Procedural generation of levels fot he 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

Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

***Abstract*—This project consisted in the generation of an evolutionary computation-based system capable of generate and evolve the structures of which one level of the Angry Birds game is composed, these structures are evaluated according to the stability of the structure as well as for the complexity of said structure.**

For this a level generation system was designed based on the data obtained from the game, said data consist in the number and type of pieces that appear and the applied gravity of the game, the individuals of the group are evaluated by a simulation of the generated level and then checking how the structures are affected by the gravity of the game.

The evolutionary computation system as the main objective of generating structures based on the existent pieces of the game and evolving said pieces by combining them in a process that simulates the rules of the Open-Ended Evolution algorithm in which the evolution of this compounds is not inclined towards a numeric objective rather than to extend the diversity from which the pieces may be selected for a level.

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

I. Introduction

Procedural content generation is an interesting subject on the entertainment industry, primarily in the video game development because of the easines and quickness provided to generate content for different video game projects [ref] . In this paper an evolutionary content generation system is proposed to generate levels for the popular Angry Birds video game [1], taking inspiration in concepts of open-ended evolution.

For this project a procedural generation content system is proposed for the game angry birds, angry birds is a game developed by Rovio Entrtainment 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 different resistances 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 the structures and the objectives, it is possible to obtain a high score by doing the most destruction as possible with the least birds as you can. [1]

While it is possible to create a evolutionary agent to play the levels and obtain the best score the main objective of this project is to generate the levels that can be played by naive agents or physical players, the levels themselves need to be complex in the sense that interesting structures can be created but it has to be possible to complete in some way [2, 3].

Procedural content generation provides a peculir case of study because of the way content has to be made for a each particular type of game, in this particular case the restrictions that the game and the competition track have provide an interesting challenge for a evolutionary algorithm because we have to tanke into account the different pieces that can be used, their interactions with one another because of gravity and the possible effects the trajectory of a shot from a player can affect all of this.

In order to tackle this case of study we proposed to use a genetic algorithm capable of creating content, more specifically levels for the angry birds video game in which the generated levels satisfy the objectives of being *interesting* and *playable*.

After concluding this paper we expect to identify the places were a evolutionary algorithm can be used alongside procedural content generation applied to video games in order to be able to clarify in which areas a evolutionary algorithm can used to further improve video game development cycles.

As mentioned above in this paper we use procedural content generation (PCG) which denotes the way to automatically generate content by algorithms instead of generating the same content by manual means such as with different personnel.

In the videogame industry many developers use a PCG style software that produces raw images and designs used by other personnel to improve it and use it in order to reduce the development cycles, and the productions cost. PCG as six focus areas listed below

PCG as six focus areas listed as follows:

- Level Design: This aspect involves only the design of specific sections of a game, primarily those in which the player is able to interact.
- Audio: This aspect is focused on designing soundtracks that are able to react to a player specific action.
- Visuals: This area is focused on creating or improving visual representations of objects on games such as giving

raw designs of players faces of improving the images of other graphics.
- Narrative: This aspect is focused on the generation of readable and coherent stories or plotlines of games.
- Gameplay: This aspect uses agents to test and evaluate the mechanic on which a game is based by trying to play as a human would in order to improve said mechanics.
- Game Design: This aspect is focused on generating the rulesets and mechanics of a new game. [4, 5, 6]

Furthermore we use genetic algorithms which is a method developed by John Holland on 1975 [7] the genetic algorithm (GA) is an optimization and search technique based on the principles of genetic and natural selection, it allows a population composed of many individuals, which by themselves represent a possible solution to the problem, to evolve under specific selection rules in order to obtain the maximum fitness.

In other words a genetic algorithm is a method to solve optimization problems by creating a population that will represent a possible solution and repeatedly doing genetic operations as selection, crossover and mutation on each member of the population in order to generate children for the next generation that would be closer to the best solution of the problem than their parents, the cycles in this method where the operations are made are called generations and by moving over this the population "evolves" o in other words, they get closer to the best possible solution to a specific problem.

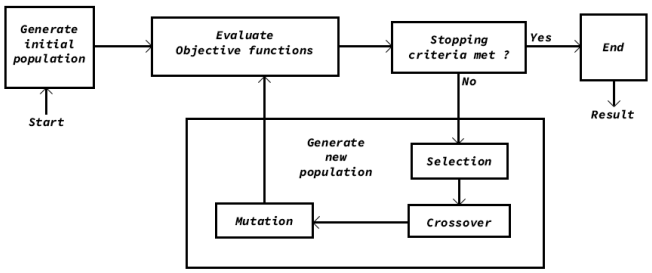

Fig. 1. Life cycle of a genetic algorithm.

The main life cycle of a genetic algorithm shown of figure 1 is described as follows:

1) Initialize a random n population of values suitable for the solution.
2) Evaluate the fitness of each individual (solution) in the population.
3) Check if the stopping criteria has been met.
4) If not.
    a) Introduce new individuals to the population by crossover operations.
    b) Mutate a random value of the new individuals.
    c) Select new cross over individuals according to their fitness for the next generation.
5) Loop from step 2 until an end condition is satisfied.
6) Select the best individual of the population as the solution for the problem

And finally we use our own interpretation of an Open-ended evolution (OEE) algorithm which is a variant of genetic evolution algorithm where the objective differs from the regular genetic algorithm one, in this case the final objective is not to reach a final state of the evolution but to continue evolving and creating new types of entities more complex on each new generation, these same entities have the possibility to evolve not only inside the frame of their type but create entirely new groups. [8]

Another explanation provided by T. Taylor et al. [9, 10] defines OEE has 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 has a process of genetic evolution that encapsulates a group of similar processes and that is able to increase their number by the more complex the entities become.

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 basis of the development, modification to the original proposed method as well as different approaches that were taken to solve the problem. 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. Overview of existing work

In this Section we present some of the different approaches to the procedural generation of content for the angry birds video game.

M. Kaidan et al. [11] 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.

J. Renz et al. [12] created an article in which the angry birds level generation contest mechanics of the IEEE Conference on Computational Intelligence and Games are based on, it explains some of the solving methods proposed by previous competitors.

Y. Jiang et al. [13] proposed the use of predefined letter style patterns and combined the to create a small pool of word 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

III. Approach

In this Section we describe the first approach that was used for this project, the game contains a total of 11 pieces as shown on figure 2 that 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 this same pieces in different locations and angles in order to create structures that can be used as building blocks for more complex levels.

Since there is no way to *add* a composite to the pool of pieces of the game then a way to combine both groups is needed, in order to do this first the original 11 pieces of the game will be converted into composites, since the easiest way to read the data to create composites is by using lists then a *composite* of one element will be created for each piece, after this in order to add a composite of pieces as a new component first a group of existing composites is selected and placed in a imaginary enviroment then as a unique group they are measured, an imaginary bounding box is defined by calculating the lengths of all the pieces, finding the borders of said pieces and then find 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 4, 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.

So using the composite generator the order of this sequence of events needs to be done as follows:

1) Gnerate composited to add to the pool of options
2) Generate the first generation of the population
3) Run a Simulation
4) Evaluate each member of the population
5) If a member of the population has remaining pieces obtain the location and angle of the pieces, create a composite and add it to the pool
6) Select the parents for the next generation
7) Do the crossover and mutation operations
8) Repeat the cycle

Fig. 2. 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 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 infornmation 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 des 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.

After taking into account this restrictions the strcture of our current approach had to be changed, so this section will be divided in tow subsections, the first will explain the approach that was used and after that second will explain the modifications that had to be made in order to integrate the configuration files for the competition.

A. *Original approach*

Since the algorithm has to generate more complex structures the remaining pieces of a member of the population have to be used has a starting point, this new

additions to the piece pool are added to the population by the mutation operation, the objective in the first phase of the project is to generate structures that are capable of maintaining their own balance as shown on figure 3, since the objective of open-ended evolution for this project is to add new composites to the pool of options the algorithm will first run a simulation with a group of pieces and after it the ones that were not destroyed by falling because of gravity will be saved and added to the pool of options.

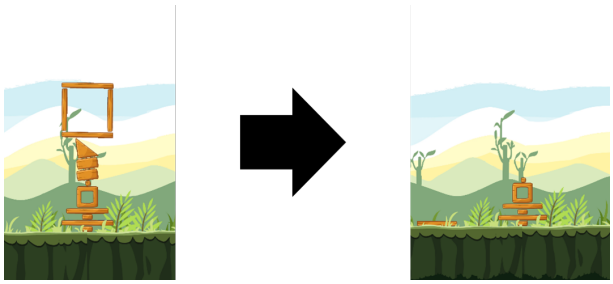

Fig. 3. Structure at beginning and end of a simulation.

The way to represent the elements of a member of the population is by using the id of each piece and the new added ones has a pointer to the respective data that needs to be obtained, this way and individual can be represented as shown on figure 5 where each chromosome represents a certain item in the pool of pieces, the numbers for this elements are assigned in a first come first served basis, the numbers are infinitely incremented and this same pieces can be repeated on each individual of the population

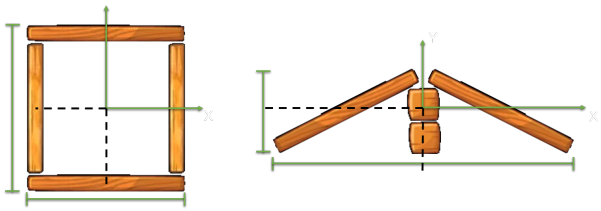

Fig. 4. Bounding box calculation.

Once the pieces have been added and the individuals are ready to be evaluated a simulation on the game environment is scheduled by order in the population, all the individuals have a maximum of ten seconds to maintain balance or reach absolute stability, if absolute stability is reached it means one

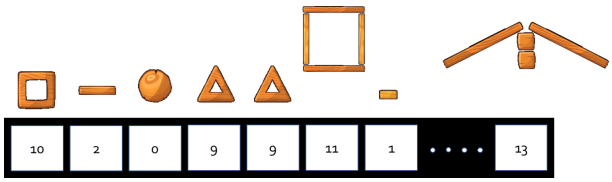

Fig. 5. Chromosome chain of an individual.

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, however in order to check that the evolution is working has it should a criteria has been temporarily made as follows, each individual in the population has 100 points so before the simulation starts each of them are viable candidates to be a elite member of the population regardless relative height or complexity, then the remaining of the process is separated in three phases has follows.

First, after each of the individual simulations are over the evaluation process takes place, in this phase the resulting level is checked for pieces, 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 in order to calculate the first element of the evaluation process, each piece represents a certain percentage of 100% so the missing pieces are subtracted from the total.

Second, after removing percentage according to the missing pieces an individual remaining pieces are checked for their positions in comparison to the original ones before the simulation, using the formula 1 each piece within the individual is evaluated according to the center point of the piece, if the piece moved more than a certain threshold half the percentage for one piece is removed from the remaining percentage, after this using the formula 2 the position of the piece is further checked this time by the difference in angle from the original one, in case the difference of angles is more than a certain threshold half a piece value is further removed from the total.

$$error_{xy} = \begin{cases} 0.1 & \text{if } 0.08 > d, \\ \dfrac{100}{lenght\_pieces} * 0.5 & \text{if } 0.08 < d. \end{cases} \quad (1)$$
$$where: d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$error_r = \begin{cases} 0 & \text{if } -5 \leq r \leq 5, \\ \dfrac{100}{lenght\_pieces} * 0.5 & \text{if } r < 5 or r > 5. \end{cases} \quad (2)$$
$$where: r = |rotation_0| - |rotation_f|$$

Finally, the total height of the remaining group of pieces is calculated by combining the bonding boxes of all pieces, then the individuals are ordered by the obtained percentage and then further ordered by the total height, this way the most balanced ones will have the best probability to be selected together for a crossover operation to create the next generation.

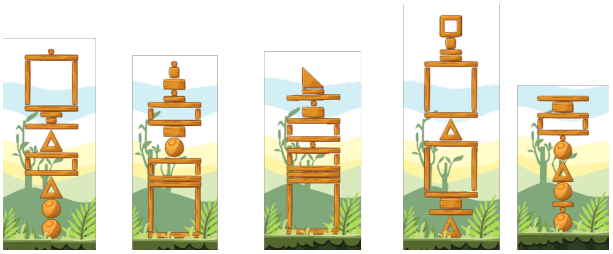

Fig. 6. Result samples of current system.

Since there is a need to create complex structures than can cover a great section of the play map, a method to evaluate the distribution is needed, for this instance we propose using the rule of thirds [14] which is a guideline used while creating imagen in which 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 7, using this rule different distribution groups or masks are created has shown on figure 9 in order to use them before the simulation of an individual takes place in order to distribute the pieces in a way that can cover more area and create more balanced levels instead of using a tower like distribution as shown on figure 6.

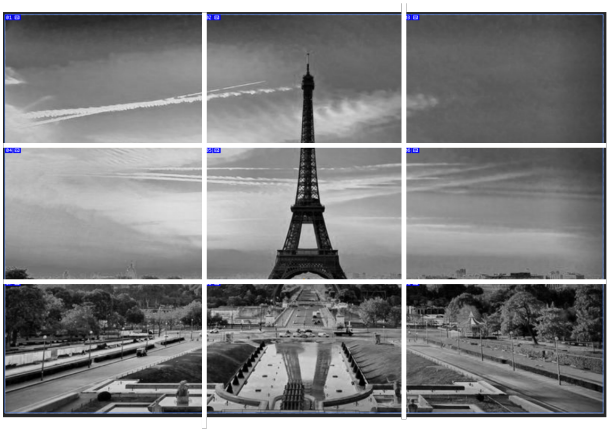

Fig. 7. Rule of thirds example.

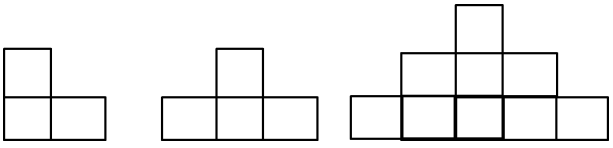

Fig. 8. Rule of thirds applied to an individual.

Using this same distribution a new evaluation will be added in which according to how much of the area is filled a score will be given and this score will be used as a new sorting

value at the moment of selecting individuals for the crossover operation, since as shown on figure 8 there is a possibility that a huge mask is given to an individual it can be possible that the amount of pieces will not be able to cover the full area given, so in order to prevent this kind of problems we also proposed to give each mask a value that refers to the minimum amount of pieces and height required to fill the most of the mask then check the amount of pieces and height before simulation for a single individual and then randomize which mask of the ones that can be used will be assigned to that individual, this way the results shown on figure 10 are obtained, this however gave us a new problem because as shown in the image and the previous image as well the were instances in which a certain combination of pieces could be to big for the area and this pieces resulted in a level that had structural flaws from the beginning which cold provoke the entire structure and immeadiately nearby others to fall at the beginning of the simulation however it was able to

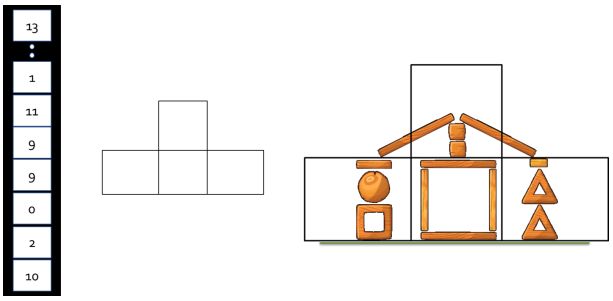

Fig. 9. Rule of thirds applied to an individual.

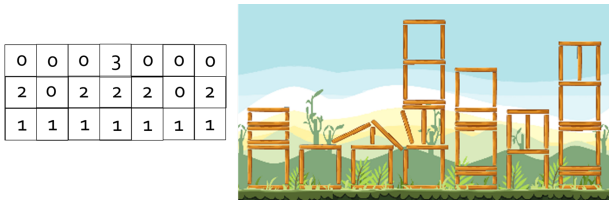

Fig. 10. Results obtained with rule of thirds.

B. New approach

The first differece that the new approach as is that the masks that are assigned to each individual are no longer pre-generated ones, this is because we decided that pre-generating them 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 the modification to this section allows the individuals to generate a mask in which a number of columns is defines by default but the amount of pieces in each one is random, however the total of assigned pieces in all columns must be the same as the total pieces alloweed in each individual which is a number assigned at before beginning the system, an example of this can be seen in figure 11 where the total of pieces in a individual is 18 and this pieces are distributed on all the columns available. This is further explained in this section.

The competition also requires that a generator gives a set output a file is used to simulate the ruleset provided at a competition, this *text file (parameters.txt)* provides the specifications that must be met in the competition, the parameters in the file are as follows:

- Total required levels
- Restrictions to pieces or piece-material combinations that cannot be placed or used in the levels, in this case if the file specifies that a *CIRCLE* made out of *ICE* cannotplaced the system must be able to verify that said combination is not used.
- Total number of pigs, this value can be a single value or a range of values that represent how many pigs must be placed on a level.
- The amount of time in which the system must give an output.

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

Fig. 11. New chromosome chain.

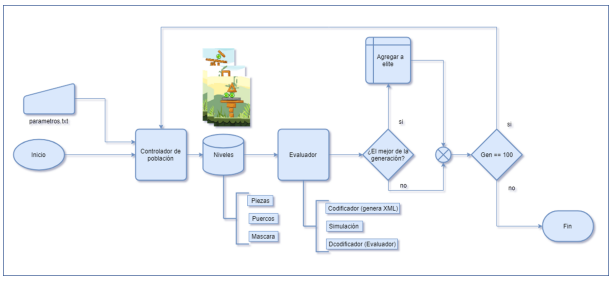

Fig. 12. New system diagram based on the parameter file and the new chromosome representation.

Using this as a base a new system diagram is proposed as shown of figure 12 in which the file with parameters is integrated with the input data the system requires, the way the individuals of the population are further presresented and the way the simulation phase is taken, the method to representate the individuals is proposed by dividing the individual on three layers represented as follows:

- Element layer: also known as pieces layer, this layer represent the elements assigned to a particular individual by an array of numbers from 0 to infinite where each number represents a reference to the class of a piece in the selection pool including the ones generated by the OOE algorithm, the representetion of this is was shown in figure 5.
- Mask layer: this layer is represented by a 7 digit array in which the value assigned in each column represents the amount of pieces from the element layer that will be

assigned to said column, a representation of this is shown on figure 13.

- Enemy layer: also known as pig layer, this layer is represented by two elments 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 14.

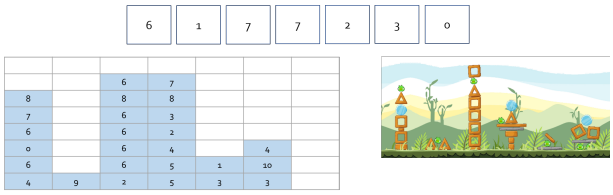

Fig. 13. Mask array with column and fenotype representation.

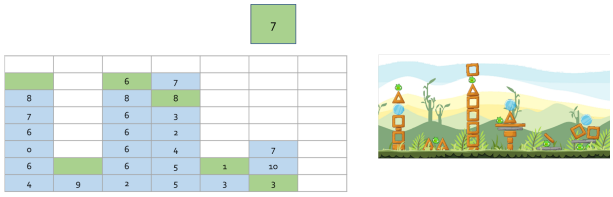

Fig. 14. Amount of pigs in a level, their location relative to the mask layer (green) and fenotype representation.

In order to select the parents of the next generation the parents are ordered by their fitness, then the system can choose between 2 options for the selection, tournament selection or roulette 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 thypes cna be seen in figure 15 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.

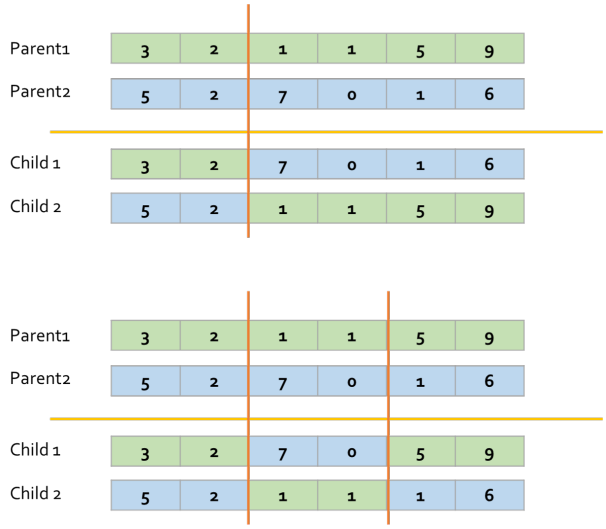

Fig. 15. Single point crossover(up) and double point crossover(bottom) aplied to an individual mask.

The mutation operations of the generator work with the pieces layer of an individual, the four types of mutations used 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 once 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.

Since this group of mutations further ensures diversity in the populations a different way to evaluate the individuals is needed, by using the previous proposed method as a base a way to evaluate the *"diversity"* and *"uniqueness"* of a individual, for this we first remove the x and y position as well as the *angle* errors in order to add two new evaluation formulas as follows:

- Hamming distance: a simle explanation for the *Hamming distance* is that it is a calculation applied to a pair of strings where the obecive 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 3, since the formula itself locates the minimum required number of substitutions then we compare the individual with the rest of the population trying to found 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 4 where the elements calculated are the pieces of a level and their appearance probability against all others.

$$d = min\{\ d(x,y) : x,y \in C, if\ x \neq y, 1\ else\ 0\ \} \tag{3}$$

$$S = -\Sigma_i P_i \log P_i \tag{4}$$

Using all this configurations the system is capable of creating levels, some of them playable as shown in figure 16 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.

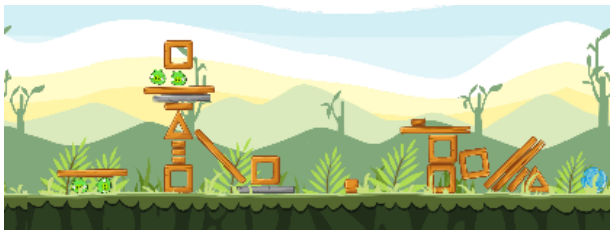

Fig. 16. Example of resulting level with current system.

IV. Conclusions and future work

The system as 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 have two was to evaluate the results:

- 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] Rovio Entertainment Corporation. *Angry Birds*. 2009. URL: https://www.angrybirds.com/.

[2] Matthew Stephenson et al. *Competitions – IEEE Conference on Computational Intelligence and Games, 14-17 August 2018, Maastricht (The Netherlands)*. URL: https://project.dke.maastrichtuniversity.nl/cig2018/competitions/%7B%5C#%7Dangrybirds (visited on 03/14/2019).

[3] Matthew James Burgess Stephenson et al. "The 2017 AIBIRDS Level Generation Competition". In: *IEEE Transactions on Games* (2018), pp. 1–1. ISSN: 2475-1502. DOI: 10.1109/TG.2018.2854896. URL: https://ieeexplore.ieee.org/document/8410472/.

[4] Georgios N Yannakakis and Julian Togelius. "Artificial Intelligence and Games (First Public Draft)". In: (2017), p. 359. DOI: 10.1007/978-3-319-63519-4. URL: http://gameaibook.org/book.pdf.

[5] Gillian Smith. "Understanding procedural content generation". In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. New York, New York, USA: ACM Press, 2014, pp. 917–926. ISBN: 9781450324731. DOI: 10.1145/2556288.2557341. URL: http://dl.acm.org/citation.cfm?doid=2556288.2557341.

[6] Noor Shaker, Julian Togelius, and Mark J. Nelson. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.

[7] John H. Holland. “Adaptation in Natural and Artificial Systems”. In: *Sgart Newsletter* (1975). ISSN: 0928-1231. DOI: 10.1007/s11096-006-9026-6. arXiv: 0262082136.

[8] RUSSELL K. STANDISH. “OPEN-ENDED ARTIFICIAL EVOLUTION”. In: *International Journal of Computational Intelligence and Applications* 03.02 (June 2003), pp. 167–175. ISSN: 1469-0268. DOI: 10.1142/S1469026803000914. URL: http://www.worldscientific.com/doi/abs/10.1142/S1469026803000914.

[9] Tim Taylor et al. “Open-Ended Evolution: Perspectives from the OEE Workshop in York”. In: *Artificial Life* 22.3 (Aug. 2016), pp. 408–423. ISSN: 1064-5462. DOI: 10.1162/ARTL_a_00210. URL: http://www.mitpressjournals.org/doi/10.1162/ARTL%7B%5C_%7Da%7B%5C_%7D00210.

[10] Tim Taylor. “Evolutionary Innovations and Where to Find Them: Routes to Open-Ended Evolution in Natural and Artificial Systems”. In: *Artificial Life* 25.2 (June 2018). arXiv: 1806.01883.

[11] Misaki Kaidan et al. “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. ISBN: 978-1-4799-8751-1. DOI: 10.1109/GCCE.2015.7398674. URL: http://ieeexplore.ieee.org/document/7398674/.

[12] Jochen Renz et al. “Angry Birds as a Challenge for Artificial Intelligence”. In: *Thirtieth AAAI Conference on Artificial Intelligence* (Mar. 2016). URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewPaper/12527.

[13] Yuxuan Jiang, Tomohiro Harada, and Ruck Thawonmas. *Procedural generation of angry birds fun levels using pattern-struct and preset-model*. IEEE Conference on Computational Intelligence and Games, 2017, pp. 154–161. ISBN: 9781538632338. DOI: 10.1109/CIG.2017.8080429. URL: https://aibirds.org/Level-Generation/cig16-competition-rules.pdf.

[14] Darren Rowse. “Rule of thirds”. In: *Digital Photography School* (2012).