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

Jaime Salinas-Hernández and Mario Garcia-Valdez

Abstract This paper 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.

keywords Genetic Algorithm, Procedural generated content, Open-ended evolution, Evolutionary computation

1 Introduction

Procedural content generation is an interesting subject on the entertainment area, primarily in the video game industry because of the easiness and quickness provided to generate content for different video game projects, in this paper a evolutionary computation-based system is proposed to generate complex content for videogames based on an evolutionary algorithm and open-ended- evolution.

Jaime Salinas-Hernández
Tijuana Institute of Technology, Tijuana, México e-mail: salinas.jaime1217@gmail.com

Mario Garcia-Valdez(✉)
Tijuana Institute of Technology, Tijuana, México e-mail: mariosky@gmail.com

For this paper a procedural generation content system is proposed for the game angry birds, 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 mechanics is to use a slingshot to shoot a certain number of birds with different abilities to destroy the structures and the objectives. [1]

While it is possible to create an evolutionary agent to play the levels and obtain the best score the main objective of this paper is to generate the levels that can be played by the 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].

In Section 2, a background of the subjects used on this paper will be presented, this subjects, while not all being used on by different authors on their works provide a great help on the development of the proposed system, Section 3, contains a quick overview of some aspects used by different authors in order to tackle this problem, their design and basic functionality, in Section 4 the description of the current state of the generated system is explained, from the basis of the development to the different methods that have been used to progress on the different sections of the system, in Section 5 we explain the future work of the paper, where we are in the development phase and how the future implementations are to further improve the paper as well as the conclusions that we reached.

2 Background

In this Section we provide a quick overview of the literature about the areas that provide the basis to the development of the current paper, this section is divided in three parts, the first one gives us an explanation of what procedural content generation entails, the second one provides an explanation on genetic algorithms and how can they be used to solve optimization problems, and finally, the third one provides us with an overview of Open-Ended Evolution and what it means to have an Open-Ended Evolutionary algorithm.

2.1 Procedural content generation

Procedural content generation (PCG) 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. [4] Nowadays there are game developer of AAA games that use software to auto generate some elements of

the enviroment such as vegetation of terrain layout specialy for open-world games, this in turn cuts a great amount of development time for said games [5, 6]. 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. [7, 8, 9]

2.2 Genetic algorithm

Is a method developed by John Holland on 1975 [10] 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 the members of the population 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.

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.

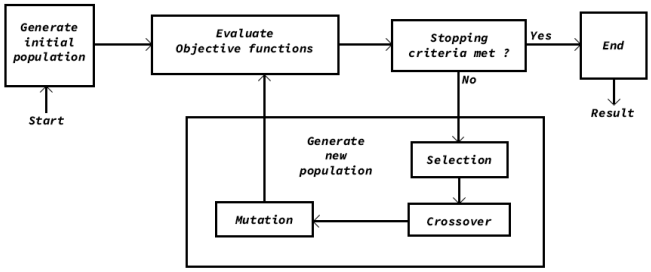

Fig. 1 Life cycle of a genetic algorithm.

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

2.3 Open-Ended Evolution

Open-ended evolution (OEE) is a variant of genetic evolution algorithm in which 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. [11]

Another explanation provided by T. Taylor et al. [12, 13] 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 paper 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.

3 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. [14] 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. The generation of the levels is based on a formula which take into

account the average velocities to evaluate the levels however the generation of the levels did not take into account the number of birds and pigs placed on the level and as such the levels were able to be cleared with ease by the participants.

J. Renz et al. [15] 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. Here it is explained that there are two tracks of the angry birds competition, the first one is focused on training agents capable of playing the game as well as a human could, taking into consideration the angle of the shots and the movement or destruction of the pieces created with that shot, the second track of the competition is based on the levels themselves in which a system can automatically create levels that are interesting to a human player, playable as well as difficult able to be completed with a given amount of birds.

Y. Jiang et al. [16] proposed the use of predefined letter style patterns and combined to create a small pool of words 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, the way the requirements for the competition were fulfilled were by creating the letter structures as stable as possible, use the provided layered structures, generating a said number of usable birds and pigs by using an heuristic calculation.

4 Current approach

In this Section we describe the approach that was used for the development of this paper, first presenting a quick overview of the integration of a genetic algorithm with the combination of pieces, the way the open-ended segment of the system handled the creation of new compounds for the pool of pieces and then an explanation on the way the individuals are evaluated.

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 bundles 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, in order to do this a sequence of events needs to be done as follows:

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

Fig. 2 List of pieces in the angry birds game.

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 system 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 system is to add new bundles 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.

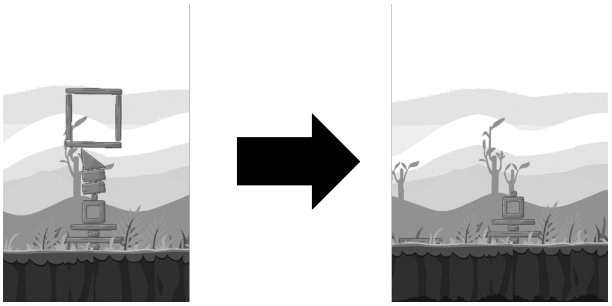

Fig. 3 Structure at beginning and end of a simulation.

In order to add a bundle of pieces as a new component first it has to be measured, an imaginary bounding box is defined by calculating the lengths of all the pieces, finding the borders of said pieces and find the points that could define a box around all of them, calculate the height and width then calculate the center point of said box and calculate 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 element so it can be used to place the pieces in their respective new positions when the files needed for the simulation are being created.

In case one or more of this pieces is rotated or needs to be rotated we defined a function that first obtains the data of a each piece in the group, finds the 4 edges of it and then if the piece needs a rotation different than 0 degrees a function to rotate the figure is called, the function returns a list of points that represent the borders of the rotated piece, after this the other pieces are processed the same way and with the points we calculate the entire bounding box as well as the center of the group that will be added.

In order to create a level generation algorithm there has to be a way to represent the level structure in a a way that the genetic algorithm is able to interpret it and at the same time be able to be interpreted by the game itself [17], so in this case 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.

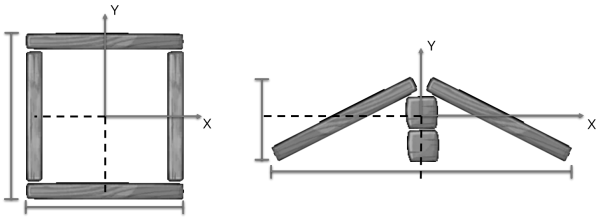

Fig. 4 bounding box calculation.

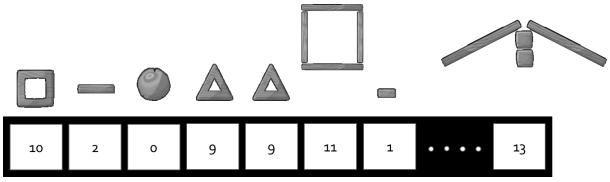

Fig. 5 Chromosome chain of an individual.

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 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 paper is still in the development stages, according to another work by Yannakakis et al. [18] there are various ways to evaluate the generated content, one example of this a an adaptive generation to optimize the player experience [19], however in order to check that the evolution in this current system 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 an 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.

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

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

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.

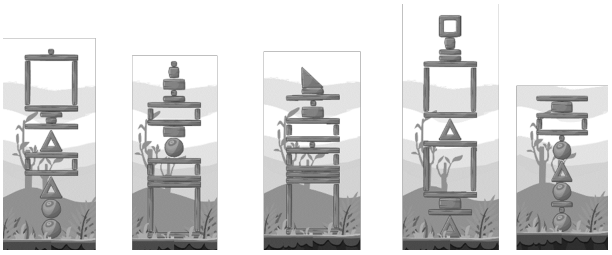

Fig. 6 Result samples of current system.

Using the proposed approach a modification to the generation of individuals was developed, this modification allowed the generation to create structures that are able to maintain balance except for some cases where some element of the individual where irregular pieces, this pieces like triangles and circles when placed in positions between the base of the tower to before the top caused the tower to fall depending if the subsequent pieces in the individual where able to outbalance the structure.

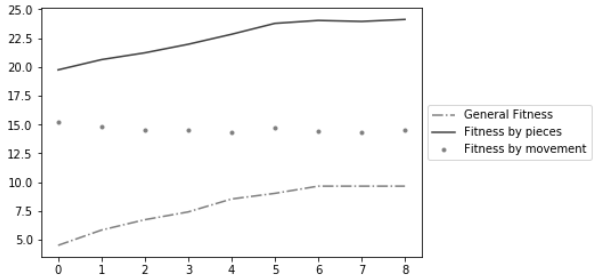

Fig. 7 Results with current approach.

With this in consideration the results as shown in figure 6 and subsequently figure 7 where able to be obtained, however since them objective of the generated system is to construct more complex structures the obtained results will be used as a base for the further development of the system with the implementation of the concepts presented in the future work section.

5 Conclusions and future work

The use of artificial intelligence in videogames is a common thing almost since before the first generation of videogames on 1972 as simply entities that would follow a set group of actions but as the generation kept advancing the need for more intelligent entities appeared, as of this generation some videogames use AI as ways to improve the interaction between man and game, but also using it in order to facilitate the way videogames are created, so for this system the focus of using evolutionary algorithms as great interest, as a starting point a rather simple game can be used as a workbench of the work but it also helps us define how much more can be done when all the required objects are obtained, improve on the same system and find ways for it to be used on different other games.

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 [20] 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 8, 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 10.

Using this same distributions as a base, the masks that will be used on the individuals to balance the content in the level will be defined the example below where a value of 1 represents an area of the level that needs to have pieces and 0 represents sections were no piece will be added.

```
type_castle = [[0,0,0,1,0,0,0],
               [1,0,1,1,1,0,1],
               [1,1,1,1,1,1,1]]
type_house = [[0,0,1,1,0,0,0],
              [0,1,1,1,1,0,0],
              [0,1,1,1,1,0,0]]
```

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 propose 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.

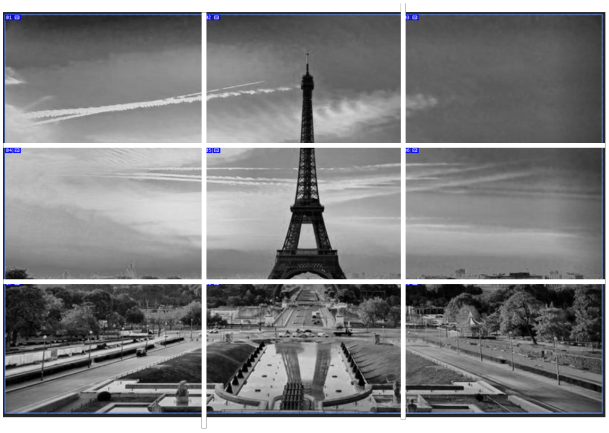

Fig. 8 Rule of thirds example.

Another proposed method for the evolution of the individuals is to use a mutation operation that is capable of adding or removing pieces from an individual, this way

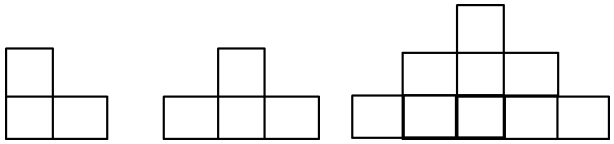

Fig. 9 Masks examples with the rule of thirds.

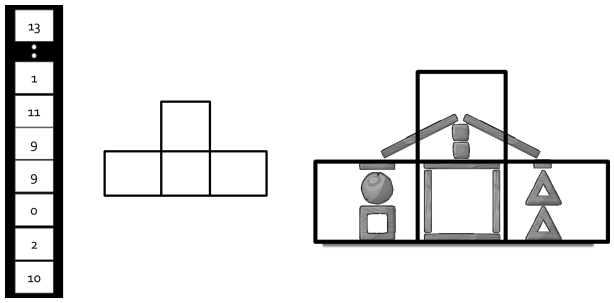

Fig. 10 Rule of thirds applied to an individual.

the diversity of the population will be greater on future generations, also by using the same mutation idea a change on the mask assigned to a particular individual can be modified as well, in the case a mask is either too small or too big for an individual because of the orientation of the pieces a precaution has been created, in case the mask is too small for the number of pieces after filling the mask areas the code will take the remaining pieces and begin placing them as if the mask was repeated, and in the case the number of pieces is too small the system will simply return the pieces that were able to be fitted in and continue with the process as usual.

References

1. Rovio Entertainment Corporation, "Angry Birds," 2009.
2. 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.
3. M. Stephenson, J. Renz, L. Ferreira, and J. Togelius, "Competitions – IEEE Conference on Computational Intelligence and Games, 14-17 August 2018, Maastricht (The Netherlands)."
4. J. Togelius, N. Shaker, and M. J. Nelson, "Introduction," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research* (N. Shaker, J. Togelius, and M. J. Nelson, eds.), pp. 1–15, Springer, 2016.
5. J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-Based Procedural Content Generation: A Taxonomy and Survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, pp. 172–186, sep 2011.
6. G. Smith, A. Othenin-Girard, J. Whitehead, and N. Wardrip-Fruin, "PCG-Based Game Design: Creating Endless Web," tech. rep., 2012.
7. G. N. Yannakakis and J. Togelius, "Artificial Intelligence and Games (First Public Draft)," p. 359, 2017.

8. G. 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), pp. 917–926, ACM Press, 2014.
9. N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.
10. J. H. Holland, "Adaptation in Natural and Artificial Systems," *Sgart Newsletter*, 1975.
11. R. K. STANDISH, "OPEN-ENDED ARTIFICIAL EVOLUTION," *International Journal of Computational Intelligence and Applications*, vol. 03, pp. 167–175, jun 2003.
12. 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, pp. 408–423, aug 2016.
13. T. Taylor, "Evolutionary Innovations and Where to Find Them: Routes to Open-Ended Evolution in Natural and Artificial Systems," *Artificial Life*, vol. 25, jun 2018.
14. 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)*, pp. 535–536, IEEE, oct 2015.
15. 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.
16. 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.
17. 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.), pp. 159–179, Springer, 2016.
18. G. N. Yannakakis and J. Togelius, "Experience-Driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, vol. 2, pp. 147–161, jul 2011.
19. N. Shaker, G. Yannakakis, and J. Togelius, "Towards Automatic Personalized Content Generation for Platform Games," *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*, oct 2010.
20. D. Rowse, "Rule of thirds," *Digital Photography School*, 2012.