Chapter 1 - Introduction

Diego Perez-Liebana

The General Video Game AI (GVGAI) framework and its associated competition have provided to the AI research community with a tool to investigate General AI in the domain of games. For decades, games have been used as benchmarks to perform AI research: they are fast and cheap simulations of the real world and (this can't be overlooked) they are fun. Soon thereafter, work in games led to establishing comparison of AI performance among researchers, which subsequently took investigators and practitioners to establish competitions around them.

Game-based AI competitions are a very popular and fun way of benchmarking methods, becoming one of the most central components of the field. Research around competitions provides a common benchmark in which researchers can try their methods in a controlled, fair and reliable manner, opening the possibility of comparing multiple approaches within exactly the same environment. Using games for competitions allows researchers to show the results in a way most people can understand and interpret, in a domain that is typically challenging for humans as well, helping raise awareness of AI research among different groups and communities. Either as vehicles for creating game playing agents or generation of content, participants submit controllers or generators which are evaluated in the same conditions to proclaim a winner.

Without any doubt, advances in research have been achieved when working in one game only (such as Transposition Tables and Iterative Deepening from Chess, or Monte Carlo Tree Search from Go), but it is also true that an agent that performs well at one of these contests would likely not perform above chance level in any of the others (even if the same interface was provided). The initial thought that an AI that beats humans at Chess would achieve general intelligence needed to be evaluated again: single-game competitions have the drawback of overfitting a solution to the game used. Then, the value of the competition as a benchmark for general AI methods is limited: participants will tend to tailor their approaches to the game to improve their results. The results of the competition can therefore show the ability of the contestants to engineer a domain specific solution rather than the quality of the AI algorithm that drives the competition entry. For example, participants of the Simulated Car Racing Competition have gotten better at lap times during the more than five years this contest took place, but the presence of game-specific engineering arguably made the approaches less general. Transferring the algorithms

and learning outcomes from this competition to other domains has become more and more difficult. It is sensible to think that a competition designed with this in mind would allow us to learn more about artificial *general* intelligence.

The present book details the efforts of GVGAI as a framework (and competition) designed with the intention of promoting research focused on not one but many games. This is done in a series of different chapters that cover the distinct angles of GVGAI.

- Chapter 1 (the one you are reading) introduces GVGAI, from its predecessors until its origins, and provides a vision of the uses of this framework and competition in education, as well as the potential impact it may have in the games industry.
- Chapter 2 describes the Video Game Description Language (VGDL) used to implement games and levels, as well as how the GVGAI framework works and its associated competitions are organised.
- The first one of the more technical chapters is number 3, which focuses on the planning problem tackled in GVGAI. In this case, agents are asked to play any game is given with the aid of a Forward Model (FM), a simulator that allows agents to play-test moves during their thinking time. The main two methods that this chapter presents are Monte Carlo Tree Search (MCTS) and Rolling Horizon Evolutionary Algorithms (RHEA), as well as several enhancements of these.
- The planning problem proposed for GVGAI is the one that has received more attention. Due to this, Chapter 4 introduces the state-of-the-art controllers that have achieved better results in the competition during the years. It also discusses what are the main hazards and open problems of this challenge, to then propose avenues of future work in this area.
- Chapter 5 discusses the use of GVGAI in a learning setting. In this case, the game playing agents must learn how to play a given game by repeatedly playing the same game during a training phase, this time without the aid of a Forward Model. This chapter describes the particularities of the framework for this end, as well as its integration with Open AI Gym, facilitating the use of reinforcement learning algorithms for this challenge.
- GVGAI has also applications in Procedural Content Generation (PCG). The framework and competition allow the implementation of rule and level generators. Chapter 6 describes these two aspects of the framework, explaining how can these generators be created and what are the current approaches in this area.
- Chapter 7 focuses on Automatic Game Tuning within GVGAI. This chapter explores how VGDL games can be parameterised to form a *game space*: a collection of possible variants of the same game that could include hidden gems or a perfect

gameplay balance from a design point of view. An evolutionary algorithm (the N-Tuple Bandit Evolutionary Algorithm; NTBEA) is presented and used to tweak variables of several games in order to obtain variants that serve a pre-determined purpose.
- Chapter 8 introduces a new concept to GVGAI, which is the usage of games and AI agents described in a high-level language that are compatible with games and agents from GVGAI. This chapter describes what is needed to have games that support a Forward Model in an efficient way, as well as presenting a common interface to facilitate the compatibility between systems, illustrated with concrete examples.
- Finally, Chapter 9 concludes this book by asking 'What's next?', outlining outstanding research with the current challenge, but also the possibilities of the framework and competition to keep proposing new and interesting problems.

This books brings an overall perspective on the motivation, framework, competition and related research work that forms the GVGAI world. It also provides insights about how the framework has been used in Education during these years and how it can be useful for the Games Industry. Last but not least, each technical chapter proposes, at the end, a series of exercises for those who want to dive deeper in this framework and the challenges it proposes. All these exercises can also be found at the book's website: `https://gaigresearch.github.io/gvgaibook/`.

1 A Historical View: from Chess to GVGAI

The utilisation of games as benchmarks for AI research is as old as the field itself. Alan Turing proposed Chess as an AI benchmark and studied how to play this game with a manual execution of Minimax [38]. Inaugurated in the 1970s, the World Computer Chess Championship has systematically compared AI algorithms in this game until this day [22]. Since IBM's *Deep Blue* defeated the best human player at its time, Garry Kasparov, the championship has continued pitting computer against computer in a chess AI arms race. Nowadays, we are living through the same history again with the game Go, after Deepmind's *AlphaGo* [35] beat the World Go Champion Lee Sedol in 2017, or the first Starcraft II professionals in 2019 [3].

Since the start of the 21^{st} century, many competitions and benchmarks for video games have come to existence. The nature of the games that form the basis of these competitions is incredibly varied. Some competitions focus on first-person shooter games, such as Unreal Tournament 2004 [11] and VizDoom [13]; platform games such as Super Mario Bros [37] and Geometry Friends [27]; car racing, such as TORCS [16];

classic arcade games, such as Ms. Pac-Man [28] real-time strategy games, such as StarCraft [24] and Starcraft II [40]; and navigation problems such as the Physical TSP game and competition [25]. The large majority of the existent competitions, however, focus on a single game.

In the study of Artificial General Intelligence (AGI), a predominant idea is that the intelligence of an agent can be measured by its performance across a large number of environments. Schaul et al. [32] formalized this idea by arguing that these environments could be games sampled from a game space. Game spaces can be defined via a Game Description Language (GDL) and an engine that parses and executes it in a playable way.

Many authors have developed GDLs to define at different levels of abstraction. Examples are the Ludi language for combinatorial board games [2] and the Card Game Description Language [5]. An overview of several GDLs and representations for game rules and mechanics can be found in [19].

However, the first attempt at finding a remedy to this problem via a GDL was done by the Stanford University Logic Group in 2005 [9], when they ran the first *General Game Playing Competition* (GGP). In this contest, competitors submit agents that can play a number of previously unseen games, usually turn-based discrete games or variants of existing board games. Games were described in a GDL [17] and could be single or two-player deterministic games (Figure 1 shows a few examples of existing games). GDL is based on first-order logic (syntactically, it's a version of Prolog), rather low-level. For instance, the description of a game like Tic-Tac-Toe is several pages long. GGP agents have approximately 1 second of thinking time to make a move and the description of the game is given to the players so it can be analysed.

Fig. 1: Example games from the General Game Playing Competition.

Another popular environment is the Arcade Learning Environment (ALE), which is based on an emulator of the Atari 2600 game console [1]. This collection of video-games include simple-looking (but some of very high quality, such as Pac-Man or Space Invaders) games. Two of these games can be seen in Figure 2.

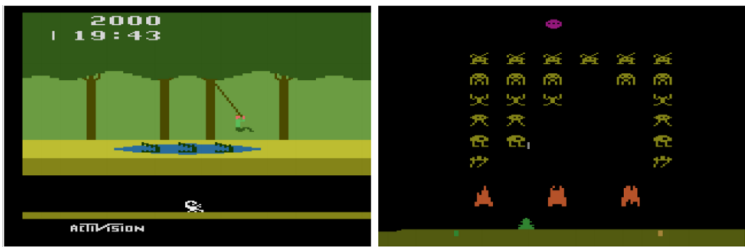

Fig. 2: Example games from the Arcade Learning Environment: *Pitfall* (left) and *Space Invaders* (right).

Agents developed for playing ALE games are meant to be able to play any game within this framework in real-time, providing actions every few milliseconds. In recent years, research with this framework has become increasingly popular, especially using deep learning to train agents that receive the raw screen capture as input, plus a score counter [18].

In most cases, the algorithms learn to play one ALE game at a time, having to train again when the environment is different and forgetting what was learnt for the first game. Multi-task and transfer learning are one of the main focuses of research in this type of problems at the moment [42]. Due to the nature of the Atari emulator, all games in ALE are deterministic. This limits the adaptability of the methods trained to play these games and makes them too brittle when facing changes in the environment. Additionally, games are loaded via ROMs instead of being defined in a GDL, so they can't be modified easily.

Another fruitful area of research for multi-task learning is that of multi-agent. This domain does not only have the challenge of learning in many games at the same time, but also dealing with opponent modeling. The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) framework and competition is a new challenge that proposes this type of research in the game Minecraft [26]. The MARLÖ competition ran for the first time in November/December 2018 and it proposes multi-task, multi-agent learning via screen capture through the OpenAI Gym interface. Figure 3 shows the three games that were used in this challenge.

Fig. 3: MARLÖ 2018 Competition games. From left to right, *Mob Chase* (two players collaborate to catch a mob in a pin), *Build Battle* (two players compete to build a structure that matches a model) and *Treasure Hunt* (where two players collaborate in collecting treasures in a maze while avoiding enemy zombies).

MARLÖ proposes a similar task to the one tackled in ALE, but stepping up the game by using 3D environments and a more general approach. Each one of these games can be parameterised, so blocks, objects and character appearance can vary from instance to instance (also known as *tasks*). Game settings and tasks are described in XML files and the final rankings of this challenge are computed in task a priori unknown by the participants.

Ebner et al. [4] and Levine et al. [15] described the need and interest for a framework that would accommodate a competition for researchers to tackle the challenge of General Video Game Playing (GVGP). The authors proposed the design of the Video Game Description Language (VGDL), which was later developed by Schaul [30], [31] in the Python framework `py-vgdl`. VGDL was designed so that it would be easy to create games both for humans and algorithms, eventually allowing for automated generation of test-bed games. VGDL describes real-time arcade games with stochastic effects, hidden information and played by an avatar. VGDL is a language that allows the creation of a potentially infinite number of games in a very compact manner, describing them in fewer lines than GGP's GDL and MARLÖ's XML format.

VGDL is the base of the General Video Game AI (GVGAI) Framework and Competition, subject of this book. GVGAI was developed as a new iteration of `py-vgdl` in Java, exposing an API for agents to play games defined in VGDL. Researchers should develop their agents without knowing which games would they be playing. A competition server was made be available so participants could submit their bots and be evaluated in an unseen set of games. In order to make games more appealing to human players, a special care was put into providing a nice set of graphics and sprites within the framework. One of the still standing goals of GVGAI is to be able

to compare bot and human performance, and even creating environments in which human and AIs can collaborate and compete in GVGP. Figure 4 shows the evolution of one of the VGDL/GVGAI games, Pac-Man, through the different versions of the framework.

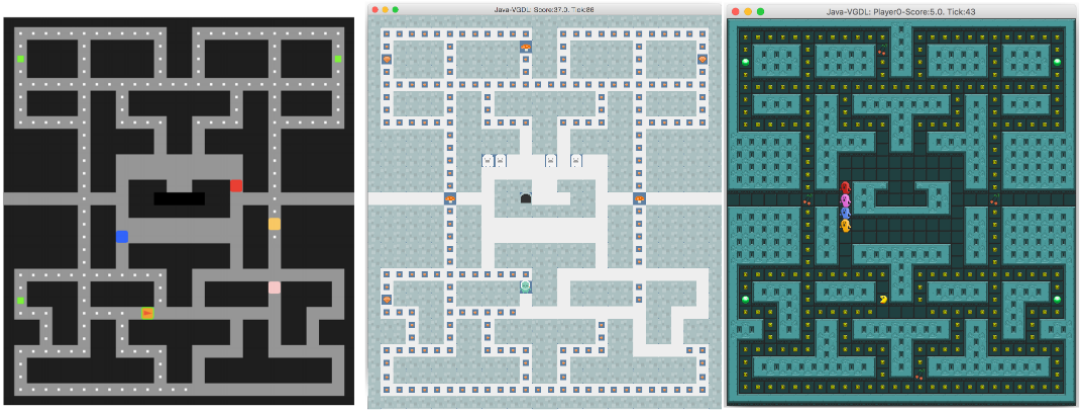

Fig. 4: Evolution of Pac-Man in `py-vgdl` and GVGAI frameworks. From left to right, Pac-Man in py-vgdl (with coloured squares), initial (with simple sprites) and current GVGAI framework (with sprites made by a graphic designer[1], with support for transparencies, animations and background auto-tiling).

Initially, GVGAI was mainly focused on proposing a competition for single-player game agents, in which controllers have access to the model of the game (albeit not the VGDL description) promoting research in model-based reinforcement learning algorithms. Shortly after, VGDL was expanded to account for 2-player games, adding the challenge of opponent player modelling in GVGP planning. The learning track of the GVGAI competition completes the available settings for general game playing agents. In this last scenario, no Forward Model is provided, challenging model-free reinforcement learning methods that would learn from static game observations, screen capture, or both.

The framework was also expanded to exploit the versatility of VGDL by proposing challenges on procedural content generation (PCG). Two extra tracks dedicated to the generation of game rules and levels for any game within the framework have been recently created. Another potential use of GVGAI is for game prototyping, and

[1] Oryx Design Lab, `https://www.oryxdesignlab.com/`

there is a growing body of research using the framework for game design and the implementation of mixed-initiative design tools.

GVGAI has been used in multiple research domains, and this is the focus of the present book. Nevertheless, GVGAI has had an important impact in Education, being used in undergraduate and master taught modules as assignments, final year undergraduate and master projects and PhD topics. We also strongly believe that GVGAI can also have a great impact in the games industry. We conclude this chapter highlighting these impact cases, before diving into the research usages of the framework and competition.

2 GVGAI in Education

The GVGAI framework has been employed by instructors from many institutions around the globe to set up engaging assignments in their taught modules. Projects around the GVGAI framework and competition have been proposed for undergraduate and master thesis and PhD topics. This section describes some of these usages, although it does not intend to be an exhaustive list.

The GVGAI framework has been used in at least two different ways in taught modules. The most common usage of the framework is around the game playing challenge for single and 2-player games. Students are first taught of the main GVGAI concepts to then explore how AI algorithms can be implemented in an assignment. These agents, sometimes developed individually and others in groups, participate in either a private league or in the overall competition. The assignment's mark can include how well the entry of each student or group performs in the mentioned league. The following is a not-exclusive list of the institutions that, in the knowledge of the authors, have used the GVGAI framework in their taught modules:

- Otto Von Guericke Universität, Magdeburg, Germany.
- University of Essex, Colchester, United Kingdom.
- University of Muenster, Muenster, Germany.
- Universidad Carlos III de Madrid, Madrid, Spain.
- Universidad de Málaga, Málaga, Spain.
- New York University, New York, United States.
- Southern University of Science and Technology, Shenzhen, China.
- Nanjing University, Nanjing, China.

Some of these institutions have run private leagues (via the GVGAI website and server[2]). In a private league, the module supervisor has full control over how the

[2] `www.gvgai.net`

league is set up, including when and how students can submit entries, how are these evaluated, the set of games that are made available for the competition and, in the 2-player track, the available opponents within the league.

The game design and development capabilities of GVGAI have also been used in taught modules. In this case, VGDL has been explained as a high-level language to create games in. The objective of the assignments is the design and development of interesting games, either manually or guided by AI methods. Examples are creation of new puzzle games or exploring the parameter space of VGDL games (see Chapter 7). Some of the games created in this module have joined the corpus of VGDL games in the GVGAI framework. Examples of higher education institutions that have used the framework in this way are:

- IT University of Copenhagen, Copenhagen, Denmark.
- University of Essex, Colchester, United Kingdom.
- Queen Mary University of London, London, United Kingdom.

Some of the interesting research challenges GVGAI offers have been addressed in the form of master dissertation projects. Most of these have focused on the planning tasks (single and 2-player), which is not surprising given that this is the first track that became available. The GVGAI framework includes sample agents for all tracks, providing an ideal starting point for these projects. These can be used as baselines for comparison and/or as a starting point for algorithmic improvements. Our experience shows that this usage tends to provide an excellent educational experience for the student.

An illustrative example is the work by Maarten de Waard on Monte Carlo Tree Search (MCTS) with options, which showed how the use of options in MCTS outperformed the vanilla method in most of the games studied. This work started as an master project and was later published as a conference paper [41]. Other uses of GVGAI games in a planning setting for master thesis GVGAI include other enhancements of MCTS [33], real-time enhancements [36], MCTS knowledge-based[3] improvements [39] and goal-oriented approaches [29].

Episodic learning from screen capture was the focus of Kunanusont's master thesis [14] and the level generation track was the topic of two master projects: Neufeld [20], who applied Answer Set Programming and was also published in a relevant conference as a paper [21]; and Nichols [23], who used genetic algorithms for the first level generation competition. Last but not least, the master project by Gaina (one of the co-authors of this book) [6] expanded the framework itself to incorporate

[3] See Chapter 3 for the Knowledge-Based MCTS method this work is based on.

2-player games and run the first 2-player GVGAI competition [7]. This original work was also published as a paper in a conference [8].

Running the GVGAI competition (and others before) for several years and using the framework in our own taught modules has shown us that accessibility to a framework, documentation and the competitive element of competitions can motivate students to learn and propose new ideas for the problems tackled. The objective of this book is, apart from bringing the latest and most relevant research on GVGAI, to provide a resource for AI module supervisors with examples of possible projects for their students. As such, every technical chapter of this book concludes with a list of proposed exercises for students to attempt. Some of them can be seen as practical tasks, while others can spark new ideas for research projects at different levels.

3 GVGAI and the Games Industry

The original Dagstuhl report on General Video Game Playing stated that GVGAI games would have their "*unique story and goal which should be achieved by the player. The method of how the player interacts with the game is the same across all these games [...]. This will allow us to gain new insights in working towards the holy grail of artificial intelligence, i.e. development of human-like intelligence*" [15]. Although this may sound like a primarily research focused statement, we claim that research on General Video Game Playing can be fruitful for the games industry.

Initially, it is sensible to think that no company would include in their games in development a general agent that can play any game at an average performance level when they can include ad-hoc bots that fit perfectly the environment they are supposed to be immersed in. General agents do not necessarily address certain characteristics that these bots should have.

Namely, AI agents in games need to behave in a `specific` manner: they need to accomplish a certain task and do it at the level it is required to in order to provide a good player experience. They also need to be `predictable`: within reason, agents should behave in a way they are designed to. For instance, games where different types of enemies interact with the player in different ways should be consistent through the lifetime of the game according to certain design specifications. The player would not understand how a particular character that behaves, for example, recklessly during most of the game, exhibits a cautious and planned behaviour without any explainable reason for it. Finally, AI agents must perform their duties in an `efficient` manner. For this, they typically use scripted behaviour, objectives given by a designer and data structures or object representation built for the game being developed. A general

agent that aims not to use these resources may not perform as efficiently as one that does, both in terms of computational cost and actual efficacy of the task at hand.

Before you decide to throw this book to the bin, let us stop here and highlight why we think research on GVGP can be beneficial for the games industry. One of the main goals of GVGP (and, by extension, GVGAI) is to advance the state of the art on general algorithms. For instance, this research aims to improve the capabilities of algorithms that can work in *any* game, such as Monte Carlo Tree Search (MCTS) and Rolling Horizon Evolutionary Algorithms (RHEA). The vanilla versions of these methods, as used in GVGAI (see Chapter 3), do not incorporate any domain knowledge of the game. This means the algorithms do not have information about the goal of the game, the best strategy to win, not even details about what the game is about or what do other entities in it do. Therefore, we are aiming at improving the algorithm *itself*, not how it works in one particular game. We are focused on deviating the attention from algorithmic improvements in one game to advances for a broader set of domains. If improving on the reasoning and search capabilities of an algorithm is beneficial to *all* games, then it must be a step forward for *any* game. By doing this, GVGAI aims to provide a better starting point for those programmers interested in taking MCTS and adapt it for their particular game(s).

It is important to highlight that, when we refer to agents that play games, we may not necessarily aim at agents that play those games to win. For instance, agents could be trained to assist the player in any game within the 2-player GVGAI context, in order to identify behaviours for Non-Player Characters (NPC) via opponent modeling. But what if the agents are trained to do other things in any game? For instance, a general agent could be aiming at exploring the environment, or interacting with it, or even finding new and unexpected ways of solving a puzzle or a level. Or maybe you could have a system of agents in which each one of them is able to play with a different role [10]. A system of general agents that can do this could be used to perform automatic play-testing and Q/A for *any* game.

We are convinced that, while reading these lines, you are thinking of a particular game in which this can be applied. It is quite likely that the game you are thinking of is different to the one other readers have in mind. These ideas still apply, and this is the case because the concepts mentioned here are *general*. Could we build such system that works in many games? Could you then adapt it to work in *your* game?

One of the many different configurations that an agent could have is to play as closely as possible as a human would. Creating believable non-player characters is one of the standing challenges in games research and development [12]. This is a standing problem for not just one, but multiple games. Research can be focused on what are the concepts that make bots believable in general. We believe some of these

concepts cut across different games, certainly in the same genre. Researching how an agent can be believable in a group of games will advance the state of the art and provide valuable insights into how to make believable bots for particular games.

As mentioned above, these general agents can be used for automatic play-testing. There is only one step from here to build Procedural Content Generation (PCG) systems. If agents can evaluate games by playing them, they can be incorporated into an automatic system that generates new content. This approach, known as Relative Algorithm Performance Profiles (RAPP), is a simulated-based approach very common in the literature [34]. Generality in AI agents allows for generality in content generation. A system that is able to generate content (be this levels, mazes, items, weapon systems, etc.) for any game is given would be rich enough to be adapted to any particular game and provide a good starting point.

GVGAI takes on the ambitious goal of doing research for many games and from many different angles. We hypothesise that, by taking a step away from working on specific games, we are actually improving the state of the art in a way that is more valuable for more researchers and developers.

References

1. M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: an evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, no. 1, pp. 253–279, 2013.
2. C. Browne and F. Maire, "Evolutionary Game Design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 1–16, 2010.
3. G. Deepmind, "AlphaStar," 2019, https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.
4. M. Ebner, J. Levine, S. M. Lucas, T. Schaul, T. Thompson, and J. Togelius, "Towards a Video Game Description Language," *Dagstuhl Follow-Ups*, vol. 6, 2013.
5. J. M. Font, T. Mahlmann, D. Manrique, and J. Togelius, "A card game description language," in *Applications of Evolutionary Computing, EvoApplications 2013.*, ser. LNCS, vol. 7835. Vienna: Springer Verlag, 3-5 Apr. 2013, pp. 254–263.
6. R. D. Gaina, "The 2 Player General Video Game Playing Competition," Master's thesis, University of Essex, 2016.
7. R. D. Gaina, A. Couëtoux, D. J. Soemers, M. H. Winands, T. Vodopivec, F. Kirchgessner, J. Liu, S. M. Lucas, and D. Perez-Liebana, "The 2016 Two-Player GVGAI Competition," *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.
8. R. D. Gaina, D. Perez-Liebana, and S. M. Lucas, "General Video Game for 2 Players: Framework and Competition," in *IEEE Computer Science and Electronic Engineering Conference*, 2016.
9. M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the AAAI competition," *AI Magazine*, vol. 26, no. 2, p. 62, 2005.
10. C. Guerrero-Romero, S. M. Lucas, and D. Perez-Liebana, "Using a Team of General AI Algorithms to Assist Game Design and Testing," in *Conference on Computational Intelligence and Games (CIG)*, 2018.
11. P. Hingston, "A New Design for a Turing Test for Bots," in *Proceedings of the IEEE Conference on Computational Intelligence in Games*. IEEE, 2010, pp. 345–350.

12. ——, *Believable Bots: Can Computers Play Like People?* Springer, 2012.
13. M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based AI Research Platform for Visual Reinforcement Learning," in *Conference on Computational Intelligence and Games*. IEEE, 2016, pp. 1–8.
14. K. Kunanusont, "General Video Game Artificial Intelligence: Learning from Screen Capture," Master's thesis, University of Essex, 2016.
15. J. Levine, C. B. Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, and T. Thompson, "General Video Game Playing," *Dagstuhl Follow-Ups*, vol. 6, 2013.
16. D. Loiacono, P. L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. V. Butz, T. D. Lönneker, L. Cardamone, D. Perez, Y. Sáez *et al.*, "The 2009 Simulated Car Racing Championship," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 2, pp. 131–147, 2010.
17. N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth, "General Game Playing: Game Description Language Specification," 2008.
18. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
19. M. J. Nelson, J. Togelius, C. Browne, and M. Cook, "Rules and Mechanics," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, N. Shaker, J. Togelius, and M. J. Nelson, Eds. Springer, 2014, pp. 97–117.
20. X. Neufeld, "Procedural level generation with answer set programming for general video game playing," Master's thesis, University of Magdeburg, 2016.
21. X. Neufeld, S. Mostaghim, and D. Perez-Liebana, "Procedural level generation with answer set programming for general video game playing," in *Computer Science and Electronic Engineering Conference (CEEC), 2015 7th*. IEEE, 2015, pp. 207–212.
22. M. Newborn, *Computer chess*. John Wiley and Sons Ltd., 2003.
23. J. Nichols, "The Use of Genetic Algorithms in Automatic Level Generation," Master's thesis, University of Essex, 2016.
24. S. Ontanon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 4, pp. 293–311, 2013.
25. D. Perez, P. Rohlfshagen, and S. M. Lucas, "The Physical Travelling Salesman Problem: WCCI 2012 Competition," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.
26. D. Perez-Liebana, K. Hofmann, S. P. Mohanty, N. Kuno, A. Kramer, S. Devlin, R. D. Gaina, and D. Ionita, "The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition," in *Challenges in Machine Learning (CiML; NIPS Workshop)*, 2018, pp. 1–4.
27. R. Prada, F. Melo, and J. Quiterio, "Geometry Friends Competition," 2014.
28. P. Rohlfshagen and S. M. Lucas, "Ms Pac-Man Versus Ghost Team CEC 2011 Competition," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2011, pp. 70–77.
29. B. Ross, "General Video Game Playing with Goal Orientation," Master's thesis, University of Strathclyde, 2014.
30. T. Schaul, "A Video Game Description Language for Model-based or Interactive Learning," in *IEEE Conference on Computational Intelligence in Games (CIG)*, 2013, pp. 1–8.
31. ——, "An Extensible Description Language for Video Games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 4, pp. 325–331, 2014.
32. T. Schaul, J. Togelius, and J. Schmidhuber, "Measuring Intelligence through Games," *CoRR*, vol. abs/1109.1314, pp. 1–19, 2011.
33. T. Schuster, "MCTS Based Agent for General Video Games," Master's thesis, Maastricht University, 2015.
34. N. Shaker, J. Togelius, and M. J. Nelson, *Procedural content generation in games*. Springer, 2016.

35. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
36. D. Soemers, "Enhancements for Real-Time Monte-Carlo Tree Search in General Video Game Playing," Master's thesis, Maastricht Univ., 2016.
37. J. Togelius, N. Shaker, S. Karakovskiy, and G. N. Yannakakis, "The Mario AI Championship 2009-2012," *AI Magazine*, vol. 34, no. 3, pp. 89–92, 2013.
38. A. M. Turing, "Chess," in *Faster than thought*, B. V. Bowden, Ed. Pitman, 1953, pp. 286–295.
39. J. van Eeden, "Analysing and Improving the Knowledge-based Fast Evolutionary MCTS Algorithm," Master's thesis, 2015.
40. O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser *et al.*, "Starcraft II: A New Challenge for Reinforcement Learning," *arXiv preprint arXiv:1708.04782*, 2017.
41. M. d. Waard, D. M. Roijers, and S. C. Bakkes, "Monte Carlo Tree Search with Options for General Video Game Playing," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 47–54.
42. K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016.