Essay : Reinforcement Learning for Automated Playtesting

- Sadhana (Reshi Krish Thanga Jawahar) EC22185

Introduction

Playtesting is an essential part in the development pipeline of a game, It provides developers with insight on how enjoyable the game mechanics that have been developed are and also helps in identifying bugs and glitches which creep into the game. But it is also a very resource intensive process as testers are required to run through the game multiple times trying out different components of the game. Playtesting is also required to be done for every major change which is implemented. With how resource intensive this process is, it is often ignored leading to ill optimised and buggy games being released into the market.

Parallelly the Game development Industry has been moving towards a Data Driven Decision Making (DDDM) model[1], where data guides almost all decisions which are being taken and the ways in which we can use data to make informed decisions has increased exponentially. Though these techniques would prove immensely valuable in the playtesting phase of development, the most prominent use case for DDDM still remains to be post the release of games in fields such as player retention, balance patches and some monetization strategies. A preliminary reason for this could be the requirement for a large amount of data to run the DDDM techniques and this data can currently only be sourced post release where a large user base plays the game and provides the required data.

On the other hand, Reinforcement learning agents have developed by leaps and bounds in recent years, with agents using games as a method to showcase their capabilities, with some even able to defeat the world's best players in their respective games. Their performance however hasnt been limited to playing to win. By modifying the rewards system with intrinsic rewards or unique extrinsic rewards, we are able to simulate interesting behaviours by the reinforcement learning agents.

Coming at the intersection of these ideas is Automated playtesting, where we use computational methods to collect playtesting data to run through DDDM algorithms and inform the decisions to be made. Automated playtesting on its own is fairly new and a lot of academic research is being explored in recent years but there still exists a gap from academics and the Industry use case[2]. In this essay, I will focus on the current state of Reinforcement learning and its usability for Automated playtesting and try to address what are the issues stopping it from being adopted in the industry.

Current Research

Automated playtesting has been implemented using various approaches from statistical planning agents to scripted agents. Reinforcement learning for Automated playtesting is a

relatively new area of study which has huge implications, some base level research and analysis has been done in 2D and puzzle games with simple mechanics.
Starting from the very base, Reinforcement learning agents have been studied as an alternative to statistical planning agents in strategy games and have had some success, with the trained Reinforcement learning agents performing better than the rule based agents which are currently used to playtest these games in the industry[3]. Moving on to slightly more complicated games, it has been used to develop agents which mimic human behaviours by defining personas based on player data, training agents based on that data and then using the agents to find alternative paths[4]. It is quite evident that a lot of research has been done around the different types of Reinforcement learning agents that we can use to playtest games but the question is, What does all this translate to when looking at modern games with huge worlds and complex navigation mechanics and can we form any meaningful inferences from how the agents behave or how to use the data generated for DDDM.
Now many big studios have shown interest in studying Reinforcement learning methods especially in huge worlds where manual coverage of all possible states is almost impossible and having unchecked bugs is highly likely. For the scope of this essay, I will be looking in depth into developments made by the research division of EA, SEED(Search for Extraordinary Experiences).
For formally assessing the developments made in each research paper on Reinforcement learning for automated playtesting for navigation in complex worlds,I have identified some key factors

1. The world setup and movement mechanics - World Complexity
2. The agents perception, rewards system and available interactions - RL agent
3. State abstraction, data collection and visualisation for aiding in design considerations - Data usability

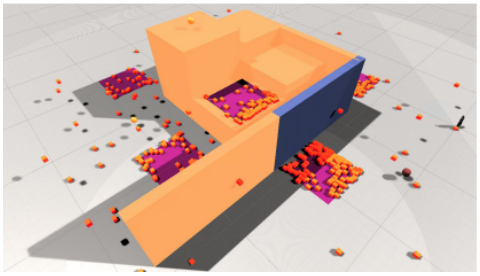
These are the key aspects based on which we will now be discussing the following papers.

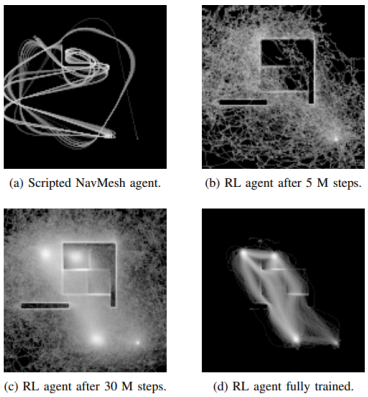
Augmenting Automated Game Testing with Deep Reinforcement Learning[5]

Being the pilot paper for Reinforcement learning in automated playtesting by SEED, this paper serves as a starting point for future research, identifies few basic types of bugs and exploits which can be identified by implementing Reinforcement learning agents and runs comparisons for the basic agent deployed when compared with other types of agents used for automated playtesting.
The agent is run in a very simple world with external rewards for giving the agent a goal, intentionally placed bugs and exploits to see if the placed bugs can be identified through the analysis of data which is generated when playtesting. The agent perceives its current state as a feature vector of its location relative to rewards, the current velocity and the state in which it is in. A key point to make note of is the use of continuous input for the agent controller instead of Navmeshes which are generally used for NPC movement in games, this has been done to ensure that the interaction of the agent with the game is similar to a human testing the game.

But as it is the first paper touching upon this subject from SEED, the data visualisation hasn't been focused on and the paper is focused completely on building the perception of the Reinforcement learning agent.
Even though there was no focus on data visualisation, the paper proposes some methods via which it can be used for Bug detection.

1. They had identified platforms where a player could get stuck(a bug which was implemented intentionally) by looking at the locations in which the agent had timed out before reaching the goal in each playthrough





(a) Scripted NavMesh agent. (b) RL agent after 5 M steps.

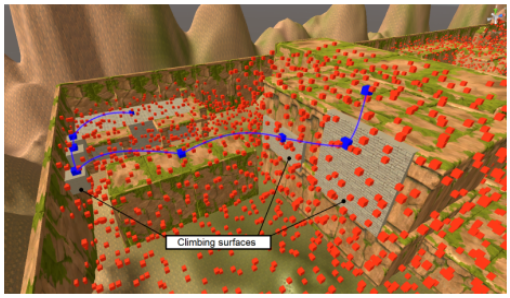
(c) RL agent after 30 M steps. (d) RL agent fully trained.

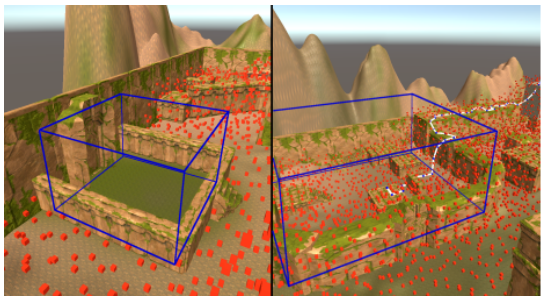
2. They had also compared the game state coverage of the developed Reinforcement learning agent and compared it with Scripted agent, and proposed the agent as a method for identification of possible paths to the destination and also for difficulty analysis of the game

Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents [6]

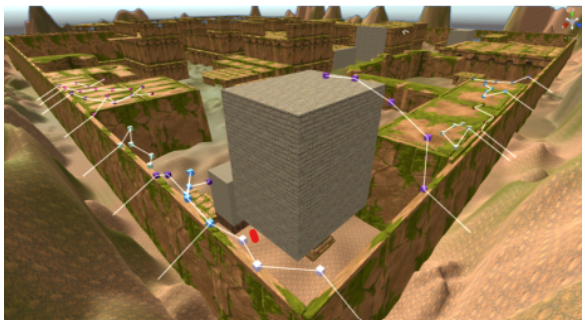
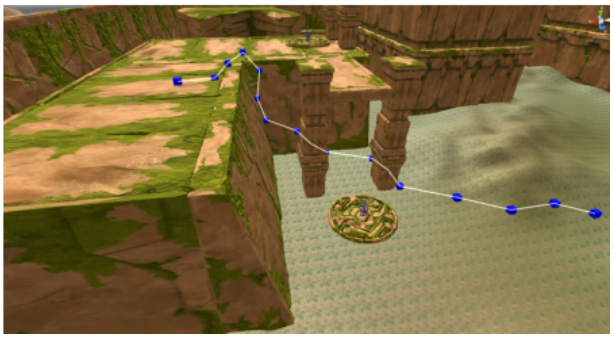
Building upon the previous paper, this paper fills in the gaps left by the previous paper and brings the focus to data visualisation techniques, all the while proposing a different Reinforcement learning agent with other playtesting goals.
The agent is defined with a curiosity based intrinsic reward system so that external rewards for guiding the agent behaviour doesn't need to be placed and the agent traverses all possible states present within the world. The paper also builds upon the complexity of the world, identifies regions which for design considerations should not be accessible by the agent and checks whether the agent is truly restricted from accessing those regions. Even with all these developments the most important aspect of the paper though, is the types of data visualisations that it has proposed namely the states coverage visualization by using state abstraction and a connectivity graph showing the paths taken by the agent to move from one abstracted state to another. Using these visualization methods the paper showcases multiple types of bugs which can be detected using Reinforcement learning agents.
1. The accessibility and inaccessibility of different regions in the map can be identified through the use of state coverage. The image on the left shows the agent accessing spaces which require complex navigation mechanics whereas the image on the right shows the agent getting

into regions which shouldn't be accessible to it and changes to the environment can be made inferring from the available data.

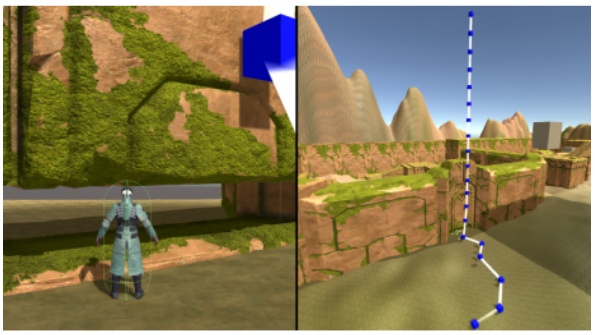




2. The agent is also able to discover paths that exit the Game worlds play area and alternative behaviours from the desired behaviour that reaches some desired locations. With this information, the designers can decide how they would like to modify the game world. The image on the left shows the paths the agent took to get outside the game world and the image to the right shows an alternate path it found instead of using the moving platform.

3. Additionally the paper also shows a case in which a bug caused by a random physics interaction was detected using the Reinforcement learning agent and using the connectivity graph, the detected bug is also easily recreateable which often ends up being a problem with human playtesters where a hard to find bug may be detected but recreating that said bug will be very difficult.



Towards Informed Design and Validation Assistance in Computer Games Using Imitation Learning[7]

The previous paper had showcased multiple ways of data collection and how information can be inferred from the generated data. Moving ahead this paper tries to optimise the means through which this is achieved focusing primarily on the Reinforcement learning agent as the training time required to produce an agent through which data could be inferred is extremely high. The paper proposes a method of using data generated by human playtesters as a starting point for

the agent for reducing the amount of time and samples that are required for training the agent acting somewhat like a midway point between scripted agents and Reinforcement learning agents. It also brings the complexity of the worlds up to par with modern platformers, with advanced gameplay mechanics.
The paper also conducts a survey on the desirability of the proposed methods for automated playtesting and claims that most of its respondents responded positively about how they feel the proposed agents will help in their own Game development process by cutting short the extremely timetaking process of playtesting.

Modl AI and Deepmind lab

Filling in for the need of a framework for automated playtesting by Reinforcement learning agents comes in Modl AI and Deepmind lab, both of which are frameworks built as a bridge between modern game development and automated playtesting methods.
Modl AI is built such that it can be layered on top of a developed game on unreal or unity and run automated playtesting with Reinforcement learning agents. Currently the level of playtesting is very limited and there needs to be improvement before this can be reliably used for game development.
Deepmind Lab on the other hand is a testing environment for Reinforcement learning agents and is a completely research oriented framework.

Conclusion

Reinforcement learning has a lot to offer to the field of automated playtesting as seen from the research papers that are mentioned above but there is still a gap from what has been developed by the academics and what is being used in the industry. Contrary to what is claimed by the survey conducted by SEED[8], An alternative academic survey[2] claims that most developers do not see Automated playtesting methods being used for development in the near future. They claim that almost all the research that has been done has made the environment in such a way that it tests the agent and not the agent testing the environment. I find myself accepting this claim as a shortcoming that has to be fixed before the industry takes on Automated playtesting.
By directly addressing the shortcoming mentioned, we realise the need to have a framework which has the flexibility of adding different types of agents, generating different types of data from it and most importantly it must be possible to directly layer it on top of a developed game without much modification. For automated playtesting in Unity, a framework for scripted agents is being developed by SamanthaSlake called pathOS[9]. In that framework we are easily able to port in developed games and run tests with scripted agents, with very minimal intrusion and provides some data visualization in the form of heatmaps. The most immediate need for using Reinforcement learning for automated playtesting in games from the industry would be to build a similar framework for Reinforcement learning agents.Further research needs to also continue on the front of different agent architectures, different types of data generation/visualization and testing of the built agents in more complex worlds to show the benefits of developing these methods for the industry.

Overall, I believe Reinforcement learning has huge scope for Automated Playtesting. It is primarily because Reinforcement learning agents behaviour ingame are simulated very similar to how players interact with the game and can provide a lot of data in much lesser time when compared to manual playtesting. Though it has a lot of positives it can not completely replace manual playtesting as playtesting not only deals with bugs and glitches but also with some more subjective questions such as how the game felt and was it fun and for these we would still require a human playtester, having an AI answer these questions doesn't seem to be very far off but that is not today.

References

[1] Data-Driven Game Development: Ethical Considerations
[2] Towards Automated Video Game Testing: Still a Long Way to Go
[3] Playtesting in Match 3 Game Using Strategic Plays via Reinforcement Learning
[4] Playtesting: What is Beyond Personas
[5] Augmenting Automated Game Testing with Deep Reinforcement Learning
[6] Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents
[7] Towards Informed Design and Validation Assistance in Computer Games Using Imitation Learning
[8] Artificial Players in the Design Process: Developing an automated testing tool for game level and world design
[9] PathOS+ : A New Realm in Expert Evaluation