Procedural Generation of Angry Birds Fun Levels Using Pattern-Struct and Preset-Model

Yuxuan Jiang
Intelligent Computer Entertainment Laboratory
Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
Email: ukenplayerx@yahoo.co.jp

Tomohiro Harada
Intelligent Computer Entertainment Laboratory
College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
Email: harada@ci.ritsumei.ac.jp

Ruck Thawonmas
Intelligent Computer Entertainment Laboratory
College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
Email: ruck@is.ritsumei.ac.jp

***Abstract*—In this paper, we describe our design of a generator for automatically generating fun levels in the famous game Angry Birds and its clone games. In particular, we aim at generation of character-sequence levels. In order to generate fun levels, we propose a pattern-struct approach and a preset-model approach. The pattern-struct approach generates each time different appearance for one of the alphabetical letters, numeral numbers, or some symbols. The preset-model approach is used for increasing the legibility and diversity, where models are built in advance for some capital letters and numbers. Our level generator, called Funny Quotes, was the winner of the Fun Track of the CIG 2016 Level Generation Competition held in 2016 IEEE Conference on Computational Intelligence and Games. After raising issues residing in Funny Quotes through user-evaluation results in terms of legibility of characters in generated levels and of playability of those levels, we propose and evaluate new mechanisms for ensuring the disparity in difficulty among easy, normal, and hard levels for the 2017 competition.**

I. Introduction

Nowadays, due to continuing rise of costs and scale expansion in game development, procedural content generation (PCG) has gained much attention as a promising area whose findings can solve such issues. PCG can be used for generating contents such as maps or models. According to Shaker et al. [1], PCG refers to computer software that can create game contents on its own, or together with one or many human players or designers.

In this research, we aim to design a generator that can automatically generate fun levels for the famous game Angry Birds and its clones. The key feature of our level generator is that it generates character-sequence levels. In order to realize this feature, we propose a pattern-struct approach and a preset-model approach. The former approach generates each time an appearance in a varying style for each of the lower-case letters, capital letters, numbers and some symbols. The latter approach is used for increasing the readability and variation in letters and numbers, where models are built in advance for some capital letters and numbers. Our level generator, called Funny Quotes, was the winner of the Fun Track of the Angry Birds Level Generation Competition held in the IEEE Conference on Computational Intelligence and Games (CIG) in 2016.

The aforementioned track evaluates each level generator based on the overall fun or enjoyment factor of the levels it creates. Other factors considered include whether the generated levels can be cleared and whether the difficulty of levels are properly adjusted. According to the organizers, Funny Quotes was evaluated as follows: “Despite being similar in style an despite not being very challenging to solve, it was always fun to play and never got boring.” To cope with “not being very challenging to solve” for the 2017 competition, we have decided to make resulting levels more challenging overall, but maintaining variation in difficulty.

The remainder of this paper is organized as follows. We introduce related work and the CIG 2016 Angry Birds Level Generation Competition Fun Track in Section II. Section III presents the design strategies of our generator and how we use the aforementioned pattern-struct approach and preset-model approach to generate fun levels for Angry Birds. User studies and results of Funny Quotes and its improved version for the 2017 competition are described in Sections IV and V, respectively, followed by conclusions and future work in Section VI.

II. BACKGROUND

A. Related Work

Angry Birds is a famous action-puzzle game developed by a Finnish company called Rovio Entertainment. The first Angry Birds game in the series was initially released in December 2009. The purpose of the game is that of using a slingshot to launch a bird to rubbish all pigs in the stage. Previous studies [2]-[9] have also explored the use of PCG for Angry Birds.

Ferreira et al. [2] used genetic algorithm (GA) [10] to optimize stability of levels. They also proposed a method that can generate Angry Birds levels containing columns of

978-1-5386-3233-8/17/$31.00 ©2017 IEEE

some predefined structures or single objects [3]; this method uses probability tables updated by an estimation of distribution algorithm [11] to locate objects in a given level. In their most recent work [4], a machine learning technique was used to estimate the stability and feasibility of generated levels instead of simulation that requires high computational time.

Kaidan et al. [5], from our research group, proposed a level generation method that improves Ferreira's work [2] by adapting the difficulty of levels to the player's skills measured according to his or her gameplay results. They later proposed another GA-based algorithm [6] that uses the extended rectangle algebra to quantitatively evaluate the levels' difficulty, according to the structural relation of blocks and pigs in generated levels, and they introduced this evaluation into the fitness function of GA. Another previous study [7] by our group proposed 2D building construction grammars that represent the structural patterns of typical buildings with Chinese-style and/or Japanese-style models in order to imitate Chinese and Japanese architectural styles on Angry Birds levels.

Stephenson and Renz [8] proposed an algorithm that creates complex stable structures using a variety of 2D objects without pre-defined substructures or composite elements. Resulting structures are evaluated based on a fitness function that considers several important structural aspects such as the number of pigs and blocks. In their other work [9], they presented a procedural generation algorithm that generates levels consisting of various self-contained structures placed throughout the 2D area, and they also considered the number of birds and the placement of pigs in addition to the number of pigs and blocks.

All of the previous studies did not focus on generation of fun levels. Their algorithms do not aim at generating character-sequence levels either. On the contrary, our work aims to propose a level generator that can generate fun and interesting levels by expressing quotes or formulas on levels.

PCG has also been applied to other 2D games such as platformer Mario and mazes. Mario level generators [12] make use of objects that are not affected by gravity. Methods for generating mazes (or maze-like levels) for rogue-like games [13-15] do not need to consider the influence of gravity either. However, all objects (blocks) of Angry Birds, except a block called Terrain (described below), are affected by gravity and physics. As a result, in Angry Birds care must be taken to ensure that generated levels are stable.

B. CIG 2016 Angry Birds Level Generation Competition Fun Track

This competition was based on the physics game implementation "Weird Aliens" by Lucas Ferreira using Unity 3D. Unlike the original Angry Birds game having several game objects, the Weird Aliens project for the 2016 competition supported only 15 objects: Bird, Pig, 12 different blocks with 3 materials (wood, ice, stone), and finally Terrain, the only block type which is neither affected by gravity nor destroyed. The image of each object is shown in Fig. 1, and the three materials in Fig. 2.

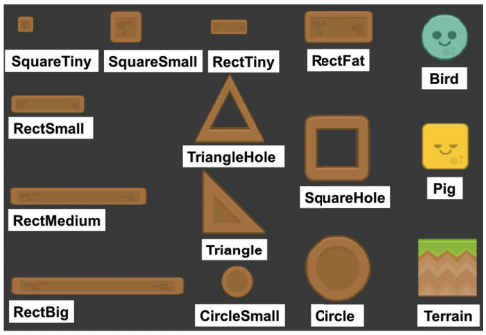

Fig. 1. The graphics of game objects

Fig. 2. Three block materials provided

In addition, the following two rules[1] were employed.

Rule 1: Up to five pairs of material type and block type will be randomly selected, e.g., "stone Triangle". The selected pairs will be forbidden for use in level generation; henceforth this list of them is called the forbidden list.

Rule 2: The minimum (Min) and maximum (Max) number of pigs that a level can contain will be given to the system, and, thereby, all levels generated must contain pigs between the given range.

Besides the above two rules, generated levels should be
- Fun and interesting to play
- Able to be cleared
- Of a large variety of content
- In good use of available space
- Assigned the number of birds suitable to level difficulty
- Initialized by stable structures (no falling structures in the beginning).

III. STRATEGIES AND METHODS

A. Strategies

In order to respond to the aforementioned requirements and rules of the competition, we employ a number of strategies as summarized in Table 1.

For each of the two rules in II-B, our strategies are as follows:

For Rule 1: This rule is coped with by the pattern-struct approach. To generate a character in a quote, this approach uses only one block type and two material types, two types for contrasting each other. To ensure that this approach can always generate a character, each character is provided more than two designs, each using a different block type. This

[1]https://aibirds.org/Level-Generation/cig16-competition-rules.pdf

TABLE I
OUR STRATEGIES BEHIND FUNNY QUOTES

Requirement	Strategy
Fun and interesting to play	Use of character-sequence levels, most of which are funny quotes
Able to be cleared	Use of some heuristics for arrangement of pigs and determination of the number of birds
Large variety of content	Use of 100 pre-defined funny quotes, based on each of which a visually different level can be generated according to the proposed pattern-struct and preset-model approaches, plus additional use of randomly generated formula levels
Good use of available space	Use of double-layered levels and implementation of a number of styles for single-layered levels
Assigned the number of birds suitable to level difficulty	Use of a heuristic that sets the number of birds equal to the number of pigs, accept for that the number of birds is set to two if there are less than two pigs, for quote levels.
Initialized by stable structures	Ensuring of the stability by the proposed pattern-struct and preset-model approaches

is because in the worst case according to this rule, up to five pairs of block and material are forbidden, resulting in at most two blocks that cannot be used in this approach (e.g., if ice-SquareTiny, wood-SquareTiny, stone-RectTiny, wood-RectTiny, wood-SquareSmall are in the forbidden list, SquareTiny and RectTiny cannot be used, but SquareSmall and the rest can be used); in other words, it can be guaranteed that there always exits at least one unforbidden block type for generating a given character.

For Rule 2: In a level of interest, if there are initially more pigs than a desired number, some pigs will be deleted, accordingly, and increased vice versa.

Before giving more details on our system, we note here that all game objects, except Terrain, are affected by gravity, so if a character is constructed without considering the effect of gravity, it will be collapsed as shown in Fig. 3.

B. System Overview

From 100 funny quotes compiled by us, the system first attempts to generate 100 levels, each of which displays a different quote, according to the methods in III-C and III-D. If generation of all 100 levels is successful, five levels will be randomly selected for being replaced by formula levels to increase variation in levels. Figures 4 and 5 show an example of a quote level and a formula level, respectively.

Each quote—and all levels displaying it—was subjectively

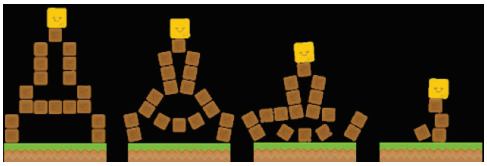

Fig. 3. An example of an unstable character and how it collapses

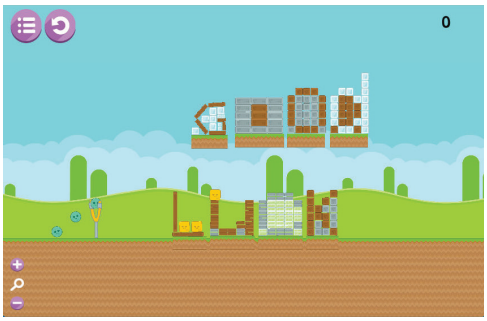

Fig. 4. An example of a quote level

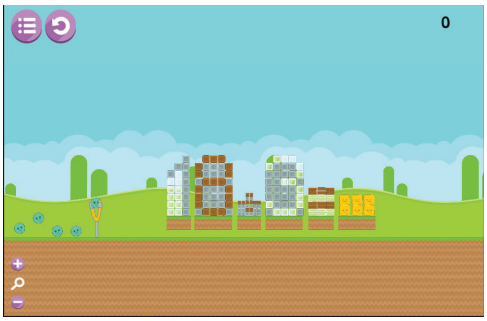

Fig. 5. An example of a formula level

assigned to one of the three difficulty groups: easy, normal, and hard. For example, we considered that a level showing "We will be back" sends a message to the player that the pigs will leave (be destroyed) soon, so this level should be easy to clear and thus be in the easy group, while "Relax bro!" should be in the hard group. After assigning quotes to the easy and hard groups, the rest, such as "Need your Vote", were assigned to the normal group. In summary, there are 33, 32, and 35 levels in our quote database for easy, normal, and hard, respectively.

C. Pattern-struct approach

The pattern-struct approach is used for generating the patterns of the alphabetical letters (both small and capital), numerical numbers, and the symbols in a set of $\{!, ?, +, -, \times, \div, ', =\}$. Each of these characters is provided a number of pattern-struct files. Each file corresponds to a pair of a character design and its block type, selected from four or five block types: SquareTiny, SquareSmall, SquareHole, RectTiny, and RectFat, empirically used in this work for stability. All of these five block types are used for all characters considered in this work, except for SquareHole which is only used for the numbers and letters exclud-

ing $\{B, G, M, N, Q, S, W, a, c, e, f, g, j, m, n, p, q, w, y, z\}$ to avoid low legibility. In the following descriptions, for a given character, only those pattern-struct files whose block type appears in less than two pairs in the forbidden list are considered; in other words, only block types that can be used in this approach are considered.

Figure 6 shows an example of a pattern-struct file for "A" (Left) and a resulting character (Right). In a pattern-struct file, "1" describes the contour of the corresponding character displayed by the block type in the file with a randomly selected unforbidden material type, "2" ensures the character stability by displaying the same block type but using another randomly selected unforbidden material type. In addition, "P" indicates the positions of pigs, whose range from 0 to 5 was empirically pre-determined; after all characters in a level are generated, the number of pigs is adjusted according to the mechanism in III-F.

In order to meet the requirement about "good use of available space", we subjectively divided all quotes into single-layered and double-layered quotes. For example "Good Luck" is defined as a double-layered quote level, from which an example level is shown in Fig. 4. To enhance the diversity of single-layered quote levels, we provide three styles, i.e., Random, Platform, and Wave, as shown in Fig. 7; one of these styles will be randomly selected in generation of single-layered quote levels. In addition, the following five constraints are employed to ensure that a displayed quote does not protrude the game stage's width.

1. A quote cannot consist of more than 20 characters.

2. RectFat cannot be used for any single-layered quotes consisting of 7 characters.

3. For all quotes consisting of 8 to 12 characters, regardless of being displayed in single or double layers, if there exists a layer whose number of characters is more than 7, then SquareTiny can only be used in that layer. In addition, for such quotes, if there exists a layer whose number of characters is more than 6, then RectFat cannot be used in that layer.

4. SquareTiny can only be used for quotes consisting of between 13 and 20 characters.

5. SquareHole can be used up to two times in a quote level.

For a quote level that cannot be generated by any of constraints 2-5, a formula level will be generated.

For each formula level, the number of pigs is set to the result of simple arithmetic calculation expressed by one of the four basic mathematical operations $(+, -, \times, \div)$ of two integer numbers. First, we randomly select an integer between Min and Max as the number of pigs or the result of calculation, and the pigs will always be placed rightward of the equal sign as shown in Fig. 5. Then, an operation is randomly selected, which is followed by a brute-force mechanism to find two integer numbers, the operation between which results in the number of pigs. In a formula level, there is a constraint that SquareHole can be used at most once, but, regardless of this constraint, generation of a formula level by the pattern-struct approach is always successful.

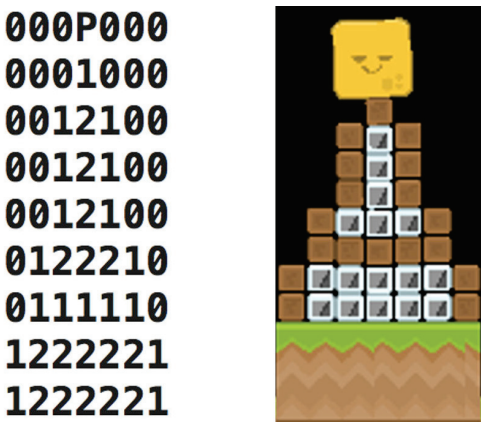

Fig. 6. A pattern-struct file for "A" (Left) and a resulting character (Right)

TABLE II
THE RULES FOR DECIDING THE NUMBER OF PIGS IN A QUOTE LEVEL

Difficulty	Number of Pigs
Easy	Min
Normal	A random integer number in $[Min + 1, Max - 1]$
Hard	Max

D. Preset-model

The preset-model approach is used for increasing the legibility and diversity of levels with predefined stable and highly readable models (Fig. 8). Unlike the pattern-struct approach, multiple block types are used, each with pre-assigned material types, in a preset-model. Only one model is given to each of the capital letters, except for $\{E, F, H, I, T\}$, and the numbers from 1 to 9. As in the previous approach, the positions of pigs, whose number is set to [0, 2], however, in this approach, are fixed in each model.

E. Selection of pattern-struct and preset-model approaches

For each character in a quote, if its preset model exists and the model's block types and their materials are not prohibited, the model will be selected for displaying the character with a 95% probability. This is because such characters are more legible when displayed by preset models. For a formula level, however, to maintain a variety in resulting levels, the preset-model will be selected for each of the two involving numbers, except for "0", with a 30% probability if its block types and their materials are not prohibited. For any character with no preset-model or its preset-model not selected, the pattern-struct approach will be used.

F. Difficulty of levels, arrangement of pigs, and determination of the number of birds

For quote levels, we heuristically design that the number of pigs should increase as the level difficulty increases. Our rules for deciding the number of pigs in a quote level are shown in the Table 2. For a formula level, rather than the number of pigs, we empirically consider a level to be hard

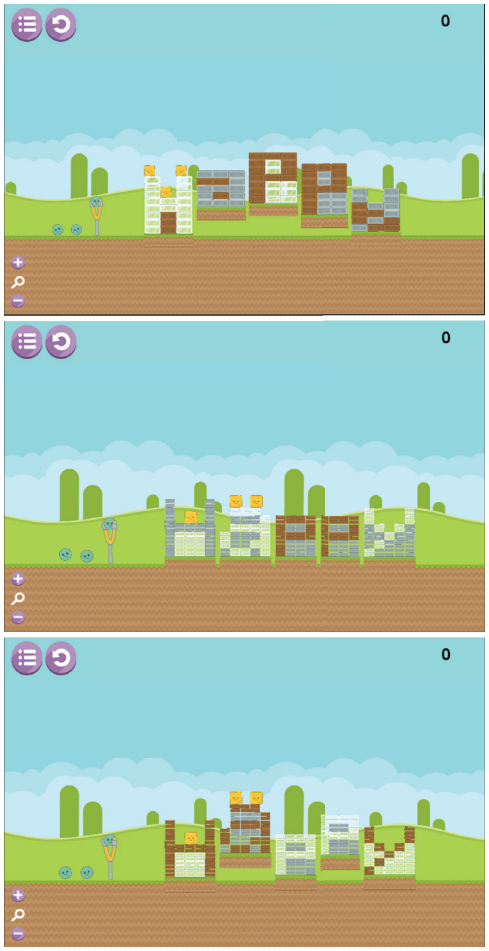

Fig. 7. Random Style (Up), Platform Style (Middle), and Wave Style (Down)

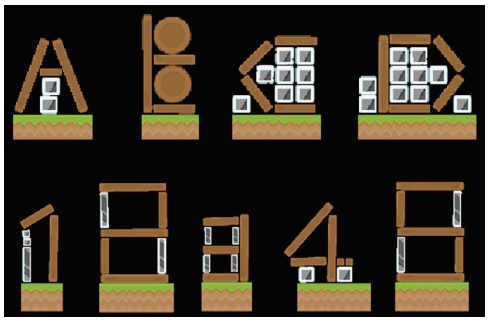

Fig. 8. Examples of preset-models

if SquareHole, which can potentially form large obstacles, is used in generation of at least one of the two involving numbers; otherwise, normal.

If the initial number of pigs in a quote level of interest is more than the number of pigs derived according to Table 2, excessive pigs will be eliminated from right to left in the level. On the contrary, if such number is less, new pigs will be added and placed on two Terrain blocks leftward of the slingshot, as shown in Fig. 9.

Based on our empirical experience, we set the number of

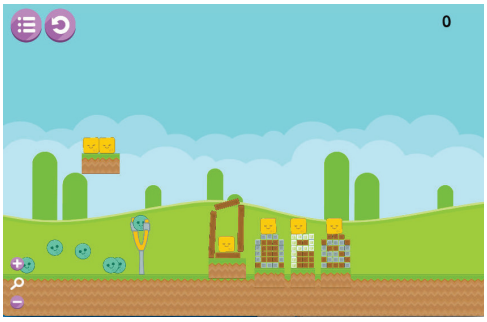

Fig. 9. New pigs added leftward of the slingshot

birds in a quote level to the greater value between two and the number of pigs, while the number of birds in a formula level is from eight to ten for a hard level and from five to eight otherwise.

IV. EXPERIMENT AND RESULTS

We carried out an experiment in order to verify whether resulting levels are legible and not only clearable but have difficulty as designed. In addition, we wanted to find if there were any issues that required improvement. This experiment involved 10 multi-national players, among which eight are male and two are female, having the age range of 21∼25 with the average age of 23.3. Their TOEIC scores were all higher than 500, which means that they have sufficient English capability to recognize English words.

For each player, five quote levels per each difficulty (easy, normal, and hard) were randomly generated by Funny Quotes with Min and Max set to 2 and 6 and all combinations of block and material types being allowed (the forbidden list is null). Then, 15 such levels were presented to the player in random order. Before he or she began to play a level, the player was asked to input all words displayed in the level. For each level, two retries of word input and two retries of play were allowed.

A. Recognition rate

The recognition rate, the ratio of levels whose words could be correctly recognized among all generated levels, is 91.33%. This result was obtained by judging that the words in a given level could be recognized if a player could correctly input them within two retries. Because previous work on Human Interactive Proofs [16] argued that having more than a 90% recognition rate is successful, we consider that words in generated levels by Funny Quotes are sufficiently legible, and have found room for improving the legibility in the following.

There were six levels where “B”, generated by its preset-model, was misidentified as another character or was not considered as a character at all and thereby omitted. For example, in a level shown in Fig. 10, “B” was misidentified by one player—apparently not a Star Wars fan—as “E”. This result indicates that the design of “B” in its preset model must be improved.

Another character worth mentioning is the apostrophe, generated by the pattern-struct approach. There were three levels

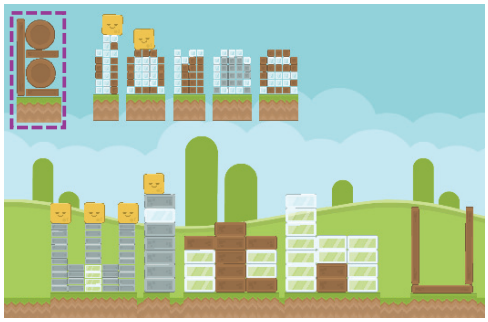

Fig. 10. B force with U

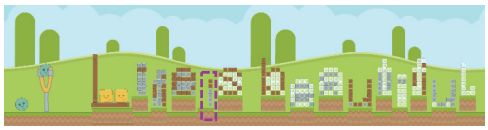

Fig. 11. Life's beautiful

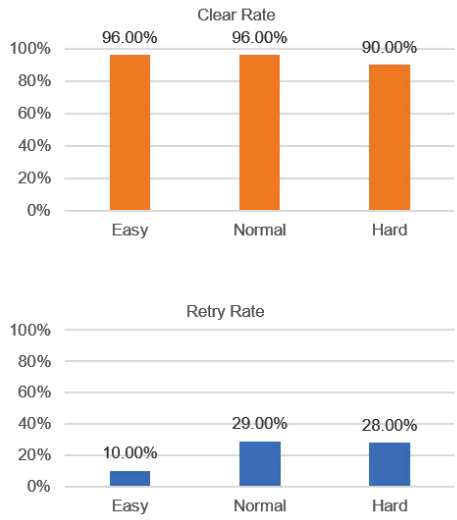

Fig. 12. Clear rate and retry rate

where this symbol was misidentified as "i" (two levels), as shown in Fig. 11, or simply omitted (one level). This result indicates that improvement must be done to the design of this symbol's pattern-struct files.

B. Clear rate and retry rate

The clear rate and the retry rate for each difficulty level are shown in Fig. 12. The former was calculated by dividing the number of levels that could be cleared within two retries by the number of generated levels (50), while the latter was derived by dividing the number of retries by twice the number of generated levels (100). The average clear rate and average retry rate are 94% and 22.3%. These results indicate that Funny Quotes can automatically generate a level of Angry Birds that can be cleared in most cases. However, according to the result of a Friedman test shown in Table 5 (Original), the difference in the clear rate and retry rate among easy, normal and hard levels was not statistically significant; in other words, there was no clear disparity among the three difficulty settings.

V. IMPROVEMENT AND RESULTS

Although there are issues in recognition of some characters as shown in IV-A, we describe in V-A our mechanisms to solve the issue raised in the end of IV-B in particular for quote levels. The results discussed in V-B were obtained using the same players and experiment setting and players as IV.

A. An improved version

The following three mechanisms are proposed and examined.

1. Adjustment in the arrangement of pigs (AP)
2. Adjustment in the determination of the number of birds(AB)
3. Combination of AP and AB (APB)

In AP, first, an adjustment was done to the way excessive pigs in a quote level are removed. In this adjustment, all initial pigs in a quote level are indexed according to their position from left to right, starting from 1. The probability of selecting each of them for removal, in a roulette wheel fashion, is calculated by

$$p(n) = \frac{e^{\frac{n}{D}}}{\sum_{i=1}^{\#pig} e^{\frac{i}{D}}} \tag{1}$$

where $p(n)$ is the probability associated to the nth pig, #pig denotes the number of initial pigs, and D is a distribution parameter.

According to roulette wheel selection based on (1), pigs are removed until the number of pigs in the level becomes equal to the desired number of pigs, $N_{desired}$, derived according Table 2. The parameter D controls the distribution of pigs in a quote level. When D is small, pigs nearer to the right-side are more easily removed, while when D is large, all pigs have a similar probability to be selected for removal. We empirically set D to 1, 2, and 10 for easy, normal, and hard, respectively.

Next, for double-layered quote levels, we also adjusted balance in the number of pigs on the upper layer and that on the lower layer. When N_{up} and N_{low} pigs are initially placed on the upper and the lower layer, respectively, the number of pigs to be removed from the upper and lower layers is calculated using (2) and (3) as follows:

$$N_{upE} = \left\lfloor \frac{N_{up}}{(N_{up} + N_{low})} \times N_{Delete} \right\rfloor \tag{2}$$

$$N_{lowE} = \left\lfloor \frac{N_{low}}{(N_{up} + N_{low})} \times N_{Delete} \right\rfloor \tag{3}$$

where $N_{Delete} = N_{up} + N_{low} - N_{desired}$. When the sum of N_{upE} and N_{lowE} is less than N_{Delete}. One pig will be additionally removed from the layer with more initial pigs.

In AB, our new rules to determine the number of birds in a quote level are shown in Table 3. According to this adjustment, an easy level always has one bird more than pigs, giving the

player more birds per pig to try than in the other difficulty settings, while normal levels have birds less than or equal to pigs and hard levels have least birds in relative to pigs. As a result, this adjustment should widen the gap in difficulty between easy and normal levels and that between normal and hard levels, while making normal and hard levels more challenging than the original mechanism in Funny Quotes.

We conducted another experiment to examine the effectiveness of each proposed mechanism in terms of the clear rate and the retry rate. The same set of players participated in this new experiment. Based on the clear rate of each player in the previous experiment, 10 players were divided into three groups of three, four, and three players, such that there was no significant difference in the clear rate among them (p-value = 0.94 for a Kruskal-Wallis test).

For each group of players, the experimental order based on a reduced Latin square is shown in Table 4. The players were not informed of mechanisms used in generating levels. For each mechanism, every player was asked to play 15 quote levels, five per difficulty setting, presented in a random order.

B. Results

The clear rate and the retry rate of each mechanism are shown in Figs. 13 and 14. We can see that the clear rate of AB, AP, APB decrease as the difficulty setting rises, indicating clearer disparity than the original Funny Quotes among the three difficult settings. As for the retry rate, that of both AB and APB rises as the difficulty setting increases. Table 5 shows the results of a Friedman test on the clear rate and retry rates among the three difficulty settings. These results indicate that there is a statistically significant difference for APB's clear rate and retry rate, and a result, APB is adopted our 2017 version of Funny Quotes.

We also investigated whether those levels generated by APB that could not be cleared by players are actually non-playable. After several attempts, only one normal level and two hard levels could not be cleared by the first author, the best player among the authors. The other levels under this investigation are challenging, but they can be cleared. However, although only a few, the fact that APB generates non-clearable levels remains as an unresolved issue at the time of writing.

VI. CONCLUSIONS AND FUTURE WORK

This paper presented the authors' approach behind Funny Quotes for generating fun levels in the Angry Birds game or its clone games. The first experiment confirmed that generated quote levels are legible and indicated that most of them can be cleared. However, there was no statistically significant difference in difficulty among easy, normal, and hard level settings, which led the authors to conduct an improvement to arrangement of pigs and determination of the number of birds in the original Funny Quote. The effectiveness of this improvement was confirmed in the second experiment.

At the time of writing, in order to not generate non-clearable levels, we are developing a high-performance AI to test the playability of all generated levels in advance and will be submitting this AI to the Angry Birds AI competition. We are also improving the pattern-struct and preset-model approaches to raise the recognition rate of words present in quote levels. In future, we will study mechanisms to select quotes according to the player's current playing performance and/or emotional status.

TABLE III
NEW RULES FOR DETERMINING THE NUMBER OF BIRDS

Number of pigs	Number of birds
$\#pig \leq (Min + Max)/2$	$Min + 1$
$\#pig > (Min + Max)/2$	$(Min + Max)/2$

TABLE IV
EXPERIMENTAL ORDER

Group/Order	1	2	3
1	APB	AP	AB
2	AP	AB	APB
3	AB	APB	AP

TABLE V
RESULTS OF A FRIEDMAN TEST ON EACH MECHANISM

Version	p-value for the clear rate	p-value for the retry rate
Original	0.5092	0.2176
AP	0.1225	0.5916
AB	0.1028	0.0024
APB	0.0051	0.0007

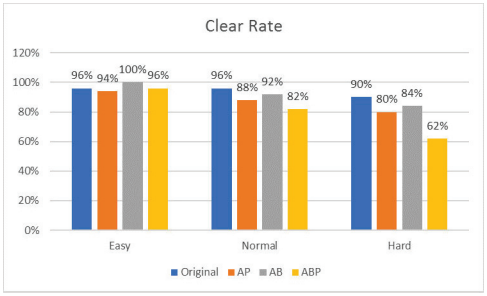

Fig. 13. Clear rate of each mechanism

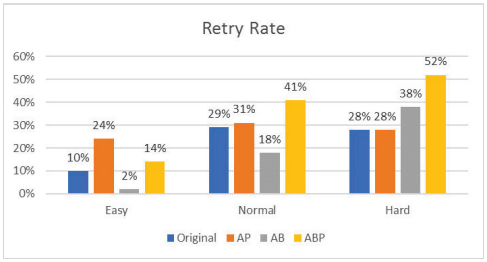

Fig. 14. Retry rate of each mechanism

ACKNOWLEDGMENT

We thank Prof. Jochen Renz, A, Prof. Julian Togelius, Lucas N. Fereira , Matthew Stephenson, and XiaoYu (Gary)

Ge for organizing the Fun Track of the Angry Birds Level Generation Competition in IEEE Conference on Computational Intelligence and Games 2016, and for the very nice winning certificate. We also thank all players who voluntarily participated in our experiments.

REFERENCES

[1] N. Shaker, J. Togelius, and Mark J. Nelson, Procedural Content Generation in Games: A textbook and an overview of current research, Springer, 2016.

[2] L. Ferreira, C. Toledo, "A Search-based Approach for Generating Angry Birds Levels," Proc. of the 2014 IEEE Conference on Computational Intelligence and Games (CIG 2014), pp. 1-8, 2014.

[3] L. Ferreira, C. Toledo, "Generating Levels for Physics-based Puzzle Games with Estimation of Distribution Algorithms," Proc. of the 11th Conference on Advances in Computer Entertainment Technology (ACE 2014), Article No. 25 (6 pages), 2014.

[4] L.T. Pereira, C. Toledo, L.N. Ferreira, L.H. Lelis, "Learning to Speed Up Evolutionary Content Generation in Physics-based Puzzle Games," Proc. of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2016), pp. 901-907, 2016.

[5] M. Kaidan, CY. Chu, T. Harada, and R. Thawonmas, "Procedural Generation of Angry Birds Levels That Adapt to the Player's Skills Using Genetic Algorithm," Proc. of the 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE 2015), pp. 535-536, 2015.

[6] M. Kaidan, T. Harada, CY. Chu and R. Thawonmas, "Procedural Generation of Angry Birds Levels with Adjustable Difficulty," Proc. of the 2016 IEEE Congress on Evolutionary Computation (CEC 2016), pp. 1311-1316, 2016.

[7] YX. Jiang, M. Kaidan, CY. Chu, T. Harada and R. Thawonmas, "Procedural Generation of Angry Birds Levels using Building Constructive Grammar with Chinese-Style and/or Japanese-Style Models," Proc. of ASIAGRAPH 2016, pp. 53-54, 2016.

[8] M. Stephenson, J. Renz, "Procedural Generation of Complex Stable Structures for Angry Birds Levels," Proc. of the 2016 IEEE Conference on Computational Intelligence and Games (CIG 2016), pp.178-185, 2016.

[9] M. Stephenson, J. Renz, "Procedural Generation of Levels for Angry Birds Style Physics Games," Proc. of Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2016). 7 pages, 2016.

[10] D. E. Goldberg, Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[11] M. Hauschild, M. Pelikan, "An introduction and survey of estimation of distribution algorithms," Swarm and Evolutionary Computation, 1(3), pp. 111-128, 2011.

[12] N. Shaker, J. Togelius, G. N. Yannakakis, B. Weber, T. Shimizu, T. Hashiyama, N. Sorenson, P. Pasquier, P. Mawhorter, G. Takahashi, G. Smith and R. Baumgarten, "The 2010 Mario AI Championship: Level Generation Track," IEEE Transactions on Computational Intelligence and AI in Games, 3(4), pp. 332-347, 2011.

[13] C. McGuinness and D. Ashlock, "Decomposing the level generation problem with tiles," Proc. of IEEE Congress on Evolutionary Computation (CEC 2011), pp. 849856, 2011.

[14] C. McGuinness and D. Ashlock, "Incorporating required structure into tiles," Proc. of the 2011 IEEE Conference on Computational Intelligence and Games (CIG 2011), pp. 16-23, 2011.

[15] V. Valtchanov and J. A. Brown, "Evolving dungeon crawler levels with relative placement," Proc. of the 5th International Conference on Computer Science and Software Engineering (C3S2E-12), pp. 27-35, 2012.

[16] K. Chellapilla, K. Larson, P. Simard and M. Czerwinski, "Designing human friendly human interaction proofs (HIPs)," Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2005), pp. 711-720, 2005.