Essay

Agent Testing in Video Games

Author: Adam Melander

DD2482
Automated Software Testing and DevOps
2021-06-24

Contents

Abstract

As video games become larger and more complex, ensuring thorough testing can be challenging. Relying solely on human testers can slow down development and allow for human errors. To speed up the process, autonomous agents using reinforcement learning have shown promise in exploring and testing game scenarios. This goes for both single player environments and multiplayer environments, although the latter would require multiple agents as well.

As testing is a core component in devops, automated testing could be a massive improvement in terms of thoroughness and efficiency. Tools like Agent Testing could potentially save a lot of time in game development pipelines and decrease the risk of letting bugs out in the production environment.

In this essay I will look into the current state of Agent Testing technology and try to analyse it and its current usefulness.

1 Introduction

Today's AAA video games are a massive accomplishment of software engineering, creative teams and technology. They usually require hundreds, if not thousands, of developers, artists, testers and a lot of other functions working in an organized unit to produce these high end products [Gam20].

But regardless of that, whenever a new game is released or patched, it is not uncommon at all that plenty of bugs are found by the gamers that are notoriously quick of both finding them and abusing them.

This could have a variety of different consequences. Sometimes finding and abusing a path that should be unreachable but achievable through a lot of effort and difficult features leads to speed runners getting an increased replay value of the game [Spe21]. Which is not necessarily a problematic error, as its usually requires a lot of effort to both find and perform, i.e. it does not happen by accident, and only impact the single player.

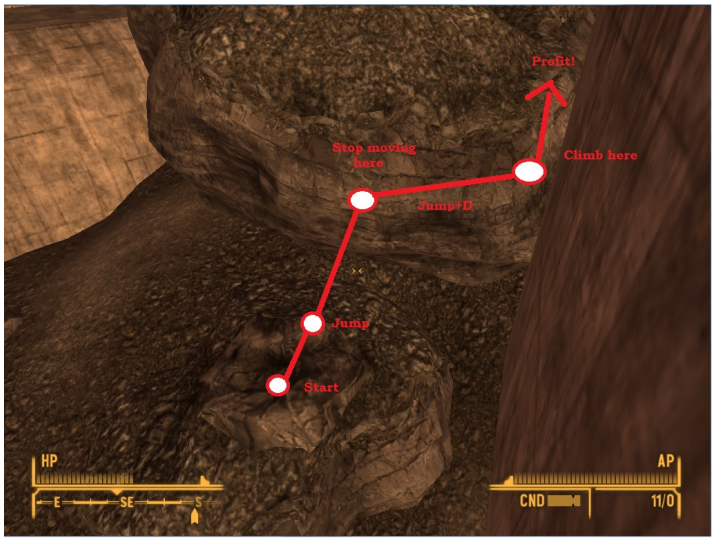

Figure 1: Red line showing a speed runners path to skip a game segment in "Fallout: New Vegas".

But sometimes it could lead to game breaking bugs occurring ruining the experience for regular players, everything from collision bugs locking the character in place to not well thought through design leading to the player being hard-locked in a certain state of the game.

In multiplayer games it could also lead to exploits being abused by players and as a result ruining the experience for other regular players resulting in a broken experience.

Figure 2: A Mario Kart 8 Deluxe player setting themself out of bounds to respawn right before the goal line, effectively getting another lap completed in mere seconds [Uni].

These bugs often occur due to the nature of human testing and due to the sheer scale of the projects. It is becoming unfeasible for a limited group of people, with a limited amount of time, to always find every single exploit in a game that later will be introduced to a significantly larger group with significantly more time. Especially since changes to small things in the game, such as character hitboxes or jump mechanics, would require a big amount of, if not all, parts of the games to be re-tested.

This is where Agent Testing comes in. With artificial intelligence (AI) advancements it has started to become possible to train AI:s to test the games and try to locate as many glitches and exploits as possible to avoid letting them out in production.

1.1 Generative AI Statement

I certify that generative AI, incl. ChatGPT, has not been used to write this essay. Using generative AI without permission is considered academic misconduct.

2 Background

2.1 What is Agent Testing

Agent Testing in video games is a technique in which AI agents are modelled to simulate human-like behavior and interact with a game world with the purpose of finding problems or exploits while trying to perform tasks, make decisions or respond to different scenarios in the game. This is to ensure that when human players get access to the game these problems have been solved. An advantage that Agent Testing has over the traditional human tester is that it is possible given enough computational resources to run an incredible high amount of agents in parallel, potentially making it a lot more efficient than human testers if the agents can perform well enough [ABS19].

2.1.1 Multiple Agent Testing

When testing games where multiple players would interact with each other it becomes a bit trickier as the most efficient strategy and the way the game is "supposed" to be played changes depending on the other persons actions. This is where Multiple Agent Testing comes in, which is a technique in which multiple AI agents are introduced to the game, each with their own objectives [BKM+19].

3 Technologies

3.1 EA Reinforcement Learning Agents

I will mostly base this technique and its results on a state of the art report paper published by Electronic Arts (EA) called "Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents" [GBTG21]

In EA's report they created a relatively video game level (500m x 500m x 50m) with some of the most classic video game mechanics: walking, jumping and climbing. Then they set different AI agents loose on the map to see how they performed.

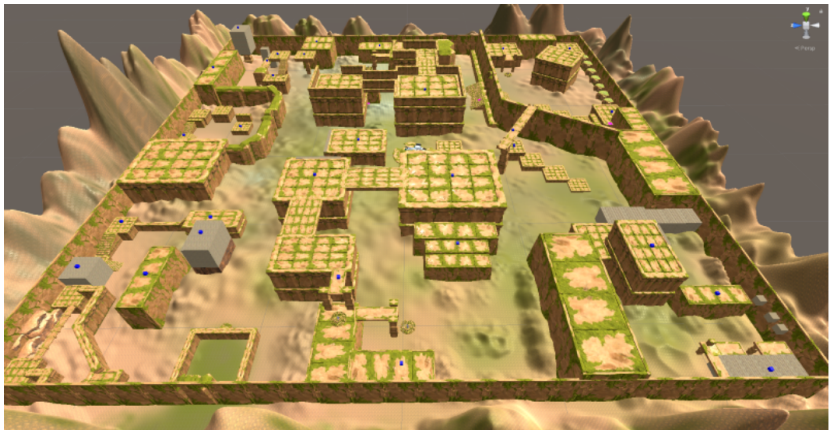

Figure 3: The video game level EA designed to test AI agents on.

They started out with a simple agent based on a random policy which managed to explore/test a measly 50% of map. The agent was not able to solve complex navigation tasks, even when simulated for longer periods of time, which locked it inside the more simply accessible starter areas of the map.

Later a agent with a reinforcement learning was introduced. This agent managed to solve more complex challenges and after being simulated for 60 hours it managed to explore approximately 95% of the map. It also showed some very fascinating results in terms of finding alternative (non-intentional) paths, which gave the developers very useful information on how to adjust the level to make the game better.

An example of a unintentional path was in the center of the map, where the designers had intended for the player to unlock and use an elevator to reach a higher elevated area of the game. How ever the reinforcement learning agent managed to find with an elaborate combination of jumps that a ledge nearby was not steep enough to keep the player out (figure 4). The developers could then adjust the steepness of the ledge and ensure that the path were used correctly (figure 5).

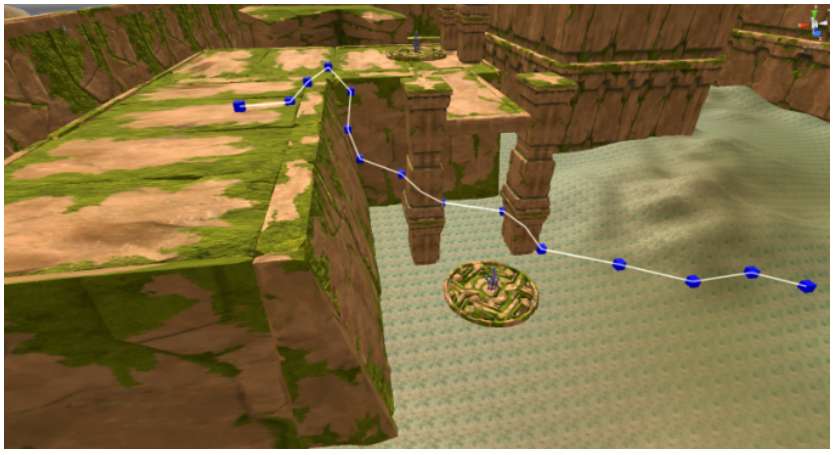

Figure 4: The elaborate combination of jumps the reinforcement learning agent managed to perform to avoid unlocking the elevator.

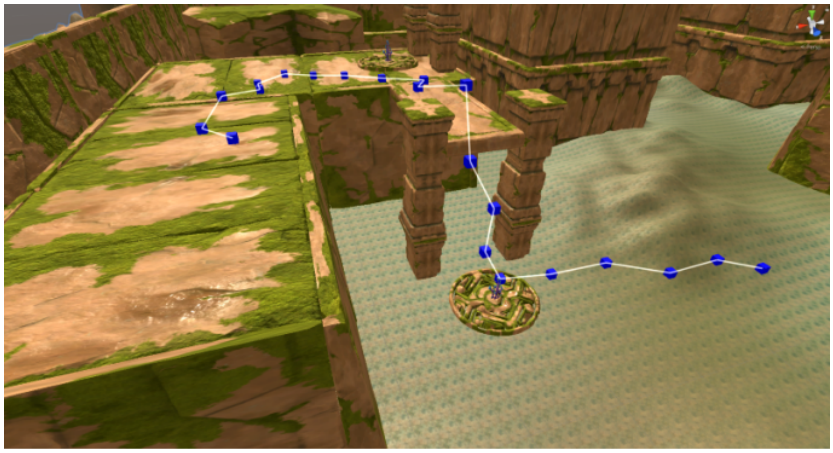

Figure 5: The correct path and the only option for the agent to perform after developers adjusted the map.

The reinforcement learning agent was also very successful in finding different paths to set the player out of bounds of the given map. This by performing elaborate jump/climb combinations at the objects near the map edges.

3.2 OpenAI Multiple Agent Hide and Seek

I will mostly base this technique and its results on a state of the art report paper published by OpenAI called "Emergent Tool Use From Multi-Agent Autocurricula" [BKM+19]

In OpenAI's report they created a simple hide-and-seek game in which different amount of players were put in a small confided map, of which some were "hiders" (i.e. trying to avoid getting detected) and some were "seekers" (i.e. trying to detect the hiders). On the map there were small structures, some immovable and some that the player could move by dragging/pushing them. The hiders were also able to "lock down" these movable objects, making them impossible for the seekers to use. They then let multiple reinforcement learning agents take the places of the hiders and the seekers to see how the playstyle developed. It should also be noted that in this game of hide-and-seek the hiders are not in any way punished for working together, i.e. there is no reward for being the last person hidden like in many real world examples of the game. Instead, collaboration is encourage by having the shared goal of avoiding the seekers.

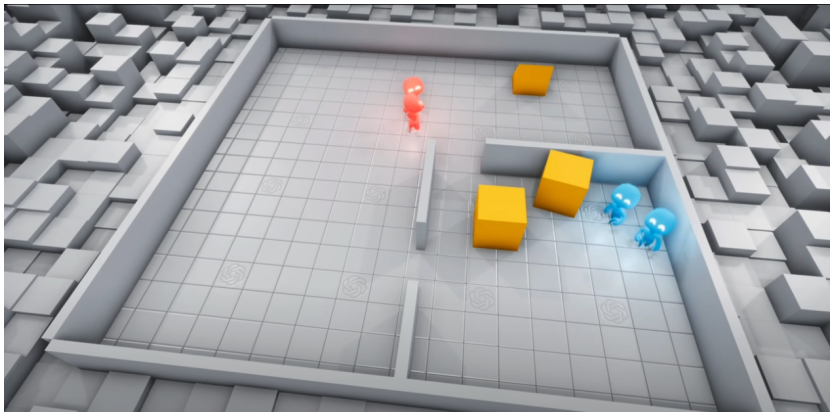

Figure 6: A map with some walls, two hiders (blue), two seekers (red) and three movable object (yellow)

The original purpose of the paper was to research how the agents would learn and develop when facing another agent. Which strategies would emerge and what the learning process would look like.

These are all interesting questions but outside of the scope of this essay. However what was interesting in this experience was that the AI agents would start to break the game in different unintentional ways when developing their strategies.

The first strategies that the hiding agents developed were to use the movable objects to block out the seekers outside of their space (figure 7). This strategy was very effective until the seekers learned to adapt and overcome by using the movable ramp object to climb above the walls of the hiders hideout (figure 8).

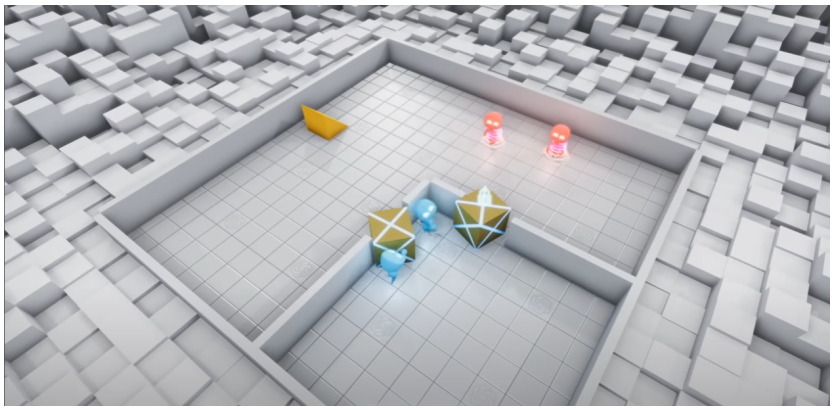

Figure 7: Hiders blocking off their hideout with the movable blocks.

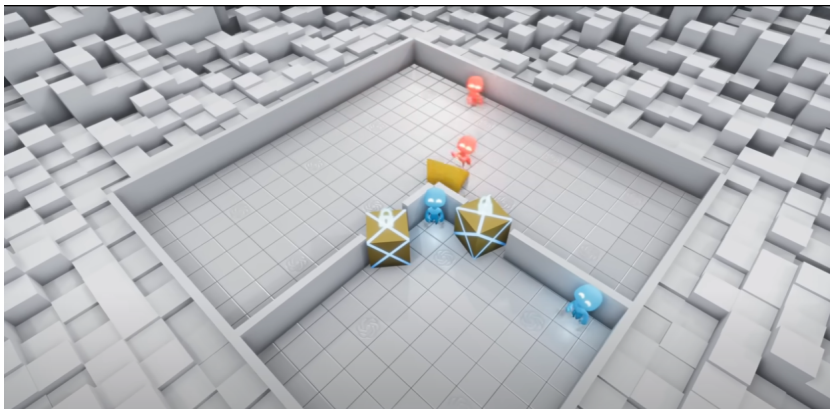

Figure 8: Seekers breaching the hideout by using a ramp.

This type of adapting and overcoming behavior proceeded and as more complicated maps were introduced more advanced strategies emerged. The hiders started to use the movable objects found on the map to build shelters and trying to lock down as many objects outside of their shelter that they had time for before the seekers started searching. In this type of lockdown the hiders prioritized locking down ramps as they had learn that they could easily be used to breach the shelter if left unlocked (figure 9).

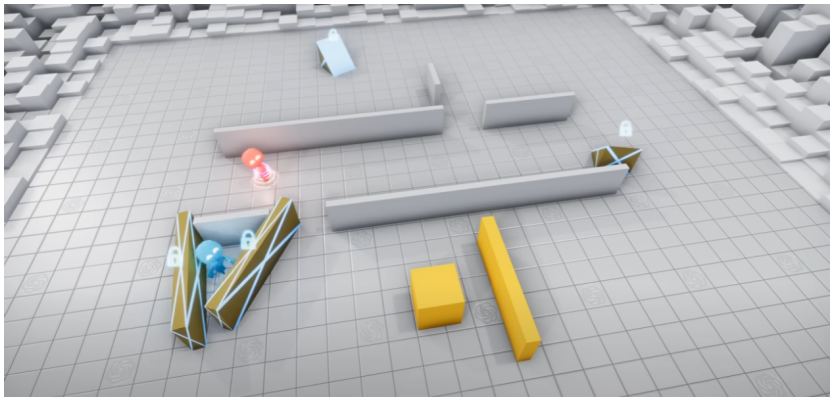

Figure 9: Hider has built a shelter and locked down both ramps on the map.

The seeker was stuck in this state of the game for a long period of time and this was the predicted end-game strategy from the OpenAI researchers, however the seeker agent managed to prevail in a unorthodox way. The seeker managed to move one of the movable boxes towards one of the locked down ramps and climbed up on the movable box, and then triggered the push-mechanic while standing on top of the box to perform a bug that the paper refers to as "box-surfing". With this new technique the seeker now manage to win against the hider and also reveal an interesting exploit.

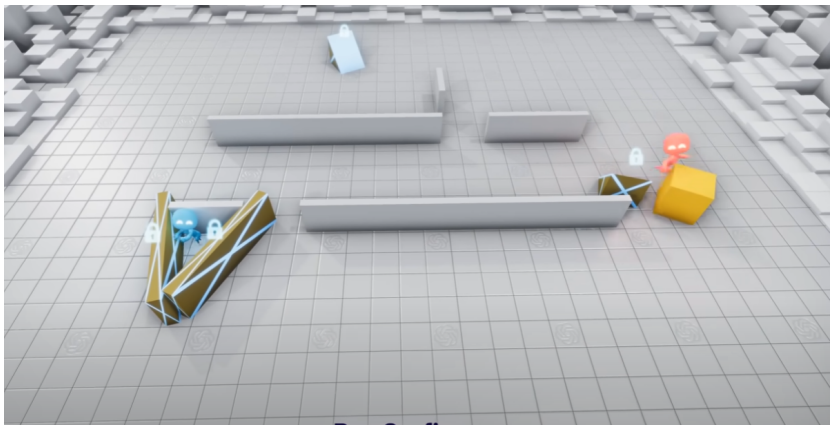

Figure 10: Seeker drags movable box to ramp and climbs on top of it.

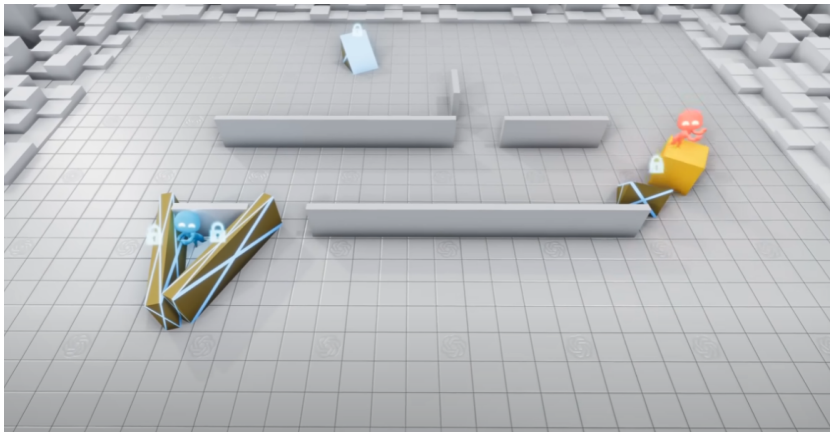

Figure 11: Seeker starts to perform "box-surfing" by using an exploit of the push mechanic.

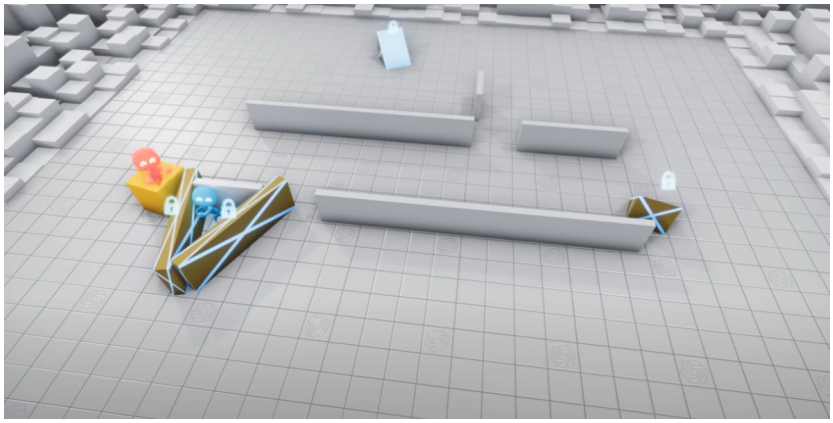

Figure 12: Seeker discovers the hider after successfully "box-surf" across the map.

There was also some other interesting exploits of the physics engine that emerged that the developers did not see coming. In one of them the hiders were able to abuse a wall corner to send a movable ramp object out of bounds, effectively making it impossible for the seekers to reveal them (figure 13). Another interesting physics exploit which the seekers found was that through some strange movements with the ramps against a wall they manage to send themself flying high and with some precision work they could breach the hiders shelters (figure 14).

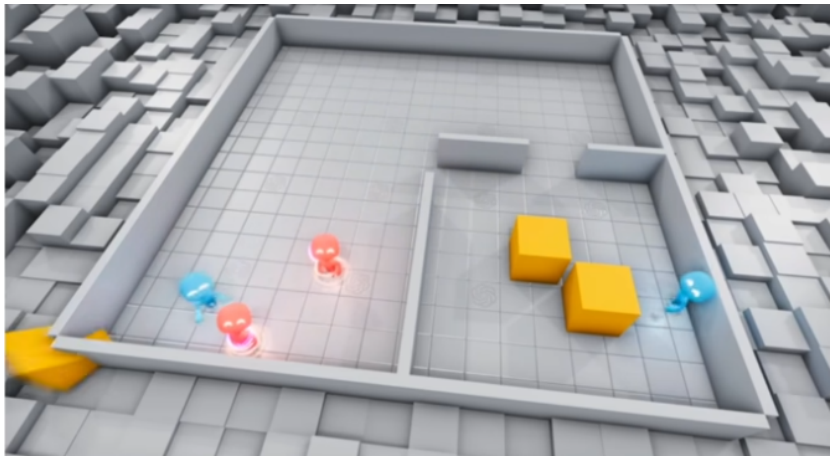

Figure 13: Hider sending ramp out of bounds by abusing wall corner.

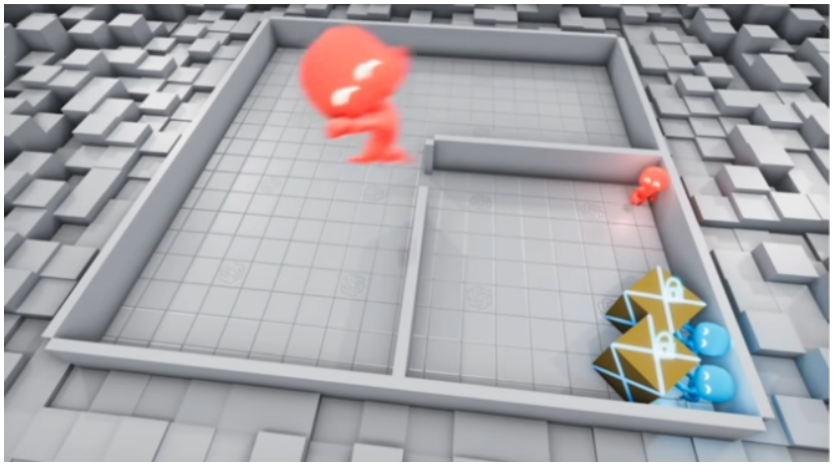

Figure 14: Seeker launching themself at the hiders shelter with a ramp exploit (ramp flew outside of map in the image).

4 Reflection

The results from both the EA study and the OpenAI experiment was quite interesting and showed a lot of potential from AI agents for testing purposes.

4.1 EA Study

To start out with the EA study managed to create a curious AI agent that effectively could playtest 95% of their game level in 60 hours and find unique and interesting bugs that would not too unlikely be unnoticed by human testers. They showed in their study that automating the playtesting and finding bugs through AI agents was not only possible, but could be very efficient. It should be said however that I am not sure how good 95% coverage in the context of video game testing is and it is hard to find sources discussing this (feels like internal policies amongst game developing companies), but it should definitly be taken into consideration and perhaps in combination with human testing for the remaining parts.

4.2 OpenAI Experiment

The OpenAI experiment was very fascinating to look into, especially since it really emphasize that the nature of AI agents is to not yield to conventional rules when exploring a game, rather it has quite the free mind. This could be the result of developers being human when developing games and having a very human mindset when planning for bugs and exploits. It is in our nature to consider things that we would think of exploiting and can be hard to consider the things that are outside of our regular behavior, but for an AI agent exploring the world it is just another path like any other. If it is there it will consider it as much as any other available path leading to a potentially exceptional play tester.

4.3 Limitations

As the AI agents only are able to do certain actions based on what input is given to them, rather than fully understand the ongoing experience, it is limited to only being able to understand and find certain bugs. For example the AI agents looked at performed really well in finding physics and logic bugs/exploits and managed to find both collision bugs and moving themselves and objects out of bounds, but these are not all the things that video game testers need to look out for. There are also for example graphical bugs, interface bugs, language bugs, etc. [hea] that the AI agents would not be able to find in their current state. There are also some scenarios where bugs/problems would not be game breaking or having a major impact but still would create a slight annoyance, this types of bugs are human tester better at noticing and recording as they have a more nuanced way of classifying problems[end].

4.4 Bug or Feature?

One thing that I think should be taken in to consideration as a end user of the video game products, and as a speed run enthusiast, is that not necessarily are all bugs and exploits are "bad". There is a certain beauty and charm in finding some of the more niche exploits which not necessarily are game breaking but can both add a complexity and create a massive replay value if "implemented" correctly. For example the wall climbing exploit found in the EA study would not impact regular gameplay as it is a complex bug to perform and would not have been discovered by casual players enjoying their playthrough. However the speed running community would both discover it and enjoy exploiting it internally to create the fastest possible completion of the game.

There are also many games where different exploits in casual games later have developed to become features in the more competitive communities, an example of this can be seen in many Nintendo games such as Super Mario Bros[ars] and Super Smash Bros [Des].

I am neither saying that all bugs and exploits should be removed from the game, nor am I saying that game developers should intentionally leave some bugs behind, but what I am saying is that I do not think the answer is black and white and hopefully it will be reflected upon by the developers.

5 Conclusion

Using AI agents for video game testing is a very promising technique that show great potential. It has the potential to be a very effective way to automatize video game testing. However, the technique and the usage is still relatively young and it will be interesting to see how it develops further moving forward. It can, in combination with human testing, be a good option as is and hopefully it will develop further to not need the human complement in the future.

5.1 Key Take-Away

Keep an eye on the development of AI agents, they could change the video game testing industry!

References

[ABS19] Sinan Ariyurek, Aysu Betin-Can, and Elif Sürer. Automated video game testing using synthetic and human-like agents. *CoRR*, abs/1906.00317, 2019.

[ars] arstechnica. How a speedrunner broke Super Mario Bros biggest barrier. url: https://arstechnica.com/gaming/2021/04/new-super-mario-bros-record-breaks-speedrunnings-four-minut

[BKM+19] Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *CoRR*, abs/1909.07528, 2019.

[Des] Destructoid. Super Smash Bros Melee Wavedash Physics Exploit That Became A Key Tactic. url: https://www.nintendo.destructoid.com/super-smash-bros-melee-wavedash-physics-exploit-th

[end] endava. THE JOY AND CHALLENGE OF BEING A VIDEO GAME TESTER. url: https://www.endava.com/en/blog/Engineering/2023/The-Joy-and-Challenge-of-Being-a-Video-Game-Tester.

[Gam20] Gamespot. AAA Devs Say 2000 People Working On One Game Isn't At All Unusual. url: https://www.gamespot.com/articles/aaa-devs-say-2000-people-working-on-one-game-isnt-/1100-6480243/, 2020.

[GBTG21] Camilo Gordillo, Joakim Bergdahl, Konrad Tollmar, and Linus Gisslén. Improving playtesting coverage via curiosity driven reinforcement learning agents. *CoRR*, abs/2103.13798, 2021.

[hea] headspin. A Complete Guide to Game Testing - Its Types and Processes. url: https://www.headspin.io/blog/game-testing-a-complete-guide-to-its-types-and-processes.

[Spe21] Speedrun. GLITCH HUNTING: WHAT IS IT AND HOW DO YOU DO IT? url: https://www.speedrun.com/news/826-2021-7-1-glitch-hunting-what-is-it-and-how-do-you-do-it, 2021.

[Uni] Nintendo Unity. How Broken is Mario Kart 8 Deluxe? url: https://www.youtube.com/watch?v=5Cc60ZBl9Cs&ab_channel=NintendoUnity.