The use of different artificially intelligent agents in software development using different programming paradigms.

COMP130 - Software Development

1702208

March 20, 2018

Please include an abstract of at most 100 words (these do not count towards your word count). In this paper, the author is going to discuss the use of different artificially intelligent agents in software development using different approaches. This includes five agents classified by Stuart Russell and Peter Norvig (CITE): 1) Simple reflex agent; 2) Model-based reflex agent; 3) Goal-based agent; 4) Utility-based agent; 5) Learning agent. An agent is anything that can perceive its environment through sensors and act through actuators depending on the environment. This can be anything, starting from simple machines, such as thermostats (which are programmed to act depending on the conditions) to worms (which can learn a small repertoire of behaviour) to humans (who can learn very quickly).

1 Introduction

Throughout the last few centuries "computer" has become one of the most important and powerful technologies we have now. Therefore, the field of engineering has become very severe today. Software engineering refers to the disciplined and scientific approach to developing advanced software. The field has made a wide range of software that supports different systems and applications. Such are artificially intelligent agents, or autonomous agents, which are going to be discussed in this paper. Artificially intelligent agents are a big field in software development, as there are being produced very advanced systems using agents. This would improve and ease people's lives as well as developing games using complex artificial intelligent (AI) systems. Moreover, in this paper will be discussed what programming paradigms are better for this area. In addition, whether it is valuable to use such approaches. Conclusions will include how these agents can be used in software development.

2 Artificially intelligent agents

"An agent is anything that perceives an environment through sensors and acts upon it through effectors." (Russell and Norvig, p. 31) This can be anything, starting from simple machines, such as thermostats (which are programmed to act depending on the conditions) to worms (which can learn a small repertoire of behavior) to humans (which can learn very quickly). The most basic agent is simple reflex one. It follow the *condition-action rule*, which is described as - if condition then action. This type of agents is not used in advanced software or game development. Rather they are used in everyday life, such as a thermostat. To have some meaningful use in software, agents must, at least, have different algorithms, which will make them act differently depending on the given task. Such can be considered as model-based agents. These have a certain list of rules, which they follow depending on the given task. Those rules are considered by the programmers beforehand, so it would not be flexible enough to execute random tasks. In addition, rules are executed as soon as it suits the condition, which doesn't usually lead to the best outcome. For this reason, goal-based agents are introduced. Goal-based agents have the same list of rules, but they consider executing the best one to have the best outcome so that it reaches its goal faster. That way the flexibility and capabilities of the agent are expanded. In this type of agent, search and planning are the subfields of the AI. That's why it first considers different scenarios before acting. The utility-based agent is more adaptive by adding a utility function. Unlike the previous one, this agent has "happiness" to consider. It compares how "happy" will the agent be for every outcome. A programmer cannot predict all possible outcomes that our world can give, the given goals can help to constantly reassess its situation. It will make the agent learn through errors adding new ways of reaching the best possible outcome. These two agents are based out of model-based one, giving them the basic behavior algorithm, while making it possible to improve however the customer wants it to be. However, they cannot act upon its environment when not given such rule by the programmers. Thus, the learning agent becomes very useful. As the name speaks for itself, it can act in an unknown environment. This type of agent has the "learning element", which is responsible for making improvements. This element uses feedback from the"critic" on how the agent is doing and determines how the "performance element" should be modified to do better in the future. Whereas the "performance element" is the entire agent - it takes in percepts and decides on actions. And the last, but nonetheless important component is "problems generator". It suggests actions to the agent that will lead to new and informative experience. This agent can exist in many environments, as it teaches itself. And is very flexible to help many softwares ease the usage of the product. Although there is no need to make every softwares with such agent. Many softwares are doing simple actions, which require basic agents. But there are also so many opportunities to make an advanced agent that would improve everytime its being used. This would lead to even more complex softwares that would be very useful in complicated devices. Which in turn would ease people's lives as well as different developments for games.

3 Programming paradigms

This sections will discuss about major programming paradigms and how they can be used in making agents discussed in the previous section.

4 Agents in games development

In this section, the author is going to discuss about how agents can be used in games development and whether it is useful doing so as well as discussing what paradigms would be best in making AI.

5 Conclusion

In conclusion, the author is going to write a conclusion about everything that has been discussed earlier and ponder whether it can be used in many cases.

References

[1] C. J. Hanna, R. J. Hickey, D. K. Charles, and M. M. Black, “Modular reinforcement learning architectures for artificially intelligent agents in complex game environments,” in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on.* IEEE, 2010, pp. 380–387.

[2] G. G. Helmer, J. S. Wong, V. Honavar, and L. Miller, “Intelligent agents for intrusion detection,” in *Information Technology Conference, 1998. IEEE.* IEEE, 1998, pp. 121–124.

[3] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach.* Prentice hall Upper Saddle River, 2003, vol. 2, no. 9.

[4] N. R. Jennings and M. Wooldridge, “Applications of intelligent agents,” in *Agent technology.* Springer, 1998, pp. 3–28.

[5] C. Si, Y. Pisan, and C. T. Tan, “A scouting strategy for real-time strategy games,” in *Proceedings of the 2014 Conference on Interactive Entertainment.* ACM, 2014, pp. 1–8.

[6] “Introduction to intelligent agents,” http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php, accessed: 2018-02-25.

[7] D. Johnson and J. Wiles, “Computer games with intelligence,” in *Fuzzy Systems, 2001. The 10th IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 1355–1358.

[8] C. Fairclough, M. Fagan, B. Mac Namee, and P. Cunningham, “Research directions for ai in computer games,” Trinity College Dublin, Department of Computer Science, Tech. Rep., 2001.