Procedural generation of levels for the angry birds videogame using evolutionary computation

Jaime Salinas-Hernández
Department of Graduate Studies
Tijuana Institute of Technology
Tijuana, México
salinas.jaime1217@gmail.com

Mario Garcia-Valdez
Department of Graduate Studies
Tijuana Institute of Technology
Tijuana, México
mario@tectijuana.edu.mx

JJ Merelo
Department of Graduate Studies
Universidad de Granada
Granada, Spain
jmerelo@ugr.es

Abstract—In this paper an evolutionary computation-based system is proposed to generate levels for the Angry Birds video game, the levels themselves are composed of a set of birds that the player can use, a certain amount of pigs that must be destroyed in order to progress to the next level and a given number of pieces that conform structures that may or may not protect the pigs from the player. The focus in this stage is the generation of diverse structures that can be played in the game having the additional characteristics of being interesting and playable. We propose a multi-layered search, first constructing composite structures from basic components, to then build more complex structures building from these components, following an open-ended evolution approach; in which the evolution of structures is not guided towards a single objective but rather is free to evolve and generate novelty or diversity. The objective is then that the proposed levels are evaluated by how complex they become and how different they are from the rest of the group. The experiments conducted show that a evolutionary approach allows the generated levels to have a level of uniqueness because on most of the cases the way they are presented on screen is a way a normal human would have trouble to create but at the same time are interesting to see.

Index Terms—genetic algorithm, procedural generated content, open-ended evolution, evolutionary computation

I. INTRODUCTION

Procedural content generation is an interesting subject on the entertainment industry, primarily in video game development because of the savings in time and human resources when generating content for different video game projects [1], [2]. In this paper an evolutionary content generation system is proposed to generate levels for the popular Angry Birds video game, taking inspiration in concepts of open-ended evolution. Angry birds is a game developed by Rovio Entertainment in which the objective is to destroy structures placed on the map as well as eliminating green pig-like entities in order to obtain the highest score possible, the game uses a gravity and collition system that allows the structures to fall and be destroyed by means of damage as well as the use of three different construction materials *ice*, *wood* and *stone* each one with a different resistance, ranging from fragile, normal and sturdy in that order. The game mechanic is to use a slingshot to shoot a certain number of birds with different abilities, to destroy both the structures and the objectives. A high score is achived by doing the most destruction with the least birds possible [3].

The main objective of this project is to generate levels that can be played by naïve or inteligent agents as well as physical players. The levels themselves need to be complex in the sense that interesting structures can be created, but at the same time, it is possible to complete in some way [4], [5].

Procedural content generation provides a peculiar case of study because of the way content has to be made for a each particular type of game. In this use case we have those restrictions dictated by the game, and also those imposed by a competition track [6].provide an interesting The challenge for the participating algorithms is to be flexible on the pieces that can be used and the runing time of the algorithm, as they are determined at runtime. In order to tackle this case study, we proposed the use a genetic algorithm capable of creating game levels having the objective of being both *interesting* and *playable*, the way to test our candidate solutions will be by using a open-source software called Science Birds [7] which uses the same resources used on the original angry birds game.

In this work we found that an evolutionary algorithm can be used to create this type of content not only because it is able to create levels that are interesting and diverse but because an evolutionary algorithm is capable of creating things a normal human could have trouble imagining of.

We propose a basic Open-Ended Evolution (OEE) algorithm, a variant of genetic evolution algorithm where the objective differs from that of a regular genetic algorithm, in this case the final objective is not to reach a final state of the evolution (an optimal solution) but to continue evolving and creating new types of entities, perhaps more complex on each new generation. Solutuion entities have the possibility to evolve not only inside the frame of their type, but they can create entirely new groups [8]. Taylor et al. [9], [10] defines OEE as a system capable of creating novelty beyond a point where nothing changes on the current group of interest rather than just converging to a stable or quasi-stable state.For this project, OEE is defined simply as a genetic evolution process with different clasess of objects, with compositional and taxonomic relationships between them. Objects of each class can evolve in parallel or as in this case, independently of each other. The idea is based on the object oriented model, in which complex behavior can be model using these principles. The rest of the paper is as follows.

Section 2, contains a quick overview of some aspects used by different authors in order to tackle this problem, their design and basic functionality, in Section 3 the description of the current state of the project is explained, from the way the candidate solutions are represented, to the way the mutations are applied to them, as well as the way we use our fitness fucntion to determine the playability and complexity of the generated levels, finally in Section 4 we provide our conclusions about the development of this paper as well as improvements that can help to refine the proposed work.

II. STATE OF THE ART

In this Section we present PCG works relevant to the Level generation for the Angry Birds video game. Renz et al. [6], [11] organized an AI competition with the aim of developing intelligent agents that could play new Angry Birds levels better than the best human players. The competition was then collocated in other AI conferences. In order to train and test these agents, more novel and some times difficult levels must be generated. A related competition was then proposed by Stephenson et al. [4] this time with the goal of developing a level generator for the game. Several generators have been proposed in the following years of competition. M. Kaidan et al. [12] propose a model in which the generation of content is controlled by the skill of a player while actively playing the game, the moment the player finishes a certain level the algorithm obtains the score and generates the next level to be played. Y. Jiang et al. [13] proposed the use of predefined letter style patterns and combined to create a small pool of words and letters used to be combined in order to create levels with text on them, the participants would play the game and after that they would be asked if they liked the layout of the levels. A search based approach has been also used in several works. Laura Calle [14] proposed a method in which the pieces in the game are arranged in groups using the gravity of the game to create structures that are later evolved into more complex levels, this method was used as a base to work with since it used a modified version of the angry birds game that was able to obtain the data of a level after a simulation.

III. PROPOSED METHOD

In this Section we describe the approach used in this work, explaining first, the restrictions of the game. The game contains a total of 11 block shapes as shown in Figure 1. These basic blocks cannot be modified, it is not possible to add more or modify the existing ones, with this in mind the purpose is to create composites of these blocks in different locations and angles in order to create structures that can be used as building blocks for more complex structures.

A. Composite Generation

Since there is no way to *add* a new composite to the pool of blocks, then a way to combine both is needed, in order to do this first the original 11 pieces of the game will be converted into composites of a single element. A composite is defined as a lists of blocks, then a *composite* of one element will

be created for each piece. Composites are the first layer the building blocks for the structures to come. A composite could include different types of blocks, rotated at a certain angle and compose a basic structure. An ideal composite could support other composites on top with out falling apart, but this is not required. A composite is measured using maginary bounding box, defined by calculating the lengths of all the pieces, finding the borders of said pieces and then finding the points that could define a box around all of them, then use this defined box to calculate the height and width of the new composite and then calculate from the center point of said box the offsets from the center of the group to the center of each respective piece as shown on figure 3, this information is needed in order to add it as a new composite so it can be used to place the pieces in their respective new positions when the files needed for the simulation are being created. This is similar to creating a group of elements in a drawing application.

Good composites can be found using an evolutionary approach proposed earlier by Calle and Merelo, where basic structures are generated by searching a space of falling blocks and selecting those that remain standing. This algorithm is not capable of creating complex strucures, but it can generate very novel basic structures. We can see that the evolution at this level can benefit from having more freedom in how basic structures are built. We do not want to form piles or sequences of blocks, this will be done at a higher level in the composite hirarchy.

At this level the composite generator has the following steps:

1) Generate the first generation of the population, consisting of randomly generated list of blocks.
2) Run a Simulation.
3) Evaluate each individual of the population.
4) If a member of the population has remaining pieces obtain the location and angle of the pieces, assign a fitness.
5) Do the selection, crossover and mutation operations.
6) Repeat the cycle.
7) From the best individuals select composites. Several algorithms could be proposed for this step, for instance selcting clusters of blocks.

Another option is to just manually generate interesting composites to be used by the algorithm. We could say that the previous approach is an option for a procedural generation of composites.

Fig. 1. List of pieces in the angry birds game.

The Angry Birds level generation track is a competition where the objective is to generate levels that are *fun* and *enjoyable* however they also have to be challenging but not

imposible. However the rules for the competition specify that a configuration file must be used to generate the levels of the game in a limited amount of time, the information on the configuration file is first the number of *different* levels that must be generated, the second value is the different materials *not allowed* to be used, for instance a group of generated levels must not have circles which composition material is ice or stone, moreover it does not have to integrate a triangle piece with ice, stone or wood material so this combinations must be removed from the pool of options, the third configuration value is the *number of pigs* that must be placed in a level, and finally the *limit* of time that the system has to create the required amount of levels.

In our original approach, composites could added to the structure generator algorithm at runtime as a mutation of an individual. The structure generator is explained in the next section.

In our current approach the composite generator is only run once at the beginning, as a setup process.
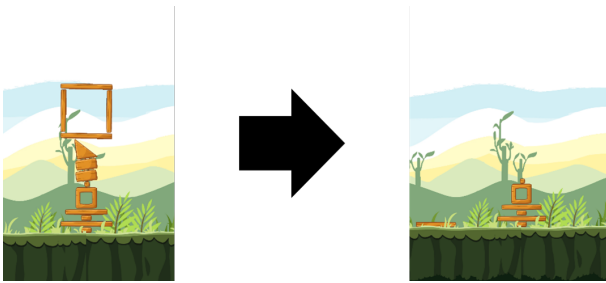
B. Structure Evolution

Fig. 2. Structure at beginning and end of a simulation.

Structures are the next evolutionary layer, at a higher level of composition. Structures are compositions of composites. A level can be seen as a collection of structures and each structure is a pile of composite blocks. This way of constructing levels has the advantage of limiting the search space, to structures that have a better chance to be standing after the simulation ends. There is no warranty that piles of composite blocks will maintain balance, so again we must search for those structures.

Structures are represented in a way similar to Togelius et al. [15]. Each block has an id assigned to it. A structure is then a list of ids, refering to composite blocks, the first 11 are basic composite and generated composites are assigned a higher number. This way and individual can be represented as shown on figure 4 where each chromosome represents a certain item in the list of blocks, the numbers for this elements are assigned in a first come first served basis, the numbers are infinitely incremented and the same blocks can be repeated in a structure.

These structures represent individuals in the evolutionary algorithm. Once an individual is created it is ready to be evaluated in a simulation on the game environment. Using a similar approach as the composite generator, all individuals
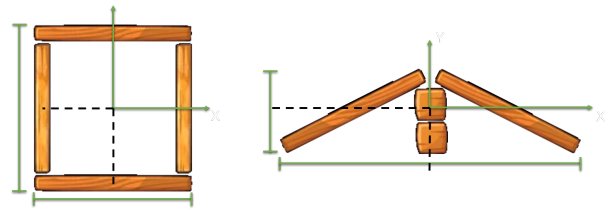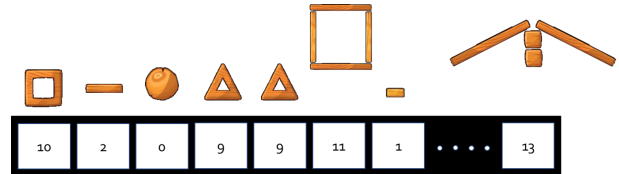

Fig. 3. Bounding box calculation.

Fig. 4. Chromosome chain of an individual.

have a maximum of ten seconds to maintain balance or reach absolute stability, if absolute stability is reached it means one of two things, first the generated structure is completely stable or second, the tower fell to either side and the pieces were destroyed or stopped moving by gravitational pull, when either of this happens by more than two seconds the simulation ends regardless of time remaining and the next individual is simulated.

The evaluation process for this project is still in the development stages, and in our experience is one one of the more difficult components in this kind of systems. In this stage of the evolution, we first need to assign a good fitness to those structures that keep their balance and not fall apart. There are other variables we take in to account, but they will be explained later. In this initial stage, each individual in the population has 100 points so before the simulation starts each of them are viable candidates to be an elite member of the population regardless of their relative height or complexity.

First, after each of the individual simulations are over the evaluation process takes place, in this phase the resulting level is checked for those blocks that survived, that is they where not destroyed by falling. Each one of them is counted and then the resulting number is compared to that of the initial quantity of pieces that were originally put in before the simulation, each block represents a certain percentage of 100% so the missing pieces are subtracted from the total. In this moment, each block is counted as if it was not a member of a composite, this is becuse after the simulation there is only data about each individual block.

C. Structure Positioning

The positioning of structures in the map is important in order to generate levels that are fun and aesthetic. For this we need a method to indcate the positioning of each structure. We propose the use of a mask where the positioning of each pile and the number of blocks it contains is specefied. Again these masks could be manually created, for instance using the rule of thirds [16] which is a guideline used while creating images,

the purpose is to create aesthetical pleasant images where the focus point or the most important area is at the center of the image as shown on figure 5, using this rule different distribution groups or masks are created has shown on figure 6 in order to use them before the simulation of an individual by placing the pieces in order to distribute them in a way that can cover more area and create more balanced levels instead of using a tower like distribution as shown on figure **??**. There is an option in the system to generate in a first step a collection of masks, following this rules, but this qpproach has a drawback. Pre-generating masks would not allow the generator to *evolve* because the individuals would be limited to a certain number of possibles combinations which will in turn prevent diversity in the population, so in order to prevent this we proposed an approach that allows the individuals to generate a mask in which a number of columns is defined by default but the amount of pieces in each one is random, however the total amount of assigned blockes in all columns must be the same as the total pieces allowed in each individual. The number of maximum blocks is as a parameter. An example of this can be seen in figure 8 where the total of pieces in a individual is 18 and this pieces are distributed on all the columns available.
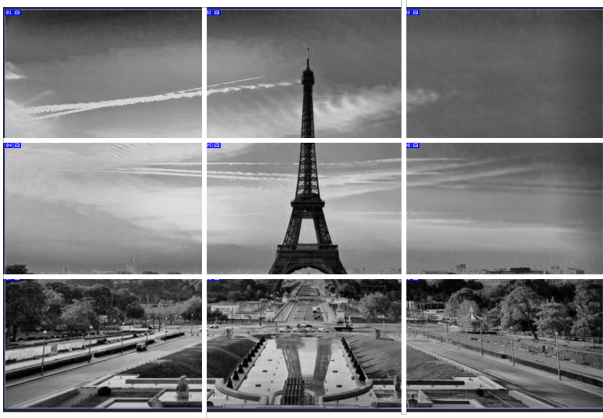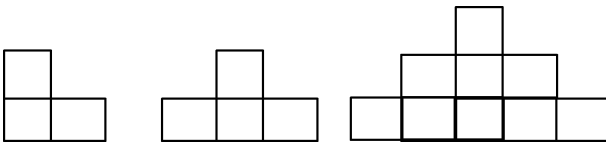

Fig. 5. Rule of thirds example.

Fig. 6. Masks created using the rule of thirds.

A diagram showng the main components and data flow of the proposed system is shown in Figure 9 in which the parameter file of the competition is integrated as an input. The layers of the system is exaplained next:

- Composite Generator: This layer represents the composite blocks assigned to a particular individual as list of integer numbers, where each number represents a reference to a composite object including the ones generated by the
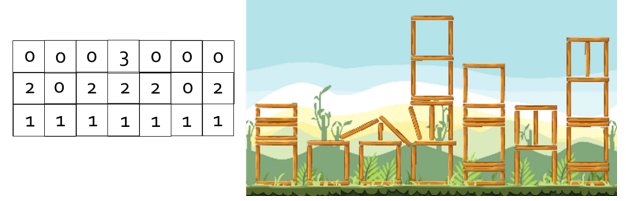

0	0	0	3	0	0	0
2	0	2	2	2	0	2
1	1	1	1	1	1	1

Fig. 7. Results obtained with rule of thirds.

			0			
4		1	2	6		2
1		5	8	9		3
0	0	4	3	2	4	5

Fig. 8. New chromosome chain.

OOE algorithm, the representetion of this is was shown in figure 4.

- Structure layer: Individuals are represented by a list of seven integers, representing a mask in which integers assigned in each column represents the amount of composite blocks from the element layer that will be assigned to said column, a representation of this is shown on figure 10.
- Enemy layer: also known as pig layer, this layer is represented by two elements a single numeric value representing the total amount of pigs assigned to the level and an array of locations for the placement of this same pigs, a representation of it is shown on figure 11.

D. Evolution of Levels

Once structures are able to stand, there are other aspects we need to take in to account to evaluate their fitness. In this case we propose the use to reward both the *"diversity"*
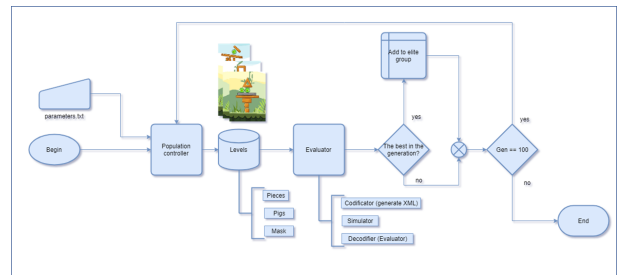

Fig. 9. New system diagram based on the parameter file and the new chromosome representation.

| 6 | 1 | 7 | 7 | 2 | 3 | 0 |

		6	7			
8		8	8			
7		6	3			
6		6	2			
0		6	4		4	
6		6	5	1	10	
4	9	2	5	3	3	

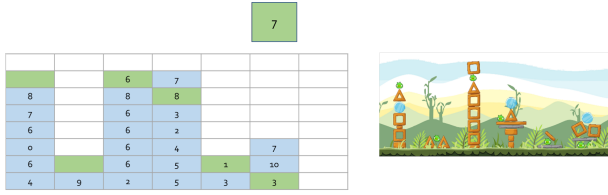Fig. 10. Mask array with column and fenotype representation.

Fig. 11. Amount of pigs in a level, their location relative to the mask layer (green) and fenotype representation.

and *"uniqueness"* of a individual defining a fitness function considering:

- Hamming distance: a simle explanation for the *Hamming distance* is that it is a calculation applied to a pair of strings where the objective is to find the minimum required number of substitutions required in order to convert a string exactly the same as the other, its formula is shown in equation 1, since the formula itself locates the minimum number of substitutions then we compare the individual with the rest of the population trying to find the maximum number of substitutions required efectively using the formula to give a better fitness to those individuals that are different from the group.
- Shannon entropy: the shannon entropy is used on information theory to calculate the disorder or uncertainty, in this case it will be used to determine if a level is *"boring"* or *"interesting"* based on the amount of entropy calculated according to the pieces used in a level where a low entropy is boring because if a determined level as a lot of repeated pieces then it will look simple, the formula used is shown in equation 2 where the elements calculated are the pieces of a level and their appearance probability against all others.

$$d = min\{\ d(x,y) : x,y \in C, if\ x \neq y, 1\ else\ 0\ \} \qquad (1)$$

$$S = -\Sigma_i P_i \log P_i \qquad (2)$$

In order to select the parents of the next generation the parents are ordered by their fitness. There are two types of selection:

- Tournament selection: in order to select the parent by this method the system checks the amount of crossovers assigned, then calculates the ammount of parent required to compete in pairs, then by doing a descendent selection the slots for the tournament are filled and the winners are calculated in each group.
- Roulette selection: this method of selection was first decided to be used directly by giving a proportional amount of chances to be selected according to the fitness of a individual, however since one of the objectives is to obtain variety of levels the roulette must first be ranked in order to ensure that low fitness individuals can be selected as parents.

As for the crossover operations the system currently uses two types, single point crossover and double point crossover, however this crossover operation is not only applied to the piece array of an individual, but it is also applied to the mask of said individual, this way the mask can ensure that some pieces of some childs in the population are now placed in the level, this can bring the possibility that a mask that uses less pieces that it normally as can obtain a better fitness than one that uses all of them. An example of this two crossover types can be seen in figure 12 in which the mask of an individual is combined with another and the resulting mask for the childs have different amount of pieces to be placed.
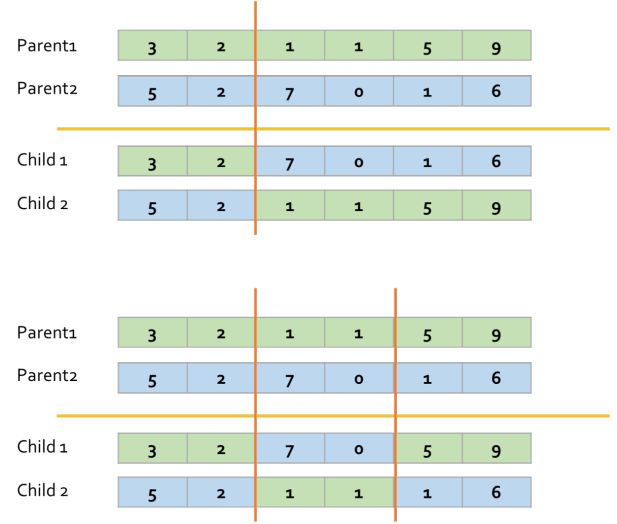

Fig. 12. Single point crossover(up) and double point crossover(bottom) applied to an individual mask.

The mutation operations of the generator work with the pieces layer of an individual, the four types of mutations are:

- Add or remove elements from the piece list
- Change the piece that is being used in a point of the list
- Change the material a piece is made offsets
- Change the x and y coordinates of a piece

This mutations do not have to be used all at othe same time on a single individual but depending on the probabilities they can happen all at once, since most of the pieces on a individual are mostly the same type of material then at least the material mutation can occurr on all or most of the individuals to ensure that all levels will not have all the pieces with the same material which could render a level *"boring"* by the fitness function.

Finally figure 13 and figure 14 represent the way all the layers of an individual al combined to create a level after all the selection, combination and mutation operations, first in figure 13 the composites layer is obtained from an individual, by a background process the width and height of all composites are obtained, then using the mask from the same individual the composites are placed in the order they are in their respective list (left to right in the level) in case a mask is not able to accommodate all the composites the remaining ones are not

placed nor taken into account on the evaluation process, in the opposite case that there are not enough composites to fill a mask then the remaining areas are not taken into account as well, then in figure 14 the previous combination of layers is evaluated by cheking which placed composites can hold a pig in the center or on the top of it, then the pigs are placed in the same oreder the composites are placed (left to right) in the locations where one is able to be placed.
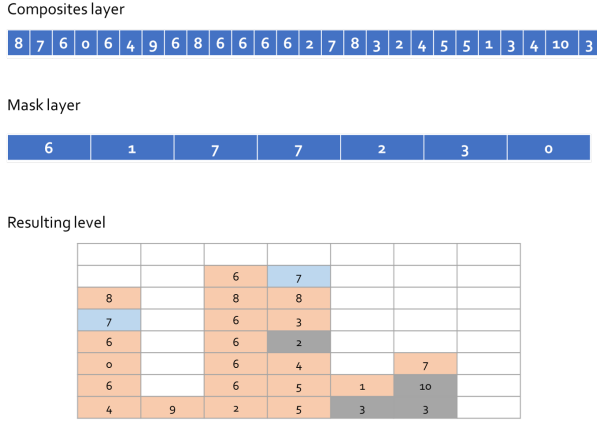

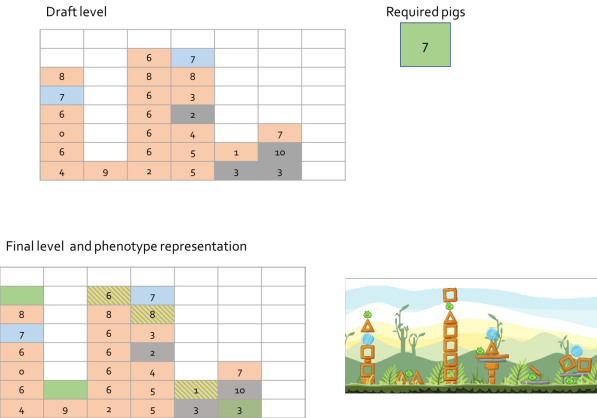Fig. 13. Combination of composites layer and mask layer.

Fig. 14. Combination of figure 13 layers with pig layer.

Using all this configurations the system is capable of creating levels, some of them playable as shown in figure 15 however the levels that can be generated are far from optimized to be played on, this is an issue that must be changed in future instances but the generator itself is able to be used as a base for a more complex one, and as shown in the future work section this issues can be solved by different means.

IV. CONCLUSIONS AND FUTURE WORK

The system has been able to generate levels that fulfill the requirements provided by the parameters file, however a way to evaluate that the levels generated are indeed *"fun"* or *"interesting"* is still needed, at this point we could evaluate the levels by appling an aproach used in previous competitions:

Fig. 15. Example of resulting level with current system.

- User survey, in case a survey is used as an evaluation tool then first the system has to generate the levels and a group of people must participate by playing the levels and answering a set of questions afterwards, the questions themselves must try to capture the feeling of a person towards the game like questioning if a level was too easy or too dificult to solve, if the generated structures look interesting or if the level required them to strategize how to use the provided birds to solve the level, the survey must be applied to a equal quantity of people of three different game skill levels,*"non video game players"*, *"casual video game players"* and *"avid video game players"*, this way we can evaluate a level or the entire system by knowing if all skill levels did not enjoyed a level because it was too simple or easy.
- By using a naive agent, if a naive agent is used to play the levels we can obtain data of how the agent solves the levels placed before them, and the way to evaluate if the level is or not *"interesting"* will be by checking how easily the aggent is able to solve a level, the point here is to have a level that cannot be solved by a single movement but to have one that a human player has to think a while to solve tryingto place the difficulty of a level just at the middle point between easy and hard difficulty levels.

Other aspects that the system need to improve is in the processing time required, since the current way to calculate the fitness and create the next generations of a population the system requires the system to execute simulations of the levels on every generation that passes therefore increasing the time it takes to complete an entire cycle of execution, so in order to change this we require to create a way to evaluate a level using only the information of the pieces, locations and pigs locations and using this data to predict the way a level will behave when placed in a real simulation environment therefore drastically reducing execution times.

Finally a different method to evaluate the individuals must be created, this is because the current evaluation method allows us to generate playable levels, this levels are far from ideal in the way that they only use a certain range of items present in the game therefore reducing the potential of a level, we expect to make the current fitness function work the most eficient that it can but another possible solution is found a good combination of formulas to generate the fitness function.

REFERENCES

[1] G. N. Yannakakis and J. Togelius, "Artificial Intelligence and Games (First Public Draft)," p. 359, 2017. [Online]. Available: http://gameaibook.org/book.pdf

[2] ——, "Experience-Driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, jul 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5740836/

[3] Rovio Entertainment Corporation, "Angry Birds," 2009. [Online]. Available: https://www.angrybirds.com/

[4] M. Stephenson, J. Renz, L. Ferreira, and J. Togelius, "Competitions – IEEE Conference on Computational Intelligence and Games, 14-17 August 2018, Maastricht (The Netherlands)." [Online]. Available: https://project.dke.maastrichtuniversity.nl/cig2018/competitions/#angrybirds

[5] M. J. B. Stephenson, J. Renz, X. Ge, L. N. Ferreira, J. Togelius, and P. Zhang, "The 2017 AIBIRDS Level Generation Competition," *IEEE Transactions on Games*, pp. 1–1, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8410472/

[6] J. Renz, X. Ge, R. Verma, and P. Zhang, "Angry Birds as a Challenge for Artificial Intelligence," *Thirtieth AAAI Conference on Artificial Intelligence*, mar 2016. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewPaper/12527

[7] L. Ferreira, "Science birds," https://github.com/lucasnfe/Science-Birds, accessed: 2018-06-17.

[8] R. K. STANDISH, "OPEN-ENDED ARTIFICIAL EVOLUTION," *International Journal of Computational Intelligence and Applications*, vol. 03, no. 02, pp. 167–175, jun 2003. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S1469026803000914

[9] T. Taylor, M. Bedau, A. Channon, D. Ackley, W. Banzhaf, G. Beslon, E. Dolson, T. Froese, S. Hickinbotham, T. Ikegami, B. McMullin, N. Packard, S. Rasmussen, N. Virgo, E. Agmon, E. Clark, S. McGregor, C. Ofria, G. Ropella, L. Spector, K. O. Stanley, A. Stanton, C. Timperley, A. Vostinar, and M. Wiser, "Open-Ended Evolution: Perspectives from the OEE Workshop in York," *Artificial Life*, vol. 22, no. 3, pp. 408–423, aug 2016. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/ARTL_a_00210

[10] T. Taylor, "Evolutionary Innovations and Where to Find Them: Routes to Open-Ended Evolution in Natural and Artificial Systems," *Artificial Life*, vol. 25, no. 2, jun 2018.

[11] J. Renz, X. Ge, S. Gould, and P. Zhang, "The angry birds ai competition," *AI Magazine*, vol. 36, pp. 85–87, 2015.

[12] M. Kaidan, C. Y. Chu, T. Harada, and R. Thawonmas, "Procedural generation of angry birds levels that adapt to the player's skills using genetic algorithm," in *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. IEEE, oct 2015, pp. 535–536. [Online]. Available: http://ieeexplore.ieee.org/document/7398674/

[13] Y. Jiang, T. Harada, and R. Thawonmas, *Procedural generation of angry birds fun levels using pattern-struct and preset-model*. IEEE Conference on Computational Intelligence and Games, 2017. [Online]. Available: https://aibirds.org/Level-Generation/cig16-competition-rules.pdf

[14] L. Calle, "Generating structures for angry birds via free form evolution," p. 10, 2019.

[15] J. Togelius, N. Shaker, and M. J. Nelson, "Representations for search-based methods," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, N. Shaker, J. Togelius, and M. J. Nelson, Eds. Springer, 2016, pp. 159–179.

[16] D. Rowse, "Rule of thirds," *Digital Photography School*, 2012.