Otto-Friedrich-Universität Bamberg

Smart Environments und Kognitive Systeme

SME-Projekt-B:
AI-Birds

Topic:

Meta

Autoren: Anne Schwarz, Susanna Cao
Betreuer/-innen: Prof. Dr. Diedrich Wolter, Prof. Dr. Ute Schmid
Bamberg, Sommersemester 2017

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Abstract

2 Introduction

3 Background

3.1

4 Key Results

Dieser Abschnitt beschäftigt sich mit unserer Umsetzung der Aufgaben des Meta-Bereichs. Unser Ziel war es, nach jedem Durchlauf zu "lernen", indem man das vorherige Handeln speichert, evaluiert und dies beim wiederholten Durchlauf berücksichtigt.

4.1 Level und Datenbank

4.1.1 Level

Zur Speicherung der ausgeführten Aktionen wird eine Datenbank benötigt. Doch bevor man solch eine für die verschiedenen Level bauen kann, müssen die grundlegenden Informationen kompatibel sein: das Level selber. Unsere Java-Klasse `Level` befindet sich im Ordner `Meta`. Diese besteht aus der Level-ID, den geschätzten maximal zu erreichenden Punkten dieses Levels, den tatsächlich erreichten Punkten, der Anzahl der gespielten Durchgänge und einer Liste von ausgeführten Schüssen. Das Zusammenfassen der Schüsse erfolgt durch die selber errichtete Klasse `Triplet`, die sich auch im `Meta`-Ordner befindet. Hierbei wird neben dem eigentlichen Schuss (`shot`) zusätzlich noch das anvisierte Zielobjekt (`target`) und die allein aus diesem Schuss erreichten Punkte (`damagePoints`) gespeichert, welche später in der `ShotSelection` relevant sind.
Die Klasse an sich dient als Grundlage und enthält dementsprechend nur wenige Methoden, wobei einige bereits von der Gruppe aus dem letzten Jahr geschrieben wurden und wir nur noch unsere Änderungen anpassen mussten (siehe `addExecutedShot`) bzw. die Methoden verbessert haben (siehe `calculateEstimatedMaximalPoints`).

4.1.2 Datenbank

Nach jedem Durchlauf eines Levels werden die oben genannten Informationen in ein `Level`-Objekt gespeichert. Jedes einzelne `Level`-Objekt wird dann in eine Datenbank hinzugefügt, welche unter database » LevelStorage zu finden ist.
Der Ordner `database` enthält eine weitere enum-Klasse `LevelState`, welche nur der Markierung der Level dient, weiter aber noch keine Verwendung findet.
Die `LevelStorage` besteht aus einer privaten Map, die die `Level` und den dazugehörigen `LevelState` beinhaltet. Zusätzlich enthält die Klasse eine öffentlichen Liste aus Integer, die in der gleichen Reihenfolge wie der Map die Level-IDs der gespielten Level speichert, sodass man von auSSen schnell auf die Information zugreifen kann, welche Level bereits gespielt wurden, sowie den Index der Level leichter abfragen kann.
Beim Speichern der Level muss darauf geachtet werden, dass man die Level nicht doppelt speichert im Falle eines wiederholten Versuchs. Daher prüfen wir in unserer öffentlichen Methode `addLeveltoStorage`, ob das übergebene Level bereits in der Datenbank erhalten ist. Falls es einen Eintrag mit dieser Level-ID gibt, wird die Hilfsmethode

`updateLevelInfo` aufgerufen, welche nur die geänderten Einträge aktualisiert, anstatt einen komplett neuen Eintrag zu erstellen.
Die `LevelStorage` ist in der Evaluation von groSSer Bedeutung und wird in den Klassen der nachfolgenden Kapitel verwendet.

4.2 Level Selection

Die Klasse `LevelSelection`, die sich im Ordner `Meta` befindet, ist für die Levelauswahl zuständig. Sie hält die Information über die gesamte Anzahl der zu spielenden Level und das Level, das gerade gespielt wird. Die Hauptmethode `selectNextLevel` beginnt mit einer zufällig ausgewählten Levelnummer und geht beim ersten Durchlauf alle Level der Reihenfolge nach durch.
Sobald alle Level einmal durchgespielt wurden, muss nun entschieden werden, welche Level in welcher Reihenfolge und wie oft wiederholt werden sollen.
Die Auswahl erfolgt nach einer simplen Wahrscheinlichkeitsberechnung für jedes einzelne Level, wobei das Level mit der höchsten Wahrscheinlichkeit ausgewählt wird:

$$Probability = 1 - (actualScore/maximalReachableScore)$$

Eine Besonderheit gibt es für verlorene Level. Natürlich werden diese als erste ausgewählt, da ihre Wahrscheinlichkeit 1 beträgt. Da man für jedes Level im Durchschnitt mindestens drei Minuten erhält [1] und wir von einer Durchschnittsspieldauer von 1 - 1,5 Minuten pro Level ausgingen, entschieden wir uns, die verlorenen Level zunächst höchstens zweimal wiederholen zu lassen. Falls die Quote der verlorenen Level dann immer noch zu hoch war, sollte noch ein Durchgang gestartet werden. In unserem Fall, lieSSen wir den Agent die verlorenen Level noch ein letztes Mal spielen, wenn die Quote der Verlorenen über 15% beträgt, d.h. der Agent würde bei einer Gesamtanzahl von 21 Level einen erneuten Versuch starten wenn mehr als 3 Level noch verloren sind. Falls dann noch verlorene Level übrig sind, sollen diese ignoriert werden. PseudoCode:

Algorithm 1 miniSAT

loop

[1] https://aibirds.org/angry-birds-ai-competition/competition-rules.html (zuletzt abgerufen: 05.09.2017)