Building a Planner for Minecraft*

Julian Neubert, Carlo Rapisarda, Mart Kartaev, Nicolas Zimmermann
KTH Royal Institute of Technology
juliann@kth.se, carlora@kth.se, kartasev@kth.se, nzim@kth.se

Abstract

In this work, we exploit the open-source framework Malmo to build an Artificial Intelligence connected to the popular computer game Minecraft. We explore planning methods with hierarchical actions to control the main character and achieve predefined goals. Our final results were encouraging, and we believe that with further research we could manage to the guide the agent into performing complex tasks such as building a house within the game.

1 Introduction

Minecraft is a sandbox survival video game about gathering resources and crafting, or building things with what we have gathered, making it an excellent platform to test out Artificial Intelligence. For this project, we used an open source agent simulation framework named Project Malmo and developed by Microsoft [Johnson *et al.*, 2016]. Malmo is essentially a fully featured AI research platform. In the terms of the game, it is a modification allowing us to interact with the game-world and control characters by creating agents and giving them commands which can be any action a real player could have done.

Using planning we attempted to build an AI capable of interacting with this world, accomplishing simple and more complex tasks in order to both survive and create buildings or other structures. We started by defining our problem in a PDDL-like format and carried out the development component by component, accomplishing some of our initial goals despite limited time and resources.

2 Related Work

Before we begun our work, we researched the topics and gathered a few existing papers related to our project:

- The Malmo Platform for Artificial Intelligence Experimentation. [Johnson *et al.*, 2016]
- Offline Planning with Hierarchical Task Networks in Video Games. [Kelly *et al.*, 2008]

*https://minecraft.net

3 Approach

We started by defining the list of every possible action we would need for the highest level action of building a house (simplified as a box), dividing them into 3 categories: low, middle or high level action. We also defined the list of precondition fluents associated with those actions and the effects they would have. We defined these in a PDDL-like format so that we could easily use a planner to solve given problems. The low level actions were predefined by the APIs of the Malmo project (such as *move left* or *craft*), so we started by implementing a set of middle level actions.

Initially we developed a basic planner without optimization. We kept improving it as our agent became more complex to accommodate all of the tasks. The PDDL inspired planner has knowledge of the possible actions that can be correlated to each action, as well as some game specific knowledge and the list of precondition fluents to accomplish a given goal/action, together with the actions to perform to satisfy the preconditions.

4 Case study

As our case study, we decided to have the AI player interact with a simple flat world without additional factors, such as mountains or enemies. This allowed us to focus more on the planning aspects of the project, rather than having to deal with difficulties related to the game-play.

We divided our development of both the actions and planner into incremental steps getting more complicated each time, adding more capability as time went on and different sub-goals were reached. Our main goal was to get a planner as fast as possible, so we could do experiments and grow complexity in a controlled manner.

4.1 Observations from the world

This first fundamental part consists on retrieving information about the world in the form of a list representing the set of blocks around the character and extracting the information from this list. In this part we used Google's Gson library to simplify the interpretation of JSON observation strings to classes.

4.2 Simple movement

We then implemented a simple action to reach any (accessible) point in space using the low level actions provided by

Malmo's APIs. Originally we used a very simply movement, but as we saw this was not enough a more complex search based movement was implemented. We find a path around obstacles by using a Breadth First Search implementation. In order to test movement and action resolving we simultaneously implemented a basic planner that could evaluate the action based on a goal state and a the correlation between an effect and an action.

4.3 Looking around

Since to interact with blocks we need to point the camera at them, another important action is the one that allows us to control the *pitch* and the *yaw*. As this one is also a very basic action, we did not have to improve the planner in order to implement and test this idea. It was developed more or less simultaneously with the move action.

4.4 Selecting items

This was an important step in order to achieve our goals. We had to implement correlation mappings to select the correct tool for gathering resources and selecting the block we want to place. The improvements to the planner were mainly about dynamics as we focused on building a versatile structure that could independently evaluate a plan as we saw it.

4.5 Gather block

The action of gathering blocks is essential to build items; in this case we locate the interesting block with the observations, move next to it, look at it and gather it, making sure that we have an unobstructed line of sight.

4.6 Place block

This action is used when the agent wants to place a block in the world to build something; this is more complex than it seems, because when forming structures the agent needs to be aware of the position of the existing blocks and their spacial relationships.

4.7 Craft items

Sometimes we want to place a block different from the one we gathered, or we want to create new tools. This can be implemented using crafting, essentially combining items from the inventory together to receive new items.

4.8 Multiple goals

Before controlling the agent, a series of goals is set up and executed sequentially. Some accommodations to the planner were necessary. However, most of the infrastructure was already in place at this point. This step also heralded the end of our experimental and action development phase and brought us mainly into a planning world. Bugs arose later of course, as they often do. But the main development focus was on improving the planning process after this point.

At this point it had also become clear that some of our original aspirations were far too complex in order to be solved in a pure planning context. Instead of trying to implement the final stage of our goal we decided to focus on improving the planning process and experiment as much as possible within the capabilities of the Agent we had created.

4.9 Planner optimization

The optimization can be further split up into these sub-tasks:

- Dealing with failed states
- Plan memorization
- Implementing cost to evaluate preferable actions
- Reduction of actions to explore more efficient and sorting based on cost. Essentially a loose constraint approach

4.10 Result

We achieved most of the goals we were originally planning to look at. Amongst these are the following capabilities:

- Movement
- Inventory management
- Interaction with the world
- Resource gathering
- Use of blocks for basic building and crafting
- Combination of the above

Our main regret was the inability to fully realize a small house as we had intended. What had seemed like a simple task in the beginning turned out to possess numerous difficulties that we did not expect. Moving in the third dimension, line of sight and path-finding issues were amongst these problems. In addition, we decided not to focus on solving these issues as it would have taken the focus away from the planning perspective.

5 Conclusion

During the development, we encountered some unexpected problems unrelated to planning, which unfortunately hindered our progress.

In particular, some actions were difficult to put in place, requiring complex space geometry (for instance calculating if the *line of sight* from the character to a block is free, or computing the location of the spot to target to perform a certain action on it); the scale of this problem is substantially increased by the fact that the Minecraft world is a 3D world.

Implementing the planner as a mix of various fluents and actions was difficult and took the largest portion of our time. Although the advantages of using a PDDL-like structure quickly proved to greatly simplify the task of planning. Once we had all the definitions and ideas implemented as we wanted them, playing around with different preconditions, effects and action mappings provided some very interesting insights into AI planning. The way the program ended up working seems rather natural and it's easy to see how it can be expanded to do more thing. Having the planner infrastructure in place actually makes future development seem rather simple. One of the main avenues of research would be to find learning algorithms so that instead of defining actions manually, the AI could learn to behave in the world on its own.

In conclusion, we saw that in order to build a house as we had originally planned, we would have to implement complex learning methodologies (unrelated to planning, in some

cases) outside of the scope of this project. We saw a lot of parallels to other very interesting research topics, in particular artificial intelligence in video games and robotics. We were fascinated by the research papers we read on these topics and wished we had more time to implement our ideas. Every team member had at least one topic in the domain of AI which he will further look into in his free time. In honesty, our original dream goal was not fulfilled, but despite this, we were able to implement advanced planning methods and obtain satisfying results.

References

[Johnson *et al.*, 2016] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. page 4246, 2016.

[Kelly *et al.*, 2008] John-Paul Kelly, Adi Botea, and Sven Koenig. Offline planning with hierarchical task networks in video games. 2008.

[Russell and Norvig, 2009] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Pearson, 2009.