Monsters of Darwin: a strategic game based on Artificial Intelligence and Genetic Algorithms

Daniele Norton
University of Milan
Milan, Italy

Laura Anna Ripamonti
University of Milan
Milan, Italy
ripamonti@di.unimi.it

Mario Ornaghi
University of Milan
Milan, Italy
ornaghi@di.unimi.it

Davide Gadia
University of Milan
Milan, Italy
gadia@di.unimi.it

Dario Maggiorini
University of Milan
Milan, Italy
dario@di.unimi.it

ABSTRACT

The production of video games is a complex process, which involves several disciplines, spanning from art to computer science. The final goal is to keep entertained the players, by continuously providing them novel and challenging contents. However, the availability of a large variety of pre-produced material is often not possible. A similar problem can be found in many single-player game genres, where the simulated behaviour generated by the Artificial Intelligence algorithms must be coherent, believable, but also adequately variegate to maintain a satisfactory user experience. To this aim, there is a growing interest in the introduction of automatic or semi-automatic techniques to produce and manage the video game contents. In this paper, we present an example of strategic card battle video game based on the applications of Artificial Intelligence and Genetic Algorithms, where the game contents are dynamically adapted and produced during the game sessions.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; I.2.1. Applications and Expert Systems: Games; K.8.0. General: Games

Author Keywords

Strategic Games; Video games; User experience; Artificial Intelligence for video games; Genetic Algorithms

INTRODUCTION

The production of video games is a complex process, which involves several disciplines, spanning from art to computer science. Regardless of the specific game genre, the design and development processes must have as final goal the production of a video game able to propose an enjoyable gaming experience even after long periods of time.

GHITALY17: 1st Workshop on Games-Human Interaction, September 18th, 2017, Cagliari, Italy.

The analysis and design of the overall user experience in a video game is not a straightforward task, because several factors can play a role. Recently, there has been a growing interest in the analysis of a number of features that, even if considered apparently of minor importance, could, nonetheless, impact deeply on players' user experience with the game [2, 4, 14, 15]. In particular, in some video game genres, one of the problematic features is that after spending a certain amount of time playing, players become well aware of the overall characteristics of the game and of its contents, and consequently they begin to lose interest [13]. The solution would be to continuously provide novel and challenging contents, and to dynamically adapt the game behaviour, in order to maintain the game enjoyable and fun. To this aim, a larger number of video games are introducing Procedural Content Generation (PCG) techniques, together with Artificial Intelligence (AI) methods specifically developed to dynamically change and adapt the game contents and behaviour.

Procedural Content Generation (PCG) is an approach with long and established history in different fields of Computer Graphics, aimed at creating data algorithmically. Examples of procedurally generated contents are textures [5] , buildings and cities [27, 28]. In the context of video games, PCG can be exploited to increase randomness of content and/or gameplay, with also the positive effects of reducing development time. For example, it can be used to automatically create a game level for platform games to achieve a desired level of complexity [26]. In other works, the application on PCG and AI techniques has been applied to several specific aspects of game optimization, such as: impact of Non Player Characters (NPCs) ([1, 18, 25, 31, 33]), and adaptive or personalized content generation ([10, 32]). Finally, there is a large literature related to the applications of Genetic Algorithms (GAs) in video games. The proposed techniques are mainly used to generate or evolve the game environment [3, 7, 24, 29], or for the evolution of the game agents behaviour, in order to produce more challenging opponents to the players [6, 11, 12, 21, 22, 23].

Differently from the other works on GAs in video games, the recently proposed GOLEM (Generator Of Life Embedded into MMOs) algorithm [9] addresses explicitely the need to introduce more variety and unpredictability in the monsters inside MMOs, in order to avoid that the players consider the

game repetitive and less enjoyable after a long period of time spent playing. The main idea in GOLEM is to characterize each monster specie present in the game through its genome, and to generate new species by recombining their chromosomes, which represent a set of physical characteristics and skills. Each monster in GOLEM is represented by a chromosome composed by 53 genes, and the recombination process evaluates also the probability for the new monster to actually survive in the habitat in which it is born (e.g., a marine-like animal will unlikely survive in a desert). Moreover, other parameters are included in order to manage and, if needed, limit the population growth.

The concept of evolution of a population of creatures have been addressed also in a small number of commercial video games [8, 30], even if in a very different way than the approach proposed in GOLEM.

In this paper, we present a strategic card battle video game called Monsters of Darwin (MOD). The game applies the GA approach of GOLEM, but to a different game genre: in this case, the monsters are depicted on virtual cards, and at each turn, a new card with a new monster may be generated by the combination of the monsters on the cards currently played. Moreover, being MOD a single-player video game, an AI component has been implemented, to manage the game dynamics and to act as Non Playable Character (NPC).

The paper is organized as follows: in the following sections we will recall the fundamental concepts related to AI and GA in video games, and then we will describe MOD design and implementation choices. Finally, we will draw conclusions and discuss major future developments.

ARTIFICIAL INTELLIGENCE IN VIDEO GAMES

Contemporary video games are often characterized by the use of advanced AI techniques. AI solutions for video games [17] are quite different from those typical of *classic* AI, because while the latter are usually aimed at optimizing solutions for specific problems under no time or resources limits, the former must provide a believable solution to complex decisional processes in (nearly) real-time with limited computational resources. AI in video games has a huge amount of possible usages, mostly depending on the specific game genre. One example is represented by Non-Playing Characters (NPCs) in First Person Shooter (FPS) games, where the AI is responsible of the simulation of a believable behaviours for a group of enemies [16, 25].

In this paper, we consider a strategic video game. The video games in this genre are characterized mainly by actions based on tactics and planning, in order to achieve victory. Player's decisions have a key role in the game, while chance is less relevant than players' ability. There are two sub-genres, *turn-based* or *real-time*, and the role of AI can vary relevantly depending on the nature of the strategic video game. *Real-time* strategic games have usually a simpler AI, which is mainly used as a *strategic layer*. On the contrary, *turn-based* games may have very sophisticated and complex AI, which may even run on dedicated hardware, like in case of simulated complex board games (e.g., Chess) [17].

The characterization of an AI algorithm for strategic video game shares its terminology with the *game theory*. A game is classified according to the number of players (usually two, even if it can be extended to a higher number of players), the goal of the game, and the information each player has about the game. With respect to the goal, two subclasses of strategic games exist:

- *Zero-sum game*: a player wins only if the other opponent lose.
- *Non zero-sum game*: the focus of the player is on the winning, which may happen even without the opponent's loss.

Moreover, we can identify two cases also for the level of information the players have regarding the game:

- *Perfect information*: the player knows everything about the state of the game, and about the possible options the opponent has for the next move.
- *Imperfect information*: there is some random element in the game that make uncertain the possible evolution of state of the game.

The most popular strategic video game AI technique is the *minimax* algorithm. The main concept is that, during a turn-based game, a player tries to play the best move possible to achieve victory, while the opponent tries to use the best strategy to avoid this, by minimizing the player's score. The *minimax* algorithm is a recursive algorithm, which works on a tree data structure (the *game tree*) where each node represents a board position, and each branch represents one possible move, leading from one board position to another. At each iteration, the AI algorithm evaluates the current state, considers each possible move and corresponding new board positions, and recursively calculates the score for each new possible situation, in order to choose the most appropriate one. The choice is performed using a heuristic called *static evaluation function* [17].

GENETIC ALGORITHMS IN VIDEO GAMES

GAs are a particular class of algorithms, mainly belonging to the AI field. They are applied to solve many classes of problems but, more in general, they are useful in heuristic search processes and in optimization problems. They have been inspired by Darwin's evolution theory: a population is represented by the chromosomes of a set of individuals, and a new generation in the population is produced by recombining the genetic material according to specific rules. For each generation, a *fitness* function selects the most "suitable" parents and iterates the reproduction process on them. To produce "children", the algorithm applies *crossover* (a genetic recombination technique) and *mutations* on chromosomes, represented by bit sequences. The algorithm then evaluates, once the new generation has been created, if the population registers any improvement in any relevant feature: if this is the case, parents will be discarded and the new children will substitute them in the reproduction process. Generally, these steps are iterated until some optimal solution is reached [19, 20].

Crossover is used by GAs to mix the genes of two elements of the population. Some different approaches can be applied:

1. *Single point crossover*: the chromosomes of the parents are split into two parts in a randomly selected point. Two new chromosomes are generated, by combining the first and second parts of the original parents.

2. *Two points crossover*: the chromosomes of the parents are split into three parts. The chromosome of one child is composed by the first and third part of the first parent, merged with the second part of the second parent. Another chromosome is created using the remaining parts.

3. *Uniform crossover*: this approach provides a higher genetic variation, since each gene of the child is copied randomly from one of the corresponding genes belonging to one of the parents. The genes not chosen for the first child are used for the second one

4. *Arithmetic crossover*: the offspring chromosomes are the results of some arithmetic operation on the parents' genes.

Finally, genetic *mutations* can be useful for inserting into the new generation's chromosomes some characteristics not inheritable from parents, since not present in their genetic heritage. Similarly to what happens in nature, mutations can introduce a new characteristic or modify/destroy an existing one.

MONSTERS OF DARWIN (MOD)

Monsters of Darwin has been developed in order to investigate novel approaches aimed at maximizing the user experience in a particular game genre that, after long periods, may suffer because of the repetition of game contents.

MOD is a *zero-sum*, *imperfect information*, *turn-based* single-player strategic card battle video game. The player plays against an AI opponent in order to collect the highest possible number of *Monster Cards*.

The game dynamics of MOD are quite simple. At each turn, two cards are played, one from the player on turn (the "real" player or the AI component) and one from the opponent. Then, the player on turn must choose between two different actions among the two cards: *Monster Duel* or *Monster Coupling*. *Monster Duel* follows rules similar to other card battle games, while *Monster Coupling* represents a procedural approach aimed at lower the game contents repetitivity, by the automatic generation of new Monster Cards depicting new monsters. *Monster Coupling* is based on the application of a GA, derived from the GOLEM algorithm [9]. A game session ends when a player defeats all the cards of the opponent. The final winner is the first player winning 3 game sessions.

In the following subsections, we provide further details on the features and contents of MOD.

Monster Cards

The main contents of MOD are the Monster Cards, depicting different kinds of monsters, using a graphic style inspired by grimoires and medieval manuscripts. Figure 1 shows an example of Monster Card. Each card is composed by three areas: in the bottom area it is indicated the *natural element* of the monster; in the central area it is depicted the monster, and, finally, the *attack*, *defense* and *vital forces* of the monster are shown in the top area.

Figure 1. An example of Monster Card. In the bottom area it is indicated the *natural element* of the monster. The *attack*, *defense* and *vital forces* of the monster are shown in the top area.

The four *natural elements* (*air*, *earth*, *water* and *fire*) are used to create a hierarchy between the Monster Cards, used during the *Monster Duel* or *Monster Coupling* stages. In particular:

- *air* rules over *earth*
- *earth* rules over *water*
- *water* rules over *fire*
- *fire* rules over *air*

To allow the graphical representation of the new monsters generated by the GA, each monster is composed by a set of 2D patches, each having a texture of a different physical part (head, body, legs, etc). With a similar approach, a new monster can be simply represented by assembling patches from different sets. Figure 2 shows a monster, and the patches composing its graphical representation.

At the beginning of the game, the player has a card deck composed by 8 Monster Cards (2 for each natural element). In the following turns, the player will choose 8 cards among those available in her deck. The AI module creates its deck choosing the Monster Cards randomly, even if balancing among the number of cards belonging to the different natural elements.

Monster Duel

During a *Monster Duel*, the damage a Monster Card can inflict to the *vital force* of the opponent's card is established by the difference between the corresponding *attack* and *defense force* values (e.g., a card with an *attack force* of 8 will inflict a damage of 2 to a card with *defense force* 6). The damage can be modified on the basis of their relation in the *natural elements* hierarchy: the damage is raised of 1 unit in case the element of the attacking card rules over the element of the other card, otherwise it is lowered of 1 unit. All the Monster Cards have an initial *vital force* equal to 6. When the value reaches 0, the Monster Card is removed by the current game session, and it will become available only in the following games.

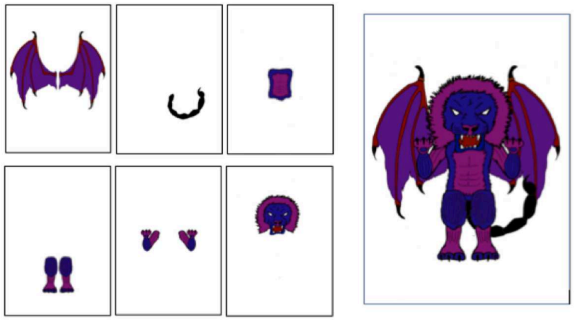

Figure 2. A monster of MOD, and the patches composing its graphical representation

Characteristic	*Type*	*Range*
Head	physical	8
Eyes	physical	8
Arms	physical	8
Body	physical	8
Legs	physical	8
Tail	physical	8
Wings	physical	8
Attack	force	3
Defense	force	3
Vital	force	3
Element	natural element	4

Table 1. Characteristics and skills in a chromosome of a monster.

Monster Coupling and Genetic Algorithm in MOD

The GA implemented in MOD is applied during the *Monster Coupling* stage. As in GOLEM [9], each monster is described by a chromosome, which maps its characteristics and skills. The characteristics are related to the *physical* aspect (e.g., number of legs, wings, etc.), to the *force* (i.e., the *attack*, *defense* and *vital forces*), and to the *natural element*, as illustrated in Table 1.

Each characteristic can assume a value in a certain range. For the physical features, the range from 1 to 8 indicates one Monster Card from the initial deck: therefore, a value of 2 for the *Head* means that the new monster will be created using the head patch from the monster of the second card selected in the deck, etc. For the force features, the value changes depending on the subcategories: *attack force* can have values from 4 to 7, *defense force* can have values from 1 to 3, *vital force* can have values from 0 to 6. These values have been chosen after a tuning and testing stage of the game.

The *Monster Coupling* stage begins evaluating the *compatibility* between the two Monster Cards currently played. Two monsters are evaluated as compatible for the coupling if the *natural element* of the card of the player on turn rules over the element of the opponent's card, and if the difference in *vital force* between the two monsters is equal of higher of 2. If these constraints are not valid, then the Monster Card of the player on turn (i.e., the one who has tried the *Monster Coupling*) will be inflicted a damage following the rules of the *Monster Duel* stage. Otherwise, the *Crossover* function of the GA algorithm is applied to the chromosomes of the two monsters. To obtain the broadest diversity among the generated monsters, we have opted for the *uniform crossover* approach: for each characteristic in the chromosome, one value from the two parents is randomly selected and assigned to the new monster. Even if the crossover creates two new monsters, only the first is considered, and it is added to the player's personal deck, while the two original cards subject to the coupling are discarded from the game session.

Artificial Intelligence in MOD

The main operations of the AI component in MOD are: the selection of the Monster Card to play, and the choice between *Monster Duel* or *Monster Coupling* actions. In both cases, the AI component uses the *minimax* algorithm for the evaluation. During the selection of the Monster Card to play, if the AI is the player on turn, the choice of the card is completely random. If, otherwise, the AI is responding to the player's card choice, than the algorithm evaluates the played card, and performs a selection of a "stronger" card among the available ones in the deck. A card is "stronger" if its *natural element* rules over the element of the first card, and/or its *attack force* is higher. During the choice between *Monster Duel* or *Monster Coupling*, the AI algorithm evaluation is based on the number of remaining Monster Cards in the deck, and on the difference of *vital force* and *compatibility* between the two Monster Cards.

MOD implementation details

MOD has been implemented using the Unity 3D game engine. For its characteristics, Unity 3D represents a powerful and flexible solution well suited for fast implementation of different game genres, allowing the integration of different techniques and material, and the development for different gaming platforms.

Figure 3 shows some screenshots of the *Monster Duel* and *Monster Coupling* stages.

CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a strategic card battle video game called Monsters of Darwin (MOD). The game uses a combination of AI and GAs in order to generate new *Monster Cards*, by combining the characteristics of the cards played in each turn.

The approach we are proposing with MOD is a preliminary attempt to address one of the features that can impact on players' user experience. In fact, repetition of contents and mechanics in some game genres can lead to a loss of interest in the game after some time. One possible solution to avoid this issue is to introduce procedural and dynamic techniques to continuously generate novel contents and to adapt the game behaviour to the players' actions, in order to maintain the game enjoyable and fun.

From the preliminary evaluation tests, the performances of MOD are promising: the GA in MOD is able to produce a

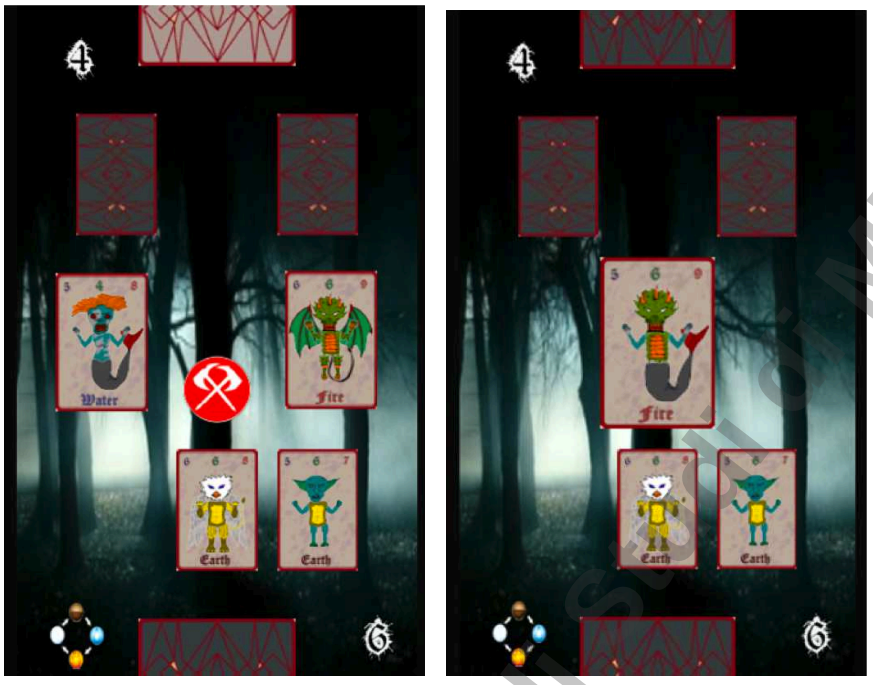

Figure 3. Screenshots of the *Monster Duel* and *Monster Coupling* stages in MOD.

high number of new monsters even from a limited set of basic Monster Cards, while the evaluation made by the AI algorithm is performed smoothly, limiting the waiting time to the minimum.

Future developments will consider the extension of the characteristics of the monsters chromosomes, and of the database of available monsters among which the players can choose the initial deck. Moreover, we aim at extending the AI component, by introducing new possible actions during the game turns, like e.g., a defense action to oppose the attack from the opponent player.

REFERENCES

1. Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. 2005. Automatic Computer Game Balancing: A Reinforcement Learning Approach. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*. ACM, New York, NY, USA, 1111–1112.
2. Richard Bartle. 2003. *Designing Virtual Worlds*. New Riders Games.
3. Leonardo F. B. Silva de Carvalho, Helio C. Silva Neto, Roberta V. V. Lopes, and Fábio Paraguaçu. 2010. An application of genetic algorithm based on abstract data type for the problem of generation of scenarios for electronic games. In *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, Vol. 2. 526–530.
4. Daniele De Felice, Marco Granato, Laura Anna Ripamonti, Marco Trubian, Davide Gadia, and Dario Maggiorini. 2017. Effect of Different Looting Systems on the Behavior of Players in a MMOG: Simulation with Real Data. In *eHealth 360°: International Summit on eHealth, Revised Selected Papers*. Springer International Publishing, 110–118.
5. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. 2003. *Texturing & Modeling: a procedural approach*. Morgan Kaufmann.
6. Anna I. Esparcia-Alcázar and Jaroslav Moravec. 2013. Fitness approximation for bot evolution in genetic programming. *Soft Computing* 17, 8 (2013), 1479–1487.
7. Miguel Frade, Francisco Fernandez de Vega, and Carlos Cotta. 2012. Automatic evolution of programs for procedural generation of terrains for video games. *Soft Computing* 16, 11 (2012), 1893–1914.
8. Galactic Arms Race homepage. 2017. (2017). http://galacticarmsrace.blogspot.it.
9. Andrea Guarneri, Dario Maggiorini, Laura Anna Ripamonti, and Marco Trubian. 2013. GOLEM: Generator Of Life Embedded into MMOs. In *Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems: Advances in Artificial Life, ECAL*. 585–592.
10. Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. 2009. Automatic Content Generation in the Galactic Arms Race Video Game. *IEEE Transactions on Computational Intelligence and AI in Games* 1, 4 (Dec 2009), 245–263.
11. Yuan Hong and Zhen Liu. 2010. A First Study on Genetic Algorithms Based-Evolvable Motivation Model for Virtual Agents. In *2010 International Conference on Multimedia Technology*. 1–4.

12. Johannes Inführ and Günther R. Raidl. 2012. Automatic Generation of 2-AntWars Players with Genetic Programming. In *Computer Aided Systems Theory – EUROCAST 2011: 13th International Conference, Revised Selected Papers, Part I*. Springer Berlin Heidelberg, Berlin, Heidelberg, 248–255.

13. Raph Koster and Will Wright. 2004. *A Theory of Fun for Game Design*. Paraglyph Press.

14. Dario Maggiorini, Alessandro Nigro, Laura Anna Ripamonti, and Marco Trubian. 2012a. Loot Distribution in Massive Online Games: foreseeing impacts on the player base. In *8th International Workshop on Networking Issues in Multimedia Entertainment (NIME'12) Int. Conf. on Computer Communication Networks ICCCN*.

15. Dario Maggiorini, Alessandro Nigro, Laura Anna Ripamonti, and Marco Trubian. 2012b. The Perfect Looting System: Looking for a Phoenix?. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. 371–378.

16. Dario Maggiorini, Laura Anna Ripamonti, and Samuele Panzeri. 2013. Follow the Leader: a Scalable Approach for Realistic Group Behavior of Roaming NPCs in MMO Games. In *Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems: Advances in Artificial Life, ECAL 2013, Sicily, Italy, September 2-6, 2013*. 706–712.

17. Ian Millington and John Funge. 2009. *Artificial Intelligence for Games, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

18. Olana Missura and Thomas Gärtner. 2009. Player Modeling for Intelligent Difficulty Adjustment. In *Discovery Science: 12th International Conference*. Springer Berlin Heidelberg, Berlin, Heidelberg, 197–211.

19. Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. The MIT Press.

20. Charles. J. Mode and Candace K. Sleeman. 2012. *Stochastic Processes in Genetic and Evolution: Computer Experiments in the Quantification of Mutation and Selection*. World Scientific Publishing Company.

21. Antonio M. Mora, Antonio Fernández-Ares, Juan J. Merelo, Pablo García-Sánchez, and Carlos M. Fernandes. 2012. Effect of Noisy Fitness in Real-Time Strategy Games Player Behaviour Optimisation Using Evolutionary Algorithms. *Journal of Computer Science and Technology* 27, 5 (2012), 1007–1023.

22. Antonio M. Mora, Ramón Montoya, Juan J. Merelo, Pablo G. Sánchez, Pedro Á. Castillo, Juan L. J. Laredo, Ana I. Martínez, and Anna Espacia. 2010a. Evolving Bot AI in Unreal™. In *Applications of Evolutionary Computation: EvoApplicatons 2010*. Springer Berlin Heidelberg, Berlin, Heidelberg, 171–180.

23. Antonio M. Mora, Mária A. Moreno, Juan J. Merelo, Pedro A. Castillo, Maribel G. Arenas, and Juan L. J. Laredo. 2010b. Evolving the cooperative behaviour in Unreal™ bots. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. 241–248.

24. Fausto Mourato, Manuel Próspero dos Santos, and Fernando Birra. 2011. Automatic Level Generation for Platform Videogames Using Genetic Algorithms. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology (ACE '11)*. ACM, New York, NY, USA, Article 8, 8 pages.

25. Laura Anna Ripamonti, Silvio Gratani, Dario Maggiorini, Davide Gadia, and Armir Bujari. 2017a. Believable group behaviours for NPCs in FPS games. In *Proceedings of IEEE Digital Entertainment, Networked Virtual Environments, and Creative Technology Workshop (DENVECT 2017)*.

26. Laura Anna Ripamonti, Mattia Mannalà, Davide Gadia, and Dario Maggiorini. 2017b. Procedural content generation for platformers: designing and testing FUN PLEdGE. *Multimedia Tools and Applications* 76, 4 (2017), 5001–5050.

27. Manlio Scalabrin, Laura Anna Ripamonti, Dario Maggiorini, and Davide Gadia. 2016. Stereoscopy-based procedural generation of virtual environments. In *Proceedings of IS&T's Stereoscopic Displays and Applications XXVII (28th Symposium on Electronic Imaging : Science and Technology)*. 1–7.

28. Michael Schwarz and Pascal Müller. 2015. Advanced Procedural Modeling of Architecture. *ACM Trans. Graph.* 34, 4, Article 107 (July 2015), 12 pages.

29. Nathan Sorenson and Philippe Pasquier. 2010. Towards a Generic Framework for Automated Video Game Level Creation. In *Applications of Evolutionary Computation: EvoApplicatons 2010*. Springer Berlin Heidelberg, Berlin, Heidelberg, 131–140.

30. Spore homepage. 2017. (2017). http://www.spore.com.

31. Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. 2006. Adaptive Game AI with Dynamic Scripting. *Mach. Learn.* 63, 3 (2006), 217–248.

32. Julian Togelius, Renzo De Nardi, and Simon M. Lucas. 2007. Towards automatic personalised content creation in racing games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games. Togelius is working at IDSIA on SNF grant 21-113364 to J. Schmidhuber.*

33. Georgios N. Yannakakis and John Hallam. 2009. Real-Time Game Adaptation for Optimizing Player Satisfaction. *IEEE Trans. Comput. Intellig. and AI in Games* 1, 2 (2009), 121–133.