Procedural Generation of Angry Birds Levels with Adjustable Difficulty

Misaki Kaidan, Tomohiro Harada, Chun Yin Chu and Ruck Thawonmas
Intelligent Computer Entertainment Laboratory, Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan

***Abstract*—This paper proposes a procedural generation algorithm that creates game levels for the *Angry Birds* game. Game levels are automatically generated using a genetic algorithm (GA), and their difficulty is adjusted by a parameter introduced in the fitness function of GA. In addition, the Extended Rectangle Algebra (ERA) is used in order to analyze these levels. By calculating ERA relations between objects, effects of a bird hitting on an object are quantified by the aforementioned parameter. Our experiment proves that this parameter has a strong correlation of 96% with the players' winning percentage. Accordingly, it shows that any difficulty level can be generated by regulating the aforementioned parameter.**

Keywords— Genetic Algorithm; Extended Rectangle Algebra; Angry Birds; Levels; Adjustable Difficulty

I. Introduction

Recently, the scale and complexity of video game development have soared drastically, and players' expectation for more quality contents has never been higher[1]. As such, the cost and workload of game development have skyrocketed to a worrying level. To mitigate this problem, the industry is now focusing on procedural content generation (PCG), a technology that can create game content automatically instead of manually. Generating game levels procedurally (procedural level generation, PLG) is the most common type of PCG. Generating in-game dungeons on the fly can be seen as early as 1980, when *Rogue* hit the market [2]. In recent years, PLG is widely used in game industry, and has become a research subject for computer scientists.

There are many recent studies in how evolutionary algorithms (EA) [3] can be applied in PLG of different types of games. For instance, researchers have studied procedural generation of tracks in a racing game [4], levels of a platformer game [5] and maps in *StarCraft* [6]. Moreover, in the Mario AI Championship, held between 2009 and 2012, a Level Generation Track was organized to entertain participants who attempted to create Mario levels using PLG techniques [7].

Many researchers have proposed EA such as genetic algorithm (GA) to be used in PLG. Among researchers who suggested using GA for PLG, Ferreira and Toledo published a research paper on procedural level generation in *Angry Birds* [8], an action puzzle game. While procedurally generating *Angry Birds* levels was proven to be plausible, in their study there is no way to limit the number of pigs in the level, indicating that adjusting difficulty of a generated level is impossible.

In this work, we propose a new approach, for generating *Angry Birds* levels, that can adjust the level difficulty by controlling h_{level}, a parameter in the fitness function of GA. The Extended rectangle Algebra (ERA) [10] is used to analyze the structure of *Angry Birds* levels. ERA is a set of qualitative spatial calculus for representing and analyzing structures in 2D space. It can assess the relations among all objects in an *Angry Birds* level, and such relations are used to quantify the influence of each object on other objects into a heuristic value. The aforementioned h_{level} represents the difficulty of a level and is derived using the heuristic value of each object in the level. The effectiveness of our method is successfully testified in our experiment.

II. Related Work

A. Angry Birds

Angry Birds [9] is a mobile game released by Rovio Entertainment in 2009. It is an action puzzle game that features a slingshot as its core mechanics. A screenshot of the *Angry Birds* game is shown in Fig. 1. The goal of the game is to defeat all green pigs in a given level by shooting red birds with the slingshot. A level in *Angry Birds* is formed by a slingshot, birds, pigs and blocks and is thus defined by the number of birds, as well as the numbers and positions of the pigs and blocks. In this work, we adopted an *Angry Birds* clone[1], developed and made publicly available by Ferreira.

B. Genetic Algorithm

As done in Ferreira and Toledo [8], in our work, the difficulty of each generated level is adjusted by GA. GA is an optimization algorithm inspired by the concepts of biological evolution (selection, mutation). In GA, a fitness function is defined to evaluate each individual, which is selected or eliminated according to its fitness value. Furthermore, crossover and mutation are repeated so as to maximize or minimize the fitness value of all individuals.

C. Procedural Level Generation in Angry Birds Using GA

In their work [8], Ferreira and Toledo proposed procedural generation of *Angry Birds* level using GA for minimizing the fitness function in (1). In the chromosome of each *Angry Birds* level, every column in the level is defined as a stack. Each element of a stack is a pre-defined game object. Each

1. https://github.com/lucasnfe/AngryBirdsCover (Last accessed on January 28, 2016)

individual's chromosome is formed by multiple columns of stacks, as illustrated in Fig. 1. The number of columns and the height of each column are determined randomly, within the range of the game screen. The distance between consecutive columns is decided randomly. In column 3 of Fig.1, the 32nd object, the 4th object, the 13nd object and the 4th object are formed in that order from the bottom.

In the existing research, the simulation was executed to evaluate the stability of the level when evaluating fitness. In the simulation, the fitness was evaluated after the speed of all objects in the level are 0.

Crossover (Fig. 2) and mutation (Fig. 3) are performed in the following manner:

- Crossover: For two individuals p_1 and p_2, with m and n columns, respectively, the i^{th} columns of both individuals are exchanged in a pre-defined crossover probability. If p_1 has more columns than p_2, the last $m - n$ columns from p_1 will have a 50% chances to be copied into the new child. Similarly, if p_2 has more columns than p_1, the last $n - m$ columns from p_2 will have a 50% chances to be copied into the new child.
- Mutation: Mutation is performed after crossover. Elements of a column is re-generated according to the mutation probability.
- Selection: Parents are selected by tournaments, a fixed number of individuals are selected from the population randomly, and the individual with the smallest fitness value is selected as a parent of the next generation.

Ferreira and Toledo's formula for the fitness function is

$$\min f_{ind} = \frac{1}{n}\sum_{i=0}^{n-1} v_i + \frac{\sqrt{(b-B)^2}}{Max_b - B} + \frac{1}{1+p}, \qquad (1)$$

where v_i ($0 \leqq v_i \leqq 1$) is the average velocity of element i, n is the sum of the number of blocks b and that of pigs p.

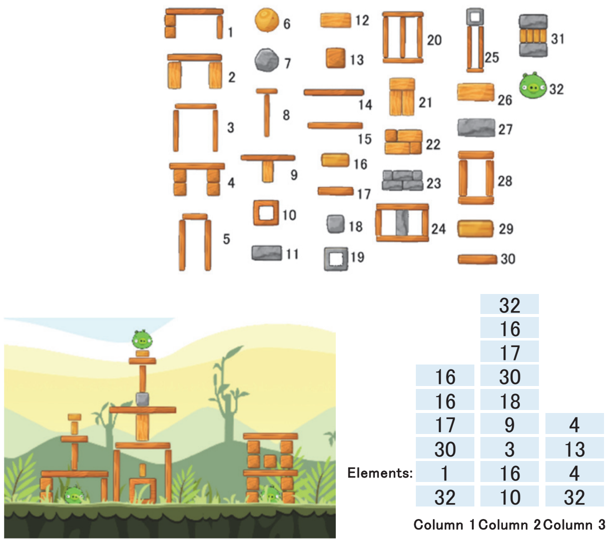

Fig. 1. A level in Angry Birds and its chromosome representation

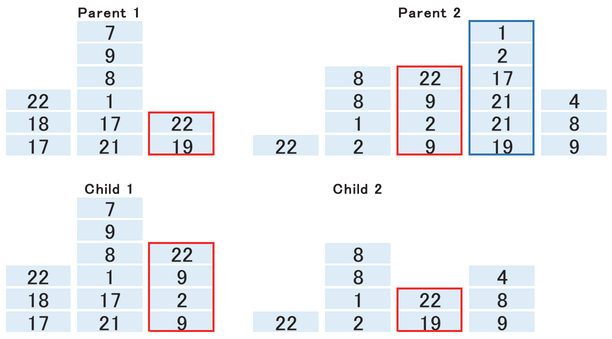

Fig. 2. An example of a crossover operation

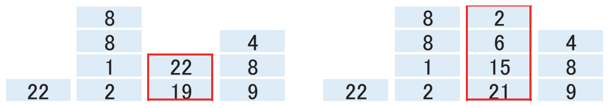

Fig. 3. An example of a mutation operation - before (left) and after (right)

B and Max_b are the ideal number of blocks and the maximum number of blocks, respectively. The first term of formula (1) evaluates the stability of the level using the average velocity of blocks and pigs. The second term normalizes the difference between the ideal number of blocks B and the actual number of blocks b. The third term ensures the number of pigs is never zero.

Their method succeeded in procedurally generating *Angry Birds* levels, but due to the third term of the formula, more pigs lead to a lower fitness value, and therefore levels with too many pigs may be generated. Having too many pigs in a level is not desirable, as the player has to destroy all pigs in order to clear the level; in other words, having too many pigs means the level may be not possible to clear. The difficulty of level are adjusted by regulating the number of blocks B in formula (1) however it is difficult to adjust difficulty because the number of blocks B has no relevance to difficulty.

III. Angry Birds Structure Analysis using Extended Rectangle Algebra

A. *Extended Interval Algebra and Extended Rectangle Algebra*

The Extended Interval Algebra (EIA) [10] is an extended version of Allen's Temporal Interval Algebra [11], for which Zhang and Renz have expanded the number of relations from 13 to 27. EIA was further developed into ERA that is used for analyzing 2D space. In order to apply ERA relations to *Angry Birds* objects, access to the Minimum Bounding Rectangle (MBR) of every object is needed. An MBR represents the smallest rectangle that can bound an object of interest. Using the coordinates of the vertex and pivot of the object's MBR, ERA relations between two objects can be represented.

A supporter of an object (Fig. 4) can be identified using ERA relations. In this paper, a supporter that directly supports the target object is called direct supporter. Likewise, an object being supported, called supportee, can be found reversely through the supporter relations between objects.

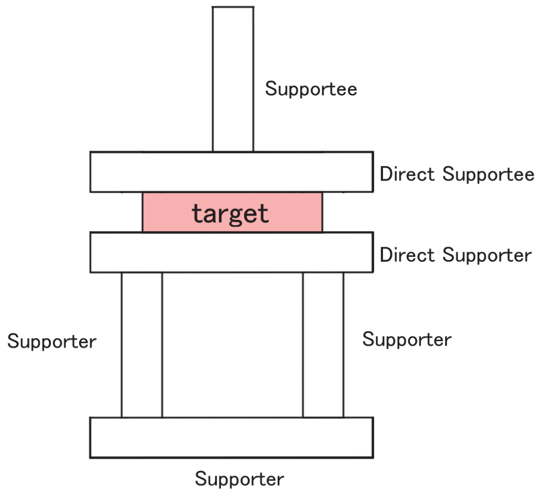

Fig. 4. An example of supporters of a target

B. Evaluation of the Effects of a Shot Using ERA Relations

To predict an object's effects exerted on the level structure when it is hit by a bird, Zhang and Renz [10] listed out eight possible cases that may occur when a bird hit the structure, and in each case, all affected objects are recorded in a list called Affected List. The Affected List is used for calculating the heuristic value of each object, and such a value predicts effects on other objects when the object is hit by a bird.

In this paper, we adopt the following four cases, Case 1-4, in [10], while we do not adopt. Case 5-8 that deal with blocks not placed vertically or horizontally. This is because blocks in Angry Birds clone are placed vertically or horizontally. In this paper, we adopted the following four cases that deal with blocks placed vertically or horizontally because such placements are only viable in the Angry Birds clone in use:

- Case 1: When an object *x* (including a bird) hits another object *y*
- Case 2: When a bird hits object *x* that is a supportee of object *y*, and the bird causes *x* to collapse
- Case 3: When a bird hits object *x* that is a supporter of object *y*, and the bird causes *x* to collapse
- Case 4: When a bird hits object *x* that is a supporter of object *y*, and the bird destroys *x* (Fig. 6(b))

In the following, *o.life*, *o.height* and *o.width* refers to the life points, height and width of object *o*, respectively. Effects of a bird hitting the structure in each of the four cases below are shown in Tables 1-4, respectively.TABLE I shows objects that are affected in Case 1. The affected objects vary depemds on life or the ratio of height to width of object *y*.

TABLE II shows objects that are affected in Case 2. The affected objects vary depends on whether the ratio of height to width of object y is greater than a certain threshold (Fig. 5(a)) or not (Fig. 5(b)).

TABLE III shows objects that are affected in Case 3. The affected objects vary depends on whether the object *y* applies to Rules 1.1-1.4 in [10] after object *x* collapses or not (Fig. 6(a)).

TABLE IV shows objects that are affected in Case 4 (Fig. 6(b)).

TABLE I. CASE 1

Resulting condition of *y*	**Objects included in the Affected List of *x***
y.life≦8 ⇒ *y* is destroyed	• *y* • all direct supportees of *y*
y.life > 8 && (*y*.height/*y*.width)≧1.5 ⇒ *y* collapses	• *y* • all direct supporters of *y* • all direct supportees of *y*
y.life > 8 && (*y*.height/*y*.width) < 1.5 ⇒ *y* remains stationary	• all direct supporters of *y*

TABLE II. CASE 2

Resulting condition of *y*	**Objects included in the Affected List of *x***
(*y.height/y.width*)≧2 ⇒ *y* collapses (Fig. 5(a))	• *Y* • all supportees of *y*
(*y.height/y.width*) < 2 ⇒ *y* remains stationary (Fig. 5(b))	• all direct supporters of *y*

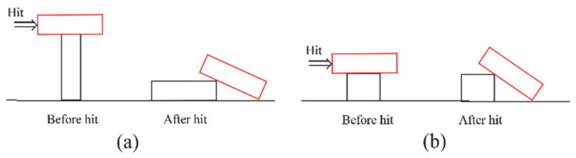

Fig. 5. Illustration of Case 2 (created based on Fig. 8 in [10] and shown here for this paper to be self-contained)

TABLE III. CASE 3

Resulting condition of *y*	**Objects included in the Affected List of *x***
y still fulfills Rules 1.1-1.4 in [10] after the collapse of *x* (*y* remains stationary)	• all direct supporters of *y*
otherwise (*y* is not stationary as shown in Fig 6(a))	• *y* • all supporters of *y* • all supportees of *y*

TABLE IV. CASE 4

Resulting condition of *y*	Objects included in the Affected List of *x*
/	• *y* • all supporters of *y* • all supportees of *y*

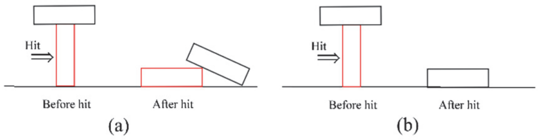

Fig. 6. Illustration of Case 3 (a) and Case 4 (b) (created based on Fig. 9 in [10] and shown here for this paper to be self-contained)

IV. PROPOSED METHOD

In our proposed method, the heuristic value of each object is calculated based on the object's Affected List. It is not realistic to generate levels according to evaluation by using AI because the evaluation by AI takes a long time to complete. Therefore, this research explores a method to generate a certain difficulty of levels controlled by analyzing structure of levels with ERA without AI. A target object is selected from all reachable objects and heuristic value is evaluated from objects in Affected List. The heuristic value is added to h_{level} that reversely approximates the level difficulty. Furthermore, the third term of formula (1) is changed so as to adjust the difficulty of the level.

A. Calculation of an Object's Impact factor

An impact factor that quantifies the effects of each object when hit by a bird is calculated as follow. First let us denote the total number of blocks and total number of pigs by b and p, respectively. In order to calculate the impact factor of object x, all objects that are affected when x is hit by a bird, i.e., those objects in x's Affected List, are considered. For the i^{th} object in x's Affected List, o_i:

- If object x or o_i is a supporter of another object or a pig shelter, $1/b$ is added to x's impact factor.
- If object x or o_i is a pig, $19/p$ is added to x's impact factor.

The adding amount in each of the above two cases was empirically set so as to limit the maximum impact factor of each object to 20.

B. Evaluation of a Level Using Impact factors

The aforementioned impact factor of each object is used in the following procedure for evaluating an Angry Birds level. In this procedure, a generated level is evaluated by using ERA where the highest heuristic value among all reachable objects is added to h_{level}. This is because a smart player would first aim the reachable object that has the highest heuristic value.

First, all objects (both blocks and pigs) are put into a tentative list. Next, among those that can be directly hit by a bird, let x_1 be the object that has the highest impact factor, and h_1 its impact factor. All objects on x_1's Affected List are removed from the tentative list. Then, among the remaining objects in the tentative list, object x_2 is selected that can be hit by a bird directly and has the highest impact factor, and its impact factor as h_2 is recorded. As done for x_1, all objects on x_2's Affected List are removed from the tentative list. This procedure is iterated for a_{init} (the initial number of birds) times or until all pigs have been removed from the tentative list. Afterwards, h_{level} is calculated as follows:

$$h_{level} = \frac{a_{init}}{X} \times \frac{1}{X} \sum_{i=1}^{X} h_i , \tag{2}$$

where X is the number of iterations, where $1 \leqq X \leqq a_{init}$, and h_i is the impact factor of x_i. The sooner all pigs are removed from the tentative list, i.e., fewer birds are needed to destroy all pigs, the value of a_{init}/X will be larger. The value of h_{level} will thus be large if the impact factor of each involving object is large and few birds are needed to destroy all pigs in the level.

C. Improvement of the Fitness Function

The third term of formula (1) is modified so as to adjust the level difficulty. The updated fitness function formula is depicted by formula (3).

$$\min f^{new} = \frac{1}{n}\sum_{i=0}^{n-1} v_i + \frac{\sqrt{(b-B)^2}}{Max_b - B} + \tanh\left(\frac{|H - h_{level}|}{30}\right) \tag{3}$$

In this formula, h_{level} represents the impact factor of the level calculated with formula (2). H is the ideal value of h_{level}, specified by the level designer. The third term of formula (3) normalizes the difference between the ideal impact factor and the actual one. The denominator 30 is determined empirically, by picking the middle value between H_{max}=60 (the maximum value of h_{level} for a_{init} = 3, a typical number of birds) and $H_{\min}$=1. Figures 7 and 8 show a level generated at $H_{\min}$=1 and another level generated at H_{max}=60, respectively.

V. EXPERIMENT

To test the correlation of H and difficulty, an *Angry Birds* procedural-level-generation experiment was performed with 20 participants (18 males, 2 females, between 21and 26 years old).

A. Level Pools

In this experiment, level pools, each of a pre-defined H value, were generated beforehand. In particular, level pools for H=6, 16, ..., 56 were generated, respectively, each level pool having 100 levels.

B. Settings

Parameters were set to the following values in the experiment.

- Number of birds: 3
- Maximum number of generations: 1000
- Population size: 20
- Crossover probability: 0.95
- Mutation probability: 0.05

- Elitism enabled
- Number of individuals in each tournament: 2
- Max_b =100
- B=50
- Terminal condition of the evolution: evolution is terminated when either of the following conditions is fulfilled

  ①The number of generations reaches 1000

  ②The fitness of the best individual remains the same for 10 consecutive generations

Those parameters are the same as [7]. Several settings of B are tested in the previous research, however, we used 50 as the parameter B that is the half of Max_b.

C. Procedure

Six level pools were prepared for H=6, 16, ... , 56. Each participant played a total of 18 levels, with three levels randomly selected from each level pool, in a random sequence. The value of H was then compared with the players' winning percentage.

D. Results

Figure 7 depicts the average winning percentage of all participants for each level of H. The value of H and the winning percentage had a correlation coefficient of 0.96. Both Fig. 7 and the high correlation coefficient demonstrate that there is a strong positive correlation between the value of H and the winning percentage. In particular, it is indicated that the higher H generates a level with the higher winning percentage, i.e., an easy level, while the lower H generates a level with the lower winning percentage, i.e., a difficult level.

To further discuss on generated levels with the proposed method, Figs. 8-10 show example levels generated with H=1, 30, and 60, where H=1 is the minimum setting of H, while H=60 is the maximum setting of H. From Fig. 8, a level generated by low H value has more pigs and complex structure, which makes difficult to clear this level. While from Fig. 10, a level generated by a higher H value has fewer pigs and simple structure, which is easily cleared by destroying objects under the pig. Figure 9 shows the medium difficulty of level that has a few pigs obstructed with blocks.

Those finding implies that level designers or game developers who use the proposed method can easily adjust the difficulty of generated levels by tweaking the value of H.

VI. Conclusions

This paper proposed the use of GA and ERA for PLG in *Angry Birds*. The proposed method is an improvement over an existing method based also on GA, as the proposed method allows its users to adjust the difficulty of generated levels. The effectiveness of the proposed method was confirmed in the presented experiment. In our future study, we strike to assess and adjust not only the difficulty, but also the visual design of generated levels.

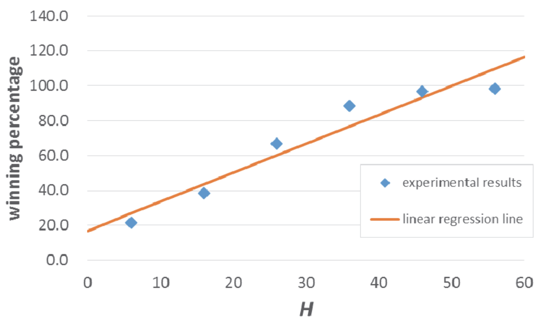

Fig. 7. The correlation between H and the winning percentag

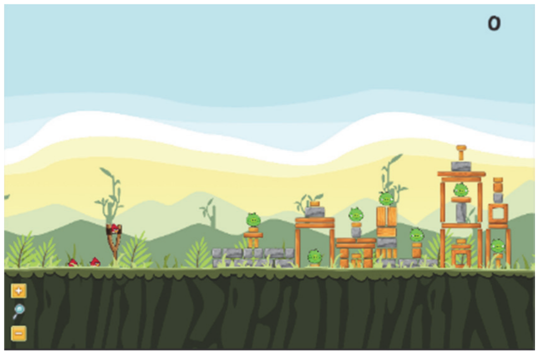

Fig. 8. An example level generated at $H_{\min}$=1

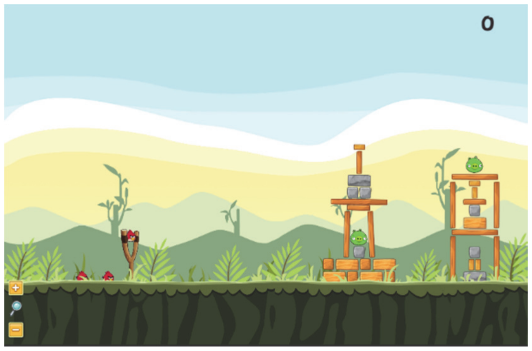

Fig. 9. An example level generated at H=30

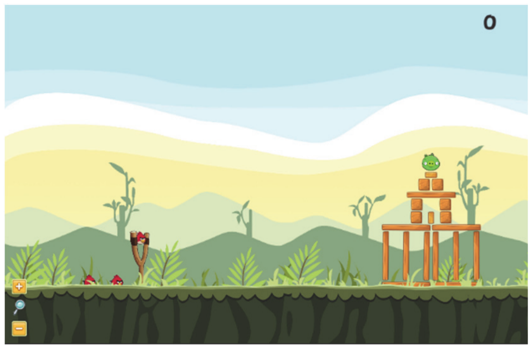

Fig. 10. An example level generated at H_{max}=60

Acknoledgement

The authors wish to thank Zhang and Renz (2014) for sending us the rule-part code of their work. This work was supported in part by Grant-in-Aid for Scientific Research (B), Number 15H02939, JSPS.

References

[1] Yo Takatsuki. “Cost headache for game developers”. http://news.bbc.co.uk/2/hi/business/7151961.stm. 2007. [Online; last accessed March 31, 2016]

[2] Glenn R. Wichman. A Brief History of "Rogue". http://www.wichman.org/roguehistory.html, 1997. [Online; last accessed January 28, 2016].

[3] John Henry Holland. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, 1992.

[4] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi, “Interactive Evolution for the Procedural Generation of Tracks in a High-End Racing Game,” Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 395-402, 2011.

[5] Fausto Mourato, Manuel Próspero dos Santos, and Fernando Birra, “Automatic level generation for platform videogames using genetic algorithms,” Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, Article No. 8, 2011.

[6] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelback, Georgios N. Yannakakis, and Corrado Grappiolo, “Controllable procedural map generation via multiobjective evolution,” Genetic Programming and Evolvable Machines, 14(2), pp. 245-277, 2013.

[7] Julian Togelius, Noor Shaker, Sergey Karakovskiy, and Georgios N. Yannakakis, “The Mario AI Championship 2009-2012,” AI Magazine 34(3), pp. 89-92, 2013.

[8] Lucas Ferreira and Claudio Toledo, “A Search-based Approach for Generating Angry Birds Levels,” Proceedings of 2014 IEEE Conference on Computational Intelligence and Games, pp. 1-8, 2014.

[9] Rovio Entertainment. “AngryBirds.com - The Official Home of Angry Birds". (2009) https://www.angrybirds.com. Last Accessed on January 28, 2016.

[10] Peng Zhang and Jochen Renz, “Qualitative Spatial Representation and Reasoning in Angry Birds: The Extended Rectangle Algebra,” Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning, pp. 378-387, 2014.

[11] James F. Allen. “Maintaining Knowledge about Temporal Intervals,” Communications of the ACM, 26(11), pp. 832-843, 1983.