Improved Free Form Evolution for Angry Birds Structures

Laura Calle
Universidad de Granada
laucalle09@gmail.com

Juan-Julián Merelo-Guervós,
Antonio Mora-García
Universidad de Granada/CITIC
Granada, Spain
(jmerelo|amorag)@ugr.es

Mario García Valdez
Tecnológico Nacional de México
Tijuana, México
mario@tectijuana.edu.mx

ABSTRACT

This paper presents an original approach based on evolutionary algorithms for building structures that are stable under gravity for the physics-based puzzle game Angry Birds, with the ultimate objective of creating Angry Birds levels with the minimum number of constraints. We have created a custom open source evolutionary computation library that implement an evolutionary algorithm whose main challenges have been to design a fitness function that, first, avoids when possible the time-consuming actual execution of the game, and, then, takes into account the different ways in which a structure is not sound and eliminates them. In order to test the method six experiments have been carried out, obtaining a variety of stable structures, which is the first path for the generation of levels that are aesthetically pleasing as well as playable.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; *Search methodologies*; **Heuristic function construction**;

KEYWORDS

Search-Based Procedural Content Generator, Evolutionary algorithm, Game development, Angry Birds, Level generation

ACM Reference Format:
Laura Calle, Juan-Julián Merelo-Guervós, Antonio Mora-García, and Mario García Valdez. 2019. Improved Free Form Evolution for Angry Birds Structures. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3322061

1 INTRODUCTION AND PROBLEM DESCRIPTION

Angry Birds is a mobile game created by Rovio Entertainment Corporation. In the game, there is a variety of defensive structures made out of blocks which protect *pigs* from the birds fired by the player. There is an ongoing competition on the generation of this kind of structures, which is a challenge from several points of view; most authors [6] evolve fixed-form structures known to be stable. Simulators are used to test procedurally generated game content; ours was forked from Science Birds [5] to produce usable output for an evolutionary algorithm [2]. In this paper we focus in the generation of free-form structures in an attempt to enhance replayability of the game. Since we are interested in structural integrity of the generated structures, neither actual gameplay nor players are taken into account; we will use a *direct fitness function*, which takes measures directly on the generated content, a time-consuming procedure since it actually involves entering the simulator and running it, so we will apply heuristics to the evolved structures to compute a fitness even before simulation.

The main feature of a stable level is that it is not in motion, so we will evaluate its steadiness as opposed to its speed as a baseline

$$fitness_{ind} = \frac{1}{|V|}\sum_{i=0}^{b} V_i + P_{broken} \cdot (b - |V|)$$

The modified simulator provides the average magnitude of velocity for each block, a set noted as V. The number of blocks in an individual is b, used to calculate . the number of broken blocks and apply a penalization to invalid levels ($P_{broken} = 100$). Since in-game simulation is a time consuming process, we will skip some levels based on indicators such as its distance to the ground, the number of overlapping blocks and the number of blocks broken during the simulation. In later experiments height is also taken into account [3], but still the result was not satisfactory. We needed a deeper exploration of the design space, which led us to the two new experiments we are presenting in this poster.

For representation, we aim at flexible and simple structures to allow a less directed search. Individuals are composed by an unordered, variable length list of blocks defined by their type (shape and size), position and rotation. This new representation needed a new, open source, evolutionary computation framework [1].

2 EXPERIMENTATION AND RESULTS

After the initial exploration of free form evolution in [3], there were two main problems: first, the time it took to enter the simulator and obtain the speed of the blocks; this is an implementation-level issue, but it had influence on the number of generations we could actually employ. Thus, in this new experiment, which we called E5, we will use Box2D [4], the physics engine used in the original game, adjusted the parameters so this simulation and the game behave in the same way. Execution time drastically dropped from 5 hours to less than 20 minutes on average, even running more generations in the process. Lower execution time allowed us to perform more operations like penalizing not only the distance to the ground but also the *gaps* in the Y-axis, which will make objects drop and maybe break. This will encourage individuals to grow vertically and not

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3322061

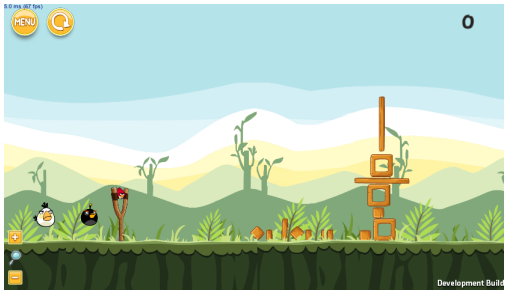

Figure 1: One of the solutions from Experiment 5

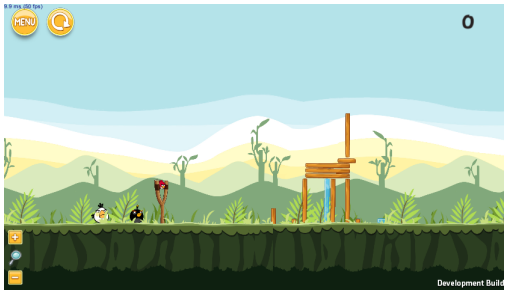

Figure 2: One of the solutions from Experiment 6

only horizontally like in previous experiments. This changes the fitness function, so we will have to compare by the actual obtained structure, one of which is shown in Figure 1.

In general, this penalization of gaps creates a faster path to higher stable structures. Still, it led to mostly flat structures with some higher block in unstable positions, which are structurally solid, but not interesting. One of the stable results is shown in Figure 1.

Observing results from the previous experiment we realized that what evolution found was that laying many blocks on the ground was enough to get a high fitness: the average speed was decreased and it will place unstable blocks to cover gaps in Y-axis. Even if those blocks just free-fall and break, a high amount of static block will compensate. In order to correct this behaviour we changed the fitness function to take into account the fastest moving object. That way, other blocks will not mask the fact that there are unstable elements being placed in the level. Additionally, we initialized levels including one of a list of pre configured blocks (disposed as in [5]) in addition to the random initialization used until now.

$$fitness_{indV2} = \max(V)$$

Again, with a different fitness function we cannot compare the fitness value with the rest of the experiments. One of the results shown in Figure 2. The results were in general more stable and complex than the ones obtained with the previous fitness function.

3 CONCLUSIONS AND FUTURE WORK

This paper was developed with objective of exploring the expressiveness and variability of SBPCG with evolutionary techniques and producing stable structures under gravity.

For this aim we have implemented an Evolutionary Algorithm able to optimize game level structures to meet this criteria. The method studied was sufficiently general and flexible to draw some conclusions about the topic. SBPCG methods are a potential good solution to offline content generation but it requires a great amount of problem-specific knowledge. The more rules the author adds, results tend to be variations of the same idea. However, a lack of knowledge will lead to unexpected outcomes. In our case, the fitness function used the stability of the structure and only considered other features to ensure the levels would be valid.

The main issue is how we define levels and how the definition plays with the paths of evolution. Producing stable structures under gravity was effectively achieved, but the consequence of evolving in a path of minimum movement results in squat structures that are not playable, although undeniably sturdy and stable. When we also consider height in the definition, stable structures were harder to find. An important element was missing in the experiments presented here and it was the ability of the blocks to break during simulation. Since Box2D only simulates physics, the blocks did not have *life points* like in the original game, so we could not penalize these levels like we did in [3]. As a result, some solutions obtained a good fitness value but will not be considered valid levels. A relevant improvement would be adding this *durability system* to the physics simulation in order to discard invalid levels.

Next step would be treating this as a multiobjective optimization problem: stability is the first, but we can sacrifice a bit of stability for height or some other aesthetic quality.

To conclude on an optimistic tone, this work provides an interesting insight into the SBPCG, through the completion—and failure—of the goals we set out to achieve at the beginning.

ACKNOWLEDGMENTS

This paper has been supported in part by DeepBio (TIN2017-85727-C4-2-P) from the Ministerio de Economía y Competitividad in Spain.

REFERENCES

[1] Laura Calle. [n. d.]. Angry Birds Level Generator Github repository. https://github.com/Laucalle/AngryBirdsLevelGenerator. ([n. d.]). Accessed: 2018-10-30.
[2] Laura Calle. [n. d.]. Science Birds adaptation. https://github.com/Laucalle/Science-Birds. ([n. d.]). Accessed: 2019-04-17.
[3] Laura Calle, Juan Julián Merelo Guervós, Antonio Mora García, and José Mario García Valdez. 2019. Free Form Evolution for Angry Birds Level Generation. In *Applications of Evolutionary Computation - 22nd International Conference, EvoApplications 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24-26, 2019, Proceedings (Lecture Notes in Computer Science)*, Paul Kaufmann and Pedro A. Castillo (Eds.), Vol. 11454. Springer, 125–140. https://doi.org/10.1007/978-3-030-16692-2_9
[4] Erin Catto. 2011. Box2D: A 2D Physics engine for games. (2011).
[5] Lucas Ferreira and Claudio Toledo. 2014. A search-based approach for generating angry birds levels. In *Computational intelligence and games (cig), 2014 ieee conference on*. IEEE, 1–8.
[6] Matthew Stephenson and Jochen Renz. 2016. Procedural generation of complex stable structures for angry birds levels. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 1–8.