What can we learn about AGI from game AI?

Julian Togelius, Sam Earle, Graham Todd, Georgios N. Yannakakis[*]

Abstract

It is sometimes thought that we are done with game-playing as an AI benchmark, as AI methods have surpassed humans in many hard games. But these winning methods tend to be tailor-made for individual games. When it comes to *general* game playing we are far from done. In fact, modern foundation models are surprisingly bad at playing unseen games, especially compared to their performance on other tasks. We look back at the long history of research in games to see what we can learn about this phenomenon. We then look forward to what would be a useful game-based benchmark of general machine intelligence.

1 Introduction

What are the core learnings from 70 years of research on game-playing AI, and what can they tell us about the prospect for more general AI? And can it help us contextualize the recent success of LLMs? Here are some key lessons:

There are many different ways of playing games, and which approach plays a particular game best depends on the game, how the game is represented to the algorithm, and what kind of resources you have, such as training time, player data, and domain knowledge [53].

The field of games is vast and getting vaster. You could see game design as an ongoing exploration of the shape of human cognition, where new game designs are proposed that challenge human minds in novel ways, and/or cater to novel tastes. Humans like games because they like having their cognitive, social and affective abilities challenged, and because they learn as they play. Good games are activities you gradually learn to do as you play them; approachable yet deep. Games have been designed for thousands of years; so it is no wonder that the myriad of board, card, and video games in existence challenge our brains in many different ways. Long-term planning, logical reasoning, visual processing, quick reactions, collaborative dynamics, emotional narrative parsing, and many others [49].

But games' appeal lies not only in the extent to which they tickle our cognitive abilities—by and large, they are much more than glorified IQ tests. Games also reflect and remix aspects of the real world, presenting

[*]JT, SE, and GT are with New York University, while GNY is with the University of Malta. Contact: julian@togelius.com

microcosmic toy versions of complex real world systems, or inventing hypothetical ones by recombining known parts. Good games leverage their (often cognitively challenging) systems to comment on or reinterpret the world [5], and in this way resonate with us semantically; approachable yet novel. There are games about framing photos, dating pigeons, and micromanaging the German army. (Yes, these are different games [17, 33, 45].) If we want AI agents that can adapt to challenging unseen situations in the real world the way a human would, then training and evaluating these agents on the vast array of more-or-less alien spin-offs of this world as embodied in the vast library of human games is a good place to start.

In sum, all well-designed games are expertly tailored to human capabilities, intuition, and common sense. By virtue of this, they make for great tests of intelligence. And by virtue of their storied relationship with the real world, they also happen to be tests which we care deeply about, and in which we have a thorough understanding of the breadth of human abilities and behavior. (Unlike some contemporary benchmarks that seek to reveal the blind spots in the abilities of frontier AI methods, we won't need to coerce researchers to reinvent the wheel of good game design from the ground up, nor pay human subjects to bang their heads against the un-fun results.) We are not just interested in one game, or one type of intelligence, but in the best that the whole human catalog has to offer, both in terms of games, and the diverse intelligences that are brought to bear on them. In the current landscape, there is no cure-all solution; different types of algorithms are appropriate for different types of games.

2 AI Playing Games

Concretely, then, how can you use AI (for some value of the overloaded term AI) to play a game? You can use planning, reinforcement learning, imitation learning, or program the behavior explicitly [53].

Planning is the driving engine behind the great triumphs in AI playing board games, from Othello [41] to Chess [8] to Go [43]. Here, moves are made in simulation, and the consequences of sequences of moves are evaluated. Various algorithms are used, with Monte Carlo Tree Search (MCTS) powering the leading approaches for many games [7]. In a sense, planning approaches really do generalize—or at least, they are general; the same method that is used for playing one game can, with minimal modifications, play other games without any data or training time. The cost of this generality, however, is tractability. Planning methods require fast and reliable simulators in order to practically tackle even moderately complex games—ideally these simulators can process games orders of magnitudes faster than real-time. It turns out that board games like those used in the success stories described above are an outlier in the sense that this rapid simulation is possible at all. Most video games cannot easily be simulated faster than real-time, partly because of computational complexity and partly because of how game engines are built (with rendering logic often intertwined with the underlying simulation, and with the speed of this simulation often bound to the system clock). The real world, of course, cannot currently be simulated well at all, never mind fast.

Reinforcement learning and game-playing are almost synonymous in some people's minds. The first reinforcement learning algorithm was invented in the 1950s in order to play checkers [40], and for a long time successful applications of RL could almost exclusively be found in game-playing. A decade ago, DeepMind's results on playing Atari games from visual inputs ushered in the current paradigm of modern deep reinforcement learning [32], and Atari games are probably the most common benchmark for RL methods [13]. It is easy to see why: many games provide limited episodes, clearly delineated action and state spaces, fast simulation, and clear rewards. Exactly the things that RL needs.

Yet, RL for game-playing has serious shortcomings. The most obvious one is the long training time. RL also has issues with games with unclear rewards or episode lengths. The less obvious drawback is overfitting. If you train a neural network to play a single game, it will learn only to play that game. In fact, it will typically learn to only play the particular level(s) it was trained on from the particular starting position(s) that were used in training [22]. The overfitting is often so bad that doing tiny cosmetic changes to the game, such as perturbing the color space or translating the pixels a little bit, renders the learned policy useless [60, 54]. While there are networks that have been trained to play, for example, multiple Atari games, they are generally not capable of playing unseen Atari games well [26]. In other words, transfer is very limited.

It seems that the kind of policy that results from reinforcement learning on individual games neither creates nor relies on the kind of representations that would be needed for more general game-playing capacity. When it can, RL will learn a brittle strategy rather than a robust one. For example, it may learn to jump only when it sees a particular object at a particular place, rather than the general concept of an obstacle that needs to be jumped over. This is true both for RL via gradient descent and for evolutionary RL, such as neuroevolution and genetic programming.[1] There are also methods which attempt to learn to play games in a more human-like manner and with a more human-like amount of data [51, 1]. While impressive, these approaches remain somewhat constrained in the kinds of domains and games they are able to solve.

You can also learn a game-playing policy from imitation, for example via behavior cloning. All you need is lots of traces of the game being played (preferably played well) by humans. Unfortunately, such data is scarce. Game developers rarely record their players' playthroughs at the scale and resolution that is needed to train competent imitation models. To get around this, a recent line of work trains on publicly available "Let's Play"-style YouTube videos in which a visual representation of the player's controller inputs is overlaid with in-game video [30, 57, 38, 39].[2] But such

[1]To be fair, why shouldn't these methods lazily learn the brittlest strategies? If this simplistic approach works on the task at hand, then it is as good as any other, as much as it may be misaligned with our own intuitions. The tendency of policies generated by single-objective optimization to overfit to the training distribution is perhaps more feature than bug; but that doesn't make it any less unintelligent.

[2]This approach can be extended to allow behavior cloning on video lacking the controller overlay as well: first, an inverse dynamics model is trained on videos with the controller, then this model is used to label raw videos.

systems remain limited to predicting the next frame given only the current one. With no effective memory, long-term planning remains well out of reach. More generally, even where sufficient training data exists—and supposing models architected to most effectively capture their distribution—imitation-learned policies tend to overfit just like reinforcement-learned ones [50, 20]. So in either case, irrespective of the spoils of data or compute in our possession, as long as some designer comes along with a truly ingenious and novel game—the next *Baba is You* [47], *Fez* [18], or *Braid* [4], say—it seems unlikely that any model begot by modern deep learning would be able to cope.

In video games themselves, incidentally, almost all the behavior of non-player characters is programmed explicitly. Typically, the policy relies on human-defined states and state transitions, with movement guided by pathfinding algorithms. Game developers prefer these kinds of handcrafted policies partly because it gives them more fine-grained control over the NPCs' behavior. But it is also testament to the general inadequacy of machine learning to the task of learning good NPC behavior.

3 How about Large Language Models?

A reader who is familiar with recent progress in AI might at this point ask if large language models or vision-language models can play games at the level we are aiming for [19]. LLMs seem to be able to do just about anything; surely they can play *Super Mario Bros* [42] and *Slay the Spire* [12]? The answer is that it depends, but mostly no. There have been success stories where LLMs have been able to play and even win nontrivial games. The *Voyager* agent which learned to play *Minecraft* [31] is one example [52], and the *Claude plays Pokemon* project, where the leading LLM plays the classic *Pokemon Red* Game Boy game [46], is another [36]. However, these examples feature elaborate game-specific integration. For Voyager, the LLM has access to a *Minecraft*-specific API which allows it to query the game world and execute relatively complex actions. In the case of Pokemon, various tools are used to read the screen and find paths. Both games also require complex information management architecture—often referred to as retrieval-augmented generation (RAG) [27]—to summarize, store, and retrieve information, as the context of the LLM rapidly fills up. The extensive harnesses built for each game would need to be redesigned for new games.

One should also not forget that *Minecraft* and *Pokemon* are two of the world's most famous game franchises, for which enormous amounts of commentary has been posted online in the form of strategy guides, forum discussions, and walkthroughs. Even obscure details about these games are already known to LLMs from pretraining.

If you pit an LLM against a game it has not seen before, the result is almost certain failure [28]. Without an agentic harness to handle context and game-specific integration, the result is typically not only failure to win the game, but failure to do anything useful at all. If a vision-language model gets a screenshot of an unknown game and is asked what to do next, and the game has a first-person view, the model will typically respond with a reasonable answer such as "go left". But from there to actually being able to play the game is a very long way. For games that do not have a first-person perspective, all bets are off. It is common to see the LLM make reasonable guesses about the rules of the game, but completely fail at reasoning about physical space. Translating the observation to text may lead to better performance, but this requires substantial game-specific integration and essentially moves much of the policy burden onto whatever system interprets the game state observation. (On top of this, LLMs that are large enough to know anything about the game are generally much too slow for playing real-time games, as taking an action takes at least several seconds.)

The obvious question at this point is: why are LLMs so bad at playing games if they are so good at so many other things? Or, how are LLMs able to write complex code and translate poems from Icelandic if they cannot even play Space Invaders?

The simplest answer is that LLMs are not trained to play games. Sequences of $< game\_state, action, reward >$ are not in their pre-training data. Or at least very rare. They have also not been post-trained on game-playing. But perhaps, if they had been trained on games, LLMs (or rather VLMs) would be able to play video games, including unseen video games, competently. Answering this question properly would require a large amount of training data and compute. It would probably take the form of finding a very large number of playtraces from thousands of games to pretrain on, and then posttraining using reinforcement learning on hundreds of games. It is likely that this would succeed to at least some extent. However, the failures of standard LLMs to reason about space leave the possibility open that a different architecture would be needed.

The recent SIMA agent from DeepMind would seem to fit the bill [37]. Building on the frontier vision-language model Gemini [11], the base model is fine-tuned via supervised learning on (an undisclosed amount of) variously-annotated and synthetically-augmented human playtraces. It is then fine-tuned again, this time via reinforcement learning, on some number of in-game tasks with verifiable rewards. The end result is a kind of text-controllable player assistant which can commandeer the player avatar in order to do follow explicit instructions like "go to the house colored like a ripe tomato", "check out the egg-shaped objects", or "cook the burger patty".

SIMA's focus seems to be on embodied navigation and exploration within photorealistic environments, with the suite of training and evaluation games including *MineCraft*, *Space Engineers* and *No Man's Sky*. Long term planning, self-directed behavior, and disembodied or puzzle-forward games seem out of scope for the time being. A self-improvement loop is also introduced, in which Gemini defines new tasks, and rates newly-generated trajectories for their success on these tasks. The resulting dataset is used to train a reward model, which is then used to further RL-finetune SIMA. While promising, this kind of continual learning would need to be massively scaled to close the gap between the simple, low-level navigation/exploration tasks currently at the frontier of the agent's capabilities, and the holistic challenge of winning the game(s). And the overall loop seems vastly more compute- and data-intensive than what humans do when adapting to a novel game.

4 General Game Playing

So what would success mean? In other words, what would it mean to be able to train a model or system that would be able to play video games in general? We could define this as an agent that could play and win the top 100 games on Steam or the iOS App Store that it had not been trained on (or even seen before). For simplicity, we could restrict ourselves to only those games that have a reasonably well-defined concept of "winning" or "completing". We could also limit ourselves to single-player games that only rely on screen and sound outputs, and use standard inputs such as touchscreen, mouse, and/or keyboard. For completeness, let us stipulate that we have access to neither a fast simulation of the game nor a way of quickly restoring arbitrary game states, because otherwise a parallel tree search on a large cluster could win lots of games.[3]

With all that covered, what would a general game playing agent look like?

We cannot demand that it would be able to win those games on the first playthrough, because that would likely be impossible. That's not how video games are designed. It's not a skill issue. The information you need to play a game well, let alone complete it, must typically be learned by trial and error. This involves things such as combat mechanics, how to allocate resources, which areas to explore, movement patterns of non-player characters, and so on. To make this more vivid: it does not matter how intelligent you are, you cannot win every fight in *Street Fighter II* [2] or reach the end of *Elden Ring* [56] on your first attempt if you have never seen the game before. You do not have the information. The reason video games are designed this way is that they are meant to be learned from, and failure is a key part of learning and having fun while playing [23].

A reasonable criterion for a general game playing agent would instead be that it would be able to learn to play the game to completion within approximately the same time as a skilled human game player. This would probably be somewhere around ten to a hundred hours for a typical video game.

Testing an agent on its ability to learn to play unseen video games without game-specific integration is a different, much harder, and much more interesting challenge than testing an agent's ability to play a game which it has spent tens of thousands of simulated hours playing, or for which it has read immense amounts of descriptions and strategy discussion. It is not at all clear that any currently existing methods are up to the task. At the time of writing, no existing "agentic" tasks require independent action over tens of hours. It is also not clear how the learning would take place. Tens of hours, and maybe a hundred or so episodes, is too little data for standard reinforcement learning algorithms. But it is also orders of magnitude too much to fit into the context window of

[3]Perhaps we could solve any number of combinatorially explosive games if the world was our hypothetical quantum computer, but our instinct is that brute force tree search over uncompressed states is obviously not the kind of general intelligence we are after.

any existing LLM. And the type of learning needed does not seem easily amenable to the kind of text summarization used in RAG systems. In other words, this is a very hard challenge that would probably require us to invent new methods.

If we managed to create a system that could play unseen video games according to the criterion described above, would that constitute Artificial General Intelligence? It would arguably have a stronger claim to this title than, for example, a system that could pass a Turing test, simply because the task is harder. But more importantly, it would test different things, because the task is different. For example, a general game-playing system would need to be able to read plenty of text and reason about complex storylines because of the role-playing games that are common on the leaderboards of Steam and the App Store, but it would not need to be good at writing convincing text. Is being able to navigate virtual fortresses more important to AGI than being able to write poems? This depends on how you interpret "general intelligence," and it is not likely that we will reach a consensus on that anytime soon.

The outstanding question is how LLMs can do all the marvelous things they can do when they cannot even play unseen games. In particular, to take one of the most prominent and economically important use cases, how can they be so good at writing code? You would imagine that writing code is harder than playing new video games, given that you can get paid very well for the former activity but not so much for the latter. Perhaps not. A solution to this conundrum is to consider programming as a game.

Programming is actually a very well-behaved game. Given a task expressed in natural language (e.g., write a function that sorts this array) you have a clear problem statement and a clear success criterion. Unit tests give you very clear feedback and the observation space is well-behaved (i.e. it is all strings and stack traces that reveal exactly where an error occurred). When you fail (the program crashed or did the wrong thing), you can easily backtrack by erasing the changes you made and rewriting that part. Most video games have less clear feedback, more confusing action spaces, and are more punishing when you fail. This might help explain why programming is such an oddly satisfying activity.

Now consider that modern LLMs have been pretrained on obscene amounts of program code, and then further trained via reinforcement learning to solve coding problems. You can think of programming in one particular language as one game and the different coding tasks as different levels of that game. From this perspective, it becomes easier to understand how LLMs can be so good at playing this particular game. This intuition is supported by the fact that, anecdotally, LLMs are less proficient in lesser-known languages; take for example more specialized domains like the JAX library for GPU-accelerated computation [6], or more niche game description languages like Ludii [44] or PuzzleScript [25].

So while LLMs may be experts—by virtue of extensive pretraining—on a handful of (admittedly vast and open-ended) coding games, they cannot necessarily adapt as readily as human players to other entries in the genre at large. Indeed, it would be interesting to see how these models would fare when faced with *bona fide* coding games, like *Opus Magnum* [58], *Last Call BBS' 20th Century Food Court* [59], or *Factorio* [24]. We'd expect them to struggle quite a bit (and recent work treating the latter domain as a benchmark would seem to support this hypothesis [21]) despite the fact that these games each transmute the shared essence of programming in different ways.

5 What Can we Learn about AGI from Game-playing AI?

Let us finally return to the original question: what can we learn from 70 years of game-playing research and what can it tell us about AGI? For one thing, different games are very different and pose very different types of challenges. The type of challenge depends not only on the game, but also on what kind of information you have available, whether there is a fast forward model, if you have time to train, data to learn from, and other factors. As a result, the landmark successes of AI systems becoming strong players of various games, from Chess and Go to *StarCraft* [9] and *DOTA* [16] via *Super Mario Bros* and *Montezuma's Revenge*, rely on quite different AI methods. What they have in common is that plenty of game-specific engineering and/or training went into each of these successes.

Which brings us to the other big takeaway. General video game playing, in the sense of being able to play any game of the top 100 on Steam or iOS App Store after only the same amount of playing time that a human would need, is a very hard challenge that we are nowhere near solving and not even seriously attempting. It is not at all clear that current methods and models are suited to this problem. General video game playing is as good an AGI test as any. More generally, if we take game-playing seriously by looking beyond overfitting on specific games, there is a wealth of problems to solve and progress to be made in this field.

Earlier attempts at providing some kind of general video game playing benchmark include ubiquitous Arcade Learning Environment [3], built on Atari Games, which unfortunately allowed for and therefore resulted in blatant overfitting. Another example is General Video Game AI, which, while the competition ran, evaluated submitted controllers on novel games [29]. A more recent and in some ways more ambitious attempt is the "AI GameStore" [55], which explicitly aims to replicate popular existing games to test the cognitive capacities of LLMs.

Let's be optimistic for a moment, and suppose that these problems all get solved, and we wind up with an artificial player that is able to master new games over dozens of hours at the level of an intelligent human. Would this alone be enough to convince us to call such an agent "generally intelligent"? Probably not. Intelligent humans do more than satisfy predefined win conditions, no matter how cleverly. In fact, humans spend inordinate amounts of time doing things that would seem completely superfluous to an optimal trajectory through the game of life, insofar as it can be defined. We play, we invent, and we invent new ways of playing. We sink hours upon hours into playing and designing video games, because it's fun and we want to (though even seemingly silly games may also hold adaptive value [10]).

For an artificial general intelligence to be considered genuinely human-level, then, it too would need to be autotelic, open-ended and creative. And what better way to gauge this creative life force than in the ability to create novel, challenging, and interesting video games in a manner on par with humans?

While it is true that you can ask a current coding agent to create you a new video game and (often) receive a playable game, the output is generally neither novel or good. And this is not for lack of trying. We've taken several stabs at this ourselves. In some instances, we have confined ourselves to domain specific languages, in effect carving out a space of relevant possible games from that of all possible programs [48, 14, 34]. This comes at the cost of having to few-shot-prompt or fine-tune models, but increases the likelihood that changes to code will result in meaningful changes in the game it represents. In others, we have let models implement games or simulations in more general programming languages [35, 15], allowing models the flexibility to more easily implement whatever they like, at the risk of having to spend more time sifting through minor variations of the same underlying idea. But LLMs consistently come up short, no matter whether we ask them to bring non-trivial ideas to life under practical constraints, or to come up with anything interesting to do in the first place.

And so we have come full circle, in a way, because what is missing from these systems is precisely a general player model. No matter how much we may find partial success in steering these generative systems toward reasonable game-like outputs by shaping the search space and fitness landscape in advance, a truly general model of automatic game creation involves an open-ended feedback loop between the designer and some model(s) of the player, with likely very little other assumptions built in. Or, as is common wisdom among human designers: creating a good game requires frequent playtests from a diverse range of players. Creating AI systems capable of general human game playing thus serves not only as a grand challenge itself, but also as a crucial stepping stone to the *next* grand challenge: coming up with the kinds of games that are worth playing in the first place.

References

[1] Zergham Ahmed, Joshua B Tenenbaum, Chris Bates, and Samuel J Gershman. Synthesizing world models for bilevel planning. *Transactions on Machine Learning Research*, 2025.

[2] Akira Yasuda Akira Nishitani. Street fighter ii, 1991. Video game. Arcade.

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47:253–279, 2013.

[4] Jonathan Blow. Braid, 2008. Video game. Xbox 360, PC, PlayStation3.

[5] Ian Bogost. *The rhetoric of video games.* MacArthur Foundation Digital Media and Learning Initiative, 2008.

[6] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

[7] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[8] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[9] James Phinney Chris Metzen. Starcraft, 1998. Video game. PC, Nintendo 64.

[10] Junyi Chu, Joshua B Tenenbaum, and Laura E Schulz. In praise of folly: flexible goals and human cognition. *Trends in Cognitive Sciences*, 28(7):628–642, 2024.

[11] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[12] Mega Crit. Slay the spire, 2019. Video game. PC, Nintendo Switch, PlayStation 4, Xbox One, iOS, Android.

[13] Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. OCAtari: Object-centric Atari 2600 reinforcement learning environments. *Reinforcement Learning Journal*, 1:400–449, 2024.

[14] Sam Earle, Ahmed Khalifa, Muhammad Umair Nasir, Zehua Jiang, Graham Todd, Andrzej Banburski-Fahey, and Julian Togelius. Scriptdoctor: Automatic generation of puzzlescript games via large language models and tree search. In *2025 IEEE Conference on Games (CoG)*, pages 1–5. IEEE, 2025.

[15] Sam Earle, Samyak Parajuli, and Andrzej Banburski-Fahey. Dreamgarden: A designer assistant for growing games from a single prompt. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2025.

[16] Eul. Dota, 2003. Video game. PC, Android, iOS.

[17] Naphtali Faulkner. Umurangi generation, 2020. URL `https://store.steampowered.com/app/1223500/Umurangi_Generation/`. Video game. Available on Steam and Nintendo Switch.

[18] Phil Fish. Fez, 2012. Video game. Xbox 360, PC, PlayStation 3, PlayStation 4, PlayStation Vita, iOS, Nintendo Switch.

[19] Roberto Gallotta, Graham Todd, Marvin Zammit, Sam Earle, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Large language models and games: A survey and roadmap. *IEEE Transactions on Games*, 2024.

[20] Nathan Gavenski and Odinaldo Rodrigues. Quantifying generalisation in imitation learning. *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[21] Jack Hopkins, Mart Bakler, and Akbir Khan. Factorio learning environment. *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[22] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. In *NeurIPS Workshop on Deep Reinforcement Learning Workshop*, 2018.

[23] Raph Koster. *Theory of fun for game design.* " O'Reilly Media, Inc.", 2013.

[24] Michal Kovařík. Factorio, 2020. Video game. PC, Nintendo Switch, Switch 2.

[25] Stephen Lavelle. Puzzlescript. `https://www.puzzlescript.net/`, 2013. Accessed on: February 19, 2026.

[26] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in neural information processing systems*, 35:27921–27936, 2022.

[27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[28] Yuchen Li, Cong Lin, Muhammad Umair Nasir, Philip Bontrager, Jialin Liu, and Julian Togelius. Gvgai-llm: Evaluating large language model agents with infinite games. *arXiv preprint arXiv:2508.08501*, 2025.

[29] Diego Pérez Liébana, Simon M Lucas, Raluca D Gaina, Julian Togelius, Ahmed Khalifa, and Jialin Liu. *General video game artificial intelligence*. Springer Nature, 2022.

[30] Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, et al. Nitrogen: An open foundation model for generalist gaming agents. *arXiv preprint arXiv:2601.02427*, 2026.

[31] Jens Bergensten Markus Persson. Minecraft, 2011. Video game. PC.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[33] Hato Moa. Hatoful boyfriend, 2014. URL `https://store.steampowered.com/app/310080/Hatoful_Boyfriend/`. Video game. PC, iOD, Android, PlayStation 4, PlayStation Vita.

[34] Muhammad U Nasir, Steven James, and Julian Togelius. Word2world: Generating stories and worlds through large language models. *arXiv preprint arXiv:2405.06686*, 2024.

[35] Muhammad U. Nasir, Yuchen Li, Steven James, and Julian Togelius. Mortar: Evolving mechanics for automatic game design, 2025. URL `https://arxiv.org/abs/2601.00105`.

[36] Tharin Pillay. Why the world's best ai systems are still so bad at pokémon. *TIME*, Jan 2026. URL `https://time.com/7345903/ai-chatgpt-claude-gemini-pokemon/`.

[37] Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, et al. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*, 2024.

[38] Nemanja Rašajski, Chintan Trivedi, Konstantinos Makantasis, Antonios Liapis, and Georgios N Yannakakis. Behave: Behaviour alignment of video game encodings. In *European Conference on Computer Vision*, pages 321–338. Springer, 2024.

[39] Nemanja Rašajski, Konstantinos Makantasis, Antonios Liapis, and Georgios N Yannakakis. InvPatch: Prefix-Based Conditional Generation for Inverse Dynamics. In *IEEE Conference on AI*. IEEE, 2026.

[40] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[41] Jonathan Schaeffer, Robert Lake, Paul Lu, and Martin Bryant. Chinook the world man-machine checkers champion. *AI magazine*, 17 (1):21–21, 1996.

[42] Takashi Tezuka Shigeru Miyamoto. Super mario bros., 1985. Video game. Nintendo Entertainment System.

[43] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[44] Matthew Stephenson, Eric Piette, Dennis JNJ Soemers, and Cameron Browne. An overview of the ludii general game system. In *2019 IEEE Conference on Games (CoG)*, pages 1–2. IEEE, 2019.

[45] Paradox Development Studio. Hearts of iron iv, 2016. URL `https://store.steampowered.com/app/394360/Hearts_of_Iron_IV/`. Video game. PC.

[46] Satoshi Tajiri. Pokémon red, 1996. Video game. Game Boy.

[47] Arvi Teikari. Baba is you, 2019. Video game. PC, Nintendo Switch, Android, iOS.

[48] Graham Todd, Alexander G Padula, Matthew Stephenson, Éric Piette, Dennis J Soemers, and Julian Togelius. Gavel: Generating games via evolution and language models. *Advances in Neural Information Processing Systems*, 37:110723–110745, 2024.

[49] Julian Togelius. *Playing smart: On games, intelligence, and artificial intelligence*. MIT Press, 2019.

[50] Julian Togelius, Renzo De Nardi, and Simon M Lucas. Towards automatic personalised content creation for racing games. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 252–259. IEEE, 2007.

[51] Pedro A Tsividis, Joao Loula, Jake Burga, Nathan Foss, Andres Campero, Thomas Pouncy, Samuel J Gershman, and Joshua B Tenenbaum. Human-level reinforcement learning through theory-based modeling, exploration, and planning. *arXiv preprint arXiv:2107.12544*, 2021.

[52] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

[53] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*, volume 2. Springer, 2025.

[54] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*, 2020.

[55] Lane Ying, Ryan Truong, Prafull Sharma, Kaiya Ivy Zhao, Nathan Cloos, Katherine M. Collins, Kelsey R. Allen, Thomas L. Griffiths, José Hernandez-Orallo, Phillip Isola, Samuel J. Gershman, and Joshua B. Tenenbaum. Ai gamestore: Scalable open-ended evaluation of machine general intelligence with human games. 2026.

[56] Ryu Matsumoto Yosuke Kayugawa. Elden ring, 2022. Video game. PlayStation 4, PlayStation 5, PC, Xbox One, Xbox Series X/S, Nintendo Switch 2.

[57] Yuguang Yue, Chris Green, Samuel Hunt, Irakli Salia, Wenzhe Shi, and Jonathan J Hunt. Pixels to play: A foundation model for 3d gameplay. In *2025 IEEE Conference on Games (CoG)*, pages 1–4. IEEE, 2025.

[58] Keith Holman Zach Barth. Opus magnum, 2017. Video game. PC.

[59] Zachtronics. Last call bbs, 2022. Video game. PC.

[60] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.