2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Neuroevolution of Content Layout in the PCG: Angry Bots Video Game

William L. Raffe, Fabio Zambetta, and Xiaodong Li
School of Computer Science and Information Technology
RMIT University
Melbourne 3001, Australia
Email: {william.raffe, fabio.zambetta, xiaodong.li}@rmit.edu.au

***Abstract*—This paper demonstrates an approach to arranging content within maps of an action-shooter game. Content here refers to any virtual entity that a player will interact with during game-play, including enemies and pick-ups. The content layout for a map is indirectly represented by a Compositional Pattern-Producing Networks (CPPN), which are evolved through the Neuroevolution of Augmenting Topologies (NEAT) algorithm. This representation is utilized within a complete procedural map generation system in the game PCG: Angry Bots. In this game, after a player has experienced a map, a recommender system is used to capture their feedback and construct a player model to evaluate future generations of CPPNs. The result is a content layout scheme that is optimized to the preferences and skill of an individual player. We provide a series of case studies that demonstrate the system as it is being used by various types of players.**

I. Introduction

In game design theory, the amount of enjoyment that a player receives from a game is often linked directly to the challenge that they experience [1], [2]. The flow of content throughout a game map plays an important role in the challenges that the player will face in a single-player game. In this paper, *maps* are defined as the virtual environments that a player navigates around during game-play. The term *content* here refers to all virtual objects that a player interacts with during game-play, which may include friendly and enemy non-playable characters, item pick-ups, puzzles, or narrative elements that can be initiated. For example, if the relationship between the quantity of enemies and the quantity of pick-ups is analyzed, it can be seen that with too many enemies and not enough pick-ups the game will become more challenging. Meanwhile, an increase in pick-ups and a reduction in enemies will lead to less challenge.

In this paper we demonstrate a technique to procedurally distribute content throughout a map to provide a player with an appropriate challenge. This is achieved by using a *Compositional Pattern-Producing Network* (CPPN) [3] to calculate the location and quantity of content in each area of a map. A population of CPPN candidates is optimized through *Neuroevolution of Augmenting Topologies* (NEAT) [4]. NEAT has been successfully utilized in numerous *Search-based Procedural Content Generation* (SBPCG) applications [5], including the generation of art through the Picbreeder website [6], the calculation of the path of particle weapons in the game Galactic Arms Race [7], and the creation of flower designs in the social media game Petalz [8]. All of these applications also use a CPPN representation, which gives them all distinctive visual patterns. In the Picbreeder application, a CPPN candidate takes as input the (x,y) coordinates of a pixel of a blank image and outputs a color for that pixel to be set as. Similarly, in the content layout system presented here, the coordinates of each room in a game map are used as input to a CPPN and the output determines the quantity of each type of content.

Generating game maps is a sub-field of SBPCG and there exists work that has explored possible solutions that utilize evolutionary computing [9]. Both Frade et al. [10] and Raffe et al. [11] have proposed evolutionary techniques to creating virtual terrain to be used in open world games. Likewise, Ashlock et al. [12] and Cardamone et al. [13] evolve closed world maps in the form of mazes and first-person shooter game maps. Togelius et al. [14] generate more complete game maps by using a multiobjective evolution algorithm to optimize both the terrain and the location of key game objectives in maps of Real-Time Strategy games. Similarly, in this paper we present a system to generating both the geometry and the content layout of a map, which are separated into two distinct, yet interconnected, processes.

Furthermore, there has been work within the field of *Experience-driven Procedural Content Generation* (EDPCG) [15] to generate maps to suit the needs and preferences of players. Togelius et al. [16] use AI controllers that are modelled after individual players to evaluate procedurally generated tracks in a racing game. Shaker et al. [17] use artificial neural networks to predict a player's emotional state after playing a map in the game *Super Mario Bros* (Nintendo, 1985). In this paper, we frame the player modeling component of our system as a *content-based recommender system* [18], building a single model per player and using it as the fitness evaluator to the NEAT content layout algorithm.

To summarize, this paper makes the following contributions:

- We demonstrate how a CPPN can be used to calculate the quantity of content within an action-shooter game map.
- We utilize a combination of NEAT and a recommender system to generate personalized game maps.
- We show results of a recommender system being used to learn player preferences in PCG setting.

The rest of the paper is organized as follows; in Section II we describe the game that is used as the test bed for the

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

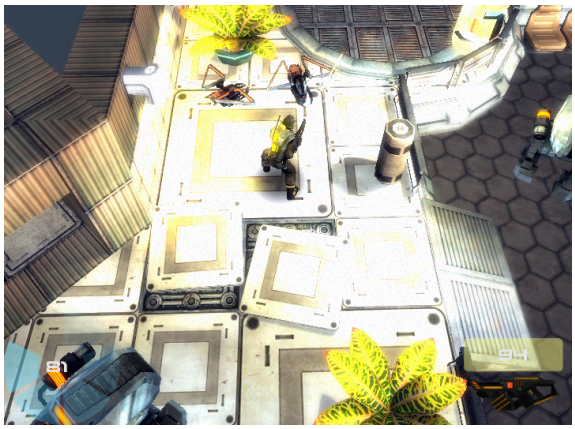


Figure 1. A screenshot of the game-play in PCG: Angry Bots.

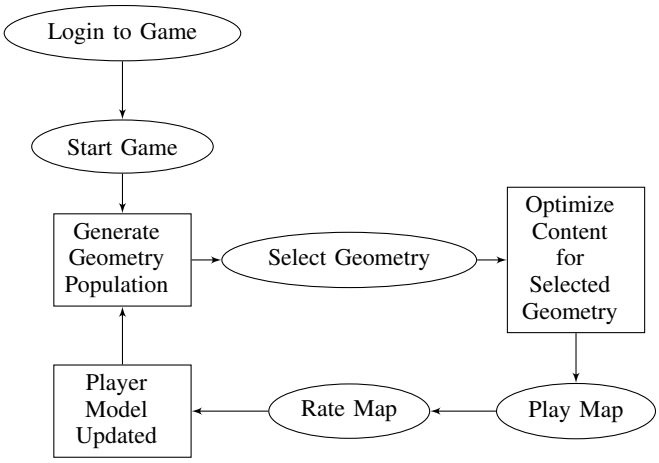


Figure 2. Simplified game cycle in PCG: Angry Bot. All blocks are system processes and all clouds are player actions.

experiments, as well as the general procedural map generation system. In Section III we provide details on the algorithms used to optimize the layout of content within a map. Case studies of numerous players interacting with the game are shown in Section IV and a conclusion is provided in Section V.

II. PCG: Angry Bots

The game used in the experiments in this paper has been dubbed *Procedural Content Generation: Angry Bots* (PCG: Angry Bots) and is built upon a technical demonstration that is provided freely with the Unity Game Engine [19]. PCG: Angry Bots is single-player, top-down action-shooting game that pits a lone soldier against swarms of security robots. A screenshot of the gameplay can be seen in Figure 1. The only real objective of the game is to get from one side of a map to the other. However, the player can choose to skip a map at any time if they feel it to be too difficult.

PCG: Angry Bots was designed with procedural map generation in mind and there are a few processes that the player interacts with to achieve this. Figure 2 shows a simplification of the game cycle, with all system processes represented as rectangles and all player actions represented as ovals. From a player's perspective, they first log into the game and choose a menu option to begin playing. They are then shown possible map choices and are required to select a map that appeals to them. Next, the player will see a brief loading screen and then be able to play through the map. After they complete the map they are required to rate it on a six point scale. The process then repeats.

The underlying system can be broken into three major components: geometry generation, content optimization, and player modeling. Instead of approaching the map generation process as a single process, as has been done in the past [20], it is instead separated into two important entities that form a map: the *geometry* and the *content*. Dormans and Bakkes [21] similarly use the terms Spaces and Missions. However, we use the term geometry to better relate to the architecture of a map. We also do not use the term Mission because our game does not have the type of segmented objectives that are present in the game created by Dormans and Bakkes. Instead, the term Content is used to refer to all objects that a player will interact with during game-play.

A. Evolving Geometry

The term *geometry* refers to the physical layout of the walls, floors, and doors of the map, encompassing the area that the player navigates through while playing the game. In PCG: Angry Bots a map is constructed as a series of rooms connected by corridors. All of the rooms and corridors here are pre-designed building blocks and so procedurally generating the geometry involves connecting them in an appropriate and valid manner. In this implementation, there are a total of 10 pre-designed rooms and 4 corridors.

The underlying genetic representation is a Fixed N-ary Tree where each node in the tree is a room and each edge is a corridor. Each node and edge holds a reference to a pre-designed room or corridor to be used when rendering. It should also be noted that as each of the 10 pre-designed rooms have between 2 and 4 doors, the room that is used at each node will determine how many children that node can have.

In the Fixed N-ary Tree, the root node is a starting room where the player first enters the map. Similarly, exactly one leaf node is an exit room, to which the player must try to reach to complete the map. As the representation is a tree and not a graph, there are no circular paths and there is only one path between the start and exit. Here, a 3-ary Tree is used, so each node can have a maximum of 3 children. This value is set as such simply because there are no pre-designed rooms with more than 4 doors.

The term "Fixed" means that once a node is put in the tree, its coordinates relative to nodes at the same depth never change, even if other nodes are removed. An example of a map is shown in Figure 3, where *node S* is the starting room, *node E* is the exit room, and all other nodes are labeled with a [depth, sibling number] coordinate. With this representation, if node $[2, 0]$ is removed, for example, nodes $[2, 1]$, $[2, 2]$, and $[2, 3]$ do not shift left, their coordinates stay the same. This representation is required for the content optimization process described in Section III.

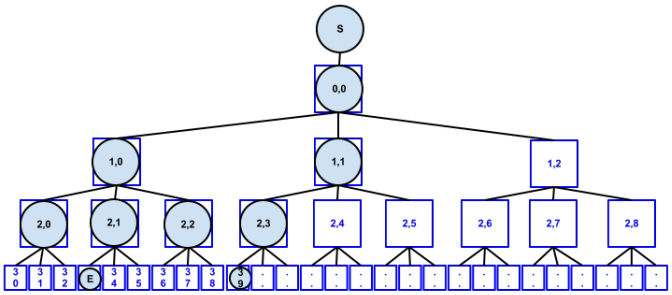


Figure 3. The geometry of a map represented as a Fixed N-ary Tree. Possible nodes are marked with squares while actual instantiated nodes are indicated as circles within the squares.

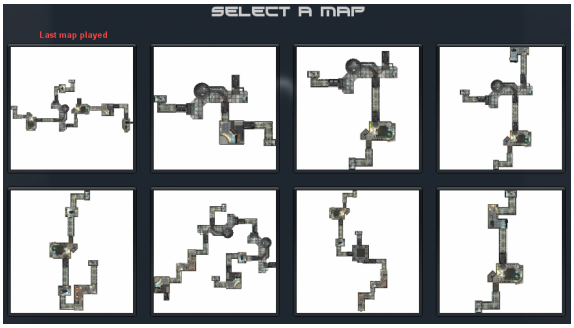


Figure 4. Screenshot of the geometry selection menu.

The geometry of maps is optimized through Interactive Evolution with a population size of 8. When a new player account is created, the first generation of maps is generated randomly. The player is then shown a preview of all 8 maps, as depicted in Figure 4, and they choose which map they would like to play next. This chosen map is then the single parent for the next generation. The offspring are generated through mutation only. Each node in the tree of the parent map has a random chance of mutating. If a node is to mutate, then either a new node is added above the current node and branches are created from it or the current node and all of its children that do not lead to node *E* are removed. Validation of each offspring is done to ensure there are no illegal intersections in the map. The mutation and validation is quick enough such that invalid candidates are simply discarded and mutation is attempted again. Interactive Evolution is sufficient here to capture the player's preferences as the previews of the maps clearly show the size of the map, the types of rooms that will be experienced, and the number of branching paths.

III. Content Layout through NEAT

For many game genres, the placement of content in a game map plays an important role in controlling the challenge, pace, and fairness of the game. For example, successive areas of high enemy density can lead to high tension and increased challenge as the player runs out of ammo and health. Thus, this section details the systems that we have designed to generate content layouts that are appropriate for an individual player in a linear single-player game.

Firstly, it is important to identify the types of content within the game. The PCG: Angry Bots game has 6 types of content in the two categories below:

- Enemies: Spiders, Buzz Bots, and Mechs.
- Pick-ups: Health, Ammo, and New Weapons.

Enemies will attack the player and can either be killed or avoided in order to complete a map. Pick-ups are items that the player can interact with to help them complete the map. Health and Ammo pick-ups give a fixed amount of their respective resource, while New Weapon crates provide a randomly generated weapon.

In order to discretize the content layout system, each room of the geometry can contain each of the 6 content types in the quantity settings of *None*, *Low*, *Medium*, or *High*. For example, the first room of a map may have a High setting of Mech enemies, a Medium setting of Ammo, and a None setting for all other content. The pre-designed room Geometries discussed in Section II-A also have the content locations pre-defined, thus the process of procedurally generating the content layout in a room is that of removing unwanted content rather than adding new content. This approach was chosen to reduce the need for additional systems to choose logical and valid positions of content within a room.

The process of evolving and optimizing maps requires that a setting (None, Low, Medium, or High) is chosen for each content type (Spiders, Buzz Bots, Mechs, Health, Ammo, and Weapons) in each room of the user selected geometry. To achieve this, the content settings of a map are indirectly represented by a Compositional Pattern-Producing Network (CPPN) [3]. A CPPN is an artificial neural network in which the activation function at each neuron can be selected from a variety of possible functions, rather than restricting every neuron to using the same function. This property can lead to patterns of output occurring, which can be seen visually in the Picbreeder application [6]. In PCG: Angry Bots, the inputs to a CPPN are a nodes depth and sibling coordinates in the *Fixed N-ary Tree*, as shown in Figure 3, as well as a bias of 1.0. The *Fixed N-ary Tree* representation becomes important here as a single mutation in the geometry will not affect the coordinates passed to the CPPN for the rest of the map.

There are 6 outputs from a CPPN, each corresponding to a content type. Each output specifies a value between $[-1, 1]$, where -1 to -0.5 indicates a *None* setting, -0.5 to 0 indicates a *Low* settings, 0 to 0.5 for a *Medium* setting, and 0.5 to 1 for a *High* setting. A final, playable map is generated by combining a geometry and a CPPN. That is, each node in the geometry is passed into the CPPN and the content is set by the output.

A population of CPPN candidates is evolved using Neuroevolution of Augmenting Topologies (NEAT) [4]. PCG: Angry Bots uses the *SharpNEAT* [22] implementation of CPPN and NEAT. A population size of 50 was used with a species count of 5. An acyclic network scheme was used, as well as an absolute complexity regulation strategy with a complexity threshold of 50. This large complexity threshold was used because we did not want to limit the complexity of the networks, which was due to initial testing showing that more complex networks led to more appropriate patterns of content within a map. For each map that is played, 10,000 fitness evaluations are allowed to take place in order to allow time

for an appropriate CPPN to be found. The fitness evaluation method detailed in Section III-A is an efficient process and all of the evaluations for a single map can be accomplished in less than 5 seconds on a 3GHz CPU.

A. Fitness Evaluation

An important objective of this work is to generate game maps that are appealing to individual players. During the evolution of the geometry, interactive evolution was capable of clearly representing options to the player and capturing their feedback in an almost reflexive time frame. However, there is no such clear representation of the content layout; players would need to carefully study figures or variable values carefully to properly evaluate the layouts of each map.

Therefore, the fitness evaluation is automated by creating a player model and using it to evaluate each CPPN candidate. To do this, the solution is framed as a content-based recommender system [18]. A model-based approach is used, which builds a classifier from user ratings of items and is used to predict the ratings of items that have not yet been experienced. Here, similarities are drawn between items, not users, and thus a separate model is constructed for each user without influence from the models of any other users.

In our system, after a player has finished a map, they provide a rating for it. These ratings, along with features that are extracted from the maps, are used to update the player model. The player model is then used to evaluate new map candidates within NEAT. The map candidates are not fully rendered and instead the features are extracted directly from the tree structure. The candidate with the highest fitness is presented for the player to experience and the cycle continues with the player model becoming more accurate with every new rating.

In PCG: Angry Bots, a Naive Bayes (NB) classifier [23] was used as the player model due to its ability to learn from limited data samples. Initial testing with numerous classifiers in the WEKA machine learning suite [24] also showed NB to perform the best with our chosen features. There are a total of 18 map features that were used in the NB model, which included:

- Enumerated Sums - The settings {None, Low, Medium, High} are enumerated as $\{0, 1, 2, 3\}$ and a sum of each of the six content types across the entire map is calculated.
- Room Composition Counts - The six content types are condensed to two categories, Enemies (E) and Pick-ups (P), and the four settings are condensed into Low and High. This creates 4 possible room types: LowE-LowP, LowE-HighP, HighE-LowP, and HighE-HighE. The quantity of each room composition is counted throughout the map.
- Room Transition Counts - As with the Room Compositions, content types and settings are reduced. However, these features determine the transitions from one room to another. There are 4 enemy transition type: LowE-to-LowE, LowE-to-HighE, HighE-to-LowE, and HighE-to-HighE. There are also 4 similar transition types for pick-ups and so each edge of the geometry tree will belong to one enemy transition type and one pick-up transition type.

All of the above feature values are normalized by dividing by the total number of rooms in the map. These features were chosen as a means of understanding the flow of content throughout a map without using a large number of features, which would have been detrimental to the NB classifier.

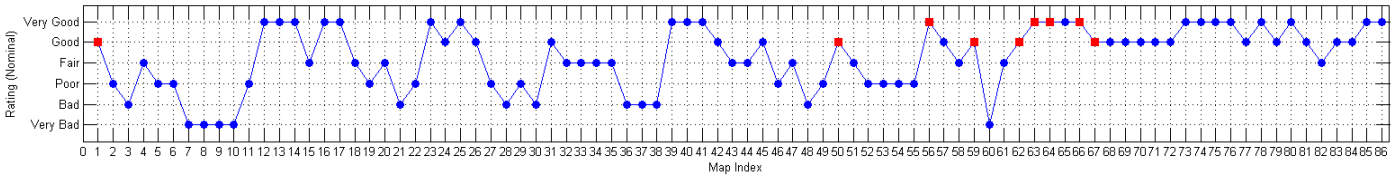
After the player has finished a map, they are required to rate it on a six point scale, from "Very Bad" to "Very Good". These ratings can be seen later in Figure 6. However, early testing showed that the NB classifier did not perform well with a multinomial class setup. An alternative is to use a rating system of "Dislike" and "Like", which would provide a binary class setup. Binary classification worked well in early testing, however, the disadvantage was that it did not capture a player's preferences to a fine enough granularity. For example, a player may have enjoyed a map but not as much as they enjoyed a prior map. Thus, a trade off was made by converting the player's multinomial rating into weighted binary class for the NB classifier. The ratings {Very Bad, Bad, Poor, Fair, Good, Very Good} were evenly divided into the classes {Dislike, Like} and given weights in those classes of $\{2, 1, 0.5, 0.5, 1, 2\}$ respectively. This means that ratings of Very Bad and Very Good would have the most influence over their respective classes, while Poor and Fair have a weak influence over the class. The fitness value for a CPPN is its predicted membership to the "Like" class given the extracted map features.

IV. Results

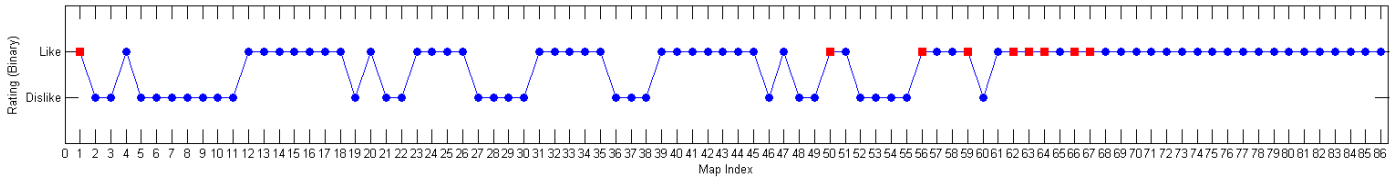
As of the writing of this paper, the game PCG: Angry Bots, including all of the mechanisms described above, has recently been released in an open experiment [25]. Invitations to participate have been distributed through various social networking channels and participation is anonymous. The game can be downloaded online and requires an internet connection during play to allow for player data to be collected and stored on secure servers. There are no minimum or maximum limits on the number of maps a participant can play.

As this experiment is still being conducted, the initial results shown here do not make a claim to the overall success of the system but rather give a qualitative analysis of the type of experiences that players are having as a result of the CPPN approach to content layout. Along with an initial analysis of the learning capabilities of the player models, these results show that, at least for these three players, the system is able to gain an understanding of the player's preferences and in turn generate maps that are more appealing to them.

The results in Figures 6 and 9 show three of the longest playing participants so far, henceforth referred to as Player 1, Player 2, and Player 3, who played 17, 18, and 16 maps respectively. These three players were also chosen because they demonstrate quite distinct preferences of game-play, with Player 1 enjoying easy maps, Player 3 preferring challenging maps, and Player 2 desiring a level of challenge somewhere between that of the other two participants. It should be pointed out that as participation in this experiment is anonymous, we cannot truly know the preferences of the players and below are only our estimations.

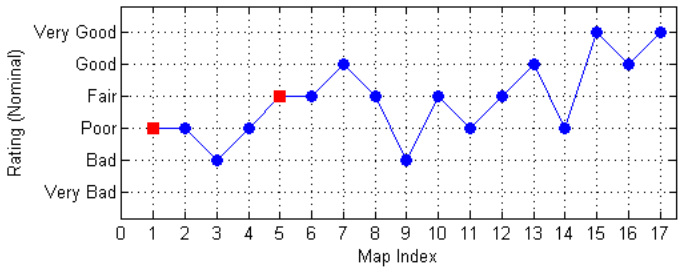


(a) Multinomial ratings

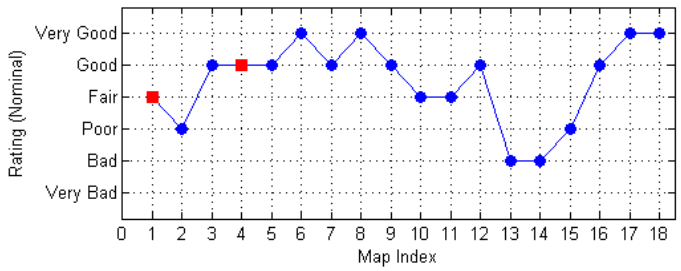


(b) Binary ratings

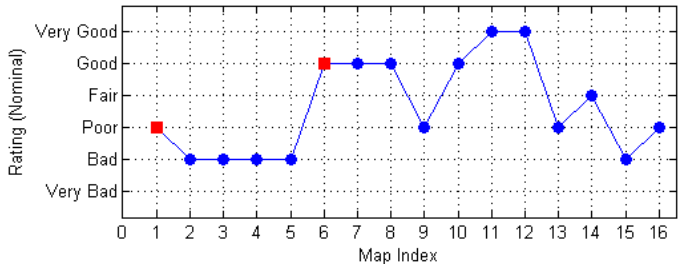
Figure 5. (a) The ratings provided by the player who experienced the most maps. (b) The same ratings converted to the binary scale used by the Naive Bayes classifier.



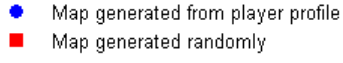
(a) Player 1



(b) Player 2



(c) Player 3



(d) Legend for the rating plots above.

Figure 6. Plots of the ratings provided by each of the three chosen player's for every map that they played.

A. Raw Ratings

Firstly, Figure 6 shows the rating that each player gave to each map they experienced. The plots show the multinomial rating scale that is presented to the player during the game, as oppose to the weighted binary scale used by the NB classifier. The plot for Player 1 shows a steadily improving rating over time, suggesting that the system is properly identifying his preferences. Both Player 2 and 3 show an increase in rating after a few games but a sudden drop towards the end of play. It is believed that this trend occurs for one of the following reasons.

- The NB classifier is pushing the NEAT evolution in a specific direction (e.g. adding more enemies) but has pushed the boundaries of what is acceptable too far (e.g. too many enemies have now been added).
- The geometry tree grows in depth and the CPPN population is not optimized for the lower levels of the tree.
- The player has become bored of the repetitive content layout that is a result of the NEAT algorithm being stuck in a local optimum. Alternatively, the player's skill or preferences have changed and what previously interested them no longer does. Finally, this change in preference may also be due to player fatigue during long sessions of play [26] that can lead to altered decision making or a reduction in skill.

Regardless, if a player is to continue to play beyond these few bad maps, the ratings appear to improve again shortly after the sudden drop. The most maps played by a single player so far is 86, with the next highest number of maps played being 18. Figure 5a shows the ratings provided by this player. When such a large number of maps are played, the trend of sudden drops in ratings followed by a return to appealing maps appears to re-occur over the lifetime of play, before remaining constantly positive after map 62. This trend is more easily noticed in Figure 5b, which depicts the binary ratings used by the NB classifier.

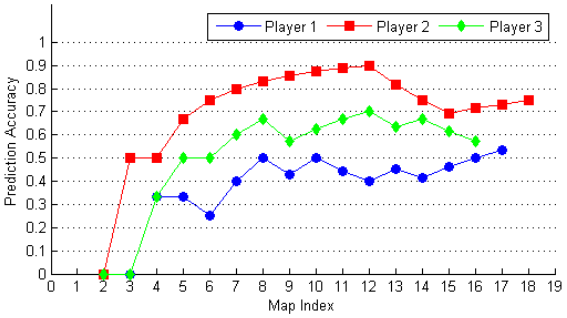


Figure 7. Prediction accuracy of the three classifiers of the chosen players. The accuracy at any map index includes the predictions of all maps before it.

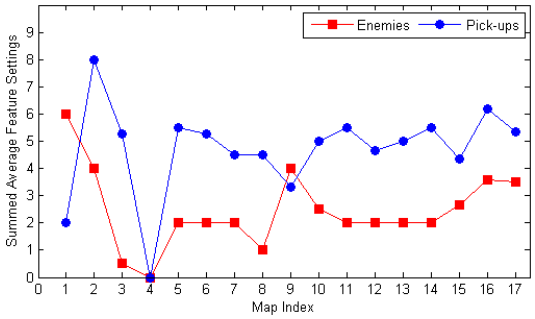


Figure 8. Feature values for the maps played by Player 1. The features values are summed into Enemies and Pick-ups for clearer viewing.

B. Learning of Player Models

Figure 7 shows the prediction accuracy of each of the three players' classifiers. The prediction accuracy was gathered by comparing the prediction given to a map by the player's current classifier and the rating that is actually given by the player. Note that this is based upon the binary rating system used by the Naive Bayes classifier and so it is possible to build a confusion matrix and prediction accuracy can be calculated as a function of true classifications over all classification instances.

It is possible to experience True Negatives and False Negatives because not all maps that are provided to the player are believed to be suited for them; sometimes the system has no choice but to provide a map that is believed to be unsuitable for the player because the NEAT evolution was not able to discover a map that they would like. These plots do not include predictions for maps that were randomly generated, such as the first map played by every player.

In Figure 7, there is a trend of positive learning for all of the players. Both Player 1 and Player 3 experience fluctuations in prediction accuracy. For Player 1, this is most likely due to many of the provided ratings being either "Fair" or "Poor" and without strong samples of either "Very Good" or "Very Bad" the classifier is struggling to learn. The classifier for Player 2 shows the greatest strength with a maximum accuracy close to 90%. However, there is a drop in accuracy between Map 13 and 15 and by comparing this to the ratings in Figure 6 it can be deduced that the classifier had predicted that Player 2 would enjoy those maps but they didn't, most likely introducing confusion into the training data. By Map 16, however, the classifier is beginning to learn from these new data samples and is improving in accuracy again.

Figure 8 shows the Enumerated Sums feature values of each map played by Player 1. These values have been summed into two categories: Enemies and Pick-ups. For example, a value of 9 indicates that there is the maximum setting of every enemy type in every room of the map. This plot is an example of feature convergence as a result of the NB classifier learning. The features begin to stabilize in the region of Maps 5 and 7 and then again between Maps 10 and 14, especially with respect to enemy quantities. This indicates that the classifier has learned an Enemy to Pick-up ratio that is believed to be suitable for the player. However, the features begin to fluctuate again in Map 15 and this is due to the sudden drop in rating of Map 14. A closer look at Map 13 and 14 (not shown in this paper) indicate that the content settings were nearly identical but, for unknown reasons, the player chose to rate Map 14 much worse. This contradicted the classifiers prior knowledge and therefore causes it to promote adjustments in feature values.

C. User Experiences

This section describes the experience of the three chosen players by showing a sample of maps that were generated for each of them. Figure 9d shows the color legend that is used in Figures 9a, 9b, 9c. Each colored square in the images of the maps represents a single piece of content. The room that the player starts in is marked with an "S" while the exit room is marked with an "E". The map identification numbers match those shown in Figure 6.

Figure 9a shows four maps that Player 1 experienced. Player 1 appears to be a novice, preferring maps with fewer enemies. At Map 3 there is not enough classifier data and the maps are being optimized in a negative direction. Map 3 has only a single Spider enemy and while Player 1 wants an easy map, this appears to be too easy. Map 5, which is generated randomly, gathers a positive rating to balance the knowledge of the NB classifier. This is built upon in Map 7, which now has a few more Spider enemies in each room. However, this trend of optimizing towards more enemies continues and as a result Map 9 is overly dense with enemies. By Map 15 the system has returned to an appropriate enemy quantity.

Some of the maps that Player 2 experienced are shown in Figure 9b. Map 2 has a high density of all content types but it is likely that there were too many enemies for this player. Map 3 is optimized away from Map 2, receives a good rating, and thus optimization continues towards fewer enemies. Map 6 shows the pinnacle of this optimization, which contains one Mech enemy in each room, a few Buzz Bot enemies, and plenty of ammo. However, this balance is broken in Map 13 and 14, which both contain higher numbers of Mech and Spider Enemies. In Map 18, the Spider enemies have been removed, leaving only the Mech enemies and plenty of Ammo and Weapon pick-ups, which seems to be an appropriate challenge for this player.

The plot for Player 3 in Figure 6 and the maps in Figure 9c demonstrates how the map optimization process can move in the wrong direction and get stuck in a local optimum early on. The first map that is shown in Figure 9c, which is randomly

generated, is dense with enemies but comparatively scarce with ammo and health. In Map 2, a few of the enemies have been removed but so has the ammo and health and is rated worse than the first map. By Map 5 almost all content has been removed. This behavior is a result of no positive examples for the NB classifier to learn from. This is why the option to completely randomly generate a map was provided to the players. Map 6 is randomly generated and is rated highly. The system utilizes this positive rating and the result is Map 7, which has returned to higher densities of enemies and pick-ups.

V. Conclusion

This paper presented an application of CPPN to laying out content within a game map. This approach has been implemented into the 3D action-shooter game *PCG: Angry Bots*, in which players navigate from one end of a map to the other. The result is a system that dictates how many enemies and pick-ups are presented to the player in each room of the map. A population of CPPN candidates is evolved through NEAT, utilizing a recommender system during fitness evaluations to encourage content layouts that will be appropriate to the preferences and skill of an individual player.

While the results presented here only show initial samples from a public user experiment, they demonstrate the potential of this system. The results from three players were shown as examples of how the combination of NEAT and a recommender system were able to generate maps that appealed to players with different preferences. Noticeable rating fluctuations are believed to occur when the recommender system does not have adequate data to learn from or when a player's own preferences change, creating conflicts in a previously strong data set. As the public user experiment continues, it is hoped that the trends that were observed for these three players will be also be apparent for other participants. Also, with more participant data we will be able to establish a clearer picture of how well the system is working and why certain phenomena, such as the recurrent rating drops, may occur.

Acknowledgment

The experiment in this paper is built upon the "Angry Bots" technical demonstration that is provided with the Unity Game Engine [19]. All rights to the graphical and audio assets, as well as many core game-play scripts, belong to Unity Technologies and are used here under a Unity Pro Educational license.

References

[1] T. Malone, "What makes things fun to learn? a study of intrinsically motivating computer games." *Pipeline*, 1981.

[2] E. Byrne, *Game level design*. Delmar Thomson Learning, 2005.

[3] K. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007.

[4] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[5] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based Procedural Content Generation: A Taxonomy and Survey," *Computational Intelligence and AI in Games, IEEE Transactions on*, no. 99, pp. 1–1, 2011.

[6] J. Secretan, N. Beato, D. D'Ambrosio, A. Rodriguez, A. Campbell, and K. Stanley, "Picbreeder: Collaborative interactive evolution of images," *Leonardo*, vol. 41, no. 1, pp. 98–99, 2008.

[7] E. Hastings, R. Guha, and K. Stanley, "Evolving content in the galactic arms race video game," in *IEEE Symposium on Computational Intelligence and Games (CIG)*. IEEE, 2009, pp. 241–248.

[8] 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 *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*. AAAI, 2012, pp. 63–68.

[9] W. Raffe, F. Zambetta, and X. Li, "A survey of procedural terrain generation techniques using evolutionary algorithms," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[10] M. Frade, F. F. de Vega, and C. Cotta, "Evolution of artificial terrains for video games based on obstacles edge length," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–8.

[11] W. Raffe, F. Zambetta, and X. Li, "Evolving patch-based terrains for use in video games," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 363–370.

[12] D. Ashlock, C. Lee, and C. McGuinness, "Search-based procedural generation of maze-like levels," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, no. 3, pp. 260–273, 2011.

[13] L. Cardamone, G. N. Yannakakis, J. Togelius, and P. L. Lanzi, "Evolving interesting maps for a first person shooter," in *Applications of Evolutionary Computation*. Springer, 2011, pp. 63–72.

[14] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelback, and G. Yannakakis, "Multiobjective exploration of the starcraft map space," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. IEEE, 2010, pp. 265–272.

[15] G. Yannakakis and J. Togelius, "Experience-Driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, 2011.

[16] J. Togelius, R. De Nardi, and S. Lucas, "Towards automatic personalised content creation for racing games," in *IEEE Symposium on Computational Intelligence and Games (CIG)*. IEEE, 2007, pp. 252–259.

[17] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE). AAAI Press*, 2010.

[18] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, 2005.

[19] Unity Technologies. (2012) Unity Game Engine 4.0. [Online]. Available: http://unity3d.com/ Accessed: 1 January 2013

[20] J. Togelius, M. Preuss, and G. Yannakakis, "Towards multiobjective procedural map generation," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010, pp. 1–8.

[21] J. Dormans and S. Bakkes, "Generating missions and spaces for adaptable play experiences," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, no. 3, pp. 216–228, 2011.

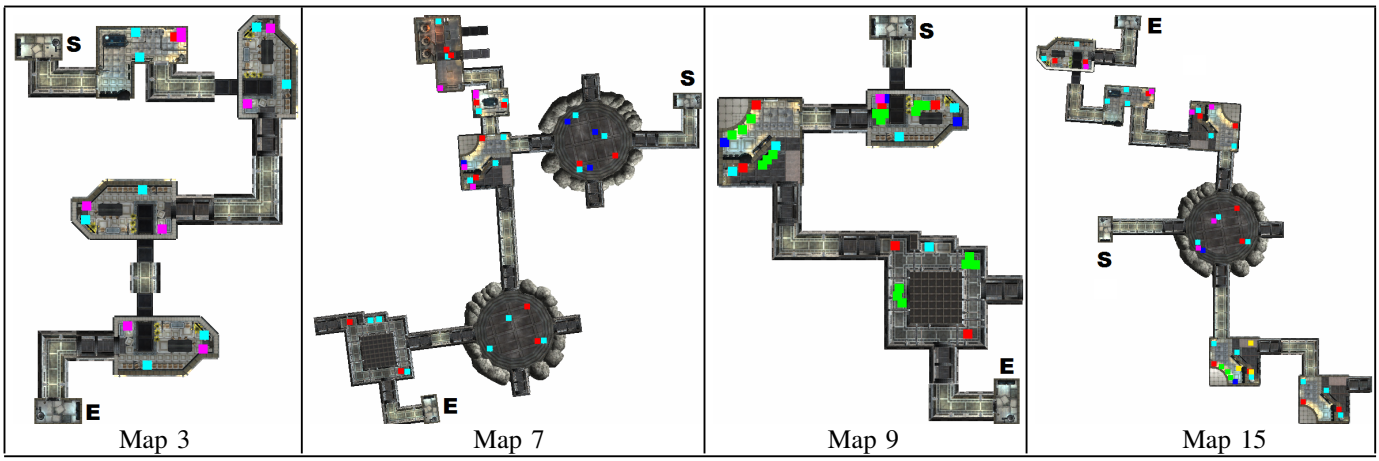
[22] C. Green. (2004) Phased Searching with NEAT: Alternating Between Complexification and Simplification. [Online]. Available: http://sharpneat.sourceforge.net/phasedsearch.html

[23] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting web sites," *Machine learning*, vol. 27, no. 3, pp. 313–331, 1997.

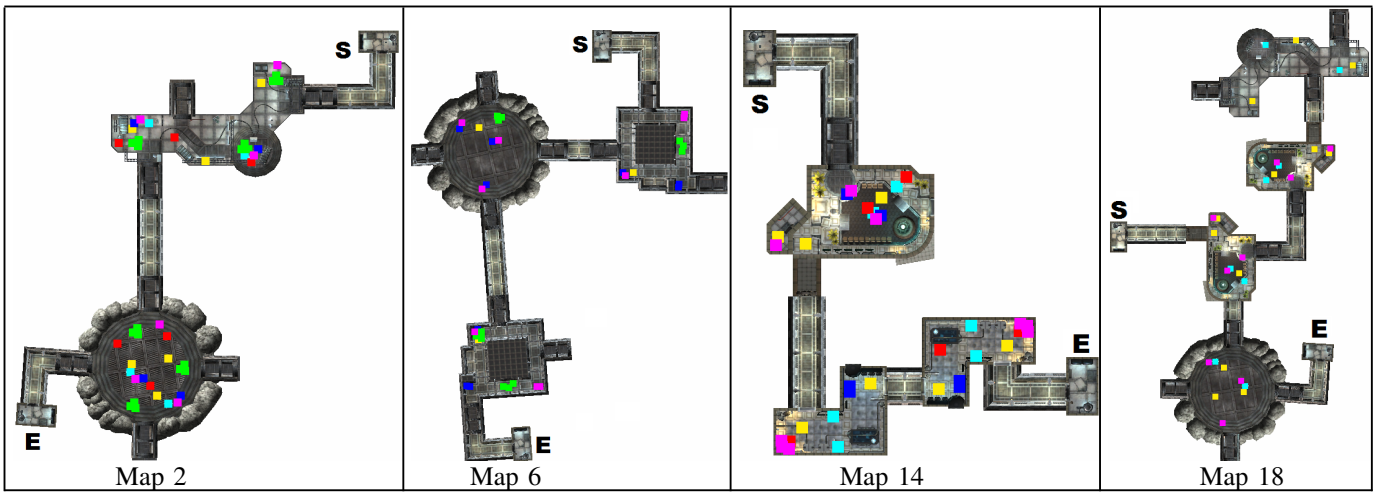
[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[25] W. L. Raffe. (2013) PCG: Angry Bots. [Online]. Available: http://goanna.cs.rmit.edu.au/~wraffe/ExperimentHome.html Accessed: 1 January 2013

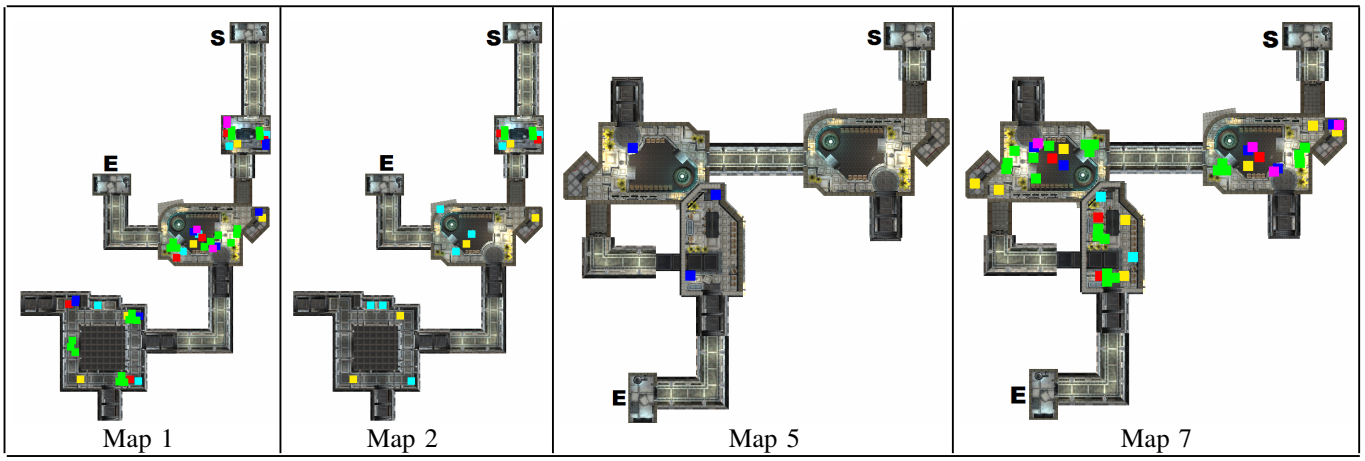
[26] C. Cotta and A. J. Fernández-Leiva, "Bio-inspired combinatorial optimization: notes on reactive and proactive interaction," in *Advances in Computational Intelligence*. Springer, 2011, pp. 348–355.



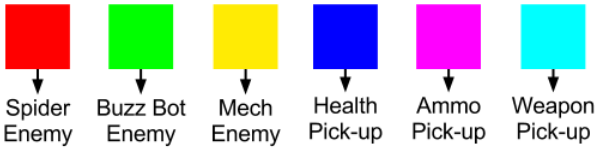
(a) Player 1



(b) Player 2



(c) Player 3



(d) Color legend for the content in the maps shown above.

Figure 9. A sample of the maps played by (a) Player 1, (b) Player 2, and (c) Player 3. Each colored square in the map samples indicates a single piece of content, corresponding to the color legend in (d).