.com/). The Shapeways API provides means for three-dimensional models to be posted algorithmically to a virtual storefront from which users can purchase their flowers. After purchasing, the models are printed on 3D printers owned by Shapeways and the resulting physical artifacts are then shipped to the user. Note that the approach is not specific to Shapeways in particular; overall, the idea is that by integrating with any 3D printing service, it is possible to create a cascade of events that is initiated in-game by a user and culminates in a 3D-printable flower being added to an online storefront (https://www.shapeways.com/shops/finchbeak).

The gateway to trigger such a cascade in Petalz is implemented through adding an icon to the pop-up menu made visible when a player hovers their mouse over a flower (Figure 1). Once a player clicks on this icon, a new interface displays the results of the bitmap smoothing algorithm designed to remove unprintable artifacts. The motivation is to alert the user that the three-dimensional model may be slightly different than they expect. The user is then given the option to send the flower to Shapeways, which initiates the conversion pipeline and the player is then notified that in a few minutes the flower will then be available through the Petalz storefront on Shapeways. A banner displayed below the game provides a link through which the storefront can be reached.

Highlighting the culmination of the three-dimesional conversion algorithm and its integration into Petalz, a variety of three-dimensional flowers automatically converted to a three-dimensional model and printed through Shapeways can be seen in Figure 11.2. A total of 24 flowers have been printed so far (mostly ordered by the authors and their colleagues), which validate the functionality of the service and conversion process. As the feature has only recently been introduced, the hope is that many future users will enjoy bringing their simulated flowers into the real world.

VII. Discussion and Future Work

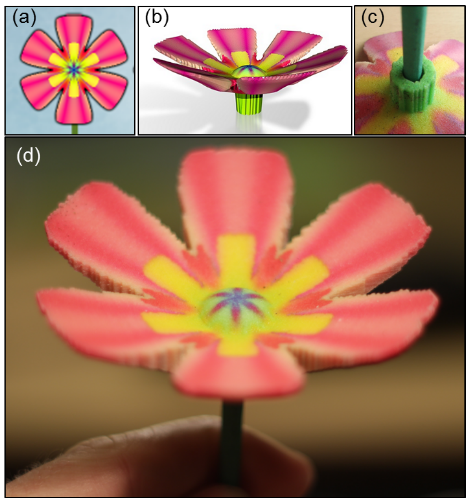
Overall this paper presents an ongoing experiment (the Petalz game) demonstrating how search-based PCG can act as a central game mechanic for casual online social games. Because such games are often singularly focused on content, the content-generating capabilities of user driven search-based PCG can provide a practical alternative to continuously paying professionals to generate novel content. In particular, in Petalz creating new content through IEC is a core game mechanic, facilitating a self-sustaining ecosystem of continual novelty. While Petalz provides an anecdote on the feasibility of this approach for casual social games, future work can explore applications beyond evolved flowers; for example, breedable virtual clothing or pets might also enable interesting casual games. However, beyond simply importing PCG into a social game, this paper also introduces specific mechanisms by which PCG can enhance such games, which are discussed next.

First, the marketplace in Petalz suggests that user-discovered PCG can potentially enable new kinds of in-game economies. For example, beyond the value of the artifact itself, there is value in each artifact's genetic potential for creating further novelty. That is, a player may want to purchase a flower from another user not only because they find it pleasing, but because they are interested in breeding variants of it. Furthermore, the analysis of the market data suggests that the marketplace both facilitates meaningful collaboration between users and positively influences the game dynamics. Importantly, these new insights open up unexplored possibilities in social gaming. Purchasing a flower from another player creates a new social interaction that links the players in an ongoing collaboration. Additionally, players can post their flowers to friends' Facebook walls, a unique gift possible to grant even to those who are not Petalz players. These novel social interactions ultimately allow users to discover and experience a continual stream of new content beyond what the original artists and developers can provide.

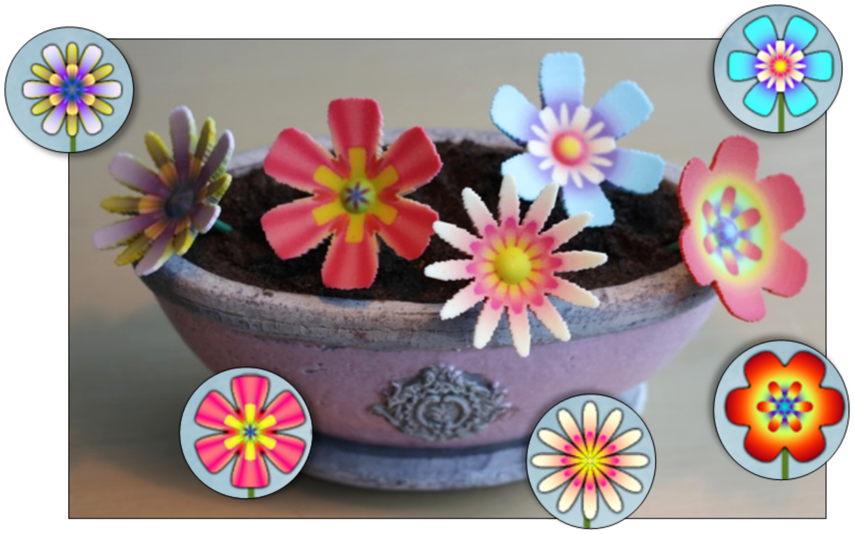
Second, the results also represent a proof of concept for a general method to augment PCG games with a collection game mechanic. Importantly, such a game mechanic can add meaning and structure to a game player's experience, which otherwise might be missing when PCG is the game's central focus. That is, 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 game

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games



11.1 3D Flower Transformation



11.2 Examples of Printed Artificially-Evolved Flowers

Figure 11: **Automated 3D Flower Printing** The automated three-dimensional flower transformation is shown in 11.1. The original flower (a) is processed to remove any outcroppings that would be too narrow and then converted into a three-dimensional model (b). The three-dimensional model, together with a separate stem that fits into the baes of the flower (c) is then printed in 3D through Shapeways (d). Figure 11.2 shows flowers evolved in Petalz and then physically instantiated through Shapeways. The ability to 3D print evolved artifacts allows the evolved content to be visible outside the context of a PCG-focused game.

player may have little direction 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.

Automatically clustering procedurally generated content also opens up applications beyond the singular collection game mechanic presented here. For example, many video games segment levels into higher-level groups that are often called worlds. Clustering procedurally-generated levels may provide a means for such higher-level 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 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.

Finally, seamless integration of 3D printing into a PCG-driven casual game offers a concrete mechanism to realize this possibility; such 3D printing enables evolved game content easily to escape the virtual boundaries of the Petalz game, thus enabling casual games more directly to affect the real world. In this way, beyond providing in-game economic value, 3D printing offers one way for evolved content to realize *real-world* economic value. That is, people appear willing to purchase three-dimensional incarnations of evolved flowers with real currency, which may provide an alternative means to monetize social games. Interestingly, physical procedurally-generated flowers may also have sentimental value; much like real flowers, 3D printed flowers can be given as a gift. In this light, PCG offers the potential for personalized creation of such gifts that goes beyond the traditional paradigm (e.g. at a florist's shop) of choosing from a limited selection of prescripted options.

VIII. Conclusion

By presenting several special casual gaming features in the Petalz video game, this paper highlighted the natural potential for PCG to impact casual games in particular. Features like the global marketplace, content collection dynamics, and printable custom three-dimensional models only reach their full potential in the context of a world in which content is continually rejuvenated and resupplied. By creating the possibility for such a continual stream of content to draw from the behavior of the players as opposed to requiring a dedicated team of artists and programmers, PCG in Petalz establishes the practical possibility of such features and raises the intriguing question of what other types of casual games might benefit from similar content-generating capabilities in the future.

IX. Acknowledgment

Special thanks to the Petalz testers and Petalz artist Ashlee Hynes.

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCIAIG.2015.2416206, IEEE Transactions on Computational Intelligence and AI in Games

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] Casual Games Association. Social Network Games 2012: Casual Games Sector Report, 2012.

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

[4] 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.

[5] S. Draves. The electric sheep screen-saver: A case study in aesthetic evolution. *Proceedings of EvoMUSART*, Lausanne, Switzerland, 2005.

[6] 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.

[7] 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.

[8] 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.

[9] 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 *Proceedings of the 3rd International Conference on Computational Creativity (ICCC-2012)*, 2012.

[10] D. Ines and G. Abdelkader. Facebook games: between social and personal aspects. *International Journal of Computer Information System and Industrial Management Applications*, 3:713–723, 2011.

[11] J. Juul. *A Casual Revolution: Reinventing Video Games and Their Players*. The MIT Press, 2012. ISBN 0262517396, 9780262517393.

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

[13] D. J. Mabberley. *Mabberley's plant-book: a portable dictionary of plants, their classifications, and uses*. Cambridge University Press, 2008.

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

[15] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67, 2010.

[16] 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.

[17] 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.

[18] S. Risi, J. Lehman, D. B. D'Ambrosio, and K. O. Stanley. Automatically categorizing procedurally generated content for collecting games. In *Proc. of the Workshop on Procedural Content Generation in Games (PCG) at the 9th International Conf. on the Foundations of Digital Games (FDG-2014)*, New York, NY, 2014. ACM.

[19] 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.

[20] N. Shaker, G. N. Yannakakis, and J. Togelius. Towards automatic personalized content generation for platform games. In *Proc. of Artificial Intelligence and Interactive Digital Entertainment (AIIDE-10)*. AAAI Press, 2010.

[21] 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.

[22] 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.

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

[24] 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.

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

[26] 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.

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

[28] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley. Procedural Content Generation: Goals, Challenges and Actionable Steps. In *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 61–75. Dagstuhl, Germany, 2013.

1943-068X (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.