HerdCraft technical documentation

Julius Lindsberg

February 10, 2024

Contents

1 Introduction **2**
1.1 Reinforcement Learning 2
1.2 Work smarter, not harder! 2

2 XML Applications Programming Interface **3**
2.1 Logical and arithmetic operations 4
2.2 Element A 4
2.2.1 Attribute A 4
2.2.2 Attribute B 4
2.2.3 Attribute C 4
2.3 Element B 4
2.3.1 Attribute A 4
2.3.2 Attribute B 4

3 Functional Specification **5**
3.1 Class 1 5
3.1.1 Method A 5
3.2 Class 2 5
3.2.1 Method A 5
3.2.2 Method B 5
3.3 Class 3 5
3.3.1 Method A 5
3.3.2 Method B 5
3.3.3 Method C 5

4 Sequence of Implementation **6**
4.1 XML and planning phase 6
4.2 Dynamic Programming, 2D simulation 6
4.3 Forge mod, 3D proof-of-concept 6
4.4 title 6

1 Introduction

[Insert brief description of HerdCraft here]

1.1 Reinforcement Learning

The primary goal of AI in the mod is to implement a path-finding system which is simultaneously in line with the requirements for Richard Bellman's principle of optimality and is also capable of adapting to changes in the environment during the game in a way how real animals would (with limited line of sight). These requirements necessitate Reinforcement Learning and environmental models capable of dealing with statistical probabilities and uncertainty. This appears to be a classic use case for the Markov Decision Process (MDP).

An auxiliary goal of the project is to study machine learning in practice by implementing a reasonably modular RL library for the internal purposes of the mod. However, the plan here is to not use elements of machine learning or a training set before they become truly necessary. AI is occasionally treated as a magic wand which provides easy solutions under circumstances where traditional programming would be sufficient. Such cargo culting of AI is not the intended usage of machine learning for the mod.

1.2 Work smarter, not harder!

In addition to providing an API for XML editing and the technical documentation for internal classes and methods this document aims to divide up the project's workload into manageable subsections, which can be pursued in a chronological order on a timeline but without a set timetable for completion. Since there is no necessity for releasing a minimum viable product based on a set schedule, clear planning of classes and their relations beforehand reduces the risk for having to do major refactoring during later phases of development. This being a hobby project with no clients or pressures for release it appears preferable approach using a "Big Design Up Front" (BDUF) approach.

2 XML Applications Programming Interface

Listing 1: animals.xml

```
1
2 <!--
3     Herds are common livestock mobs, such as sheep, cows or pigs
          .
4     Ratio demarks the amount of generated vanilla mob herds that
           transition into Herd agents.
5 -->
6 <Herd mod="Sheep" ratio="1.0">
7     <!--
8     If one of these three behavioral patterns is not implemented
          , the probability for transitioning into them will just
          always be 0.
9     The actual underlying logic of Herd, Flock and School models
           will be kept as hard coded, but
10    the utilities and probabilities can be weighted depending on
           the mob (or rather, the RL agent) in question.
11    -->
12    <Exploration default="1">
13       <Transition mode="Domestication" epsilon="0.01">
14       </Transition>
15
16       <Transition mode="Dispersion" epsilon="0.1">
17       </Transition>
18
19    </Exploration>
20
21    <Domestication>
22       <Transition mode="Domestication">
23       </Transition>
24
25       <Transition mode="Dispersion">
26       </Transition>
27    </Domestication>
28
29    <Roaming>
30       <Transition mode="Domestication">
31       </Transition>
32
33       <Transition mode="Dispersion">
34       </Transition>
35    </Roaming>
36    <!--
37    Utility can be derived from a lot of things, such as blocks,
           environmental factors such as a biome type or
          temperature or a time of day.
38    Utility could be thought of as an energy source for a living
           organism.
39    -->
40    <Utility weight = "1.0">
41       minecraft:grass
42       <AND>sunlight</AND>
43    </Utility>
44
```

```
</Herd>
<!--
   Flocks are swarms of flying creatures, such as bats or birds.
-->
<Flock mob="Bat" ratio="0.0">
    <!-- Generates -->
    <Generates probability="0.01" min = "5" max = "30">
        <NOT>sunlight</NOT>
        <AND>minecraft:stone
            <OR>minecraft:granite</OR>
            <OR>minecraft:gravel</OR>
            <OR>minecraft:diorite</OR>
            <OR>minecraft:andesite</OR>
        </AND>
    </Generates>
    <Utility>

    </Utility>

</Flock>
<!--
   Schooling fish unsurprisingly always stay in bodies of water
       while frequently changing their location
-->
<School>
    <Migrates>
    temperature = COLD
    </Migrates>
</School>
```

[After animals.xml file is ready, divide it up into three separate files with herd, flock and school]

2.1 Logical and arithmetic operations

Some elements (utility and generation) support the use of basic linear arithmetic- (summation, multiplication, subtraction) and logical AND, OR and XOR operators. Recursive statements are also supported. However, excessively complicated utility functions are subject to produce sub-optimal results.

2.2 Element A

2.2.1 Attribute A

2.2.2 Attribute B

2.2.3 Attribute C

2.3 Element B

2.3.1 Attribute A

2.3.2 Attribute B

3 Functional Specification

[Insert class diagram image here]

3.1 Class 1

3.1.1 Method A

3.2 Class 2

3.2.1 Method A

3.2.2 Method B

3.3 Class 3

3.3.1 Method A

3.3.2 Method B

3.3.3 Method C

4 Sequence of Implementation

[General description of the roadmap.]

4.1 XML and planning phase

Where most of the contents and structure of this document are produced. The goal is to have a working understanding of what the major functions and ML elements are going to be before actually implementing the Forge mod.

4.2 Dynamic Programming, 2D simulation

During this phase of development a path-finding solution making use of dynamic programming is implemented in a 2D map. The map contains paths which are traversable or not and the traversibility of the map is subject to change during. This is the first time where it can be known what the algorhitmic requirements for training the RL Agent can be discovered.

4.3 Forge mod, 3D proof-of-concept

One of the big unresolved questions with this project is in how well a real three dimensional Minecraft map translates into the a graph structure. It is already known from a code-review that the Herd AI movements can be inserted into a priority-queue of Minecraft mobs.

4.4 title