



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Diplomarbeit

**Imitation von Verhaltensmustern eines
Agenten durch
Reinforcement Learning**

Andreas Diek
3577805

vorgelegt bei
Prof. Dr. Hans Kleine Büning

und
Dr. habil. Benno Stein

Inhaltsverzeichnis

1. Einleitung	9
1.1. Motivation	9
1.2. Aufbau der Arbeit	9
2. Beschreibung der virtuellen Umgebung	11
2.1. Das Spiel Quake3 Arena	11
2.1.1. Die Quake3 Welt	12
2.1.2. Agenten	13
2.1.3. Waffen	14
2.1.4. Bonusgegenstände	16
2.1.5. Die Arbeitsumgebung	16
2.2. Abgrenzung Bot - Avatar	18
2.2.1. Der Quake3 Bot	18
2.2.2. Der Avatar	23
2.2.3. Vergleich Bot - Avatar	23
3. Datengewinnung in Quake3	24
3.1. Auswahl der Attribute	24
3.1.1. Attribute des Agentenstatus	24
3.1.2. Attribute der Wahrnehmung	26
3.1.3. Attribute der Handlung	27
3.2. Extrahierung der Daten	28
4. Datenverarbeitung und Lernen	30
4.1. Methodik	31
4.1.1. Clustern	31
4.1.1.1. Distanzberechnung	32
4.1.1.2. Clusterverfahren	34
4.1.1.3. Clusteralgorithmen	35
4.1.1.4. Wahl eines Clusteralgorithmus	42
4.1.2. Reinforcement-Learning	43
4.1.2.1. Reinforcement-Learning-Verfahren	44
4.1.2.2. Wahl einer Methode des Reinforcement-Learnings	47
4.2. Umsetzung	48
4.2.1. Das Clustermodul	48

Inhaltsverzeichnis

4.2.1.1.	Vorüberlegungen zum Clustern	48
4.2.1.2.	Strukturierung der Daten	53
4.2.1.3.	Clustern der Daten	54
4.2.1.4.	Aufbereiten der Ergebnisse	54
4.2.1.5.	Das Modul	55
4.2.2.	Das Reinforcement-Learning-Modul	55
4.2.2.1.	Zustandsberechnung	56
4.2.2.2.	Wahl der nächsten Aktion	57
4.2.2.3.	Lernen	59
4.2.2.4.	Bilden eines online definierten Clusters	64
4.2.2.5.	Anpassung von Aktionswahrscheinlichkeiten	66
4.2.2.6.	Korrektur	69
4.2.2.7.	Das Modul	70
5.	Testphase und Evaluierung	72
5.1.	Die Steuerung des Reinforcement-Agenten in Quake3	72
5.2.	Testphase	74
5.2.1.	Parameter	75
5.2.2.	Szenarien	77
5.3.	Evaluierung	81
5.3.1.	Tests und Auswertungen	83
5.3.1.1.	Testphase 1	84
5.3.1.2.	Testphase 2	90
5.3.1.3.	Testphase 3	93
5.3.1.4.	Testphase 4	99
5.3.1.5.	Testphase 5	104
6.	Fazit und Ausblick	108
6.1.	Fazit	108
6.2.	Ausblick	110
A.	Anhang	112
A.1.	Dateien	112
A.1.1.	LOG-Datei	112
A.1.2.	ARFF-Datei	113
A.1.3.	CLU-Datei	115
A.1.4.	RLQ-Datei	116
A.1.5.	TST-Datei	116
A.2.	Funktionen	118
A.2.1.	Funktionen für die Datengewinnung	118
A.2.2.	Funktionen des Clustermoduls	118
A.2.3.	Funktionen des Reinforcement-Learning-Moduls	119
A.2.4.	Funktionen der Steuerung	119
A.2.5.	Gemeinsame Funktionen	120

Inhaltsverzeichnis

A.3. Bedienung	121
A.3.1. Installation	121
A.3.2. Aufzeichnung der Daten	121
A.3.3. Clusterprogramm	123
A.3.4. Der modifizierte Agent in Quake3	124
A.4. Inhalt der CD's	126

Abbildungsverzeichnis

2.1. Ein Agent in Quake3	13
2.2. Waffe Machinegun mit Munition	14
2.3. Waffe Shotgun mit Munition	15
2.4. Waffe Plasmagun mit Munition	15
2.5. Waffe Rocketlauncher mit Munition	15
2.6. Bonusgegenstände	16
2.7. Das Szenario	17
2.8. Die Bots Bones und Phobos	20
2.9. Schichtenmodell der Botarchitektur	20
2.10. Bereiche und ein Pfad im AAS	21
2.11. AINetwork	22
3.1. Berechnung der Munition für die aktuelle Waffe in Prozent	25
4.1. Cluster	31
4.2. Manhattan und Euklidische Distanz	32
4.3. Clustern mittels kmeans	36
4.4. Dendrogramm	38
4.5. Begriffe für DBSCAN	40
4.6. Agentensteuerung	44
4.7. Value-Iteration	45
4.8. Q-Learning	47
4.9. Beispiel für die Berechnung von Distanzen	49
4.10. Zustand	57
4.11. Wahrscheinlichkeitsberechnung	57
5.1. Der Agent Crash	72
5.2. Einlesen der modifizierten Steuerung in Quake3	73
5.3. Szenario 1	78
5.4. Szenario 2	79
5.5. Szenario 3	79
5.6. Szenario 4	80
5.7. Szenario 5	81
A.1. Auszug aus einer Log-Datei	112
A.2. Datenblock einer Arff-Datei	113

Abbildungsverzeichnis

A.3. Kopf einer Arff-Datei	114
A.4. Auszug einer Datei zur Speicherung der Cluster	115
A.5. Auszug einer Datei zur Speicherung der Aktionswahrscheinlichkeiten . . .	116
A.6. Beispiel für eine Datei zur Speicherung der Lernergebnisse	117
A.7. Start	122
A.8. Setup	122
A.9. Hinzufügen eines Agenten	123
A.10. Eingabe von Befehlen über die Konsole.	125

Tabellenverzeichnis

2.1. Boteigenschaften	19
3.1. Attribute Zustand - Wahrnehmung - Handlung	28
4.1. Einteilung der Attribute in Zustandsklassen	50
5.1. Einstellmöglichkeiten für das Lernverfahren	76
5.2. Bewertungskriterien	82
5.3. Parameterbelegung und Ergebnisse für Test 1A	84
5.4. Parameterbelegung und Ergebnisse für Test 1B	85
5.5. Parameterbelegung und Ergebnisse für Test 1C	86
5.6. Parameterbelegung und Ergebnisse für Test 1D	88
5.7. Parameterbelegung und Ergebnisse für Test 1E	88
5.8. Parameterbelegung und Ergebnisse für Test 1F	89
5.9. Parameterbelegung und Ergebnisse für Test 2A	91
5.10. Parameterbelegung und Ergebnisse für Test 2B	91
5.11. Parameterbelegung und Ergebnisse für Test 2C	92
5.12. Parameterbelegung und Ergebnisse für Test 2D	93
5.13. Parameterbelegung und Ergebnisse für Test 3A	94
5.14. Parameterbelegung und Ergebnisse für Test 3B	95
5.15. Parameterbelegung und Ergebnisse für Test 3C	96
5.16. Parameterbelegung und Ergebnisse für Test 3D	97
5.17. Parameterbelegung und Ergebnisse für Test 3E	98
5.18. Parameterbelegung und Ergebnisse für Test 4A	99
5.19. Parameterbelegung und Ergebnisse für Test 4B	100
5.20. Parameterbelegung und Ergebnisse für Test 4C	101
5.21. Parameterbelegung und Ergebnisse für Test 4D	102
5.22. Parameterbelegung und Ergebnisse für Test 4E	103
5.23. Parameterbelegung und Ergebnisse für Test 5A	104
5.24. Parameterbelegung und Ergebnisse für Test 5B	105
5.25. Parameterbelegung und Ergebnisse für Test 5C	106

Vorwort

An dieser Stelle möchte ich denjenigen danken, die zum Gelingen dieser Diplomarbeit beigetragen haben.

- Alexander Weimer und Andreas Goebels für die sehr gute Betreuung während der Arbeit.
- Sascha Großkurth für die gemeinsame Einarbeitung in Quake3 und die Implementierung einiger Funktionen für die Datengewinnung.
- Prof. Dr. Hans Kleine Büning für das Einverständnis die Bearbeitungszeit meiner Arbeit zu verlängern.
- Prof. Dr. Hans Kleine Büning und Dr. habil. Benno Stein für die abschliessende Lektüre und Bewertung meiner Arbeit.

1. Einleitung

1.1. Motivation

Durch die fortschreitende Entwicklung der Informationstechnologie hat die Menge elektronisch verfügbarer Daten in den letzten Jahren stark zugenommen. Diese enorm großen Datenmengen können wertvolle Informationen enthalten, die zum Verständnis von wichtigen Zusammenhängen innerhalb der Daten beitragen können. Im gleichen Zeitraum hat eine immer weiter ansteigende Automatisierung von Prozessen stattgefunden, so dass heute viele Aufgaben von Computern übernommen werden. In vielen Bereichen, z.B. in der Medizin, ist es aber nur zum Teil möglich Aufgabenbereiche von Computern übernehmen zu lassen. Ein weiteres Problem ist, dass ein Computer vorher stets so programmiert werden muss, dass er die Aufgabe erfüllen kann. Was aber wenn es gelingen würde, den Computer oder ein Programm anzuweisen, eine bestimmte Tätigkeit zu beobachten und sie perfekt nachzuahmen? Ein Mensch, eine Maschine oder ein anderes Programm könnte etwas vormachen und ein anderes Programm könnte ohne zusätzlichen Programmieraufwand dieses Verhalten imitieren. Da die Voraussetzungen für Mensch und Maschine allein durch ihre Anatomie völlig unterschiedlich sind und Roboter noch nicht so weit sind alle Bewegungen des Menschen zu simulieren, beschränkt sich diese Arbeit auf eine virtuelle Welt.

Am Beispiel des Computerspiels Quake3 Arena soll das Verhalten eines Agenten imitiert werden. Hierzu werden Datensätze aus dem laufenden Spiel generiert, die die Wahrnehmung, den Status und die Handlung eines Agenten enthalten. Zunächst wird mittels Clustern versucht, Zusammenhänge in diesen Daten zu finden, die Rückschlüsse darauf geben, ob sich bestimmte Momentaufnahmen eines Agenten wiederholen und ob daraus eine bestimmte Verhaltensweise abgeleitet werden kann. Kann eine Anhäufung dieser Momentaufnahmen gefunden werden, so können Zustände ermittelt werden, in denen sich der Agent im Laufe des Spiels befindet. Sein Verhalten wird für jeden Zustand dabei als Wahrscheinlichkeit für eine bestimmte Aktion abgelegt. Anschließend wird ein Agent in die virtuelle Welt integriert, der aufgrund der ermittelten Zustände und Aktionen durch Reinforcement Learning versuchen soll, Verhalten zu imitieren. Ziel dieser Arbeit ist es zu zeigen, ob es möglich ist, Verhalten zu erlernen und nachzuahmen.

1.2. Aufbau der Arbeit

Das Kapitel 2 erklärt den Aufbau einer virtuellen Welt und gibt eine Einführung in das Spiel Quake3 Arena. Zunächst wird ein Einblick in den Spielablauf gegeben, danach werden einzelne Bausteine des Spiels, sowie den in dieser Arbeit verwendeten Teil der Welt

1. Einleitung

beschrieben. Anschließend wird die Informationsverarbeitung beim computergesteuerten Agenten mit der Wahrnehmung des menschlichen Spielers innerhalb der Welt von Quake3 verglichen und Unterschiede herausgearbeitet.

Kapitel 3 diskutiert, welche Daten im Spielverlauf aufgezeichnet werden, um eine möglichst gute Kopie von Agentenverhalten zu erlangen und wie sie aus Quake3 extrahiert werden. Das nächste Kapitel stellt den Hauptteil dieser Arbeit da und beschäftigt sich mit der Vorgehensweise des Reinforcement-Learning bei der Imitation von Verhaltensmustern in der virtuellen Welt von Quake3. Dabei wird zunächst die grundlegende Vorgehensweise des Reinforcement-Learnings erklärt und Voraussetzungen geschaffen, um diese Lernmethode anwenden zu können. In diesem Zusammenhang wird ein Programm vorgestellt, mit dessen Hilfe die extrahierten Daten strukturiert, analysiert und daraufhin untersucht werden können, ob Zustände gefunden werden können.

In Kapitel 5 werden die aus Kapitel 4 umgesetzten Verfahren mit unterschiedlichen Bedingungen und Aufgabenstellungen für den Agenten getestet. Eine Auswertung dieser Tests soll danach Aufschlüsse bringen, wie gut die eingesetzten Lernverfahren zur Lösung der Problemstellung in dieser Arbeit beigetragen haben.

Im Kapitel Fazit und Ausblick werden noch einmal die Ergebnisse zusammengefasst und Anregungen für Erweiterungen und Verbesserungsvorschläge gemacht.

2. Beschreibung der virtuellen Umgebung

Um keine Zeit mit der Erstellung einer virtuellen Welt zu verlieren, wird eine bereits entwickelte Welt gesucht. Bei der Auswahl einer solchen Umgebung sollen verschiedene Kriterien erfüllt sein. Um Verhaltensmuster analysieren zu können, wird ein Umfeld benötigt, indem ein Agent interagieren kann. Dazu ist es erforderlich, Informationen zu protokollieren, die Einfluss auf die Aktion des Handelnden nehmen. Für die Protokollierung dieser Informationen und die Implementierung eines Verhaltenimitierenden Agenten ist es nötig, Modifikationen an der Welt vorzunehmen. Deshalb ist es wünschenswert, dass der Source-Code frei verfügbar ist.

Diese Kriterien werden von dem Spiel Quake3 Arena erfüllt, weil es einerseits für Agenten Möglichkeiten gibt, Einfluss auf die virtuelle Welt, in Form von Interaktion mit Objekten oder Bekämpfung von Agenten, zu nehmen. Andererseits können aufgrund des frei verfügbaren Source-Codes Schnittstellen implementiert werden, die Informationen über das Agentenverhalten im Spiel liefern und für die Programmierung eines Agenten geeignet sind.

Zunächst wird die hier verwendete virtuelle Welt untersucht und einige immer wiederkehrende, im Zusammenhang mit Quake3 stehenden Begriffe erklärt. Im weiteren Verlauf werden die Eigenschaften eines Agenten in Quake3 erläutert und explizit darauf eingegangen, wodurch ein bereits implementierter Agent in Quake3 charakterisiert werden kann. Als nächstes werden in zwei Abschnitten die im Spiel verfügbaren Objekte vorgestellt. Die wichtigsten sind zum einen Waffen und zum anderen Bonusgegenstände. Zum Abschluss der Einführung in Quake3 wird eine Teilwelt, welche als Arbeitsgrundlage dient, ausgewählt und die besonderen Merkmale dieser Umgebung herausgestellt. Im zweiten Teil dieses Kapitels wird die unterschiedliche Herangehensweise bei der Steuerung eines Agenten vom Mensch und vom Computer erläutert. Dazu wird in Kapitel 2.2 herausgearbeitet, inwieweit der menschliche Spieler und der computergesteuerte Agent Informationen aus ihrer Umwelt aufnehmen und verarbeiten.

In allen Abschnitten werden nur die Aspekte des Spiels genauer betrachtet, die für diese Arbeit relevant sind. Mehr Informationen zum Computerspiel Quake3 gibt es im Handbuch des Spiels[1] und im Internet[26].

2.1. Das Spiel Quake3 Arena

Quake3 Arena ist ein 3D-Shooter Spiel, welches in einer fiktiven Welt spielt. Zur Verfügung stehen vorgefertigte Welten (*Maps*) und Agenten, die sowohl vom Computer (*Bots*), als auch vom Mensch (*Avatar*) gesteuert werden können. Nach dem Start wird eine Map und bis zu 64 Agenten ausgewählt, die an dem Spiel teilnehmen. Eine Map enthält verschiedene Einstiegspunkte, in denen die Bots und die menschlichen Spieler beginnen.

2. Beschreibung der virtuellen Umgebung

Über das Hauptmenü können verschiedene Einstellungen vorgenommen werden, die sich auf das Spielgeschehen auswirken. So können Agenten mit unterschiedlichen Schwierigkeitsgraden hinzugefügt oder entfernt werden und ein Wechsel des Spielers in die Beobachterposition vorgenommen werden. Außerdem besteht die Möglichkeit einige Spieloptionen an die eigenen Vorstellungen anzupassen. Hierzu zählen u.a., der automatische Waffenwechsel, das Beschränken des vertikalen Blickwinkels auf eine feste Stellung und eine Option, welche es erlaubt auszuwählen, ob der Agent ständig läuft oder er die Möglichkeit besitzt sein Tempo zu variieren. Im Spielverlauf selbst können über die Konsole zusätzlich Befehle eingegeben werden, die beispielsweise, die Darstellung am Bildschirm ändern oder das Spielverhalten beeinflussen.

In Quake3 gibt es unterschiedliche Waffen mit verschiedenen Eigenschaften und Munition, außerdem Bonusgegenstände, mit denen der Agent seine Lebensenergie erhöhen, seine Rüstung verbessern oder für eine bestimmte Zeit seine Fähigkeiten steigern kann. Diese erscheinen während des Spiels immer wieder an festen Punkten. Ist ein Agent ausgeschaltet worden, steigt er nach ein paar Sekunden an einem zufälligen Einstiegsplatz wieder in das Spiel ein.

Die Spielhandlungen bestehen aus schnellen virtuellen Kämpfen, die zwischen Agenten ausgetragen werden. Dies kann sowohl im *Multiplayer* Modus, Mensch gegen Mensch, als auch im *Singleplayer* Modus, Mensch gegen Maschine erfolgen. Standardmäßig unterstützt Quake 3 Arena vier Spielvarianten.

Free For All - Deathmatch Ziel ist es, innerhalb einer bestimmten Zeit oder nach einer vorher definierten Anzahl an Abschusspunkten die meisten Gegner zu eliminieren.

1on1 - Tournament Wie "Free For All - Deathmatch", die Anzahl der Teilnehmer ist jedoch auf zwei Agenten begrenzt.

Team - Deathmatch Wie "Free For All - Deathmatch", mit dem Unterschied, dass zwei Teams gebildet werden, die gegeneinander antreten.

Capture The Flag Wie "Team - Deathmatch", Ziel ist es aber in dieser Variante, die Fahne des Gegners zu erobern und zur eigenen Basis zu bringen.

2.1.1. Die Quake3 Welt

Die 3D-Umgebung Quake3 besteht aus statischen Objekten und dynamischen Objekten (*Entities*). Statische Objekte sind Treppen, Wände, Lichtquellen, etc. Der Agent kann mit diesen nicht interagieren. Sie dienen lediglich als Rahmen in der Welt und als Mapbegrenzung in Form von geschlossenen Räumen, unüberwindbaren Hindernissen und Himmel.

Dynamische Objekte lassen sich in vier Gruppen einteilen. Eine Gruppe besteht aus Objekten, die sich an festen Punkten befinden und eine Funktionalität aufweisen. Dazu zählen Teleporter, Türen, Fahrstühle und Sprungplattform. Ein Sonderfall bilden Flüssigkeiten wie Wasser, Lava und Schleim, da der Agent *innerhalb* dieser Elemente agieren kann.

2. Beschreibung der virtuellen Umgebung

Zur nächsten Gruppe zählen aufnehmbare Objekte wie Waffen, Munition und Bonusgegenstände. Wie oben erwähnt, liegen diese an festen Punkten und erscheinen, nachdem sie aufgenommen worden sind, nach ca. 5-40 Sekunden an der gleichen Stelle wieder. Eine weitere Objektgruppe besteht aus allen temporär auftretenden Ereignissen. Hierzu gehören alle mit einer Waffe ausgelösten Schüsse wie beispielsweise eine Rakete. Zur letzten Gruppe zählen die frei beweglichen Objekte. Diese beinhaltet ausschließlich die Agenten.

Eigenschaften der Quake3 Welt:

- Jeder Punkt in der Welt hat feste Koordinaten.
- Wetteränderungen und verschiedene Tageszeiten werden nicht berücksichtigt.
- Objekte können Geräusche erzeugen.
- Die statischen Objekte sind unveränderbar und lassen sich nicht manipulieren.
- Die Welt kann dem Agenten Schaden zufügen in Form von Ertrinken, Verbrennen, Überschreiten der Mapbegrenzung und durch das Abfeuern einer Waffe.

2.1.2. Agenten

Ein Agent in Quake3 ist eine virtuelle Spielfigur, zumeist in menschenähnlicher Gestalt. Er kann sowohl vom Computer gesteuert, als auch vom Menschen befehligt werden. Jeder Agent besitzt Lebenspunkte (*Health*), die seinen Gesundheitszustand repräsentieren. Zusätzlich kann er sich durch eine Rüstung (*Armor*) verstärken. Die Lebensenergie und die Rüstung sind, abhängig vom Agenten und des eingestellten Schwierigkeitsgrades, begrenzt. Meist liegt diese bei 100 Punkten. Ist der Wert größer als diese vorgegebene Begrenzung, so verringert sich die Anzahl der Punkte pro Sekunde um eins. Sein Status wird weiterhin definiert durch seine aktuelle Waffe, sein Waffeninventar und die dazugehörige Munition. Außerdem beinhaltet der Status die Position des Agenten und seinen aktuellen Blickwinkel. Dem Agenten stehen eine Reihe von Aktionsmöglichkeiten



Abbildung 2.1.: Ein Agent in Quake3 (Quelle [1])

2. Beschreibung der virtuellen Umgebung

sogenannten *Basic Actions* zur Verfügung. Eine grundlegende Bewegung ist das Laufen, sowohl vorwärts und rückwärts als auch seitlich. Die Gesamtänderung der x und y Position des Agenten beträgt dabei im Durchschnitt 40. Der Blickwinkel bleibt bei diesen Aktionen stets der gleiche. Um in eine andere Richtung zu schauen, hat er die Möglichkeit, sich um 360 Grad zu drehen oder den Kopf auf der vertikalen Achse nach oben oder unten zu bewegen. Jede Drehbewegung des Agenten entspricht dabei einer Änderung des Blickwinkels um 15 Grad. Weitere Aktionsmöglichkeiten sind springen, ducken, schwimmen und tauchen. Die Aufnahme von Gegenständen sind ein Spezialfall. Hierbei ist zu beachten, dass es sich nur insofern um eine Aktion handelt, als dass sie automatisch ausgeführt wird, sobald der Agent über einen Gegenstand läuft. Besonders für eine Kampfsituation bieten sich dem Agenten noch zwei weitere Optionen, das Schießen und das Wechseln der Waffen. Ein Angriff auf einen anderen Agenten beinhaltet dabei, das Zielen auf diesen und das Abfeuern eines Schusses mit der aktuellen Waffe.

2.1.3. Waffen

Im Folgenden werden die im Spiel benutzten Waffen und deren Eigenschaften beschrieben. Diese sind dem Handbuch zu Quake3 entnommen[1].

Gauntlet Der *Gauntlet* ist seine Nahkampfwaffe. Sie befindet sich in ständigen Besitz und hat eine mittlere Durchschlagskraft.

Feuergeschwindigkeit:	—
Maximale Munition:	—
Anfängliche Munition:	—
Munitionstyp:	—

Machinegun Die *Machinegun* ist die Standardwaffe mit mittlerer Durchschlagskraft.

Feuergeschwindigkeit:	hoch
Maximale Munition:	200
Anfängliche Munition:	100
Munitionstyp:	bullets



Abbildung 2.2.: Waffe Machinegun mit Munition

2. Beschreibung der virtuellen Umgebung

Shotgun Die *Shotgun* ist besonders wirksam auf geringe Entfernungen. Sie verbraucht nur eine Munitionseinheit pro Schuss.

Feuergeschwindigkeit:	mittel
Maximale Munition:	200
Anfängliche Munition:	10
Munitionstyp:	shells

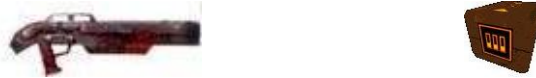


Abbildung 2.3.: Waffe Shotgun mit Munition

Plasmagun Die *Plasmagun* feuert Plasmakugeln, welche bei Erreichen des Zieles mittlere bis hohe Durchschlagskraft erzielen. Die Wirkung einer Plasmakugel ist auch in einem kleinen Radius um den Aufprall herum bemerkbar.

Feuergeschwindigkeit:	hoch
Maximale Munition:	200
Anfängliche Munition:	50
Munitionstyp:	cells

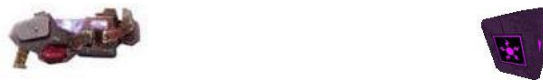


Abbildung 2.4.: Waffe Plasmagun mit Munition

Rocketlauncher Der *Rocketlauncher* gehört zu den Waffen mit dem größten Zerstörungspotential. Sie ist für den Nahkampf nicht zu empfehlen.

Feuergeschwindigkeit:	mittel
Maximale Munition:	200
Anfängliche Munition:	10
Munitionstyp:	rockets



Abbildung 2.5.: Waffe Rocketlauncher mit Munition

2. Beschreibung der virtuellen Umgebung

Desweiteren gibt es in Quake3 noch Waffen wie *Grenadelauncher*, *Railgun*, *BFG* und *Lightninggun*.

2.1.4. Bonusgegenstände

In der virtuellen Umgebung sind vor allem Bonusgegenstände zu finden, die den Schutz vor Treffern (Armor) und die Lebensenergie (Health) erhöhen (siehe Abbildung 2.6). Zusätzlich gibt es Bonusgegenstände, die die Fähigkeiten des Agenten beeinflussen. Diese erhöhen beispielsweise für eine kurze Zeit die Geschwindigkeit und die Kampfkraft oder lassen ihn unsichtbar werden.



Abbildung 2.6.: Bonusgegenstände *Health* (a) erhöht die Lebensenergie um 25 bis 50 Punkte, *Armor Shard* (b) verbessert den Schutz um 5 Punkte, *Red Armor* (c) vergibt einen 100 Punktebonus auf die Rüstung.

2.1.5. Die Arbeitsumgebung

Für die Verhaltensanalyse eines Agenten wird eine spezielle Umgebung gesucht, in der sich wiederholende Verhaltensmuster Anwendung finden können. Dies können das mehrmalige Passieren eines bestimmten Punktes in der Welt, das Durchqueren mehrerer getrennter Abschnitte in der gleichen Reihenfolge, die Aufnahme von bestimmten Objekten im selben Agentenzustand, bis hin zu Auseinandersetzungen mit anderen Agenten sein. Eine genaue Beschreibung der einzelnen Szenarien und Aufgabenstellungen findet sich in Kapitel 5. Dafür wird eine Welt gesucht, in der bestimmte Punkte eindeutig zu identifizieren sind und es mindestens zwei voneinander getrennte Räume gibt. Für die Benutzung von Objekten sollen Waffen und Bonusgegenstände in der 3D-Umgebung plaziert werden.

Im Folgenden wird eine spezielle Umgebung (*Map*) aus Quake3 vorgestellt, welche als Grundlage dieser Arbeit dient und den oben erwähnten Ansprüchen genügt. Die in dieser Arbeit verwendete Map ist in drei Abschnitte gegliedert und durch die Koordinaten 150 bis 1200 auf der x-Achse und -150 bis 2400 auf der y-Achse begrenzt. Die Anziehungskraft ist so gewählt, dass sie der auf der Erde entspricht.

Der erste Abschnitt ist ein grosser Platz unter freiem Himmel, der zweite Abschnitt ist eine Halle und der dritte Abschnitt verbindet die beiden mittels eines Durchgangs mit zwei separaten Flügeln (siehe Abbildung 2.7).

Auf dem Platz befinden sich zwei kleine Erhöhungen in den Ecken und zwei Statuen. In der Nähe der Statuen liegen jeweils drei Rüstungsteile (*Armor Shard*), in der Mitte

2. Beschreibung der virtuellen Umgebung

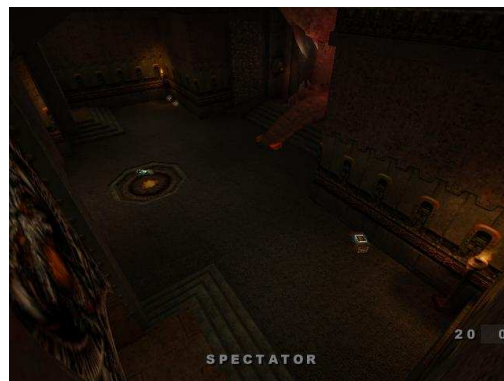
ein Raketenwerfer (*Rocketlauncher*), in einer Ecke *Plasmagun* Munition (*Ammo Cells*) und auf einer Erhöhung zwei Gesundheitspacks (*Health*). Der Bereich ist umgeben von einer unüberwindbaren Mauer, als Ausgang dienen drei Torbögen. Diese führen zum Durchgang, wobei sich auf ihrer Strecke eine *Shotgun*, *Machinegun* Munition und *Health* finden lassen. Über die beiden Flügel ist die Halle zu erreichen. Hier liegt in der Mitte eine *Plasmagun*, in einer Ecke zwei *Health* und in der anderen *Shotgun* Munition. Am Ende der Halle existiert ein Aufgang, der zu einer kompletten Rüstung führt.



(a)



(b)



(c)

Abbildung 2.7.: In Bild (a) ist der Platz zu sehen, am oberen rechten Bildrand die zwei Torbögen. In Bild (b) sind die drei Durchgänge zu erkennen, welche zu zwei verschmelzen und in der Halle (c) enden.

Um das Analysieren von Verhaltensmustern zu erleichtern, sollen die Aktionsmöglichkeiten des Agenten auf ein Minimum eingeschränkt werden. Zu diesem Zweck wurde bereits eine Welt gewählt, in der es nicht möglich ist, zu tauchen und zu schwimmen. Damit entfallen zwei verfügbare Aktionen eines Agenten. Mit den folgenden Einstellungen, soll

2. Beschreibung der virtuellen Umgebung

das Spiel in Bezug auf die Handlungsfreiheit eines Agenten weiter vereinfacht werden:

- Es findet ein automatischer Waffenwechsel statt, wenn eine Waffe aufgenommen wird. Ansonsten wird die Waffe nicht gewechselt. Hierdurch entfällt die Aktion „Waffenwechsel“.
- Der Agent befindet sich immer im Laufmodus. Dadurch, dass kein Umschalten zwischen mehreren Modi erfolgt, werden auch hier die Aktionsmöglichkeiten des Agenten eingeschränkt.
- Der Blickwinkel des Agenten ist auf der vertikalen Achse immer der gleiche, d.h. eine Bewegung ist nur auf der horizontalen Ebene möglich. Dies hat außerdem den Vorteil, dass unter Kenntnis der Position nur noch der horizontale Blickwinkel entscheidend ist, für die visuelle Wahrnehmung.
- Alle Bewegungen werden als Vorwärtsbewegung betrachtet. Damit entfällt eine Unterscheidung in Kriechen, Springen, Rückwärtslaufen oder Seitwärtslaufen.

2.2. Abgrenzung Bot - Avatar

In diesem Abschnitt werden die Gemeinsamkeiten und Unterschiede der Informationsverarbeitung eines in Quake3 vorgefertigten Bot und eines Avatars herausgearbeitet. Dies ist nötig, da aus den Daten, welche über einen Avatar gewonnen werden, ein Quake3-Bot entwickelt werden soll, der bestimmte Verhaltensmustern abbilden soll. Da von beiden Agentenformen voneinander verschiedene Informationen genutzt werden, um Agenten zu steuern, sollen die Unterschiede hier beschrieben werden. Viele der hier getroffenen Aussagen beziehen sich auf das Dokument[2].

2.2.1. Der Quake3 Bot

Als Bot wird ein computergesteuerter Agent bezeichnet. In Quake3 gibt es 32 vorgefertigte Bots, die jeweils über 35 unterschiedliche Charaktereigenschaften und fünf Schwierigkeitsstufen verfügen. Davon betreffen 14 Merkmale die Kommunikation, die ausschließlich für das Teamspiel benutzt werden. Da in Kapitel 5.2.2 diese Art des Spiels ausgeschlossen wird, kann auf eine nähere Betrachtung dieser Merkmale verzichtet werden. Die übrigen 21 sind in Tabelle 2.1 zusammengefasst (siehe hierzu auch [2]). Die meisten Eigenschaften haben Werte zwischen 0 und 1; je höher die Werte sind, desto mehr ist die Eigenschaft erfüllt. Sie beziehen sich fast ausschließlich auf das Kampfverhalten des Bots. Exemplarisch werden an dieser Stelle diejenigen Bots, *Bones* und *Phobos*, vorgestellt, die in dieser Arbeit zum Einsatz kommen.

Bones hat ein hohes Angriffs- und Zielgeschick und eine kurze Reaktionszeit. Während des Laufens springt er oft, sein Blickwinkel ist so ausgerichtet, dass er den Feind im Visier hat. Seine bevorzugte Waffe ist die Shotgun. Er ist fixiert auf Kampfhandlungen und sucht besonders die Konfrontation mit den Agenten, die ihn unter Beschuss nehmen. Er attackiert den Gegner auch, wenn sein eigener Gesundheitszustand gering ist.

2. Beschreibung der virtuellen Umgebung

Boteigenschaften		
Eigenschaft	Beschreibung	Inhalt
Name	Name des Bots	Anarki, Angel, Crash...
Geschlecht	Geschlecht des Bots	männlich, weiblich, neutral
Angriffsgeschick	Geschicklichkeit des Bots	0.0 - 0.2 keine Bewegung 0.2 - 0.4 Bewegung vorwärts/rückwärts 0.4 - 1.0 Bewegung im Kreis 0.7 - 1.0 zufällige Bewegung im Kreis 0.3 - 1.0 beim Rückzug auf Feind zielen
Waffenvorliebe	bevorzugte Waffen	Bsp.: Shotgun 5, Machinegun 20, ...
Zielgeschick	Schussgeschick des Bots	0.0 - 0.9 bei Feindbewegung zielen 0.4 - 0.8 in Richtung Feind zielen 0.8 - 1.0 exakt auf Feind zielen 0.6 - 1.0 auf Feindradius zielen 0.5 - 1.0 Zielvorhersage
Zielgenauigkeit	Zielgenauigkeit des Bots	0.0 - 1.0
Blickwinkel	Güte des Blickwinkels	0.0 - 1.0
max. Blickwinkel	max. Grad des Blickwinkelwechsels pro Sek.	0.0 - 360.0
Reaktionszeit	Reaktionszeit in Sek.	0.0 - 5.0
Hockstellung	Tendenz zum Ducken	0.0 - 1.0
Sprung	Tendenz zum Springen	0.0 - 1.0
Gehen	Tendenz zum Gehen	0.0 - 1.0
Raketensprung	Tendenz zum Raketensprung	0.0 - 1.0
Objektvorliebe	bevorzugte Objekte	Bsp.: Health 2, Armor 3...
Agression	Agressivität des Bots	0.0 - 1.0
Selbstschutz	Tendenz zum Selbstschutz	0.0 - 1.0
Rachsüchtigkeit	Rachsüchtigkeit des Bots	0.0 - 1.0
Stehenbleiben	Tendenz zum Stehenbleiben	0.0 - 1.0
Ordinäres Töten	Tendenz zur Gewalt	0.0 - 1.0
Wachsamkeit	Wachsamkeit des Bots	0.0 - 1.0
Dauerfeuer	Tendenz zum Dauerfeuern	0.0 - 1.0

Tabelle 2.1.: Boteigenschaften

2. Beschreibung der virtuellen Umgebung

Demgegenüber verhält sich der Bot *Phobos* wesentlich defensiver. Er sucht nicht so häufig den Kampf wie *Bones* und verfolgt seine Gegner weniger. Sein eigener Gesundheitszustand hat Priorität. Er zielt im Gegensatz zu *Bones* sehr ungenau, hat dafür eine höhere Reaktionszeit. Sein Blickwinkel ist nicht immer ideal. Diese Schwächen gleicht er aus,



(a)



(b)

Abbildung 2.8.: Die Bots *Bones* (a) und *Phobos* (b)

indem er häufiger in Deckung geht und seine bevorzugte Waffe der durchschlagskräftige *Rocketlauncher* ist.

Vier-Schichtenmodell der Informationsverarbeitung:

Die Informationsverarbeitung des Quake3 Bots lässt sich in einer Architektur mit vier Schichten darstellen (siehe Abbildung 2.9 aus [2]). Dabei fließen Informationen generell von den unteren in die oberen Schichten und Ergebnisse in die entgegengesetzte Richtung.

Erste Schicht:

Die erste Schicht enthält den Input (*Area Awareness System/AAS*) und den Output (*Basic Actions*) des Bots. Als Input erhält das Steuermodul des Bots alle Informationen

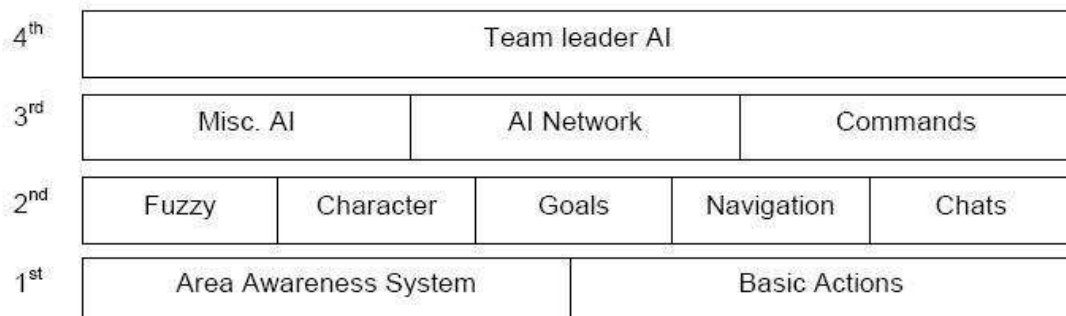


Abbildung 2.9.: Schichtenmodell der Botarchitektur

2. Beschreibung der virtuellen Umgebung

über den Zustand der Welt in Form von Variablen. Diese Abbildung der 3D-Welt beinhaltet Informationen über Navigation, Routing und alle *Entities* der Umgebung.

Damit der Bot navigieren kann, benutzt er kein klassisches *Way-Point-System*, sondern eine Verlinkung vieler kleiner Bereiche (*Areas*), die im AAS errechnet werden. Areas werden dabei so gebildet, dass innerhalb eines Bereiches der Bot von jedem Punkt aus jeden anderen Punkt durch eine Vorwärtsbewegung erreichen (siehe Abbildung 2.10) kann. Dies bedeutet, dass zwei Punkte immer dann, unterschiedlichen Areas zugeordnet werden, wenn eine imaginiäre Linie zwischen ihnen durch ein statisches Objekt der Welt führen würde. In der oben beschriebenen Map gibt es z.B. 1714 Areas.

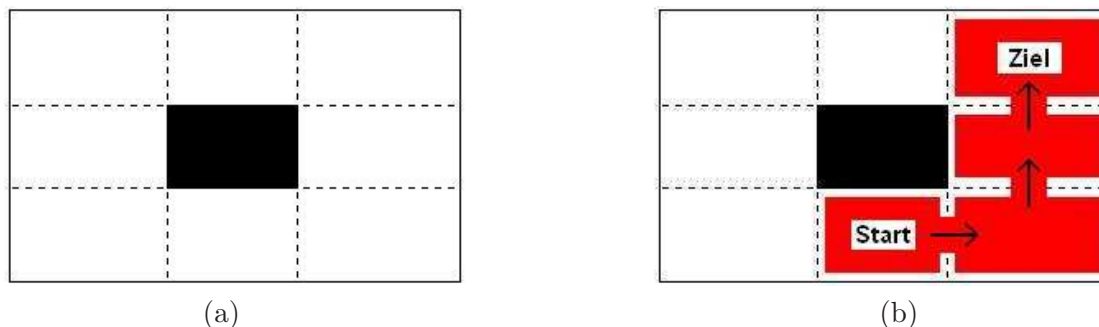


Abbildung 2.10.: Bereiche (a) und Pfad (b). Der dunkle Block in der Mitte symbolisiert ein unüberwindbares statisches Objekt in der Welt.

Sobald der Bot einen Punkt außerhalb seines aktuellen Bereiches erreichen will, sucht er sich mittels Routing einen Pfad dorthin. Durch die Abbildung der *Entities* erhält der Bot Informationen darüber, welches Objekt sich an welcher Position befindet.

Als Output werden mittels der Ergebnisse der Informationsverarbeitung grundlegende Aktionen generiert. Diese entsprechen ausschließlich den Aktionen, die in Kapitel 2.1.2 beschrieben sind.

Zweite Schicht:

In der zweiten Schicht werden die nächsten Ziele des Bots mittels Fuzzy Logik¹ ermittelt. Hierbei wird spezifiziert, in welchem Maße ein Bot etwas unternehmen, haben oder benutzen soll. Dabei werden Ziele durch die Fuzzy Logik aufgrund der Eigenschaften des Bots (siehe 2.1) und Informationen aus dem AAS bewertet. Das oberste Ziel (*Goal*) des Bots ist es, ein Spiel, wie z.B. Deathmatch, zu gewinnen. Um dies zu erreichen verfolgt er im Spiel Teilziele (*subgoals*). Ein Teilziel ist beispielsweise die Eliminierung eines Gegners oder das Ausweichen in einer Kampfsituation. Anhand dieser Ziele wird die kürzeste Route für den Bot berechnet (*Navigation*), um diese zu erfüllen. Diese Route ist nur ein "Vorschlag" für den Bot, da er jederzeit die Möglichkeit hat, einen anderen Weg

¹Mathematische Logik, die nicht nur zwei klar definierte Zustände kennt (wie wahr oder falsch, an oder aus), sondern auch Zwischenwerte, die als Grad der Wahrscheinlichkeit verstanden werden können (siehe freie Enzyklopädie [3]).

2. Beschreibung der virtuellen Umgebung

einzuschlagen, z.B. um ein Teilziel zu erreichen. Zusätzlich erlaubt die Navigation dem Bot, nicht nur einem Ziel zu folgen, sondern sich auch in eine beliebige Richtung zu bewegen. Die zweite Schicht ist außerdem dafür verantwortlich, eigene Nachrichten zu generieren und zu empfangen.

Dritte Schicht:

In der darüberliegenden Schicht wird das Kampfverhalten bestimmt. Das Hauptmodul des Bots ist das *AI Network*. Hier laufen alle Informationen zusammen und es wird entschieden, welches Ziel als nächstes verfolgt wird.

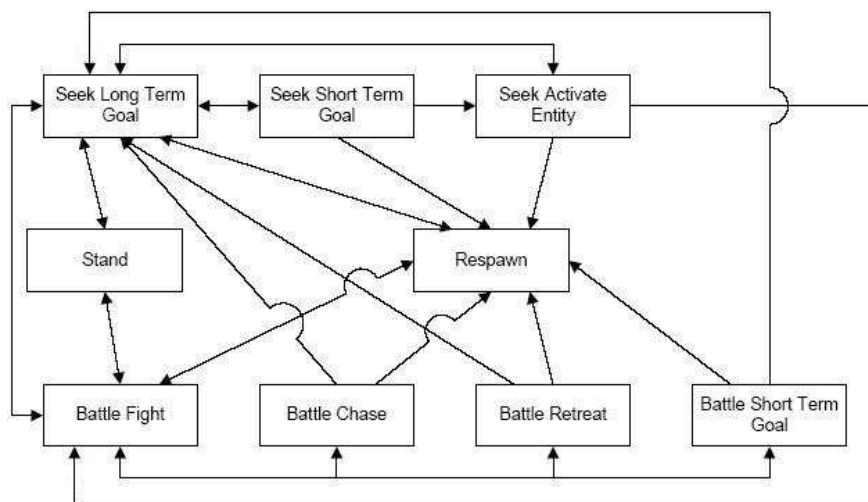


Abbildung 2.11.: AINetwork

Das AI Network ist ein endlicher Automat² mit elf Zuständen (siehe Abbildung 2.11) für verschiedene Situationen und Ziele. Aufgrund von Produktionsregeln sucht sich der Bot in der aktuellen Situation und den vorgegebenen Zielen den nächsten Zustand.

Die meisten seiner Ziele sind langfristige Ziele (*Long Term Goals*). Kurzfristige Ziele (*Short Term Goals*) hingegen ergeben sich immer dann, wenn das langfristige Ziel kurz unterbrochen wird, um beispielsweise einen zufällig in der näherliegenden Gegenstand aufzunehmen. In der dritten Schicht werden zusätzlich Befehle aus der Kommunikation entgegengenommen und abgesetzt.

Vierte Schicht:

Die oberste Schicht ist ausschließlich für das Teamspiel vorgesehen. Hier werden aufgrund des Kampfverhaltens Nachrichten an andere Mitspieler gesendet.

²Endliche Automaten (Abk. EA, engl. finite automaton - FA oder finite state Machine - FSM) sind mathematische Modelle von einfachen idealen Rechenmaschinen. Es sind Systeme mit Zuständen, Aktionen und einer Übergangsfunktion, die für jede Aktion den Nachfolgezustand des aktuellen Zustands berechnet[3]. Als endlich werden sie bezeichnet, wenn die Anzahl der Zustände einen festen Wert hat.

2.2.2. Der Avatar

Ein Avatar ist ein von einem Menschen gesteuerter Agent³. Als Output gibt das Steuermodul des Avatars eine Abbildung der 3D Umgebung auf dem Bildschirm und dem Lautsprecher aus, als Input erwartet er eine Eingabe über die Tastatur und die Maus. Für den menschlichen Spieler bedeutet dies, dass er sich aufgrund dieser audiovisuellen Informationen zu einer Aktion entschließt und diese mittels Eingabegeräte an das Programm zurückgibt.

2.2.3. Vergleich Bot - Avatar

Im Gegensatz zum Bot sind beim Avatar Input und Output vertauscht, d.h. das Steuermodul des Avatars verhält sich entgegengesetzt zu dem des Bots. Das Quake3 Programm gelangt zehn Mal pro Sekunde an den Punkt, wo es wissen muss, wie der Quake3 Agent als nächstens gesteuert wird. Hierzu gibt es zwei Möglichkeiten. Befiehlt das Programm den Agenten in Form eines Bots selbst, bekommt ein entsprechendes Steuermodul als Input den Zustand der Welt und gibt als Output die nächste Aktion zurück. Wird der Agent von einem menschlichen Spieler übernommen, so wartet das Programm auf eine Eingabe (Input) von außen. Die ausführbaren Aktionen sind also bei Avatar und Bot identisch, die Informationen werden unterschiedlich aufgenommen. Dies bedeutet, dem Avatar steht das AAS nicht zur Verfügung.

Hieraus folgt vor allem ein Unterschied in der Navigation. Der Grund dafür ist, dass die Welt des Bots in Bereiche eingeteilt ist, die miteinander verknüpft sind. Da dem Menschen diese Bereiche nicht zur Verfügung stehen, muss er sich einen Weg durch die virtuelle Welt ausschließlich über die Ausgabe des Bildschirms suchen. Ebenso unterscheidet sich die Zielfindung des Avatars gegenüber der des Bots, weil Position und Erreichbarkeit der Entities nicht immer aus der Bildschirmausgabe abzuleiten sind. Ein weiterer Unterschied liegt in der Lernfähigkeit. Diese beschränkt sich beim originalen Bot auf die Veränderung der in Abbildung 2.1 aufgelisteten Eigenschaften und Fähigkeiten.

Es müssen also Informationen in Quake3 bereit gestellt werden, die sowohl von einem Bot, als auch von einem Avatar genutzt werden können. Außerdem wird eine Modifikation benötigt, die einem Agenten das Erlernen eines vorgegebenen Verhaltens ermöglichen.

³Die Bedeutung des Wortes Avatars kommt aus dem Sanskrit und bedeutet die Fleischwerdung (Inkarnation) eines Geistes. In digitalen Umgebungen bekommt der Begriff eine ähnliche, jedoch sehr spezielle Bedeutung: Repräsentation einer Person innerhalb einer virtuellen Gemeinschaft in Form einer (Spiel-)Figur. Dabei treten Avatare in unterschiedlichsten virtuellen Gemeinschaften auf, in asynchronen Diskussionsforen, Chaträumen oder in grafischen (Spiele-)Welten[4].

3. Datengewinnung in Quake3

In diesem Kapitel wird erläutert, welche Informationen benutzt werden, um Verhaltensmuster eines Agenten zu finden und warum. Dazu werden Attribute gebildet, die die entsprechenden Informationen enthalten. Anschließend wird gezeigt, wie die dafür benötigten Daten aus Quake3 gewonnen werden.

3.1. Auswahl der Attribute

Wenn das Verhalten eines Agenten in einer virtuellen Welt analysiert werden soll, stellt sich die Frage, wodurch es beeinflusst wird. Es ist daher das Ziel, diejenigen Informationen zu erhalten, die einen Zusammenhang zwischen Handlungen eines Agenten und seiner Umwelt herstellen. Dazu werden Attribute gewählt, die sowohl die Bildschirmausgabe für den menschlichen Spieler als Informationen speichern, als auch später für die Steuerung eines Bots nutzbar sind. Wie in Kapitel 2.2 beschrieben sind diese Informationen für Avatar und Bot sehr unterschiedlich, weswegen am Programmcode von Quake3 einige Änderungen vorgenommen werden müssen. Diese werden im Anhang näher betrachtet.

Zunächst müssen Überlegungen angestellt werden, aufgrund welcher Daten ein Agent von einem Punkt A nach B gesteuert wird, wann er neue Munition, eine Waffe, etc. aufnimmt, wie er diese findet und welche Informationen benötigt werden, wenn es zu einem Konflikt zwischen Agenten kommt. Dazu werden Attribute definiert, die drei Klassen zugeordnet werden können. In der ersten Klasse werden alle Informationen bereitgestellt, die den Zustand des Agenten beschreiben. Die zweite Klasse beinhaltet Daten zur Wahrnehmung des Agenten. Alle Handlungen während des Spielgeschehens werden der dritten Klasse zugewiesen.

Insgesamt werden 18 Attribute ausgewählt, 13 nominale und fünf numerische (siehe Tabelle 3.1). Diese 18 Attribute spiegeln eine Momentaufnahme eines Agenten wieder und bilden eine Instanz.

3.1.1. Attribute des Agentenstatus

Zur Klasse des Agentenstatus gehören alle Informationen, die die Konstitution des Agenten beschreiben. Dies sind seine Gesundheit, seine Rüstung und die vorhandene Munition seiner aktuellen Waffe.

Das Attribut *state_health* beinhaltet den Gesundheitszustand eines Agenten. Es wird gebraucht, weil mit sinkender Lebensenergie der Einfluss auf die Aktion eines Agenten steigt. So könnte ein kritischer Gesundheitszustand zur Folge haben, dass ein Kampf vermieden oder *Health* aufgenommen wird.

3. Datengewinnung in Quake3

Durch *state_armor* werden die Rüstungspunkte des Agenten beschrieben. Für dieses Attribut gilt das gleiche wie für *state_health*, nur in einer abgeschwächten Form, weil eine schwache Rüstung nicht in dem Maße bedeuten muss, einen Kampf aus dem Weg zu gehen.

Auf ein Attribut zur Speicherung der aktuellen Waffe wird verzichtet, da auf die aktuelle Waffe durch den ausschließlich automatischen Waffenwechsel nur sehr wenig Einfluss genommen werden kann.

Das Attribut *state_munition* enthält die Anzahl an Munition der aktuell verwendeten Waffe und wird gebraucht, um zu erkennen, wann der Munitionsvorrat erschöpft ist. Eine mögliche Aktion kann daraufhin die Aufnahme eines Munitionspaketes sein. Da die

Aktuelle Waffe: Machinegun Verfügbare Munition für diese Waffe: 95 Anfängliche Munition für diese Waffe: 100 Munition in Prozent: 95 %
Neue Waffe aufgenommen
Aktuelle Waffe: Shotgun Verfügbare Munition für diese Waffe: 10 Anfängliche Munition für diese Waffe: 10 Munition in Prozent: 100 %
Neue Munition für die aktuelle Waffe aufgenommen
Aktuelle Waffe: Shotgun Verfügbare Munition für diese Waffe: 15 Anfängliche Munition für diese Waffe: 10 Munition in Prozent: 100 %

Abbildung 3.1.: Berechnung der Munition für die aktuelle Waffe in Prozent. Zu beachten ist, dass mehr als 100 % Munition nicht möglich ist.

anfänglichen Munitionen der Waffen stark voneinander abweichen, wird hier der prozentuale Wert bezogen auf den anfänglichen Wert angenommen. Dies ist erforderlich, weil die aktuelle Waffe des Agenten nicht zusätzlich in einem Attribut gespeichert wird und ansonsten die Munition von zwei unterschiedlichen Waffen nicht miteinander vergleichbar wären (siehe Abbildung 3.1).

Die nächsten Attribute gehören in die Kategorie der Wahrnehmung des Agenten. Sie werden hier behandelt, da ein Teil der Wahrnehmung durch Informationen ersetzt werden, die dem Zustand des Agenten zugeordnet werden.

Die Bewegung des Agenten ist stark von der aktuellen Umgebung beeinflusst. Wie in Kapitel 2.2 beschrieben, haben der Bot und der Avatar unterschiedliche Voraussetzungen für die Navigation. Wenn aus den extrahierten Daten analysiert werden soll, wie ein Agent von einem Punkt A zu einem Punkt B gelangt ist, können die Bereiche des AAS nicht genutzt werden, da sie dem menschlichen Spieler nicht zur Verfügung stehen. Da der Mensch aufgrund der Ausgabe des Bildschirms navigiert, folgt daraus, dass ein komplettes Abbild (Wände, Treppen, Gegenstände etc.) der Ausgabe in Daten gespeichert werden müsste, was eine riesige Datenmenge erzeugen würde. Um dies zu verhindern

und eine Analyse zu erleichtern, wird die Position und der Blickwinkel des Agenten als Hilfsmittel benutzt. Abhängig von Position und Blickwinkel ist die Ausgabe der statischen Objekte auf dem Bildschirm immer die gleiche. Genauso ist bei Bekanntheit der Position, der aktuelle Bereich des AAS ermittelbar. Das bedeutet, dass bei der Analyse der Daten äquivalent ist, ob Position und Blickwinkel, die Ausgabe von festen Objekten in der Welt oder die Bereiche des AAS analysiert werden. Somit können bei der Aufzeichnung der Daten alle statischen Objekte der Welt unberücksichtigt bleiben.

Zur Bestimmung der Position werden die Attribute *state_pos_x* und *state_pos_y* benötigt. Die Höhe des Agenten in der beschriebenen Map kann vernachlässigt werden, weil es in Abhängigkeit von der x und y-Position keine unterschiedlichen Standorte bezüglich der Höhe gibt. Der Blickwinkel des Agenten wird durch seine Aktionsrichtung ersetzt. Der Grund dafür wird in Kapitel 3.1.3 erläutert.

3.1.2. Attribute der Wahrnehmung

Attribute der Wahrnehmung werden der Kategorie des Sehens und des Hörens zugeordnet. Dabei werden nur dynamische Objekte berücksichtigt. Im Unterschied zu den statischen Objekten können dynamische Objekte nicht stets in gleicher Position und mit gleichem Blickwinkel wahrgenommen werden. Außerdem kann der Agent mit diesen dynamischen Objekten interagieren, somit ist es für die spätere Analyse der Daten wichtig zu wissen, wo der Agent etwas sieht und was es ist.

Sechs der insgesamt zehn Attribute beinhalten eine Richtungsangabe relativ gesehen vom Agenten. Hierzu gehören *see_dir_health*, *see_dir_weapon*, *see_dir_amm0*, *see_dir_armor*, *see_dir_bot*, *see_dir_enemy*.

Unter gesehenen Gegenständen fallen Gesundheitsboni, Waffen, Munition oder Rüstungsteile, unter gesehenen Gegnern Bots und Feinde, wobei ein Feind(Enemy) derjenige ist, der den Agenten zuletzt getroffen hat. Erblickt er beispielsweise eine Waffe, so wird in das Attribut *see_dir_weapon* die relative Position zum Agenten eingetragen. Zur besseren Lesbarkeit werden hier nicht Werte im 360 Grad Winkel benutzt, sondern die Ausrichtung an Himmelsrichtungen. Der Fall, dass mehr als eine Waffe gleichzeitig gesehen wird, kann in der vorgegebenen dreidimensionalen Umgebung vernachlässigt werden, da es nur selten vorkommt. Analog trifft dies auch auf die anderen Attribute dieser Kategorie zu. Wird in der aktuellen Momentaufnahme kein entsprechendes Objekt gesehen, so ist der Inhalt des Attributes „not_visible“.

Zwei weitere Attribute (*watch_player* und *watch_item*) speichern, ob sich ein anderer Agent oder ein Gegenstand im Visier eines Agenten befindet. Dabei wird davon ausgegangen, dass diese in einem eingeschränkten Blickfeld von horizontalen 5 Grad und vertikalen 6 Grad gesehen werden. Falls der Agent jemanden im Visier hat, enthält das Attribut *watch_player* ein „somebody“, falls dieser zuletzt auf den Agenten geschossen hat ein „enemy“, andernfalls ein „nobody“. In *watch_item* wird der jeweilige Gegenstand eingetragen, falls dieser sich unmittelbar im Blickfeld befindet. Der Nutzen dieser beiden Attribute liegt darin, dass durch das, was der Agent im Blickfeld hat, in vielen Fällen auf die momentane und zukünftige Handlung geschlossen werden kann. Ein Agent, der einen Gegner im Visier hat, wird eher genau diesen attackieren als einen anderen Gegner, der

nur im Sichtfeld ist.

Da der menschliche Spieler seine Informationen nicht nur durch den Bildschirm erhält, sondern auch durch die Lautsprecher, wird diese Klasse von Attributen durch das Hören erweitert. Dazu wird ein Attribut *hear_noise* definiert. Berücksichtigt werden nur die Geräusche, die von Agenten erzeugt werden, weil diese im speziellen das Kampfverhalten beeinflussen. Hierzu gehören Schritt- und Schussgeräusche sowie das Aufnehmen von Gegenständen und Waffen. Geräusche werden je nach Entfernung in zwei unterschiedlichen Lautstärken wahrgenommen (mit Ausnahme der Schrittgeräusche, da sie nur sehr leise und in einem geringen Umkreis wahrgenommen werden). Selbst verursachte Geräusche erhalten stets den Vorzug. Das Attribut enthält die Ursache des Geräuschs und dessen Lautstärke. Andere Geräusche haben in der virtuellen Welt wenig spielerische Bedeutung und werden deshalb vernachlässigt.

Ein letztes Attribut *see_wall* soll sich bei der Imitation der Navigation eines Agenten als nützlich erweisen. Der Grund für dieses Attribut ist die Annahme, dass sich die Richtung, in die sich ein Agent bewegt, immer dann eine Änderung erfährt, wenn der Agent sich einem unüberwindbarem Hindernis nähert. Das Verhalten für eine Richtungsänderung ist also u.a. unmittelbar diesem Attribut zu entnehmen. Als Werte werden ausschließlich „wall“ und „not_visible“ angenommen.

3.1.3. Attribute der Handlung

Die Klasse der Handlung beinhaltet drei Attribute. Um das Verhalten eines Agenten zu imitieren, ist es erforderlich zu wissen, welche Aktionen er in einer bestimmten Situation ausgeführt hat. Dazu gehören laufen, sich drehen, schießen, etwas aufnehmen, Waffe wechseln.

Da in der Beschreibung der virtuellen Welt bestimmte Einstellungen bezüglich des Spielverhaltens vorgenommen worden sind, entfallen einige Aktionen, so z.B. das Wechseln einer Waffe, da dies immer automatisch geschieht, wenn eine Waffe aufgenommen wird. Genauso verhält es sich mit dem Aufnehmen eines Gegenstandes. Als elementare Aktionen bleiben somit „laufen“, „schießen“ oder „stehen“ welche als „forward“, „attack“ und „standing“ im Attribut *action_move* eingetragen werden.

Eine separate Betrachtung findet die Aktion der „Drehung“. Diese wird ersetzt durch ein Attribut *action_direction*, welches die aktuelle Richtung speichert, in die sich der Agent augenblicklich bewegt. Eine Richtung entspricht dabei einer von zwölf Himmelsrichtungen¹. Da eine Drehung eines Agenten im Durchschnitt einer Änderung des Blickwinkels um 15 Grad entspricht, sind hier $\frac{360^{\circ}}{15^{\circ}} = 24$ Richtungen denkbar. Um eine gute Balance zwischen der Anzahl von möglichen Bewegungsrichtungen und nicht zu vielen unterschiedlichen Aktionsmöglichkeiten, bei gleichbleibender Qualität der Imitation, zu erhalten, wurde die Anzahl halbiert. Dies ergibt in der Summe $3 * 12 = 36$ verschiedene Aktionsmöglichkeiten für drei Grundaktionen und zwölf Richtungen. Somit ist gewährleistet, dass sich der Agent bei der späteren Imitation des Verhaltens auch tatsächlich in die richtige Richtung dreht.

¹Dies sind N, NNO, NOO, O, SOO, SSO, S, SSW, SWW, W, NWW und NNW

3. Datengewinnung in Quake3

Eine Drehung lässt sich also ableiten aus zwei hintereinanderfolgenden Aktionen mit Richtungswechsel. Auf ein Attribut zur Speicherung des aktuellen Blickwinkels, kann damit verzichtet werden, weil der Blickwinkel immer der letzten Aktionsrichtung des Agenten entnommen werden kann.

Ein Spezialfall bildet das dritte Attribut *action_hit_from*, da es sich auf eine Handlung bezieht, die von außen auf den Agenten einwirkt. In diesem Attribut wird die Richtung gespeichert, aus der der Agent beschossen wurde. Falls ein Agent beispielsweise aus westlicher Richtung getroffen wird, kann eine mögliche Reaktion sein, sich in diese Richtung zu drehen und zurück zuschießen. Möglich wäre aber auch in eine andere Richtung zu fliehen. In jedem Fall würde als Attributwert „W“ angenommen.

Attribute		
Nr.	Attributname	Beschreibung
1	state_pos_x	Position des Agenten (x-Koordinate)
2	state_pos_y	Position des Agenten (y-Koordinate)
3	state_health	Gesundheit
4	state_armor	Rüstung
5	state_munition	Munition für aktuelle Waffe in Prozent
6	see_wall	Angabe, ob sich der Agent unmittelbar vor einem Hindernis befindet
7	see_dir_health	Richtung eines sichtbaren Health-Item
8	see_dir_weapon	Richtung einer sichtbaren Waffe
9	see_dir_amm0	Richtung einer sichtbaren Munition
10	see_dir_armor	Richtung einer sichtbaren Rüstung
11	watch_item	Bonusgegenstand, der im Visier ist
12	see_dir_bot	Richtung eines sichtbaren Bots
13	see_dir_enemy	Richtung eines sichtbaren Feindes
14	watch_player	Agent, der im Visier ist
15	hear_noise	Geräusche und Lautstärke, die der Agent hört
16	action_hit_from	Richtung aus der der Agent getroffen worden ist
17	action_direction	Aktionsrichtung
18	action_move	Letzte Aktion

Tabelle 3.1.: Nr.1-5: Attribute des Zustands, Nr.6-15: Attribute der Wahrnehmung, Nr.16-18: Attribute der Handlung

3.2. Extrahierung der Daten

Nachdem in Kapitel 3.1 beschrieben wurde, welche Attribute untersucht werden, geht es im Folgenden darum, wie die Daten extrahiert und gespeichert werden. Hierbei sind für die Arbeit nur die Daten relevant, die die Handlungen eines Agenten betreffen und

3. Datengewinnung in Quake3

beeinflussen.

Während des Spielablaufs werden alle Attributwerte eines Spielers mit einer Frequenz von 10 Hz² in eine Log-Datei geschrieben. Zusätzlich geschieht dies, wenn der Agent eine Aktion ausführt (z.B. Schießen) oder wenn von außen eine Aktion auf ihn einwirkt (z.B. wenn der Agent getroffen wird), damit diese Informationen nicht verloren gehen. Dies ist der Fall, wenn eine Aktion zwischen zwei Aufzeichnungsintervallen liegt. Bei Bewegungsänderungen ist eine zusätzliche Ausgabe nicht notwendig, da eine Bewegung im Spiel länger als 1/10 Sekunde andauert. Somit ist sichergestellt, dass der gesamte Bewegungsablauf protokolliert wird. Um die Lesbarkeit zu verbessern, werden nicht nur die Attributwerte gespeichert.

Einige Attribute liegen bereits in Form von Variablen im Programm vor. Für andere Attribute werden zusätzliche Funktionen benötigt. Diese und weitere für die Ausgabe implementierten Funktionen werden im Anhang beschrieben.

Eine Ausgabe ist immer eine Momentaufnahme eines Agenten. Sie beinhaltet seinen aktuellen Zustand, seine letzte Bewegung und die Komponenten, die von ihm wahrgenommen werden. Die Aktionen „attack“ oder der Beschuss durch einen anderen Agenten werden exklusiv ausgegeben in dem Moment, in dem sie geschehen. Ein Auszug einer Log-Datei ist in Abbildung A.1 im Anhang zu sehen.

²die Frequenz entspricht der des Quake3 Bots

4. Datenverarbeitung und Lernen

Nach der Extrahierung der Daten aus Quake3, wie in Kapitel 3 beschrieben, steht zu Beginn dieses Abschnitts eine *Log-Datei* zur Verfügung, welche relevante Informationen über Agentenverhalten und seine Umgebung gespeichert hat. Ziel ist es nun einen neuen Agenten zu erstellen, welcher dieses Verhalten imitiert. Dazu müssen mittels eines geeigneten Verfahrens die vorliegenden Informationen, daraufhin untersucht werden, ob bestimmte Zusammenhänge und Regelmäßigkeiten gefunden werden können. Die zentrale Frage, die sich dabei stellt ist: „Mit welcher Wahrscheinlichkeit führt der Agent in einer bestimmten Situation eine bestimmte Aktion aus?“ Können diese Wahrscheinlichkeiten in den Daten zuverlässig gefunden werden, so kann ein Agent entwickelt werden, der dieses Verhalten nachahmt. Andernfalls lassen sich keinen Zusammenhänge erkennen oder es gibt sie nicht, beispielsweise dann, wenn die aufgezeichneten Daten von Agenten stammen, die sich willkürlich bewegen oder keine bestimmten Verhaltensmuster aufweisen.

Eine Idee ist es den neuen Agenten anhand der gespeicherten Daten Verhalten erlernen zu lassen. Dies läuft nach folgendem Schema ab: Die Datenmenge dient dazu, Zustände, in denen sich ein zu imitierender Agent befunden hat, zu definieren und die Wahrscheinlichkeit zu berechnen, mit der eine bestimmte Aktion in dieser Situation ausgeführt wurde. Bewegt sich ein imitierender Agent jetzt innerhalb der virtuellen Welt, so bewertet er seine Aktion immer dann positiv, wenn er sich in Zuständen befindet, die den bereits gefundenen möglichst ähneln, bzw. bewertet die Aktion negativ, wenn die Zustände sich weniger gleichen. Auf diese Weise soll der neue Agent, Zustände bevorzugen, die auch von einem zu kopierenden Agenten erreicht wurden.

Für die Güte dieses Ansatzes ist die Definition der Zustände von entscheidender Bedeutung. Bisher wurde lediglich eine große Datenmenge generiert, in der jede einzelne Instanz eine Momentaufnahme des Agenten darstellt. Wenig sinnvoll ist es jede einzelne Instanz als eigenständigen Zustand zu betrachten. Erstens ergäbe dies eine unüberschaubar große Menge von Zuständen, zweitens wären viele dieser Zustände fast identisch, so dass sie zusammengefasst werden könnten. Um dieses Problem zu lösen, werden Zustände definiert, die jeweils eine bestimmte Anzahl der Instanzen repräsentieren. Ein System, in dem diese Zustände per Hand festgelegt werden würden, wäre in einer großen Welt sehr aufwendig und zu unflexibel, wenn es in der virtuellen Umgebung nur zu kleinen unvorhersehbaren Änderungen kommen würde. Deshalb wird nach einer Möglichkeit gesucht, Zustände automatisch in der erarbeiteten Datenmenge zu finden.

Für die automatische Definition von Zuständen und Wahrscheinlichkeiten für Aktionen, wird im nächsten Abschnitt das Clustern von Daten beschrieben. Für die Bewertung von Aktionen und das Erlernen von Handlungen, die in Zustände führen, die auch von einem zu imitierenden Agenten erreicht wurden, wird das Reinforcement-Learning vor-

gestellt. Nach der Einführung in die Lernmethodik (Kapitel 4.1) in dieser Arbeit, werden in Kapitel 4.2 zwei Module beschrieben, die dazu implementiert wurden, die Aufgaben des Clusters und des Reinforcement-Learnings zu übernehmen.

4.1. Methodik

Ausgehend von einer allgemeinen Betrachtung des Clusters und des Reinforcement-Learnings werden in diesem Abschnitt Lernverfahren dieser beiden Methoden vorgestellt. Danach werden die Methoden ausgewählt, die in dieser Arbeit verwendet werden.

4.1.1. Clustern

Ursprünglich wurde das Clustern für die Taxonomie in der Biologie entwickelt, um verwandte Arten von Lebewesen zu ordnen (vgl. freie Enzyklopädie [3]). Heute wird es zur automatischen Klassifikation, zur Erkennung von Mustern in der Bildverarbeitung und zum Data Mining¹ eingesetzt. Bei der Clusteranalyse geht es darum, eine gegebene Menge von Instanzen in verschiedene Teilmenge, sogenannte Cluster, aufzuteilen.

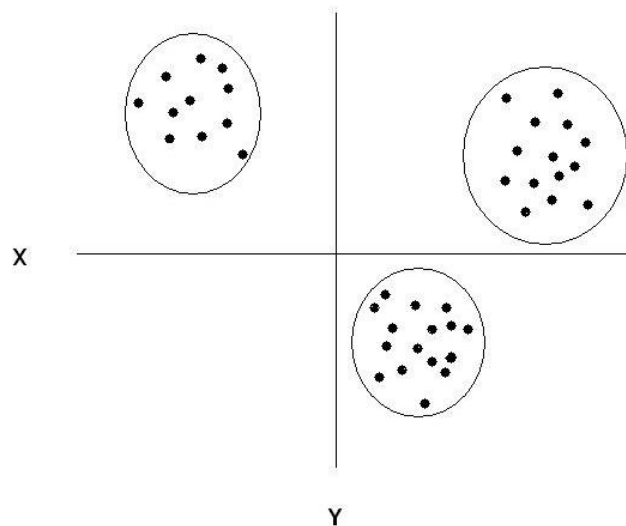


Abbildung 4.1.: Drei Cluster mit Instanzen, symbolisiert durch Punkte.

Es wird daher auch von einem instanzbasierten Verfahren gesprochen. Dabei werden Instanzen so eingeteilt, dass die Objekte innerhalb eines Clusters möglichst ähnlich sind,

¹„Unter Data Mining versteht man das systematische(in der Regel automatisierte oder halbautomatische) Entdecken und Extrahieren unbekannter Informationen aus großen Mengen von Daten.“ (vgl. [3])

und dass Objekte verschiedener Cluster einander möglichst unähnlich sind. Ein Cluster ist also eine Anhäufung von Punkten mit geringerem Abstand zu Punkten des gleichen Clusters als zu Nachbarn anderer Cluster bzw. eine Gruppen von Punkten, die untereinander oder in Bezug auf einen berechneten Schwerpunkt eine minimale Abstandssumme haben (siehe [5]). Die Abbildung 4.1 zeigt z.B. Daten, die in drei verschiedene Cluster eingeteilt wurden.

Im folgenden wird zunächst erläutert, wie der Abstand zwischen den einzelnen Objekten berechnet werden kann und welche Berechnung in dieser Arbeit verwendet wird. Anschließend werden die Verfahren besprochen mit denen Cluster gebildet werden. Im Kapitel 4.1.1.3 werden mehrere Algorithmen vorgestellt, die auf der Basis der Verfahren, Cluster ermitteln und miteinander verglichen. Zum Abschluss von Kapitel 4.1.1 wird ein Algorithmus für die Clusterberechnung in dieser Arbeit ausgewählt.

4.1.1.1. Distanzberechnung

Für die Distanzberechnung zwischen zwei Objekten können unterschiedliche Distanzmaße definiert werden. Ein Distanzmaß bestimmt dabei die Art und Weise der Berechnung einer Entfernung (vgl. [6],[9] und [17]). Die beiden bekanntesten Distanzmaße sind die *Manhattandistanz* und die *Euklidische Distanz* und haben folgende Form:

$$|a_1^{(1)} - a_1^{(2)}| + |a_2^{(1)} - a_2^{(2)}| + \dots + |a_k^{(1)} - a_k^{(2)}| = \textit{Manhattandistanz}$$

und

$$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \dots + (a_k^{(1)} - a_k^{(2)})^2} = \textit{EuklidischeDistanz}$$

$a_1^{(1)}, a_2^{(1)}, \dots, a_k^{(1)}$ sind dabei die Attributwerte einer Instanz 1 und $a_1^{(2)}, a_2^{(2)}, \dots, a_k^{(2)}$ einer anderen Instanz 2, wobei k die Anzahl der Attribute ist.

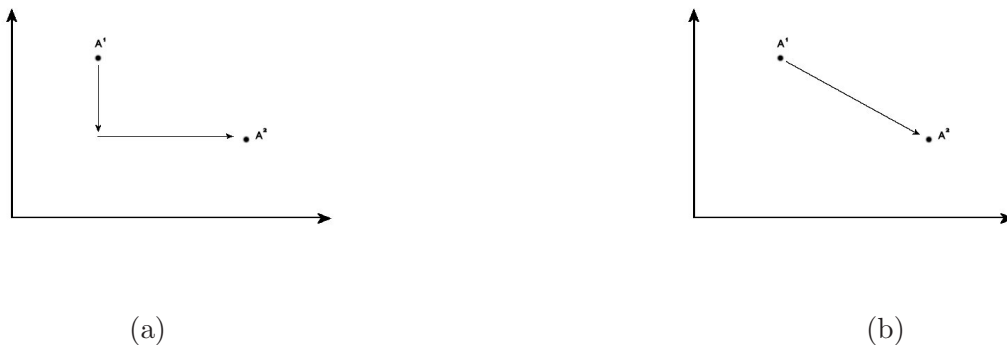


Abbildung 4.2.: Manhattandistanz (a) und Euklidische Distanz (b) zwischen Attributen einer Instanz 1 und 2.

4. Datenverarbeitung und Lernen

Die *Manhattandistanz* verdankt ihren Namen rechteckig angelegten amerikanischen Straßenzügen, in denen die Entfernung von einem Punkt zum anderen nicht durch die in Form der Luftlinie angegebene kürzeste Distanz zwischen zwei Punkten, sondern nur durch Entlangfahren an den Straßenzügen zurückgelegt werden kann. Diese Methode bietet sich an, wenn vor allem die Distanz für unterschiedliche Standorte berechnet werden soll.

Die Berechnung der euklidischen Distanz ist bei instanzbasierten Verfahren die beliebteste Methode um die Entfernung zwischen Objekten berechnen zu können. Sie gibt die kürzeste Distanz zwischen zwei Objekten an. Im Vergleich zur *Manhattandistanz* wird also die Luftlinie zwischen zwei Punkten gemessen.

Weitere Distanzmaße ergeben sich, indem höhere Potenzen als das Quadrat verwendet werden. Dadurch erhöht sich der Einfluss großer Differenzen einzelner Attributwerte gegenüber Differenzen mit einer kleinen Differenz. Die allgemeine Formel lautet (siehe auch [16]):

$$((a_1^{(1)} - a_1^{(2)})^r + (a_2^{(1)} - a_2^{(2)})^r + \dots + (a_k^{(1)} - a_k^{(2)})^r)^{\frac{1}{r}} = \text{Minkowski - Metrik}^2.$$

r ist die sogenannte Minkowski-Konstante, die für $r=1$ die Manhattandistanz und für $r=2$ die Euklidische Distanz ergibt.

Außerdem besteht zusätzlich die Möglichkeit jedem Attributpaar bei der Berechnung der Differenz eine eigene Gewichtung beizumessen. Die allgemeine Form wird folgendermaßen ergänzt:

$$(w_1(a_1^{(1)} - a_1^{(2)})^r + w_2(a_2^{(1)} - a_2^{(2)})^r + \dots + w_k(a_k^{(1)} - a_k^{(2)})^r)^{\frac{1}{r}} \\ = \text{Minkowski mit Gewichtung}$$

w_1 bis w_k sind dabei die Gewichtungen der Differenzen. Hierdurch läßt sich erreichen, dass einzelne Attribute unabhängig von ihrem Wert bei der Distanzberechnung stärker berücksichtigt werden, als andere. Es existieren noch viele weitere Distanz- und Ähnlichkeitsmaße wie: „Das innere Produkt“, Tanimoto, Kosinus, Dice (Quelle [12]). Für die Analyse von Clustern wird aber meist ein Distanzmaß der oben beschriebenen Metrik verwendet.

Damit Attribute, die höhere Werte annehmen können, in ihrer Wirkung auf das Distanzmaß nicht größer sind, als dies bei anderen Attributen der Fall ist, werden üblicherweise alle numerischen Attributwerte normalisiert, so dass sie zwischen 0 und 1 liegen:

$$a_i = \frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)}$$

wobei v_i der tatsächliche Wert des Attributs i ist, und das Minimum und Maximum aller Instanzen der zu analysierenden Datenmenge verwendet werden.

²Minkowski(1864-1909), Mathematiker, Gründer des Gebietes „Geometrie der Zahlen“.

Metrik (aus dem griechischen - Zählung/Messung) wird in der Mathematik für die Definition eines Abstandsmaßes verwendet (vgl [3])

Da diese Formeln davon ausgehen, dass es sich ausschließlich um numerische Attribute handelt, müssen nominale bezüglich des Distanzmaßes umgerechnet werden. Dies geschieht, indem die Differenz zwischen ungleichen Werten als 1 angenommen, während die Differenz bei gleichen Werten 0 beträgt. Eine Normalisierung ist nicht mehr nötig, weil die Werte ausschließlich entweder 0 oder 1 annehmen.

Aus dem letzten Absatz folgt, dass für die Wahl einer der hier vorgestellten Distanzmaße nominale Attribute nicht betrachtet werden müssen. Dies liegt daran, dass die Differenz nominaler Attribute unabhängig von der Berechnung 1 oder 0 beträgt. Da nur zwei numerische Attribute Standortangaben (`pos_x` und `pos_y`) beinhalten, wird das Distanzmaß der Manhattan-Distanz nicht verwendet. Um diese beiden Attribute aber nicht zu benachteiligen, werden höhere Potenzen als zwei ($r > 2$) für die Distanzberechnung ausgeschlossen³. Aus diesem Grund wird aus den beschriebenen Verfahren zur Distanzberechnung das Euklidische Distanzmaß gewählt. Eine Gewichtung der Attribute wird, wie in Kapitel 4.2.1.3 beschrieben, vorgenommen.

4.1.1.2. Clusterverfahren

Der nächste Schritt ist die Auswahl einer geeigneten Methode, um die einzelnen Instanzen den verschiedenen Gruppen (Clustern) zuzuordnen. Dabei unterscheiden sich Clusteranalyseverfahren in den Zuordnungsprinzipien und in der Vorgehensweise. Um ein Objekt einem Cluster zuzuweisen bestehen drei Möglichkeiten (vgl. [14]):

- exakte Zuordnung
Objekte werden mit Wahrscheinlichkeit 1 einem Cluster (nicht-überlappende Zuordnung) oder mehreren Clustern (überlappende Zuordnung) zugeordnet.
- probabilistische Zuordnung
Objekte werden mit einer zwischen 0 und 1 liegenden Wahrscheinlichkeit einem oder mehreren Clustern zugeordnet.
- possibilistische Zuordnung
Objekte werden über eine Zugehörigkeitsfunktion, die Werte zwischen 0 und 1 annehmen kann, jedem Cluster zu einem bestimmten Zugehörigkeitsgrad zugeordnet.

Die Vorgehensweise des Clusters unterscheidet sich in ihrer Methodik. Hier gibt es eine Vielzahl von Ansätzen. In Han und Kamber[8] werden diese Ansätze in 5 Methoden unterteilt.

- Partitionierende Methoden
Ausgangspunkt ist eine zufällige gewählte Anfangspartition von Clustern. Die Anzahl der Cluster wird dabei vorgegeben. In jedem Iterationsschritt werden die Ob-

³Angenommen eine Position des Agenten (x_1/y_1) hat die gleiche kürzeste Distanz zu zwei anderen Punkten (x_2/y_2) und (x_3/y_3). Weiterhin wird angenommen, dass x_3 den gleichen Abstand zu x_1 , wie y_3 zu y_1 hat. Dann sind alle anderen Positionen (x_2/y_2) in der Welt, zwar gleich weit entfernt von (x_1/y_1), werden aber durch eine Distanzberechnung mit höheren Potenzen einer größeren Entfernung zugeordnet.

jekte dem jeweils nächsten Cluster zugeordnet, das durch seinen Mittelwert bzw. ein zentrales Objekt repräsentiert wird.

- **Hierarchische Methoden**
Diese Methode wird durch zwei unterschiedliche Vorgehensweisen charakterisiert. Von agglomerativen Clustern wird gesprochen, wenn zunächst jedes Objekt ein eigenes Cluster darstellt und schrittweise benachbarte Cluster zu einem zusammengefasst werden, bis die Anzahl der gewünschten Cluster erreicht ist oder nur noch ein Cluster übrig bleibt.
Um divisives Clustern handelt es sich, wenn ausgehend von einem Cluster, welches im ersten Schritt alle Objekte umfasst, Cluster solange aufgespalten werden, bis die gewünschte Anzahl von Clustern erreicht wird oder jedes Objekt genau ein Cluster ist.
- **Dichtebasierte Methoden**
Dieses Verfahren betrachtet Cluster als Regionen innerhalb des Instanzraumes, die selbst eine hohe Objekt-Dichte aufweisen und durch Regionen geringerer Instanzdichte voneinander getrennt sind. Ausgehend von einzelnen Objekten verschmelzen diese zu einem Cluster, solange die Dichte von Objekten in ihrer Nachbarschaft einen Schwellenwert überschreiten.
- **Gitterbasierte-Methoden**
Gitterbasierte Methoden teilen den Instanzraum zunächst in ein vorgegebenes Raster und führen die Clusterzuordnung anschließend innerhalb der Zellen dieses Gitters durch. Das Zusammenfügen von Gitterzellen führt danach zum Entstehen von größeren Clustern.
- **Modellbasierte Methoden**
Statistische Verfahren gehören zu der Klasse der modellbasierten Clustermethoden. Dabei wird jedes Cluster durch ein mathematisches Modell beschrieben. Eine Optimierung der Modelle für jedes Cluster erfolgt dadurch, dass die Parameter für jedes Modell an die Clusterobjekte angepasst und Modelle für verschiedene Cluster gegeneinander abgegrenzt werden.

4.1.1.3. Clusteralgorithmen

Im folgenden werden einige Algorithmen betrachtet, die zur Clusteranalyse von Daten entwickelt worden sind. Dabei werden zu den, im vorherigen Abschnitt, erklärten Methoden einige Algorithmen näher untersucht. Dies sind:

- k-means (Partitionierende Methode)
- AGNES (Hierarchische Methode)
- DBSCAN (Dichtebasierte Methode)

4. Datenverarbeitung und Lernen

- EM (Modellbasierte Methode)

k-means

Der populärste Algorithmus, der die *Partitionierungs-Methode* nutzt, ist k-means (vgl. MacQueen67[15]). Das Ergebnis dieses Algorithmus ist eine Zuordnung jeden Objektes zu genau einem Cluster. Diese läuft nach folgendem Schema ab:

1. **Initialisierung:** Zu Beginn werden die Anzahl der erwarteten Cluster k vorgegeben und der Algorithmus startet mit einer zufälligen Auswahl von k Clusterzentren.
2. **Zuordnung:** Im zweiten Schritt wird jedes Objekt, dem ihm am nächsten liegenden Clusterzentrum (means) zugewiesen. Dies geschieht mittels Distanzberechnung zwischen Objekt und Clusterzentrum.
3. **Neuberechnung:** Für jedes Cluster werden die Zentren neu ermittelt, indem der Mittelwert der zugehörigen Objekte berechnet wird.
4. **Wiederholung:** Solange sich die Zuordnung der Objekte zu den Clustern noch ändert oder eine festgelegte Anzahl von Iterationen nicht erreicht ist, wird mit dem zweiten Schritt fortgefahren.

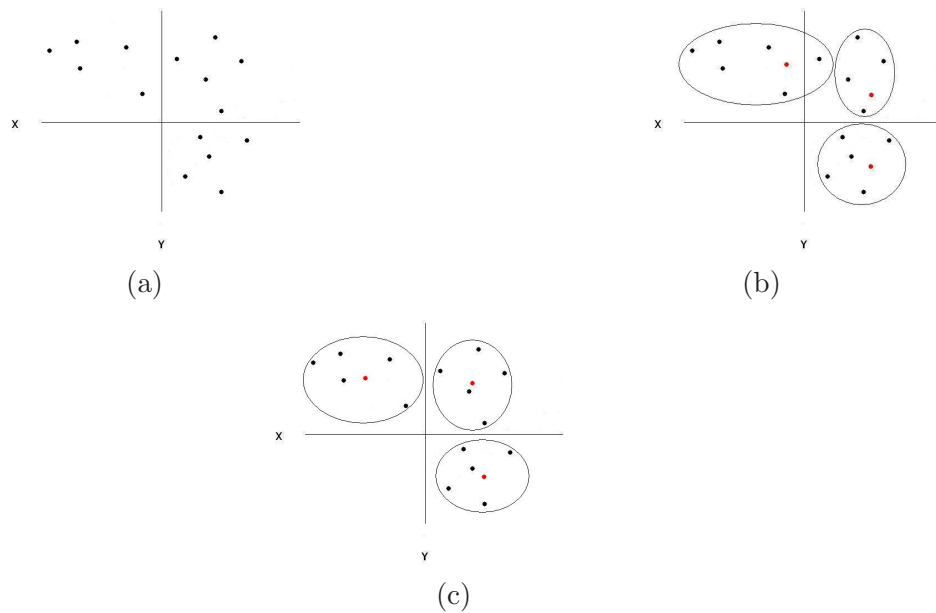


Abbildung 4.3.: Instanzen vor kmeans (a), Initialisierung (b) und nach der Neuberechnung der Clusterzentren. Die Zentren sind dabei andersfarbig markiert. (c)

4. Datenverarbeitung und Lernen

Der Vorteil dieser Methode ist die Laufzeit von $O(n * k * i)$ mit n = Anzahl der Objekte, k = Anzahl der Cluster und i = Anzahl der Iterationen. Da die Anzahl der Cluster und Iterationen meist sehr viel kleiner sind, als die Anzahl der Objekte ergibt sich hier $O(n)$ (vgl. [17] und [18]).

Wegen der guten Laufzeit müssen aber auch einige Nachteile in Kauf genommen werden. So muss bei dieser Clustermethode zum einen die Anzahl der Cluster k vorher bekannt sein, was besonders bei nicht vorhersehbaren Zusammenhängen in einer Datenmenge sehr schwierig ist. Zum anderen ist das Ergebnis dieses Verfahrens stark von der Initialisierung der Clusterzentren abhängig, weshalb es eventuell sinnvoll ist, den Algorithmus mit unterschiedlichen Zufallswerten zu starten und die beste Lösung auszuwählen. Ein dritter Nachteil bezieht sich auf die Neuberechnung der Clusterzentren. Durch die Mittelwertbildung können Zentren entstehen, die mehr oder weniger weit von den eigentlichen Objekten des Clusters liegen.

Es gibt einige mit dem k-means Verfahren verwandte Algorithmen, wie k-medoids, CLARA (Clustering LARge Applications) und CLARANS (Clustering Large Applications based upon RANdomized Search) [vgl. [20]]. Im Unterschied zur k-means Methode werden Cluster hier nicht durch ihren Mittelwert, sondern durch Zentralobjekte repräsentiert. Durch die Berechnung dieser Objekte verschlechtert sich die Laufzeit dieser Algorithmen aber im schlechtesten Fall auf $O(n^2)$.

AGNES

AGNES (AGglomerative NESTing) ist ein Algorithmus, welcher auf der hierarchisch-agglomerativen Methode basiert [19]. Das Ergebnis dieses Verfahrens ist ein Dendrogramm mit folgenden Eigenschaften [10]:

- Jeder Knoten repräsentiert ein Cluster.
- Die Wurzel ist ein Cluster bestehend aus allen Objekten der Datenmenge.
- Die Blätter repräsentieren einzelne Objekte.
- Innere Knoten bilden die Vereinigung aller Objekte aus dem darunterliegenden Teilbaum.
- Ein horizontaler Schnitt durch das Dendrogramm bildet eine Hierarchieebene.
- Auf jeder Hierarchieebene ist jedes Objekt genau einem Cluster zugeordnet.

Folgende Schritte durchläuft der Algorithmus:

1. **Erstellen** einer Distanzmatrix. Hier wird für jedes Objektpaar aus der Datenmenge mittels eines Distanzmaßes, wie in Kapitel 4.1.1.1 gezeigt, die Distanz zwischen diesen beiden Objekten gespeichert.
2. **Start:** Jedes Objekt bildet zu Beginn ein selbständiges Cluster. Die Anzahl der Cluster k wird der Anzahl der Objekte n in der Datenmenge gleichgesetzt.
3. **Suche** das Clusterpaar (c_1, c_2) mit der größten Ähnlichkeit und **verschmelze** das Paar zu einem neuen Cluster $\{c_1, c_2\}$. Reduziere die Clusterzahl k um 1.

4. Datenverarbeitung und Lernen

4. **Beende** den Algorithmus wenn $k=1$ ist, da alle Objekte einem einzigen Cluster angehören.
5. **Berechne** die Distanzen des neu gebildeten Clusters $\{c_1, c_2\}$ zu den verbleibenden Clustern k . Dies geschieht, indem die einzelnen Objekte von zwei Clustern miteinander verglichen werden und die kleinste Distanz, als neue Distanz angenommen wird.
6. **Wiederhole** die Schritte 3 bis 5..

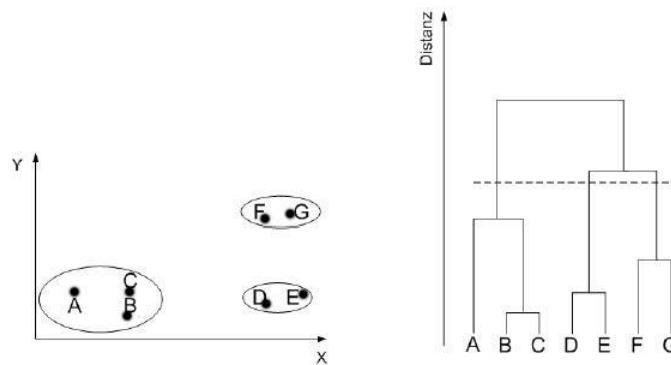


Abbildung 4.4.: Links sind die einzelnen Objekte zu sehen. Rechts ein Dendrogramm, wobei mit steigender Hierarchieebene auch die maximalen Distanzen der Objekte in einem Cluster ansteigen. Die gestrichelte Linie entspricht der Clusterung der Objekte im linken Bild (Quelle [18]).

Die Vorteile dieser Methode sind das Resultat in Form eines Dendogramms. Dies ist graphisch sehr anschaulich und gibt nicht nur ein einziges Clusterergebnis aus, sondern eine Hierarchie von Clustern. Ein weiterer Vorteil ist, dass keine Kenntnisse der Anzahl der Cluster k bestehen muss. Zu den Nachteilen zählen, dass einmal gebildete Cluster nicht mehr aufgelöst werden können, weil jedes Cluster Grundlage für die darüberliegende Hierarchieebene ist. Ein anderer Nachteil ist die Laufzeitkomplexität von mindestens $O(n^2)$ für n Objekte[18]. Allein die Berechnung der Distanzmatrix benötigt $\frac{n(n-1)}{2}$ Zeit. Diane(DIvisive ANALysis) ist ein Verfahren welches die hierarchisch-divisive Methode verwendet(siehe [19]). Für diesen Algorithmus gelten die gleichen Vor- und Nachteile.

DBSCAN

(Density Based Spatial Clustering of Applications with Noise) ist ein dichtebasierter Algorithmus. Das Ergebnis ist eine Zuweisung, die je nach gewählten Parametern des Algorithmus mehr oder weniger Objekte aus der Datenmenge genau einem Cluster zuordnen. Für die beiden benötigten Parameter $MinPts$ und ε werden folgende Begriffe definiert (vgl. [10]):

- Die ε -Umgebung eines Objektes p aus der Datenmenge D ist definiert durch:

$$N_\varepsilon(p) = \{q \in D \mid d(p, q) \leq \varepsilon\}$$

- Ein Punkt p heißt Kernobjekt, wenn gilt:

$$|N_\varepsilon(p)| \geq MinPts$$

- Ein Objekt p ist ein Randobjekt, falls es kein Kernobjekt ist und in der ε -Umgebung eines Kernobjekts liegt.
- Ein Objekt p ist direkt dichte-erreichbar von q bzgl. ε und $MinPts$, wenn gilt:
 - $p \in N_\varepsilon(q)$
 - q ist ein Kernobjekt.
- Ein Objekt p ist dichte-erreichbar von q , wenn es eine Kette von direkt dichte-erreichbaren Objekten zwischen q und p gibt.
- Zwei Objekte p und q sind dichte-verbunden, wenn sie beide von einem dritten Objekt o aus dichte-erreichbar sind.

Der Parameter ε gibt also den maximalen Abstand von Objekten aus der Datenmenge zu einem Objekt p an, bis zu welchem diese Objekte zum Kern von p gezählt werden können. Durch $MinPts$ wird die Anzahl der Objekte festgelegt, die zu einem Kern eines Objektes p mindestens gehören müssen, bis p ein Kernobjekt ist.

Ein Cluster C bezüglich ε und $MinPts$ ist eine nicht leere Teilmenge von D , für die folgende Bedingungen erfüllt sind:

- Maximalität: $\forall p, q \in D$: wenn $p \in C$ und q dichte-erreichbar von p ist, dann ist auch $q \in C$.
- Verbundenheit: $\forall p, q \in C$: p ist dichte-verbunden mit q .

Die Tatsache, dass sich von einem beliebigen Kernobjekt innerhalb eines Clusters alle Objekte des Clusters finden lassen, indem die jeweils dichte-erreichbaren Objekte abge-sucht werden, macht sich DBSCAN zu nutze, wie der Algorithmus zeigt:

1. **Wähle** einen noch nicht bearbeiteten Punkt p .
2. **Ermittle** alle dichte-erreichbaren Punkte von p bezüglich ε und $MinPts$.

4. Datenverarbeitung und Lernen

- Falls p ein Kernobjekt ist, wird dadurch ein Cluster gebildet.
 - Falls p ein Randobjekt ist, sind keine Punkte von p aus dichte-erreichbar. Fahre mit Schritt 1 fort.
3. **Wiederhole** die Schritte 1 bis 3, falls noch nicht alle Objekte in der Datenbank bearbeitet sind.

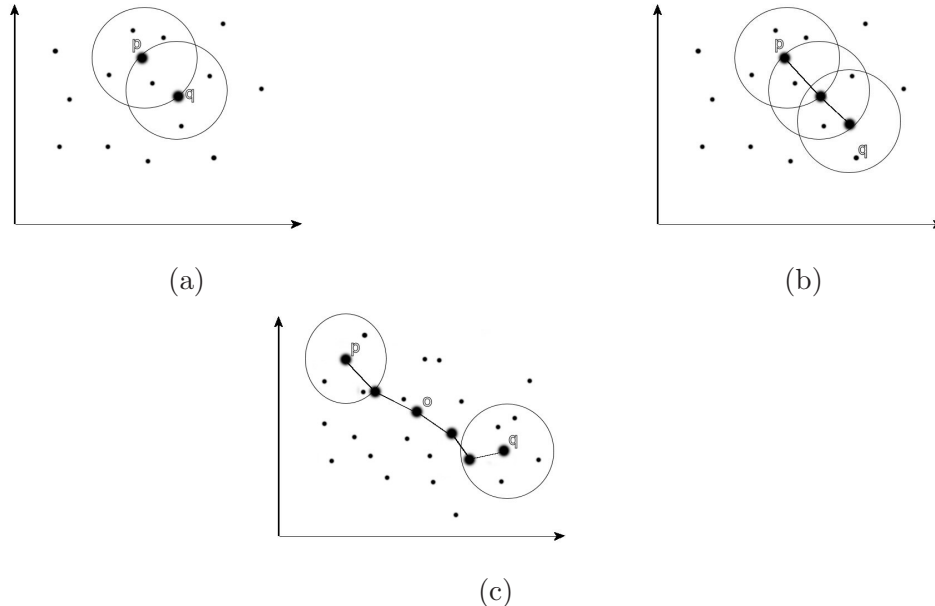


Abbildung 4.5.: Zwei Objekte p und q sind direkt dichte-erreichbar (a), zwei Objekte p und q sind dichte-erreichbar (b) und zwei Objekte p und q sind dichte-verbunden. (c)

Die Vorteile dieses Algorithmus sind, dass die Anzahl der Cluster k nicht vorgegeben werden muss und sogenannte Ausreißer⁴ in den Daten erkannt werden können, die keinem Cluster zugewiesen werden.

Zu den Nachteilen zählen, dass zwei Parameter vorgegeben werden müssen. Außerdem findet der Algorithmus sehr kleine Cluster, d.h. Cluster, deren Objektanzahl kleiner als MinPts ist, nicht.

Die Laufzeit beträgt je nach Präsentation der Datenmenge zwischen $O(n * \log n)$ und $O(n^2)$ (vgl. [18]).

Andere Algorithmen, die zu der Kategorie der dichte-basierten Methoden zählen, sind OPTICS und DENCLUE(siehe auch [21]).

EM

Der EM-Algorithmus (Expected Maximization) gehört zu den modell- oder wahrscheinlichkeitsbasierten Methoden. Die Idee basiert auf dem Clustern nach k-means mit dem

⁴Ausreißer sind Objekte, die in dünn besiedelten Regionen liegen. Beim dichte-basierten Algorithmus sind dies weder Kern- noch Randobjekte.

4. Datenverarbeitung und Lernen

Unterschied, dass keine exakte Zuordnung der Objekte zu den Clustern vorgenommen wird. Das Ergebnis dieses Verfahrens ist eine Zugehörigkeitsfunktion, für die berechnet wird, wie groß die Wahrscheinlichkeit ist, dass ein Objekt o zu einem Cluster c gehört (siehe auch [13]). Das Zuordnungsprinzip folgt daher der in Kapitel 4.1.1.2 beschriebenen possibilistischen Zuordnung. Folgende Voraussetzungen werden für den Algorithmus benötigt:

- Ein Cluster wird durch eine Wahrscheinlichkeitsverteilung beschrieben. Dies ist typischerweise die Gaußverteilung (Normalverteilung).
- Ein Cluster wird folgendermaßen repräsentiert:

- durch den Mittelwert der Objekte eines Clusters $\mu_c = \frac{a^{(1)}+a^{(2)}+\dots+a^{(n)}}{n}$
- durch die Kovarianzmatrix $d \times d \sum_c$ für alle Objekte im Cluster c mit $\sum_c = \frac{(a^{(1)}-\mu)^2+(a^{(2)}-\mu)^2+\dots+(a^{(n)}-\mu)^2}{n-1}$.

- Die Wahrscheinlichkeitsdichte eines Clusters c ist definiert als:

$$P(a|c) = \frac{1}{\sqrt{(2\pi)^d |\sum_c|}} * e^{\frac{1}{2} * (a-\mu_c)^T * (\sum_c)^{-1} * (a-\mu_c)}$$

- Die Wahrscheinlichkeitsdichte eines Clusterings $M = \{c_1 \dots c_k\}$ ist definiert als:

$$P(a) = \sum_{i=1}^k W_i * P(a|c_i)$$

- Ein Punkt wird einem Cluster mit folgender Wahrscheinlichkeit zugeordnet:

$$P(c_i|a) = W_i * \frac{P(a|c_i)}{P(a)}$$

- Die Güte des Clusterings für M wird abgeschätzt durch:

$$E(M) = \sum_{a \in D} \log(P(a))$$

Je größer dieser Wert ist, desto mehr werden die Objekte und Cluster durch das Modell repräsentiert.

Dieser Algorithmus durchläuft zwei Schritte. In einem ersten Schritt wird für jedes Objekt berechnet, mit welcher Wahrscheinlichkeit es zu einem Cluster gehört, im zweiten Schritt wird das Modell, welches die einzelnen Cluster beschreibt, anhand der ermittelten Zuordnungswahrscheinlichkeiten neu bestimmt:

1. **Initialisierung** des Modells für $M = \{c_1 \dots c_k\}$.
2. **Berechnung** der Wahrscheinlichkeitsverteilung im Modell $P(a|c_i)$, $P(a)$ und $P(c_i|a)$ für jedes Objekt aus der Datenmenge und jede Normalverteilung.

4. Datenverarbeitung und Lernen

3. **Neubestimmung** eines Modells M' durch Neuberechnung von W_i, μ_c und \sum_c für jedes i .
4. **Wiederhole** die Schritte 2 und 3, bis die Änderung am Modell einen Schwellenwert ε unterschreitet $E(M') - E(M) \leq \varepsilon$.

Ein Vorteil dieses Algorithmus ergibt sich, wenn keine exakte Zuordnung der Objekte zu den Clustern erwünscht ist.

Da k-means ein Spezialfall des EM-Algorithmus ist, ergeben sich die gleichen Nachteile, wie für k-means. Die Laufzeit hängt stark von der Initialisierung des Modells ab und beträgt $O(n * |M| * i)$ mit n = Anzahl der Objekte, M = Größe des Modells und i = Anzahl der benötigten Iterationen. Die Anzahl der Iterationen, bis der Algorithmus terminiert, ist meist sehr hoch (siehe auch [13]).

Ein anderer modellbasierter Algorithmus ist Cobweb [siehe [23)]. Im Unterschied zum EM-Algorithmus werden hier die Cluster hierarchisch angeordnet.

4.1.1.4. Wahl eines Clusteralgorithmus

Beim Clustern von Daten geht es in dieser Arbeit darum, Instanzen so anzuordnen, dass diese Objekte einen Zustand eines Agenten in der Welt von Quake3 repräsentieren. Dazu werden folgende Anforderungen an den Algorithmus geknüpft:

- Da die Anzahl der Zustände unbekannt ist, sollte die Anzahl der Cluster k nicht vorgegeben werden.
- Die Laufzeit des Algorithmus soll so gering, wie möglich sein. Grund dafür ist, dass in jeder Sekunde mindestens zehn Momentaufnahmen eines Agenten/Instanzen gespeichert werden. Dies führt bei einer längeren Aufzeichnung von Agentenverhalten zu einer großen Datenmenge⁵.
- Es soll das Prinzip der exakten Zuordnung eines Objektes zu einem Cluster verfolgt werden. In Kapitel 4.1.2 wird ein Verfahren des Reinforcement-Learnings ausgewählt, für das es nötig ist, zu wissen in welchem Zustand sich der Agent befindet.

Bei der Betrachtung des Anforderungsprofils an den Clusteralgorithmus für diese Arbeit fällt auf, dass keiner der oben erläuterten Methoden die Ansprüche zu 100% erfüllt. Da es erforderlich ist, Instanzen Zuständen genau zuzuordnen, kann der EM-Algorithmus ausgeschlossen werden. Beim Vergleich der übrigen drei Algorithmen ergibt sich, dass dieses Kriterium von den anderen Algorithmen erfüllt wird. Der Vorteil von k-means liegt in der Laufzeit. Die Algorithmen AGNES und DBSCAN haben ein Plus dadurch, dass die Anzahl der Cluster nicht vorgegeben werden muss. Hier muss abgewogen werden, welcher Vorteil bzw. welcher Nachteil stärker wiegt. Wie in Kapitel 4.2.1.3 zu sehen, lassen sich bestimmte Zusammenhänge in den Daten bereits vor dem Clustervorgang

⁵In Kapitel 5 wird im fünften Szenario 30 min. lang Verhalten aufgezeichnet. Dies ergibt 6808 Instanzen bzw. Objekte, die zu clustern sind.

ermitteln. Diese können dazu genutzt werden, die Anzahl der Cluster abzuschätzen. Da sehr viele Tests nötig sind, um die Lernverfahren des Clusters und Reinforcement-Learnings zu untersuchen, ist es daher erstrebenswerter einen Algorithmus zu benutzen, der möglichst schnell arbeitet. Aus diesen Gründen fiel die Wahl des Algorithmus zum Clustern der Zustände eines Agenten auf k-means.

Das Kapitel 4.2.1 zeigt, wie diese Methode in dieser Arbeit umgesetzt wurde.

4.1.2. Reinforcement-Learning

Reinforcement-Learning ist ein Lernverfahren, welches darauf beruht, dass in einem System selbständig ein Weg zu einem Ziel erlernt wird. Dies geschieht durch „Trial and Error“ Interaktionen mit seiner Umwelt, für die es Belohnungen und Bestrafungen gibt. Ziel ist es diese Belohnungen zu maximieren, um einen optimalen Lösungsweg zu erhalten. Im Gegensatz zu Klassifikationsverfahren werden, dabei keine Beispiele imitiert, sondern Verhalten allein durch Performance-Feedback erlernt. Es wird häufig in Systemen eingesetzt, in denen eine Strategie erlernt werden soll. Beispiele sind ein Roboter, der sich in einer für ihn unbekanntem Umgebung zu einem festgelegten Standort bewegen soll, ohne dabei an Hindernissen anzustoßen oder ein Computerprogramm, welches so gut Backgammon spielt, wie die besten menschlichen Spieler (vgl [25]). Die optimale Strategie in dieser Arbeit, ist eine möglichst exakte Imitation von Agentenverhalten in einer virtuellen Umgebung.

Ein RL-System besteht aus den folgenden Elementen (vgl. [22]):

- **Agent:** in einer Umgebung handelndes Subjekt mit einem definierten Ziel (Roboter, Backgammonprogramm).
- **Zustand s :** aufgrund von Merkmalen der Umgebung definierte Eigenschaft, die dem Agenten zugewiesen wird (Standort des Roboters in einem Raum, Spielbrett)
- **Zielzustand:** Zustand oder Zustände, die vom Agenten angestrebt werden (Umfahren eines Hindernisses, Gewinnen des Spiels)
- **Aktion x :** Handlung, die der Agent ausführen kann (Robotorbewegung, Ziehen von Spielsteinen)
- **Belohnung/Bestrafung $r(s,x)$** Belohnung/Bestrafung (reward), die der Agent bei der Ausführung der Aktion x im Zustand s erhält.
- **Steuerung $(\pi(s,x))$** Wahrscheinlichkeit, dass der Agent in Zustand s Aktion x ausführt
- **Zeit t** Zustände, Aktionen, Belohnungen und die Steuerung werden einem festen Zeitpunkt zugeordnet.

Abbildung 4.6 zeigt wie die einzelnen Komponenten eines Reinforcement-Learning-Systems zusammenarbeiten.

Beim Reinforcement-Learning wird davon ausgegangen, dass es nur endlich viele Zustände

4. Datenverarbeitung und Lernen

gibt, die sich gegenseitig ausschließen und die sogenannte Markov-Eigenschaft erfüllt ist. Diese besagt, dass die Wahrscheinlichkeit eines Folgezustands nur vom aktuellen Zustand abhängt (vgl. [24] und [25]). Ist dieser bekannt, so ist die Zukunft unabhängig von der Vergangenheit. Dies bedeutet, dass der Agent nur seinen gegenwärtigen eindeutigen Zustand und eine Steuerung benötigt, um seine nächste Aktion auszuwählen.

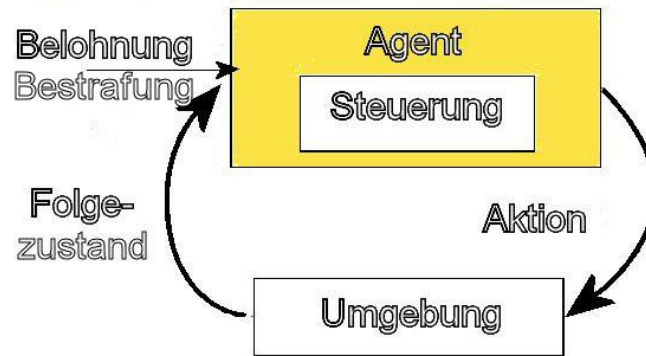


Abbildung 4.6.: Agentensteuerung (Quelle [25])

Ziel des Lernprozesses in RL-Systemen ist es die Steuerung (policy) zu optimieren, indem der Agent versucht die erhaltenen Belohnungen zu maximieren.

$$R = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots + \sum_{t=1}^{\infty} \gamma^t r_t$$

γ ist dabei der sogenannte Discountfaktor, der dafür sorgt, dass Belohnungen in der Zukunft weniger stark gewichtet werden.

Im folgenden werden zwei Verfahren des Reinforcement-Learning vorgestellt. Anschließend wird ausgewählt, mit welcher Methode in dieser Arbeit gelernt werden soll.

4.1.2.1. Reinforcement-Learning-Verfahren

Value-Iteration-Methode

Das erste zu betrachtende R-L-Verfahren geht davon aus, dass der Agent alle Zustandsübergänge und alle Belohnungen kennt (siehe [22]). Bei der Value-Iteration-Methode wird eine Wertefunktion berechnet, die den erwarteten Gewinn widerspiegelt, wenn ein Agent sich im Zustand s befindet und einem festen Kontrollgesetz π folgt.

$$V^{t+1}(s) = \max_x [r(s, x) + \gamma * V^t(\delta(s, x))]$$

4. Datenverarbeitung und Lernen

$\delta(s, a)$ ist dabei der Wert des Nachfolgezustands.

In Abbildung 4.7 ist das Ergebnis einer Wertefunktionsberechnung zu sehen. Jedes Quadrat symbolisiert einen möglichen Zustand des Agenten. Der Zielzustand ist entsprechend gekennzeichnet. Die Kanten zwischen den Zuständen stehen für Aktionen. Die Werte entsprechen den bekannten Belohnungen. Nach fünf Iterationen ist jedem Zustand eine Wertigkeit nach oben beschriebener Formel zugewiesen worden. Es ist zu erkennen, dass umso näher sich ein Zustand am Ziel befindet, umso besser wird ein Zustand bewertet. Ein Agent würde nach dem Lernen mittels Value-Iteration einen Pfad wählen, indem der nächste Zustand besser bewertet wird, als der aktuelle. Dies führt ihn zum Zielzustand.

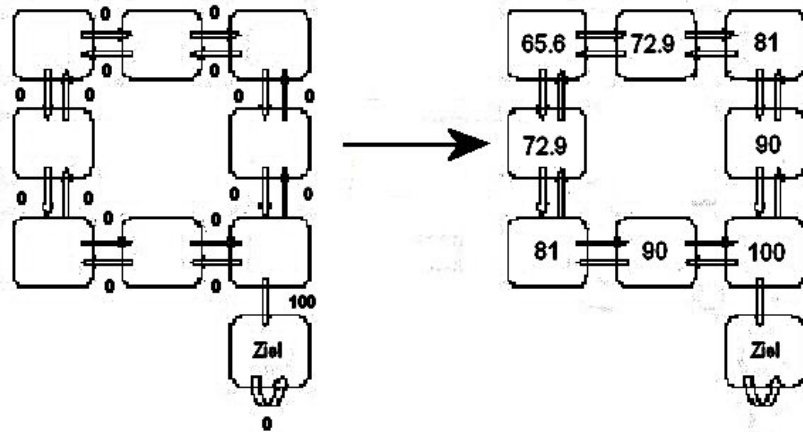


Abbildung 4.7.: Beispiel Value-Iteration mit $\gamma = 0.9$ nach 5 Iterationen (Quelle [22]).

Q-Lernen

Die Methode des Q-Lernens lernt durch eine direkte Bewertung der Aktionen in einem Zustand (siehe [22]). Die Q-Funktion ist definiert als der erwartete Gewinn, den ein Agent erhält, wenn er im Zustand s die Aktion x unternimmt und danach der Steuerung π folgt:

$$Q(s, x) = r(s, x) + \sum_{t=1}^{\infty} r(s_t, x(s_t))$$

Die Steuerung des Agenten besteht aus zwei Teilen: einen Bewerter und einem Selektor. Der Bewerter speichert in einer Matrix für alle Zustände s und alle Aktionen x die Summe

4. Datenverarbeitung und Lernen

der zu erwartenden Reinforcement Signale $Q(s,x)$. Der Selektor ist dafür zuständig mit Hilfe von Informationen des Bewerter und Kenntnis des aktuellen Zustandes die nächste Aktion auszusuchen. Ausgehend von einem perfekten Agenten, dessen Matrix $Q(s,x)$ auch tatsächlich der realen Situation entspricht, so ist die Aufgabe des Selektors einfach: Er fragt in der aktuellen Situation s für alle möglichen Aktionen x die, auf lange Sicht zu erwartende, Belohnung $Q(s,x)$ ab und nimmt die Aktion, die die meiste Belohnung verspricht. In der Lernphase übernimmt der Selektor jedoch noch eine zweite Aufgabe. Er unterstellt der Q Matrix nicht perfekt zu sein. Es erscheint deshalb nicht sinnvoll immer nur die vermeintlich beste Aktion zu wählen, sondern hin und wieder auch eine zufällige. Der Agent probiert also manchmal auch neue Aktionen aus, obwohl er aus seiner bisherigen Erfahrung glaubt, sie seien schlechter als andere. Der Bewerter versucht hingegen aus dem Wissen über die den aktuellen Zustand s , der aktuellen Aktion x und dem dadurch erreichten neuen Zustand s_{t+1} plus Belohnung $r(s,x)$ seine Q Matrix zu perfektionieren. Die oben beschriebene Gleichung wird daher ergänzt.

$$Q(s, x) = (1 - \alpha)Q(s, x) + \alpha(r(s, x) + \sum_{t=1}^{\infty} r(s_t, x(s_t)))$$

Die Lernrate α mit $0 \leq \alpha \leq 1$ symbolisiert, wie sehr der Agent an das bisher gelernte glaubt. Würde α gleich null gesetzt, so würde er nichts lernen, da zu hundert Prozent der alte Q Wert übernommen wird. Ist α gleich eins, so ist nur der rechte Teil der Gleichung aktiv. Er würde alle bisherigen Erfahrungen in dem aktuellen Zustand vergessen und sich nur am aktuellen reward orientieren. Da beide Extremfälle unerwünscht sind, ist eine optimale Strategie am Anfang des Lernprozesses ein relativ hoher Wert, der gegen Ende des Lernvorganges, mit schwindender Änderung der Q -Matrix, immer weiter reduziert wird.

Der Algorithmus läuft nach folgendem Schema ab:

- Initialisiere Tabelle $Q(s,x)$ auf 0.
- Solange sich die Q -Tabelle noch ändert
 - Beobachte Zustand s
 - Wähle Aktion x aus
 - Beobachte $r(s,x)$ (kurzfristige Belohnung und Nachfolgezustand)
 - Aktualisiere $Q(s, x) = r(s, x) + \sum_{t=1}^{\infty} r(s_t, x(s_t))$
 - Gehe in den Folgezustand

Abbildung 4.8 zeigt den Vorgang des Q -Lernens. Im Unterschied zu Abbildung 4.7 ist zu erkennen, dass durch Q -Lernen nicht Zustände bewertet werden, sondern die Aktionen.

4.2. Umsetzung

Um die Imitation von Agentenverhalten durch die in Kapitel 4.1.1.4 und 4.1.2.2 ausgewählten Verfahren zu realisieren, müssen die Ideen dieser Methoden mit dem Ziel dieser Arbeit verknüpft werden. Dazu werden zwei Module erstellt. Ein erstes Modul sorgt offline dafür, aus den in Kapitel 3 gewonnenen Daten Zustände und Agentenaktionen zu ermitteln. Dazu werden die Daten aus einer Log-Datei (siehe Abbildung A.1) mittels des ausgewählten Clusterverfahrens analysiert. Ein zweites Modul ist für das Reinforcement-Learning zuständig. Es dient dazu online, einen Agenten während seiner Aktionen in der virtuellen Welt lernen zu lassen.

4.2.1. Das Clustermodul

In diesem Modul werden offline Zustände erlernt, in denen sich ein zu imitierender Agent bei der Aufzeichnung seiner Daten befunden hat. Außerdem wird berechnet mit welcher Wahrscheinlichkeit dieser Agent welche Aktion ausgeführt hat. Zustände und Aktionen sind dabei genau durch die Merkmale definiert, welche in Kapitel 3 beschrieben wurden. Das Ergebnis ist eine Steuerung für einen Agenten, in der jeder Momentaufnahme ein Zustand zugewiesen und nach den Wahrscheinlichkeiten eine Aktion ausgewählt werden kann.

Zu diesem Zweck werden in einem ersten Schritt die gespeicherten Daten umgeformt, um das Clustern zu erleichtern. Anschließend werden im Hauptteil dieses Moduls die Daten mittels des in Kapitel 4.1.1 beschriebenen k-means-Verfahrens analysiert. Im letzten Schritt werden die daraus gewonnenen Ergebnisse so gespeichert, dass sie als Steuerung eines neuen Agenten Verwendung finden können.

Im folgenden werden in Kapitel ?? einige Vorüberlegungen zur Umsetzung des Clusters gemacht. In den Kapiteln 4.2.1.2 bis 4.2.1.4 werden die oben beschriebenen Schritte, von der Datenstrukturierung über das Clustern, bis hin zur Aufbereitung der daraus resultierenden Ergebnisse für eine Agentensteuerung, erläutert. In Kapitel 4.2.1.5 wird der Ablauf dieses Moduls aufgezeigt.

4.2.1.1. Vorüberlegungen zum Clustern

Wie bei der Wahl eines Clusteralgorithmus in Kapitel 4.1.1.4 diskutiert, hat k-means den Nachteil, dass die Anzahl der Cluster zu Beginn bekannt sein muss. Aus diesem Grund wird in diesem Abschnitt ein Verfahren entwickelt, wie diese Zahl sinnvoll abgeschätzt werden kann.

Zunächst wird aber darauf eingegangen, welche Objekte bzw. Instanzen ein Cluster enthält und wie dadurch ein Agentenzustand in dieser Arbeit definiert ist.

Definition von Agentenzuständen durch Cluster

In Kapitel 4.1.1.1 wurde ein Distanzmaß beschrieben, mit dessen Hilfe es möglich ist, festzustellen, wie ähnlich sich zwei Objekte sind. Ein Objekt ist in dieser Arbeit dabei eine Momentaufnahme/Instanz eines Agenten oder ein Cluster. Die Distanz zwischen Objekten wird dabei ermittelt, indem die einzelnen Attribute aus Kapitel 3.1 mitein-

4. Datenverarbeitung und Lernen

ander verglichen werden. Das Beispiel aus Abbildung 4.9, bei dem drei unterschiedliche

<p>Instanz A <i>pos_x</i>: 850 <i>pos_y</i>: ... <i>see_dir_health</i>: not_visible <i>see_dir_weapon</i>: ... Instanz B <i>pos_x</i>: 730 <i>pos_y</i>: ... <i>see_dir_health</i>: N <i>see_dir_weapon</i>: ... Instanz C <i>pos_x</i>: 210 <i>pos_y</i>: ... <i>see_dir_health</i>: N <i>see_dir_weapon</i>: ... Minimum <i>pos_x</i>: 200 Maximum <i>pos_x</i>: 1000</p> <p>Distanz A ⇒ B $\sqrt{\left(\frac{850-200}{1000-200} - \frac{730-200}{1000-200}\right)^2 + \dots + 1 + \dots} = 1.011$ Distanz A ⇒ C $\sqrt{\left(\frac{850-200}{1000-200} - \frac{210-200}{1000-200}\right)^2 + \dots + 1 + \dots} = 1.28$ Distanz B ⇒ C $\sqrt{\left(\frac{850-200}{730-200} - \frac{210-200}{1000-200}\right)^2 + \dots + 0 + \dots} = 0.65$</p>

Abbildung 4.9.: Beispiel für die Berechnung von Distanzen

Instanzen miteinander verglichen werden, soll die Vorgehensweise dabei noch einmal verdeutlichen: Dabei werden exemplarisch nur die Attribute *pos_x* und *see_dir_health* betrachtet. Genauso kann auch ein Vergleich zwischen einer Instanz und einem Cluster vorgenommen werden.

Das Ergebnis des Clusters sind Zustände, wobei jeder Zustand durch ein Cluster beschrieben wird. Die Frage, die sich stellt, ist, ob ein Zustand eines Agenten immer durch alle Attribute beschrieben werden soll oder nur durch einen Teil.

Der Vorteil ein Cluster immer durch alle Attribute definieren zu lassen, liegt in einer leichteren Umsetzung, da keine Überlegungen bezüglich der Einteilung der Attribute erfolgen muss und Instanzen immer im Gesamten miteinander verglichen werden können. Wie die erste Testphase in Kapitel 5.3 zeigt, ergeben sich bessere Ergebnisse bezüglich der Imitation von Agentenverhalten, wenn ein Zustand nicht durch alle Attribute gleichzeitig definiert wird. Dies ist u.a. mit der Unflexibilität eines Clusters, indem alle Attribute enthalten sind, zu begründen. Das erste Szenario, welches in Kapitel 5.2.2 vorgestellt wird, verlangt z.B. die Imitation eines Verhaltensmusters bei dem ein Agent um eine Statue herumlaufen soll. In diesem Fall ist weder von Interesse, ob dabei andere Agenten gesehen werden oder wieviel Munition dieser Agent noch besitzt, wichtig ist nur an welcher Position er sich momentan befindet. Ein Cluster, indem stets alle Attribute vorhanden wären, hätte also für bestimmte Anforderungen und Situationen zu viele Informationen.

Ein weiterer Kritikpunkt stellt die Anzahl der Cluster dar. Angenommen es sollen s unterschiedliche Zustände gefunden werden, dann sind dafür c unterschiedliche Cluster nötig, wenn stets alle Attribute in jedem Cluster vorhanden sind. Würde die Menge der Attribute aber beispielsweise in zwei Teile $a_{1\dots x}$ und $a_{x+1\dots y}$, die beide einen Teil der Attribute beinhalten, aufgespalten werden, so können damit $a_{1\dots x} * a_{x+1\dots y} = s$ Zustände definiert werden. Die Anzahl der benötigten Cluster hingegen sind nur $a_{1\dots x} + a_{x+1\dots y} = c < a_{1\dots x} * a_{x+1\dots y} = s$.

Ebenfalls nachteilig wirkt sich aus, wenn bei einem Vergleich zweier Instanzen, mehrere Attributtypen miteinander verglichen werden. Aus Abbildung 4.9 geht hervor, dass nominale Attribute stärker ins Gewicht fallen, als numerische, da die Differenz bei no-

4. Datenverarbeitung und Lernen

minimalen Werten bei Ungleichheit 1 beträgt, während sie bei numerischen im Intervall von 0 bis 1 liegen. Dies führt dazu, eine Trennung zwischen nominalen und numerischen Werten innerhalb eines Clusters vorzunehmen, um kein Ungleichgewicht zwischen den Werten zu erzeugen.

Diese Argumente zeigen, dass es sinnvoll ist, Cluster nur durch einen Teil der Attribute zu definieren. Zu diesem Zweck werden Clusterklassen gebildet. Jede dieser Klassen definiert, welche Attribute zu einem Cluster dieser Klasse gehören sollen. Ein Cluster einer Klasse gibt dabei einen Teil eines Agentenzustands wieder, somit ist jede Clusterklasse eine Zustandsklasse. Bei der Aufteilung der Attribute auf die einzelnen Zustandsklassen wird von folgenden Grundsätzen ausgegangen:

1. Ausschluß von Zuständen mit überlagernden Attributen.
2. Trennung von numerischen und nominalen Attributen.
3. Definition von Zuständen, welche durch einen Teil der Attribute ausgedrückt werden können.
4. Die Attribute *action_move* und *action_direction* dienen ausschließlich der Berechnung der Aktionswahrscheinlichkeiten und sind nicht Bestandteil eines Zustands.

Der erste Grundsatz ergibt sich den Argumenten, die zu dem Entschluß geführt haben, Cluster in Klassen einzuteilen. Die Trennung von numerischen und nominalen Attributen folgt aus der Distanzberechnung. Attribute gehören nur zu jeweils einer Klasse, um die Anzahl der Cluster, die für die Beschreibung eines Zustands nötig sind, nicht wieder zu erhöhen. Der letzte Grundsatz für die Einteilung von Attributen ist bedingt durch die Aufgabe des Clusters in dieser Arbeit. Diese ist es Zustände und Aktionen aus einer vorgegebenen Datenmenge zu lernen. Aktionen sind in den beiden Attributen *action_move* und *action_direction* gespeichert und sind nicht Bestandteil des Zustands eines Agenten. Der Einteilung von Attributen in einzelne Klassen wurde folgende Fragestel-

Zustandsklassen			
Klasse 1 (Position)	Klasse 2 (Status)	Klasse 3 (Wahrnehmung)	Klasse 4 (Feind)
pos_x	state_health	see_wall	see_dir_bot
pos_y	state_armor	see_dir_health	see_dir_enemy
	state_munition	see_dir_weapon	watch_player
		see_dir_amm0	hear_noise
		see_dir_armor	action_hit_from
		watch_item	

Tabelle 4.1.: Einteilung der Attribute in Zustandsklassen

lung zu Grunde gelegt: „Wo befindet sich der Agent innerhalb der virtuellen Welt, wie ist sein gegenwärtiger Status, was sieht er und in wie fern hat er Feindkontakt?“ Daraus

ergeben sich vier Klassen. Ein Zustand des Agenten ergibt sich durch die Kombination dieser vier Klassen. Die Tabelle 4.1 zeigt die Einteilung der Attribute in Zustandsklassen.

Abschätzung der zu berechnenden Cluster:

Ein zweite Überlegung, die bezüglich des Clusters angestellt werden muss, ist die sinnvolle Vorgabe der zu suchenden Cluster. Wie in Kapitel 4.1.1.2 beschrieben, muss für das k-means-Verfahren die Anzahl der zu suchenden Cluster vorgegeben werden. Da für jede Zustandsklasse unterschiedliche Cluster ermittelt werden, wird die Clusteranzahl für jede Klasse gesondert berechnet. Damit diese Zahl nicht fest ist, sondern aufgrund der Daten sinnvoll für jede der vier Zustandsklassen abgeschätzt werden kann, wird versucht diese an die jeweiligen Daten anzupassen. Dies geschieht, indem für alle Attribute die Anzahl der unterschiedlichen Werte ermittelt wird und Anschließend festgelegt wird, wieviele unterschiedliche Werte in einem Cluster zusammengefasst werden. Hierzu wird ausgenutzt, dass nach einer Strukturierung der Daten, wie sie in Kapitel 4.2.1.2 vorgenommen wird, einige Informationen über die Attributwerte zur Verfügung stehen. In Kombination mit diesen Informationen können zusätzlich Rückschlüsse auf die Anzahl der benötigten Cluster, aus Erfahrungswerten der eigenen Praxis in Umgang mit Quake3 gewonnen werden. Da die Anzahl der unterschiedlichen Attributwerte für nominale und numerische Attribute stark voneinander abweichen, wird bei der Berechnung der Clusteranzahl eine Trennung zwischen diesen beiden Attributtypen vorgenommen.

Die im folgenden vorgestellten Definitionen dienen dazu, in der Testphase in Kapitel 5 eine Möglichkeit zu schaffen, die benutzten Lernverfahren mit einer unterschiedlichen Anzahl von Clustern zu überprüfen. Um die Anzahl der Cluster nicht zufällig zu bestimmen, werden im nächsten Abschnitt Parameter eingeführt, die im Zusammenspiel mit ausgewerteten Informationen aus der Datenmenge, die Zahl der Cluster abschätzen.

Abschätzung für numerische Attribute:

Für numerische Attribute wird für die Abschätzung der benötigten Anzahl von Clustern der Parameter *distancenum* eingeführt. Dieser findet für die Attribute der ersten beiden Zustandsklassen Anwendung. Da die Berechnung der Distanz für numerische Attribute auch Werte zwischen 0 und 1 erlaubt, wird durch diesen Parameter die Anzahl der Cluster in zweifacher Hinsicht berechnet. Zum einen bestimmt er, für wieviele unterschiedliche Werte eigene Cluster gebildet werden. Zum anderen wird durch den Parameter *distancenum* festgelegt, wie groß die Differenz zweier Attributwerte sein muss, damit sie bezüglich der Ermittlung der Clusteranzahl, als unterschiedliche Werte angesehen werden sollen. Zu diesem Zweck wird die Anzahl der zu berechnenden Cluster in Abhängigkeit des minimalen und maximalen Wertes eines numerischen Attributes ermittelt. Durch den Parameter *distancenum* wird die Spanne zwischen dem minimalen und maximalen Wert in Intervalle eingeteilt. Ein Intervall hat dabei die Größe in Form der maximalen Differenz, für die zwei unterschiedliche Werte noch in einem Cluster vereinigt werden sollen. Die Anzahl der Intervalle gibt die Anzahl der zu berechnenden Cluster für ein Attribut wieder.

Um einer Explosion der Anzahl von Clustern entgegenzuwirken, wird davon ausgegangen, dass die Intervallgrößen für die Attribute mit einer geringen Amplitude der Werte kleiner gewählt werden, als für die Attribute mit einer höheren Differenz zwischen ihrem

4. Datenverarbeitung und Lernen

Minimum und Maximum. Aus diesem Grund werden die numerischen Attribute in drei Kategorien eingeteilt. Attribute der zweiten Zustandsklasse haben meist Werte zwischen 0 und 100. Hier ist die Differenz zwischen Minimum und Maximum am geringsten, weswegen die Abstände zwischen den Clustern am kleinsten gewählt werden können (Fall 1). Über kurze Zeiträume können die beiden Attribute *state_health* und *state_armor* auch Werte bis zu 200 annehmen. Um in diesem Fall die Anzahl der Cluster nicht zu groß werden zu lassen, werden die Abstände verdoppelt (Fall 2). Amplituden größer als 200 sind ausschließlich bei den beiden Positionsattributen anzutreffen. Hier wird die festgelegte Distanz für zwei Attributwerte innerhalb eines Clusters, vervierfacht (Fall 3). Der Grund dafür ist der gleiche, wie für Fall 2.

Für numerische Attributwerte folgt die Wahl von k folgender Formel:

Definition 1 (Abschätzung der Clusteranzahl k für Klassen mit numerischen Attributen)

$$k(x) = \begin{cases} \prod_{i=1}^n \frac{\max_{a_i} - \min_{a_i}}{\text{distanzenum}} & \text{für } \max_{a_i} - \min_{a_i} < 100 \text{ (Fall 1)} \\ \prod_{i=1}^n \frac{\max_{a_i} - \min_{a_i}}{\text{distanzenum} * 2} & \text{für } 200 > \max_{a_i} - \min_{a_i} \geq 100 \text{ (Fall 2)} \\ \prod_{i=1}^n \frac{\max_{a_i} - \min_{a_i}}{\text{distanzenum} * 4} & \text{für } \max_{a_i} - \min_{a_i} \geq 200 \text{ (Fall 3)} \end{cases}$$

mit x = eine der vier Zustandsklassen, n = Anzahl der Attribute der Zustandsklasse, a_i = numerisches Attribut, i = Attributindex, und \max_{a_i} , \min_{a_i} als Maximum, sowie Minimum eines Attributwertes in der Datenmenge.

Anhand eines Beispiels soll der Umgang mit diesem Parameter erläutert werden.

Beispiel 1 Sei $\text{distanzenum} = 25$ gewählt. Angenommen ein nachzuahmender Agent hätte sich innerhalb einer Map der virtuellen Welt auf den Positionen 0 bis 1000 für x und y bewegt. Die Werte der zweiten Zustandsklasse wären zwischen 0 und 100 gewesen. Dann ergeben sich für k folgende Werte:

$$k(1) = \frac{1000}{25 * 4} = 10 * 10 = 100 \text{ Zustände}$$

und

$$k(2) = \frac{100}{25} = 4 * 4 * 4 = 64 \text{ Zustände}$$

Daraus ergibt sich im vorliegenden Fall, dass für jeden $10 * 10$ großen Bereich innerhalb der Welt ein neuer Zustand gebildet werden würde. Dies wären für die Imitation eines Agenten schon zuviele Zustände, da der Agent, wie in Kapitel 2.1.2 beschrieben mit einer Vorwärtsbewegung einen Abschnitt von $20 * 20$ überbrückt und somit in diesem Fall einen Zustand überspringen würde.

Die Anzahl der zu suchenden Cluster für die zweite Zustandsklasse liefert eine Einteilung der Werte in vier Abschnitte. Diese könnte beispielsweise, wie hier aufgezeigt, aussehen:

4. Datenverarbeitung und Lernen

- 100 optimaler Zustand
- 70 - 100 guter Zustand
- 30 - 70 normaler Zustand
- 0 - 30 kritischer Zustand

Mit der Bildung dieser Zustände könnte somit je nach Agentenstatus ein Agent imitiert werden, der abhängig von seiner Munition, seiner Gesundheit oder seinem Schutz ein auffallendes Verhalten erkennen lässt.

Abschätzung für nominale Attribute:

Für die Abschätzung von k für die Anzahl der Cluster in Zustandsklassen mit nominalen Attributen wird ein zweiter Parameter *distancenom* eingeführt. Mit diesem kann ermittelt werden, für wieviele unterschiedliche Attributwerte neue Cluster gebildet werden sollen. Wird *distancenom* = 1 gewählt, so ergibt sich für jeden Attributwert ein eigenes Cluster.

Die Berechnung für k bei nominalen Attributwerten ist:

Definition 2 (*Abschätzung der Clusteranzahl k für Klassen mit nominalen Attributen*)

$$k(x) = \sum_{i=1}^n \frac{m_{a_i}}{\text{distancenom}}$$

mit x = eine der vier Zustandsklassen, n = Anzahl der Attribute der Zustandsklasse, a_i = nominales Attribut, i = Attributindex, und m_{a_i} = Anzahl der unterschiedlichen Werte für dieses Attribut

Im Gegensatz zu Clustern der ersten beiden Zustandsklassen sind in der virtuellen Welt von Quake3 nicht alle Kombinationen von Attributwerten möglich, weshalb die Anzahl der Cluster durch eine Summierung geschätzt wird⁷.

Damit sind die Vorüberlegungen abgeschlossen.

4.2.1.2. Strukturierung der Daten

Wie in Kapitel 3.2 gesehen, ist die Ausgabe in eine Log-Datei so gewählt, dass sie für den Anwender gut lesbar ist, um eventuelle Fehler besser erkennen zu können. Für die automatische Weiterverarbeitung innerhalb eines Programmes, wäre aber eine Form wünschenswert, die erste Informationen aus den Daten bereitstellt und einen separaten Block erzeugt, welcher nur die Daten enthält. Hierzu gibt es den weitverbreiteten ARFF-Standard (Attribute-Relation File Format). ein ARFF-File ist ein Text in ASCII-Format

⁷In Quake3 sind innerhalb einer Map alle Positionen denkbar, also alle Kombinationen von x und y -Positionen. Genauso ist jeder Agentenstatus möglich. In Klassen mit nominalen Attributen gibt es aber viele Kombinationen von Werten, die in der Welt nicht vorkommen. Beispielsweise ist es nicht möglich einen Agenten im Visier zu haben *watch_player* = „somebody“, ihn aber nicht zu sehen *see_dir_bot* = „not_visible“.

und wurde an der Universität von Waikato entwickelt, um Daten zu strukturieren (siehe [7]). Dies geschieht in zwei Teilen. Der erste Abschnitt eines solchen Files ist der Kopf und enthält allgemeine Informationen über die Daten, wie die Attribute, die Attributtypen (nominal, numerisch, etc.), die Anzahl der Attribute und die Anzahl der Instanzen. Der zweite Teil enthält ausschließlich die Attributwerte.

In Abbildung A.3 ist der Kopf einer solchen Datei zu sehen. Der Teil, welcher die Instanzen in Form der Attributwerte enthält, ist in A.2 abgebildet.

4.2.1.3. Clustern der Daten

Nachdem durch die beschriebenen Formeln eine grobe Abschätzung der zu suchenden Cluster erfolgt ist, dient der nächste Schritt dazu die k Cluster für jede Zustandsklasse mit zufälligen Werten zu initialisieren. Danach schließt sich die in Kapitel 4.1.1 erklärte Prozedur des k -means an.

Dabei arbeitet das Clustermodul nach folgendem Schema: Wurden einem Cluster Instanzen zugewiesen, so wird es in die nächste Clusterroutine mit Neuberechnung der Mittelwerte übernommen. Ist das Cluster Instanzlos, so wird ein neues zufälliges Cluster gebildet. Treten keine Änderung mehr auf oder ist eine zuvor festgelegte Anzahl von Iterationen erreicht, so ist der Clustervorgang beendet. Gibt es zu diesem Zeitpunkt noch Cluster ohne Instanzzuweisungen, so werden diese ersatzlos gelöscht. In diesem Fall wird davon ausgegangen, dass die Anzahl der Cluster zu hoch abgeschätzt worden ist⁸.

4.2.1.4. Aufbereiten der Ergebnisse

Das Clustermodul sorgt nicht nur dafür Zustände zu definieren, sondern auch aus den Daten eines zu kopierenden Agenten, Aktionen zu ermitteln. Damit ein imitierender Agent eine Steuerung erhält, die einem Verhalten eines anderen Agenten ähnelt, muss in dieser Steuerung die Wahrscheinlichkeit enthalten sein, mit der ein nachzunehmender Agent eine Aktion ausgeführt hat. Dazu wird für jedes Cluster gespeichert, mit welcher Wahrscheinlichkeit eine Aktion von einem zu imitierenden Agenten ausgeführt worden ist. Dies geschieht, indem die Vorkommen der Attributwerte für die Attribute *action_move* und *action_direction* in den Clustern gezählt und ihre Wahrscheinlichkeiten berechnet werden. Da diese beiden Attribute zu keiner der Clusterklassen gehören, wird bei jeder Instanzzuweisung zu einem Cluster, vermerkt, welche Aktion und welche Aktionsrichtung zu dieser Instanz gehören. Die Wahrscheinlichkeit für eine Aktion ergibt sich anschließend aus der Anzahl mit der eine bestimmte Aktion ausgeführt worden ist, dividiert durch die Anzahl der Instanzen, die dem entsprechenden Cluster zugeordnet worden sind. Für die Wahl der nächsten Aktion ist es in dieser Arbeit erforderlich, dass die Gesamtwahrscheinlichkeit für Aktionen oder Aktionsrichtungen immer 100% beträgt. Abschließend werden in zwei Dateien die Ergebnisse gespeichert. Eine Datei enthält die

⁸Ein Anzeichen für eine zu hohe Abschätzung der Clusteranzahl ist es, wenn das Clustermodul über mehrere Iterationen hinweg, keine neuen Cluster mehr findet. Dies bedeutet, dass mehrere Iterationen lang immer die gleichen Cluster instanzlos bleiben und ihnen neue zufällige Attributwerte zugewiesen werden.

gefundenen Cluster unterteilt in die jeweiligen Zustandsklassen und eine Auflistung der Minimas und Maximas der einzelnen Attributwerte, eine andere Datei soll später die Aktionswahrscheinlichkeiten liefern. Die Abbildungen A.4 und A.5 im Anhang zeigen ein Beispiel für diese Dateien.

4.2.1.5. Das Modul

Für das Clustern wurde ein Programm entwickelt, welches in den Daten nach Zuständen sucht und diese für die spätere Steuerung eines Agenten zusammen mit Wahrscheinlichkeiten für Aktionen bereitstellt. Die einzelnen zu diesem Zweck implementierten Funktionen sind im Anhang beschrieben.

Das Clustermodul arbeitet nach folgendem Schema:

1. Lesen der Daten aus einer Log-Datei.
2. Bilden einer ARFF-Datei.
3. Ermittlung von wichtigen Informationen (Minimas und Maximas der Attribute).
4. Festlegen der Parameter für das Clustern (Anzahl der Iterationen, *distancenum*, *distancenom*).
5. Abschätzung der Anzahl an Clustern für jede Zustandsklasse.
6. Clustern.
7. Berechnung der Aktionswahrscheinlichkeiten für jedes Cluster.
8. Speichern der Cluster und Aktionswahrscheinlichkeiten.

4.2.2. Das Reinforcement-Learning-Modul

In diesem Kapitel geht es darum die Ideen des Reinforcement-Learnings auf das Lernen in dieser Arbeit zu übertragen. In den in Kapitel 4.1.2.1 beschriebenen Verfahren sollte ein Agent einen Weg zu einem Zielzustand lernen. Hier soll dieser Agent ein Quake3-Agent sein, der genau die Zustände kennt, die durch das Clustern definiert worden sind. Dieser Agent soll das Verhalten eines anderen Agenten mittels Reinforcement-Learning nachahmen. Die in Kapitel 4.1.2 beschriebene Steuerung eines Agenten zeichnet sich durch genau diese Zustände und die Wahrscheinlichkeiten, eine der möglichen Aktionen auszuführen, aus. Als Aktionen stehen dem Agenten, die in Kapitel 3.1.3 definierten Handlungsmöglichkeiten, zur Verfügung. Die Aktionswahrscheinlichkeiten werden für jeden Zustand einzeln berechnet. Die Zielzustände sind für ihn, diejenigen Zustände, in denen sich auch der zu imitierende Agent befindet hat.

Um einen Agenten mittels Reinforcement-Learning, wie in Kapitel 4.1.2 in Quake3 lernen zu lassen, muss dieser zu jedem Zeitpunkt wissen in welchem Zustand er sich befindet.

4. Datenverarbeitung und Lernen

Hierzu wird in Kapitel 4.2.2.1 erläutert, wie ein nachahmender Agent seinen aktuellen Zustand berechnet. Damit der Agent einen Weg zu einem Zielzustand finden kann, muss er die Möglichkeit haben, Aktionen auszuführen, die ihn in Folgezustände bringen. In Kapitel 4.2.2.2 wird dazu erklärt, wie ein Verhaltenimitierender Agent eine Aktion auswählt. Das Q-Lernen basiert auf der Belohnung und Bestrafung von Aktionen. Hierzu wird in Kapitel 4.2.2.3 definiert, wann eine Aktion bestraft oder belohnt wird. Zu diesem Zweck werden Zustände in zwei Kategorien aufgeteilt, erwünschte Zustände und unerwünschte Zustände. Hierdurch können Aktionen belohnt werden, die in erwünschte Zustände führen und Aktionen bestraft werden, die in unerwünschte Zustände führen. Da erwünschte Zustände, diejenigen Zustände sind, die durch das Clustermodul (siehe Kapitel 4.2.1) ermittelt worden sind, müssen unerwünschte Zustände während des Lernens in Quake3 definiert werden. Hierzu werden, wie in Kapitel 4.2.2.4, neue Cluster gebildet. Das Kapitel 4.2.2.5 erläutert, wie die Belohnung und Bestrafung von Aktionen umgesetzt werden. Dies geschieht, indem die Wahrscheinlichkeit für eine Aktion, die in einem Zustand belohnt werden soll, erhöht wird. Bei einer Bestrafung wird die Aktionswahrscheinlichkeit entsprechend verringert. Da die Gesamtwahrscheinlichkeit nach jeder Belohnung/Bestrafung wieder 100% ergeben muss, wird in Kapitel 4.2.2.6 erklärt, wie dieser Prozentsatz gehalten wird. Dazu werden bei einer Belohnung einer Aktion mit gleichzeitiger Erhöhung der Wahrscheinlichkeit für die Ausführung dieser Handlung, alle anderen Wahrscheinlichkeiten im betrachteten Zustand so verringert, dass sie zusammen wieder 100% ergeben. Für den Fall der Bestrafung einer Aktion werden, die übrigen Wahrscheinlichkeiten entsprechend erhöht.

Die Arbeitsweise des Lernmoduls in Quake3 wird zum Abschluss dieses Kapitels besprochen.

4.2.2.1. Zustandsberechnung

Eine Momentaufnahme des Agenten wird immer durch die in Kapitel 3.1 ausgewählten Attribute beschrieben. Dazu werden vor jeder Zustandsberechnung die Werte dieser Attribute aktualisiert.

Der momentane Zustand wird getrennt für jede Zustandsklasse (siehe Tabelle 4.1) ermittelt. Hierzu wird für jede Klasse die aktuelle Instanz oder Momentaufnahme mit den jeweiligen Clustern der Klasse verglichen, indem die Distanz zwischen beiden berechnet wird. Bei der Ermittlung dieser Distanz werden nur die Attribute, der jeweiligen Zustandsklasse miteinander verglichen. Das Cluster mit der kleinsten Distanz zur aktuellen Momentaufnahme ist der aktuelle Zustand bezüglich einer Zustandsklasse. Der Gesamtzustand des Agenten ergibt sich aus den so errechneten vier Zuständen der verschiedenen Klassen. Der aktuelle Zustand $State(i)$ wird dabei folgendermaßen berechnet:

Definition 3 (*Berechnung des aktuellen Zustands getrennt für jede Zustandsklasse*)

Sei i eine Zustandsklasse, n die Anzahl der Attribute der Zustandsklasse, a^j ein beliebiges Cluster dieser Klasse mit den Attributwerten $a_1^j, a_2^j \dots a_n^j$, m die Anzahl der Cluster in

4. Datenverarbeitung und Lernen

dieser Klasse und $b_1, b_2 \dots b_n$ die Attributwerte der aktuellen Momentaufnahme, so ist:

$$State(i) = j \text{ mit } distance(b, a^j) \leq distance(b, a^{1\dots m})$$

Die Distanz wird nach dem in Kapitel 4.1.1.1 beschriebenen Verfahren ermittelt. Abbildung 4.10 zeigt einen Zustand.

Aktueller Zustand:
Pos: 160 0.023751 **State:** 2 0.458617 **See:** 1 1.000000 **Enemy:** 2 1.000000

Abbildung 4.10.: eines Zustandes, wobei die einzelnen Zustände der unterschiedlichen Klassen unterstrichen sind. Der Wert dahinter gibt die berechnete Distanz einer aktuellen Momentaufnahme zu diesem Zustand an.

4.2.2.2. Wahl der nächsten Aktion

Da der Gesamtzustand eines lernenden Agenten in dieser Arbeit aus der Kombination von vier Zuständen aus einer der Klassen besteht, ergeben sich vier Wahrscheinlichkeiten für die Aktionen und Aktionsrichtungen des Agenten. Diese müssen zu einer Wahrscheinlichkeit für jede Aktion miteinander verknüpft werden. Dies geschieht bei einer Gleichbehandlung der Zustandsklassen, indem für jede Aktion die vier ermittelten Wahrscheinlichkeiten addiert und durch die Anzahl der Klassen geteilt werden. Das Beispiel aus Abbildung 4.10 wird ergänzt durch die Berechnung für die Wahrscheinlichkeit einer Aktion.

Aktueller Zustand mit Wahrscheinlichkeiten für die Aktion „forward“:
Pos: 160 0.023751 $\pi(160, forward) = 0.5$ **State:** 2 0.458617 $\pi(2, forward) = 0.1$
See: 1 1.000000 $\pi(1, forward) = 0.2$ **Enemy:** 2 1.000000 $\pi(2, forward) = 0.4$

$$\pi(forward) = \frac{1.2}{4} = 0.3$$

Abbildung 4.11.: Berechnung der Wahrscheinlichkeit für die Aktion „forward“, bei Gleichbehandlung der vier Zustandsklassen. $\pi(j, x)$ steht dabei für die Wahrscheinlichkeit, dass in Zustand j einer Klasse, Aktion x ausgeführt wird

Die Zustandsklassen wurden in Kapitel 4.2.1.1 u.a. dazu eingeführt, für unterschiedliche Imitationsaufgaben, einige Attribute stärker zu berücksichtigen als andere. Um dies in einer Agentensteuerung in Quake3 umzusetzen, wird ein Parameter *weighting* eingeführt. Dieser dient dazu, die einzelnen Zustandsklassen unterschiedlich zu gewichten. Hierdurch soll sich z.B. die Möglichkeit ergeben, einzelne Klassen und deren Attribute ganz auszublenden, was dann sinnvoll ist, wenn davon auszugehen ist, dass Aktionen unabhängig von diesen Attributen ausfallen, beispielsweise das Zurücklegen einer bestimmten Strecke ohne Feindkontakt⁹.

⁹In diesem Fall sind alle Attribute, welche Daten über andere Agenten enthalten unwichtig.

4. Datenverarbeitung und Lernen

Mit Hilfe des Parameters *weighting* kann für jede Zustandsklasse individuell eingestellt werden, zu welchem Prozentsatz die Wahrscheinlichkeiten in die Gesamtwahrscheinlichkeit einfließen. Im Beispiel aus Abbildung 4.11 ist dieser Prozentsatz für jede Klasse 25%. Je höher dieser Prozentsatz für eine Zustandsklasse gewählt wird, umso mehr haben die Attribute dieser Klasse Einfluss auf die Aktionen eines imitierenden Agenten. Die Aktionswahrscheinlichkeiten mit Gewichtung werden nach folgender Formel berechnet:

Definition 4 (*Berechnung der Aktionswahrscheinlichkeiten mit Gewichtung*)

Sei x eine der möglichen Aktionen oder Richtungen, i eine Zustandsklasse, j_i der ermittelte Zustand der Klasse i , n die Anzahl der Zustandsklassen und $weighting(i)$ die Gewichtung der Aktionswahrscheinlichkeiten für die Klasse i , dann gilt für die Wahrscheinlichkeit $\pi(x)$, dass im aktuellen Gesamtzustand diese Aktion oder Richtungsänderung ausgeführt wird:

$$\pi(x) = \sum_{i=1}^n \pi(j_i, x_i) * weighting(i)$$

$$\text{mit } \sum_{i=1}^n weighting(i) = 1$$

Der Parameter *weighting* muss immer so gewählt werden, dass sich eine Gesamtwahrscheinlichkeit für alle Aktionen oder Richtungen von 100% ergibt.

Durch die Markov-Eigenschaft des Reinforcement-Learnings wird die nächste Aktion nur in Abhängigkeit des aktuellen Zustands ausgewählt. Da davon auszugehen ist, dass diese Eigenschaft für den Avatar nicht gegeben ist¹⁰, wird ein zusätzlicher Parameter *directionsfaithfulness* definiert, der eine Verbindung zwischen dem momentanen Verhalten und zurückliegenden Verhalten herstellt.

Diesem Parameter geht folgende Überlegung voraus. Wenn ein zu imitierender Agent seine Aktionen an einem bestimmten Ziel ausrichtet, so wird er einen bestimmten Weg innerhalb der Welt durchlaufen. Hierbei ist anzunehmen, dass dieser Agent bei einer Bewegung von einem Punkt A zu einem Punkt B in der Welt, den direkten Weg nimmt. Dies bedeutet, dass er auf dem Weg zwischen diesen beiden Punkten seine Aktionsrichtung selten wechselt.

Die Qualität der Verhaltensimitation soll durch diesen Parameter so verbessert werden, dass ein nachahmender Agent seine Aktionsrichtung genauso selten wechselt, wie der zu imitierende Agent. Dies wird dadurch erreicht, dass im Clustermodul unter Punkt 3, Ermittlung von wichtigen Informationen, die prozentuale Wahrscheinlichkeit für die Einhaltung der letzten Aktionsrichtung eines zu kopierenden Agenten ermittelt und gespeichert wird. Anschließend werden die Aktionswahrscheinlichkeiten, aus oben stehender Definition, durch den Parameter *directionsfaithfulness*, so angepasst, dass der imitierende Agent seine Richtung genauso häufig beibehält, wie dies der Agent vor ihm getan hat. Dies geschieht dadurch, dass die Wahrscheinlichkeit für diejenige Aktionsrichtung

¹⁰Der Avatar agiert u.a. aus seiner Spielerfahrung und aus Zielen, die er sich während seiner Handlungen in der virtuellen Welt setzt. Diese Ziele können in Zuständen gesetzt worden sein, die weit vor dem aktuellen Zustand liegen.

4. Datenverarbeitung und Lernen

um den Faktor *directionfaithfulness* erhöht wird, die der Richtung der letzten Aktion des Agenten entspricht.

Damit wird einerseits eine Aktionsrichtung des Agenten nicht nur von seinem aktuellen Zustand, sondern auch von seiner letzten Aktionsrichtung beeinflusst. Andererseits wird verhindert, dass der Agent sich bei jeder Aktion in eine andere Richtung dreht, wenn dies der zu imitierende Agent nicht getan hat.

Definition 5 (*Berechnung der Aktionswahrscheinlichkeit unter Einbezug der letzten Aktionsrichtung*)

Sei x die Richtung der letzten Aktion, i die Zustandsklasse j_i der ermittelte Zustand der Klasse i , n die Anzahl der Zustandsklassen und $weighting(i)$ die Gewichtung der Aktionswahrscheinlichkeiten für die Klasse i , dann gilt für die Wahrscheinlichkeit $\pi(x)$, dass im aktuellen Zustand diese Richtung ausgeführt wird:

$$\pi(x) = \sum_{i=1}^n \pi(i_j, x_i) * weighting(i) * (1 + directionfaithfulness)$$

Da die Gesamtwahrscheinlichkeit wieder auf 100% korrigiert werden muss, werden für die restlichen elf Aktionsrichtungen die Wahrscheinlichkeiten insgesamt um den Faktor *directionfaithfulness* verringert.

Die nächste Aktion des Agenten wird im Anschluss an diese Berechnungen über die Wahrscheinlichkeitsverteilung ermittelt.

4.2.2.3. Lernen

Lernen heisst im Zusammenhang mit der Imitation von Verhaltensmustern, zu lernen, das Verhalten eines anderen Agenten zu kopieren. Dabei wird angenommen, dass, umso mehr sich das Verhalten zweier Agenten gleicht, umso identischer sind die Zustände, in denen sich beide aufhalten. Es wird daher davon ausgegangen, dass die Imitation von Verhaltensmustern gelungen ist, wenn die Zustände, in denen sich ein vorgebender und nachahmender Agent befinden, gleich sind.

Um das Verhalten eines Agenten zu imitieren, wurde offline bereits eine Agentensteuerung in Form von Zuständen und Aktionswahrscheinlichkeiten eines nachzuahmenden Agenten erlernt. Es geht nun online darum, die Steuerung eines Agenten auf der Grundlage des offline erlernten, zu verbessern. Zu diesem Zweck soll, während der Interaktion in der Welt von Quake3, nach jeder Momentaufnahme eines imitierenden Agenten überprüft werden, wie ähnlich sich der aktuelle Zustand dieses Agenten und Zustände aus dem Clustern sind. Dies geschieht durch die Berechnung der Distanzen, wie in Kapitel 4.2.2.1. Aufgrund oben getroffener Annahme, bedeutet eine Verbesserung der Steuerung, die Minimierung dieser Distanzen für jede Zustandsklasse.

Um die Distanzen zwischen einer Momentaufnahme eines lernenden Agenten und Zuständen aus dem Clustern, möglichst gering zu halten, verfolgt das Lernverfahren in dieser Arbeit folgender Idee: Bewirkt eine Aktion des Agenten, dass die Distanz zu einem, wie in Kapitel 4.2.2.1 berechneten, Zustand genügend klein ist, so wird die Aktion „belohnt“.

4. Datenverarbeitung und Lernen

Umgekehrt wird eine Aktion „bestraft“, wenn die Distanz zu groß ist.

Ist die berechnete Distanz zu groß, so ergeben sich zwei Probleme. Erstens repräsentiert der ermittelte Zustand, die aktuelle Momentaufnahme nicht ausreichend, da die einzelnen Attributwerte zu stark voneinander abweichen. Zweitens bedeutet aus oben getroffener Annahme eine zu große Distanz, dass Verhalten nicht imitiert wird. Für diesen Fall wird ein Zustand, in Form eines neuen Clusters, gebildet, der der aktuellen Momentaufnahme des Agenten entspricht. Wie dies geschieht, wird in Kapitel 4.2.2.4 erklärt.

Ein zweiter Fall, für den davon ausgegangen wird, dass es dem Agenten nicht mehr möglich ist, ein Verhalten zu imitieren, sind „ausweglose“ Momentaufnahmen. Dies sind Situationen, in denen dem Agenten ein Zustand mit Aktionen zugewiesen werden, die es ihm nicht mehr erlauben, in einen anderen Zustand zu wechseln, obwohl der nachzuahmende Agent dies in dieser Situation getan hat. Der Grund dafür ist, dass der Handlungsspielraum in diesem Zustand aufgrund von äusseren Einflüssen der Welt so eingeschränkt ist, dass, egal welche Aktion der Agent ausführt, ein Zustandswechsel unmöglich ist. Auch für diesen Fall wird online ein neues Cluster gebildet, um Aktionen zu erlernen, die einen Zustandswechsel erlauben.

Danach werden Momentaufnahmen in zwei Kategorien eingeteilt. Momentaufnahmen, die denjenigen gleichen, welche auch durch den zu imitierenden Agenten erreicht wurden, werden als erwünschte Zustände betrachtet. Momentaufnahmen, denen ein online definiertes Cluster zugeordnet wird, werden als unerwünschte Zustände betrachtet. Bisher wurden durch das Clustern der Daten nur erwünschte Zustände gefunden. Dies liegt daran, dass diese Zustände, eine Kopie der Zustände und Aktionen eines nachzuahmenden Agenten widerspiegeln. Unerwünschte Zustände werden nur durch die beiden oben beschriebenen Fälle online gebildet. Diese Zustände dienen dazu die Steuerung eines imitierenden Agenten so zu verbessern, dass er möglichst direkte Wege findet, um von einem erwünschten Zustand zum nächsten zu gelangen.

Sinn und Zweck des Lernens ist es nun, die Steuerung eines imitierenden Agenten so zu optimieren, dass durch seine Handlung Zustände erreicht werden, die möglichst identisch mit den Zuständen des nachzuahmenden Agenten sind. Dies soll zum einen dadurch erreicht werden, dass Aktionen gelernt werden, die aus unerwünschten Zuständen in erwünschte Zustände führen. Zum anderen soll dies dadurch gelingen, dass Aktionen bevorzugt werden, die nah an diese erwünschten Zustände heranzuführen. Um unerwünschte Zustände zu vermeiden, soll zusätzlich durch Belohnen und Bestrafen von Aktionen verhindert werden, dass der Agent sich über eine festgelegte Distanz von einem erwünschten Zustand entfernt.

Im folgenden wird hierzu beschrieben, wie gemessen wird, ab wann eine Momentaufnahme einem erwünschten und ab wann diese einem unerwünschten Zustand zugewiesen wird. Im zweiten Teil dieses Kapitels wird definiert, ab wann eine Momentaufnahme des Agenten nah bzw. fern zu einem erwünschten Zustand ist. Zum Abschluss wird noch einmal zusammengefasst, wann Aktionen belohnt werden und wann sie bestraft werden.

Beurteilung von Zuständen (Unerwünscht/Erwünscht):

Wie oben beschrieben gibt es zwei Möglichkeiten, damit online ein unerwünschter Zustand definiert wird. Der erste Fall ist, dass die aktuelle Momentaufnahme, den bisher

4. Datenverarbeitung und Lernen

ermittelten Zuständen zu unähnlich ist, der zweite Fall ist, dass sich der Agent in einem sogenannten „ausweglosen“ Zustand befindet. Für beide Fälle wird hier ein Maß vorgestellt, mit dem ermittelt werden kann, ob einer dieser Fälle eingetreten ist.

Für den ersten Fall wird davon ausgegangen, dass bei der Berechnung der Cluster alle Instanzen ausreichend ähnlich zu ihrem zugewiesenen Cluster sind. Dies bedeutet es gibt für jede Zustandsklasse mindestens eine Instanz, die ihrem Cluster ähnlich ist, aber weiter entfernt vom Mittelwert dieses Clusters ist, als alle anderen Instanzen dieser Klasse von ihrem zugewiesenen Mittelwert. Diese Entfernung wird als die maximale Distanz einer Instanz zu ihrem Cluster in einer Zustandsklasse betrachtet.

Der Wert *maxdistance*, der während des Clusters für alle Zustandsklassen berechnet wird, ist daher definiert als:

Definition 6 (*Maximale Distanz einer Instanz zu seinem Cluster einer Zustandsklasse*)

Sei i eine Zustandsklasse, a^j und a^k beliebige Cluster dieser Klasse, b eine Instanz, die Cluster a^j zugeordnet wurde, C die Menge aller Instanzen eines Clusters a^k und m die Anzahl der Cluster einer Zustandsklasse.

$$\text{maxdistance}(i) = \text{distance}(b, a^j) \text{ mit } \text{distance}(b, a^j) >= \text{distance}(C, a^k)$$

für alle $1 \leq k \leq m$)

Aus der Annahme der Ähnlichkeit aller Instanzen zu ihrem Cluster, kann mit Hilfe dieses Wertes folgende Aussage getroffen werden. Alle Momentaufnahmen, deren Distanz zum ermittelten Zustand kleiner als die berechnete maximale Distanz ist, sind ihrem Zustand ausreichend ähnlich, so dass kein neuer Zustand gebildet wird. Um festlegen zu können, um wieviel diese maximale Distanz überschritten werden muss, damit die Bildung eines neuen Zustands eingeleitet wird, wird ein Parameter *divergencefactor* definiert.

Satz 1 (*Maß für die Bildung eines neuen Zustands bei zu großer Distanz*)

Sei i wieder eine Zustandsklasse, $\text{distance}(b, a^j)$ die in 4.1.1.1 ermittelte kleinste Distanz eines Clusters dieser Klasse zur aktuellen Momentaufnahme und *maxdistance* die größte maximale Distanz einer Instanz zu ihrem Cluster, so wird ein neues Cluster, bzw. ein neuer Zustand gebildet, wenn gilt:

$$\text{distance}(b, a^j) > \text{maxdistance}(i) * \text{divergencefactor}$$

Mit Hilfe des *divergencefactors* wird das Vielfache der Überschreitung der maximalen Distanz angegeben, die nötig ist, damit ein neuer Zustand definiert wird. Dieser Wert muss größer als 1 gewählt werden, da er sonst der oben getroffenen Annahme (Ähnlichkeiten der Instanzen zum zugeordneten Cluster) widerspricht.

Der zweite Fall besagt, dass ein neuer Zustand dann gebildet werden soll, wenn sich der Agent in einem Zustand befindet, aus dem er nicht mehr in einen anderen Zustand wechseln kann. Um zu überprüfen, ob der Agent sich in einer solchen Situation befindet wird von folgendem ausgegangen.

4. Datenverarbeitung und Lernen

Satz 2 (*Beurteilung von „ausweglosen“ Momentaufnahmen*)

Ein Agent befindet sich in einem Zustand, aus dem er nicht wechseln kann, wenn seine letzten 20 Aktionen identisch waren und sich an seinem Gesamtzustand keine Veränderung ergeben hat.

Mit der folgenden Aussage wird festgelegt, wann ein neuer Zustand definiert wird.

Satz 3 (*Bildung eines neuen Zustands für „ausweglose“ Momentaufnahmen*)

Es wird ein neuer Zustand gebildet, wenn der Agent sich in einer ausweglosen Situation befindet.

Jetzt ist es möglich zu beurteilen, ob sich ein Agent in einem erwünschten oder unerwünschten Zustand befindet.

Satz 4 (*Beurteilung eines Zustands (erwünscht/unerwünscht)*)

Ein Agent befindet sich in einem unerwünschten Zustand, wenn ihm durch die Distanzberechnung, entweder ein unerwünschter Zustand zugewiesen wurde oder durch die beiden beschriebenen Fälle in der aktuellen Momentaufnahme ein neuer unerwünschter Zustand gebildet wird.

Um beurteilen zu können, ob ein Zustand erwünscht oder unerwünscht ist, wird für jedes Cluster dieser Status gespeichert. Alle Zustände, die durch das Clustern ermittelt worden sind, werden demnach mit dem Status „Erwünscht“ alle Online definierten Zustände mit dem Status „Unerwünscht“ ausgezeichnet.

Beurteilung von erwünschten Zuständen (Nah/Fern):

In der Einleitung zu diesem Kapitel 4.2.2.3 wurde festgelegt, dass Aktionen bevorzugt werden sollen, die nah an einen erwünschten Zustand heranzuführen und Aktionen vernachlässigt werden, die sich über eine bestimmte Distanz entfernen. Um eine Aussage, darüber treffen zu können, ob ein Agent sich nah oder weniger nah zu einem erwünschten Zustand befindet, müssen diese beiden Aussagen definiert werden. Zu diesem Zweck wird ein Wert gesucht, der diese beiden Annahmen voneinander trennt. Wie weiter oben definiert, lässt sich für jede Zustandsklasse die maximale Distanz einer Instanz zu ihrem Cluster berechnen. Genauso kann auch die durchschnittliche Distanz aller Instanzen einer Zustandsklasse ermittelt werden:

Definition 7 (*Durchschnittliche Distanz aller Instanzen zu ihren Clustern einer Zustandsklasse*)

Sei i eine Zustandsklasse, a^k ein beliebiges Cluster dieser Klasse, C die Menge aller Instanzen eines Clusters a^k , n die Anzahl der Instanzen aus den Daten und m die Anzahl der Cluster einer Zustandsklasse.

$$averagedistance(i) = \frac{\sum_{k=1}^m distance(C, a^k)}{n * m}$$

4. Datenverarbeitung und Lernen

Durch diese Berechnung lässt sich beurteilen, wie nah sich ein zu imitierender Agent, bei der Aufzeichnung der Daten in den geclusterten Zuständen im Durchschnitt aufgehalten hat. Da sich die hier berechnete durchschnittliche Distanz ebenso auf erwünschte Zustände bezieht, kann sie für die Trennung von nahen und fernen Zuständen benutzt werden. Der folgende Satz geht davon aus, dass sich ein Agent in der Nähe eines erwünschten Zustands befindet, wenn die berechnete Distanz zu diesem Zustand kleiner ist, als die durchschnittliche Distanz, wie oben berechnet. Ist diese Distanz größer, als die durchschnittliche Distanz, so ist der Zustand fern.

Satz 5 (*Beurteilung von erwünschten Zuständen (Nah/Fern)*)

Sei i wieder eine Zustandsklasse, $distance(b, a^j)$ die in 4.1.1.1 ermittelte kleinste Distanz eines Clusters dieser Klasse zur aktuellen Momentaufnahme und $averagedistance$ die durchschnittliche Distanz aller Instanzen zu ihrem Clustermittelpunkt einer Zustandsklasse, so gilt:

Ein erwünschter Zustand heisst nah wenn, $distance(b, a^j) < averagedistance(i)$

Ein erwünschter Zustand heisst fern wenn, $distance(b, a^j) > averagedistance(i)$

Belohnung und Bestrafung:

Reinforcement-Learning lernt nach dem Prinzip gewinnbringende Aktionen zu belohnen und entgegengesetzte Aktionen zu bestrafen. Eine gewinnbringende Aktion soll in dieser Arbeit eine Handlung sein, die den nachahmenden Agenten nah an einen erwünschten Zustand heranbringt. Umgekehrt soll eine Aktion bestraft werden, die entweder in einen unerwünschten Zustand führt oder eine höhere Entfernung zu einem erwünschten Zustand bewirkt. Dabei ist davon auszugehen, dass Verhalten durch den Übergang in einen unerwünschten Zustand weniger gut imitiert wird, als wenn der Agent sich *nur* von einem erwünschten Zustand entfernt.

Um zu überprüfen, ob eine Aktion belohnt oder bestraft wird der aktuelle Zustand mit dem letzten Zustand verglichen. Hierzu wird zum einen der Status(erwünscht/unerwünscht) dieser beiden Zustände abgefragt, zum anderen werden die berechneten Distanzen miteinander verglichen.

Folgende Kriterien lösen dabei eine Belohnung bzw. Bestrafung einer Aktion aus:

Satz 6 (*Belohnung und Bestrafung*)

Befindet sich ein Agent zum Zeitpunkt t in einem erwünschten Zustand und wechselt zum Zeitpunkt $t+1$ in einen unerwünschten Zustand, so wird seine letzte Aktion bestraft.

Befindet sich ein Agent zum Zeitpunkt t in einem unerwünschten Zustand und wechselt zum Zeitpunkt $t+1$ in einen erwünschten Zustand, so wird seine letzte Aktion belohnt.

Bewirkt die letzte Aktion eines Agenten, dass er sich zu weit von einem erwünschten Zustands befindet, so wird seine letzte Aktion bestraft.

Bewirkt die letzte Aktion eines Agenten, dass er sich in der Nähe eines erwünschten Zustands befindet, so wird seine letzte Aktion belohnt.

Belohnungen oder Bestrafungen werden, wie beim Q-Lernen, in der Form verteilt, dass Wahrscheinlichkeiten für Aktionen, die zu der Erfüllung dieser Kriterien geführt haben, erhöht oder verringert werden. Mit dieser Anpassung von Aktionswahrscheinlichkeiten soll erreicht werden, dass ein Agent sich möglichst nah an den durch das Clustern definierten Zuständen hält. Die Anpassung dieser Wahrscheinlichkeiten wird in Kapitel 4.2.2.5 beschrieben.

4.2.2.4. Bilden eines online definierten Clusters

Das Bilden neuer Cluster hat ihren Grund darin, dass ab einer gewissen Distanz zu einem Cluster, eine Ähnlichkeit zu diesem nicht mehr gegeben ist und somit dieses Cluster den aktuellen Zustand eines Agenten nicht repräsentieren kann. Außerdem werden diese Cluster benötigt, um einen Weg zu erlernen, der aus ausweglosen Situationen führt. Soll ein neues Cluster innerhalb einer Zustandsklasse gebildet werden, so werden die Attributwerte der aktuellen Momentaufnahme diesem Cluster zugeordnet. Die Steuerung des Agenten in diesem neuen Zustand in Bezug auf die Aktionswahrscheinlichkeiten wird so ergänzt, dass für die Wahrscheinlichkeiten des neuen Clusters zum einen Durchschnittswerte angenommen werden, zum anderen die Wahrscheinlichkeiten des nächstgelegenen Zustands. Damit soll erreicht werden, dass sich das Verhalten in diesem neuen Zustand, sowohl am Verhalten in einem ähnlichen Zustand, als auch am Gesamtverhalten orientiert.

Für die Berechnung der Steuerung für diesen neuen Zustand, ist es zunächst erforderlich die Durchschnittswerte für Aktionswahrscheinlichkeiten zu ermitteln. Dies geschieht unter Punkt 3: *Ermittlung von wichtigen Informationen* im Clustermodul. Die durchschnittlichen Aktionswahrscheinlichkeiten beziehen sich dabei auf alle Instanzen aus der Datenmenge.

Definition 8 (*Berechnung der durchschnittlichen Aktionswahrscheinlichkeiten*)

Sei n die Anzahl der Instanzen aus der Datenmenge und $m(x)$ die Gesamtanzahl einer Aktion in der gleichen Datenmenge so ist:

$$\text{averagemove}(x) = \frac{m(x)}{n} \text{ mit } x \text{ als Bewegung (forward...)}$$

$$\text{averagedirection}(x) = \frac{m(x)}{n} \text{ mit } x \text{ als Richtung (N, NNO, NOO...)}$$

Die Aktionswahrscheinlichkeiten für den neuen Zustand ergeben sich zum gleichen Teil, aus diesen durchschnittlichen Wahrscheinlichkeiten und aus den Wahrscheinlichkeiten des Zustands, der die geringste Distanz zu diesem neuen Zustand aufweist.

Definition 9 (*Berechnung der Aktionswahrscheinlichkeiten für einen online definierten unerwünschten Zustand*)

Sei x eine der möglichen Aktionen oder Richtungen, new das neue Cluster, j das Cluster mit der geringsten Distanz zum neuen Cluster, $\pi(new, x)$ die Wahrscheinlichkeit für

4. Datenverarbeitung und Lernen

Aktion x im neuen Zustand, so gilt:

$$\pi(\text{new}, x) = \begin{cases} \frac{\text{averagemove}(x) + \pi(j, x)}{2} & \text{falls } x \text{ eine Bewegung ist} \\ \frac{\text{averagedirection}(x) + \pi(j, x)}{2} & \text{falls } x \text{ eine Richtung ist} \end{cases}$$

4.2.2.5. Anpassung von Aktionswahrscheinlichkeiten

In Kapitel 4.2.2.3 wurde festgelegt, wann eine Aktion belohnt wird und wann diese bestraft wird. So findet eine Änderung der Steuerung des imitierenden Agenten immer dann statt, wenn eine der Bedingungen aus dem Satz Belohnung und Bestrafung erfüllt ist. Diese Änderungen an der Steuerung sollen bewirken, dass sich ein imitierender Agent möglichst nah an die Zustände hält, die durch das Clustern vorgegeben worden sind.

Um den Grad der Änderung festlegen zu können, wird ein Parameter *modifier* eingeführt. Dieser bestimmt die prozentuale Auf- oder Abwertung einer Aktionswahrscheinlichkeit durch Belohnen oder Bestrafen. Umso höher dieser Parameter gewählt wird, umso größer sind die Modifikationen in den Aktionswahrscheinlichkeiten. Hat er den Wert 0, so werden keine Änderungen durchgeführt. Er fungiert also als eine Art Lernfaktor.

In Kapitel 4.2.2.3 wird angenommen, dass ein Agent, der sich in einem unerwünschten Zustand befindet, ein Verhalten weniger gut imitiert, als ein Agent, der sich in einem erwünschten Zustand befindet. Aus diesem Grund wird eine Möglichkeit eingeführt, Aktionen anders zu belohnen oder zu bestrafen, die aus einem unerwünschten Zustand heraus- oder hineinführen, als Aktionen, die eine Annäherung oder Entfernung von einem erwünschten Zustand bewirken. Zu diesem Zweck wird der *modifier* in zwei Varianten benutzt, *modifier1* und *modifier2*. Durch den Parameter *modifier1* kann dabei bestimmt werden, wie groß die Änderung an der Steuerung eines Agenten sein soll, wenn seine letzte Aktion einen Wechsel zwischen erwünschten und unerwünschten Zuständen gebracht hat. Die Höhe der Modifikation, für den Fall, dass der Agent sich an einen erwünschten Zustand angenähert oder sich von ihm distanziert hat, wird durch den *modifier2* festgelegt.

Im folgenden wird zunächst erklärt, wie diese beiden Parameter eingesetzt werden, um die Agentensteuerung zu ändern. Da diese Änderung an der Steuerung nur eine(die letzte) Aktion betrifft, wird am Ende dieses Kapitels erläutert, wie die Steuerung des Agenten modifiziert werden kann, wenn gleichzeitig auch zurückliegende Aktionen belohnt oder bestraft werden sollen. Hierzu wird ein weiterer Parameter *discountfactor* eingeführt, der die Anzahl der zurückliegenden Aktionen bestimmt.

Änderung der Agentensteuerung für eine(die letzte) Aktion:

Die erste Definition an dieser Stelle legt fest, wie die Steuerung eines imitierenden Agenten angepasst wird, wenn der Status des Zustands sich durch eine Aktion ändert.

Definition 10 (*Anpassung einer Aktionswahrscheinlichkeit bei Statuswechsel des Zustandes*)

Sei t der aktuelle Zeitpunkt, π die Steuerung des Agenten, x_{t-n} die Aktion, für die die Wahrscheinlichkeit verändert werden soll und s_{t-n} der Zustand, indem die Aktion ausgeführt wurde.

- *Fall 1:*

Der Agent befindet sich zum Zeitpunkt $t-n$ in einem erwünschten und wechselt zum Zeitpunkt t in einen unerwünschten Zustand, so wird die Steuerung des Agenten

4. Datenverarbeitung und Lernen

nach folgender Formel geändert:

$$\pi(x_{t-n}, s_{t-n}) = (1 - \text{modifier1}) * \pi(x_{t-n}, s_{t-n})$$

- *Fall 2:*

Der Agent befindet sich zum Zeitpunkt $t-n$ in einem unerwünschten und wechselt zum Zeitpunkt t in einen erwünschten Zustand, so gilt:

$$\pi(x_{t-n}, s_{t-n}) = (1 + \text{modifier1}) * \pi(x_{t-n}, s_{t-n})$$

Als nächstes wird festgelegt, wie mittels des Parameters *modifier2* Aktionswahrscheinlichkeiten modifiziert werden. Wie oben beschrieben, bezieht sich diese Variante des Parameters *modifier* auf die Anpassung von Aktionen, die Auswirkungen auf eine Annäherung oder Entfernung zu einem erwünschten Parameter haben. Da diese Annäherung bzw. Entfernung aber nicht immer gleich ist, soll die Änderung an der Steuerung dadurch beeinflusst werden, wie groß oder wie klein die Distanz zu einem erwünschten Zustand ist. Damit sollen Aktionen höher belohnt werden, die besonders nah an das nachzuahmende Verhalten reichen und Aktionen stärker bestraft werden, die weiter von diesem Verhalten abweichen.

Für die Umsetzung werden Belohnungen und Bestrafungen in jeweils vier unterschiedlichen Stufen vorgenommen. Dazu werden für Belohnungen ein *positivedistancelevel* und für Bestrafungen ein *negativedistancelevel* definiert. Diese teilen das Intervall von der minimalen bis maximalen Distanz in acht Bereiche auf. Die maximale Distanz wurde in Kapitel 4.2.2.3 (Maximale Distanz einer Instanz zu seinem Cluster einer Zustandsklasse) definiert. Die minimale Distanz ist die kleinste Distanz einer Instanz zum zugewiesenen Cluster, die während des Clusters ermittelt worden ist. Ist eine Instanz deckungsgleich mit ihrem Cluster, so folgt daher der Wert 0. Genau wie die maximale und durchschnittliche Distanz wird sie, gesondert für jede Zustandsklasse, berechnet.

Definition 11 (*Minimale Distanz einer Instanz zu seinem Cluster einer Zustandsklasse*)

Sei i eine Zustandsklasse, a^j und a^k beliebige Cluster dieser Klasse, b eine Instanz, die Cluster a^j zugeordnet wurde, C die Menge aller Instanzen eines Clusters a^k und m die Anzahl der Cluster einer Zustandsklasse.

$$\text{mindistance}(i) = \text{distance}(b, a^j) \text{ mit } \text{distance}(b, a^j) \leq \text{distance}(C, a^k)$$

für alle $1 \leq k \leq m$)

In Abhängigkeit der maximalen, durchschnittlichen und minimalen Distanzen wird der Grad einer Modifikation für eine Annäherung oder Entfernung an erwünschte Zustände vorgenommen. Dabei wird der Grad für eine Erhöhung einer Aktionswahrscheinlichkeit durch einen Parameter *postivedistancelevel* bestimmt. Dieser teilt den Bereich von der minimalen bis durchschnittlichen Distanz in vier gleich große Intervalle. Analog dient der Parameter *negativedistancelevel*, um den Grad einer Verringerung einer Aktionswahrscheinlichkeit festzulegen. Hier werden die vier Intervalle durch die Differenz von

4. Datenverarbeitung und Lernen

maximaler und durchschnittlicher Distanz bestimmt.

Definition 12 (*Grad der Belohnung bzw. Bestrafung in erwünschten Zuständen*)

Sei i eine der vier Zustandsklassen, dann ist:

$$\text{positivedistancelevel}(i) = \frac{\text{averagedistance}(i) - \text{mindistance}(i)}{4}$$

und

$$\text{negativedistancelevel}(i) = \frac{\text{maxdistance}(i) - \text{averagedistance}(i)}{4}$$

Die Idee, die hinter der Berechnung dieser Intervalle steckt, ist, dass ein Maß definiert werden kann, wie nah eine Aktion einen Agenten an einen erwünschten Zustand gebracht hat, bzw. wie weit weg. Dieses Maß entspricht einer Einordnung in die ermittelten Intervalle. Wenn jetzt eine Aktion auf- oder abgewertet werden soll, geschieht dies in Abhängigkeit des gefundenen Maßes. Dadurch wird eine Aktion stärker aufgewertet, die sehr nah an einen erwünschten Zustand herangeführt hat und eine Aktion stärker abgewertet, die weiter weg von einem erwünschten Zustand geführt hat. Dabei wird davon ausgegangen, dass um so weiter die berechnete Distanz zu einem erwünschten Zustand von der durchschnittlichen Distanz abweicht, um so näher, bzw. weiter weg befindet sich der Agent von diesem Zustand.

Die Änderung an den Aktionswahrscheinlichkeiten geschieht in Abhängigkeit des Parameters *modifier2* und eines Distanzlevels.

Definition 13 (*Anpassung einer Aktionswahrscheinlichkeit bei Annäherung zu bzw. Entfernung von einem erwünschten Zustand*)

Sei t der aktuelle Zeitpunkt, π die Steuerung des Agenten, x_{t-n} die Aktion, für die die Wahrscheinlichkeit verändert werden soll, s_{t-n} der Zustand, indem die Aktion ausgeführt wurde, i eine Zustandsklasse und $\text{distance}(s_t, a_t^j)$ die kleinste Distanz eines Clusters dieser Klasse zum momentanen Zeitpunkt.

- *Fall 3:*
Das Distanzmaß eines zurückliegenden Zustandes zum Zeitpunkt $t-n$ ist höher, als das des aktuellen Zustandes $\text{distance}(s_{t-n}, a_{t-n}^j) > \text{distance}(s_t, a_t^j)$ und die berechnete Distanz des aktuellen Zustandes zu ihrem nächstgelegenen Cluster ist kleiner, als die Durchschnittsdistanz aller Instanzen zu ihrem nächsten Clustermittelpunkt in der entsprechenden Zustandsklasse $\text{distance}(s_t, a_t^j) < \text{averagedistance}(i)$. Es gilt:

$$\pi(x_{t-n}, s_{t-n}) = \left(1 + (\text{modifier2} * \frac{\text{averagedistance}(i) - \text{distance}(s_t, a_t^j)}{\text{positivedistancelevel}(i)})\right) * \pi(x_{t-n}, s_{t-n})$$

- *Fall 4:*
Das Distanzmaß eines zurückliegenden Zustandes zum Zeitpunkt $t-n$ ist kleiner, als

4. Datenverarbeitung und Lernen

das des aktuellen Zustandes $distance(s_{t-n}, a_{t-n}^j) < distance(s_t, a_t^j)$ und die berechnete Distanz des aktuellen Zustandes zu ihrem nächstgelegenen Cluster ist größer, als die Durchschnittsdistanz aller Instanzen zu ihrem nächsten Clustermittelpunkt in der entsprechenden Zustandsklasse $distance(s_t, a_t^j) > averagedistance(i)$. Es gilt:

$$\pi(x_{t-n}, s_{t-n}) = (1 + (modifier2 * \frac{distance(s_t, a_t^j) - averagedistance(i)}{negativedistancelevel(i)}) * \pi(x_{t-n}, s_{t-n}))$$

Änderung der Agentensteuerung für zurückliegende Zustände:

Eine Eigenschaft von Markov-Prozessen im Reinforcement-Learning ist es, dass die Wahrscheinlichkeit für eine Aktion nur vom aktuellen Zustand, nicht aber von den letzten Aktionen abhängig ist. Um eine Möglichkeit zu schaffen in den oben beschriebenen vier Fällen der Anpassung auch weiter zurückliegende Aktionen miteinzubeziehen, wird ein *discountfactor* definiert. Dieser sorgt dafür, eine festgelegte Anzahl von Aktionen, die vor dem aktuellen Zustand ausgeführt wurden, ebenfalls mitanzupassen. Dadurch kann gewählt werden, wieviele Aktionen ausschlaggebend für den momentane Zustand sein sollen. Es ist davon auszugehen, dass umso weiter eine Aktion zurückliegt, umso weniger hat sie Einfluss auf die aktuelle Situation. Deshalb wird dieser Faktor so eingesetzt, dass für Aktionen, die weit in der Vergangenheit liegen, Aktionswahrscheinlichkeiten weniger angepasst werden, als für Aktionen, die noch nicht so weit zurückliegen.

Definition 14 (Anpassung von Aktionswahrscheinlichkeiten mit Einbeziehung zurückliegender Aktionen und Zustände:)

Sei *modifier*, wie aus den vier beschriebenen Fällen bekannt, *t* der aktuelle Zeitpunkt, s_{t-n} ein zurückliegender Zustand und x_{t-n} die ausgeführte Aktion, so findet eine Anpassung der Aktionswahrscheinlichkeiten für einen vorherigen Zustand $\pi(s_{t-n}, x_{t-n})$ nach folgendem Schema statt:

- Berechnung des Modifiers für zurückliegende Zustände:

$$modifier = modifier * (1 - \frac{(t - n - 1)}{discountfactor})$$

$$\text{mit } 1 \leq n \leq discountfactor$$

- Anpassung der Aktionswahrscheinlichkeiten für $\pi(s_{t-n}, x_{t-n})$ wie in den oben beschriebenen Fällen mit neuem modifier.

4.2.2.6. Korrektur

Die Summe der Wahrscheinlichkeiten für Aktionen und Richtungen in einem Zustand muss immer 1 bzw. 100 % ergeben. Deshalb müssen nach der Anpassung einer Aktionswahrscheinlichkeit, wie im vorherigen Kapitel beschrieben, auch immer alle anderen Wahrscheinlichkeiten im jeweiligen Zustand angeglichen werden. Dies geschieht indem gespeichert wird, um wieviel Prozent die Wahrscheinlichkeit für eine Aktion auf- oder

4. Datenverarbeitung und Lernen

abgewertet wurde. Dieser Wert entspricht der prozentualen Modifikation¹¹, die entweder zu den restlichen Aktionswahrscheinlichkeiten in diesem Zustand addiert oder von ihnen abgezogen werden muss. Eine Korrektur der anderen Wahrscheinlichkeiten wird für jede Aktion in Abhängigkeit ihrer momentanen Wahrscheinlichkeiten durchgeführt. Das heißt, höhere Wahrscheinlichkeiten werden stärker auf- oder abgewertet, kleinere Wahrscheinlichkeiten schwächer.

Die folgende Definition zeigt, wie Aktionswahrscheinlichkeiten in einem Zustand korrigiert werden, wenn eine Belohnung oder Bestrafung für eine Aktion stattgefunden hat.

Definition 15 (*Korrektur von Aktionswahrscheinlichkeiten*)

Sei s ein Zustand, indem die Aktionswahrscheinlichkeiten verändert werden sollen, x_i die Aktion, für die diese Änderung, wie in Kapitel 4.2.2.5 geändert wird, x_j eine beliebige andere Aktion in diesem Zustand, n die Anzahl der möglichen Aktionen, $\pi_1(x, s)$ die Steuerung des Agenten mit den Aktionswahrscheinlichkeiten vor der Modifikation im Zustand s und $\pi_0(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n, s)$ die Wahrscheinlichkeiten für die Aktionen ohne $\pi_1(x_i, s)$ so gilt:

$$\pi(x_j, s) = \frac{\pi_0(x_j, s)}{\sum_{m=1}^{i-1} \pi(x_m, s) + \sum_{m=i+1}^n \pi(x_m, s)} * \text{modification}$$

mit $\text{modification} = \pi(x_i, s) - \pi_1(x_i, s)$

Damit Aktionswahrscheinlichkeiten nicht unwiderruflich verloren gehen, ist es nicht möglich sie auf 0 zu reduzieren. Dies heißt im Umkehrschluss, dass es nicht möglich ist Aktionswahrscheinlichkeiten auf 1 zu erhöhen. Der Grund dafür ist folgender. Hat ein Zustandsübergang ergeben, dass eine Aktion positiv bewertet wurde und es wäre erlaubt, die Wahrscheinlichkeit auf 1 zu erhöhen, so würden alle anderen Wahrscheinlichkeiten automatisch auf 0 gesetzt. Ergäbe die gleiche Aktion danach in einem anderen Fall aber im gleichen Zustand ein negatives Feedback, so wäre es nach der oben genannten Formel nicht möglich, die ursprünglichen Werte wieder herzustellen. Aktionswahrscheinlichkeiten mit den Werten 0 oder 1 können deshalb nur bei der Initialisierung der Cluster vorkommen.

4.2.2.7. Das Modul

Für das Reinforcement-Learning war es nötig, einige Änderungen am Programmcode von Quake3 zu machen. Auf diese wird im Anhang näher eingegangen.

Das Lernverfahren läuft nach folgendem Schema ab:

1. **Ausgangsbasis:**

Der Agent befindet sich in einem Zustand, welcher durch die vier Zustandsklassen definiert ist.

2. **Wahl der nächsten Aktion:**

Der Agent wählt seine nächste Aktion, aufgrund der Wahrscheinlichkeitsverteilung von Aktionen im aktuellen Zustand

¹¹Dieser wird in der Definition (Korrektur von Aktionswahrscheinlichkeiten) modification genannt.

3. **Aktion:**
Der Agent führt die gewählte Aktion aus.
4. **Beobachtung:**
Beobachtung der Veränderungen innerhalb der virtuellen Welt und Aktualisierung der Attributwerte, die für die Zustandsklassifikation von Bedeutung sind. Dies sind die in Kapitel 3.1 vorgestellten Attribute.
5. **Berechnung des neuen Zustands:**
Der Agent berechnet aufgrund der Attributwerte und der Cluster seinen aktuellen Zustand.
6. **Bewertung des aktuellen Zustands:**
Anhand der Distanz zum aktuellen Zustand oder dem Status von Zuständen „erwünscht“ oder „unerwünscht“ wird überprüft, ob Aktionen vorteilhaft oder negativ bewertet werden sollen.
7. **Definition eines neuen Clusters:**
Falls der Agent ein vorgegebenes Distanzmaß zum berechneten Zustand überschritten hat, wird ein neuer Zustand gebildet. Ebenso geschieht dies, wenn der Agent sich aus einem Zustand befreien soll, aus dem er mit seiner momentanen Steuerung nicht herauskommt.
8. **Anpassung der Aktionswahrscheinlichkeiten:**
Auf- oder Abwertung einer positiv oder negativ bewerteten Aktion im jeweiligen Zustand.
9. **Korrektur:**
Änderung der übrigen Aktionswahrscheinlichkeiten, zur Justierung der Gesamtwahrscheinlichkeit für eine Aktion je Zustand auf 100%.
10. **Wiederholung:**
Die Schritte 2 bis 9 werden solange wiederholt, bis keine Änderungen mehr in der Aktionswahrscheinlichkeitstabelle des Agenten erfolgt oder der Lernprozess abgebrochen wird.

Da dieses Modul einige Daten, die sich auf das Clustern beziehen, immer wieder benötigt, werden diese zusätzlich mit den gefundenen Clustern in der gleichen Datei gespeichert. Diese wird deshalb ergänzt durch die Angaben der minimalen, maximalen und durchschnittlichen ermittelten Distanz jeder Zustandsklasse, die durchschnittlichen Wahrscheinlichkeiten für Aktionen und Richtungen, der Wert für die Einhaltung der Aktionsrichtung und die Anzahl der Cluster vor und nach dem Online-Lernen.

In Abbildung A.4 zeigt der Teil unter den definierten Clustern diese Erweiterungen.

5. Testphase und Evaluierung

In diesem Kapitel geht es um die Umsetzung der in Kapitel 4 beschriebenen Verfahren in die Praxis. Es soll analysiert werden, wie sich ein Agent in der virtuellen Welt verhält, welcher einer Steuerung folgt, die sich aus den Ergebnissen der vorherigen Kapitel ergeben. Dazu werden verschiedenen Szenarien untersucht. Diese reichen, von zu imitierendem Verhalten, wie das Umkreisen eines festen Punktes, bis hin zu komplexen Abläufen, wie der Kampf gegen andere Agenten. Als Grundlage wird dazu, die in Kapitel 2.1.5 beschriebene Map genommen. Jedes der Szenarien wird dabei wieder mit anderen Einstellungen für das Lernen betrachtet. Hierzu werden, die in Kapitel 4 vorgestellten Parameter, für jedes Szenario mit unterschiedlichen Werten gewählt. Nach jedem Test wird anhand ausgewählter Kriterien beurteilt, welche Qualität die Lernverfahren mit den gewählten Parametern in Bezug auf die Imitation von Verhaltensmustern haben. In Kapitel 5.1 werden zunächst die Grundlagen gelegt, damit ein Agent aus den gespeicherten Ergebnissen (Abbildung A.4 und A.5) eine Steuerung in Quake3 ableiten und während des Spiels umsetzen kann. Anschließend werden in Kapitel 5.2 noch einmal die, aus Kapitel 4.2.2 bekannten, Parameter zur Anpassung des Lernverfahrens aufgezeigt und die unterschiedlichen Szenarien besprochen. In Kapitel 5.3 werden die Ergebnisse der verschiedenen Testläufe diskutiert.

5.1. Die Steuerung des Reinforcement-Agenten in Quake3

Um die Ergebnisse durch die in Kapitel 4 aufgeführten Verfahren anwenden zu können, ist es erforderlich, diese in die Bot-Intelligenz eines Agenten in Quake3 zu integrieren. Dazu wird ein beliebiger bereits vorgefertigter Agent aus Quake3 so modifiziert, dass er



Abbildung 5.1.: Der Agent Crash (Quelle [1])

5. Testphase und Evaluierung

nicht mehr die original Steuerung eines Quake3-Agenten verwendet, sondern eine Steuerung die auf den Zuständen und Aktionswahrscheinlichkeiten aus dem Clustermodul basieren. Außerdem wird für diesen Agenten eine Möglichkeit implementiert, wodurch während seiner Handlungen in Quake3 das Verfahren aus dem Reinforcement-Modul angewendet werden kann. In dieser Arbeit wird dazu der vorgefertigte Agent mit Namen „Crash“ benutzt. Die Abbildung 5.1 zeigt diesen Agenten.

Um eine Steuerung, wie sie von den beiden Modulen 4.2.1 und 4.2.2 erzeugt wird, für einen Agenten in Quake3 nutzbar zu machen, wird eine Botschnittstelle in Quake3 implementiert. Diese sorgt in einem ersten Schritt dafür, aus den beiden im Anhang be-

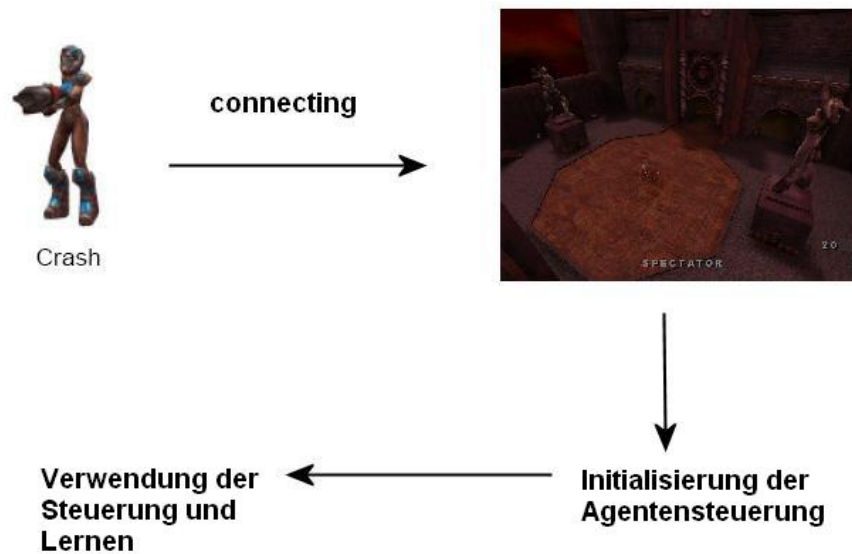


Abbildung 5.2.: Einlesen der modifizierten Steuerung in Quake3

schriebenen Dateien A.4 und A.5 die Cluster und die für jeden Zustand berechneten Aktionswahrscheinlichkeiten einzulesen. Zusätzlich werden, die in Kapitel 4.2.2 erklärten Werte für die maximalen, minimalen und durchschnittlichen Distanzen (*maxdistance*, *middistance*, *averagedistance*), sowie die durchschnittlichen Aktionswahrscheinlichkeiten der Zustände (*averagemove*, *averagedirection*) eingelesen. In einem zweiten Schritt werden einige wichtige Variablen initialisiert. Hierzu werden u.a. die positiven und negativen Distanzintervalle (*postivedistancelevel*, *negativedistancelevel*) berechnet, um die entsprechenden Belohnungen und Bestrafungen des Lernens zu berechnen. Weitere zu initialisierende Variablen gehören zu den Parametern des Reinforcement Learnings, auf die in Abschnitt 5.2.1 eingegangen wird.

Die Abbildung 5.2 zeigt die beschriebenen Schritte. Die Verwendung der Steuerung und das Lernen durch Reinforcement-Learning funktioniert so, wie in Kapitel 4.2.2 erklärt.

5.2. Testphase

Die Testphase dient dazu die Güte der besprochenen Verfahren abzuschätzen. Hierzu bestehen zwei Möglichkeiten. Die erste besteht in einem visuellen Vergleich eines Agentenverhaltens, dessen Daten aufgezeichnet wurden. Die zweite Möglichkeit führt einen Vorher-Nachher Vergleich durch, indem sowohl über den nachzuahmenden Agenten, als auch den imitierenden Agenten die gleichen Daten gesammelt und miteinander verglichen werden.

Ein Testlauf besteht immer aus sieben Schritten:

1. **Wahl eines Szenarios**

An dieser Stelle wird eine von fünf Aufgaben für den nachzuahmenden Agenten gewählt. Die Aufgabenstellungen werden in Kapitel 5.2.2 vorgestellt.

2. **Aufzeichnung von Agentenverhalten (Datengewinnung)**

Es wird ein vom Menschen gesteuerter Agent beobachtet, wie er die jeweilige Aufgabenstellung löst. Dabei werden die aus Kapitel 3.1 definierten Attribute aufgezeichnet.

3. **Wahl der spezifischen Parameter des Lernens**

Das Experimentieren mit unterschiedlichen Einstellungen für das Lernen, soll Rückschlüsse darüber geben, wie groß der Einfluss der einzelnen Parameter auf das Ergebnis ist und welche Einstellungen am geeignetsten sind. Die Parameter werden im nächsten Kapitel vorgestellt.

4. **Anwendung des Lernens**

Hier werden nach der in Kapitel 4.2.1 beschriebenen Vorgehensweise im ersten Lernschritt, aus den aufgezeichneten Daten Zustände und Aktionswahrscheinlichkeiten gelernt. Die zweite Lernphase dient dazu, die Aktionswahrscheinlichkeiten dahingehend zu verbessern, dass der imitierende Agent sich möglichst ähnlich zum Original verhält und insbesondere Zustände vermeidet, in denen sich der nachzuahmende Agent nicht aufgehalten hat. Das Verfahren dazu wurde in Kapitel 4.2.2 erklärt.

5. **Beobachtung des imitierenden Agenten mit modifizierter Steuerung und Evaluierung**

Den Abschluss jeder Testphase bildet die Bewertung des nachahmenden Agenten und die Beurteilung der Parametereinstellungen. Hierzu werden Informationen aus dem Verhalten eines nachahmenden Agenten aufgezeichnet, die dazu benutzt werden, den Lernerfolg zu beurteilen. Anschließend kann miteinander verglichen werden, ob die jeweilige Aufgabenstellung von nachahmenden und zu imitierenden Agenten in gleicher Weise gelöst worden ist. Die Bewertung kann dabei als einfachste Möglichkeit durch die Betrachtung am Monitor geschehen. Andere Möglichkeiten sind ein quantitativer und qualitativer Vergleich von Agentenverhalten. Die Bewertungskriterien werden in Kapitel 5.3 erläutert.

5.2.1. Parameter

Für jedes Szenario ist es möglich das Lernverhalten mittels Parametern zu justieren. Einstellungen können dabei sowohl für das Clustern als auch das Reinforcement Learning vorgenommen werden. Die Parameter und ihre Einstellungsmöglichkeiten sind in der Tabelle 5.2.1 zu finden.

Die Parameter des Clusters ($distancenum + distancenom$) dienen dazu, die Anzahl der Cluster für das kmeans-Verfahren abzuschätzen. Diese wurden eingeführt, um eine Möglichkeit zu haben, die Abschätzung zu beeinflussen. Die Abschätzung selbst ist notwendig, weil das hier verwendete Clusterverfahren, die Anzahl der Cluster als Vorgabe benötigt. Für den Parameter gilt, je höher der Wert, desto weniger Cluster werden definiert. Die Beschränkungen für die zu wählenden Werte ergeben sich dabei aus der in 4.2.1 definierten Formel zur Ermittlung der Clusteranzahl.

In Kapitel 4.2.2 wurden eine ganze Reihe von Parametern definiert, mit deren Hilfe die in dieser Arbeit verwendete Methode des Reinforcement-Learning beeinflusst werden kann. Mit den Parametern *modifier1* und *modifier2* lässt sich beispielsweise anpassen, wie hoch Belohnungen und Bestrafungen für Aktionen ausfallen und sich diese auf zurückliegende Aktionen und Zustände auswirken. Unterschieden wird dabei, ob Aktionen beurteilt werden sollen, die in Zusammenhang mit erwünschten und unerwünschten Zuständen stehen (*modifier1*) oder Aktionen, die nur in Zusammenhang mit erwünschten Zuständen stehen (*modifier2*). Diese Unterscheidung wurde eingeführt, um Aktionen, die in oder aus unerwünschten Zuständen führen, stärker bestrafen oder belohnen zu können. Durch die Änderungen des Parameters kann eine Belohnung in folgender Form vorgenommen werden: Je höher die Belohnung ausfällt, je größer sind die Änderungen, die an der Steuerung des Agenten ausgeführt werden. Ist die Belohnung = 0, so werden keine Änderungen praktiziert und es wird nichts gelernt, da in diesem Fall an der Steuerung keine Änderung vorgenommen wird. Analog trifft dies auch auf eine Bestrafung von Aktionen zu. Um zu verhindern, dass zu schnell Aktionen gelernt werden, die sich im weiteren Verlauf, als weniger positiv erweisen, werden hier Werte zwischen 0 und 1 angesetzt. Dies garantiert eine kleine Belohnung/Bestrafung und damit eine geringfügige Änderung der Steuerung des Agenten in jedem Zyklus. Gleichzeitig besteht dadurch die Möglichkeit, die Steuerung im weiteren Lernverlauf in eine andere Richtung noch korrigieren zu können. Hier gilt je kleiner der Wert, umso mehr Zeit wird für den Lernprozess benötigt, je größer der Wert, umso weniger ist die Möglichkeit zur Korrektur der Aktionswahrscheinlichkeiten zu einem späteren Zeitpunkt gegeben.

Durch die Einstellung eines *discountfactors* wird eine Belohnung oder Bestrafung in stufenweise abgeschwächter Form für zurückliegende Aktionen und Zustände vorgenommen. Dieser Parameter wurde wegen der Markov-Eigenschaft des Reinforcement-Learnings eingeführt. Er dient dazu zu lernen, dass nicht nur die letzte Aktion für den momentanen Zustand des Agenten verantwortlich ist, sondern auch eine Anzahl von zurückliegenden Zuständen, die durch den Parameter bestimmt werden. Dabei kann davon ausgegangen werden, dass umso weiter eine Aktion in der Vergangenheit liegt, umso weniger hat sie Einfluss auf den derzeitigen Zustand des Agenten genommen. Für diesen Parameter werden Werte zwischen 0 und 50 angenommen. Der Wert 0 bedeutet dabei, dass

5. Testphase und Evaluierung

nur Änderungen an der letzten Aktion durchgeführt werden, ein höhere Wert gibt die Anzahl der zurückliegenden Aktionen, die betrachtet werden sollen, an. Werte größer als 50 werden ausgeschlossen, da sich dadurch für weit zurückliegende Zustände je nach *modifier* Änderungen von weniger als 1% ergeben.

Die Parameter *weighting(1-4)* bestimmen die Gewichtung der Zustandsklassen. Wie in Kapitel 4.2.1.3 diskutiert, sind die Zustandsklassen eingeführt worden, weil es u.a. dadurch möglich wird, für bestimmte Aufgabenstellungen, einige Attribute stärker zu berücksichtigen als andere. Dies macht es im Extremfall möglich eine Klasse von Attributen überhaupt nicht zu betrachten oder exklusiv nur eine Zustandsklasse. Da das Gesamtgewicht 100% betragen muss, sind hier nur Werte zwischen 0 und 1 möglich (1 entspricht dabei 100%). Die Summe alle vier Gewichte muss daher ebenfalls 1 ergeben.

Parameter des Clusterverfahrens		
Parameter	Beschreibung	Intervall
Distancenum	legt bei numerischen Attributen fest, wie groß die Distanz zwischen ihnen sein muss, damit sie unterschiedlichen Clustern zugeordnet werden.	[0,100]
Distancenom	legt bei nominalen Attributen fest, wieviele unterschiedliche Attributwerte zu unterschiedlichen Clustern führen.	[0,100]
Parameter des Reinforcement Learnings		
Parameter	Beschreibung	Intervall
Modifier(1,2)	beeinflusst die Stärke von Belohnungen und Bestrafungen.	[0,1]
Discountfactor	bestimmt, wie stark zurückliegende Aktionen für den derzeitigen Zustand verantwortlich gemacht werden sollen.	[0,50]
Weighting(1-4)	definiert die Gewichtung jeder Zustandsklasse. Das Gesamtgewicht der vier Zustandsklassen muss 1 ergeben.	[0,1]
Maxdistance-factor	legt fest, ab welchem Vielfachen der gemessenen maximalen Distanz ein neuer Zustand definiert werden soll.	[1,10]

Tabelle 5.1.: Einstellmöglichkeiten für das Lernverfahren

Der letzte für das Reinforcement-Learning definierte Parameter *maxdistance* gibt vor, wie groß die Differenz der Attributwerte der aktuellen Momentaufnahme eines Agenten zu den Werten des naheliegenden Zustands sein muss, damit ein neuer Zustand gebildet werden soll. Dieser Parameter wurde definiert, weil angenommen wird, dass ab einer bestimmten Distanz, die Ähnlichkeit des ermittelten Zustands und der aktuellen Momentaufnahme des imitierenden Agenten zu gering ist. Für diesen Parameter sind Werte zwischen 1 und 10 möglich. Der Wert 0 bedeutet, dass jede Berechnung zu einem neuen Zustand führt und wird daher ausgeschlossen. Hier gilt: je höher der Wert, umso geringer die Anzahl der online ermittelten Zustände. Der Wert 10 wird verwendet, um auszuschließen, dass online neue Zustände gebildet werden. Dieser Wert kann angenommen werden, da aufgrund der Formel aus Kapitel 4.2.1 für das Bilden neuer Zustände

für die meisten Fälle ausgeschlossen werden kann, dass die Distanz um das zehnfache überschritten wird, selbst wenn wenig Cluster definiert worden sind.

Tabelle 5.1 gibt noch einmal einen Überblick über die verschiedenen Parameter und ihre Einstellmöglichkeiten.

5.2.2. Szenarien

Ein Szenario beschreibt in dieser Arbeit eine spezifische Aufgabe für einen Agenten. Diese beziehen sich alle auf die in Kapitel 2.1.5 beschriebene spezielle virtuelle Welt. Sie werden so gewählt, dass der zu imitierende Agent zunächst möglichst einfache Problemstellungen zu bewältigen hat (siehe Szenario 1). Diese dienen dazu zu untersuchen, ob es möglich ist, mittels der in dieser Arbeit implementierten Lernverfahren, Verhalten nachzuahmen, dass nur von wenigen Faktoren der virtuellen Welt beeinflusst wird. Stellt sich in der Testphase heraus, dass die Qualität der Verhaltensimitation für diese Szenarien gering ist, so kann davon ausgegangen werden, dass mit steigender Anforderung an das Szenario keine Verbesserung erzielt werden kann. Der Grund dafür ist, dass umso mehr Faktoren ein Verhalten beeinflussen, umso schwieriger wird, es die Faktoren zu bestimmen, die den größten Einfluss für eine ausgeführte Aktion haben.

Für jedes Szenario wird das Verhalten eines Agenten aufgezeichnet, der von einem menschlichen Spieler zuvor über eine festgelegte Zeitspanne gesteuert wird. Diese Zeit richtet sich nach dem Schwierigkeitsgrad der Aufgabe, so dass für Szenarien mit einer einfachen Aufgabenstellung eine kürzer Zeit zugrunde gelegt wird und für komplexere eine längere. Grund dafür ist die Annahme, dass sich Verhalten in einfacheren Aufgaben öfter wiederholt und eindeutiger zu erkennen ist, als dies in schwereren Szenarien der Fall ist.

Der Startpunkt ist in allen Szenarien eine Erhöhung auf dem Platz in der Map. Dies gilt, sowohl für den imitierenden, als auch den zu kopierenden Agenten. Damit ist gewährleistet, dass die Aktionen der beiden Agenten von Beginn an vergleichbar sind.

Im folgenden wird jedes Szenario beschrieben. Dazu wird zum einen die jeweilige Aufgabenstellung des Agenten skizziert. Zum anderen wird erläutert, warum dieses Szenario ausgewählt wird und was mit ihm getestet werden soll. Eine Abbildung zeigt jeweils eine Momentaufnahme des Szenarios. Zum Abschluss jeder Beschreibung wird angegeben, wieviel Zeit (*Dauer der Aufzeichnung*) für die Aufzeichnung eines zu imitierenden Agenten verwendet wurde. Außerdem werden Angaben gemacht, die sich auf die jeweilige Aufgabenstellung eines Szenarios beziehen. Dies sind die *Anzahl der Runden* für Szenarien, die sich auf das Umrunden eines bestimmten Punktes in der Welt beziehen. Bei Aufgaben, in denen es zum Konflikt mit anderen Agenten kommt, sind dies die Anzahl der finalen Treffer¹. Insgesamt gibt es fünf unterschiedliche Szenarien.

¹Ein finaler Treffer, ist ein Treffer, der dafür sorgt, dass der getroffene Agent für eine kurze Zeitdauer vom Spiel ausgeschlossen wird,

5. Testphase und Evaluierung

Szenario 1:

In der ersten Aufgabe soll das Verhalten eines Agenten nachgeahmt werden, welcher einen festen Punkt innerhalb der Welt stets in der gleichen Richtung umkreist. Hierzu wird der aus Abbildung 2.7 bekannte Platz genommen und als fester Punkt die Statue rechts im Bild genommen. Abbildung 5.3 zeigt die erste Aufgabe.

Dieses Szenario dient dazu zu zeigen, ob es überhaupt möglich ist durch den beschriebenen Ansatz des Clusters und Reinforcement-Learnings Agentenverhalten zu imitieren. Außerdem sollen aus den einzelnen Parametern die Werte herausgefiltert werden, die sich als am erfolgsversprechendsten erweisen und mit denen in den weiteren Szenarios weitergearbeitet wird.

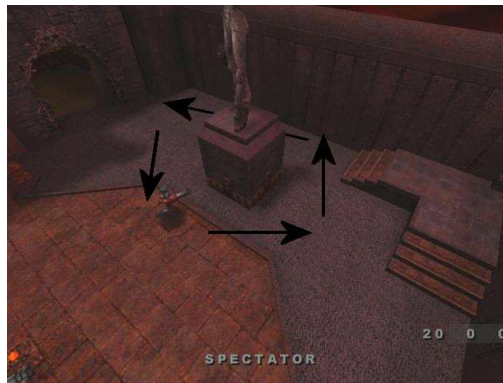


Abbildung 5.3.: Szenario 1: Ein Agent soll in Pfeilrichtung einen festen Punkt mehrmals umkreisen.

Dauer der Aufzeichnung : 1:23 min.

Anzahl der Runden : 20

Szenario 2:

Die Aufgabe des zweiten Szenarios besteht darin, in einem erweiterten Kreis um zwei feste Punkte herumzulaufen. Dies sind die Statue aus dem ersten Szenario und eine zweite Statue, welche sich ebenfalls auf dem Platz in der virtuellen Welt befindet.

Mit dieser Aufgabe soll u.a. getestet werden, wie sich eine längere Route auf die Anzahl der Zustände auswirkt. Der Schwierigkeitsgrad wird angehoben, da zum einen eine längere Strecke überbrückt werden muss, zum anderen weil beide Hindernisse nur an einer Seite passiert werden sollen.

Dauer der Aufzeichnung : 2:30 min.

Anzahl der Runden : 20

5. Testphase und Evaluierung

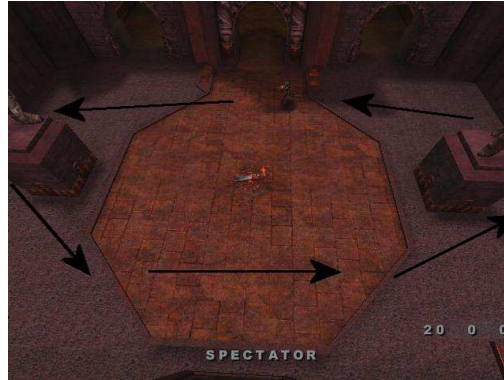


Abbildung 5.4.: Szenario 2: Ein Agent soll in Pfeilrichtung zwei feste Punkte mehrmals umkreisen.

In Abbildung 5.4 ist die Route, welcher der Agent folgen soll, eingezeichnet.

Szenario 3:

In dieser Aufgabe wird der Kreis der beiden vorherigen Szenarien noch einmal erweitert. Diesmal soll der Agent sich zusätzlich auch im Rest der in Kapitel 2.1.5 beschriebenen Map aufhalten und zwar in der Form, dass er beginnend an seiner Startposition rechts an der Statue vorbeiläuft, den rechten Torbogen durchquert, den rechten Gang wählt und die Halle durchläuft. Sein Rückweg führt über den linken Gang, den linken Torbogen, um die linke Statue herum zurück zum Ausgangspunkt.



(a)



(b)



(c)

Abbildung 5.5.: Szenario 3: Der Weg des Agenten durch den rechten Torbogen (a) den rechten Gang (b), durch die Halle (c) und auf der anderen Seite wieder zurück zum Ausgangspunkt.

Die Schwierigkeit in diesem Szenario besteht dabei darin, dass der Agent zum einen eine längere Strecke zurückzulegen hat, zum anderen, dass es ihm gelingen muss, Verhalten so nachzuahmen, dass er einen Weg durch die statischen Objekte (Tore und Durchgänge) der Welt findet. Dabei wird es besonders dadurch noch einmal etwas schwieriger, weil im Gegensatz zu den ersten beiden Aufgaben, der Agent Positionen einnimmt, in denen

5. Testphase und Evaluierung

er sich in entgegengesetzte Richtungen bewegt. So gibt es in der Mitte der Map einen Abschnitt, den er sowohl auf dem Hin- als auch auf dem Rückweg durchläuft, was für die Aktionswahrscheinlichkeiten zwei Bewegungen in entgegengesetzte Richtungen vermuten lässt. Dadurch muss der Agent lernen, ob er sich in einer solchen Position auf dem Hin- oder auf dem Rückweg befindet, um das Verhalten kopieren zu können. Die Abbildung 5.5 stellt den Weg des Agenten durch die Map dar.

Dauer der Aufzeichnung : 4:36 min.
Anzahl der Runden : 20

Szenario 4:

Das vierte Szenario bringt eine weitere Komponente in das Spiel. Der Agent soll sich diesmal nicht nur auf einer festen Route bewegen, sondern zusätzlich auch auf den Kontakt mit einem anderen Agenten reagieren. Hierzu wird der in Kapitel 2.1.2 vorgestellte Agent „Phobes“ in die virtuelle Welt gesetzt. Als Route wird das Umkreisen der Statue aus dem ersten Szenario gewählt mit dem Unterschied, dass der Agent sich nicht die gesamte Zeit bewegen, sondern immer wieder zwischen Statue und rechten Torbogen stehenbleiben soll. Wenn der Agent einen anderen Agenten sieht, so soll er auf ihn schießen. Das Umkreisen der Statue in Abständen von ca.30 Sekunden dient dazu die Munition aufzufrischen und wieder an den selben Punkt zurückzugelangen. Wird der Agent final getroffen, so steigt er am Startpunkt wieder in die Map ein.

Dauer der Aufzeichnung : 5:00 min.
Anzahl der Runden : 8
Anzahl der finalen Treffer : 9

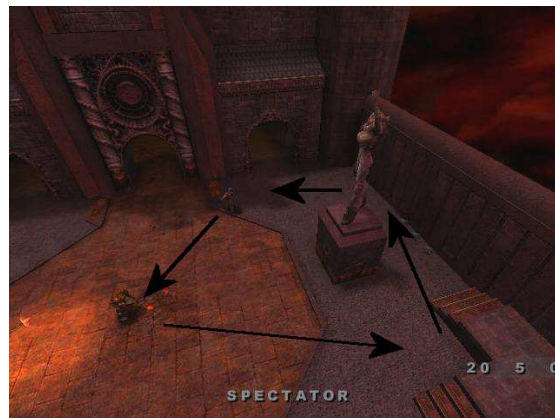


Abbildung 5.6.: Szenario 4: Ein Agent soll sich in die Nähe des rechten Torbogens bewegen und auf Gegner warten. In unbestimmten Abständen soll er zum Raketenwerfer laufen und Munition auffrischen.

5. Testphase und Evaluierung

Zweck dieser Aufgabe soll sein, ob Verhaltensmuster in den Daten gefunden und erlernt werden können, die sich aufgrund einer dynamisch veränderten Umwelt ergeben haben. Die Veränderung der Umwelt bezieht sich dabei auf den Kontakt mit anderen Agenten. Abbildung 5.6 zeigt eine Momentaufnahme aus dieser Aufgabenstellung.

Szenario 5:

Die komplexeste Aufgabe stellt sich dem Agenten im letzten Szenario. Hier werden die Daten aus einem Deathmatch-Spiel zwischen drei verschiedenen Agenten aufgezeichnet. Bei den anderen Agenten handelt es sich um die Agenten „Bones“ und „Phobos“ aus Kapitel 2.1.2. Um die Zeit für die Datenaufzeichnung selbst bestimmen zu können, wird auf zeitliche und punktebasierende Spielbegrenzungen verzichtet². Im Gegensatz zu den anderen Szenarien werden beliebige Einstiegspunkte gewählt.

Das besondere an diesem Szenario ist, dass sich der Agent diesmal völlig frei in der Welt bewegen darf, wodurch ein viel breiteres Spektrum an Verhaltensweisen gefunden werden könnte. Seine Aktionen werden in diesem Fall nicht nur von seiner Position abhängig sein, sondern seinem Status, seiner Wahrnehmung und immer wieder auftretendem Kontakt mit anderen Agenten. Mit diesem letzten Szenario sollen die Grenzen der hier vorgestellten Methode des Reinforcement-Learnings in Zusammenhang mit der Imitation von Verhaltensmustern aufgezeigt werden.

Dauer der Aufzeichnung : 30:00 min.

Anzahl der finalen Treffer : 45



Abbildung 5.7.: Szenario 5: Deathmatch zwischen drei Agenten.

5.3. Evaluierung

In diesem Kapitel sollen die beschriebenen Verfahren anhand der Szenarien bewertet werden. Jede Bewertung erfolgt auf der Basis eines Szenarios, einem Satz von Parame-

²Für den Quake3 Bot sind diese Begrenzungen unerheblich, da er unabhängig von diesen handelt.

5. Testphase und Evaluierung

tern und einem Agenten, der aufgrund dieser Vorgaben versucht einen anderen Agenten zu imitieren. Für die Analyse eines imitierenden Agenten wird dabei die zehnfache Zeit der Dauer der Datenaufzeichnung angenommen³. Dies soll gleichzeitig gewährleisten, ausreichend Daten für die Bewertung zu sammeln und die Testphase, aufgrund der zur Verfügung stehenden Zeit, nicht zu ausgedehnt werden zu lassen.

Die Evaluierung umfasst zwei Phasen. In der Beobachtungsphase soll ein Agent aufgrund der Steuerung aus den Lernverfahren den zuvor analysierten Agenten imitieren. Dabei wird mit verschiedenen Parameterwerten experimentiert, um analysieren zu können, welche Parameter, welche Auswirkungen auf die Imitation von Verhaltensmustern in dieser Arbeit haben. In der Bewertungsphase geht es darum, anhand verschiedener Kriterien herauszuarbeiten, wo die Unterschiede im Verhalten der beiden Agenten liegen.

Quantitative Bewertungskriterien	
Kriterium	Bewertungsansatz
Number of runs	Eine Abweichung bei der Anzahl von Umrundungen oder finaler Treffer innerhalb einer festgelegten Zeit, ist nur dadurch zu erklären, dass sich die Agenten entweder in anderen Zuständen befunden haben oder andere Aktionen ausgeführt wurden.
Number of frags	
Qualitative Bewertungskriterien	
Kriterium	Bewertungsansatz
Distances	Mit Hilfe der Distanzen kann ermittelt werden, wie nah der imitierende Agent tatsächlich an den gefunden Zuständen war.
Action-probabilities	Der Vergleich der durchschnittlichen Aktionswahrscheinlichkeiten verrät, in wie weit Aktionen bei den Agenten voneinander abgewichen sind.
Direction-probabilities	Der Vergleich der durchschnittlichen Richtungswahrscheinlichkeiten verrät, in wie weit Richtungswechsel bei den Agenten voneinander abgewichen sind.
Direction-faithfulness	Die Richtungstreue dient dazu festzustellen, ob es einen Unterschied in der Beibehaltung der zuletzt eingeschlagenen Richtung bei imitierendem und vorgebendem Agenten gab.
Number of cluster	Der Vergleich der Anzahl der Zustände gibt Aufschluss darüber wieviele neue unerwünschte Zustände erlernt worden sind.

Tabelle 5.2.: Bewertungskriterien

Diese Bewertungskriterien werden eingeführt, um zum einen einen Vergleich zwischen direkt messbaren Verhaltensweisen (quantitative Kriterien) anstellen und zum anderen indirekt messbare Verhaltensmuster (qualitative Kriterien) vergleichen zu können. Direkt messbar ist z.B. wie groß der Unterschied zwischen absolvierten Runden in den Szenarien

³Für das erste Szenario bedeutet dies, dass der nachahmende Agent in jedem Test $10 * 1:23 = 12$ min. lang beobachtet wird.

5. Testphase und Evaluierung

1-4 ist. Indirekt messbar ist es, ob zwei Agenten dafür immer den gleichen Weg gewählt haben. Zu diesem Zweck werden Bewertungskriterien gebildet, die Aufschluss darüber geben sollen, in wie weit sich dieser Weg unterscheidet.

Im folgenden werden zum Vergleich zweier Agenten, sowohl Kriterien definiert, mit dessen Hilfe sowohl ein quantitativer, als auch ein qualitativer Vergleich der Verhaltensmuster von zwei Agenten durchgeführt wird.

Die quantitative Bewertung kann direkt in der Beobachtungsphase vorgenommen werden. So können z.B. in Szenario vier und fünf die Anzahl der finalen Treffer gezählt und mit der Anzahl aus dem Lauf des nachzuahmenden Agenten verglichen werden (*Number of frags*). Genauso ist in den ersten vier Szenarien ein Vergleich möglich, indem für beide Agenten die Anzahl der Umrundungen innerhalb einer bestimmten Zeit gezählt werden (*Number of runs*).

Eine qualitative Bewertung erfolgt auf allen Verhaltensähnlichkeiten, die nicht durch die quantitativen Kriterien bestimmt werden können. Diese können in der Beobachtungsphase abgeschätzt werden, beispielsweise, wenn zu erkennen ist, dass der Agent einen bestimmten Weg imitieren kann oder er in einem bestimmten Zustand, ähnliche Aktionen ausführt, als der zu imitierende Agent vorher.

Um zuverlässigere Erkenntnisse über die Ähnlichkeit der beiden Agenten zu bekommen, werden in der Bewertungsphase Daten gesammelt, die sowohl für den imitierenden Agenten, als auch den vorgebenden Agenten berechnet werden. Hierzu gehören die aus Kapitel 4.2.2 bekannten durchschnittlichen Distanzen (*Distances*), die durchschnittlichen Aktionswahrscheinlichkeiten (*Actionprobabilities / Directionprobabilities*), der Wert für die Richtungstreue (*directionfaithfulness*) und die Anzahl der Zustände (*Number of Cluster*). Diese Daten werden im Anschluss an die Beobachtung des Agenten in einer Datei für die Bewertungsphase gesichert. Ein Beispiel für diese Datei ist der Abbildung A.6 zu entnehmen. Die Tabelle 5.2 listet alle Bewertungskriterien auf und erklärt, welche Rückschlüsse aus ihnen gezogen werden können.

5.3.1. Tests und Auswertungen

Insgesamt gibt es fünf Testphasen. Jede dieser Phasen bezieht sich auf eines der fünf Szenarien. Dabei bezieht sich die erste Testphase auf das erste Szenario, die zweite auf das zweite Szenario, usw.. Zu jedem Test wird der Versuchsaufbau und die dazugehörigen Resultate in einer Tabelle zusammengefasst.

Der linke Teil dieser Tabelle gibt dabei die Einstellungen der Parameter für das Lernen an. Da das Clustern die Grundlage für die Steuerung eines Agenten und für die weitere Anwendung des Reinforcement-Learning ist, müssen aus den aufgezeichneten Daten in jedem Test Cluster gebildet werden. Die Parameter *distancenum* und *distancenom* sind daher immer angegeben. Ebenfalls angegeben werden zu jedem Test die einzelnen Gewichtungen *weighting(1-4)*, da diese für die Wahl einer Aktion entscheidend sind. Alle anderen Parameter auf der linken Seite der Tabelle beziehen sich auf Einstellungen bezüglich des Reinforcement-Learnings. Hier werden nur die Parameter angegeben, die in einem Test angewendet werden.

5. Testphase und Evaluierung

Der rechte Teil der Tabelle listet die einzelnen Bewertungsmerkmale auf, wobei die Differenz angibt, um wieviel Prozent ein Wert gestiegen oder gefallen ist. Negative Ausprägungen bedeuten, dass der erreichte Wert im Vergleich zum nachzuahmenden Agenten um die angegebene Prozentzahl gesunken ist, positive Zahlen, dass der Wert gestiegen ist. Eine Ausnahme bilden die Differenzen der Aktions- und Richtungswahrscheinlichkeiten. Hier wird nicht der Unterschied aller durchschnittlichen Wahrscheinlichkeiten für eine Aktion oder Richtung aufgeführt, sondern die größte gemessene Differenz⁴. Ist eine Differenz 0, so ist kein Unterschied zwischen den Agenten zu erkennen. Das Bewertungskriterium *Number of Cluster* gibt an wieviele neue Cluster innerhalb des Online Lernens definiert wurden. Die Zahl ist absolut und beschränkt sich auf die erste Zustandsklasse (Positionsattribute), da in allen anderen Klassen online in keinem der Tests neue Zustände erlernt wurden. Es werden nur Kriterien angegeben, die bewertet werden können.

5.3.1.1. Testphase 1

Die erste Testphase bezieht sich auf die Aufgabenstellung des ersten Szenarios (Umrunden einer Statue). Im ersten Teil wird untersucht, mit welcher Anzahl an Clustern welche Ergebnisse der Verhaltensimitation erzielt werden können (Test 1A-C). Im zweiten Teil (Test 1D-F) wird überprüft, welche Auswirkungen die Gewichtung der einzelnen Zustandsklassen auf die Steuerung eines nachahmenden Agenten haben. Das Lernen beschränkt sich in den Tests zum ersten Szenario daher auf das Clusterverfahren.

Test 1A:

Im ersten Test wurden die Parameter so gewählt, dass wenig Zustände ermittelt werden, um zu überprüfen, ob bereits eine geringe Anzahl von Zuständen ausreichend ist, das Verhalten eines Agenten abzubilden. Auf eine unterschiedliche Gewichtung der Zustandsklassen wurde verzichtet, da es in den ersten Tests darum geht, eine Annäherung an eine gute Anzahl von Zuständen zu bekommen.

Szenario 1: Wenig Zustände							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	50	Weighting(1)	0.25	Number of runs	-98	Dist to Pos	53
Distancenom	4	Weighting(2)	0.25	Number of frags	-	Dist to State	7
Modifier1	-	Weighting(3)	0.25	Actionprob.	1	Dist to See	4
Modifier2	-	Weighting(4)	0.25	Directionprob.	2.2	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-81	Number of Clust.	-
Anzahl Cluster: 2-1-10-1							

Tabelle 5.3.: Parameterbelegung und Ergebnisse für Test 1A

Aufgrund der gewählten Parameter ergeben sich durch das Clustern der Daten aus dem

⁴Die Differenz aller Wahrscheinlichkeiten beträgt stets 0, da die Addition der Wahrscheinlichkeiten 1 ergeben muss.

5. Testphase und Evaluierung

ersten Szenario 14 Zustände⁵. Große Abweichungen ergeben sich im Bereich der zurückgelegten Runden -98% und in der Einhaltung der Runden -81%.

Dieses Experiment kann als gescheitert angesehen werden, da der imitierende Agent bereits an den qualitativen Bewertungskriterien scheitert. So gelang es ihm nicht innerhalb der gleichen Zeit, wie im vorgegebenen Szenario, die Statue auch nur ein einziges mal zu umrunden. Die große Abweichung im Bereich der Einhaltung einer Aktionsrichtung, deutet darauf hin, dass der Agent seinen Lauf um die Statue häufiger unterbrach, als dies der Agent vor ihm getan hat. Das Resultat ist eine längere Zeitspanne, die benötigt wurde, um diesen Punkt in der Welt zu umrunden. Es ist daher möglich, dass für dieses Szenario die Anzahl der Cluster zu gering gewählt wurde.

Test 1B:

Um zu untersuchen, ob eine höhere Anzahl von Zuständen ein besseres Ergebnis der Verhaltensimitation ergibt, wird in diesem Test die Anzahl der Zustände erhöht. Dazu werden die Werte für die Parameter *distancenum* und *distancenom* verringert. Dies wird erreicht, indem die Distanz für numerische Attribute, die unterschiedlichen Clustern zugeordnet werden, im Gegensatz zum ersten Test halbiert wird. Außerdem bewirkt die Halbierung des Parameterwertes *distancenom*, dass auch mehr Zustände in Klassen mit nominalen Attributen gefunden werden. Die Gewichtung der Zustandsklassen bleibt unverändert, um das Ergebnis mit dem ersten Test vergleichen zu können.

Szenario 1: Mittlere Anzahl an Zuständen							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.25	Number of runs	-92	Dist to Pos	74
Distancenom	2	Weighting(2)	0.25	Number of frags	-	Dist to State	9
Modifier1	-	Weighting(3)	0.25	Actionprob.	0.2	Dist to See	-1
Modifier2	-	Weighting(4)	0.25	Directionprob.	1.5	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-69	Number of Clust.	-
Anzahl Cluster: 15-2-20-1							

Tabelle 5.4.: Parameterbelegung und Ergebnisse für Test 1B

Durch die Senkung der Parameterwerte für das Clustern wurden insgesamt 38 Zustände definiert. Besonders die Anzahl der Zustände in der ersten Zustandsklasse sind von 2 im ersten Test auf 15 im zweiten Test angestiegen. Auffällige Abweichungen gibt es nach wie vor im Bereich der Anzahl der Runden. Die Differenz der Einhaltung der Richtung *directionfaithfulness* ist leicht gesunken, die Differenz beim Bewertungskriterium *dist to Pos* gestiegen.

Die Erhöhung der Anzahl der Cluster hat im Vergleich zum ersten Test zwar eine Verbesserung gebracht, doch ist die Nachahmung von Verhaltensmustern auch in diesem Test gescheitert. Zwar gelang es dem Agenten, mit der aus dem Clustern resultierenden Steuerung, zweimal die Statue zu umrunden, doch ist er immer noch weit von den

⁵Anzahl Cluster 2-1-10-1: 2 Zustände für Zustandsklasse 1, 1 Zustand für Zustandsklasse 2, usw. Die Einteilung der Attribute in Zustandsklassen ist in Tabelle 4.1 abgebildet.

5. Testphase und Evaluierung

20 Runden entfernt, die der Agent in der gleichen Zeit im Szenario absolviert hat. Als Grund dafür ist anzuführen, dass dem Agenten online Zustände zu seiner aktuellen Momentaufnahme zugewiesen wurden, die aufgrund der hohen berechneten Distanz nicht der Momentaufnahme entsprachen. Daraus kann folgen, dass der Agent sich nicht so verhalten hat, wie es ein zu kopierender Agent in der gleichen Momentaufnahme getan hat. Dies bezieht sich ausschließlich auf die Attribute der Position. Die Aussagen zur Einhaltung der Richtung können aus dem Ergebnis des ersten Tests übernommen werden. Die Anzahl der Cluster oder die Art und Weise der Gewichtung der Zustandsklassen ist daher eventuell noch nicht ideal gewählt.

Test 1C:

In diesem letzten Versuch zu den Einstellungen der Parameter *distancenum* und *distancenom* wird getestet, ob mit einer weiteren Erhöhung der Anzahl an Zuständen eine Verbesserung der Verhaltensimitation bewirkt wird. Zu diesem Zweck wird der Parameter *distancenum* so gewählt, dass für die Position des Agenten für jeden 9×13^6 großen Abschnitt in der Welt ein neues Cluster berechnet wird⁷. Die Anzahl der Cluster in Zustandsklassen mit nominalen Attributwerten wird durch den Parameter *distancenom* in diesem Versuch maximiert⁸.

Szenario 1: Maximierung der Zustände							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	10	Weighting(1)	0.25	Number of runs	-92	Dist to Pos	270
Distancenom	1	Weighting(2)	0.25	Number of frags	-	Dist to State	244
Modifier1	-	Weighting(3)	0.25	Actionprob.	0	Dist to See	1
Modifier2	-	Weighting(4)	0.25	Directionprob.	1	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-70	Number of Clust.	-
Anzahl Cluster: 119-12-21-1							

Tabelle 5.5.: Parameterbelegung und Ergebnisse für Test 1C

⁶

$$k(1) = k_{xpos}(195, 566) * k_{ypos}(1828, 2360) = \frac{371}{4 * 10} * \frac{532}{4 * 10} = 9 * 13$$

mit

$$min_{xpos} = 195, max_{xpos} = 566, min_{ypos} = 1828, max_{ypos} = 2360$$

⁷Zum Vergleich sei noch einmal angemerkt, dass der Agent mit jeder Vorwärtsbewegung einen Bereich von 20×20 Punkten in der Welt überbrückt. Der Agent überspringt also bei der hier gewählten Anzahl an Clustern sogar Zustände.

⁸

$$k(3) = \frac{12}{1} + \frac{12}{1} + \frac{12}{1} + \dots = 72 \text{ mit Number_of_seedirections} = 12$$

Es ist zu beachten, dass für die Zustandsklasse 3 mit insgesamt 6 Attributen nach der Formel aus Kapitel 4.2.1.1 mehr Cluster abgeschätzt werden als tatsächlich gebildet. Dies ist damit zu begründen, dass es während des implementierten Clusterverfahrens möglich ist, die Anzahl der Cluster zu verringern.

5. Testphase und Evaluierung

Die Anzahl der Zustände ist, wie zu erwarten, auf insgesamt 153 angestiegen. Dies sind über 100 Zustände mehr als im zweiten Test, was vor allem dem enormen Anstieg in Zustandsklasse 1 zu verdanken ist. Die Anzahl der Cluster mit nominalen Attributen ist im Gegensatz kaum erhöht. Bei Betrachtung der Bewertungskriterien fällt auf, dass es zwei signifikante Abweichungen gibt. Dies sind die großen Distanzabweichungen in der Zustandsberechnung der Position und des Status des Agenten. Die restlichen Ergebnisse sind fast identisch mit den Resultaten aus Test 1B.

Eine weitere Erhöhung der Anzahl der Zustände konnte keine Verbesserung der Verhaltensimitation herbeiführen, da die Ergebnisse sich kaum von denen aus Test 1B unterscheiden. Auf den ersten Blick scheint es sogar so, dass es bezüglich der Zuordnung von Momentaufnahmen zu Zuständen eine erhebliche Verschlechterung gegeben hat. Die Erklärung hierfür ist die erhöhte Anzahl an Clustern für die erste und zweite Zustandsklasse. Bedingt durch die Tatsache, dass sehr viele Zustände gebildet wurden, lag die durchschnittliche Distanz der Instanzen beim Clustern bei 0.02. Eine Abweichung von 270% entspricht also einer durchschnittlichen Distanz von 0.075 im Test. In den ersten beiden Tests lag die Abweichung bei 0.15 bzw. 0.05 für die erste Zustandsklassen. Das gleiche gilt für die zweite Zustandsklasse. Eine Verschlechterung des Ergebnisses ist daher nicht eingetreten. Der Grund für das Misslingen der beiden letzten Experimente (Test 1B und 1C) muss also vor allem daran liegen, dass die Gewichtung der einzelnen Zustandsklassen nicht optimal gelöst wurde.

Test 1D:

Da mit der Variation der Anzahl an Zuständen keine Qualitätssteigerung der Imitation von Verhaltensmustern erreicht worden ist, wird in den nächsten Tests mit den Gewichtungen der Attribute experimentiert. Dazu wird in diesem Test auf die Clustereinstellungen aus Test 1B zurückgegriffen. Bei dieser Einstellung hat sich gezeigt, dass der Agent mit der daraus resultierenden Anzahl der Cluster die gleichen Ergebnisse erzielt, wie mit einer wesentlich höheren Anzahl von Zuständen, aber bessere, als mit einer geringeren Anzahl. Es wird also zunächst davon ausgegangen, dass die ermittelten Zustände aus 1B für dieses Szenario ausreichend sind. Die Gewichtung der Zustandsklassen erfolgt so, dass in diesem Test auf die Zustandsklasse 2 (Status) und 4 (Feind) verzichtet wird. Wie schon bei der Entwicklung der Zustandsklasse besprochen, ist es nicht für jedes Szenario nötig, alle Attribute zu betrachten. Weil in diesem Szenario kein Feinkontakt bei der vorgegebenen Aufgabenstellung existiert und sich deshalb auch am Status des Agenten keine aufgabenentscheidenden Veränderungen ergeben,⁹ kann auf diese Attribute verzichtet werden. Die Attribute der anderen beiden Zustandsklassen fließen je zur Hälfte in die Steuerung des Agenten ein.

Die Anzahl der Zustände ist mit der Anzahl aus Test 1B identisch. Die Differenz der Richtungstreue, der Distanzen für die Position, und der absolvierten Runden ist gesunken. Die Differenz der Distanz bei der Berechnung der Zustände in der zweiten Klasse erheblich gestiegen.

⁹Für alle Szenarien in denen der Agent nur einen bestimmten Weg zurücklegen soll, spielt weder seine Munition, noch sein augenblicklicher Gesundheitszustand eine Rolle.

5. Testphase und Evaluierung

Szenario 1: Beschränkung auf zwei Zustandsklassen							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.5	Number of runs	-80	Dist to Pos	42
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	160
Modifier1	-	Weighting(3)	0.5	Actionprob.	0.2	Dist to See	-1
Modifier2	-	Weighting(4)	0	Directionprob.	1.4	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-36	Number of Clust.	-
Anzahl Cluster: 15-2-20-1							

Tabelle 5.6.: Parameterbelegung und Ergebnisse für Test 1D

Der imitierende Agent bewältigte die Aufgabe besser als in den vorausgegangenen Tests. Ihm gelang es die Statue fünfmal zu umkreisen. Die Abweichungsminderung der durchschnittlichen Distanzen und die Angleichung der Einhaltung der Aktionsrichtung im Vergleich zu den ersten Tests deuteten darauf hin, dass sich die Agenten in ihrem Verhalten angenähert haben. Die erhöhte Differenz für *dist to State* spielt keine Rolle, da Zustände aus der zweiten Zustandsklasse, aufgrund der Gewichtung ($\text{weighting}(2) = 0$) nicht in den Gesamtzustand des Agenten einfließen. Somit ist die Steuerung des imitierenden Agenten unabhängig von diesen Attributen.

Test 1E:

Da das Verhalten aus Test 1D immer noch stark vom Verhalten des nachzuziehenden Agenten abweicht, wird im folgenden Versuch gezeigt, was passiert, wenn der Agent sich bei der Berechnung seiner nächsten Aktion nur an seiner Position orientiert. Die Gewichtung der Zustandsklasse 1 wird dazu maximiert, die anderen Parameter bleiben unverändert.

Szenario 1: Beschränkung auf eine Zustandsklasse							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	1	Number of runs	-20	Dist to Pos	6
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	49
Modifier1	-	Weighting(3)	0	Actionprob.	0.1	Dist to See	1
Modifier2	-	Weighting(4)	0	Directionprob.	1.8	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-20	Number of Clust.	-
Anzahl Cluster: 15-2-20-1							

Tabelle 5.7.: Parameterbelegung und Ergebnisse für Test 1E

Die drei bisher beurteilten Bewertungskriterien *number of runs*, *directionfaithfulness* und *dist to Pos* haben sich im Vergleich zu den bisherigen Tests deutlich verbessert, d.h. die Differenzen haben sich verringert. Eine weitere Steigerung beim Durchlaufen der Zustände ist insbesondere in der ersten Zustandsklasse kaum noch zu erreichen. Hier haben sich die durchschnittlich ermittelten Distanzen auf 6% angeglichen. Die Anzahl der Zustände ist die gleiche, wie in den Tests 1B und 1E.

5. Testphase und Evaluierung

Es zeigt sich durch die Einstellungen der Parameter in diesem Test, dass der Agent schon sehr nah an die vorgegebenen 20 Runden des ersten Szenarios herankam. Dies ist zum einen darin zu begründen, dass der Agent seine einmal eingeschlagene Richtung im Vergleich zu allen bisherigen Tests ähnlich, wie der nachzuahmende Agent einhielt. Zum anderen waren die Momentaufnahmen des kopierenden Agenten fast deckungsgleich mit den ermittelten Zuständen. Dadurch ist anzunehmen, dass der nachahmende Agent einen ähnlichen Weg um die Statue wählte, wie es sein Vorgänger getan hat.

Test 1F:

Da die Gewichtung der Attribute, wie sie in Test 1E vorgenommen wurde, gute Ergebnisse gebracht hat, wird in diesem letzten Versuch zur ersten Testphase analysiert, ob die Verhaltensimitation mit der Anzahl der Zustände aus Test 1C verbessert werden kann. Es wird daher die Wechselwirkung von einer erhöhten Anzahl von Zuständen mit wenigen Attributen betrachtet. Dazu wird die Gewichtung wie im letzten Test gewählt, die Parameter für die Abschätzung der Cluster wie im Test 1C.

Szenario 1: Erhöhte Anzahl an Zuständen in Wechselwirkung mit wenigen Attributen			
Parameter	Wert	Parameter	Wert
Distancenom	10	Weighting(1)	1
Distancenom	1	Weighting(2)	0
Modifier1	-	Weighting(3)	0
Modifier2	-	Weighting(4)	0
Maxdistfactor	-	Discountfactor	-
Keine Bewertung			
Anzahl Cluster: 119-12-21-1			

Tabelle 5.8.: Parameterbelegung und Ergebnisse für Test 1F

Es findet keine Beurteilung der Qualität der Verhaltensimitation aufgrund der Bewertungskriterien statt. Der Grund dafür ist, dass sich das Verhalten, des nachahmenden Agenten sichtbar von dem zu kopierenden Mustern abhebt, weshalb dieser Test vorzeitig abgebrochen wird. Eine Beurteilung der Kriterien für die Abschätzung der Güte der Imitation im Vergleich zu den bisherigen Tests wäre somit problematisch.

Das Kopieren eines Agenten mit den hier vorgenommenen Parametern ist gescheitert. So bewegte sich der Agent von seiner Startposition einen Schritt nach vorne und blieb stehen. Das Experiment wurde daher abgebrochen. Der Grund dafür ist, dass aufgrund der großen Anzahl an Zuständen, ein Cluster gebildet worden ist, welches nur aus einer einzigen Instanz besteht, in der als Aktion „standing“ vermerkt war. Der daraufhin modifizierte Agent gelangte nun genau in diesen Zustand und blieb stehen. Da sein Zustand aber nur von seiner Position abhing, änderte sich dieser somit nicht mehr und der Agent konnte diesen Zustand nicht mehr verlassen. Es erweist sich also als problematisch, die Anzahl der zu ermittelten Zustände so zu erhöhen, dass das Risiko erhöht wird, dass ein Cluster nur aus einer Instanz bestehen kann.

Zusammenfassung Testphase1:

Als Ergebnisse der ersten Testphase lässt sich vor allem ableiten, dass für Aufgaben, in denen es nur um die Navigation geht¹⁰, die Beschränkung auf die Attribute der Positionsklasse ausreichend ist (siehe Test 1D+E). Außerdem hat sich gezeigt, dass einige Attribute für die Verhaltensimitation im ersten Szenario ein schlechtes Ergebnis liefern (siehe Test 1A-C). Daraus lässt sich schliessen, dass die Position des Agenten das wichtigste Mittel ist, Verhaltensmuster in Abhängigkeit mit Aktionen zu bringen.

Für die Anzahl der Zustände hat sich ergeben, dass zu viele Zustände kein besseres Ergebnis bringen (siehe Test 1B+C) und in Zusammenhang mit wenig Attributen (siehe Test 1F) dazu führen können, dass der Agent einen Zustand nicht mehr verlassen kann. Zu wenig Zustände sind dagegen bereits im ersten Versuch gescheitert (siehe Test 1A). Um einen Agenten ausschliesslich eine Navigation erlernen zu lassen, scheinen daher die Parametereinstellungen aus den Tests 1B und 1D+E am besten geeignet zu sein. Hier ist davon ausgegangen worden, dass nach der in Kapitel 4.2.1 vorgestellten Formel für einen Bereich von $37 \cdot 53$ ¹¹ innerhalb der virtuellen Welt ein Zustand benötigt wird. Insgesamt kann gesagt werden, dass es gelungen ist, innerhalb des ersten Szenarios einem Agenten mittels Clustern Verhaltensmuster erlernen zu lassen und dieses zu imitieren.

5.3.1.2. Testphase 2

Die zweite Testphase beschäftigt sich mit der Imitation von Verhaltensmustern innerhalb des zweiten Szenarios (Umrunden von zwei Statuen). Hier geht es darum, den Agenten zusätzlich durch Reinforcement-Learning in der Welt lernen zu lassen. Dies soll Erkenntnisse über die einzelnen Lernparameter bringen und die Auswirkungen aufzeigen, wenn zusätzlich während der Verhaltensimitation gelernt wird. Zu diesem Zweck wird zum einen untersucht, wie sich die Belohnung und Bestrafung von Aktionen (Test 2B+2C) auswirkt, zum anderen wie sich das Lernen bezüglich zurückliegender Aktionen auswirkt (Test 2D). Da sich die Parameter für das Clustern und der Steuerung des Agenten aus Test 1E als bisher am besten erwiesen haben, werden diese übernommen.

Test 2A:

In diesem ersten Versuch wird zunächst getestet, ob die Parameter des Clusters auch im zweiten Szenario zum Erfolg führen. Diese werden daher aus dem Test 1E übernommen. Um die Ergebnisse mit der ersten Testphase vergleichen zu können, bleibt das Reinforcement-Learning in diesem ersten Test deaktiviert.

Es werden insgesamt 69 Zustände definiert. Der Agent schaffte 75% der Runden aus dem zweiten Szenario. Alle Werte aus den Bewertungskriterien liegen sehr nah an den

¹⁰Szenario 1 verlangt ausschließlich das Umrunden einer Statue.

¹¹

$$k(1) = k_{xpos}(195, 566) * k_{ypos}(1828, 2360) = \frac{371}{4 * 25} * \frac{532}{4 * 25} = 37 * 53$$

5. Testphase und Evaluierung

Szenario 2: Beste Zustandsanzahl und Gewichtung aus Testphase 1 ohne Lernen							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distanzenum	25	Weighting(1)	1	Number of runs	-25	Dist to Pos	36
Distanzenom	2	Weighting(2)	0	Number of frags	-	Dist to State	-6
Modifier1	-	Weighting(3)	0	Actionprob.	1	Dist to See	1
Modifier2	-	Weighting(4)	0	Directionprob.	2	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-6	Number of Clust.	-
Anzahl Cluster: 43-2-23-1							

Tabelle 5.9.: Parameterbelegung und Ergebnisse für Test 2A

Werten aus dem Clustern.

Die Parameter, die sich in der ersten Testphase bewährt haben, führen auch in diesem Szenario zum Erfolg.

Test 2B:

In diesem Test wird untersucht, wie sich das Belohnen und Bestrafen von Aktionen auf die Imitation von Verhaltensmustern auswirkt. Dabei wird zunächst nur getestet, was das Lernen innerhalb erwünschter Zustände bewirkt. Zu diesem Zweck wird der Parameter *modifier2* in diesem Test so eingestellt, dass Aktionswahrscheinlichkeiten mit einem Abzug von 20 % bestraft, falls eine Aktion negativ bewertet wird und belohnt, falls die letzte Aktion ergeben hat, dass sich der Agent dadurch einem erwünschten Zustand genähert hat. Die Einstellungen bezüglich des Clusters werden aus Test 2A übernommen.

Szenario 2: Lernen mit Aktionsbewertung							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distanzenum	25	Weighting(1)	1	Number of runs	-40	Dist to Pos	46
Distanzenom	2	Weighting(2)	0	Number of frags	-	Dist to State	0.5
Modifier1	-	Weighting(3)	0	Actionprob.	3	Dist to See	1
Modifier2	0.2	Weighting(4)	0	Directionprob.	6	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-10	Number of Clust.	2
Anzahl Cluster: 43-2-23-1							

Tabelle 5.10.: Parameterbelegung und Ergebnisse für Test 2B

Im Vergleich zum ersten Test dieses Abschnitts, sind einige Verschlechterungen der Nachahmung abzulesen. Dies sind die Anzahl der zurückgelegten Runden und eine Erhöhung der maximalen Abweichung einer Wahrscheinlichkeit für Aktionen und Aktionsrichtungen.

Das Lernen bewirkte in diesem Test, dass der Agent nach einer gewissen Zeit einigen Aktionen bzw. Aktionsrichtungen den Vorzug gab, als anderen. Dadurch kam es dazu, dass der Agent in einigen Runden seinen Kreis um eine der Statuen zu eng zog und daran für einige Sekunden hängen blieb. Aus diesem Grund brauchte er für eine Runde

5. Testphase und Evaluierung

im Durchschnitt etwas länger, als der zu imitierende Agent, wodurch er insgesamt 3 Runden weniger schaffte, als der Agent aus Test 2A. Das Belohnen und Bestrafen von Aktionen, die entweder an einen erwünschten Zustand annähern oder entfernen, brachte in diesem Test keine Verbesserung der Verhaltensimitation.

Test 2C:

Damit der Agent lernen kann, sich schneller aus Situationen zu befreien, an denen er an einem Hindernis hängengeblieben ist, wird der *modifier1* aktiviert¹². In diesem Test wird dazu so gelernt, dass das Auflaufen auf eine Statue, als kurzfristig ausweglose Situation deklariert wird und ein unerwünschter Zustand gebildet wird. Der Parameter *modifier1* sorgt dann im Anschluss daran, dass dieser Zustand verhindert wird bzw., dass der Agent einen Weg lernt, aus diesem Zustand zu entkommen. Dazu werden Aktionen, die den imitierenden Agenten in einen solchen Zustand hineinführen mit einem Abzug von 40% der Wahrscheinlichkeit bestraft, bzw. bei Entkommen mit 40% belohnt. Zum Vergleich mit dem vorherigen Test bleiben die anderen Parameter unverändert.

Szenario 2: Lernen mit Aktionsbewertung und neuen Zuständen							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	1	Number of runs	-40	Dist to Pos	51
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	-6
Modifier1	0.4	Weighting(3)	0	Actionprob.	5	Dist to See	2
Modifier2	0.2	Weighting(4)	0	Directionprob.	3	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-12	Number of Clust.	4
Anzahl Cluster: 43-2-23-1							

Tabelle 5.11.: Parameterbelegung und Ergebnisse für Test 2C

Der Agent findet insgesamt vier neue unerwünschte Zustände, die er vermeiden soll. An den Bewertungskriterien ist keine Steigerung der Qualität der Imitation abzulesen, da sich die Werte nur geringfügig von denen aus Test 1B unterscheiden.

Die Lernphase mit der Möglichkeit neue Zustände, für den Fall, dass der Agent hängengeblieben ist, zu bilden und anschließend zu lernen, diese Zustände zu vermeiden oder aus ihnen herauszukommen, hat keine Verbesserungen der Steuerung in Bezug auf die Imitation erbracht.

Test 2D:

Der Test 2D dient dazu, zu beurteilen, wie sich der *discountfactor* auf das Lernen auswirkt. Er wird in diesem Test so gewählt, dass für den Fall, dass ein Zustand positiv oder negativ bewertet wird, nicht nur die letzte Aktion dafür belohnt oder bestraft wird, sondern die letzten acht Aktionen mitbewertet werden.

¹²Es wird davon ausgegangen, dass die Imitation von Verhaltensmustern besser gelingt, wenn der Agent nicht hängengeblieben ist. Da der zu kopierende Agent, in diesem Szenario bestrebt war Runden zu drehen, würde das „Hängenbleiben“ an einem Hindernis zu einer kleinen Rundenzahl führen.

5. Testphase und Evaluierung

Szenario 2: Lernen mit Aktionsbewertung, neuen Zuständen und Vergangenheitsbezug							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenom	25	Weighting(1)	1	Number of runs	-85	Dist to Pos	106
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	125
Modifier1	0.4	Weighting(3)	0	Actionprob.	2	Dist to See	4
Modifier2	0.2	Weighting(4)	0	Directionprob.	22	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	8	Dirfaithfuln.	-58	Number of Clust.	4
Anzahl Cluster: 43-2-23-1							

Tabelle 5.12.: Parameterbelegung und Ergebnisse für Test 2D

Die Anzahl der gelaufenen Runden ist im Gegensatz zum Test 2C um die Hälfte gesunken. Genauso signifikant sind der Anstieg im Bereich der Bewertungskriterien *directionfaithfulness* und *directionprobability*. Der Agent bevorzugte nach dem Lernen eine Richtung wesentlich mehr als vorher und unterschied sich auch deutlich bei der Einhaltung der Richtung. Ebenfalls einen starken Anstieg der Differenz ergaben sich bei der Zuordnung der Momentaufnahmen zu Zuständen.

Das Ergebnis dieses Tests lässt sich unmittelbar an der Differenz der gelaufenen Runden festmachen. Auch alle anderen gestiegenen Abweichungen deuten auf eine Entfernung zum vorgegebenen Verhalten. Der Grund für diesen Anstieg ist möglicherweise, dass der Agent bei der Belohnung und Bestrafung von Aktionen in der Vergangenheit Aktionen belohnte bzw. bestrafte, die nicht zwangsläufig in den aktuell bewerteten Zustand geführt haben müssen. Das Resultat daraus ist, dass der Agent in seinen Aktionen von Richtung zu Richtung sprang und sich das Verhalten von dem des nachzuahmenden Agenten während des Lernens stetig entfernte.

Zusammenfassung Testphase2:

Das Ergebnis bezüglich der Wahl der Parameter für das Clustern aus Testphase 1 konnte bestätigt werden. Mit der daraus resultierenden Anzahl an Zuständen konnte eine Steuerung aus den Daten gelesen werden, der es gelingt einen Agenten Verhalten nachahmen zu lassen (siehe Test 2A). Nicht gelungen dagegen ist es den Grad der Verhaltensähnlichkeit aus Test 2A durch Reinforcement-Learning zu verbessern. So ergaben die Tests, in denen der Agent Annäherungen an gewünschte Zustände belohnen sollte und er die Möglichkeit besaß neue Zustände zu bilden, eine geringfügige Verschlechterung der Imitation, anstatt einer Verbesserung. Noch schlechter war das Ergebnis des Lernens, wenn der Agent zusätzlich auch vergangene Aktionen belohnen oder bestrafen sollte.

5.3.1.3. Testphase 3

Die Versuche dieser Testphase beziehen sich auf das dritte Szenario(Umrunden der gesamten Map). Da alle Versuche mit Reinforcement-Learning innerhalb der zweiten Testphase zu keiner Verbesserung der Verhaltensimitation geführt haben, erfährt das Lernen in der dritten Phase eine Änderung. Anstatt Aktionen positiv und negativ zu bewer-

5. Testphase und Evaluierung

ten, wird auf eine Bestrafung von Aktionen verzichtet. Damit soll überprüft werden, ob das Zusammenspiel von Belohnungen und Bestrafungen dafür verantwortlich war, dass das Lernen während des Agierens in der Welt von Quake3 mißlang. Der Testablauf ist danach der gleiche, wie in der vorherigen Testphase. Zunächst wird ohne Reinforcement-Learning gelernt (Test 1A), danach wird in jedem Test ein Parameter aus dem online lernen hinzugeschaltet. Da es auch in diesem Szenario in erster Linie um die Navigation eines Agenten geht, wird die Gewichtung gewählt, die sich für diesen Fall als bisher am besten herausgestellt hat (Test 3A). Diese wird im weiteren Verlauf der Testphase aber leicht verändert (Test 3B-E), um bessere Ergebnisse erzielen zu können. Die Parameter des Clusters bleiben in der gesamten Testphase unverändert, wie in Testphase 2, da sich die dadurch gefundene Anzahl an Zuständen bisher als ausreichend erwiesen haben.

Test 3A:

Ziel dieses ersten Tests zum dritten Szenario ist es zu zeigen, ob bei komplexeren Aufgaben alleine das Clustern ausreicht, um Verhalten zu imitieren. Dazu werden die bisher bewährten Einstellungen für das Clustern und die Gewichtung der Zustandsklassen aus den vorherigen Tests übernommen. Zusätzliches Lernen durch Reinforcement-Learning wird nicht durchgeführt. Die Schwierigkeiten bei der Imitation sind in der Beschreibung dieses Szenarios in Kapitel 5.2.2 erläutert worden und bestehen vor allem darin einen Weg durch die Hindernisse in der Welt in zwei verschiedenen Richtungen (Hin+Rück) zu kopieren.

Szenario 3: Beste Zustandsanzahl und Gewichtung aus Testphase 2 ohne RL				
Parameter	Wert	Parameter	Wert	Keine Bewertung
Distancenum	25	Weighting(1)	1	
Distancenom	2	Weighting(2)	0	
Modificator1	-	Weighting(3)	0	
Modificator2	-	Weighting(4)	0	
Maxdistfactor	-	Discountfactor	-	
Anzahl Cluster: 163-3-28-1				

Tabelle 5.13.: Parameterbelegung und Ergebnisse für Test 3A

Es werden insgesamt 195 Zustände gebildet. Die meisten in der ersten Zustandsklasse. Da alle Experimente mit den hier vorgenommenen Einstellungen nach einer Zeit abgebrochen werden müssen, wird eine Bewertung aufgrund der Kriterien nicht vorgenommen. Dies würde, wie schon bei Test 1F, zu einer Verzerrung des Vergleichs mit anderen Tests dieser Phase ergeben, da unterschiedlich lang Informationen für den Vergleich erhoben worden wären.

Der erste Versuch ohne Reinforcement-Learning Verhalten zu imitieren scheitert teilweise. Zwar gelang es durch das Clustern, das Verhalten in einer Steuerung für den Agenten abzubilden, es konnte aber nicht verhindert werden, dass der Agent an einer Wand oder einem Torbogen hängen blieb. Bei den Versuchen mit diesen Parametern schaffte der modifizierte Agent im Durchschnitt drei Runden über die gesamte Map, danach lief er sich fest und der Versuch wurde abgebrochen, weil der Agent sich nicht befreien konnte.

5. Testphase und Evaluierung

Dafür gibt es zwei Gründe. Erstens wird ein Zustand nur in Abhängigkeit der Position ermittelt, weshalb es viele Zustände in diesem Szenario gibt, die nur eine Bewegungsrichtung kennen¹³. Dies führte dazu, dass der Agent vor dem Abbruch eines Versuchs in einen Zustand geriet, in dem ihn die zur Verfügung stehende Aktion immer auf ein Hindernis auflaufen lies¹⁴. Zweitens war in diesem Test das Reinforcement-Learning-Modul deaktiviert, weshalb keine unerwünschten Zuständen gelernt werden konnten.

Test 3B:

Da das größte Problem von Test 3A war, dass der Agent nicht lernen konnte, sich aus ausweglosen Situationen zu befreien, soll in diesem Test durch Reinforcement-Learning gelernt werden, diese Zustände als unerwünscht zu deklarieren und sich aus ihnen zu befreien. Dazu werden mittels des Parameters *modifier2* Aktionswahrscheinlichkeiten um 40% aufgewertet, wenn die entsprechende Aktion dazu geführt hat, dass der Agent einen solchen Zustand verlassen hat. Danach wird untersucht, ob sich das Verhalten eines Agenten durch diese Maßnahme besser kopieren lässt, als in Test 3A. In einigen hier nicht aufgeführten Untersuchungen stellte sich heraus, dass es dem imitierenden Agenten gelang sich nach einiger Zeit aus den in Test 3A beschriebenen Situationen zu befreien, wenn der Agent in seine Steuerung zusätzlich die Attribute der Wahrnehmung einfließen lässt. Dies ist dadurch zu begründen, dass die Aktion des Agenten nicht mehr nur alleine von seiner Position abhängt. Einige weitere Tests ergaben hier, dass das beste Verhältnis zwischen guter Verhaltensimitation und Vermeidung von einer Endlosschleife in einem

Szenario 3: Modifizierte Gewichtung mit Lernen unerwünschter Zustände							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.95	Number of runs	-40	Dist to Pos	285
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	229
Modifier1	0.4	Weighting(3)	0.05	Actionprob.	1.3	Dist to See	0
Modifier2	-	Weighting(4)	0	Directionprob.	1.9	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-61	Number of Clust.	5
Anzahl Cluster: 163-3-28-1							

Tabelle 5.14.: Parameterbelegung und Ergebnisse für Test 3B

Zustand 0.95 zu 0.05 war. Aus diesem Grund werden für die folgenden Tests der Zustand zu einem kleinen Anteil auch von der Wahrnehmung des Agenten bestimmt.

Ein weiterhin bestehender Nachteil ist aber, dass der Agent immer noch zu häufig in die in Test 3A beschriebene Situation gelangt und es teilweise minutenlang dauert, bis er sich aus dieser Situation lösen kann. In diesem Test werden deshalb für eine solche Lage online neue Zustände gebildet und anschließend gelernt werden, sich aus diesen zu befreien,

¹³Der zu imitierende Agent hatte die Aufgabe die Map immer gegen den Uhrzeigersinn zu umrunden.

¹⁴Angenommen der Agent befindet sich in einem Zustand südlich eines Torbogens und sein Weg führt nach Norden, dann befindet er sich möglicherweise etwas rechts oder links dieses Durchgangs im gleichen Zustand. Da es für ihn in diesem Zustand keine Wahl der Richtung gibt, läuft er fortan nach Norden gegen eine Wand.

5. Testphase und Evaluierung

bzw. sie zu meiden. Dazu werden zunächst die Auswirkungen einer hohen Belohnung für den Wechsel aus einem unerwünschten in einen erwünschten Zustand betrachtet.

Der Agent erlernte fünf neue Zustände. Er absolvierte 60% der Runden, die der zu imitierende Agent schaffte. Die Abweichung bei der Einhaltung der Richtung ist im Gegensatz zur zweiten Testphase wieder gestiegen. Die Differenz bei der Zuordnung von Momentaufnahmen zu den Zuständen aus dem Clustern ist für die Positionsattribute ebenfalls hoch. Die Differenz bei den Attributen der Klasse des Agentenstatus muss wiederum nicht beachtet werden, da diese Attribute durch die Gewichtung keinen Einfluss auf die Steuerung des Agenten haben.

Das Experiment gelang. Der Agent konnte sich nach der Lernphase in diesem Szenario bewegen, ohne dauerhaft in einem Zustand hängen-zubleiben. Dass ihm dennoch nicht die gleiche Anzahl an Runden gelang, ist damit zu begründen, dass er einige male in einem unerwünschten Zustand landete. Ein weiterer Grund dafür ist, dass die Bewegungen des Agenten etwas ruckartiger wurden, was auch an der Abweichung des Kriteriums *directionfaithfulness* abzulesen ist. Diese Abweichung hat ihren Grund in der Wegfindung aus einem unerwünschten Zustand. Hierbei versuchte der Agent sich in unterschiedliche Richtungen zu bewegen, weshalb die Richtung in diesen Zuständen in der Lernphase nicht dauerhaft eingehalten wurde. Die Erhöhung im Bereich der Differenz der Distanzen ergibt sich dadurch, dass es im Schnitt mehrere Sekunden dauerte, bis der Agent sich aus einem unerwünschten Zustand befreit hatte. Da er in diesem Fall immer eine gewisse Distanz zu einem erwünschten Zustand aufwies, ist die Differenz dementsprechend hoch.

Test 3C:

Da der Agent im vorherigen Test um 285% schlechtere Distanzwerte aufzuweisen hatte und auch die Anzahl der zurückgelegten Runden nicht den Runden des nachzuziehenden Agenten entsprachen, wird in diesem Test geprüft, welche Einwirkungen die Bewertung aller Aktion auf die Imitation ergeben. Dazu werden alle Aktionen um 20% aufgewertet, die nah genug an einen Zustand heranzuführen. (siehe auch Kapitel 4.2.2 für die Definition von „nah“).

Szenario 3: Belohnen von zustandsnähernden Aktionen							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.95	Number of runs	-10	Dist to Pos	174
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	257
Modifier1	0.4	Weighting(3)	0.05	Actionprob.	1.4	Dist to See	0
Modifier2	0.2	Weighting(4)	0	Directionprob.	2.7	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-64	Number of Clust.	5
Anzahl Cluster: 163-3-28-1							

Tabelle 5.15.: Parameterbelegung und Ergebnisse für Test 3C

Die größten Änderungen im Vergleich zu Test 3B gab es bei der Anzahl der zurückgelegten Runden durch die Map und eine um über 100% gesunkene Differenz in den Distanzberechnungen. Ansonsten ergaben sich wenig Änderungen zum vorherigen Test.

5. Testphase und Evaluierung

Durch die zusätzliche Belohnung von Aktionen, die sich erwünschten Zuständen nähern, konnte eine deutliche Verbesserung der Imitation erreicht werden. Der Grund dafür ist, dass es dem Agenten durch die zusätzliche Belohnung gelang sich mehr an erwünschten Zuständen zu orientieren (Senkung der Abweichung der berechneten Distanzen *dist to pos*) und unerwünschte Zustände zu vermeiden. Dadurch blieb er weniger in unerwünschten Zuständen hängen und konnte ähnlich viele Runden drehen, wie der zu imitierende Agent.

Test 3D:

Die nächste Änderung des Versuchsaufbaus betrifft die abgestufte Modifikation von Aktionen, die zu einer positiven Bewertung des aktuellen Zustands geführt haben. Der *discountfactor* wird dazu so gesetzt, dass den letzten acht Aktionen unterstellt wird, dass sie für den derzeitigen Zustand mit verantwortlich sind. Um die Ergebnisse mit den anderen Tests der dritten Testreihe vergleichen zu können, werden die anderen Parameter wie in Test 3C gewählt.

Szenario 3: Lernen mit Vergangenheitsbezug							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.95	Number of runs	0	Dist to Pos	124
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	227
Modifier1	0.4	Weighting(3)	0.05	Actionprob.	1.4	Dist to See	0
Modifier2	0.2	Weighting(4)	0	Directionprob.	3.8	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	8	Dirfaithfuln.	-63	Number of Clust.	12
Anzahl Cluster: 163-3-28-1							

Tabelle 5.16.: Parameterbelegung und Ergebnisse für Test 3D

Die Lernphase brachte bezüglich des quantitativen Bewertungskriteriums *number of runs* eine hundertprozentige Ähnlichkeit zum nachzuahmenden Agenten. Eine weitere Annäherung konnte auch bei den durchschnittlichen Distanzen *dist to pos* erzielt werden. Alle anderen Kriterien weisen keinen besonderen Fortschritt des Lernens im Vergleich zu Test 3C auf.

Insgesamt kann gesagt werden, dass trotz einer Abweichung in den qualitativen Bewertungskriterien, mit den oben gewählten Parametern die bisher beste Imitation innerhalb der dritten Testreihe erzielt werden konnte. Der Grund hierfür liegt vor allem in der gleichen Anzahl der zurückgelegten Runden.

Test 3E:

Im letzten Versuch zum dritten Szenario wird geprüft, wie sinnvoll es bezüglich der Verhaltensimitation ist, neue Zustände zu lernen, falls sich die aktuelle Momentaufnahme zu sehr von einem zugeordneten Zustand unterscheidet. Dazu wird in diesem Test erstmals der Parameter *maxdistfactor* untersucht. Dieser wird so gewählt, dass immer dann ein neuer Zustand gebildet wird, wenn die Distanz zum zugewiesenen Zustand doppelt so

5. Testphase und Evaluierung

hoch, wie die maximale Distanz beim Clustern war (siehe Kapitel 4.2.2 zur Bestimmung unerwünschter Zustände aufgrund der Distanz). Damit soll gewährleistet werden, dass unerwünschte Zustände gelernt werden, es gleichzeitig aber nicht zu viele werden. Alle anderen Parameter bleiben unverändert.

Szenario 3: Lernen von neuen Zuständen							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenom	25	Weighting(1)	0.95	Number of runs	0	Dist to Pos	127
Distancenom	2	Weighting(2)	0	Number of frags	-	Dist to State	219
Modifier1	0.4	Weighting(3)	0.05	Actionprob.	1.5	Dist to See	0
Modifier2	0.2	Weighting(4)	0	Directionprob.	3	Dist to Enemy	0
Maxdistfactor	2	Discountfactor	8	Dirfaithfuln.	-61	Number of Clust.	43
Anzahl Cluster: 163-3-28-1							

Tabelle 5.17.: Parameterbelegung und Ergebnisse für Test 3E

Insgesamt werden so 43 neue Zustände ermittelt. Eine weitere Annäherung an den Agenten bezüglich der Distanzmaße konnte nicht erzielt werden. Auch alle anderen Bewertungskriterien lassen keine Verbesserung der Imitation erkennen.

Das Bilden neuer Zustände hat somit keine Änderungen mehr in der Testphase zu diesem Szenario erbracht. Dies könnte darauf hinweisen, dass die Ergebnisse aus dem letzten Test bereits gut gewesen sind.

Zusammenfassung Testphase3:

Die dritte Testphase hat gezeigt, dass eine bessere Imitation von Verhalten gelernt werden kann, wenn die Umsetzung des Reinforcement-Learnings in dieser Arbeit darauf reduziert wird, Aktionen nur zu belohnen. Die Ursache dafür kann darin liegen, dass Aktionen bestraft werden, die möglicherweise keine negativen Auswirkungen auf die Imitation von Agentenverhalten haben¹⁵.

Mit dieser Modifizierung des Reinforcement-Learning konnten einige weitere Erkenntnisse gewonnen werden. So zeigte sich, dass eine Einschränkung der Attribute auf eine Zustandsklasse für die Nachahmung von Verhalten negative Auswirkungen haben kann. Dies ist dann der Fall, wenn dadurch Zustände erreicht werden können, in denen die verfügbaren Aktionen nicht ausreichen, um den Zustand wieder verlassen zu können (Test 1A). Um aus diesen Zuständen entfliehen zu können, wurden neue Zustände gebildet, mit denen es gelang, den Agenten aus diesen problematischen Situationen zu lösen (Test 1B-E). Noch weiter ließen sich die Ergebnisse steigern, indem der Agent für Aktionen belohnt wurde, die ihn in möglichst ähnliche Situationen führten, wie den zu imitierenden Agenten (Test 1C-E). Die größte Verbesserung konnte aber dadurch erzielt

¹⁵ Angenommen ein Agent wechselt von einem erwünschten Zustand A in einen erwünschten Zustand B, dann ist auch dieser Zustandswechsel bezüglich der Imitation erwünscht. Wenn der Agent sich im Zustand B aber danach weit genug entfernt von diesem Zustand befindet (Definition Zustand fern siehe Kapitel 4.2.2), dann wird seine letzte Aktion, nach Definition in Kapitel 4.2.2 bestraft, obwohl sie eigentlich erwünscht war.

5. Testphase und Evaluierung

werden, dass für jeden Zustand nicht nur die letzte, sondern auch einige Aktionen davor in Zusammenhang gebrachte wurden (Test 1D+E). Hierdurch gelang es den Agenten im Bereich der Anzahl der Runden im Szenario perfekt nachzuahmen. Aufgrund der guten Ergebnisse in Test 1D wurde im abschliessenden Test 1E keine Verbesserung mehr erreicht.

5.3.1.4. Testphase 4

Die vierte Testphase beschäftigt sich mit der Imitation von Verhaltensmustern in Szenario Nummer vier. Erstmals spielen in diesem Szenario auch Auseinandersetzungen mit anderen Agenten eine Rolle. Um zu untersuchen mit welchen Attributen ein solches Verhalten nachgeahmt werden kann, werden unterschiedliche Gewichtungen für die einzelnen Zustandsklassen getestet (Test 4A+B). Dies geschieht zunächst ohne, in einem weiteren Test (Test 4C) mit Reinforcement-Learning. Abschliessend (Test 4D+E) wird experimentiert, ob mit einer Änderung der Zustandsbewertung eine Erhöhung des Lernerfolgs erzielt werden kann.

Die beste Einstellung für das Clustern wird aus den letzten Testphasen übernommen, das System des Reinforcement-Learning reduziert sich ab sofort auf das Belohnen von Aktionen.

Test 4A:

Da im vierten Szenario nicht nur die Navigation eine Rolle spielt, wird davon ausgegangen, dass durch alle Attribute, Verhalten wiedergegeben werden kann. Um zu überprüfen, ob alle Attribute in diesem Szenario den gleichen Einfluss besitzen, werden in diesem ersten Versuch alle Attribute gleich gewichtet. Dies bedeutet, es wird davon ausgegangen, dass eine Aktion eines Agenten gleichermaßen von seiner Position, seinem Zustand, seiner Wahrnehmung und anderen Agenten in seiner Nähe beeinflusst wird. Da es darum geht die Gewichtung zu untersuchen, wird auf Reinforcement-Learning verzichtet.

Szenario 4: Gewichtung der Attribute gleichverteilt ohne Reinforcement-Learning							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenom	25	Weighting(1)	0.25	Number of runs	-100	Dist to Pos	633
Distancenom	2	Weighting(2)	0.25	Number of frags	-86	Dist to State	30
Modifier1	-	Weighting(3)	0.25	Actionprob.	38	Dist to See	0
Modifier2	-	Weighting(4)	0.25	Directionprob.	4.7	Dist to Enemy	1.3
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-61	Number of Clust.	-
Anzahl Cluster: 40-13-20-22							

Tabelle 5.18.: Parameterbelegung und Ergebnisse für Test 4A

Es werden insgesamt 75 Zustände gebildet. Große Abweichungen gibt es im Bereich der Anzahl der Runden und finaler Treffer. Es gibt mindestens eine Aktion, die vom imitierenden Agenten wesentlich öfter oder weniger ausgeführt wird. Die größte Differenz gibt es bei der Zustandsberechnung der Position des Agenten.

5. Testphase und Evaluierung

Dem Agenten gelang es nicht einmal, neue Munition aufzunehmen. Dies ist damit zu erklären, dass sein Verhalten sich stark von den Verhaltensmustern des zu imitierenden Agenten abhob. Indizien dafür sind vor allem die bisher größte gemessene Differenz bei den Distanzen in den beiden Positionsattributen und die um 38% unterschiedliche Ausführung einer Aktion. Grund für die extreme Abweichung im Bereich der Position, ist dass der Agent sich auf Teilen der Map bewegte, die der nachzuahmende Agent nicht durchlaufen hat. Dies ist wiederum darauf zurückzuführen, dass er Aktionen häufiger ausführte, als der zu imitierende Agent, was ihn in andere Bereiche der Map bringt. Bezüglich von Treffern dreht sich das Ergebnis gegenüber dem nachzuahmenden Agenten komplett. Hier gelang dem Agenten im Schnitt nicht ein finaler Treffer, während er von seinem Konkurrenten in diesem Szenario dreimal so häufig markiert wurde. Die Ursache dafür kann darin liegen, dass der nachahmende Agent durch die hier vorgenommene Gewichtung seine Aktion nur zu einem viertel davon abhängig machte, ob er sich in einer Kampfsituation befindet. Die Imitation ist mit den Einstellungen aus Tabelle 5.18 deshalb mißlungen.

Test 4B:

Wie schon die vorherigen Testphasen ergeben haben, scheinen für die Imitation die Position des Agenten die wichtigsten Attribute zu sein. In diesem Test sollen sie daher doppelt so hoch bewertet werden. Da es im Vergleich zum nachzuahmenden Agenten ebenfalls deutliche Defizite bei der Anzahl der gelandeteten Treffer im ersten Test gab, sollen Außerdem die Attribute, welche mit Feindkontakt in Verbindung stehen etwas stärkeren Einfluss erhalten. Damit wird untersucht, ob die Imitation von Verhaltens-

Szenario 4: Anpassung der Gewichtung der Attribute ohne Reinforcement-Learning							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distanzenom	25	Weighting(1)	0.5	Number of runs	-100	Dist to Pos	477
Distanzenom	2	Weighting(2)	<i>0.1</i>	Number of frags	-90	Dist to State	23
Modifier1	-	Weighting(3)	<i>0.1</i>	Actionprob.	39	Dist to See	0
Modifier2	-	Weighting(4)	<i>0.3</i>	Directionprob.	9	Dist to Enemy	-1
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-77	Number of Clust.	-
Anzahl Cluster: 40-13-20-22							

Tabelle 5.19.: Parameterbelegung und Ergebnisse für Test 4B

mustern erfolgreicher ist, wenn Attribute, die sich für die Navigation als gut erwiesen haben stärker berücksichtigt werden. Gleichzeitig wird überprüft, ob das Ergebnis aus Test 4A, dadurch begünstigt wurde, dass die Attribute der vierten Zustandsklasse(Feind) zu wenig Einfluss auf das Verhalten des Agenten hatten.

Das Ergebnis dieses Test stimmt zum großen Teil mit den Resultaten aus Test 4A überein. Die größten Abweichungen sind in den selben Bereichen (*number of runs*, *number of frags*, *actionprobability*, *dist to pos*) festzustellen.

Da das Resultat dieses Tests kaum Veränderungen zum Ergebnis zu Test 4A bringt, ist davon auszugehen, dass allein mit der Gewichtung der Attribute keine Annäherung der

5. Testphase und Evaluierung

Steuerung eines Agenten an zu imitierende Verhaltensmuster erreicht werden kann.

Test 4C:

Da die ersten beiden Tests keine guten Ergebnisse gebracht haben, wird in Test 4C untersucht, ob mit Lernen von Verhalten während der Aktionen, eine besseres Ergebnis erzielt werden kann. Um die Ergebnisse mit dem Test aus 4B vergleichen zu können, werden hierfür die Gewichtungen aus diesem Test übernommen. Für das Lernen werden die Einstellungen aus der Endphase der dritten Testphase gewählt, da sich diese dort bewährt haben.

Szenario 4: Erhöhte Gewichtung der Attribute für die Position/Feind mit Reinforcement-Learning							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.5	Number of runs	-90	Dist to Pos	632
Distancenom	2	Weighting(2)	0.1	Number of frags	-50	Dist to State	23
Modifier1	0.4	Weighting(3)	0.1	Actionprob.	38	Dist to See	-2
Modifier2	0.2	Weighting(4)	0.3	Directionprob.	14	Dist to Enemy	1
Maxdistfactor	-	Discountfactor	8	Dirfaithfuln.	-59	Number of Clust.	3
Anzahl Cluster: 40-13-20-22							

Tabelle 5.20.: Parameterbelegung und Ergebnisse für Test 4C

Das Lernen mit den Parameterwerten aus Test 4B für das Clustern, brachte eine leichte Verbesserung bezüglich der quantitativen Bewertungskriterien. So gab es im Durchschnitt eine Aufnahme von neuer Munition innerhalb der gleichen Zeit, wie im Szenario und die Agenten hatten ein ausgeglichenes Verhältnis von finalen Treffern. Die Abweichungen in der Zustandsberechnung für die Position des Agenten und das Ausführen von Aktionen bleiben unverändert hoch.

Das Resultat dieses Versuchs zeigt, dass es zwar eine leichte Verbesserung bezüglich Kampfhandlungen durch das Lernen gab, sich das Verhalten aber immer noch deutlich von dem zu imitierenden Agenten unterscheidet. Zum einen befand er sich häufig in Positionen, die nicht den Positionen des nachzahnenden Agenten entsprachen, zum anderen wählte er mindestens eine Aktion, wie in den Tests zuvor, wesentlich häufiger oder seltener aus.

Test 4D:

Da eine Imitation von Agentenverhalten mit den bisherigen Versuchsaufbauten innerhalb des vierten Szenarios nicht funktionierte, wird in diesem Test untersucht, ob mit einer Modifizierung der Zustandsbewertung eine Verbesserung erreicht werden kann. Dazu wird von folgendem ausgegangen. Die ersten drei Testphasen haben gezeigt, dass die Imitation der Navigation eines Agenten gut gelungen ist, wenn die Zustände des Agenten zum überwiegenden Teil aus seiner Position definiert werden. Da aber virtuelle Kämpfe in diesem Szenario eine große Rolle spielen, müssen vor allem auch die Attribute der vierten Zustandsklasse Berücksichtigung finden. Dies wird in diesem Test so

5. Testphase und Evaluierung

geschehen, dass zwei übergeordnete Zustände gebildet werden. Der erste beinhaltet alle Situationen, in denen der Agent sich nicht in Kontakt mit anderen Agenten befindet, der andere umfasst alle Kampfhandlungen. Bei der Gewichtung der Attribute sind demnach nur noch die ersten drei Zustandsklassen von Bedeutung, da sich der Agent nur bei Feindkontakt auf die Attribute der vierten Zustandsklasse bezieht. Die Attribute der Position erhalten dabei wieder das Hauptgewicht. Alle anderen Einstellungen bezüglich des Lernens bleiben, wie im vorausgegangenen Versuch.

Szenario 4: Modifizierte Zustandsermittlung mit Lernen neuer Zustände							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenom	25	Weighting(1)	0.8	Numb.of runs	-60	Dist to Pos	428
Distancenom	2	Weighting(2)	0.1	Numb.of frags	-40	Dist to State	39
Modifier1	0.4	Weighting(3)	0.1	Actionprob.	1	Dist to See	3
Modifier2	0.2	Weighting(4)	-	Directionprob.	15	Dist to Enemy	-1
Divergencefactor	-	Discountfactor	8	Dirfaithfulness	-24	Numb.of Clust.	2
Anzahl Cluster: 40-13-20-22							

Tabelle 5.21.: Parameterbelegung und Ergebnisse für Test 4D

Alle Abweichungen konnten im Vergleich zu Test 4C gesenkt werden. Die größten Verbesserungen wurden in den Bereichen der Einhaltung der Richtung und der Ähnlichkeit für die Ausführung von Aktionen erzielt. Auch bei der Anzahl der Treffer und Runden hat sich eine Annäherung ergeben.

Mit der überarbeiteten Methode der Zustandsklassifizierung ergab sich eine deutliche Annäherung der Agenten. Darauf weisen die Werte für die Anzahl der Runden um die Statue zwecks Munitionsaufnahme und die erzielten Treffer. Die Anzahl der Runden kann dabei auf die Ähnlichkeit der Aktionen und die Einhaltung von Richtungen zurückgeführt werden. Die Anzahl der erzielten Treffer wurde möglicherweise dadurch ausgelöst, dass der Agent, nach der Modifikation der Zustandsbewertung ähnlich häufig die Aktion „attack“ in Konfliktsituationen ausführte, wie der zu imitierende Agent. Dies spiegelt sich ebenfalls in dem Resultat des Bewertungskriteriums actionprobability wider. Die Nachahmung von Agentenverhalten ist somit mit der oben beschriebenen Methode und den Einstellungen aus Tabelle 5.21 in diesem Szenario bisher am erfolgsversprechendsten.

Test 4E:

Im letzten Versuch der vierten Testreihe wird geprüft, ob sich die Imitation, durch die in dieser Arbeit verwendeten Lernverfahren, weiter verbessern lässt, wenn zu der geänderten Zustandsbewertung aus Test 4D, die bisher beste Gewichtung gewählt wird. Diese ist die Gewichtung aus der Testphase 3 mit nochmals erhöhtem Einfluss der Positionsattribute auf die Steuerung des Agenten. Alle anderen Einstellungen werden aus dem letzten Test übernommen. Damit unterscheiden sich die beiden letzten Tests nur in der Gewichtung.

Die Auswertung dieses Versuches zeigt, dass die Differenzen im Verhalten der beiden Agenten nochmals reduziert werden konnten. Die Abweichung in der Zustandsberech-

5. Testphase und Evaluierung

Szenario 4: Modifizierte Zustandsermittlung mit Gewichtung aus bisherigen Testphasen mit Lernen neuer Zustände							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenom	25	Weighting(1)	0.95	Number of runs	0	Dist to Pos	161
Distancenom	2	Weighting(2)	0	Number of frags	-15	Dist to State	37
Modifier1	0.4	Weighting(3)	0.05	Actionprob.	4	Dist to See	-1
Modifier2	0.2	Weighting(4)	-	Directionprob.	5	Dist to Enemy	2
Maxdistfactor	-	Discountfactor	8	Dirfaithfuln.	-43	Number of Clust.	3
Anzahl Cluster: 40-13-20-22							

Tabelle 5.22.: Parameterbelegung und Ergebnisse für Test 4E

nung beträgt nur noch die Hälfte im Vergleich zum letzten Test, die Abweichungen in den zurückgelegten Runden und der Anzahl der Treffer wurde gesenkt.

Durch die in diesem Test vorgenommene Gewichtung der Attribute konnte auch in dieser Testreihe, wie zuvor in Testreihe 3, das beste Ergebnis erzielt werden. Da das Kopieren einer Navigation des Agenten durch die vorgenommene Gewichtung, wie in vorherigen Tests gesehen, bevorzugt wird, gelangen dem Agenten die gleiche Anzahl an Runden in diesem Szenario. Zusätzlich wurde durch den verstärkten Einfluss der Attribute aus Zustandsklasse 1(Position) erreicht, dass sich der Agent näher an den Zuständen dieser Klasse hielt, was u.a. zu der gleichen Anzahl an Runden führt. Die gestiegene Anzahl an Treffern kann dadurch erklärt werden, dass der imitierende Agent, häufiger neue Munition aufgenommen hat¹⁶.

Zusammenfassung Testphase4:

In den Tests zum vierten Szenario wurde aufgrund der neuen Komponente von feindlichen Agenten untersucht, welchen Einfluss die einzelnen Attribute auf die Verhaltensimitation einnehmen. Hier ergab, dass eine Gleichbehandlung aller Attribute nicht den gewünschten Effekt brachte (Test 1A). Auch andere Gewichtungen konnten das Ergebnis nur unwesentlich verbessern (Test 1B). Ein Versuch die Schwächen der bis dahin verwendeten Methode der Einflussnahme der Attribute auf Agentenverhalten auszumerzen, brachte eine Verbesserung der Imitation. Hier wurde versucht sich durch Reinforcement-Learning an das Verhalten des nachahmenden Agenten anzunähern (Test 1C). Da sich die beiden Agenten in ihrem Handeln aber auch nach dem Lernprozess zu sehr unterscheiden, wurde für die beiden abschliessenden Tests die Zustandsbewertung überarbeitet (Test 1D+E). Dazu wurden zwei übergeordnete Zustände gebildet, eine für Konfliktsituationen und eine ohne. Das Ergebnis war eine gute Annäherung an das vorgegebene Verhalten eines anderen Agenten. Der Grund dafür liegt in der Aufgabenstellung des hier untersuchten Szenarios. Der Agent kennt nur zwei übergeordnete Ziele. Dies sind zum einen das Umrunden der Statue zwecks Aufnahme von Munition, zum anderen die sofortige Auseinandersetzung, wenn ein anderer Agent auftaucht. Für den ersten Teil der

¹⁶In den Tests zuvor kam es dazu, dass der Agent gelegentlich seine Waffe betätigte, er für diese aber keine Munition mehr hatte.

5. Testphase und Evaluierung

Aufgabe benötigt er die Navigation. Hier hat sich in den anderen Testphasen die fast völlige Beschränkung auf die Attribute der Position als am besten erwiesen. Für den zweiten Teil werden vor allem die Attribute der vierten Zustandsklasse (Feind) benötigt. Die Imitation von Agentenverhalten innerhalb des vierten Szenarios ist durch die neue Variante der Zustandsermittlung gelungen.

5.3.1.5. Testphase 5

In der letzten Versuchsphase sollen anhand des fünften Szenarios untersucht werden, ob es mittels Clustern und Reinforcement Learning gelingt, Verhalten zu imitieren, welches nicht auf einer Vorgabe eines bestimmten Verhaltens beruht, sondern auf allgemeinen Spielprinzipien. Dies bedeutet Verhaltensmuster zu finden, wie „Suche und Aufnahme von Munition, wenn der Vorrat erschöpft ist“ oder „Aufspüren anderer Agenten und Treffer erzielen“.

Zu diesem Zweck wird das Zusammenspiel von Gewichtung der Attribute und Verhaltensimitation getestet (Test 5A+B). Es wird untersucht, wie das Ergebnis durch die in dieser Arbeit eingesetzten Methoden des Reinforcement-Learning beeinflusst wird (Test 5B) und ob durch das Lernen von zusätzlichen Zuständen Verhalten besser nachgeahmt werden kann (Test 5C).

Es wird wieder das modifizierte Belohnungssystem(nur Belohnung) angewandt und die überarbeitete Zustandsbewertung, die zwei übergeordnete Zustände(Feindkontakt oder nicht) definiert.

Test 5A:

Die letzte Testphase führt Tests anhand des komplexesten Szenarios Nummer 5 durch. In Test 5A wird zunächst die Gewichtung untersucht, welche sich für das vierte Szenario als am besten erwiesen hat. Das Lernen beschränkt sich auf das Clustern der Daten.

Szenario 5: Optimale Gewichtung der Attribute aus vierter Testphase ohne Reinforcement-Learning							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenom	25	Weighting(1)	0.95	Number of runs	-	Dist to Pos	78
Distancenom	2	Weighting(2)	0	Number of frags	-60	Dist to State	26
Modifier1	-	Weighting(3)	0.05	Actionprob.	12	Dist to See	0
Modifier2	-	Weighting(4)	-	Directionprob.	9	Dist to Enemy	-3
Maxdistfactor	-	Discountfactor	-	Dirfaithfuln.	-60	Number of Clust.	-
Anzahl Cluster: 223-33-28-23							

Tabelle 5.23.: Parameterbelegung und Ergebnisse für Test 5A

Hierdurch soll zum einen festgestellt werden, ob die gefundene Gewichtung der Attribute auch ausreichend für ein schwierigeres Szenario ist. Zum anderen wird noch einmal überprüft, wie es sich auf die Imitation von Agentenverhalten auswirkt, wenn nur durch Clustern der Daten gelernt wird.

Es werden insgesamt 307 Cluster gebildet. Die größten Abweichungen ergeben sich bei

5. Testphase und Evaluierung

der Einhaltung der letzten Aktionsrichtung, der Anzahl der erzielten Treffer und den Distanzen zu den Zuständen aus dem Clustern der Zustandsklasse Position.

Wird nur die Auswertung des nachahmenden Agenten betrachtet, so ergibt sich ein ähnlich gutes Ergebnis, wie in Test 4E. Ein deutlicher Abfall zum vierten Szenario hat sich nur bei der Anzahl der erzielten Treffer ergeben. Bei näherer Betrachtung fällt aber auf, dass die Abweichung in den Positionsdistanzen nicht so hoch ausfallen kann, wie in der vorherigen Testphase. Dies liegt daran, dass es im vierten Szenario Positionen in der Map gab, die von allen ermittelten Zuständen weiter entfernt waren, als dies im fünften Szenario der Fall ist¹⁷. Unter Einbeziehung dieses Aspektes kann nur von einer teilweise gelungenen Imitation gesprochen werden. Dies ist vor allem in der Beobachtungsphase zu sehen, in der wenig typisches Verhalten der oben erwähnten Spielprinzipien zu erkennen ist. Eine Ausnahme bildet lediglich die Reaktion auf die Wahrnehmung eines Agenten. Hier war bemerkbar, dass sich der Agent auf seinen Widersacher zubewegte und auf ihn zielte. Ansonsten bewegte sich der der Agent oftmals in eine Richtung, um sich durch die nächste Aktion wieder zurückzubewegen. Dieses Verhalten spiegelte sich auch in der Abweichung der Einhaltung der letzten Aktionsrichtung wider. Die Ursache für dieses Verhalten ist in der Möglichkeit für den zu kopierenden Agenten gegeben, sich innerhalb dieses Szenarios völlig frei zu bewegen. Daraus folgt, dass es in jedem Zustand zwölf Aktionsrichtungen geben kann, die von einem nachzuahmenden Agenten ungefähr gleichverteilt ausgewählt werden können. Dies war in den bisherigen Szenarien nicht der Fall, da der Agent hier einer festgelegten Route folgen sollte.

Test 5B:

Der nächste Test dient dazu, zu überprüfen ob mit einer leichten Modifikation der Gewichtung und gleichzeitigem Reinforcement-Learning eine Verbesserung der Imitation erzielt werden kann. Zu diesem Zweck werden die Parameter des Lernens, so gesetzt, wie

Szenario 5: Leichte Änderung der Gewichtung mit Lernen							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.8	Number of runs	-	Dist to Pos	67
Distancenom	2	Weighting(2)	0.1	Number of frags	-60	Dist to State	51
Modifier1	0.4	Weighting(3)	0.1	Actionprob.	18	Dist to See	0
Modifier2	0.2	Weighting(4)	-	Directionprob.	17	Dist to Enemy	0
Maxdistfactor	-	Discountfactor	8	Dirfaithfuln.	-54	Number of Clust.	4
Anzahl Cluster: 223-33-28-23							

Tabelle 5.24.: Parameterbelegung und Ergebnisse für Test 5B

sie in den bisherigen Testphasen erfolgreich waren. Bei der Änderung der Attributsgewichtungen greifen die Überlegungen, dass eine Aktion auch vermehrt in Abhängigkeit

¹⁷Wenn der Agent innerhalb eines Szenarios nur einen Teil einer Map durchläuft, dann sind alle Positionen in einem anderen Bereich der Map sehr weit entfernt von den gefundenen Zuständen. Durchläuft der zu imitierende Agent aber die gesamte Map, so gibt es bei genügend großer Anzahl von Clustern keine Zustände mehr, die so weit entfernt sind.

5. Testphase und Evaluierung

von Wahrnehmung und Status eines Agenten ist.

Die Analyse der Bewertungskriterien verraten keine Annäherung an das Verhalten eines nachzuahmenden Agenten. Die Werte sind ähnlich den Ergebnissen aus Test 5A.

Durch zusätzliches Reinforcement-Learning konnte in diesem Szenario keine Verbesserung der Verhaltensimitation erreicht werden. Die Beobachtung des imitierenden Agenten ergab, dass dieser sich nach wie vor hin und her drehte und sich nur langsam bewegte. Eine klare Richtung war dabei nicht zu erkennen. Lediglich beim Kontakt mit anderen Agenten, war wiederum eine eindeutige Reaktion zu bemerken. Der Agent drehte sich in die Richtung seines Gegenübers oder bewegte sich auf ihn zu und attackierte ihn.

Test 5C:

Einige weitere Versuche mit anderen Gewichtungen und Lernparametern brachten ebenfalls keine Besserung der Nachahmung. Das Resultat war stets das gleiche. Deshalb soll in diesem abschliessenden Test überprüft werden, ob durch das Lernen zusätzlicher Zustände eine Annäherung an einen zu imitierenden Agenten stattfindet. Dazu wird der Parameter *maxdistfactor* so gewählt, dass immer dann ein neuer Zustand gebildet wird, wenn die aktuelle Momentaufnahme größer ist als die maximale Distanz einer Momentaufnahme während des Clusters. Ziel ist es festzustellen, ob dadurch Zustände gelernt werden, in denen eindeutige Handlungen vorgegeben werden, wodurch eine Hin- und herbewegung ausgeschlossen würde. Die übrigen Einstellungen bleiben so wie in Test 5B.

Szenario 5: Bilden neuer Zustände bei zu hoher Distanz							
Parameter	Wert	Parameter	Wert	Bewertungskrit.	Diff.	Bewertungskrit.	Diff.
Distancenum	25	Weighting(1)	0.8	Number of runs	-	Dist to Pos	80
Distancenom	2	Weighting(2)	0.1	Number of frags	-60	Dist to State	38
Modifier1	0.4	Weighting(3)	0.1	Actionprob.	15	Dist to See	1
Modifier2	0.2	Weighting(4)	0	Directionprob.	18	Dist to Enemy	-4
Maxdistfactor	1	Discountfactor	8	Dirfaithfuln.	-55	Number of Clust.	18
Anzahl Cluster: 223-33-28-23							

Tabelle 5.25.: Parameterbelegung und Ergebnisse für Test 5C

Zusätzlich werden online 18 Zustände gelernt. Die Differenzen die in den Bewertungskriterien abzulesen sind, haben sich im Vergleich zu den anderen beiden Tests nur unwesentlich verändert. Es werden immer noch 60% weniger finaler Treffer erzielt und die Distanz zu den Zuständen der Klasse Position ist immer noch hoch in diesem Szenario. Auch mit dem Erlernen neuer Zustände konnte keine Steigerung der Resultate mehr erbracht werden. Die teilweise misslungene Imitation von Verhaltensmustern in diesem Szenario, ist daher nicht dadurch zu begründen, dass dem Agenten Zustände zugeordnet wurden, die nicht seiner aktuellen Momentaufnahme entsprochen haben. Auch ist es nicht gelungen, dadurch Zustände zu erlernen, die eindeutige Verhaltensmuster erkennen lassen.

Zusammenfassung Testphase5:

Das fünfte Szenario ist das erste, indem eine Imitation von Verhaltensmustern nur in Ansätzen gelungen ist. Dies konnte mit keiner der vorgenommenen Einstellungen der zur Verfügung stehenden Lernparameter verbessert werden.

Das einzige Verhaltensmuster, welches deutlich zu erkennen war, war in jedem Test, dass der Agent eine Reaktion auf den Kontakt mit anderen Agenten zeigte. Dies äußerte sich darin, dass er sich auf den Agenten zubewegte und ihn attackierte.

Nicht zu erkennen war dagegen ein Verhalten, welches die übrigen Spielprinzipien wiedergeben würde. Zu keiner Zeit waren eine Reihe von Aktionen zu vermerken, welche darauf hindeuten würden, dass Verhalten nachgeahmt wurde, wie die Suche und Aufnahme von Munition. Das größte Problem, das sich in allen Tests dabei herausstellte war, dass der imitierende Agent keine durchgängige Richtung in seinen Aktion fand. Dies führte dazu, dass er viel Zeit benötigte um sich von einem Punkt der Welt zum nächsten zu bewegen, wodurch schließlich keine klaren Handlungen erkennbar waren.

6. Fazit und Ausblick

An dieser Stelle werden noch einmal die Ergebnisse dieser Arbeit zusammengefasst. Dazu werden in Kapitel 6.1 die Resultate aus der Testphase und Vor- und Nachteile der verwendeten Lernverfahren in dieser Arbeit betrachtet. Zum Abschluß werden in Kapitel 6.2 Anregungen gegeben, wie sich die Ergebnisse in dieser Arbeit möglicherweise verbessern lassen.

6.1. Fazit

Die Imitation von Verhaltensmustern kann in den Tests zu den Szenarien 1-4 als gelungen angesehen werden (siehe Kapitel 5.3). Die Tests zum letzten Szenario haben dagegen ergeben, dass mit steigender Komplexität der Aufgabenstellung innerhalb eines Szenarios, die Qualität der Nachahmung von Verhalten sinkt. Im folgenden wird dazu untersucht, wodurch ein Scheitern oder Gelingen einer Verhaltensimitation in dieser Arbeit abhängig ist. Folgende Kriterien werden dabei in Betracht gezogen:

- Attribute
- Zustände
- Reinforcement-Learning

Attribute:

Bezüglich der Auswahl der Attribute kann das Resultat der Testphase so gedeutet werden:

Werden lediglich die Aktionen eines Agenten betrachtet und nicht sein Verhalten, so lassen sich durch die beiden Attribute der Handlung *action_direction* und *action_lastmove* Aktionen eines Agenten wiedergeben. Hierbei sind allerdings die gemachten Einschränkungen zu Quake3 aus Kapitel 2.1.5 zu beachten. Ein Problem ergab sich allerdings besonders in der dritten Testphase, da das Attribut *action_direction* die tatsächliche Aktionsrichtung des Agenten nicht hundertprozentig wiedergibt. Hier kam es dazu, dass der Agent auf ein Hindernis aufließ und sich ohne online zu Lernen nicht mehr aus dieser Situation befreien konnte.

Die Testphase 1-3 haben gezeigt, dass für die Navigation von einem Punkt zu einem anderen zusätzlich zu den Handlungsattributen nur die Positionsangaben eines Agenten benötigt werden. Hier können für die Imitation von Agentenverhalten alle anderen Attribute vernachlässigt werden.

Die Verhaltensimitation in Bezug auf die Interaktion mit anderen Agenten wurde in der

6. Fazit und Ausblick

Testphase 4 und 5 überprüft. Hier war zu beobachten, dass die Attribute der Wahrnehmung, welche sich auf den Kontakt mit anderen Agenten beziehen, dafür sorgte, dass Kampfverhalten nachgeahmt werden konnte.

Attribute, die sich auf den Status eines Agenten beziehen, haben in den Tests zu keiner Verbesserung der Imitation geführt.

Zustände:

Die Ermittlung von Zuständen durch Clustern, hat sich in dieser Arbeit als praktikabel erwiesen. Dies zeigen die ersten vier Testphasen. Bei der Wahl eines geeigneten Verfahrens, ergab sich das Problem, dass alle vorgestellten Methoden zum Clustern von Daten, Nachteile hatten. Dieser war beim ausgewählten Verfahren, die Vorgabe der Anzahl der Cluster. Hier hat sich gezeigt, dass es unter Kenntnis von bestimmten Zusammenhängen aus den gewonnenen Daten über Agentenverhalten, möglich war die Anzahl der Cluster abzuschätzen. Für die Testphase hat sich hier als positiv herausgestellt, dass es die Möglichkeit gab, mittels zweier Parameter *distancenum* und *distancenom*, Einfluss auf diese Zahl nehmen zu können. Hierdurch konnte eine Clusteranzahl gefunden werden, die für die ersten vier Testphasen zum Gelingen der Imitation beitrug.

Ein Problem, das nicht für alle Szenarien gelöst werden konnte, ist die Definition von Zuständen. Die ursprüngliche Idee war es, einen Zustand immer durch alle Attribute zu beschreiben. Wie aber verschiedene Tests gezeigt haben, ergaben sich die schlechtesten Ergebnisse, bezüglich der Imitation von Verhalten, immer dann, wenn ein Zustand aus allen Attributen bestand. Um einen Zustand nur durch eine bestimmte Anzahl von Attributen ausdrücken zu können, wurden Zustandklassen eingeführt. Diese führten dazu, dass es in allen Szenarien, bei denen es nur auf die Navigation eines Agenten ankam, die Imitation gelang. Zu unflexibel erwiesen sich diese Zustandklassen aber dann, wenn, wie im letzten Szenario, ein komplexeres Verhalten erlernt werden sollte. Zusammenhänge, wie der Agent hat keine Munition mehr und sieht in einer bestimmten Richtung Munition, so bewegt er sich dorthin, konnten nicht entdeckt werden.

Reinforcement-Learning:

Die erlernte Steuerung eines Agenten konnte durch zusätzliches Reinforcement-Learning in einigen Tests verbessert werden.

Als notwendig erwies sich Reinforcement-Learning immer dann, wenn die Steuerung aus dem Clustern dazu führte, dass der Agent in „ausweglose“ Situationen geriet. Hier konnte online erlernt werden, einen solchen Zustand zu vermeiden, bzw. diesen zu verlassen. Ebenso nützlich war es, online neue Zustände zu bilden, wenn die vorgegebenen Zustände nicht ausreichend waren, die Momentaufnahmen eines nachahmenden Agenten online zu repräsentieren. Hier führten neue Cluster dazu, dass der Agent lernte, in erwünschte Zustände zurückzugelangen.

Als negativ stellte sich heraus, den Agenten lernen zu lassen, indem gleichzeitig Aktionen bestraft und belohnt wurden. Dies führte dazu, dass der nachahmende Agent, anstatt die Qualität der Imitation zu erhöhen, ab einer gewissen Lernzeit, keine Ähnlichkeit mehr zu dem zu imitierenden Verhalten aufwies. Eine Verbesserung des Lernergebnisses konnte erzielt werden, indem auf die Bestrafung von Aktionen verzichtet wurde.

Für die Testphase war es problematisch, dass die Steuerung eines Agenten nicht deter-

ministisch war. Dies hatte zur Folge, dass sehr viele Tests nötig waren, da jeder Test unterschiedliche Ergebnisse brachte. Ein zweiter Nachteil der sich ergab, weil der Agent in jedem Zustand eine andere Aktion auswählen konnte, ist, dass sich der Agent dadurch in einem Zustand abwechselnd in zwei entgegengesetzte Richtungen bewegen konnte. Das Ergebnis hieraus ist ein Agent, der sich hin und her bewegt. Dieses Verhalten kam besonders im Test zum fünften Szenario zum tragen, da der Agent hier in jedem Zustand die meisten Aktionsmöglichkeiten hatte.

Ein großes Problem des Reinforcement-Learnings ist die Markov-Eigenschaft. Das Verhalten eines zu imitierenden Agenten zeichnet sich u.a. dadurch aus, dass er eine Aktion, in Abhängigkeit von zuvor gemachten Erfahrungen, ausführt. Im Gegensatz dazu wird beim Q-Lernen eine Aktion nur durch den aktuellen Zustand bewertet. Dies führt dazu, dass zum einen der Agent, sich wie oben beschrieben hin und her bewegte, zum anderen, dass es mit, den hier beschriebenen Lernverfahren, nicht möglich war, ein komplexeres Verhalten zu lernen. D.h., wenn ein zu imitierender Agent bei seinen Aktionen einem bestimmten Plan gefolgt ist, dann konnte dieser mittels der benutzten Lernmethode nicht gefunden werden, da dieser nicht nur vom momentanen Zustand abhängt. Dieses Problem sollte bewältigt werden, indem zum einen ein Parameter *discountfactor* eingesetzt wurde, der auch zurückliegende Aktionen bestrafte und belohnte, zum anderen ein Parameter *directionfaithfulness*. Dieser zweite Parameter wurde eingeführt, weil in einem in dieser Arbeit nicht aufgeführten Testlauf, die Qualität des Verhaltensimitation im fünften Szenario erheblich schlechter war. So gelang es hierdurch, das oben beschriebene Problem, der „hin und her-“Bewegung etwas zu verkleinern. Die Einführung des *discountfactor's* brachte in einigen Tests eine leichte Verbesserung der Verhaltensimitation.

6.2. Ausblick

Den größten Teil des Lernens in dieser Arbeit übernimmt das Clustermodul. Die hier gelernten Zustände und Aktionswahrscheinlichkeiten bilden die Basis für das Reinforcement-Learning. Aus diesem Grund steckt das größte Potential zur Verbesserung der Imitation von Verhaltensmustern im Bilden von Zuständen. Da durch diese Arbeit bereits einiges an Vorarbeit geleistet worden ist¹, fällt die Testphase etwas kürzer aus. Aus diesem Grund kann versucht werden, mittels einer der Clusteralgorithmen mit einer höheren Laufzeit, zu testen, ob ein besseres Ergebnis erzielt werden kann.

Ein Problem, das ebenfalls gelöst werden muss, ist die sinnvolle Definition von Zuständen, da sich die Einteilung in vier Zustandsklassen als zu unflexibel erwiesen hat. Hier kann überlegt werden, ob die Definition von zusätzlichen Zustandsklassen, hilfreich bei der Imitation von Agentenverhalten sein kann. Beispielweise wäre eine Zustandsklasse denkbar, die die Attribute *state_health*, *see_dir_health* und die Position des Agenten beinhaltet. Damit könnte ein Verhalten, besser als in dieser Arbeit, abgebildet werden, wie:

¹Es wurden Attribute definiert, ein Lernverfahren implementiert und eine Steuerung in Quake3 eingebaut, die es ermöglicht anhand von gelernten Zuständen und Aktionswahrscheinlichkeiten Verhalten zu imitieren. Diese Komponenten sind lauffähig und müssen nicht mehr getestet werden

6. Fazit und Ausblick

„Der Agent hat einen niedrigen Gesundheitszustand, sieht einen Health-Bonus und bewegt sich auf diesen zu.“ Noch besser wäre es, wenn eine Möglichkeit entwickelt werden würde, die diese Zusammenhänge automatisch in den aufgezeichneten Daten findet und Cluster bildet, die nur diese Attribute enthalten. Dadurch könnte auf das Definieren von Zustandsklassen verzichtet werden.

Das größte Potential für Verbesserung der Imitation von Agentenverhalten im Bereich des Reinforcement-Learnings liegt darin die Markov-Eigenschaft zu umgehen. Dies bedeutet, es muss eine Möglichkeit implementiert werden, die es dem Agenten erlauben, Pläne zu lernen. D.h. eine Aktion wäre nicht nur von einem aktuellen Zustand abhängig, sondern von einem Plan, welcher in einem vorherigen Zustand definiert worden ist. Hierbei kann davon ausgegangen werden, dass die Nachahmung von Verhalten so erheblich besser funktioniert. Der Grund dafür ist, dass ein Avatar, dessen Verhalten kopiert werden soll, genauso agiert².

Insgesamt besteht also noch sehr viel Spielraum, die Imitation von Agentenverhalten in Quake3, mit den hier vorgestellten Ansätzen, zu verbessern.

²Auch der implementierte Bot in Quake3 benutzt ein System, dass auf Plänen beruht (Siehe kurz- und langfristige Ziele in Kapitel 2.2.3).

A. Anhang

Im ersten Teil dieses Anhangs werden die aus Kapitel 3, 4 und 5 gewonnenen Dateien abgebildet. Dazu wird jeweils ein Beispiel ausgewählt. Anschließend werden die implementierten Funktionen für die Datengewinnung in Quake3, das Clustermodul, das Reinforcement-Modul und die Steuerung für den modifizierten Agenten erklärt. Im letzten Teil wird dargestellt, welche Schritte in dieser Arbeit nötig sind, von der Installation der benötigten Komponenten, der Aufzeichnung der Daten, bis hin zum lernenden Agenten. Dazu wird auf den Umgang mit Quake3 und die Bedienung des Clustermoduls eingegangen.

A.1. Dateien

A.1.1. LOG-Datei

Eine Log-Datei dient der Datengewinnung aus Kapitel 3. Die Aufzeichnung einer Mo-

```
0:42 name: ULI
0:42 posX: 1089.407959 posY: 1119.143555
0:42 health: 100 armor: 100
0:42 current weapon: machinegun ammo: 100
0:42 viewangles horizontal: 109.138184
0:42 action: standing direction: NNW
+++++

0:42 name: ULI
0:42 posX: 1079.126465 posY: 1152.175293
0:42 health: 100 armor: 100
0:42 current weapon: machinegun ammo: 100
0:42 viewangles horizontal: 112.917480
0:42 see ammo ammo.bullets at position N
0:42 action: forward direction: N
+++++

0:43 name: ULI
0:43 posX: 1067.398804 posY: 1182.050537
0:43 health: 100 armor: 100
0:43 current weapon: machinegun ammo: 100
0:43 viewangles horizontal: 126.776733
0:43 see ammo ammo.bullets at position N
0:43 see health item.health_large at position NOO
0:43 has item ammo.bullets in visor
0:43 action: forward direction: N
+++++
```

Abbildung A.1.: Auszug aus einer Log-Datei

mentaufnahme beginnt immer mit dem Namen des zu imitierten Agenten. Es folgen die Attribute des Zustands, optional die Attribute der Wahrnehmung und die letzten Aktionen und Richtungen des Agenten. Eine Zeile mit Pluszeichen dient der Trennung von zwei Momentaufnahmen.

A.1.2. ARFF-Datei

Eine ARFF-Datei wird zur Strukturierung der Daten aus der Aufzeichnung eines Agenten in Quake3 benutzt. Sie besteht aus zwei Teilen, einem Kopf, der alle relevanten Informationen über die Datenmenge enthält und einem Block für die Daten. Die Abbildungen A.2 und A.3 zeigen diesen beiden Blöcke.

Eine ARFF-Datei ist folgendermaßen aufgebaut. Zeilen die mit % beginnen sind Kommentare, Zeilen, welche mit @ beginnen, markieren wichtige Abschnitte innerhalb der Datei. *@relation* ist der Name der Datensammlung. In *@attributes* werden die Attributnamen vergeben und gleichzeitig der Typ festgelegt. Handelt es sich um ein numerisches Attribut so erfolgt dies durch die Typbezeichnung *real*, handelt es sich um ein nominales Attribut, so werden die möglichen Werte direkt angegeben. Der Teil, welcher mit *@data* beginnt, listet alle Instanzen mit ihren Attributwerten in genau der gleichen Reihenfolge auf, wie dies durch die Attribute definiert wurde. So ist z.B. der Wert für das Attribut *pos.y* immer an der zweiten Stelle jeder Zeile im Datenblock. Eine Zeile bildet genau alle Werte einer Instanz.

```
@data
450.760 2338.129 124 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward N
482.352 2335.450 124 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward N
514.021 2330.617 123 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward NNO
544.970 2323.059 123 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward NNO
575.501 2314.404 122 0 100 not_visible N not_visible NNO N nothing not_visible not_visible nobody nothing not_hit forward N
606.650 2307.313 122 0 100 not_visible N not_visible NNO N nothing not_visible not_visible nobody nothing not_hit forward N
637.940 2300.981 121 0 75 not_visible N SW NNO N item_health not_visible not_visible nobody nothing not_hit forward N
669.335 2296.047 121 0 100 not_visible N SW NNO NNO item_health not_visible not_visible nobody nothing not_hit forward N
690.453 2108.721 120 50 0 100 SO S NNW not_visible W nothing not_visible not_visible nobody nothing not_hit standing W
697.446 2119.206 50 0 100 SO SW not_visible W not_visible nothing not_visible not_visible nobody nothing not_hit standing W
328.775 1892.710 50 15 100 SO SW O not_visible W weapon_shotgun not_visible not_visible nobody nothing not_hit standing O
329.240 1892.137 50 15 100 S SW O not_visible W weapon_shotgun not_visible not_visible nobody nothing not_hit standing O
329.240 1892.137 50 15 100 S SW O not_visible not_visible weapon_shotgun not_visible not_visible nobody loud not_hit standing O
329.240 1892.137 50 15 100 S SW O not_visible not_visible weapon_shotgun not_visible not_visible nobody loud not_hit standing O
329.240 1892.137 50 15 100 S S NOO not_visible not_visible nothing not_visible not_visible nobody loud not_hit standing NOO
329.240 1892.137 50 15 100 S S NOO not_visible not_visible nothing not_visible not_visible nobody loud not_hit standing NOO
329.240 1892.137 50 15 100 S W NNO NNO not_visible nothing NNO NNO nobody loud not_hit standing NNO
329.240 1892.137 50 15 100 S W not_visible NNO NNO nothing NNO NNO nobody loud not_hit standing NNO
329.240 1892.137 50 15 100 not_visible NNW not_visible N N nothing not_visible not_visible nobody loud not_hit standing N
329.240 1892.137 50 15 100 not_visible NNW NNW N N nothing not_visible not_visible nobody loud not_hit standing N
450.760 2338.129 124 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward N
482.352 2335.450 124 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward N
514.021 2330.617 123 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward NNO
544.970 2323.059 123 0 100 not_visible N N NNO N nothing not_visible not_visible nobody nothing not_hit forward NNO
575.501 2314.404 122 0 100 not_visible N not_visible NNO N nothing not_visible not_visible nobody nothing not_hit forward N
368.775 1892.710 50 15 100 SO SW O not_visible W weapon_shotgun not_visible not_visible nobody nothing not_hit standing O
369.240 1892.137 50 15 100 S SW O not_visible W weapon_shotgun not_visible not_visible nobody nothing not_hit standing O
369.240 1892.137 50 15 100 S SW O not_visible not_visible weapon_shotgun not_visible not_visible nobody loud not_hit standing O
369.240 1892.137 50 15 100 S SW O not_visible not_visible weapon_shotgun not_visible not_visible nobody loud not_hit standing O
369.240 1892.137 50 15 100 S S NOO not_visible not_visible nothing not_visible not_visible nobody loud not_hit standing NOO
369.240 1892.137 50 15 100 S S NOO not_visible not_visible nothing not_visible not_visible nobody loud not_hit standing NOO
369.240 1892.137 50 15 100 S W NNO NNO not_visible nothing NNO NNO nobody loud not_hit standing NNO
```

Abbildung A.2.: Datenblock einer Arff-Datei

A. Anhang

```
% 1. Title: DataMining with Quake
%
% 2. Sources:
% Quake-Log-Datei
%
% 3. Past Usage:
%
% 4. Relevant Information:
% - Number of Watches Items: 0
% - Number of Hears: 0
% - Number of Moves: 2
% action standing
% action forward
%
% 5. Number of Instances: 720
%
% 6. Number of Attributes: 18, 13 categorical and 5 numeric
%
% 7. Attribute Information:
% 1. state_pos_x: player's position (x-coordinate)
% 2. state_pos_y: player's position (y-coordinate)
% 3. state_health: player's health
% 4. state_armor: player's armor
% 5. state_munition: player's ammo for current weapon
% 6. see_wall: wall if player in front of wall,else no
% 7. see_dir_health: player's direction if visible
% 8. see_dir_weapon: player's direction if visible
% 9. see_dir_ammo: player's direction if visible
% 10. see_dir_armor: player's direction if visible
% 11. watch_item: item in sight
% 12. see_dir_bot: player's direction if visible
% 13. see_dir_enemy: player's direction if visible
% 14. watch_player: somebody if player in sight,enemy if enemy, else nobody
% 15. hear_noise: noise and volume if player hears something
% 16. action_hit_from: direction to shooter
% 17. action_last_move: player's last move
% 18. action_direction: player's direction

@relation 'Quake'

@attribute state_pos_x real
@attribute state_pos_y real
@attribute state_health real
@attribute state_armor real
@attribute state_munition real
@attribute see_wall { not_visible, wall }
@attribute see_dir_health { not_visible, N, NNO, NNW }
@attribute see_dir_weapon { not_visible, N, NNO, NNW }
@attribute see_dir_ammo { not_visible, N, NNO, NNW }
@attribute see_dir_armor { not_visible, N, NNO, NNW }
@attribute watch_item { nothing }
@attribute see_dir_bot {not_visible, N, NNO, NNW }
@attribute see_dir_enemy { not_visible, N, NNO, NNW }
@attribute watch_player { somebody, nobody, enemy }
@attribute hear_noise { nothing }
@attribute action_hit_from { not_hit, N, NNO, NNW }
@attribute action_last_move { standing, forward }
@attribute action_direction { NNO, N, NOO, SOO, SSO, S, SSW, SWW, W, NWW, NNW }
```

Abbildung A.3.: Kopf einer Arff-Datei

A.1.3. CLU-Datei

Die Ergebnisse aus der Berechnung des Clustermoduls (siehe Kapitel 4.2.1.4) werden in einer CLU-Datei gespeichert. Der erste Wert unter „MINMAX“ entspricht dem Mi-

```

MINMAX
212.000000 1136.874268
602.004028 2376.875000
100.000000 125.000000
0.000000 75.000000
100.000000 100.000000
10000.000000 0.000000
10000.000000 0.000000
.
.
.
POS
264.842575 807.998719 1
393.679214 648.184631 1
663.917090 1274.662256 1
349.132985 1931.253459 1
.
.
STATE
116.500000 20.333333 100.000000 1
100.518519 49.361111 100.000000 1
.
.
SEE
not_visible N NNO N not_visible nothing not_visible N N 1
not_visible NNW item_health_large not_visible N not_visible N 1
not_visible ammo_bullets not_visible N not_visible NNO N 1
.
.
ENEMY
not_visible not_visible nobody nothing not_hit 1
.
.
DISTANCES
POS 0.001173 0.093001 0.021876
STATE 0.000000 0.328303 0.089912
SEE 1.000000 1.732051 1.587133
ENEMY 1.000000 1.732051 1.134249
AVERAGEPROBABILITIES 0.622622 0.255527 0.121851 0.136247
0.235990 0.092545 0.022108 0.013882 0.022622 0.051928 0.017995 0.032391 0.056041 0.165553
0.152699
DIRECTIONFAITHFULNESS 0.812853
NumberOfClusterOLD 40 13 20 22
NumberOfClusterNEW 40 13 20 22

```

Abbildung A.4.: Auszug einer Datei zur Speicherung der Cluster

nimum des Attributes der zweite dem Maximum. Minimum und Maximum werden bei nominalen Werten auf 10000 bzw. 0 gesetzt, da sie für die Berechnung nicht benötigt werden. Anschließend folgen die Attributwerte nach Zustandsklassen eingeteilt, in der Reihenfolge, wie sie auch in Tabelle 4.1 in Kapitel 4.2.1.3 zu sehen sind. Jede Zeile entspricht dabei genau einem Cluster, wobei der letzte Wert angibt, ob es sich um einen

„erwünschten“ oder „unerwünschten“ Zustand handelt.

Der letzte Teil einer solchen Datei enthält wichtige Informationen für das Reinforcement-Learning. Dies sind für jede Zustandsklasse für die Bewertung von Zuständen die minimalen, maximalen und durchschnittlich errechneten Distanzen während des Clusters. Für die Bildung neuer Cluster werden die durchschnittlichen Aktionswahrscheinlichkeiten gespeichert und für die Imitation von Agentenverhalten die Wahrscheinlichkeit für die Beibehaltung einer Aktionsrichtung. Die letzten beiden Informationen werden im Zusammenhang mit dem Status eines Clusters benötigt. Sie geben an, wieviele Zustände im Clustermodul gefunden und wieviele Zustände durch das Online-Lernen insgesamt gebildet wurden.

A.1.4. RLQ-Datei

Eine RLQ-Datei enthält die Wahrscheinlichkeiten für Aktionen und Richtungen. Sie wird durch das Clustermodul initialisiert und während des Online-Lernens modifiziert. Zwei Zeilen stehen dabei für die Wahrscheinlichkeiten eines Clusters. Die Reihenfolge der Cluster ist dabei die gleiche, wie in der zugehörigen CLU-Datei. Zeilen in denen 3 Werte gespeichert sind, geben die Wahrscheinlichkeiten für die Aktionen *standing*, *forward* und *attack* wieder, die übrigen Zeilen, die Wahrscheinlichkeit für die Richtung der auszuführenden Aktionen in der Reihenfolge *NNO,NOO,O,SOO,SSO,S,SSW,SWW,W,NWW,NNW,N*.

```

POS
0.00000 1.00000 0.00000
0.00000 1.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 1.00000 0.00000
0.50000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.50000
0.80000 0.20000 0.00000
0.00000 0.00000 0.00000 0.40000 0.40000 0.20000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 1.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.33333 0.66667 0.00000 0.00000 0.00000 0.00000
STATE
0.05556 0.94444 0.00000
0.17778 0.05556 0.02222 0.05556 0.02778 0.22778 0.02778 0.03333 0.05000 0.02222 0.04444 0.25556
0.04444 0.95556 0.00000
0.07526 0.04696 0.05304 0.05556 0.05556 0.23737 0.05704 0.06263 0.06819 0.03737 0.05407 0.19696
SEE
0.05000 0.95000 0.00000
0.11785 0.04107 0.04642 0.07142 0.05714 0.20178 0.04285 0.05178 0.06428 0.03928 0.06964 0.19642
0.06250 0.93750 0.00000
0.00000 0.12500 0.00000 0.00000 0.00000 0.25000 0.06250 0.00000 0.00000 0.00000 0.00000 0.56250
ENEMY
0.04722 0.95277 0.00000
0.10138 0.04861 0.04583 0.05556 0.04861 0.23472 0.04722 0.05556 0.06388 0.03333 0.05416 0.21111
    
```

Abbildung A.5.: Auszug einer Datei zur Speicherung der Aktionswahrscheinlichkeiten

A.1.5. TST-Datei

Eine TST-Datei wird erzeugt, wenn in der Testphase die Lernergebnisse eines Agenten in Quake3 gesichert werden sollen.

Der erste Teil einer solchen Datei enthält alle Resultate für die Beurteilung der qualitativen Bewertungskriterien. Dies sind die Differenzen der minimalen, maximalen und durchschnittlichen Distanzen in jeder Zustandsklasse(Distances), die durchschnittlichen

A. Anhang

Wahrscheinlichkeiten für Aktionen oder Richtungen (Averageprobabilities) und der Vergleich der Einhaltung der letzten Aktionsrichtung (Directionfaithfulness). Werte hinter

```
RESULTS  
DISTANCES  
POS  
Old 0.050503 0.585164 0.294772  
New 0.053789 0.626393 0.452571  
Dif 0.003286 0.041229 0.157799  
AverageDif 0.535327  
STATE  
Old 0.148955 1.012080 0.324106  
New 0.289872 0.849567 0.349525  
Dif 0.140917 -0.162513 0.025419  
AverageDif 0.078429  
SEE  
Old 1.000000 2.000000 1.500503  
New 1.000000 2.000000 1.566929  
Dif 0.000000 0.000000 0.066426  
AverageDif 0.044269  
ENEMY  
Old 1.414214 1.414214 1.414214  
New 1.414214 1.414214 1.414214  
Dif -0.000000 -0.000000 -0.000000  
AverageDif -0.000000  
AVERAGEPROBABILITIES  
Old 0.014365 0.985635 0.000000  
New 0.024249 0.975751 0.000000  
Dif -0.009884 0.009884 0.000000  
MaxDif 0.009884  
Old 0.07292 0.07292 0.1303 0.10276 0.05635 0.05524 0.09834 0.14917 0.07403 0.06408 0.06187 0.06187  
New 0.08583 0.07274 0.14857 0.11277 0.06235 0.05966 0.09391 0.12663 0.05927 0.06620 0.05619 0.05581  
Dif -0.01297 0.00018 -0.01818 -0.01001 -0.00600 -0.00441 0.00442 0.02253 0.01475 -0.00211 0.00568 0.00606  
MaxDif 0.022535  
DIRECTIONFAITHFULNESS 0.678453 0.127021 -0.551432  
Diff Directionfaithfulness -0.812779  
PARAMETERS  
Weighting 0.250000 0.250000 0.250000  
Discountfactor 8.000000  
Modifier1 0.200000  
Modifier2 0.000000  
Divergencefactor 10.000000
```

Abbildung A.6.: Beispiel für eine Datei zur Speicherung der Lernergebnisse

„Old“ geben den Wert nach dem Clustern an, Werte hinter „New“ den Wert nach der Beobachtung des Agenten in Quake3 und Werte hinter „Diff“ die Abweichung zwischen Old und New. Die in Abbildung A.6 unterstrichenen Werte sind diejenigen, welche in der Testphase als Grundlage für die Bewertung gelten.

Der zweite Teil listet die Parameter für das Reinforcement-Learning und ihre Einstellungen auf.

A.2. Funktionen

Für die Realisierung der Imitation von Agentenverhalten mittels Reinforcement-Learning waren einige Änderungen am Source-Code des Originalprogramms nötig. Dazu wurden Funktionen implementiert, um die in Kapitel 3 beschriebenen Daten aus Quake3 zu extrahieren, einen modifizierten Agenten in Quake3 zu steuern und ihn, wie in Kapitel 4.2.2, online lernen zu lassen. Die Funktionen des Clustermoduls beziehen sich auf ein eigenständiges Programm.

Die Implementierungen sind alle in der Programmiersprache C und werden im folgenden beschrieben.

A.2.1. Funktionen für die Datengewinnung

Die Funktionen für die Datengewinnung sorgen dafür, dass die Momentaufnahmen eines nachzuziehenden Agenten, in Form der in Kapitel 3.1 beschriebenen Attribute, in eine Log-Datei geschrieben werden.

Output Hauptfunktion für die Steuerung der Ausgabe der Attribute eines zu imitierenden Agenten. Der Aufruf geschieht zehnmal pro Sekunde oder unmittelbar nach dem der Agent geschossen oder getroffen wurde. Es werden die Attribute des Zustands ausgegeben und weitere Funktionen der Datengewinnung aufgerufen.

OutputInFrontOfWall In Abhängigkeit einer vorgegebenen Distanz wird ausgegeben, wenn der Agent sich vor einem Hindernis befindet.

OutputSeeEntities Dynamische Objekte und ihre relative Position zum Agenten werden in die Log-Datei geschrieben, wenn sie in der aktuellen Situation vom Agenten gesehen worden sind.

OutputNoise Befindet sich in hörbarer Nähe ein anderer Agent, so wird dies unter Angabe der Lautstärke in der Log-Datei vermerkt.

OutputMovement Es werden sowohl die letzte Aktion, als auch die letzte Aktionsrichtung des Agenten ausgegeben.

OutputHit Ergänzung der Ausgabe für den Fall, dass der zu imitierende Agent von einem anderen Agenten getroffen worden ist.

OutputShoot Zusätzliche Ausgabe der Attribute, in dem Moment, in dem der Agent einen Schuss abfeuert.

A.2.2. Funktionen des Clustermoduls

Die Arbeitsweise des Programms zur Analyse der Daten und Clusterbildung entspricht der Vorgehensweise aus Kapitel 4.2.1.

Cluster Die Hauptfunktion berechnet die Anzahl der Cluster, ordnet Instanzen den Clustern zu und speichert alle Ergebnisse.

NewCluster Bildung eines neuen zufälligen Clusters wie in Kapitel 4.2.1 beschrieben.

OutputCluster Ausgabe der aktuell ermittelten Cluster am Bildschirm.

Kmeans Ermittlung der Minimas und Maximas der Attributwerte und Start der Clusterberechnung für jede Zustandsklasse.

ReadInfo Auslesen der Informationen aus der Log-Datei.

BuildArff Strukturierung der Daten aus der Log-Datei und Speicherung im Arff-Format.

A.2.3. Funktionen des Reinforcement-Learning-Moduls

Zu den Funktionen des Reinforcement-Learning-Moduls gehören die Funktionen zur Beurteilung von Aktionen und Korrektur der Steuerung des lernenden Agenten.

QLearn Hauptfunktion des in Kapitel 4.2.2 definierten Lernvorgangs. Es wird der aktuelle Zustand bewertet und je nach Ergebnis ein neuer Zustand gebildet oder die Steuerung des Agenten angepasst.

NewCluster Definition eines neuen Clusters, wie in Kapitel 4.2.2.4 erklärt.

UpdateSnapshotNext Speicherung der letzten 10 Aktionen.

UpdateSnapshotBest Speicherung der letzten 10 Zustände und der dazugehörigen kleinsten Distanzen.

UpdateSnapshotPos Speicherung der letzten 10 Positionen.

CorrectProbability Anpassung der Aktionswahrscheinlichkeiten.

CorrectRestProbs Korrektur der übrigen Wahrscheinlichkeiten in einem Cluster, so dass sich nach einer Änderung der Aktionswahrscheinlichkeiten wieder eine Gesamtwahrscheinlichkeit von 1 oder 100%, ergibt.

Learn Start des Lernvorgangs und Möglichkeit zur Änderung der Lernparameter durch den Benutzer.

SaveLearn Sicherung des Lernerfolges durch das Speichern aller Änderungen in den Dateien für die Cluster und Wahrscheinlichkeiten. Für die Evaluierung werden zusätzliche Informationen in einer Testdatei gesichert.

A.2.4. Funktionen der Steuerung

Diese Funktionen dienen dazu, für einen modifizierten Agenten aus den oben beschriebenen Dateien eine Steuerung einzulesen und in Aktionen für den Agenten umzusetzen.

InputControl Initialisierung des modifizierten Agenten. Hierzu gehören das Initialisieren der Steuerung, der Lernparameter und der Bewertungsvariablen. Für die Steuerung werden dazu die erforderlichen Daten aus Clu- und Rlq-Datei gelesen und Anschließend zur Überprüfung der Korrektheit in die Log-Datei ausgegeben.

InputCluster Einlesen der Cluster aus der Clu-Datei.

InputRLMove Ausführung einer Aktion und eventuellen Richtungswechseln des Agenten nach der modifizierten Steuerung.

NextAction Ermittlung des derzeitigen Zustands des Agenten und Wahl der nächsten Aktion.

NextDirection In Abhängigkeit des Zustands und der Wahrscheinlichkeiten wird eine Richtung ausgewählt, in die sich der Agent drehen soll.

UpdateCluster Für alle Attribute werden in einem gesonderten Cluster die aktuellen Werte abgelegt, um sie mit den bekannten Zuständen vergleichen zu können.

DoRandomMove Ausführung eines zufälligen Richtungswechsels des Agenten.

CalculateProbabilities Berechnung der Aktionswahrscheinlichkeiten aus den Wahrscheinlichkeiten der Zustände der vier Zustandsklassen mit eingestellter Gewichtung.

Load Start des Einlesens einer Steuerung durch Angabe des Dateinamens.

A.2.5. Gemeinsame Funktionen

Gemeinsame Funktionen, sind solche die von mehreren, der hier erwähnten, Programmstücke benutzt werden.

G_LogPrintf Eine Zeichenkette wird in die Log-Datei geschrieben.

G_Printf Eine Zeichenkette wird in Quake3 auf dem Bildschirm ausgegeben.

Normalize Normalisierung numerischer Attribute für die Distanzberechnung.

CalculateDifference Berechnung der Differenz zwischen zwei Attributwerten.

CalculateClusterDistance Ermittlung der Distanz zwischen zwei Instanzen oder einer Instanz und einem Cluster.

See Funktion, um aus der aktuellen Momentaufnahme eines Agenten die Werte für die Attribute der Wahrnehmung zu liefern.

InFrontOfWall Überprüfung der Entfernung eines Agenten zu einem Hindernis.

HearNoise Ermittlung von Geräuschen anderer Agenten, die sich in hörbarer Nähe befinden.

AllInFieldOfVision Alle Entities innerhalb des Blickfelds des Agenten, sowohl sichtbare als auch verdeckte Objekte (z.B. durch Wände) werden zurückgegeben.

FindRadius Ermittlung aller Entities, die sich in einem vorgegebenen Radius um den Agenten befinden.

IsVisible Berechnung aller tatsächlich sichtbaren Entities.

Trace Erzeugung eines imaginären Strahls, um zu überprüfen, ob sich zwischen Agent und Objekt ein Hindernis befindet.

CalculateOrigin Berechnung der Position von Objekten.

CorrectOrigin Korrektur der Position für einige Objekte, wie z.B. Türen.

CalculateDistance Messung der Entfernung zwischen zwei Objekten.

SetDirection Initialisierung der Aktionsrichtungen.

GetDirection Umwandlung der Blickrichtung des Agenten von Grad in eine der Himmelsrichtungen.

GetDirectionValue Umwandlung zurück in eine Gradzahl.

GetDirectionIndex Ermittlung des Indexes einer Himmelsrichtung aufgrund der Initialisierung der Aktionsrichtungen.

ActionIndex Index der Aktionen wird zurückgegeben.

GetAmmoPercentage Umwandlung der absoluten Munition der aktuellen Waffe in Prozent.

A.3. Bedienung

A.3.1. Installation

Die folgenden Schritte sind nötig, um das Hauptprogramm und die beschriebenen Module zu installieren:

1. Installation von Quake3 von CD1 in das Verzeichnis *c:\Quake3*.
2. Update von Quake3 durch Ausführen von *q3pointrelease_132.exe* von CD2.
3. Kopieren des Verzeichnisses *\RLmod* von CD2 nach *c:\Quake3*.

A.3.2. Aufzeichnung der Daten

Um, wie in Kapitel 3, Daten eines zu imitierenden Agenten zu gewinnen, ist wie folgt vorzugehen:

1. Starten von Quake3 mittels Ausführen von *QuakeMOD* aus dem Verzeichnis *c:\Quake3\rlmod*.

Die modifizierte Startdatei sorgt dafür, dass die aktuelle Log-Datei gelöscht wird, die geänderten Programmquellen geladen werden und Quake3 mit der aus Kapitel 2.1.5 beschriebenen Map beginnt. Das Programm startet ohne Agenten aus der Beobachterperspektive.

A. Anhang



Abbildung A.7.: Bild nach dem Start von Quake3.

2. Einstellungen in Quake3 vornehmen.

Dieser Schritt ist nur einmal erforderlich und dient dazu, Quake3 den Vorgaben in dieser Arbeit anzupassen.

- Durch Drücken der Taste „ESC“ ist das Hauptmenü zu erreichen.
- Hier *Setup* auswählen und unter *Player* den Namen des, vom menschlichen Spieler zu steuernden, Agenten eingeben. Dieser ist „RLAgent“.
- Im Setup-Menü unter *Player* nacheinander die Punkte *Controls*, *Move* und *Shoot* aufrufen.
- In *Controls* den Parameter *Freelook* auf „OFF“ setzen.
- In *Move* die Einstellung für *Alwaysrun* vornehmen und auf „On“ stellen.
- In *Shoot* den automatischen Waffenwechsel vornehmen. *Autoswitchweapon* wird zu diesem Zweck „On“ gesetzt.
- Durch mehrmaliges Drücken von „ESC“ die Menüs verlassen.

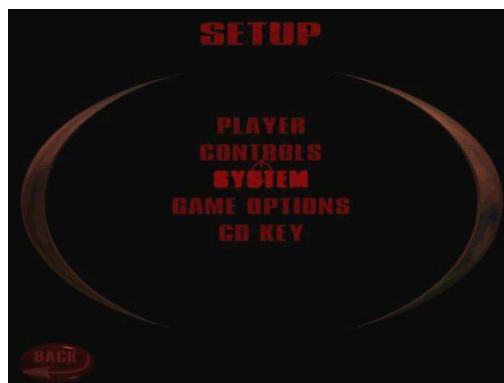


Abbildung A.8.: Einstellungen im Setup vornehmen.

A. Anhang

3. Betreten der Welt.

Im Hauptmenü unter *Start JoinGame* auswählen. Hierdurch wechselt der menschliche Spieler aus der Beobachterperspektive in einen Agenten mit Namen „RL-Agent“. Dieser befindet sich nach dem Eintritt in die Welt auf einer Erhöhung in der Map, die in Abbildung A.7 zu sehen ist.

4. Hinzufügen weiterer Agenten.

Die Szenarien vier und fünf erfordern, dass zum Zeitpunkt der Aufzeichnung der Daten sich auch andere Agenten in der Welt bewegen. Zu diesem Zweck müssen im Hauptmenü unter *AddBots* die benötigten Agenten hinzugefügt werden. Dies geschieht durch Auswahl eines Agenten und Drücken von *Accept*. Durch die Einstellung des Parameters *Skill* kann der Schwierigkeitsgrad bestimmt werden. Mittels *Back* geht es zurück zum Hauptmenü. Hier *ResumeGame* wählen und der ausgewählte Agent betritt die aktuelle Map an einem der Einstiegspunkte.



Abbildung A.9.: Hinzufügen eines Agenten.

5. Ausführung von Agentenaktionen.

Der „RLAgent“ wird von einem menschlichen Spieler durch die Welt gesteuert, wobei die Daten aufgezeichnet werden. Die Handlungen sollen dabei dem, durch das jeweilige Szenario vorgegeben, Verhalten entsprechen.

6. Beenden von Quake3.

Nachdem ausreichend Daten gesammelt worden sind, wird das Programm durch die Wahl von *ExitGame* aus dem Hauptmenü beendet. Die Log-Datei befindet sich jetzt im Verzeichnis *c:\Quake3\rlmod*.

A.3.3. Clusterprogramm

Ein eigenständiges Programm übernimmt das Clustern, wie in Kapitel 4.2.1 erklärt.

1. Start des Programms *Cluster.exe* aus dem Verzeichnis *c:\Quake3\rlmod*.

Es öffnet sich ein Dos-Fenster, indem das Einlesen der Daten aus der Log-Datei und die anschließende Konvertierung in das Arff-Format verfolgt werden kann.

2. Benutzereingabe.

Der Benutzer legt einen gemeinsamen Namen für CLU-, RLQ- und TST-Datei fest. Die Dateieindungen werden nicht eingegeben. Anschließend besteht die Möglichkeit verschiedene Einstellungen für das Clustermodul vorzunehmen. Dies sind die Anzahl der *Iterationen*, die für jede Zustandsklasse durchlaufen werden sollen, bis die Cluster gefunden worden sind und die beiden Parameter *distancenum* und *distancenom*. Sollen keine Änderungen an den Einstellungen erfolgen, so werden Default-Werte für das Clustern übernommen. Für die Anzahl der *Iterationen* sind dies 20, für den Parameter *distancenum* 25 und für *distancenom* ist der voreingestellte Wert 2.

3. Berechnung abwarten.

Je nach Größe der Log-Datei und der Anzahl der zu definierenden Cluster kann die Berechnung einige Minuten dauern. Dabei werden im Dos-Fenster die aktuell ermittelten Cluster ausgegeben. Zusätzlich kann abgelesen werden in welcher Iteration sich das Programm befindet und wievielen Clustern im letzten Durchlauf keine Instanzen zugeordnet wurden. Die Berechnungen sind beendet, wenn „Clustering ready“ ausgegeben wird.

Im Verzeichnis `c:\Quake3\rlmod\arff` ist die für die Strukturierung der Daten gewonnene Arff-Datei zu finden. Die für die Steuerung eines nachahmenden Agenten benötigten Dateien sind in `c:\Quake3\rlmod\control` abgelegt.

A.3.4. Der modifizierte Agent in Quake3

An dieser Stelle wird erklärt, welche Schritte nötig sind, damit ein nachahmender Agent in Quake3 eine erlernte Steuerung benutzen kann, online lernen kann und wie Ergebnisse der Verhaltensimitation gesichert werden.

1. Start von Quake3, wie unter Punkt 1 in Kapitel A.3.2 beschrieben.

2. Hinzufügen von Agenten.

Agenten werden, wie in Kapitel A.3.2 unter Punkt 4 gezeigt, in die Map gesetzt. Es muss der imitierende Agent „Crash“ gestartet werden. Dieser startet am gleichen Punkt in der Welt, wie der zu kopierende Agent (siehe Punkt 3 in Kapitel A.3.2). Je nach Anforderung des Szenarios werden weitere Agenten hinzugefügt.

3. Steuerung einlesen.

Solange keine, der für die Steuerung des imitierenden Agenten benötigten, Dateien eingelesen sind, bleibt „Crash“ am Einstiegspunkt stehen, ohne zu agieren. Um die Zustände und Aktionswahrscheinlichkeiten einzulesen wird mittels der Taste „^“ die Konsole in Quake3 aufgerufen. Hier wird durch den Befehl „\load X“ die gewünschte Steuerung geladen. „X“ steht dabei für den Namen, der beim Clustern für die Dateien gewählt wurde. Nach diesem Schritt beginnt der Agent gemäß seiner Steuerung zu agieren.

A. Anhang

4. Lernvorgang starten.

Nach dem Start des imitierenden Agenten ist das Lernen deaktiviert. Durch Aufruf der Konsole kann mittels „\learn X X X X“ der Lernvorgang gestartet werden. „X X X X“ steht dabei für die Werte der Lernparameter in folgender Reihenfolge: *modifier1*, *modifier2*, *discountfactor*, *divergencefactor*. Werden die Parameter nicht mit angegeben, so beschränkt sich das Lernen auf das Definieren neuer Cluster, wenn der Agent sich in einem Zustand befindet, aus dem er sich nicht befreien kann (siehe Kapitel 4.2.2.3). Ist das Lernen aktiviert, so kann durch Eingabe



Abbildung A.10.: Eingabe von Befehlen über die Konsole.

des Befehls „\learn“ in der Konsole das Lernen ausgeschaltet werden.

Während des Lernvorgangs in Quake3 werden einige Informationen bezüglich des Lernens ausgegeben. Dies sind zum einen ein summendes Geräusch mit gleichzeitiger Ausgabe von „NewCluster“ auf den Bildschirm, wenn ein neuer Zustand gebildet wird. Zum anderen wird angezeigt, wenn, bedingt durch das Lernen, die Steuerung des Agenten modifiziert wird.

5. Lernfortschritt und Ergebnisse speichern.

Sollen die Online modifizierte Steuerung des Agenten und Informationen für die Bewertung des Erlernten in der Testphase gespeichert werden, so wird auch hierfür die Konsole aufgerufen. Der Befehl „\save“ sichert die entsprechenden Dateien. Diese befinden sich Anschließend im Ordner *c:\Quake3\rlmod\results*. Soll der nachahmende Agent zu einem späteren Zeitpunkt mit dieser neuen Steuerung agieren, so müssen die entsprechenden Dateien in das Verzeichnis *c:\Quake3\rlmod\controls* kopiert werden und die alte Steuerung wird überschrieben.

6. Beenden von Quake3.

Das Programm wird wie in Kapitel A.3.2 beschrieben beendet. Mittels des Vergleichs der beiden gleichnamigen Dateien aus dem *controls*- und *results*-Verzeichnis für Cluster und Aktionswahrscheinlichkeiten und der Auswertungsdatei des Lernens „.tst“ können in der Testphase der Erfolg der Imitation bewertet werden.

A.4. Inhalt der CD's

CD1 (Quake3-Programm-CD):

- *setup.exe*
Installationsroutine von Quake3.

CD2 (Diplomarbeits-CD):

- *q3pointrelease.132.exe*
Update für Quake3.
- *Bedienungsanleitung.pdf*
Installations- und Bedienungsanleitung für Quake3 und die, in dieser Arbeit, erstellten Module¹.
- *Diplomarbeit.pdf*
Die Diplomarbeit als PDF-Dokument.
- *rlmod*
 - *QuakeMod.bat*
Modifizierte Startdatei von Quake3.
 - *Cluster.exe*
Programm für das Clustermodul.
 - *code*
Hier befindet sich der veränderte Source-Code von Quake3.
 - *Cluster*
Ordner für den Programmcode des Clustermoduls.
 - *Szenarien*
In diesem Ordner sind die Log-Dateien, die bei der Datenaufzeichnung der einzelnen Szenarien entstanden sind, abgelegt.
 - *arff*
Die konvertierten Log-Dateien der fünf Szenarien liegen in diesem Ordner.
 - *controls*
Die berechneten Cluster und Aktionswahrscheinlichkeiten sind hier gespeichert.
 - *results*
Dieses Verzeichnis beinhaltet die Ergebnisse der Testphase.

¹Dieses Dokument entspricht den Ausführungen zu Kapitel A.3.

Literaturverzeichnis

- [1] Id Software, Quake III Arena Handbuch, 1999-2001
- [2] J.M.P. van Waveren, The Quake3 Arena Bot, Revision 1 University of Technology Delft Faculty, 2001
- [3] Wikipedia, Die freie Enzyklopädie, <http://de.wikipedia.org/wiki>, 2004
- [4] Beats Biblionetz, <http://beat.doebe.li/bibliothek>, 2004
- [5] Ian H. Witten und Eibe Frank, Data Mining, Carl Hanser Verlag, 2001
- [6] Medialexikon, <http://medialine.focus.de>, 2004
- [7] <http://www.cs.waikato.ac.nz/ml/weka/arff.html>
- [8] J. Han und M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000
- [9] Richard S. Sutton und Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998
- [10] M. Ester und J. Sander, Knowledge Discovery in Databases, Springer-Verlag, 2000
- [11] R. Duda und P. Hart, Pattern Recognition and Scene Analysis. John Wiley and Sons, 1973
- [12] R. Kölle und T.Mandl, Data Mining, http://www.uni-hildesheim.de/mandl/Lehre/DataMining_SS01/DataMining_04_Clustering.pdf, 2001
- [13] A. Dempster, N. Laird, und D. Rubin. Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, 1977.
- [14] AIFB Universität Karlsruhe, Unüberwachte Data-Mining-Verfahren, http://www.kde.cd.uni-kassel.de/lehre/ss2004/kdd/fohlen/4Folie_VII.1_Clustering.pdf
- [15] MacQueen, Some Methods for Classification and Analysis of Multivariate Observations, 1967
- [16] K. Backhaus, Multivariate Analysemethoden, 1999

- [17] A. Jain, M. Murty, und P. Flynn, Data clustering: a review, ACM Computing Surveys, 1999.
- [18] D. Fisch, Clusterverfahren auf großen Datenmengen, <http://cv.fmi.uni-passau.de/fileadmin/cbir/DominiFisch-Ausarbeitung.pdf>, 2004
- [19] L. Kaufman und P. J. Rousseeuw, Finding Groups in Data An Introduction to Cluster Analysis, Wiley Interscience, 1990.
- [20] R. T. Ng und J. Han. Efficient and effective clustering methods for spatial data mining, 20th International Conference on Very Large Data Bases, 1994.
- [21] A. Hinneburg und D. A. Keim, An efficient approach to clustering in large multimedia databases with noise, 1998.
- [22] H. Kleine-Büning und O. Kramer, Neuronale Netze, Reinforcement Learning, Evolutionäre Algorithmen, http://wwwcd.upb.de/cd/ag-klbue/de/courses/ss04/lernen/NN_RL_EA.pdf, 2004
- [23] P. Uttgoff, Incremental Induction of Decision Trees, Kluwer Academics, 1989
- [24] S. Sutton und A. Barto, Reinforcement Learning, MIT Press, 1998
- [25] T. Scheffer und S. Bickel, Reinforcement-Lernen, http://www.informatik.hu-berlin.de/Forschung_Lehre/wm/mldm2004/ReinforcementLernen.pdf, 2004
- [26] ingame GmbH, Die größte deutsche Quake3 Arena Seite, <http://planetquake.ingame.de/index.php>, 2001

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig angefertigt und mich fremder Hilfe nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichtem oder unveröffentlichtem Schrifttum entnommen sind, habe ich als solche kenntlich gemacht.