Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Christian Twelkemeier

Entwicklung und Analyse von dynamischen Detailstufen in Multi-Agenten Simulationen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Christian Twelkemeier

Entwicklung und Analyse von dynamischen Detailstufen in Multi-Agenten Simulationen

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Thomas Thiel-Clemen

Eingereicht am: 10. Februar 2014

Christian Twelkemeier

Thema der Arbeit
Entwicklung und Analyse von dynamischen Detailstufen in Multi-Agenten Simulationen

Stichworte
Fußgängersimulation, Multi-Agenten Simulation, Multi-Resolution Simulation, Optimierung

Kurzzusammenfassung
In dieser Arbeit wird ein Ansatz für die Optimierung von Multi-Agenten Simulationen vorgestellt. Dafür wird ein Multi-Resolution Ansatz ausgewählt, bei welchem die Simulation der Ereignisse mit unterschiedlichen Verfahren berechnet werden können. Dabei werden die unterschiedlichen Verfahren Situationen zugeordnet und zur Laufzeit entsprechend in den Situationen verwendet. Zur Überprüfung des Ansatzes wird der Ressourcenverbrauch anhand von zwei Szenarien analysiert. Durch die dynamische Auswahl der Verfahren, ist eine Reduzierung der benötigten Ressourcen möglich, wobei sich jedoch gezeigt hat, dass die mögliche Einsparung von der Art des Szenarios abhängig ist.
Christian Twelkemeier

Title of the paper
Development and analysis of dynamic level of detail in multi-agent simulations

Keywords
Pedestrian simulation, mutli-agent simulation, multi-resolution simulation, optimization

Abstract
This thesis introduces an approach for multi-agent simulation optimization. A multi-resolution approach is used, in which the events can be simulated with different algorithms. To calculate the events, different algorithms are mapped to situations and choosen at runtime accordingly to the current situation of the simulation. To validate this approach the consumption of resources for two szenarios is analyzed. A reduction of the consumption of resources is possible because of the dynamically choosen algorithm, although it turned out that the possible amout of the reduction depends on the type of the szenario.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einführung

Multi-Agenten Simulationen werden in unterschiedlichen Bereichen, wie bspw. im Gesundheitsbereich (Beeler u. a. (2012), Macal u. a. (2012)), zur Simulation von sozialen Gruppen (Park u. a. (2012)) oder zur Simulation von Fußgängern verwendet.

Dabei können Fußgängersimulationen bspw. zur Analyse der Entfluchtung von Gebäuden oder auch zur Analyse von Veranstaltungen dienen. Bei großen Veranstaltungen kommt es immer wieder zu tragischen Vorfällen, wie bspw. bei der Loveparade 2010 in Duisburg mit 21 Toten und etwa 100 Verletzten (Abschlussbericht Loveparade (2010), S.2).

Um die Ursachen der Vorfälle besser zu verstehen, können mit Hilfe von Multi-Agenten-Simulationen diese Veranstaltungen im Nachhinein simuliert und unterschiedliche Varianten analysiert werden. Außerdem können über diese Simulationen auch Veranstaltungen im Vorhinein simuliert und auf mögliche Probleme analysiert werden.

Für diese Simulationen müssen jedoch ggf. unterschiedliche Ereignisse und Effekte, wie bspw. Feuer und Rauchausbreitung, simuliert werden. Die Berechnung für die Simulation von solchen physikalischen Effekten kann einen hohen Ressourcenbedarf verursachen. Das kann dazu führen, dass die Simulation nicht mehr auf einem Rechner ausgeführt werden kann oder dass dieser sehr lange benötigt, um die Simulation zu berechnen.

Diesem Problem kann nur bedingt mit einer vertikalen Skalierung - d.h. mit schnelleren Rechnern - entgegen gewirkt werden. Eine andere Möglichkeit ist die horizontale Skalierung. Dabei wird die Simulation auf mehrere Rechner verteilt (Mertens u. a. (2006), Thiel (2013)). Dies kann jedoch auch, je nach Simulation, zu einem nicht vernachlässigbaren Kostenfaktor führen.

Eine weitere Möglichkeit ist den Ressourcenbedarf der Simulation möglichst gering zu halten. Dafür ist es denkbar, bestimmte Aspekte der Simulation, wie bspw. die Simulation von physikalischen Effekten in unterschiedlichen Genauigkeiten mit unterschiedlichen Ressourcenbedarf zu modellieren. Dabei werden in dieser Arbeit der Aspekt als Skala bezeichnet und die unterschiedlichen Modellierungen bzw. Umsetzungen des bspw. physikalischen Effekts als Detailstufe.

1.1. Ziel der Arbeit

In dieser Arbeit wird untersucht, wie eine selektive Reduktion der Detailstufen umgesetzt werden kann und ob durch diese Reduktion der Ressourcenbedarf einer Simulation verringert werden kann, ohne dass sich die Qualität der Ergebnisse erheblich verschlechtert.

Dabei ist die Hypothese, dass nicht immer die komplette Simulation mit der gleichen Detailstufe simuliert werden muss. Es wird die These aufgestellt, dass es ausreicht, bestimmte Bereiche, die abhängig von dem Modell sind, mit einer hohen Genauigkeit zu simulieren und andere Bereiche mit Modellen abgebildet werden können, die weniger Ressourcen bei der Simulation benötigten.

Um diese Hypothese zu belegen, sollen zwei Beispielszenarien analysiert werden und entsprechend den Ergebnissen der Analyse soll eine Struktur für eine Multi-Agenten Simulation erarbeitet und prototypisch umgesetzt werden, sodass die Hypothese mit den beiden Szenarien näher untersucht und gestärkt oder widerlegt werden kann.

1.1.1. Ausgrenzungen

Ziel dieser Arbeit ist es nicht, aus den verwendeten Szenarien tatsächliche Aussagen bezogen auf die Vorgänge in der realen Welt zu treffen. Diese Szenarien werden lediglich verwendet, um die Analyse der Hypothese dieser Arbeit durchzuführen. Aus diesem Grund wird eine Validierung bzw. Plausibilisierung dieser Szenarien lediglich soweit durchgeführt, wie dies für die Betrachtung der Hypothese benötigt wird.

1.2. Aufbau der Arbeit

Nach der Einleitung und dem Ziel mit den Abgrenzungen der Arbeit aus Kapitel 1, werden in Kapitel 2 die Grundlagen für die weitere Arbeit beschrieben. Dies umfasst die Begriffsdefinitionen für die folgende Arbeit, die Aufarbeitung der Simulationsansätze und die Aufarbeitung von vorhandenen Arbeiten in Abschnitt 2.5. Im Anschluss daran werden die beiden Beispielszenarien, welche in dieser Arbeit verwendet werden, in Abschnitt 2.7 beschrieben.

In Kapitel 3 werden dann die Anforderungen für dynamische Detailstufen analysiert und beschrieben. Außerdem wird untersucht, wie eine dynamische Auswahl durchgeführt werden kann und wie bei einem Wechsel der Detailstufe zur Simulationszeit die Migration zwischen diesen erfolgen kann. Danach erfolgt in Abschnitt 3.4 die Analyse der Detailstufen in den Beispielszenarien.

Nach der Analyse wird in Kapitel 4 die Struktur der Multi-Agenten Simulation beschrieben, sowie die Algorithmen zur Umsetzung der unterschiedlichen Detailstufen. Abgeschlossen wird das Kapitel mit der Beschreibung, wie die dynamische Auswahl der Detailstufen durchgeführt werden kann, in Abschnitt 4.3.

Im folgenden Kapitel 6 wird dann die Modellierung der Beispielszenarien sowie die Durchführung der Messungen bei der Ausführung der Simulationen beschrieben. Danach wird in Abschnitt 6.3 eine Validierung eines Szenarios durchgeführt. Im Anschluss daran werden in Abschnitt 6.4 die Ergebnisse aus den Simulationen der Beispielszenarien ausgewertet und diskutiert.

Die Arbeit wird mit einem Vergleich der vorliegenden Arbeit mit den vorhandenen Ansätzen (Kapitel 7), welche in Abschnitt 2.5 beschrieben wurden, sowie einem Fazit und Ausblick in Kapitel 8 abgeschlossen.

2. Grundlagen

Für die Analyse und den Aufbau der Multi-Agenten Simulation werden einige Grundlagen benötigt. Dies umfasst sowohl die technische Struktur einer solchen Simulation als auch die Grundlagen für die Berechnung der physikalischen Effekte.

2.1. Begriffsdefinition

In dieser Arbeit wird, wie bereits in Abschnitt 1.1 beschrieben, untersucht, ob durch unterschiedliche Detailstufen eine Verringerung des Ressourcenbedarfs einer Simulation bei etwa gleichbleibender Qualität der Ergebnisse erreicht werden kann. Dafür werden im Folgenden der Arbeit unter anderem die Begriffe der Detailstufen und Skalen verwendet. Dieser Zusammenhang zwischen Detailstufen und Skalen ist in Abbildung 2.1 dargestellt.

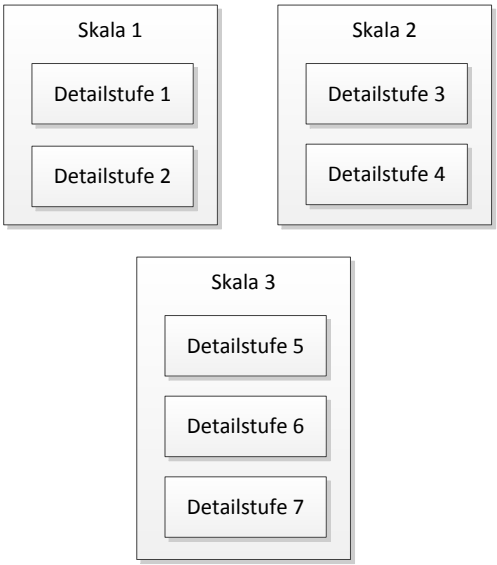


Abbildung 2.1.: Zusammenhang zwischen Detailstufe und Skala

Für einen bestimmten Teilbereich einer Simulation können verschiedene Möglichkeiten existieren, wie diese simuliert werden können. Diese unterschiedlichen Möglichkeiten können

auch unterschiedliche Qualitäten bezüglich der Ergebnisse oder des Ressourcenbedarf aufweisen. Solche unterschiedlichen Möglichkeiten der Simulation werden in dieser Arbeit jeweils als eigene Detailstufen bezeichnet.

Diese Detailstufen sind in Skalen organisiert. Eine Skala ist in dieser Arbeit eine Gruppe von Detailstufen, die fachlich zusammen gehören, indem sie den gleichen Teil der Simulation behandeln. Ein Verschieben einer Detailstufe von einer Skala in eine andere Skala ist nicht möglich, da sich die Gruppierung, wie beschrieben, direkt aus der fachlichen Aufgabe der Detailstufe ergibt.

Dies sei am Beispiel einer Kollisionserkennung verdeutlicht: Wenn sich zwei Objekte treffen, kann bspw. die Erkennung der Kollision unterschiedlich genau simuliert werden. Die Kollision kann überhaupt nicht abgeprüft werden, was dazu führt, dass diese nicht erkannt wird und sich zwei oder mehr Objekte zur gleichen Zeit am gleichen Ort befinden. Die Kollision kann aber auch sehr genau abgeprüft werden. Dies stellt in dieser Arbeit zwei unterschiedliche Detailstufen dar. Diese zwei Detailstufen gehören zu einer Skala, die bspw. Kollisionserkennung heißt.

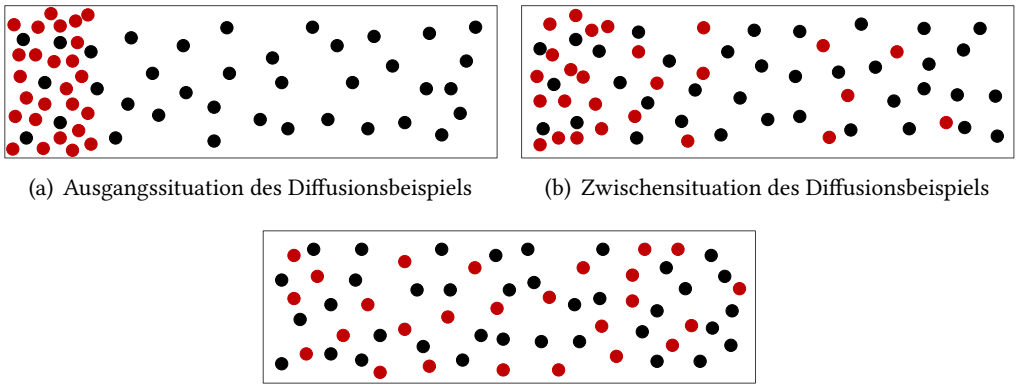
2.2. Diffusion

Die Ausbreitung von bspw. Gasen in der Luft wird, wie in Povh (2011) (Kap. 7.9) Demtröder (2013) (Kap. 7.5.1) sowie in Socolofsky und Jirka (2005) (Kap. 2) beschrieben, als Diffusion bezeichnet. Dies ist der Vorgang, der auftritt, wenn sich ein Stoff, ohne weitere Einflüsse, wie bspw. Luftströmungen, mit einem anderen Stoff vermischt. Eine Diffusion kann u.a. auch in festen oder flüssigen Stoffen auftreten, was im Folgenden jedoch nicht näher beschrieben wird.

Für die Erläuterung sei nachfolgend das Beispiel aus Povh (2011) (S. 110) verwendet. Hierbei diffundiert ein Stoff A in einen anderen Stoff B. Die Moleküle des Stoffs B sind zu Beginn gleichmäßig in dem Raum verteilt, während sich alle Moleküle des Stoffs A, wie in Abbildung 2.2(a) dargestellt, zu Beginn auf der linken Seite befinden.

Die Moleküle der jeweiligen Stoffe bewegen sich zufällig und stoßen dabei zusammen. Dadurch, dass es in dem Bereich mit der geringeren Konzentration zu weniger Kollisionen kommt, bewegen sich statistisch mehr Moleküle von dem Bereich mit der hohen Konzentration in den Bereich mit der geringeren Konzentration. Das hat den Effekt, dass nach einiger Zeit die Stoffe gleichverteilt sind (vgl. Demtröder (2013), S. 200).

Die Teilchenstromdichte, welche angibt, wie viele Moleküle sich pro Zeiteinheit durch eine Fläche bewegen (vgl. Povh (2011), S. 111), kann mit Hilfe des ersten Fick'schen Gesetzes berechnet werden (siehe Demtröder (2013), S. 201 sowie Crank (1975), S. 2):



(a) Ausgangssituation des Diffusionsbeispiels (b) Zwischensituation des Diffusionsbeispiels

(c) Endsituation des Diffusionsbeispiels

Abbildung 2.2.: Beispiel für die Diffusion von Stoffen. Moleküle des Stoffs A sind rot und Moleküle des Stoffs B sind schwarz dargestellt. [Povh (2011), S. 110]

$$F = -D\frac{\delta c}{\delta x} \tag{2.1}$$

c gibt hierbei die Konzentration des Stoffs an und x gibt die Position des Dichtegradienten an. Der Diffusionskoeffizient D gibt an, wie schnell der Stoff in den anderen Stoff diffundiert (vgl. Demtröder (2013), S. 202). Für die Diffusion von Gasen liegt nach Baehr und Stephan (2008) (S. 80) der Diffusionskoeffizient etwa zwischen $5 * 10^{-6}$ und $10^{-5}\frac{m^2}{s}$. Für eine genauere Beschreibung des Koeffizienten sei an dieser Stelle auf die Literatur verwiesen (siehe Reid u. a. (1987), Kap. 11, Povh (2011), S. 111f.).

Die zeitliche Veränderung der Ausbreitung des Stoffes durch den Vorgang der Diffusion wird nach Crank (1975) (S. 4) für den eindimensionalen Fall durch die Differentialgleichung 2.2 und für den dreidimensionalen Fall durch die Gleichung 2.3 beschrieben.

$$\frac{\delta c}{\delta t} = D\frac{\delta^2 c}{\delta x^2} \tag{2.2}$$

$$\frac{\delta c}{\delta t} = D\left(\frac{\delta^2 c}{\delta x^2} + \frac{\delta^2 c}{\delta y^2} + \frac{\delta^2 c}{\delta z^2}\right) \tag{2.3}$$

Bei dieser Gleichung liegt die Annahme zugrunde, dass der Diffusionskoeffizient nicht von der Zeit oder der Position abhängig, sondern konstant ist. Wenn der Koeffizient von der Position abhängig ist, wird die Veränderung durch die Gleichung 2.4 bzw. 2.5 beschrieben.

$$\frac{\delta c}{\delta t} = \frac{\delta}{\delta x} D \frac{\delta c}{\delta x} \tag{2.4}$$

$$\frac{\delta c}{\delta t} = \frac{\delta}{\delta x}\left(D\frac{\delta c}{\delta x}\right) + \frac{\delta}{\delta y}\left(D\frac{\delta c}{\delta y}\right) + \frac{\delta}{\delta z}\left(D\frac{\delta c}{\delta z}\right) \tag{2.5}$$

2.3. Simulationsansätze

Es existieren für die Simulation unterschiedliche Ansätze, welche sich besonders für bestimmte Simulationstypen eigenen. Dies kann bspw. wie in Zhou u. a. (2010) (Kap. 2) nach der Zeitspanne und nach der Anzahl der Objekte, welche simuliert werden sollen, eingeteilt werden. Für die zeitliche Achse kann zwischen kurzen und langen Simulationszeiten unterschieden werden. Dabei sind kurze Zeitspannen im Bereich von Minuten oder Stunden und lange Zeitspannen ab mehreren Tagen definiert.

Für die Achse der Anzahl der Agenten, kann zwischen einer geringen, mittleren und hohen Anzahl an Agenten unterschieden werden. Zhou u. a. (2010) nennen als Grenze für wenige Agenten weniger als 100, für eine mittlere Anzahl an Agenten weniger als 1000 und für eine hohe Anzahl ab 1000 Agenten.

Durch die Einteilung in die drei Gruppen der zeitlich langen, zeitlich kurzen mit geringer bis mittlerer Anzahl an Agenten und der zeitlich kurzen mit hoher Anzahl an Agenten, welche von Zhou u. a. (2010) (S. 20:5) durchgeführt und in Abbildung 2.3 dargestellt ist, wird deutlich, für welche Bereiche sich welche Simulationsansätze besonders eigenen.

2.3.1. Flussbasiert

Beim flussbasierten Ansatz wird von den einzelnen Individuen abstrahiert und lediglich die Masse der Individuen modelliert. Dies wird mit Hilfe der Boltzmann-Gleichung simuliert (Helbing (1992)). Auch bei diesem Simulationsansatz können sich emergente Effekte wie bspw. Laning entwickeln (Helbing (1992)).

Dadurch, dass keine Individuen einzeln simuliert werden müssen, eignet sich dieser Ansatz für Simulationen mit sehr vielen Individuen (vgl. Zhou u. a. (2010) S.20:6). Hierbei darf jedoch das Verhalten von einzelnen Individuen nicht relevant sein, da dies in diesem Ansatz nicht modelliert werden kann. Das bedeutet, dass bspw. Emotionen von einzelnen Fußgängern in einer solchen Simulation nicht abgebildet werden können und sich daher auch kein emergentes Verhalten aus diesen Verhaltensweisen bilden kann.

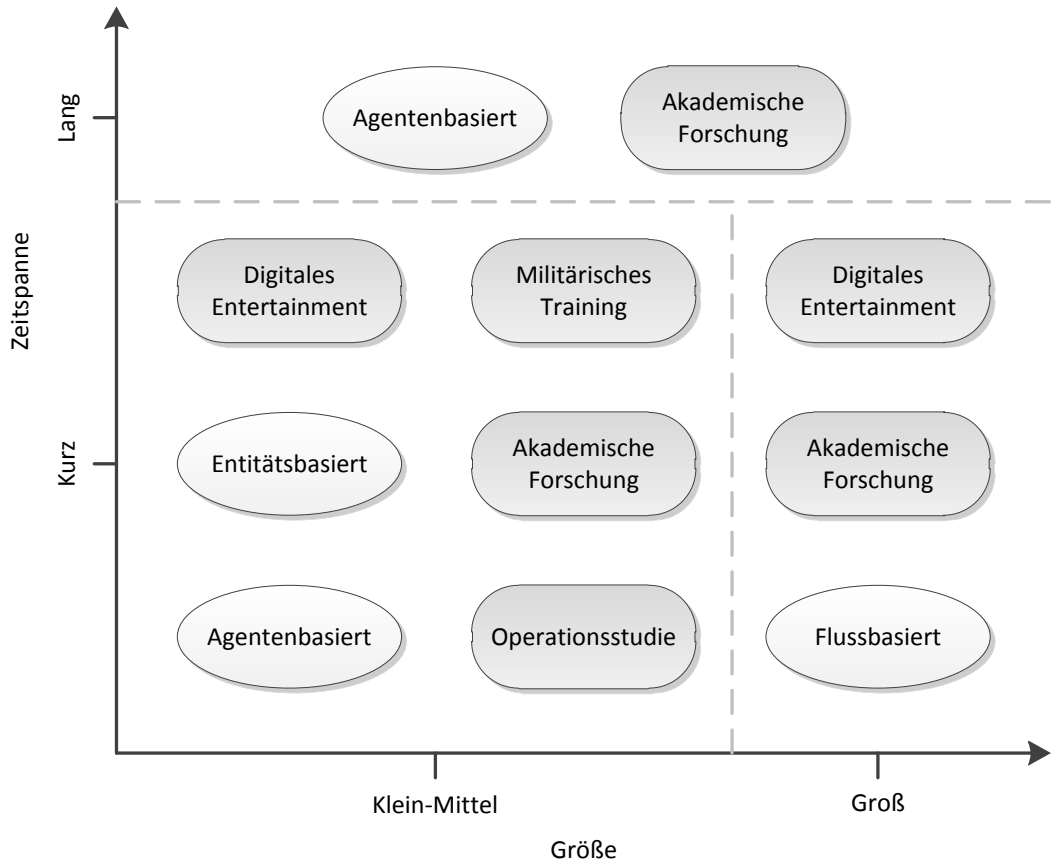


Abbildung 2.3.: Klassifikation der Simulationsansätze [Zhou u. a. (2010) (S.20:5)]

2.3.2. Entitätsbasiert

Die entitätsbasierten Ansätze modellieren die einzelnen Individuen als homogene Entitäten (vgl. Zhou u. a. (2010), S. 20:6 f.). Hierbei besitzen die Individuen keine eigenen Verhaltensweise wie in dem agentenbasierten Ansatz.

Die Entitäten werden dadurch nicht selber aktiv, sondern durch Kräfte, wie bspw. in dem Social-Force-Modell von Helbing und Moinar (1995) oder über Flow-Tiles von Chenney (2004) bewegt. Bei den Flow-Tiles werden die Flussrichtungen vor der Simulation festgelegt.

Bei diesem Ansatz können ebenfalls keine individuellen Verhaltensweisen modelliert und simuliert werden.

2.3.3. Agentenbasiert

Bei dem agentenbasierten Ansatz wird jedes Individuum als autonomer Agent mit eigenem Verhalten modelliert. Dieser Agent kann mit anderen Agenten interagieren und nimmt die Umwelt wahr. Ein Beispiel für die Umsetzung der Wahrnehmung von Agenten ist in Steel u. a. (2010b) beschrieben. Auf Basis dieser Wahrnehmungen und seinem Zustand kann dieser dann selbständig und autark Entscheidungen treffen und Aktionen in der Umwelt ausführen (vgl. Zhou u. a. (2010), S. 20:7,20:18).

Bei diesem, wie auch bereits bei dem entitätsbasierten Ansatz, kann durch wenige, einfache Regeln bereits emergentes Verhalten der abgebildeten Individuen erzeugt werden (vgl. Zhou u. a. (2010), S.20:7)

Dadurch, dass jedes Individuum als eigener Agent simuliert wird, hat dieser Ansatz im Vergleich mit den beiden anderen vorgestellten Ansätzen den größten Bedarf an Ressourcen für die Simulation. Aus diesem Grund eignet sich ein agentenbasierter Ansatz für eine geringe bis mittlere Anzahl an Agenten, wenn das individuelle Verhalten abgebildet werden soll (vgl. Zhou u. a. (2010), S.20:5). Es existieren jedoch bereits Arbeiten um die Rechenlast auf mehrere Rechner zu verteilen und so auch Simulationen mit einer hohen Anzahl an Agenten mit dem agentenbasierten Ansatz zu ermöglichen. Dies wird bspw. in Mertens u. a. (2006) sowie Thiel (2013) beschrieben.

2.4. Multi-Agenten Simulation

Multi-Agenten Simulationen bestehen u.a. aus der Umwelt und den Agenten. Diese Teile müssen jeweils entsprechend den Anforderungen an die Simulation modelliert und miteinander verbunden werden.

2.4.1. Modellierung der Umwelt

Die Umwelt einer Multi-Agenten Simulation ist die Datenstruktur, in der sich die Agenten bewegen bzw. in welcher diese agieren können. Das bedeutet, dass sich bspw. bei einer Fußgängersimulation die physische Repräsentation der Agenten bzw. Fußgänger in der Umwelt befindet und von dieser verwaltet wird.

Wie in Weyns u. a. (2005) beschrieben, kann die Umwelt als First-Class Entität betrachtet werden, welche selbständig, d.h. ohne Beeinflussung durch einen Agenten, ihren Zustand ändert (vgl. Weyns u. a. (2005), S. 2). Die Umwelt kann jedoch auch als einfache Datenstruktur modelliert werden, welche sich nicht unabhängig von den Agenten ändern kann.

Zwei andere mögliche Modellierungsansätze beschreibt Ferber (1999) (S. 211ff.). Hierbei gibt es einmal die zentralisierte Umwelt, bei welcher es genau eine Umwelt gibt, auf die alle Agenten zugreifen. Die andere Möglichkeit ist eine dezentrale Umwelt. Hierbei ist die Umwelt in Teile aufgeteilt, wobei jeder Teil eine eigene Umwelt ist. Die Agenten befinden sich dann zu einem Zeitpunkt in einem dieser Teilbereiche und so auch in der entsprechenden Umwelt.

Bei der verteilten Umwelt können die einzelnen Teilbereiche miteinander kommunizieren, wodurch sich Informationen von einem Teilbereich in einen anderen Teilbereich ausbreiten können. Dies wird bspw. in Steel und Wenkstern (2010) für die Simulation von Umweltereignissen wie Wind und Feuer angewendet. Des Weiteren kann eine dezentrale Umwelt, wie bspw. in Mertens u. a. (2006) beschrieben, für eine verteilte Simulation verwendet werden, welche auf mehrere Rechner verteilt ist.

2.4.2. Agentenmodellierung

Russel und Norvig (2010) definieren einen Agenten wie folgt:

> „An Agent is anything that can be viewed as perceiving its environment through sensors and acting upon the environment through actuators.“ (Russel und Norvig (2010), S.34)

Dabei wird deutlich, dass ein Agent, wie auch in Wooldridge (2002) (S.33 ff.) beschrieben, über eine Wahrnehmung verfügen muss und entsprechende Aktionen auf der Umwelt ausführen kann.

Diese Definition macht keine Aussage über den internen Aufbau der Agenten. Jedoch funktioniert nach Görz u. a. (2003) (S. 953ff.) ein Agent, wie in Abbildung 2.4 dargestellt, nach dem 3-Phasen-Zyklus. Die erste Phase wird von Görz u. a. (2003) „Informationsaufnahme“ genannt. Dabei nimmt der Agent die Umwelt über seine Sensoren wahr.

Abbildung 2.4.: Grobe Darstellung eines Agenten [Brenner u. a. (1998), S. 50 angelehnt an Müller (1996) S. 8]

Bei der, von Görz u. a. (2003) „Wissensverarbeitung und Entscheidung“ genannten, zweiten Phase trifft der Agent seine Entscheidung, welche Aktion dieser ausführen möchte. Dafür kann der Agent in dieser Phase auch sein Wissen über die Umwelt aktualisieren.

In der letzten, von Görz u. a. (2003) „Aktionsausführung“ genannten, Phase, werden die Aktionen des Agenten tatsächlich ausgeführt.

Der genaue interne Aufbau des Agenten hängt maßgeblich von dem verwendeten Typen des Agenten ab. So kann im einfachsten Fall von reaktiven Agenten gesprochen werden. Diese nehmen ihre Umwelt wahr und führen lediglich aufgrund des aktuellen Zustands der Umwelt eine Aktion aus (Abbildung 2.5). Das bedeutet, diese Art von Agenten kann nicht lernen, da es keine Speicherung von Wissen oder Zustandsänderungen des Agenten gibt. Diese Agenten haben jedoch den Vorteil, dass diese einfacher aufgebaut sind, als die deliberativen Agenten, welche eine Zustandsänderung besitzen können (Wooldridge (2002) S.33 sowie Brenner u. a. (1998) S.56ff.).

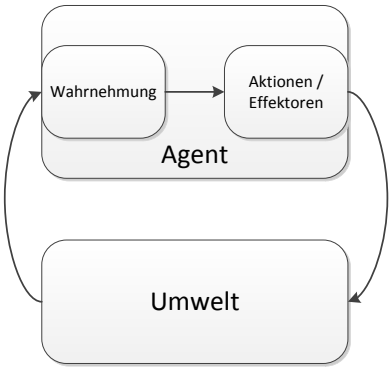


Abbildung 2.5.: Funktionsweise eines reaktiven Agenten [Wooldridge (2002), S.34]

Die deliberativen Agenten besitzen ein Modell von der Umwelt, welches diese zur Auswahl der nächsten Aktion verwenden. Diese Agenten werden auch BDI-Agenten genannt (Brenner u. a. (1998), S.52ff.). BDI steht dabei für Beliefs, Desires und Intentions. Für eine genauere

Beschreibung der BDI-Architektur sei auf Rao und Georgeff (1991) sowie Rao u. a. (1995) verwiesen.

2.4.3. Ereignisbehandlung

Für die Verbindung zwischen der Umwelt und den Agenten können zwei Schnittstellen identifiziert werden. Die Aktionsausführung der Agenten sowie die Wahrnehmung der Umwelt durch die Agenten.

Dabei können die Ereignisse in einer Simulation über das Influences-Reaction-Modell aus Ferber und Müller (1996) sowie Ferber (1999), Kap. 4.3 behandelt werden, welches in Abbildung 2.6 skizziert ist.

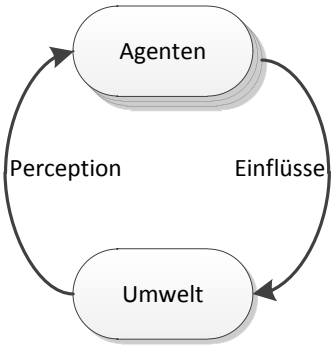


Abbildung 2.6.: Skizze des Influences-Reaction-Modell [angelehnt an Steel u. a. (2010a)]

Das Influences-Reaction-Modell gibt an, dass die Agenten Einflüsse (Influences) erzeugen, wenn diese eine Aktion ausführen. Diese Einflüsse werden dann durch eine Funktion „React“ nach den Gesetzen der Umwelt verarbeitet. Dabei werden die Einflüsse als simultan angenommen (vgl. Ferber (1999), S. 163ff.). Diese Änderungen in der Umwelt können dann als Reaktionen (Reactions) durch die Agenten wahrgenommen werden.

2.4.4. DIVAs

Bei DIVAs (Dynamic Information Visualization of Agent systems) handelt es sich um ein Framework für Multi-Agenten Simulationen (vgl. DIVAs Projekt Webseite (2012)). In dem System, welches unter anderem in Steel und Wenkstern (2010) sowie Steel u. a. (2010a) beschrieben wird, werden die Ereignisse entsprechend dem Influences-Reaction-Modell simuliert.

Dabei ist die Umwelt spatial in Zellen eingeteilt, für die jeweils ein spezieller Agent, der Zellencontroller, zuständig ist. Die Einflüsse der Agenten aus der Zelle werden durch diesen

gesammelt und entsprechend die Reaktionen berechnet, aus welchen dann der neue Zustand der Zelle berechnet wird.

Sollte sich ein Ereignis, wie bspw. Feuer, aus einer Zelle in eine Nachbarzelle ausbreiten, informiert der Zellencontroller der Quellzelle die Nachbarzelle über das Ereignis einen Zeitpunkt bevor sich der Effekt in die Nachbarzelle ausbreitet. Diese kann dann selber, unter Berücksichtigung aller Einflüsse, die Reaktionen und den neuen Zustand berechnen.

2.5. Multi-Level Simulation

Bei der Multi-Level Simulation werden unterschiedliche Detailstufen in einer einzelnen Simulation verwendet. Diese können sowohl a priori für die komplette Simulation oder auch für einzelne Bereiche festgelegt werden oder aber auch dynamisch zur Simulationszeit gewechselt werden. In dieser Arbeit wird der Fokus auf die Ansätze gelegt, welche dynamisch zur Laufzeit die Detailstufe wechseln können.

2.5.1. Holon-basiert

Die Holon-basierten Ansätze haben gemein, dass die unterschiedlichen Detailstufen über die verschiedenen Aggregationsstufen der Holon Hierarchie umgesetzt werden. Dabei ist ein Holon ein Teil eines ganzen.

Ein Holon beschreibt einen Teil, welcher, wie in Abbildung 2.7(a) dargestellt, aus weiteren Holonen bestehen kann. Mehrere Holone, welche entsprechend in einer Teil-Ganze Beziehung zueinander stehen, bilden eine Holon Hierarchie (vgl. Gaud u. a. (2008b)). Dies ist als Beispiel in Abbildung 2.7(b) dargestellt.

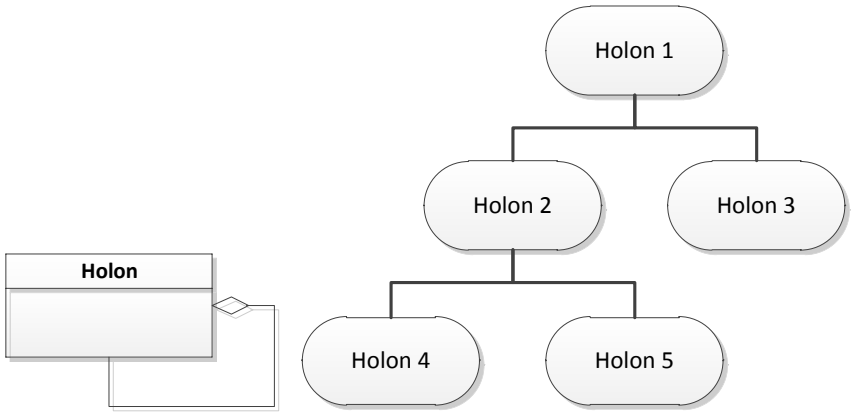
Die unterschiedlichen Detailstufen werden durch die unterschiedlichen Hierarchiestufen der Holon Hierarchien abgebildet. Dabei stellt jedes Super-Holon eine Detailstufe mit einer geringeren Genauigkeit als seine Sub-Holone bereit. Das bedeutet, dass je tiefer in der Hierarchie abgestiegen wird, umso genauer werden die Detailstufen.

Um dies zu ermöglichen, ist die Simulation in diesem Ansatz wie folgt strukturiert:

Die sogenannten Organisationen enthalten Rollen und bestehen aus einem Zusammenschluss von mehreren Holonen, die die gleichen Anforderungen erfüllen möchten.

Die Rollen beschreiben jeweils eine Aufgabe, welche je nach Art von einem oder mehreren Holonen ausgefüllt werden können. So existiert bspw. die Reactor Rolle, die dafür zuständig ist, in der Umwelt die Reaktionen auf die Influences zu berechnen.

Die Rollen können untereinander über Interaktionen interagieren um ihr Ziel zu erreichen. Diese Interaktionen sind Ereignisse, welche unter den Holonen ausgetauscht werden.



(a) Klassendiagramm für ein Holon

(b) Beispiel einer Holon Hierarchie

Abbildung 2.7.: Darstellung eines Holons und einer Holon Hierarchie

Bei den Holonen handelt es sich um Spezialisierungen von Agenten. Diese bilden aufgrund der Anforderungen der Simulationsdomäne selbständig die Hierarchien, indem diese diesen beitreten.

Für die Verwaltung der Hierarchien gibt es den sogenannten „holonischen Aspekt“ (Gaud u. a. (2008b), S.1663). Diese Organisation ist technisch und nicht spezifisch für die Anwendungsdomäne. Sie dient der Verwaltung der Holon-Hierarchie und weiteren technischen Aufgaben der Simulation.

Die Anwendungsdomäne wird in dem sogenannten „Produktions Aspekt“ (Gaud u. a. (2008b), S.1663) behandelt. Diese Organisation enthält die Rollen der Anwendungsdomäne und bildet daher den fachlichen Teil der Simulation ab.

Die Ausführung der Agenten bzw. Holone wird entsprechend der Hierarchie durchgeführt. Dafür existieren die drei Rollen Scheduler, Multi-Scheduler und Scheduled. Dabei kann ein Multi-Scheduler mehrere Sub-Holone einplanen. Diese können dann wiederrum Multi-Scheduler sein, welche wieder ihre Sub-Holone einplanen können. Jeder Scheduler entscheidet ob er seine Sub-Holone ausführt. Dadurch kann, wie in Abbildung 2.8 dargestellt, die Detailstufe jederzeit geändert werden.

Wenn in der Simulation ein Holon mehrere Anwendungsrollen ausfüllt, kann dieses Holon auch in mehreren Organisationen, bzw. Hierarchien zur gleichen Zeit enthalten sein.

Neben den Agenten und dem Scheduling wird auch die Umwelt über diese Struktur modelliert. Dabei kann bspw. die Umwelt, wie in Abbildung 2.9 dargestellt, in Bereiche aufgeteilt werden, welche wiederum in Bereiche aufgeteilt werden können. Diese einzelnen Bereiche werden

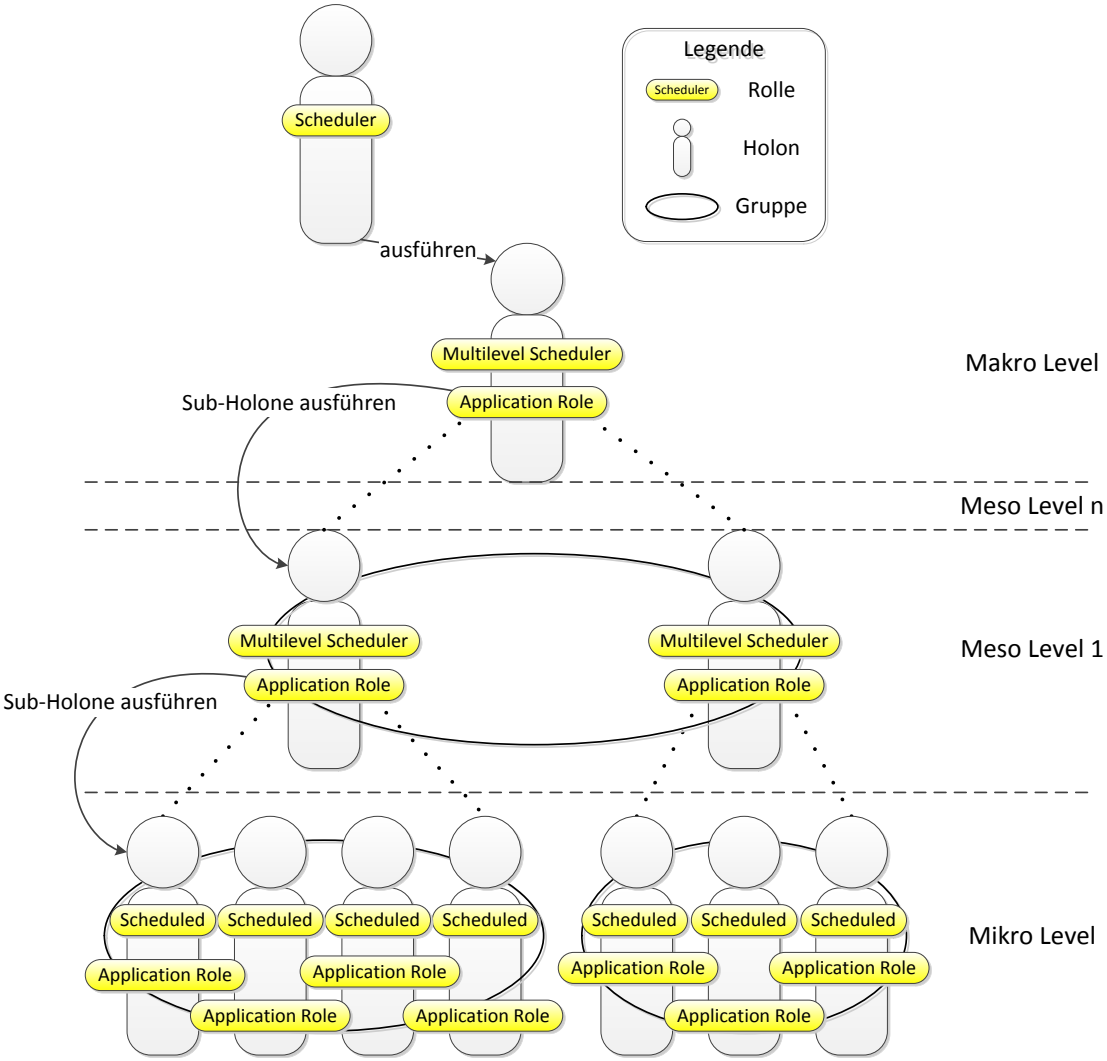


Abbildung 2.8.: Scheduling der Holone [angelehnt an Gaud u. a. (2008b)]

dann durch sogenannte Area Exchanger verbunden. Diese übertragen die Einflüsse zwischen den einzelnen Bereichen.

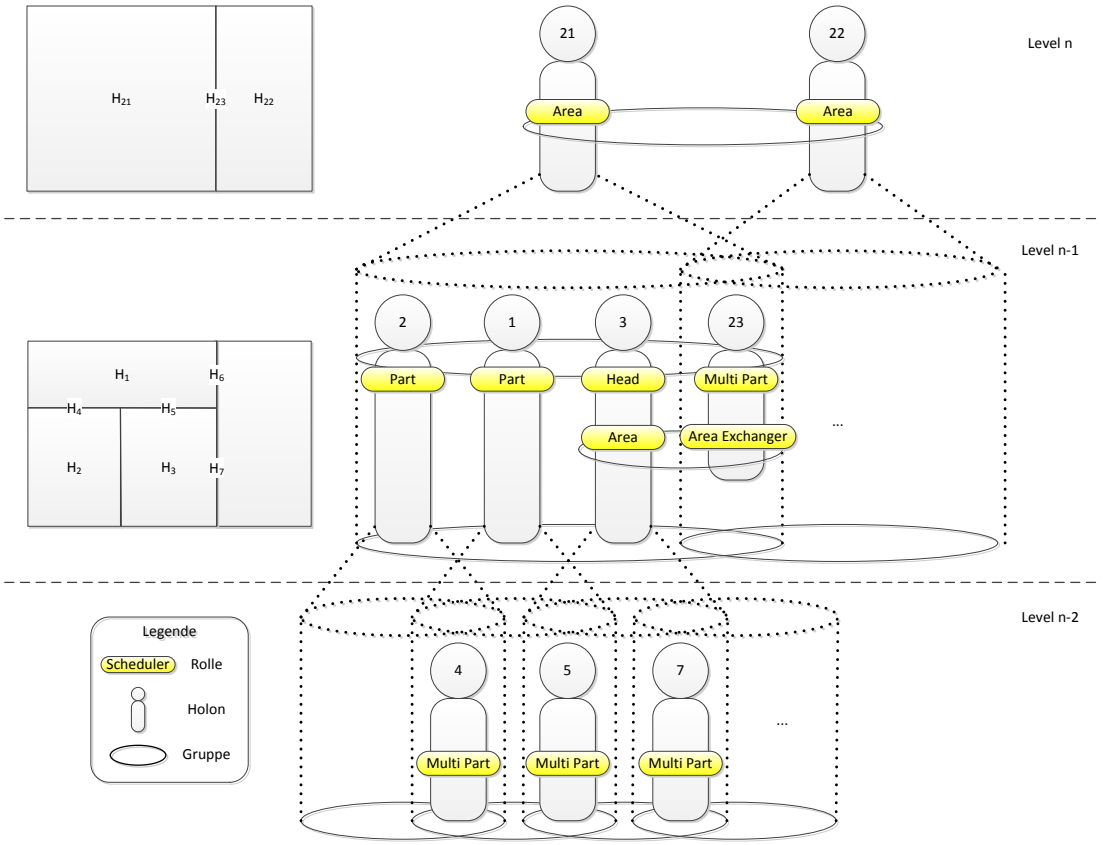


Abbildung 2.9.: Aufteilung der Umwelt [angelehnt an Gaud u. a. (2008b)]

Die Verarbeitung der Einflüsse erfolgt dann, wie bereits beschrieben, über die Reactor-Rolle.

Für weitere Details sei an dieser Stelle auf die Arbeiten von Gaud u. a. (2008a,b), Cossentino u. a. (2007, 2010), Moujahed u. a. (2007) sowie Rodriguez u. a. (2007) verwiesen.

2.5.2. Agentenzentriert

Bei dem agentenzentrierten Ansatz werden die Agenten für die Änderung der Detailstufen verwendet. In dem Ansatz von Navarro u. a. (2011) können die Agenten zu immer größeren Gruppen zusammen gefasst werden. Dieser Ansatz wurde in Navarro u. a. (2013) erweitert, sodass nicht nur ganze Agenten zusammen gefasst werden können, sondern auch nur Teilbereiche der Agenten. Ein Teilbereich kann bspw. die Navigation der Agenten oder die Wahrnehmung sein.

Welche Agenten zusammengefasst werden, wird durch eine Affinitätsfunktion definiert. Diese wird in Navarro u. a. (2011) zwischen mehreren Agenten und zwischen Agenten und Ereignissen berechnet. Diese Funktion berechnet die Distanz zwischen zwei Agenten bzw. zwei Agenten und einem Ereignis. Hierfür ist die Funktion in die zwei Bereiche physikalische Distanz D_Θ und psychologische Distanz D_ψ aufgeteilt. Die physikalische Distanz D_Θ gibt die spatiale Distanz zwischen zwei Agenten bzw. dem Agenten und einem Ereignis wieder. Dies kann bspw. eine euklidische Distanz sein.

Die psychologische Distanz D_ψ zwischen zwei Agenten gibt an, wie stark die Agenten die gleichen Attribute aufweisen. Diese Attribute sind bspw. das Ziel der Agenten, das Wissen oder Emotionen. Zwischen einem Agenten und einem Ereignis gibt die psychologische Distanz die Beeinflussung des Agenten durch das Ereignis an. So gibt Navarro u. a. (2011) als Beispiel an, dass das Ereignis, dass eine Person ohnmächtig wird, einen Arzt stärker beeinflusst, als einen Angestellten, der in Eile ist.

Wenn die berechnete Distanz zwischen Agenten einen Schwellenwert unterschreitet, werden die Agenten in Navarro u. a. (2011) zu einem Agenten zusammen gefasst. Nachdem mehrere Agenten zusammen gefasst worden sind, werden diese, wie in Abbildung 2.10 dargestellt, durch einen einzelnen Agenten in der Umwelt repräsentiert. Dabei handelt es sich dann um eine makroskopische Simulation der Agenten, da nicht mehr die einzelnen Agenten simuliert werden, sondern alle als eine Gruppe. Dies verfälscht, wie auch in Navarro u. a. (2011) beschrieben, die Wahrnehmung der Agenten, da bspw. anstatt den zehn einzelnen Agenten jetzt lediglich ein einzelner Agent wahrgenommen wird. Durch diese Verfälschung kann es zu einem veränderten Verhalten der Agenten und dadurch zu veränderten Ergebnissen der Simulation kommen.

In der Erweiterung des Ansatzes in Navarro u. a. (2013) wurde diesem Ansatz die Möglichkeit hinzugefügt, mesoskopisch zu simulieren. Hierbei wird, wie bereits erwähnt, ermöglicht, dass die Agenten nur teilweise zusammen gefasst werden. Dafür wird ein Agent als eine Menge von Prozessen modelliert, wobei jeder Prozess einer Fähigkeit des Agenten entspricht. So sind bspw. die Navigation, Wahrnehmung und Kommunikation eigene Prozesse. Diese Prozesse werden wieder jeweils über einzelne Schwellenwerte bei der Distanz aggregiert. Hierbei kann es vorkommen, dass ein Prozess eines Agenten mikroskopisch und ein anderer Prozess desselben Agenten mesoskopisch simuliert wird. Das bedeutet, dass der Prozess, welcher mesoskopisch simuliert wird mit dem Prozess eines anderen Agenten zusammengefasst wurde.

2.5.3. Proxy-basiert

Der Proxy-basierte Ansatz, wie dieser in Chenney u. a. (2001) beschrieben wird, stammt aus dem Bereich der Visualisierung. Dabei wird die Detailstufe für Bereiche reduziert, welche

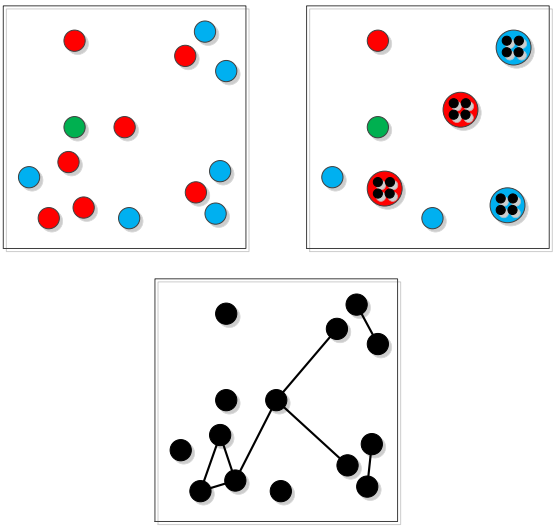


Abbildung 2.10.: Aggregation von Agenten [Navarro u. a. (2011)]

aktuell nicht sichtbar sind. Diese Reduktion der Detailstufe erfolgt, indem die Individuen in den nicht relevanten Bereichen nicht mehr mikroskopisch, sondern bspw. lediglich makroskopisch simuliert werden.

Der Proxy übernimmt die Simulation der Objekte, welche sich in seinem Bereich befinden. Objekte, welche sich bspw. aus dem Bereich der hohen Detailstufe in den Bereich des Proxys mit der geringeren Detailstufe bewegen, werden in diesen überführt. Sobald ein Objekt sich wieder im Bereich mit der hohen Detailstufe befindet, wird dieses dann wieder vom Proxy entfernt und in das Model mit der hohen Detailstufe eingefügt. Dabei müssen ggf. Attribute der Objekte, welche in der geringeren Detailstufe nicht simuliert wurden, neu erzeugt oder berechnet werden.

Chenney u. a. (2001) gibt als Beispiel eine Stadt mit Straßen und Autos an. In dem sichtbaren Bereich werden die Autos mit einer hohen Detailstufe simuliert, während in dem nicht-sichtbaren Bereich diese mit einer niedrigeren Detailstufe simuliert werden. Dabei werden bei der niedrige Detailstufe die Autos über eine diskrete Ereignissimulation berechnet, welche die Drehung der Reifen der Autos nicht berechnet. Dadurch muss, wenn die Autos wieder in den sichtbaren Bereich überführt werden, die Information über die aktuelle Drehung der Reifen zufällig gesetzt werden.

Ein anderes Beispiel aus Chenney u. a. (2001) ist die Wegfindung der Agenten: So können bspw. in einem Bereich von geringerem Interesse, wie der nicht sichtbare Bereich, die Wegfindung und Bewegung vereinfacht werden. Dabei wird, nachdem der Pfad der Objekte bestimmt

wurde, der Pfad von den Objekten, welche im nicht sichtbaren Bereich liegen, von dem Proxy in eine diskretisierte Welt übertragen. Die Welt ist dabei in Felder eingeteilt. In die Felder, welche von dem Pfad durchquert werden, wird eingetragen, wann das Objekt das jeweilige Feld betritt und verlässt. Diese Zeiten können berechnet werden, indem mögliche Kollisionen mit anderen Objekten, die sich bewegen, durch Verzögerungen abgebildet werden. Dadurch muss der Pfad nicht mehr um die anderen Objekte geplant werden.

2.6. Ereignisse

In Multi-Agenten Simulationen können die Ereignisse, wie bereits in Unterabschnitt 2.4.3 beschrieben, unterschiedlich behandelt werden. Durch die Art, wie die Ereignisse behandelt und simuliert werden, ist auch abhängig, wie unterschiedliche Detailstufen bei der Simulation umgesetzt werden können.

In dieser Arbeit werden die Ereignisse, ähnlich wie bereits in Twelkemeier (2012) (Kap. 3.2) und Twelkemeier (2013b) (Kap. 2.1) beschrieben, angelehnt an Steel und Wenkstern (2010), Steel u. a. (2010a) sowie Ferber (1999) (Kap.4.3), wie in Abbildung 2.11 dargestellt, modelliert. Dabei setzt sich ein Ereignis aus den Teilen Aktion, Einfluss und Effekt zusammen. Eine Aktion stellt eine abstrakte Handlung dar, die bspw. ein Agent ausführen kann. Diese Aktion kann dann zu Einflüssen in der Umwelt führen. Diese Einflüssen wirken dann auf die entsprechenden Effekte und können diese verändern. So kann ein Agent die Umwelt, bzw. die Effekte in der Umwelt, nicht direkt, sondern nur indirekt über Einflüsse, welche aus Aktionen hervorgerufen werden, ändern.

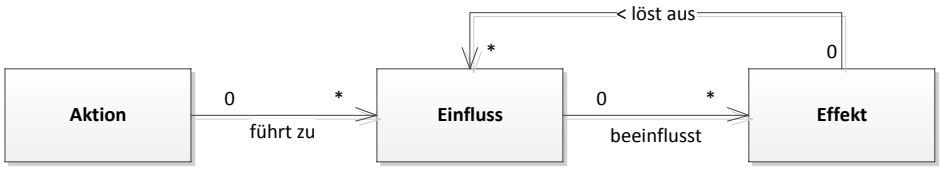


Abbildung 2.11.: Modellierung von Ereignissen

Durch eine Aktion Schubsen kann ein Agent bspw. einen Druck-Einfluss herbeiführen. Dieser Einfluss sorgt über die Beeinflussung der Umwelt mit dem Druck-Effekt bspw. zum Fallen eines Agenten.

Des Weiteren kann bspw. ein Effekt A, welcher einen anderen Effekt B beeinflussen soll, dies indirekt über Einflüsse, welche von Effekt A erzeugt werden, erreichen. Dies wird dadurch modelliert, dass ein Effekt Einflüsse erzeugen kann, welche wieder Effekte beeinflussen.

2.7. Szenarien

In dieser Arbeit wird, anhand der im Folgenden beschriebenen Szenarien, die Hypothese untersucht. Dabei dienen diese Szenarien lediglich als Beispiel sowie zur Überprüfung der Machbarkeit.

2.7.1. Evakuierung eines Kinosaals

Das erste Szenario, welches in dieser Arbeit betrachtet werden soll, ist die Evakuierung eines Kinosaals aus Klüpfel (2003) (Kap. 4.3.2).

Für diese Übung wurden u.a. von Klüpfel (2003) die Enfluchtungszeiten der Personen gemessen. Diese beschreiben, wann welche Person welchen Ausgang erreicht hat.

Bei diesem Szenario sitzen die Personen zu Beginn auf ihren Plätzen im Vorführraum des Kinos. Die Vorstellung ist nicht vollständig ausgebucht, sodass nicht alle Sitzplätze belegt sind.

In der Übung wurden die Personen instruiert, den Raum zu verlassen. Dazu wurde jeder Person ein Hut mit einer Nummer aufgesetzt, sodass der Weg und die benötigte Zeit für den Weg der Person auf den Videoaufzeichnungen nachvollzogen werden konnte (vgl. Klüpfel (2003) S.75).

Wie in Klüpfel (2003) (S. 75) beschrieben, wurde die Übung mit Studenten durchgeführt. Der Kinosaal besitzt 11 Reihen mit jeweils 14 Sitzplätzen und zwei Reihen mit 10 Sitzplätzen (siehe Abbildung 2.12). Außerdem existieren drei Ausgänge, wovon in der Übung jedoch von den Teilnehmern nur zwei verwendet wurden.

Der Ausgang A befindet sich in Abbildung 2.12 rechts oben und der Ausgang B ist über den Weg nach unten aus dem Kinosaal zu erreichen.

Bei den Abmessungen existieren in Klüpfel (2003) Inkonsistenzen. Aus diesem Grund werden im Folgenden die Abmessungen aus Abbildung 4.4, Seite 76 in Klüpfel (2003) verwendet.

An der Übung haben 101[1] Personen teilgenommen. In Tabelle 2.1 ist dargestellt, wie diese zum Beginn der Simulation in dem Kinosaal verteilt waren und welchen Ausgang jeder einzelne verwendet hat.

2.7.2. Massenflucht aus einer Diskothek

Bei dem zweiten Szenario handelt es sich ebenfalls um ein Entfluchtungsszenario, welches auf einer Begebenheit in dem Nachtclub E2 in Chicago aus dem Jahr 2003 basiert (Schneider (2011) (Kap. 3.2.5), CNN (2003a), Spiegel Online (2003)).

[1] Die Anzahl an Teilnehmern ist in der Arbeit von Klüpfel (2003) inkonsistent. Auf Seite 75 werden 100 Teilnehmer genannt, wobei auf Seite 78 101 genannt werden.

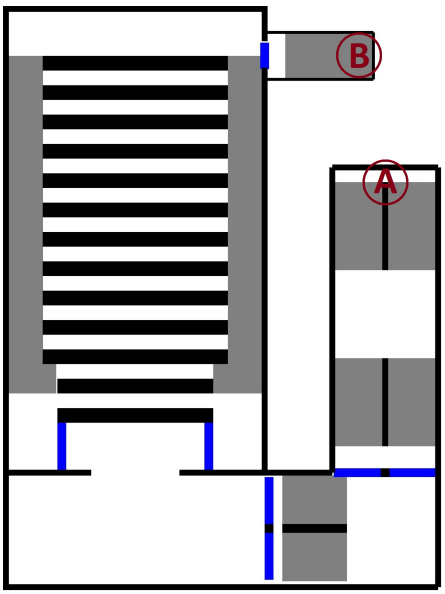


Abbildung 2.12.: Darstellung des Kinosaals (Blau dargestellt sind Türen und grau dargestellt sind Treppen) [angelehnt an Klüpfel (2003) S. 75ff.]

Sitz Reihe	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1										A		A		
2														
3	A	A	A	A	A	A			A					
4					A	A	A	A		A				
5	A	A	A	A		B	B		A	A	A			
6	A	A			B	B	B	A	A	A	A	A		
7		B		B	B	B		A	A		A	A	A	A
8		B	B	B		B	B	B	A	A	A		A	
9		B	B	B		B			A	B	B	B		A
10	B	B	B		B	B	B	B	B		B	B	B	
11	B	B	B		B		B	B	B		B	B		B
12					B	B	B	B		B	B			
13				B	B	B	B	B	B	B		B		

Tabelle 2.1.: Verteilung und gewählte Ausgänge der Teilnehmer [angelehnt an Klüpfel (2003), S. 82]

In diesem Szenario befinden sich etwa 500 Personen in einer Diskothek, welche sich im ersten Stockwerk befindet. Nach Schneider (2011) (S. 69) ist die Diskothek wie in Abbildung 2.13 dargestellt aufgeteilt. Der Hauptein- und Ausgang E1 ist über die Treppe (grau dargestellt) zu erreichen.

Während einer Veranstaltung kommt es nach CNN (2003a) zu einem Kampf auf der Tanzfläche, welcher vom Sicherheitspersonal unter Einsatz von Pfefferspray beendet wird. Dieses Pfefferspray breitet sich in dem Raum aus. Bei Personen, die die Ursache des Pfeffersprays nicht mitbekommen haben, treten Fluchtreaktionen auf. Zusätzlich entstehen Gerüchte, dass es sich um einen Anschlag handelt (CNN (2003a)).

Die Personen fliehen daraufhin zum Hauptein- bzw. Ausgang. Dabei stürzen einige Personen, was dazu führt, dass sich die Personen im Ausgang stapeln und dadurch irgendwann keine Personen mehr durch diesen Ausgang das Gebäude verlassen konnten (Schneider (2011) (S. 69) nach CNN (2003b)). Durch den hohen Druck, der auf die unten liegenden Personen wirkt, kam es zu Verletzten und Todesopfern.

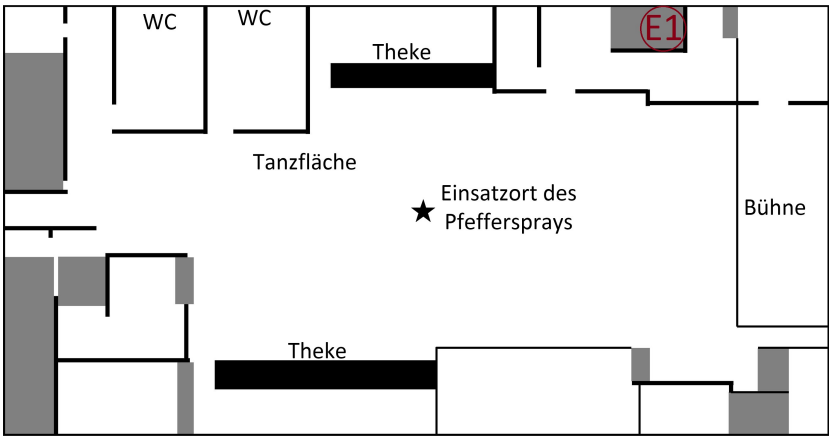


Abbildung 2.13.: Skizze der Diskothek [angelehnt an Schneider (2011) S. 69]

3. Analyse

Für die Umsetzung von unterschiedlichen Detailstufen der Skalen, existieren für diese Anforderungen. Nachdem diese Anforderungen im folgenden Abschnitt aufgestellt wurden, werden die Skalen und Detailstufen aus den Beispielszenarien entwickelt und entsprechend der Anforderungen analysiert.

3.1. Anforderungen

Für die Simulation von Effekten in unterschiedlichen Detailstufen müssen sowohl die Detailstufen als auch die Art, wie die die Detailstufen ausgewählt werden, definiert werden. Im Folgenden werden die Anforderungen, die existieren, näher beschrieben.

3.1.1. Detailstufen

Bei dem hier verwendeten Modell der Simulation werden, wie bereits in Abschnitt 2.6 beschrieben, die Berechnungen der Auswirkungen von Aktionen durch die Effekte durchgeführt. Um dabei eine Einsparung von Ressourcen bei der Simulation zu erreichen, kann, wie in Gaud u. a. (2008b) beschrieben, ein Holon-basierter Ansatz verwendet werden. In dieser Arbeit wird jedoch versucht, die Änderungen lediglich auf die Simulation der Effekte zu beziehen. Daher müssen die Effekte in unterschiedlichen Genauigkeiten simuliert bzw. berechnet werden können. Im besten Fall kann für jede Skala eine totale Ordnung der Detailstufen bezüglich des Ressourcenverbrauchs und der Genauigkeit aufgestellt werden. Daher sollte der Ressourcenverbrauch mit der Genauigkeit der Detailstufe ansteigen.

Jedoch kann auch eine Detailstufe D_1 in einer Situation S_1 genauere Ergebnisse liefern kann, als eine andere Detailstufe D_2 und gleichzeitig jedoch in einer anderen Situation S_2 liefert die Detailstufe D_2 genauere Ergebnisse als D_1. Das bedeutet, dass die, bezogen auf die Qualität der Ergebnisse, beste Detailstufe abhängig von der Situation sein kann. Dadurch ist eine solche totale Ordnung nicht immer möglich und notwendig. Daher ist es lediglich notwendig, dass pro Situation eine Ordnung aufgestellt werden kann.

3.1.2. Auswahl der Detailstufen

Bei der Auswahl der Detailstufe muss es möglich sein, dass in jeder Situation für jeden Effekt eine zu verwendende Detailstufe definiert ist. Das bedeutet, dass keine Situation existieren darf, in der für die Simulation eines Effektes keine Detailstufe definiert ist.

Aufgrund der Anforderung, dass sich die Detailstufe pro Situation ändern kann und ggf. sehr viele Effekte im Laufe einer Simulation berechnet werden, ist davon auszugehen, dass diese Abfrage der Detailstufe sehr häufig durchgeführt wird. Dass bedeutet, dass dies ein kritische Funktion für den Ressourcenverbrauch ist. Daher sollte diese Abfrage der Detailstufe eine hohe Performance aufweisen, um den Mehraufwand für die Auswahl der Detailstufe gering zu halten.

Außerdem muss es möglich sein, dass die Zuordnung zwischen der Situation und den Detailstufen der Skala ohne eine Änderung des Simulationsmodells durchgeführt werden kann. Dadurch ist es möglich, dass diese Zuordnung auch von Domänenexperten erstellt und gepflegt werden kann.

3.1.3. Migration zwischen Detailstufen

Dadurch, dass sich die zu verwendende Detailstufe für einen Effekt jederzeit ändern kann, müssen eventuell existierende Zustände der Detailstufen in die anderen Detailstufen überführt werden können. Dabei existiert, wie auch in Navarro u. a. (2011) beschrieben, das Problem, dass ggf. gewisse Werte, die von einer höheren Detailstufe benötigt werden, in einer niedrigeren Detailstufe nicht existieren. In diesem Fall muss die Skala bzw. die Detailstufen definieren, wie diese Transformation der Zustände zwischen den Detailstufen geschieht.

Diese Transformation muss jedoch beliebig oft und zu jedem Zeitpunkt möglich sein, da sich, wie bereits beschrieben, die zu verwendende Detailstufe jederzeit ändern kann.

3.1.4. Zusammenfassung

Die in diesem Abschnitt beschriebenen Anforderungen können, wie in Tabelle 3.1 beschrieben, in drei Gruppen eingeteilt werden. Die Anforderungen der Detailstufen, der Auswahl der Detailstufen sowie der Migration zwischen den Detailstufen. Nicht betrachtet wurden in diesem Abschnitt Anforderungen an die Simulation im Allgemeinen, wie bspw. das ein Agent die Umwelt wahrnehmen kann oder Aktionen in dieser ausführen kann, da dies als Grundlage vorausgesetzt wird und der Fokus der Anforderungen aus diesem Kapitel auf dem Ziel der Arbeit liegt.

#	Anforderung
Detailstufen	
A1.1	Es müssen Detailstufen mit unterschiedlicher Genauigkeiten und mit unterschiedlichem Ressourcenbedarf umgesetzt werden können.
A1.2	Für jede Situation muss eine Ordnung bezüglich der Genauigkeit und dem Ressourcenbedarf aufgestellt werden können.
Auswahl der Detailstufen	
A2.1	Es muss immer für jede Skala zu jeder möglichen Situation eine Detailstufe bestimmt werden können.
A2.2	Die Zuordnung zwischen den Situationen und der Detailstufe der jeweiligen Skala muss auch von einem Domänenexperten durchgeführt werden können.
A2.3	Nicht funktionale Anforderung: Die Abfrage der Detailstufe muss möglichst performant sein.
Migration zwischen Detailstufen	
A3.1	Die Zustände von einzelnen Detailstufen müssen in andere Detailstufen der gleichen Skala überführt werden können.
A3.2	Die Transformation der Zustände muss beliebig oft und zu jedem Zeitpunkt bzw. in jeder Situation möglich sein.

Tabelle 3.1.: Zusammenfassung der Anforderungen an die Simulation bezüglich der Detailstufen

3.2. Auswahl der Detailstufe

In den Arbeiten von Navarro u. a. (2011, 2013), Chenney u. a. (2001) sowie Gaud u. a. (2008b) werden bereits unterschiedliche Detailstufen verwendet. Diese Ansätze können für die dynamische Auswahl der Detailstufe teilweise adaptiert werden.

In den Arbeiten von Navarro u. a. (2011, 2013) liegt der Fokus, wie bereits in Unterabschnitt 2.5.2 beschrieben, auf der Anpassung der Detailstufe über eine (teilweise) (De-) Aggregation der Agenten. Bei Chenney u. a. (2001) wird lediglich mit zwei Detailstufe gearbeitet: Eine hohe und eine niedrige Detailstufe, wobei die Detailstufen ein anderes Simulationsmodell verwenden. Das bedeutet, dass die Detailstufen in der Arbeit unterschiedliche Modelle beschreiben, zwischen welchen gewechselt werden kann. Bei diesem Ansatz wird nicht zwischen verschiedenen Effekten oder Ereignissen unterschieden. In Gaud u. a. (2008b) werden die Agenten sowie die Umwelt mit Hilfe von Holonen zusammengefasst. Hierbei bildet eine Hierarchiestufe eine Detailstufe ab. Bei diesem Ansatz werden keine Unterscheidungen zwischen verschiedenen Skalen getroffen. Des Weiteren wird keine Möglichkeit beschrieben, dass zwei Effekte in zwei unterschiedlichen Bereichen unterschiedlich genau simuliert werden können. Bspw. soll in einem Bereich B_1 der Effekt E_1 mit einer hohen und der Effekt E_2 mit

einer niedrigeren Genauigkeit simuliert werden, während in einem Bereich B_2 die Detailstufen genau gedreht sind. D.h. der Effekt E_1 soll dort mit einer niedrigen und der Effekt E_2 mit einer hohen Detailstufe simuliert werden.

3.2.1. Bedingungen für die Auswahl der Detailstufe

Bei dem Ansatz von Navarro u. a. (2011, 2013) wird die (De-) Aggregation der Agenten bzw. Prozesse über Distanz-Funktionen mit Schwellenwerten definiert. Dies kann für diese Arbeit für die Auswahl der Detailstufe der Effekte adaptiert werden. So ist es denkbar, dass die benötigte Detailstufe eines Effektes von der Dichte der Agentenpopulation in einem Bereich, bspw. dem Ursprung des Effektes, abhängig ist.

Chenney u. a. (2001) verwenden als Kriterium für die Auswahl der Detailstufe den Sichtbereich. Dieser kann ebenfalls als mögliche Bedingung für die Zuordnung zwischen einer Situation in der Simulation und der Detailstufe einer Skala adaptiert werden. Durch die Definition von Fokusbereichen, welche sowohl spatial als auch temporal sein können, kann der Fokus von dem Simulationsmodellierer genauer definiert werden. So könnten bspw. Bereiche, welche für die Simulation nicht von Interesse sind, jedoch simuliert werden müssen, mit einer geringeren Detailstufe simuliert werden.

Regelbasiert

Da eine Zuordnung zwischen der Situation der Simulation und der jeweiligen Detailstufe einer Skala von dem Szenario und dem Fokus der Simulation abhängt, muss die Zuordnung unabhängig von der Implementierung der Simulation geändert werden können. Dies kann bspw. über Regeln erfolgen, welche aufgrund von Bedingungen die Situation beschreiben und eine Zuordnung der Detailstufe pro Skala durchführen.

Um mit den Regeln Bereiche abbilden zu können, welche von hohem Interesse sind und deswegen immer in einer speziellen Detailstufe simuliert werden sollen, können Bereichs-Bedingungen verwendet werden. Diese Bereiche können bspw. mit Rechtecken oder anderen Formen modelliert werden.

Durch die Bedingungen, welche Dichte von Agenten in einem Bereich herrscht, kann die Detailstufe verringert werden, wenn lediglich das makroskopische Verhalten für die Simulationsergebnisse relevant ist. Als Bereich kann ein Kreis verwendet werden. So kann die Dichte der Agenten in einem Bereich B, definiert durch den Punkt x und dem Radius r, wie folgt berechnet werden:

Sei A die Menge aller Agenten, welche sich in dem Kreis B befinden, welcher aus dem Mittelpunkt x und dem Radius r gebildet wird und r(a) der Radius der auf den Boden projizierten

Regel	Beschreibung
Bereiche	Ein räumlicher Bereich wird definiert.
Agentendichte	Die Dichte der Agenten in einem Bereich entspricht einem Kriterium.
Zeitbereich	Die Simulationszeit ist innerhalb eines definierten Bereichs.
Effekttyp	Die Art des Effekts entspricht einem definierten Wert.

Tabelle 3.2.: Bedingungen für die Regeln

Standfläche eines Agenten a aus A in der Umwelt. So kann die Dichte über $\frac{\sum_{a \in A} 2\pi r(a)^2}{2\pi r^2}$ berechnet werden. Ein Wert von 1 gibt hierbei die maximale Dichte an und ein Wert von 0 würde bedeuten, dass sich kein Agent in dem Bereich befindet.

Über die Bedingung des Zeitbereichs ist es möglich, den Fokus der Simulation auf bestimmte zeitliche Bereiche der Simulation zu legen. Dadurch wird die aktuelle Simulationszeit als Bedingung verwendet.

Um den Fokus der Simulation auf bestimmte Ereignisse und Effekte legen zu können, kann über eine Effekt-Bedingung die Detailstufe für einen Effekt festgelegt werden.

Zusammengefasst wären u.a. die in Tabelle 3.2 dargestellten Bedingungen für die Regeln denkbar. Um die Ausdrucksmöglichkeit der Regeln zu erweitern, sollten diese Bedingungen auch kombiniert werden können. Dafür können die booleschen Operationen UND und ODER verwendet werden.

Durch eine Kombination der einzelnen Bedingungen über boolesche Operationen können komplexere Situation beschrieben werden. Einer solchen Zustandsbeschreibung wird dann jeweils ein Tupel aus Skala und Detailstufe zugeordnet.

Falls es bei den Zuordnungen zu Überschneidungen kommt, sodass einer Situation zwei unterschiedliche Detailstufen der gleichen Skala zugeordnet sind, muss diese Situation definiert gelöst werden. So ist es denkbar, die Reihenfolge der Definitionen zu verwenden oder einer bezüglich der Qualität höheren Detailstufe Vorrang vor einer niedrigeren Detailstufe zu geben.

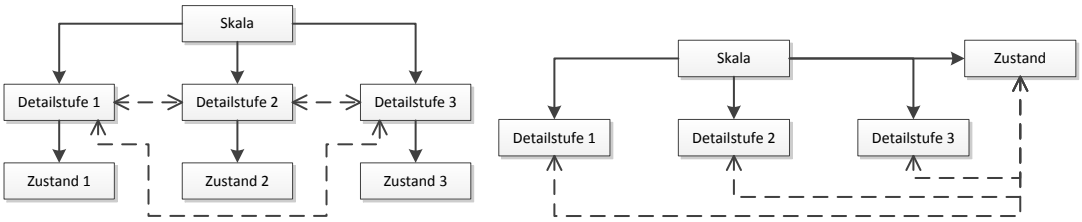
3.3. Migration zwischen Detailstufen

Wenn ein Effekt einen Zustand besitzt, muss der Zustand des Effektes ggf. auf die unterschiedlichen Detailstufen transformiert werden. Dies wird bspw. in Navarro u. a. (2011, 2013) über zwei unterschiedliche Funktionen durchgeführt, welche die einzelnen Attribute der Agenten oder Prozesse zusammenfassen. Bei der Deaggregation werden diese Werte dann wieder aufgeteilt. Wenn die Attribute, die deaggregiert werden, nicht existieren, müssen diese, wie auch in Chenney u. a. (2001) beschrieben, über statistische Werte erzeugt werden.

Bei Chenney u. a. (2001) werden auch Werte, welche für die Simulation nicht weiter relevant und in der niedrigen Detailstufe daher nicht vorhanden sind, willkürlich festgelegt.

Sollte ein Effekt einen Zustand besitzen, existieren zwei prinzipielle Möglichkeiten: Die Erste ist, dass der Zustand, wie in Abbildung 3.1(a) dargestellt, jeweils der Detailstufe zugeordnet ist. Dies hat zur Folge, dass ein Zustand ggf. jeweils beim Wechsel der Detailstufe transformiert werden muss.

Die zweite Möglichkeit ist, dass der Zustand, wie in Abbildung 3.1(b) dargestellt, jeweils der Skala zugeordnet ist, welche den Zustand als eine Art Modell benutzt. Hierbei muss jede Detailstufe der Skala den Zustand nach einer Änderung konsistent für alle Detailstufen hinterlassen. Dies bedeutet, dass die Transformation bei jeder Änderung durchgeführt wird. Der Vorteil ist, dass bei der gleichzeitigen Verwendung unterschiedlicher Detailstufen einer Skala, keine partielle Transformation von einzelnen Teilzuständen erfolgen muss. Eine solche partielle Transformation könnte auftreten, wenn ein Teilbereich die Detailstufe wechselt, während der Rest des ursprünglichen Bereiches die Detailstufe beibehält.



(a) Zustand jeweils der Detailstufe zugeordnet

(b) Zustand der Skala zugeordnet

Abbildung 3.1.: Mögliche Zuordnungen zwischen den Detailstufen und den Zuständen der Umwelt

3.4. Skalen in den Beispielszenarien

Bei der Simulation von unterschiedlichen Ereignissen und Effekten können unterschiedliche Skalen mit Detailstufen entwickelt werden. Diese Skalen können, wie in Abbildung 3.2 dargestellt, in den Bereich der Agenten und der Umwelt unterteilt werden. Der Bereich der Agenten enthält dabei alle Skalen, welche direkt den Agenten betreffen. Die Skalen der Umwelt umfassen alle, welche in der Umwelt der Simulation behandelt werden. Orthogonal zu diesen beiden Bereichen existiert der Bereich der temporalen Skala.

Der Bereich der Agenten kann, angelehnt an Navarro u. a. (2011, 2013), weiterhin in den Bereich der physischen und psychischen Skalen unterteilt werden. Die physischen Skalen beinhalten bspw. die Repräsentation der Agenten in der Umwelt. Das Zusammenfassen der

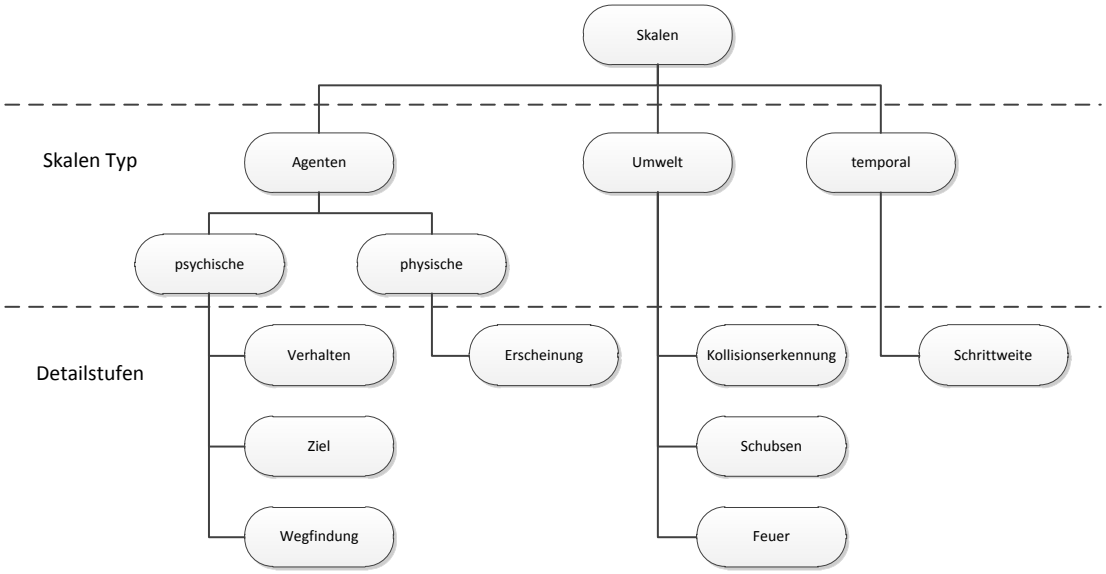


Abbildung 3.2.: Strukturierung der Skalen und Detailstufen mit Beispielen

Agenten zu Gruppen, wie bspw. in Navarro u. a. (2011), wäre ebenfalls eine physische Skala. Die psychischen Skalen der Agenten beinhalten das Verhalten der Agenten. Dies umfasst bspw. die Wegfindung.

In dem Bereich der Umwelt befinden sich die Skalen der Effekte, welche durch die Umwelt simuliert werden. Dies umfasst bspw. die Berechnung der Ausbreitung von Feuer, welches mit unterschiedlicher Genauigkeit simuliert werden kann.

Die temporale Skala umfasst die unterschiedlichen Entwicklungen der Zeit in der Simulation. Dies kann bspw. eine Time-Step-Driven oder auch eine Event-Driven Entwicklung der Zeit sein.

Im Folgenden wird der Schwerpunkt auf die Skalen für die Umwelt sowie die temporale Detailstufe der Szenarien gelegt, welche bereits in Abschnitt 2.7 beschrieben wurden. Für diese Szenarien werden die Skalen und Detailstufen analysiert.

3.4.1. Temporale Skala

Für die temporale Skala, die bereits in Twelkemeier (2013b) (Kap. 2.2.1, Kap 3.1) sowie Twelkemeier (2013a) (Kap. 2.1) beschrieben wurde, existieren zwei Extreme. Das erste Extrem ist die Simulation von Ereignissen möglichst kontinuierlich abzubilden. Das kann, wie in Guo und Tay (2008) (S. 129) beschrieben, über einen Event-Driven Ansatz versucht werden zu erreichen. Dabei wird jeweils zu den Zeitpunkten, zu welchen eine Veränderung der Gege-

benheiten auftritt, simuliert und diese Veränderung mit einbezogen. So gilt für den Zeitpunkt $t_n = t_{n-1} + \Delta t(n)$. Hierbei gibt $\Delta t(n)$ die Schrittgröße zwischen dem Simulationsschritt $n-1$ und n an. Beispielsweise können zwei Agenten sich mit einer Geschwindigkeit in eine Richtung bewegen. Wenn jedoch die beiden Agenten bei der Bewegung zusammenstoßen, ist dies eine Änderung der Gegebenheit. Dadurch kann dies Ereignis, wie in Twelkemeier (2013b) (S. 4) beschrieben, zu der Erzeugung eines neuen Simulationsschrittes führen, in dem die geänderte Situation behandelt wird.

Das andere Extrem ist, wie ebenfalls in Guo und Tay (2008) (S. 129) beschrieben, die Simulation in Zeitscheiben aufzuteilen. Dabei können zu Beginn einer Zeitscheibe jeweils Aktionen ausgeführt werden, jedoch nicht während einer Zeitscheibe. So würde bei dem Beispiel mit den zwei Agenten, die zusammenstoßen, entweder der Zusammenstoß nicht entdeckt oder es würde erst bei der nächsten Zeitscheibe auf die Situation reagiert. Dies bedeutet, dass ggf. die restliche Zeit der Zeitscheibe, in der der Zusammenstoß aufgetreten ist, für die beiden Agenten, wie in Twelkemeier (2013b) (S. 6) beschrieben, verloren ist und diese die Zeit nicht nutzen können.

Der erste Ansatz wird im Folgenden, angelehnt an Guo und Tay (2008) (S. 129), als Event-Driven bezeichnet. Dieser bietet eine hohe Genauigkeit, jedoch ist bei diesem Ansatz die Anzahl der Simulationsschritte abhängig von den Ereignissen, die auftreten. Das bedeutet, dass bspw. bei einer wachsenden Anzahl von Agenten die Dauer zwischen zwei Simulationsschritten abnimmt, bis diese den kleinstmöglichen Zeitschritt erreicht hat.

Bei dem zweiten Ansatz, angelehnt an Guo und Tay (2008) (S. 129) Hard-Time-Step-Driven genannten Ansatz, bleibt die Anzahl der Simulationsschritte konstant, jedoch nimmt mit einer steigenden Anzahl von Agenten die Anzahl der Agenten zu, die in einem Zeitschritt simuliert werden müssen. Der Vorteil bezüglich dem Ressourcenverbrauch von diesem Ansatz gegenüber dem Event-Driven Ansatz macht sich bemerkbar, wenn eine einzelne Simulationsrunde einen hohen Rechenaufwand bedeutet. Gleichzeitig besteht jedoch, wie bereits beschrieben, das Problem, das ggf. Simulationszeit nicht von den Agenten oder anderen Ereignissen genutzt werden kann. Das hat eine Verringerung der Detailtreue zur Folge, da ein Agent, der ggf. nur 50% der Zeitscheiben effektiv nutzen kann, bspw. bei der Bewegung auch nur halb so schnell ist, wie ein Agent, der 100% seiner Zeitscheiben nutzen kann.

Um dieses Problem abzuschwächen, könnte, wie in Twelkemeier (2013a) (S. 3f.) sowie Twelkemeier (2013b) (S. 5) beschrieben, ein hybrider Ansatz gewählt werden, bei dem die Simulation Time-Step-Driven Abläuft. Falls jedoch vor der nächsten Zeitscheibe bereits ein Simulationsschritt aufgrund eines Event-Driven Ereignisses stattfindet, werden die Aktionen mit eingeplant. Dieser Ansatz wird im Folgenden, als Abgrenzung zu dem Hard-Time-Step-

Driven Ansatz, Soft-Time-Step-Driven genannt. Der Soft-Time-Step-Driven Ansatz bringt jedoch nur Vorteile, wenn einige Ereignisse Event-Driven simuliert werden. Sollten alle Ereignisse (Soft-)Time-Step-Driven simuliert werden, existieren zwischen dem Hard- und dem Soft-Time-Step-Driven Ansätzen bei der Ausführung keine Unterschiede.

Eine Zusammenfassung der unterschiedlichen Ansätze ist in Tabelle 3.3 dargestellt.

Detailstufe	**Beschreibung**
Hard-Time-Step-Driven	Die Aktionen werden zu beliebigen Zeitpunkten ausgeführt und die Agenten werden zu festen Zeitpunkten eingeplant.
Soft-Time-Step-Driven	Die Aktionen werden zu beliebigen Zeitpunkten ausgeführt und die Agenten werden zu festen Zeitpunkten eingeplant. Wenn jedoch ein Agent vorher beeinflusst wird, wird dieser zu dem früheren Zeitpunkt eingeplant.
Event-Driven	Die Agenten und Aktionen werden zu beliebigen Zeitpunkten eingeplant.

Tabelle 3.3.: Temporale Detailstufen

3.4.2. Kollisionsskala

Sowohl für die Simulation der Evakuierung des Kinosaals von Klüpfel (2003) als auch des Szenarios des Nachtclubs aus Abschnitt 2.7 existiert die Kollisionsskala, welche die Erkennung und Behandlung von Kollisionen zwischen Agenten und Umgebungsobjekten behandelt. Hierbei kann die Kollisionserkennung, wie bereits in Twelkemeier (2013a) (Kap. 2.1) sowie Twelkemeier (2013b) (Kap. 2.2.2, Kap. 3.2) beschrieben und in Tabelle 3.4 dargestellt, unterschiedlich genau durchgeführt werden.

Detailstufe	**Beschreibung**
Gering	Es wird keine Kollisionserkennung zwischen den Agenten durchgeführt. Die Agenten können sich durcheinander hindurch bewegen.
Mittel	Es wird eine Kollisionserkennung zwischen Agenten lediglich an der Zielposition durchgeführt.
Hoch	Es wird eine Kollisionserkennung zwischen den Agenten durchgeführt.

Tabelle 3.4.: Detailstufen der Kollisionserkennung [angelehnt an Twelkemeier (2013b), S. 11]

Bei der geringen Detailstufe wird keine Kollisionserkennung zwischen den Agenten durchgeführt. Dies kann bspw. für Bereiche mit einer geringen Agentendichte oder für Bereiche, welche nicht von Interesse sind, jedoch simuliert werden müssen, geeignet sein.

Bei der hohen Detailstufe wird eine Kollisionserkennung zwischen den Agenten sowie zwischen den Agenten und anderen Umweltobjekten durchgeführt. Dabei wird sowohl eine Überprüfung auf dem Weg, als auch an dem Zielort durchgeführt. Dies ermöglicht eine sehr genaue Simulation der Bewegung der Agenten, da es zu keiner Überschneidung der Körper der Agenten bei der Bewegung kommen kann.

Eine Mischung aus der hohen und niedrigen Detailstufe dieser Skala ist die Mittlere, welche lediglich die Kollisionsprüfung am Zielort durchführt. Dadurch wird sichergestellt, dass am Ende einer Bewegung die Agenten untereinander und mit anderen Objekten sich nicht auf der gleichen Position befinden. Während der Bewegung können die Agenten sich jedoch durch einander hindurch bewegen.

bezüglich des Rechenaufwands wird davon ausgegangen, dass sich dieser entsprechend der Genauigkeit der unterschiedlichen Detailstufen verhält. Das bedeutet, dass bei einer höheren Genauigkeit der Rechenaufwand ebenfalls höher ist, da bei jeder hier beschriebenen Detailstufe, mit jeder Erhöhung der Genauigkeit zusätzliche Aspekte berechnet werden müssen. So müssen bspw. bei der mittleren Detailstufe lediglich Kollisionen am Zielort erkannt werden, während dies bei der hohen Detailstufe auch für den Weg überprüft werden muss.

Interaktion zwischen Detailstufen

Wenn in der Simulation unterschiedliche Objekte mit unterschiedlichen Detailstufen einer Skala interagieren, können sich die Semantiken der Detailstufen wiedersprechen. Dieses Problem sei beispielhaft anhand der Bewegung von drei Agenten mit den beschriebenen drei Detailstufen der Kollisionserkennung erklärt.

Abbildung 3.3 zeigt die Ausgangssituation. In dieser Situation befindet sich jeder der Agenten jeweils in einem Bereich, welchem eine Detailstufe fest zugeordnet ist. Der Agent A_1 befindet sich im Bereich mit der geringen, A_2 im Bereich mit der mittleren und A_3 befindet sich im Bereich mit der hohen Detailstufe. Alle drei Agenten bewegen sich auf die Position P zu. In diesem Beispiel werden die Agenten ohne Kollisionbehandlung die Position P zeitgleich erreichen und dadurch auf der gleichen Position stehen.

Für die Auswahl der anzuwendenden Detailstufen existieren unterschiedliche Möglichkeiten:

1. Die Detailstufen werden gewechselt, sobald das betreffende Objekt - in diesem Fall der Agent - in den Bereich mit einer anderen Detailstufe wechselt.

2. Für einen Effekt wird eine Detailstufe festgelegt, welche für die Laufzeit der Instanz des Effektes nicht mehr gewechselt wird. Hierbei kann bspw. die Detailstufe des Ursprungsortes verwendet werden.

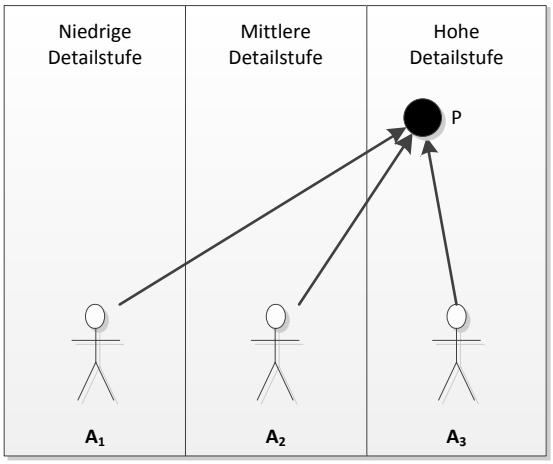


Abbildung 3.3.: Beispielsituation für die Interaktion zwischen Detailstufen

In dem ersten Fall, wenn die Detailstufe gewechselt werden soll sobald die Regeln für ein Objekt oder Effekt eine andere Detailstufe definieren, müssen die Regeln für jeden möglichen Zeitpunkt ausgewertet werden. Dies hat zur Folge, dass alle Effekte zu jedem Zeitpunkt interpoliert und die Regeln ausgewertet werden müssen. Dies stellt einen hohen Mehraufwand dar und untergräbt gleichzeitig die temporalen Detailstufen.

Eine mögliche Lösung ist, wie bereits als zweiten Punkt genannt, die Detailstufe für die Instanz eines Effektes für die komplette Laufzeit festzulegen. So kann bspw. die Detailstufe zum Zeitpunkt und Ort der Ausführung des Effekts für die komplette Dauer des Effekts verwendet werden. Die dadurch verursachte Ungenauigkeit bei der Anwendung der Detailstufen wird bei lange laufenden Effekten und schnellen Wechseln der Detailstufen auffallen.

Eine Verbesserung ist es, die Auswertung der Detailstufe mit der temporalen Detailstufe zu koppeln. So könnten bei Effekten, welche ereignisdiskret simuliert werden, die Detailstufen bei Ereignissen überprüft und ggf. aktualisiert werden. Bei Effekten, welche Time-Step-Driven simuliert werden, würden die Detailstufen nur zu den definierten Zeitpunkten aktualisiert werden.

Wenn die drei Agenten zum gleichen Zeitpunkt den Punkt P erreichen, können außerdem zwischen den Detailstufen Konflikte auftreten. Diese Konflikte sollen anhand der Kollisionserkennung verdeutlicht werden:

Die Detailstufen der Kollisionserkennung, welche bereits vorher beschrieben wurden, weisen zusammengefasst die folgenden Merkmale auf:

Niedrig Keine Kollisionserkennung:

- Die Agenten können sich während der Bewegung durch einander hindurch bewegen.
- Die Agenten können sich auf der gleichen Position befinden.

Mittel Kollisionserkennung lediglich am Zielort:

- Die Agenten können sich während der Bewegung durch einander hindurch bewegen.
- Die Agenten können sich nicht auf der gleichen Position befinden.

Hoch Kollisionserkennung auf dem Weg und am Zielort

- Die Agenten können sich nicht während der Bewegung durch einander hindurch bewegen.
- Die Agenten können sich nicht auf der gleichen Position befinden.

In dem Beispielszenario aus Abbildung 3.3 tritt die Situation ein, dass alle drei Agenten sich auf der gleichen Position P befinden. Dies ist jedoch eigentlich von den Merkmalen der hohen und mittleren Detailstufe ausgeschlossen. Die niedrige Detailstufe lässt jedoch genau diese Situation zu.

Hierbei ist zunächst unklar, wie das Verhalten der Simulation bei Kollisionen von zwei oder mehreren Agenten mit unterschiedlichen Detailstufen ist, bzw. welche Effekte mit welchen Detailstufen simuliert werden.

Um diese Situation zu lösen, existieren unter anderem die folgenden Möglichkeiten:

1. Jede Detailstufe berücksichtig lediglich die Objekte, welche in dieser aktuell berechnet werden. Dies bedeutet, dass Agenten unterschiedlicher Detailstufen sich zu einem Zeitpunkt auf der gleichen Position befinden können, auch wenn dies eigentlich in einzelnen Detailstufen nicht möglich sein soll.

2. Es wird immer genau die Detailstufe verwendet, welche zu dem Zeitpunkt an dem Ort des Effekts bzw. Objekts gilt. Um jeweils die genaue Detailstufe zu bestimmen, müsste die Simulation, wie bereits vorher in diesem Abschnitt beschrieben, für jeden Zeitpunkt interpoliert und die Detailstufe bestimmt werden. Dies ist zu aufwändig und garantiert keine Lösung des Problems, da für unterschiedliche Objekte an einem Ort zu einer Zeit auch unterschiedliche Detailstufen gelten könnten.

3. Es werden die Objekte bzw. Effekte mit einer geringeren Detailstufe bei höheren Detailstufen berücksichtigt. Dies ist in Abbildung 3.4 mit drei Effekten E_1, E_2 und E_3 dargestellt.

Die schwarzen Rechtecke stellen die ursprüngliche Detailstufe des Effekts dar. Der Pfeil zeigt an, für welche Detailstufen dieser Effekt berücksichtigt wird. So werden bspw. für die höchste Detailstufe alle Objekte bzw. Effekte berücksichtigt. Bei der mittleren Detailstufe werden dann lediglich die Objekte bzw. Effekte E_1 und E_2 der niedrigen und mittleren Detailstufe berücksichtigt. Dies entspricht einer partiellen Erhöhung der Detailstufe. Dies bedeutet, dass bspw. bei der Kollisionserkennung einer höheren Detailstufe

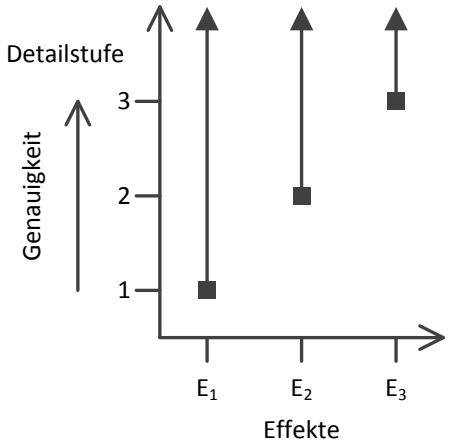


Abbildung 3.4.: Partielle Erhöhung der Detailstufe

auch alle Objekte bzw. Effekte von geringeren Detailstufen der gleichen Skala berücksichtigt werden. Für die zusätzlich berücksichtigten Effekte werden jedoch nicht die höheren Detailstufen ausgeführt. Das bedeutet, dass bspw. bei der Kollisionserkennung zwar Kollisionen zwischen einem Agent einer hohen und einer niedrigen Detailstufe erkannt werden, jedoch nicht zwischen zwei Agenten einer niedrigen Detailstufe, auch wenn einer oder beide der Agenten der niedrigen Detailstufe bei einer höheren Detailstufe berücksichtigt werden.

Diese vorgestellten Möglichkeiten sind jedoch nicht allgemeingültig, da das genaue Vorgehen von der Skala abhängig ist. Aus diesem Grund, muss jede Skala das genaue Verhalten in diesen Situationen definieren.

3.4.3. Skala der Gerüchteausbreitung

Die Ausbreitung von Gerüchten, welche bspw. in dem Nachtclub-Szenario verwendet werden können, können ebenfalls unterschiedlich genau simuliert werden.

Die eine Möglichkeit ist, die Ausbreitung des Gerüchts nicht zu simulieren. Das hat zur Folge, dass die Wahrnehmung des Gerüchts als Auslöser des Fluchtverhaltens nicht aktiv wird und sich die Charakteristik der Flucht der Menschenmenge ggf. anders entwickelt.

Die andere Möglichkeit ist, die Ausbreitung des Gerüchts ggf. über ein Infizierungsmodell zu simulieren. Dabei wird die Information des Gerüchts so modelliert, dass ein Agent sich mit dieser Information „infizieren“ kann. Ein solches Modell wird bspw. auch in Gong und Xiao (2007) verwendet. Für die Verteilung von Informationen werden Epidemie-Modelle auch in Verteilten Systemen verwendet (siehe Tannenbaum und van Steen (2007), S. 170ff.).

Für die Modellierung der Infizierung der Agenten mit dem Gerücht, können unterschiedliche Zustände der Agenten, wie in Abbildung 3.5 dargestellt, verwendet werden.

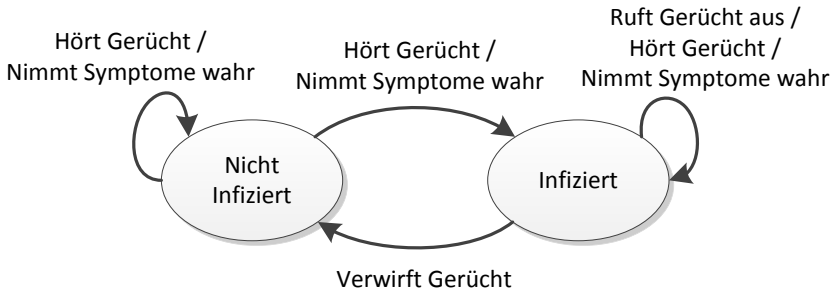


Abbildung 3.5.: Zustände der Agenten bezüglich der Infizierung mit dem Gerücht

Dabei kann eine Person, die nicht mit dem Gerücht „infiziert“ ist, sich mit diesem über zwei Wege infizieren. Der erste Weg ist, dass der Agent die Symptome des Reizstoffs wahr nimmt und dann selber das Gerücht entwickelt.

Der zweite Weg ist, dass dieser das Gerücht von einem anderen Agenten hört. In beiden Fällen muss der Agent jedoch dieses Gerücht auch als wahr annehmen. Ein Agent kann auch entscheiden, dass dieses Gerücht nicht stimmt. In dem Fall verbleibt der Agent in dem Ausgangszustand.

Eine mögliche Erweiterung ist, dass ein Agent, der das Gerücht als wahr angenommen hat, dieses nach einer gewissen Zeit wieder als nicht wahr annimmt. Dabei sollte jedoch auch die Situation, in der sich der Agent befindet, beachtet werden.

3.4.4. Skala der Gasausbreitung

Nach dem Versprühen des Pfeffersprays kommt es zu einer Ausbreitung des Wirkstoffs im Raum. Durch diese Ausbreitung wirkt der Wirkstoff auf immer mehr Personen in dem Raum

Detailstufe	Beschreibung
Niedrig	Keine Ausbreitung
Mittel	Sofortige Ausbreitung mit einer Gleichverteilung
Hoch	Spatio-temporale Ausbreitung mit Hilfe der Diffusion

Tabelle 3.5.: Detailstufen der Gasausbreitung

und führt zu Symptomen, wie dem Anschwellen der Atemwege (vgl. Wirth und Strauch (2012) S.256), bei diesen.

Diese Simulation der Ausbreitung des Wirkstoffs kann, wie in Tabelle 3.5 zusammengefasst dargestellt ist, auf unterschiedliche Arten durchgeführt werden. So kann die Ausbreitung überhaupt nicht simuliert werden. Dies hat zur Folge, dass die Personen nicht dem Wirkstoff ausgesetzt werden und dadurch auch nicht fliehen. Vom Rechenaufwand her ist dies die Detailstufe mit dem geringsten Aufwand.

Eine zweite Möglichkeit der Simulation der Ausbreitung, ist die Ausbreitung ohne den zeitlichen Verlauf zu simulieren. Dies bedeutet, dass die endgültige Ausbreitung ohne zeitliche Dauer, d.h. sofort, eintritt.

Die letzte Möglichkeit, die untersucht werden soll, ist die Ausbreitung nach den tatsächlichen Regeln der Diffusion zu simulieren, welche auch bereits in Abschnitt 2.2 beschrieben wurden.

Komplexität der Detailstufen

Diese drei Detailstufen weisen unterschiedliche Komplexitäten bezüglich des Berechnungsaufwands auf. Der geringste Rechenaufwand wird für die Detailstufe mit keiner Ausbreitung benötigt, da hierbei keine weiteren Berechnungen durchgeführt werden müssen.

Bei der Detailstufe mit der sofortigen Ausbreitung des Wirkstoffs muss lediglich berechnet werden, wie bei einer homogenen Verteilung des Wirkstoffs die Konzentration ist. Da diese Konzentration - unter der Annahme, dass es keine Ein- und Ausgänge gibt, durch welche der Wirkstoff den Raum verlassen kann - zum Schluss an jedem Punkt existiert, kann die durchschnittliche Konzentration für jeden beliebigen Messpunkt in dem Bereich dieser Detailstufe angenommen werden.

Die dritte Detailstufe weist den höchsten Rechenaufwand auf, da die Differentialgleichung 2.5 aus Abschnitt 2.2 gelöst werden muss. Dies kann bspw. über numerische Verfahren durchgeführt werden.

Die Reihenfolge der Genauigkeit der einzelnen Detailstufen ist analog zu den benötigten Ressourcen. Die Detailstufe mit dem geringsten Ressourcenbedarf bietet den geringsten Detailgrad, da keinerlei Informationen zu der Ausbreitung berechnet werden.

Die mittlere Detailstufe berechnet die durchschnittliche Wirkstoffkonzentration sowie den Bereich der Ausbreitung. Diese vernachlässigt jedoch die tatsächliche spatio-temporale Charakteristik der Ausbreitung.

Der zeitliche Verlauf der Ausbreitung wird lediglich von der hohen Detailstufe berechnet.

Berechnung der Ausbreitung in der hohen Detailstufe

Das Verfahren der Berechnung der Diffusion, mit den Grundlagen aus Abschnitt 2.2, wird anhand eines Beispiels, welches in Abbildung 3.6 dargestellt ist, für den zweidimensionalen Fall erklärt. In Shankar (2012) ist dies sowohl für den ein- als auch für den zweidimensionalen Fall in Matlab umgesetzt.

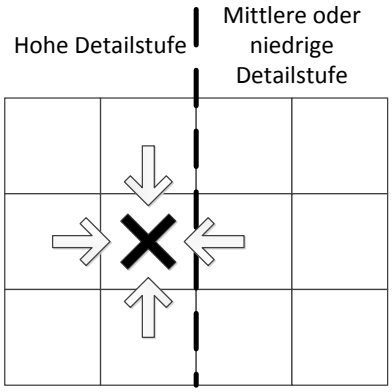


Abbildung 3.6.: Beispiel für die Berechnung der Ausbreitung des Gases. Die gestrichelte Linie stellt die Grenze zwischen zwei Detailstufen dar. Berechnet werden soll die Konzentration für den Messpunkt, welcher durch das schwarze Kreuz dargestellt ist.

Für die Berechnung der hohen Detailstufe wird die Umwelt in $dx * dy$ große Felder eingeteilt. Diese Felder stellen die einzelnen Messpunkte bezüglich der Konzentration dar. Außerdem existiert, wie in Abschnitt 2.2 bereits beschrieben, für den Diffusionskoeffizienten die Funktion D, welche sowohl von der Zeit als auch von der Quell- und Zielposition abhängt. Diese Funktion benötigt die beiden Positionen, da der Diffusionskoeffizient zwischen zwei Messpunkten definiert werden muss. Der Grund ist, dass bspw. bei Wänden der Diffusionskoeffizient ein anderer, als bspw. im Raum, sein muss.

Die Aktualisierung der Werte erfolgt dann für den jeweils zu berechnenden Zeitpunkt über die Gleichung 3.1. Die Herleitung dieser Formel ist in Abschnitt A.1 zu finden.

$$
\begin{aligned}
c(x,y,t+dt) =& c(x,y,t) + dt \\
& \Bigg(D(x+dx,y,x,y,t) * \frac{c(x+dx,y,t) - c(x,y,t)}{(dx)^2} \\
& - D(x,y,x-dx,y,t) * \frac{c(x,y,t) - c(x-dx,y,t)}{(dx)^2} \\
& + D(x,y+dy,x,y,t) * \frac{c(x,y+dy,t) - c(x,y,t)}{(dy)^2} \\
& - D(x,y,x,y-dy,t) * \frac{c(x,y,t) - c(x,y-dy,t)}{(dy)^2} \Bigg)
\end{aligned}
\tag{3.1}
$$

Hierbei kann man in Abbildung 3.6 sehen, dass sich für den zweidimensionalen Fall die Änderungen der Konzentration aus der Änderungen zu den jeweiligen Nachbarn aus der von-Neumann-Nachbarschaft zusammensetzt.

Hierbei gibt bspw. der erste Teilterm aus Gleichung 3.1 $dt * D(x+dx,y,x,y,t) * \frac{c(x+dx,y,t)-c(x,y,t)}{(dx)^2}$ die Veränderung vom rechten Nachbar an und der zweite Teilterm $dt * D(x,y,x-dx,y,t) * \frac{c(x,y,t)-c(x-dx,y,t)}{(dx)^2}$ die Veränderung vom linken Nachbar. Analog dazu geben der dritte und vierte Teilterm die Veränderung vom unteren und dem oberen Nachbarn an. Die Änderung des Wertes des Feldes (x,y) liefert die Summe der, mit dt und dem Diffusionskoeffizienten gewichteten, Werte aus den vier Teiltermen.

Stabilität der numerischen Lösung

Für die numerische Lösung der Differentialgleichung 3.1 wurde ein explizites Verfahren verwendet. Dieses Verfahren hat den Vorteil, dass es recht einfach umgesetzt werden kann und pro Iteration weniger Rechenaufwand als ein implizites Verfahren benötigt (Wendt (2009), S.94ff.). Da in dieser Arbeit davon ausgegangen wird, dass die Schrittgröße der Simulation und damit das Δt maximal 100ms beträgt, wird das explizite Verfahren verwendet, da eine Schrittgröße von 100ms oder weniger zu vielen Simulationsschritten und dadurch zu vielen Iterationen bei der Berechnung der Diffusionsgleichung führt.

Jedoch besitzt das explizite Verfahren den Nachteil, dass das Verfahren nicht immer stabil ist. Um dies zu überprüfen, kann mit Hilfe der von Neumann Stabilitäts-Analyse überprüft werden, unter welchen Bedingungen die Gleichung stabil ist. Für die Durchführung der Analyse sei an dieser Stelle auf Unterabschnitt A.1.1 verwiesen. Die Analyse ergibt die Bedingung aus Gleichung 3.2.

Von \ Nach	Niedrig	Mittel	Hoch
Niedrig		×	×
Mittel	✓		✓
Hoch	✓	✓	

Tabelle 3.6.: Mögliche Interaktionen zwischen den Detailstufen der Gasausbreitungsskala. ✓ bedeutet, dass eine Interaktion möglich ist und × bedeutet, dass eine Interaktion nicht möglich ist.

$$dt \leq \frac{1}{2D\frac{1}{(dx)^2} + \frac{1}{(dy)^2}} \tag{3.2}$$

Bei dieser Analyse wurde die Vereinfachung getroffen, dass der Diffusionskoeffizient konstant ist und nicht, wie in Gleichung 3.1 eine Funktion, die abhängig von dem Ort und der Zeit ist. Dadurch kann diese Analyse lediglich für eine Abschätzung des schlechtesten Falles bezüglich des Diffusionskoeffizienten verwendet werden. Der schlechteste Fall für die Stabilität liegt bei dem größten Diffusionskoeffizienten. Wenn für den Diffusionkoeffizienten, wie in dieser Arbeit, ein maximaler Wert angegeben werden kann, ist dies jedoch keine Einschränkung, da dadurch trotzdem eine Abschätzung des schlimmsten Falls durchgeführt werden kann.

Interaktion zwischen Detailstufen

Eine Interaktion zwischen den Detailstufen kann bei dieser Skala auftreten, wenn sich der Wirkstoff in einen Bereich ausbreitet, der durch eine andere Detailstufe simuliert wird oder wenn sich die Detailstufe durch die definierten Bedingungen in einem Bereich ändert, der eine Konzentration des Wirkstoffs aufweist.

In dem ersten Fall, wo sich der Wirkstoff in eine andere Detailstufe ausbreitet, existieren die in Tabelle 3.6 dargestellten Situationen der Interaktion zwischen Detailstufen.

Eine Ausbreitung von der niedrigen Detailstufe in eine andere Detailstufe ist nicht möglich, da in der niedrigeren Detailstufe keine Ausbreitung o.ä. berechnet wird. Eine ggf. vorhandene Konzentration des Wirkstoffs kann nur durch eine andere Detailstufe berechnet worden sein, welche jetzt für diesen Bereich aufgrund der geänderten Situation nicht mehr verwendet wird. Dabei handelt es sich dann jedoch um den später beschriebenen Fall, dass eine Beeinflussung von Detailstufen untereinander durch den Wechsel der Bereiche der Detailstufen auftritt.

Bei einer Beeinflussung der mittleren Detailstufe durch die hohe Detailstufe sollen keine Moleküle in der Umwelt entfallen oder neue entstehen. Um dies zu erreichen existieren zwei Möglichkeiten:

1. Die hohe Detailstufe beachtet nicht die Bereiche von anderen Detailstufen für die Berechnung der Konzentration und Ausbreitung des Gases. Dies hat jedoch zur Folge, dass es keine Beeinflussung zwischen den Detailstufen gibt und dadurch auch keine Ausbreitung über die Grenzen von den Detailstufen hinweg[1].

2. Die andere Möglichkeit ist, dass die hohe Detailstufe an die mittlere oder an die niedrige Detailstufe die Position und die Änderung der Konzentration übergibt, welche durch eine Aktualisierung verursacht wurde.

Die zweite Möglichkeit soll anhand des Beispiels aus Abbildung 3.7 erklärt werden.

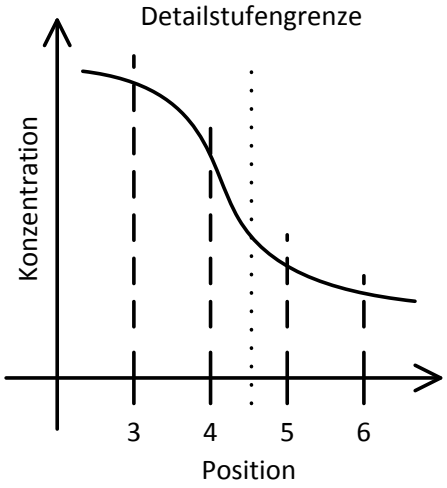


Abbildung 3.7.: Beispiel für die Berechnung der Konzentrationsänderung im zweidimensionalen Raum.

Die Gleichung 2.2 zur Berechnung der Werte im eindimensionalen Raum besteht aus zwei, mit der zeitlichen Veränderung und dem jeweiligen Diffusionskoeffizienten, gewichteten Differenzenquotienten. Dies ist einmal die Veränderung zwischen x-1 und x und einmal die Veränderung zwischen x und x+1. Bei einer Berechnung von zwei Feldern, welche nebeneinander liegen und mit der Gleichung 2.2 aktualisiert werden, wird der Differenzenquotient

[1] Einzig wenn sich die Bereiche der Detailstufen ändern, sodass ein Bereich, welcher eine Konzentration des Gases aufweist, vorher in der hohen Detailstufe berechnet wurde und jetzt in mittlerer Detailstufe simuliert wird, existiert eine Ausbreitung über die alten Detailstufen-Grenzen hinweg.

zwischen diesen beiden Feldern bei beiden Berechnungen jeweils verwendet. Dies wird an dem Beispiel der Felder 4 und 5 aus dem Beispiel in Abbildung 3.7 deutlich:

$$c(4,t+1) = c(4,t) + dt * \left(D(5,4,t)\frac{c(5,t)-c(4,t)}{1^2} + D(3,4,t)\frac{c(3,t)-c(4,t)}{1^2} \right) \quad (3.3)$$

$$c(5,t+1) = c(5,t) + dt * \left(D(6,5,t)\frac{c(6,t)-c(5,t)}{1^2} + D(4,5,t)\frac{c(4,t)-c(5,t)}{1^2} \right) \quad (3.4)$$

Da $D(5,4,t) = D(4,5,t)$ gelten muss, damit keine Moleküle bei der Berechnung entstehen oder vernachlässigt werden, ist der Unterschied zwischen den beiden Quotienten lediglich das Vorzeichen, sodass dieser bei dem einen Feld hinzu kommt und bei dem anderen Feld die Konzentration um den Wert verringert wird.

Wenn die Grenze für die hohe Detailstufe so verläuft, dass die Felder Nummer 3 und 4 mit der hohen Detailstufe simuliert werden und ab Feld 5 alle Felder mit einer geringeren Detailstufe, muss der Wert von Feld 5 nach einer Aktualisierung von Feld 4 um diesen gewichteten Differenzenquotienten geändert werden:

$$\Delta c(5,t) = dt * \left(D(5,4,t) * \frac{c(4,t)-c(5,t)}{1^2} \right) \quad (3.5)$$

In dem zweidimensionalen Fall ist dies analog dazu. Hierbei muss für die Funktion D jedoch $D(x_1,y_1,x_2,y_2,t) = D(x_2,y_2,x_1,y_1,t)$ gelten.

Bei der Beeinflussung der hohen Detailstufe durch die mittlere Detailstufe werden die Werte der mittleren Detailstufe im Grenzbereich zu der hohen Detailstufe als normale Nachbarwerte verwendet. Jedoch führt eine Änderung eines Wertes der hohen Detailstufe ebenfalls zu einer Aktualisierung der mittleren Detailstufe, wenn sich Moleküle von oder zu dieser bewegt haben.

Bei den letzten beiden Möglichkeiten der Beeinflussung handelt es sich um die Beeinflussung der niedrigen Detailstufe durch eine höhere.

In dem Fall, dass die mittlere Detailstufe eine Ausbreitung bis zu einem Bereich einer niedrigen Detailstufe aufweist, findet keine Ausbreitung in der niedrigen Detailstufe statt. Lediglich wenn sich die hohe Detailstufe in diesen niedrigen Bereich ausbreitet, wird der unmittelbare Grenzbereich geändert. Dadurch wird wieder erreicht, dass die Anzahl der Moleküle in der Welt gleich bleibt.

In dem Fall, dass sich der Bereich einer Detailstufe ändert, wird der Bereich jeweils, wie in der neuen Detailstufe definiert, simuliert. Welche Detailstufe ursprünglich diesen Bereich simuliert hat, ist für die zukünftige Simulation des Bereichs irrelevant.

4. Konzept

Nachdem die Szenarien analysiert und die Skalen mit den Detailstufen aufgestellt sind, wird im Folgenden die tatsächliche Modellierung der jeweiligen Bereiche der Simulation beschrieben. Dabei wird in diesem Kapitel der Fokus auf die allgemeine Simulation gelegt. Dazu zählt die technische Struktur der Simulation, sowie die Verfahren zur Umsetzung der Detailstufen und zur Auswahl der Detailstufen.

4.1. Simulationsstruktur

Bei der Simulationsstruktur handelt es sich um die technische Betrachtung der Umsetzung der Simulation. Das beinhaltet einen groben Überblick über die Einteilung der Simulation in Komponenten, den Ablauf einer Simulationsrunde und die Umsetzung der Effekt-Module. Dieses Konzept bzw. die Simulation baut auf Twelkemeier (2013b) auf.

4.1.1. Architektur

Die Simulation ist in zwei Hauptbereiche aufgeteilt: Den Simulationskern, welcher die Simulationslogik enthält, sowie die Oberfläche zur Steuerung der Simulation.

Die Oberfläche der Simulation wird, wie in Abbildung 4.1 dargestellt, über die Schnittstelle *Core* an den Simulationskern angebunden. Die Oberfläche kann auch, wenn ein Simulationslauf mit mehreren Durchläufen automatisch durchgeführt werden soll, gegen eine Steuerungslogik ohne Oberfläche ausgetauscht werden. Über die *Core*-Schnittstelle können die Simulation verwaltet und der Status abgefragt werden.

In dem Simulationskern werden alle Anfragen von der *Core*-Schnittstelle durch eine Fassade an den *Manager* delegiert. Dieser enthält die Logik zum Erzeugen von neuen Simulationen, sowie das Steuern einer laufenden Simulation.

Der *Manager* sorgt für die Ausführung der Agenten, welche in der *Agent*-Komponente definiert werden. Des Weiteren enthält der *Manager* ebenfalls die Steuerung der temporalen Skala. Hierfür fragt dieser über die Schnittstelle *LoD Controller* die jeweils zu verwendende Detailstufe für die temporale Skala vom *LoD Controller* ab.

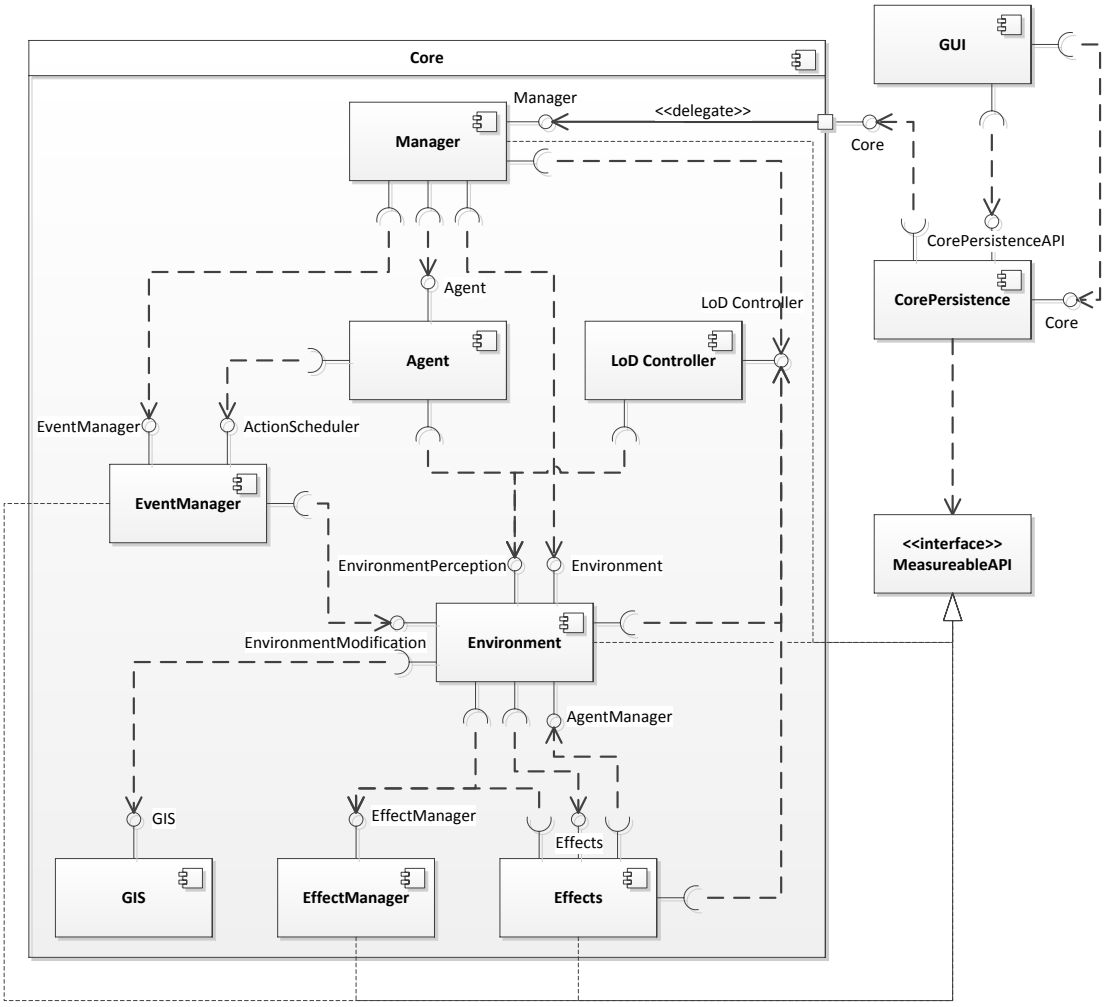


Abbildung 4.1.: Technischer Aufbau der Simulation [aufbauend auf Twelkemeier (2013b) S. 15]

Die Aktionen, welche von den Agenten eingeplant werden, werden über die Schnittstelle *ActionScheduler* an den *EventManager* übergeben. Dieser verwaltet die Aktionen und führt die, welche in einer Simulationsrunde ausgeführt werden sollen, aus.

Der *LoD Controller* verarbeitet die Regeln, welche die Zuordnung zwischen der Situation in der Simulation und der Detailstufe pro Skala beschreiben. Anhand dieser Zuordnung liefert dieser dann die jeweils zu verwendende Detailstufe für die Skala. Das bedeutet, andere Komponenten können über die Schnittstelle *LoD Controller* die zu verwendende Detailstufe für ihre Skala abfragen. Hierfür greift der *LoD Controller* auf das *Environment* zu, um die aktuelle Situation der Simulation abzufragen.

Das *Environment* enthält die Daten der Umwelt, sowie die Steuerung der Effekt-Module. Wenn Aktionen durch den *EventManager* ausgeführt werden, können diese über die Schnittstelle *EnvironmentModification* Änderungen in der Umwelt auslösen. Diese werden je nach Situation an die jeweiligen Effekt-Module weitergegeben, welche dann die Effekte erzeugen. Die Effekte von allen Effekt-Modulen werden von dem *EffektManager* verwaltet.

Die spatiale Struktur der Umwelt wird durch die *GIS* Komponente bereitgestellt. Dass bedeutet, diese überführt die Beschreibung der Umwelt in die Datenstrukturen der Simulation.

Um die Messungen bezüglich bspw. der Performance der Simulation durchzuführen, existiert orthogonal zu der Simulationslogik die Schnittstelle *MeasureableAPI*. Über diese Schnittstelle kann eine Komponente Messdaten bereitstellen, welche durch die *CorePersistence* Komponente gespeichert werden.

Die *CorePersistence* Komponente ist als Decorator (Gamma (2005), S.175 ff.) für den Simulationskern modelliert und bietet daher zusätzlich zu dem Interface *CorePersistenceAPI* auch das Interface *Core* vom Simulationskern. Dadurch können ohne Änderungen an der Oberfläche bzw. dem Aufrufer des Simulationskerns zusätzlich zu den, durch die *MeasureableAPI* bereit gestellten Messdaten, ebenfalls alle Daten, die bspw. an die Oberfläche gegeben werden, wie die Positionsdaten der jeweiligen Agenten, abgespeichert werden. Dadurch ist eine Durchführung der Simulation ohne interaktive Bedienung möglich.

Die Auswertung der Messdaten erfolgt nicht durch die *CorePersistence* zur Simulationszeit, sondern nach dem jeweiligen Simulationslauf über eine separate Evaluierungs-Komponente. Hierdurch ist es möglich, nachträglich den Schwerpunkt einer Auswertung in gewissen Grenzen zu verlagern, sollten bei den Messergebnissen Effekte aufgetreten sein, welche dies zur weiteren Analyse dieser Effekte notwendig machen.

4.1.2. Simulationskreislauf

In Abbildung 4.2 wird der grobe Ablauf einer Simulationsrunde dargestellt. Der Ablauf wird dabei hauptsächliche von dem *Manager* gesteuert.

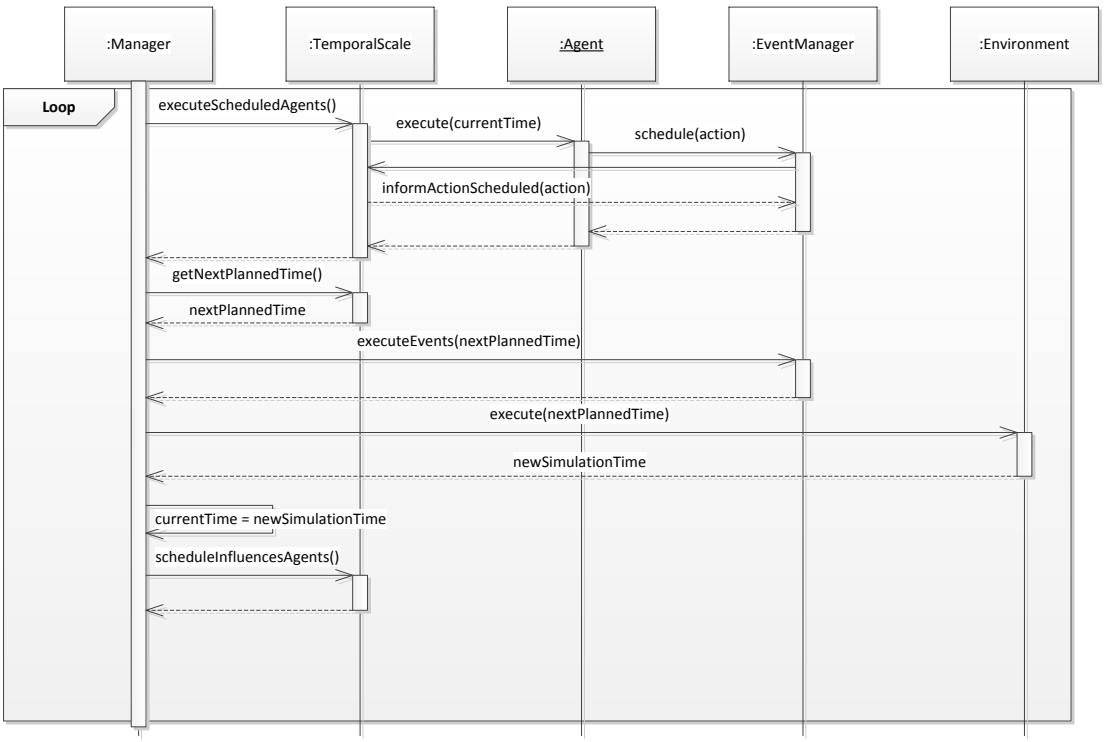


Abbildung 4.2.: Grober Ablauf einer Simulationsrunde [angelehnt an Twelkemeier (2013b), S. 20]

Die Simulation wird mit der Simulationszeit *currentTime = 0* initialisiert.

Der *Manager* sendet an die *TemporalScale* die Nachricht, dass diese alle eingeplanten Agenten ausführt. Diese führt dabei alle Agenten aus, welche unter Berücksichtigung der temporalen Detailstufe ausgeführt werden sollen. Dies kann bedeuten, dass nicht alle Agenten, welche für eine spätere Ausführung eingeplant wurden, ausgeführt werden, wenn die temporale Detailstufe für diese Agenten noch keine Ausführung vorsieht.

Die Agenten, die ausgeführt werden, bekommen die jeweils aktuelle Simulationszeit mitgeteilt. Eine Information, wie lange es bis zur nächsten Ausführung dauert, d.h. wann diese Agenten das nächste mal ausgeführt werden, kann zu diesem Zeitpunkt den Agenten nicht mitgeteilt werden. Dieser Zeitpunkt ist u.a. von allen Aktionen, die die Agenten eingeplant haben und gerade einplanen abhängig.

Die Aktionen, die die Agenten einplanen, werden bei dem *EventManager* eingeplant. Dieser ist für die Verwaltung der Aktionen zuständig und informiert gleichzeitig die *TemporalScale* über die eingeplanten Aktionen. Die *TemporalScale* führt dann ein Einplanen der Aktionen unter Berücksichtigung der temporalen Detailstufen durch.

Nachdem die Agenten ausgeführt wurden, berechnet die *TemporalScale* den nächsten geplanten Zeitpunkt, bis zu welchem simuliert werden kann. Dafür werden von der *TemporalScale* sowohl die eingeplanten Agenten, als auch die eingeplanten Aktionen berücksichtigt.

Auf Basis dieser geplanten zukünftigen Simulationszeit, lässt der *Manager* den *EventManager* alle Aktionen, die bis einschließlich dieses Zeitpunkts eingeplant sind, ausführen. Diese Aktionen führen dann zu Effekten in der Umwelt, welche i.d.R. jedoch bis zu diesem Zeitpunkt nur für die Ausführung eingeplant, jedoch noch nicht tatsächlich simuliert werden.

Im Anschluss an die Ausführung aller geplanten Aktionen wird das *Environment* ausgeführt. Diesem wird ebenfalls der geplante Zeitpunkt der Simulation übergeben. Dieser Zeitpunkt stellt für das *Environment* lediglich einen spätesten Zeitpunkt dar, bis zu welchem dieser simulieren kann. Das *Environment* kann auch bereits vor diesem Zeitpunkt den Simulationsschritt unterbrechen, wenn bspw. ein außerplanmäßiges Ereignis, wie die Kollision von zwei Agenten, eingetreten ist.

Das *Environment* führt bei der Simulation auch die Effekt-Module aus und meldet beeinflusste Agenten an den *Manager*.

Nach der Ausführung des *Environment* meldet dieser die tatsächliche Simulationszeit, bis zu welcher dieser simuliert hat, zurück an den *Manager*. Dieser übernimmt diesen Zeitpunkt als neue, aktuelle Simulationszeit und plant die beeinflussten Agenten in der *TemporalScale* ein.

Diese Reihenfolge von Schritten wird dann solange durchgeführt, bis die Simulation unterbrochen worden oder der Endzeitpunkt der Simulation erreicht ist.

4.1.3. Umsetzung der Effektmodule

Die Effektmodule werden von dem *Environment* gesteuert. Dazu besitzen die Effektmodule zwei Methoden. Über die erste Methode kann das *Environment* abfragen, was die maximale Simulationszeit für das Effektmodul ist. Wenn es bspw. zu einer Kollision bei der Simulation der Bewegung eines Agenten kommt, kann diese Zeit kürzer als die für diese Simulationsrunde vorgesehene Zeit sein. Diese Zeit ist für das Effektmodul bindend. Dieses muss dann bis zu dieser Zeit simulieren können.

Von allen Zeiten der Effektmodule nimmt das *Environment* die kleinste maximale Zeit und signalisiert allen Effektmodulen diese kleinste maximale Zeit, sodass diese sich bis zu diesem

Zeitpunkt aktualisieren können. Diese Zeit wird von dem *Environment* auch an den *Manager* zurückgemeldet.

4.2. Umsetzung der Detailstufen

Die Detailstufen werden als Erweiterung für die Umwelt umgesetzt. Dabei werden im Weiteren die Detailstufen der Temporalen-, Kollisions- und Gasausbreitungsskala beschrieben. Die Skala der Gerüchteausbreitung wird im Folgenden aufgrund des Umfangs der Arbeit nicht weiter behandelt.

Wie in Abbildung 4.3 dargestellt, verwendet die Umwelt mehrere Effekt-Module. Diese Effekt-Module bilden die einzelnen Skalen ab. Dabei kann ein Effekt-Modul auch mehrere Skalen enthalten. Jedoch kann eine Skala nicht in zwei Effekt-Modulen enthalten sein.

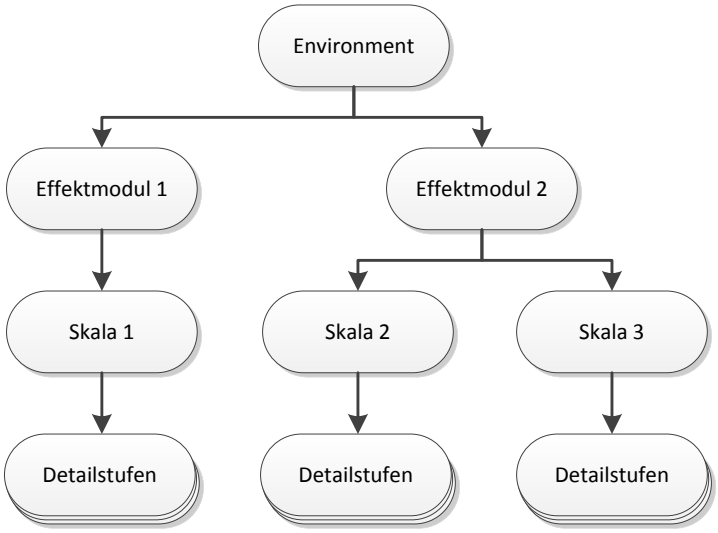


Abbildung 4.3.: Umsetzung der Detailstufen

Die Effekt-Module enthalten auch entsprechend alle Detailstufen für die Skalen, welche in dem jeweiligen Modul enthalten sind.

Dadurch, dass die Effekt-Module als Erweiterung der Umwelt modelliert sind, ist es möglich, weitere Skalen und Effekte ohne große Änderungen hinzuzufügen.

Lediglich die temporale Skala bildet eine Ausnahme bei diesem Ansatz. Hierbei handelt es sich nicht um eine Skala, welche lediglich die Umwelt betrifft, sondern um eine Skala, welche die gesamte Simulation betrifft. Aus diesem Grund wird diese Skala separat von dem sogenannten Scheduler umgesetzt.

4.2.1. Temporale Skala

In dieser Simulation bedeutet ein Zeitpunkt t=10, dass der Zeitpunkt 10 bereits stattgefunden hat und die Simulation unmittelbar vor dem Zeitpunkt t=11 steht. Das bedeutet es können keine Ereignisse mehr in t=10 ausgeführt werden.

Wie in Unterabschnitt 3.4.1 bereits beschrieben, sollen für die temporale Skala drei unterschiedliche Detailstufen umgesetzt werden. Dies ist die Event-Driven, Soft-Time-Step-Driven und die Hard-Time-Step-Driven Detailstufe. Die Umsetzung der Skala ist bereits in Twelkemeier (2013b) (Kap. 3.1) beschrieben.

Bei der Umsetzung der Skala existieren zwei unterschiedliche Schritte: Zuerst wird für jede Detailstufe berechnet, bis zu welchem Zeitpunkt die Simulation in dieser Detailstufe simulieren kann. Im Anschluss daran wird als Basis dieser Zeitpunkte der Zeitpunkt für die komplette Simulation berechnet, bis zu welchem die Simulation ausgeführt werden kann. Dafür wird aus den drei Zeitpunkten der einzelnen Detailstufen der zeitlich nächste Zeitpunkt ausgewählt.

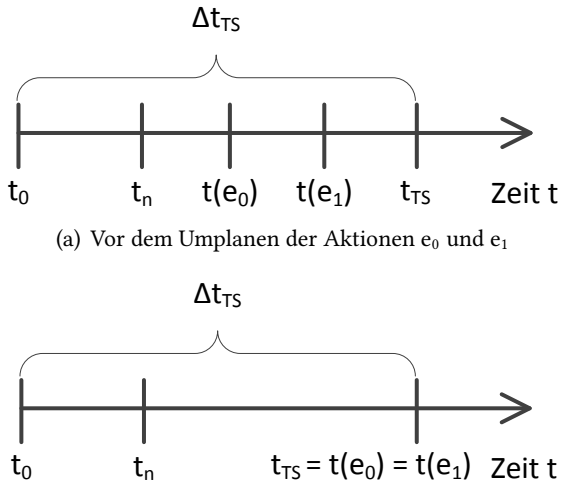
Im Weiteren werden die Verfahren der einzelnen Detailstufen für die Berechnung der Zeitpunkte, bis zu welchem die Simulation aus Sicht der jeweiligen Detailstufe simuliert werden kann, vorgestellt. Im Folgenden sei t_n die aktuelle Simulationszeit, $\Delta t = 1$ die kleinste mögliche zeitliche Änderung und Δt_{TS} die Dauer eines Time-Steps.

Hard-Time-Step-Driven

Für die Berechnung des Zeitpunktes der Hard-Time-Step-Driven Detailstufe wird jede Aktion, welche sich in dem Time-Step $(t_n, t_n - (t_n \; mod \; \Delta t_{TS}) + \Delta t_{TS}]$ befindet, für den Zeitpunkt $t_n - (t_n \; mod \; \Delta t_{TS}) + \Delta t_{TS}$ eingeplant. Dies bedeutet, dass alle Aktionen, die Hard-Time-Step-Driven simuliert werden und in dem Time-Step ausgeführt werden sollen, wie in Abbildung 4.4 dargestellt, jeweils am Ende des Time-Steps ausgeführt werden. Der Grund ist, dass während des Time-Steps ggf. weitere Aktionen von Event-Driven Aktionen eingeplant werden können, welche ebenfalls Hard-Time-Step-Driven simuliert werden können sollen. Wenn die Aktionen zu Beginn eines Time-Steps ausgeführt würden, müssten die Aktionen jeweils für den nächsten Time-Step eingeplant werden.

Wenn Aktionen für einen Time-Step eingeplant werden, ist diese Entscheidung endgültig und wird auch durch eine Änderung der Situation nicht mehr geändert. Diese Entscheidung wird jedoch jeweils nur für alle Aktionen innerhalb des aktuellen Time-Steps getroffen. Alle späteren Aktionen werden zu diesem Zeitpunkt noch als Event-Driven angenommen.

Falls innerhalb von diesem Time-Step Agenten für die Ausführung und dadurch für das Einplanen von Aktionen eingeplant wurden, werden diese zum Zeitpunkt $t_{TS} - \Delta t$ eingeplant.



(a) Vor dem Umplanen der Aktionen e_0 und e_1

(b) Nach dem Umplanen der Aktionen e_0 und e_1

Abbildung 4.4.: Umplanen von Aktionen bei den Time-Step-Driven Detailstufen

Dadurch ist sichergestellt, dass alle Agenten, welche in diesem Time-Step ausgeführt werden sollen, auf einmal ausgeführt werden. Zu diesem Zeitpunkt kann kein weiterer Agent beeinflusst werden, sodass dieser innerhalb von diesem Time-Step noch ausgeführt werden muss. Außerdem ist dadurch die zeitliche Distanz zwischen dem Einplanen und der Ausführung von Aktionen, die sofort ausgeführt werden sollen, minimal.

Der nächstmögliche Zeitpunkt, bis zu welchem diese Detailstufe simulieren kann, hängt von dem aktuellen Zeitpunkt und ob Agenten oder Aktionen für diesen Time-Step dieser Detailstufe eingeplant sind, ab. Der Zeitpunkt berechnet sich wie folgt:

1. Wenn $t_n < t_{TS} - \Delta t$, dann ist der Zeitpunkt $t_{TS} - \Delta t$.
2. Wenn $t_n \geq t_{TS} - \Delta t$, dann ist der Zeitpunkt der späteste Zeitpunkt der Simulation. Dies ist die für diesen Simulationslauf definierte maximale Simulationszeit. Nach dieser Simulationszeit endet der Simulationslauf automatisch.

Soft-Time-Step-Driven

Bei der Soft-Time-Step-Driven Detailstufe werden, genau wie bei der Hard-Time-Step-Driven Detailstufe, die Aktionen und Agenten innerhalb eines Time-Steps jeweils für das Ende des Time-Steps eingeplant. Auch bei der Berechnung des nächstmöglichen Simulationszeitpunkts unterscheidet sich nicht von der Hard-Time-Step-Driven Detailstufe.

Der einzige Unterschied liegt in dem Verhalten bei der tatsächlichen Ausführung von Aktionen und Agenten. Während dies bei der Hard-Time-Step-Driven Detailstufe jeweils nur genau zu den definierten Zeitpunkten geschieht, werden bei der Soft-Time-Step-Driven Detailstufe sowohl die Agenten als auch die Aktionen, wenn die ursprünglich vom Agenten festgelegte Ausführungszeit kleiner oder gleich der aktuellen zu simulierenden Zeit ist, auch in Simulationsschritten ausgeführt, die durch eine Event-Driven Aktion oder einen Event-Driven Agenten ausgelöst wurde.

Event-Driven

Bei der Event-Driven Detailstufe werden die Agenten und Aktionen zu genau dem Zeitpunkt ausgeführt, für welchen diese auch ursprünglich eingeplant wurden. Daher existieren für die Ermittlung des nächstmöglichen Zeitpunkts aus Sicht dieser Detailstufe zwei unterschiedliche Situationen:

Die erste Situation ist, dass zwischen der aktuellen Simulationszeit t_n und dem Zeitpunkt der nächsten geplanten Aktion mehr als der kleinstmögliche Zeitschritt Δt liegt. Diese Situation wird auch in Abbildung 4.5 dargestellt, wobei $t(e_0)$ der geplante Zeitpunkt der Aktion e_0 ist und die Aktion e_0 zeitlich die nächste geplante Aktion darstellt.

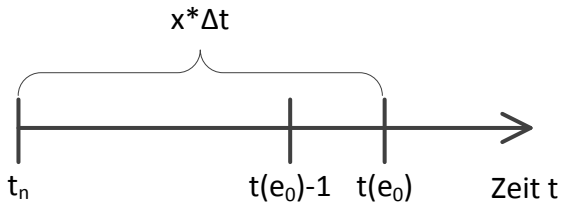


Abbildung 4.5.: Event-Driven Aktionen mit einem Abstand von $x * \Delta t$

In diesem Fall kann bis zum Zeitpunkt $t(e_0) - \Delta t$ simuliert werden, da bis dahin keine sonstigen Aktionen geplant sind.

Die zweite Situation ist, dass der aktuelle Simulationszeitpunkt $t_n = t(e_0) - \Delta t$ unmittelbar vor der geplanten Ausführung einer Event-Driven geplanten Aktion liegt.

In diesem Fall wird im nächsten Schritt die Aktion e_0 ausgeführt und es kann daher, wie in Abbildung 4.6 dargestellt, bis zur zeitlich übernächsten Aktion simuliert werden. Dabei wird, analog zu der ersten beschriebenen Situation, bis unmittelbar vor die Aktion e_1 simuliert.

Durch dieses Vorgehen ist sichergestellt, dass kein Ereignis übergangen wird oder dass ein Ereignis zu früh ausgeführt wird.

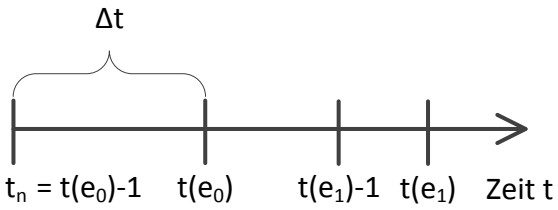


Abbildung 4.6.: Event-Driven Aktionen mit einem Abstand von Δt

Alle drei Detailstufen haben gemein, dass, wenn diese keine weiteren eingeplanten Aktionen besitzen, als Simulationszeit der größtmögliche Zeitpunkt in der Simulation verwendet wird. Dadurch wird sichergestellt, dass, wenn alle Detailstufen in dieser Situation sind, alle laufenden Effekte bis zum Ende simuliert werden. Sollten bei der Simulation der Effekte weitere Agenten beeinflusst oder Aktionen eingeplant werden, wird die Simulation zu diesem Zeitpunkt unterbrochen und dort ein neuer Simulationsschritt eingefügt.

Wie am Anfang dieses Kapitels bereits beschrieben, wird dann jeweils von allen drei Detailstufen der geringste bzw. zeitlich nächste Simulationszeitpunkt als nächste Simulationszeit ausgewählt.

Bei der Simulation der Effekte können dann die folgenden Situationen auftreten:

- Ein Effekt ist abgeschlossen.
- Ein außerplanmäßiges Ereignis, wie bspw. eine Kollision von zwei Agenten, ist aufgetreten.
- Der geplante Zeitpunkt ist erreicht.

In dem Fall, wo ein Effekt abgeschlossen ist, wird der durch den Effekt betroffene Agent als beeinflusst angenommen. In dieser Situation ist es abhängig davon, welche temporale Detailstufe aktuell für den dadurch beeinflussten Agenten vorgegeben ist. Wenn dieser Event-Driven simuliert wird, wird der Simulationsschritt beendet und mit dem Zeitpunkt ein neuer Simulationsschritt gestartet. Wenn der Effekt Time-Step-Driven simuliert wurde, wird die Simulation bis zum nächsten Time-Step ausgeführt und dann ein neuer Simulationsschritt ausgeführt. In diesem kann der Agent dann Aktionen einplanen.

Das gleiche Verhalten tritt bei den außerplanmäßigen Ereignissen auf. Wenn ein solches Ereignis auftritt, wird der beeinflusste Agent abhängig von der aktuellen Situation in der Simulation entweder Time-Step-Driven oder Event-Driven eingeplant. Dadurch kann der Agent ggf. für die Zeitspanne bis zum nächsten Time-Step untätig sein.

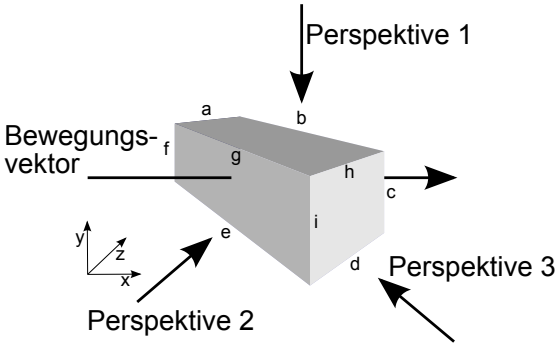
Wenn der ursprünglich geplante Zeitpunkt erreicht wurde, wird ein neuer Simulationsschritt eingeführt, außer dies ist die definierte Endzeit der Simulation.

4.2.2. Kollisionsskala

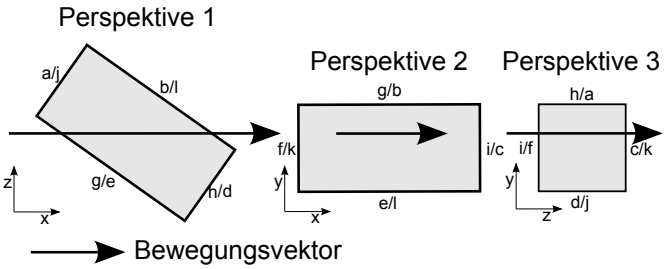
Wie bereits in Unterabschnitt 3.4.2 beschrieben, werden in dieser Simulation drei unterschiedliche Detailstufen für die Simulation der Kollisionsskala umgesetzt.

Bei der niedrigen Detailstufe wird keine Kollisionserkennung oder -behebung umgesetzt. Dadurch können sich die Agenten sowohl durcheinander als auch durch Wände oder andere Objekte bewegen.

Für die mittlere als auch die hohe Detailstufe wird der Algorithmus zur Kollisionserkennung aus Twelkemeier (2012) (S. 18ff.) verwendet. Dieser basiert auf der Schnittpunktberechnung bzw. dem Separating Axis Theorem (Metanet Software Inc (2011) sowie Eberly (2008)). Dafür werden die 3D Objekte, wie bspw. Wände, jeweils als drei zweidimensionale Flächen betrachtet, indem jeweils eine Dimension weg gelassen wird. Dies ist in Abbildung 4.7 beispielhaft dargestellt.



(a) 3D Umgebungsobjekt [Twelkemeier (2012) S.18]



(b) Aufteilung in drei 2D Rechtecke [Twelkemeier (2012) S.19]

Abbildung 4.7.: Aufteilung eines 3D Quaders in drei 2D Flächen

Aus der Bewegung des Agenten wird dann ein Bewegungsvektor gebildet, bei welchem für alle Seiten der Flächen die Schnittpunkte mit den Kanten berechnet werden. Dabei muss

beachtet werden, dass sich der Schnittpunkt auf dem Bewegungsvektor zwischen dem Start- und Zielpunkt der Agentenbewegung befindet und sich auch im Bereich der jeweiligen Kante des Rechtecks befindet.

Der Schnittpunkt im dreidimensionalen Raum kann dann berechnet werden, indem die entsprechenden Komponenten von zwei Schnittpunkten aus dem zweidimensionalen Raum zusammengefügt werden. Dabei müssen die beiden Schnittpunkte jedoch einer Zeile in der Tabelle 4.1 entsprechen.

Schnittpunkt 1		**Schnittpunkt 2**	
P1	g/e	P3	i/f
P1	b/l	P3	c/k
P1	h/d	P2	i/c
P1	a/j	P2	f/k
P2	g/b	P3	h/a
P2	e/l	P3	d/j

Tabelle 4.1.: Zuordnung der Schnittpunkte [Twelkemeier (2012) S. 19]

Wie in Twelkemeier (2012) beschrieben, ist dann der, von der Startposition nächstgelegene, Kollisionspunkt der Punkt, an dem die erste Kollision zwischen dem Agenten und dem Umgebungsobjekt auftritt.

Dieses Verfahren erkennt jedoch lediglich Kollisionen zwischen Agenten und Objekten, wie Wänden. Außerdem erkennt dieses Verfahren keine Kollisionen, wenn der Agent, wie in Abbildung 4.8 dargestellt, sehr nah an einem Objekt, wie bspw. einer Wand, entlang geht. Um dies ebenfalls zu erkennen, müsste die Bewegung des Agenten nicht nur als einzelner Bewegungsvektor in den Algorithmus einfließen, sondern es müsste die tatsächliche Ausdehnung des Agenten mit eingebracht werden. Da dies jedoch einen erhöhten Rechenaufwand zur Folge hätte, soll dies an dieser Stelle nicht umgesetzt werden.

Da die Agenten als Zylinder modelliert sind, muss für die Kollisionserkennung zwischen diesen jeweils nur die Distanz zwischen den Mittelpunkten der Agenten berechnet werden. Wenn diese kleiner als die Summe der Radien der beiden Agenten ist, hat eine Kollision stattgefunden. Diese Kollisionserkennung wird lediglich im zweidimensionalen Raum durchgeführt.

Bei der mittleren Detailstufe werden die Kollisionen lediglich am Ende der Bewegung aufgelöst. Dagegen werden bei der hohen Detailstufe die Kollisionen auf dem Weg erkannt, indem jeder mögliche Simulationsschritt auf Kollisionen überprüft wird. Dadurch wird, im Falle einer Kollision, nur bis zum Zeitpunkt unmittelbar vor der Kollision simuliert.

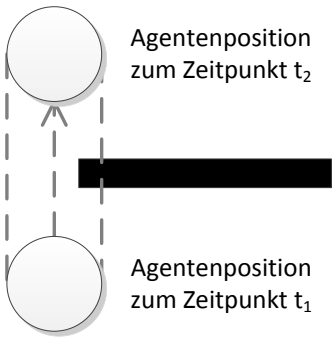


Abbildung 4.8.: Kollision, welche nicht erkannt wird

Um dies zu erreichen, wird in einer Transaktion die Zeit berechnet, bis zu welcher die Bewegung der Agenten ausgeführt werden kann, ohne dass es zu einer Kollision kommt. Dafür ist die Simulation der Effektmodule, wie bereits in Unterabschnitt 4.1.3 beschrieben, in zwei Schritte aufgeteilt.

Für die Bestimmung der maximal zu simulierenden Zeit wird in der Transaktion für jede mögliche Simulationszeit überprüft, ob es zu einer Kollision kommt. Sollte es zu einer Kollision kommen, wird der Zeitpunkt unmittelbar davor als maximale Zeit angegeben, die dieses Effektmodul simulieren kann. Nach der Bestimmung der maximalen Zeit wird die Transaktion zurückgerollt.

Wenn die Simulation jetzt bis zu einem Punkt in der Zeit berechnet wird, kann die Position des Agenten direkt aktualisiert werden. Im Anschluss daran werden mögliche Kollisionen, die ggf. durch andere als Bewegungseffekte verursacht wurden, gelöst.

Dieses Vorgehen soll an dem folgenden Beispiel verdeutlicht werden: Ein Agent bewegt sich von seiner Position für 10ms mit einer Geschwindigkeit in eine Richtung. Nach 6ms kommt es zu einer Kollision mit einem anderen Agenten. Diese wird erkannt, da die hohe Detailstufe für jeden der 10 Zeitschritte überprüft, ob es zu einer Kollision kommt. Nachdem festgestellt wurde, dass nach 6ms eine Kollision auftreten wird, wird als maximale Zeit, die simuliert werden kann, 5ms bestimmt. Wenn die Simulation jetzt 5ms simuliert, wird, nachdem die Agenten ihre Bewegung durchgeführt haben, die eventuell vorhandenen Kollisionen behoben. Die Prüfung und Behebung von Kollisionen nach der Bewegung der Agenten muss durch diese Detailstufe trotzdem durchgeführt werden, da diese bspw. durch andere Effekte auftreten können, welche einen Agenten verschieben.

4.2.3. Gasausbreitung

Für die Simulation der Ausbreitung des Gases werden die drei, bereits im Unterabschnitt 3.4.4 beschriebenen, Detailstufen umgesetzt:

Niedrig Keine Simulation der Ausbreitung.

Mittel Sofortige Ausbreitung und setzen der Konzentration im gesamten Bereich auf den Durchschnitt aller Felder.

Hoch Berechnung der Konzentration mit Hilfe der Diffusion.

Bei der Simulation der einzelnen Detailstufen greifen diese alle auf das gleiche Datenmodell zu. Dadurch entfällt eine Transformation der Zustände der einzelnen Detailstufen, wenn für einen Bereich die Detailstufe gewechselt wird. Das Datenmodell orientiert sich dabei an den Anforderungen bzw. dem Umfang, welcher für die höchste Detailstufe benötigt wird.

In diesem Fall wird die Umwelt in $\Delta x * \Delta y$ große Felder eingeteilt. Jedes Feld spiegelt dabei einen Messpunkt dar.

Niedrige Detailstufe

Bei der niedrigen Detailstufe werden die eventuell vorhandenen Konzentrationen bzw. Werte der Felder nicht geändert. Das bedeutet, wenn ein Feld in diesem Bereich eine Konzentration aufweist, wird dieser Wert durch die Detailstufe nicht geändert.

Einzig, wenn eine andere Detailstufe eine Änderung eines Feldes im Grenzbereich zu dieser Detailstufe sendet, wird diese Änderung übernommen. So kann bspw. die hohe Detailstufe die Änderungen bezüglich der Werte, die durch die Berechnung der hohe Detailstufe geändert werden müssen, an die niedrige Detailstufe weitergeben.

Mittlere Detailstufe

Bei der mittleren Detailstufe werden die Werte der Felder, die sich im Bereich dieser Detailstufe befinden, auf den durchschnittlichen Wert über alle Felder in dem Bereich gesetzt. Das bedeutet, dass Änderungen von Werten in diesem Bereich automatisch zu einer Aktualisierung der Werte in diesem Bereich führt. Dadurch wird sichergestellt, dass alle Felder in einem Bereich mit einer mittleren Detailstufe den gleichen bzw. den durchschnittlichen Wert aufweisen.

$$c(x, y, t+1) = \frac{\sum_{n \in N} c(x(n), y(n), t)}{|N|} \tag{4.1}$$

Der neue Wert eines Feldes kann mit der Gleichung 4.1 berechnet werden, wobei die Menge N alle zusammenhängenden Felder der mittleren Detailstufe enthält. Wenn zwei Bereiche mit der mittleren Detailstufe existieren, welche jedoch nicht miteinander verbunden sind, werden diese unabhängig voneinander berechnet.

Hohe Detailstufe

Wie bereits beschrieben, werden bei der hohen Detailstufe die Simulation der Ausbreitung und der Konzentration des Wirkstoffs über die Diffusion berechnet. Diese Berechnung erfolgt jedoch lediglich im zweidimensionalen Raum, um den Ressourcenaufwand zu verringern.

Für die Berechnung wird die Differentialgleichung 4.2 aus der Gleichung 2.5 für den zweidimensionalen Raum entwickelt und verändert.

$$\frac{\delta c}{\delta t} = \frac{\delta}{\delta x}\left(D\frac{\delta c}{\delta x}\right) + \frac{\delta}{\delta y}\left(D\frac{\delta c}{\delta y}\right) \tag{4.2}$$

Nach Laurien und Oertel (2009) (S. 65) sowie Wendt (2009) (S. 88ff.) werden für die numerische Lösung der Gleichung 4.2 die Forward Difference und Rearward Difference Methode verwendet. Daraus ergibt sich für die numerische Lösung die Gleichung 4.3. Hierbei geben dt, dx und dy jeweils die Änderung der Zeit, die Änderung in X-Richtung und die Änderung in Y-Richtung an. Eine Herleitung der Gleichung ist in Abschnitt A.1 beschrieben.

$$\begin{aligned} c(x,y,t+dt) =& c(x,y,t) + dt \\ & \Bigg(D(x+dx,y,x,y,t) * \frac{c(x+dx,y,t) - c(x,y,t)}{(dx)^2} \\ & - D(x,y,x-dx,y,t) * \frac{c(x,y,t) - c(x-dx,y,t)}{(dx)^2} \\ & + D(x,y+dy,x,y,t) * \frac{c(x,y+dy,t) - c(x,y,t)}{(dy)^2} \\ & - D(x,y,x,y-dy,t) * \frac{c(x,y,t) - c(x,y-dy,t)}{(dy)^2} \Bigg) \end{aligned} \tag{4.3}$$

Dabei gibt $D(x_1, y_1, x_2, y_2, t)$ den Diffusionskoeffizienten zum Zeitpunkt t zwischen den beiden Punkten (x_1, y_1) und (x_2, y_2) an. Dieser ist abhängig von dem Ort, da dadurch bspw. eine Diffusion durch Wände verhindert werden kann. Bei einer Wand wird in dieser Simulation der Diffusionskoeffizient auf 0 gesetzt. Dabei muss jedoch $D(x_1, y_1, x_2, y_2, t) = D(x_2, y_2, x_1, y_1, t)$ gelten, da ansonsten ggf. Moleküle nur in eine Richtung diffundieren würden.

Für den Diffusionskoeffizienten für die Ausbreitung von Gasen wird von Baehr und Stephan (2008) (S. 80) ein Wert zwischen $5 * 10^{-6}$ und $10^{-5} \frac{m^2}{s}$ genannt. In dieser Arbeit wird ein Wert

verwendet, welcher deutlich höher ist, da sich gezeigt hat, dass durch die fehlende Modellierung von Gerüchten, sowie der Vernachlässigung von Luftströmungen und Temperaturunterschieden, welche in einer Diskothek vorhanden sind, die Ausbreitung deutlich zu langsam ist. Daher wird in dieser Arbeit für den Diffusionskoeffizienten ein Wert von $5,5 * 10^{-2} \frac{m^2}{s}$ verwendet.

Die Berechnung erfolgt iterativ und nur für den Bereich bzw. die Felder, die in dem Bereich für die hohe Detailstufe liegen.

In dem Fall, wo der Wert eines Feldes aktualisiert werden soll, welches sich direkt an der Grenze zu einer anderen Detailstufe befindet, wird für die Berechnung des neuen Wertes die Grenze kurzzeitig überschritten, um den Wert des Nachbarfelds abzufragen. Im Anschluss daran, wird die Veränderung für das Nachbarfeld, wie in Abschnitt 3.4.4 bereits beschrieben, berechnet und diese Information an die Detailstufe des Nachbarfelds weitergegeben. Die Berechnung der Änderung des Werts des Nachbarfelds wird über die Gleichung 4.4 bzw. 4.5 durchgeführt. Hierbei geben x_n bzw y_n die x bzw y Position des Nachbarfelds n an.

$$\Delta c(x_n, y, t) = dt * D(x_n, y, x, y, t) * \frac{c(x_n, y, t) - c(x, y, t)}{(dx^2)} \tag{4.4}$$

$$\Delta c(x, y_n, t) = dt * D(x, y_n, x, y, t) * \frac{c(x, y_n, t) - c(x, y, t)}{(dy^2)} \tag{4.5}$$

Für diese Simulation wird die Umwelt in $x_n = y_n = 0.2m$ große Quadrate für die Berechnung der Diffusion eingeteilt. Auf Basis dieser Werte, kann mit Hilfe der von-Neumann-Stabilitätsanalyse eine Abschätzung der maximal möglichen Schrittgröße für Simulationsläufe mit dieser Detailstufe durchgeführt werden. Dafür wird die Ungleichung 3.2 aus Abschnitt 3.4.4 verwendet.

$$dt \leq \frac{1}{2 * 5,5 * 10^{-2} \frac{m^2}{s} \frac{1}{(0,2m)^2} + \frac{1}{(0,2m)^2}} \tag{4.6}$$

$$dt \leq 0,1818s \tag{4.7}$$

Durch einsetzen der Werte in die Ungleichung ergibt sich eine maximale Schrittweite von 181ms. Das bedeutet, dass diese Detailstufe maximal bei den hier verwendeten Werten, für eine Schrittweite von 181 ms stabil ist und verwendet werden kann.

4.3. Dynamische Auswahl der Detailstufe

Für die Auswahl der Detailstufe wird, wie in Abschnitt 3.2.1 beschrieben, eine Zuordnung zwischen der Skala und Detailstufe sowie dem Zustand der Simulation durchgeführt.

Diese Zuordnung wird, wie in Abbildung 4.9 dargestellt, durch einen LoD Controller hergestellt. Dieser Controller gibt auf Anfrage die Detailstufe für eine bestimmte Skala zurück.

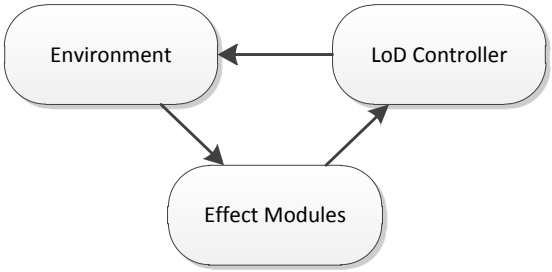


Abbildung 4.9.: Auswahl der Detailstufe

Als Parameter für eine solche Anfrage benötigt der LoD Controller folgende Werte:

- Effekt-Typ
- Spatiale Koordinaten des Effekts

Mit Hilfe dieser Werte und der definierten Zuordnungen liefert der LoD Controller die zu verwendende Detailstufe. Die Zuordnung wird dabei über Regeln durchgeführt, welche die in Unterabschnitt 3.2.1 beschriebenen Bedingungen unterstützen:

- Temporale Bereiche
- Spatiale Bereiche
- Dichte der Agentenpopulation

Diese einzelnen Bedingungen können, wie ebenfalls in Unterabschnitt 3.2.1 beschrieben, kombiniert werden. Die Definition der Regeln erfolgt über eine eigene Sprache. Die Regeln müssen zum Startzeitpunkt der Simulation bekannt sein. Ein Ändern der Regeln zur Simulationszeit ist nicht vorgesehen.

Sollten bei der Auswahl der Detailstufe mehrere Zuordnungen möglich sein, da ein Effekt sich bspw. in der temporalen Skala über zwei Bereiche erstreckt, wird immer der Quellort bzw. -zeitpunkt verwendet.

4.3.1. Regeln

Um eine flexible Möglichkeit zu schaffen, die Regeln der Simulation unabhängig von dem Simulationssystem ändern zu können, sollen die oben genannten Punkte in einer Regelsprache

umgesetzt werden. Durch diese Regelsprache ist es möglich, dass ein Domänenexperte die Regeln definiert.

Diese bestehen, wie in der Auflistung 4.1 dargestellt, aus einer Zuordnung von einem Tupel, bestehend aus der Skala und der Detailstufe, zu einer Bedingung. Dabei ist die *Skala* ein eindeutiger Name der Skala und die *Detailstufe* in der Auflistung 4.1 eine eindeutige Nummer, die die Detailstufe in der Skala identifiziert.

Die Bedingungen in der Auflistung 4.1 können dabei beliebig stark geschachtelt werden. Bei dieser Schachtelung, können die einzelnen Bedingungen über die logischen Operatoren UND und ein nicht-exklusives ODER verbunden werden.

```
1  (Skala, Detailstufe) ::= <Bedingung>
2  <Bedingung>          ::= <Bedingung> <BooleanOperation> <Bedingung>
3                           | 'Position' <Operation> Vektor
4                           | 'Zeit' <Operation> Nummer
5                           | 'Agentendichte' <Operation> Nummer
6                           | 'Aktionstyp' <GleichOperation> String;
7  <GleichOperation>    ::= 'EQ' | 'NE';
8  <Operation>          ::= <GleichOperation>
9                           | 'LT' | 'LE' | 'GE' | 'GT';
10 <BooleanOperation>   ::= 'AND' | 'OR';
```

Auflistung 4.1: Syntax der Regeln

Die Schlüsselwörter *Vektor*, *Nummer* und *String* bilden die unterschiedlichen Typen von Konstanten ab. So ist die *Nummer* eine reelle Zahl, der *Vektor* ein dreidimensionaler Vektor aus reellen Zahlen und der *String* eine beliebige Zeichenfolge.

Die Zeichenfolgen 'Position', 'Zeit', 'Agentendichte' und 'Aktionstyp' sind Variablen in den Regeln. Diese werden von dem Regelsystem automatisch durch die entsprechenden Werte ersetzt.

So liefert die Variable Position automatisch einen dreidimensionalen Vektor mit der aktuellen Position des Objekts zurück. Das Objekt ist hierbei abhängig vom Kontext der Regelauswertung. Das bedeutet, wenn die Regel für einen Agenten ausgewertet wird, ist das Objekt der Agent und die Position die aktuelle Position des Agenten in der Umwelt. Wenn die Regel für eine Aktion ausgewertet wird, wird als Position der Aktion, wenn vorhanden, die Quellposition verwendet. Dies ist bspw. bei einer Bewegungs-Aktion die aktuelle Position des Agenten, der diese Aktion eingeplant hat. Wenn eine Quellposition nicht existiert, wird die Position verwendet, an welcher die Aktion ausgeführt wird. Falls die Aktion jedoch über keine Position verfügt, wird auch keine Position zurück geliefert. Dadurch wird jeder Vergleich als falsch evaluiert.

Operation	Zahl/Vektor	String
LT	<	nicht definiert
LE	<=	nicht definiert
EQ	==	Wahr, wenn die Zeichenketten gleich sind (case sensitiv)
NE	!=	wahr, wenn die Zeichenketten nicht gleich sind (case sensitiv)
GE	>=	nicht definiert
GT	>	nicht definiert

Tabelle 4.2.: Semantik der Vergleichsoperationen

Die Variablen „Zeit“ und „Agentendichte“ liefern eine *Nummer* als Resultat zurück. Dabei liefert die Zeit-Variable die aktuelle Simulationszeit zum Zeitpunkt der Auswertung der Regel und die Agentendichte entsprechend die Agentendichte an der Position des Objekts zurück. Dabei ist das Objekt wieder von dem Kontext der Auswertung abhängig.

Die letzte Variable, die Aktionstyp-Variable, gibt den Namen der Aktion zurück, für die gerade die Regeln ausgewertet werden. Falls die Regel gerade nicht für eine Aktion, sondern bspw. für das einplanen eines Agenten ausgewertet wird, wird dieser Teil der Bedingung als falsch evaluiert.

Für den Vergleich der Variable mit der Konstante existieren die Operationen, welche die in Tabelle 4.2 dargestellte Semantik aufweisen. Dabei gibt die Spalte „Zahl/Vektor“ die Semantik bei einem Vergleich zwischen einer Variable, die eine Zahl zurück liefert und einer Zahl, sowie zwischen einer Variable, die einen Vektor zurück liefert und einem Vektor an. Die Spalte „String“ gibt die Semantik beim Vergleich zwischen einer Variable, die einen String zurück liefert und einer String-Konstante an.

Bei komplexeren Datenstrukturen, wie bspw. einem Vektor, wird die Operation jeweils auf alle Komponenten einzeln angewendet und über eine logische UND-Operation verknüpft. So wird bspw. ein Vektor $v_1^T = (1, 2, 3)$, welcher mit dem Vektor $v_2^T = (3, 4, 5)$ über die Operation EQ verglichen wird, in die folgende boolesche Funktion übersetzt:

$$(1\ EQ\ 3)\ AND\ (2\ EQ4)\ AND\ (3\ EQ\ 5)$$

Durch diese Definition der Regeln ist nicht ausgeschlossen, dass keine Widersprüche bei den definierten Regeln auftreten können. So kann zu einem Zeitpunkt bspw. eine Regel R_1 für eine Skala eine Detailstufe D_1 definieren und gleichzeitig eine andere Regel R_2 für die gleiche Skala eine andere Detailstufe D_2 definieren.

Da die Detailstufen durch Zahlen abstrahiert werden, wird bei einem Widerspruch die natürliche Ordnung auf diesen Zahlen verwendet. Das bedeutet, dass, wenn nach den Regeln mehrere Detailstufen in einer Situation angewendet werden könnten, die Detailstufe mit der höchsten Zahl verwendet wird. Dies ermöglicht es gezielt Regeln zu überschreiben. So kann bspw. für die komplette Simulation für eine Skala eine Detailstufe mit einer niedrigen Zahl als Standard verwendet werden, während in einzelnen Situationen dies durch eine Detailstufe mit einem höheren Wert überschrieben werden kann.

Dieses Verhalten muss vom Regelersteller auch verwendet werden, um sicherzustellen, dass in jeder Situation für jede Skala eine Detailstufe definiert ist. Sollte in einer Situation für eine Skala keine zu verwendende Detailstufe definiert sein, ist dies eine undefinierte Situation in der Simulation und würde zum Abbruch führen.

Berechnung der Agentendichte

Für die Auswahl der Detailstufen kann unter anderem die Anzahl der Agenten in der Nähe als Bedingung verwendet werden. Hierfür wird die Agentendichte berechnet. Diese ist ein Indiz dafür, wie stark die Agenten in einem Bereich gedrängt sind.

Für die Berechnung der Dichte in einem Bereich, wird ein Kreis mit dem Mittelpunkt P und dem Radius r verwendet. Der Fläche des Kreises wird die Summe der, auf dem Boden projizierten Standflächen, der Agenten zum Verhältnis gesetzt. Als Standfläche der Agenten wird ebenfalls ein Kreis verwendet. So ergibt sich die Agentendichte wie folgt:

$$\rho(P, r) = \frac{\sum_{a \in Agenten(P)} r(a)^2 * \pi}{r^2 * \pi}$$

$Agenten(P)$:= Menge der Agenten innerhalb des durch P und r definierten Kreises

$r(a)$:= Radius der projizierten Standfläche des Agenten a

Hierbei kann es jedoch zu folgenden Ungenauigkeiten kommen:

1. Ein Agent wird zu der Menge der Agenten, die sich im Referenzkreis befinden, gezählt, wenn sich der Mittelpunkt seiner projizierten Standfläche im Kreis befindet. Dadurch kann es, wie in Abbildung 4.10 dargestellt, vorkommen, dass sich Teile der projizierten Standfläche des Agenten nicht im Referenzkreis befinden, jedoch trotzdem für die Berechnung der Agentendichte berücksichtigt werden. Diese überschüssige Fläche, die in Abbildung 4.10 schraffiert dargestellt ist, müsste aus der Summe der Flächen der Agenten gerechnet werden. Da diese Kennzahl jedoch nur einen Anhaltspunkt darstellen

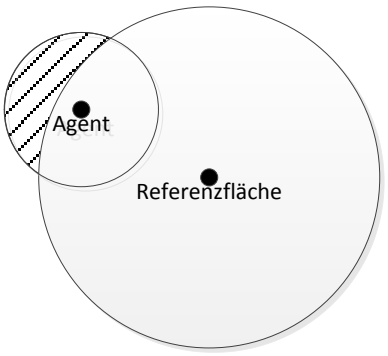


Abbildung 4.10.: Teil der projizierten Standfläche eines Agenten befindet sich nicht im Referenzkreis

soll, wird an dieser Stelle darauf verzichtet, um zusätzlichen Berechnungsaufwand zur Simulationszeit einzusparen.

2. Durch die unterschiedlichen Detailstufen können sich ggf. mehrere Agenten an der gleichen Position befinden. Dies kann die Kennzahl entsprechend auch auf über 100% erhöhen.

3. An Bereichen, wo nicht die ganze Fläche durch Agenten besetzt werden kann, kann die Kennzahl niedrig sein, obwohl die Agenten sehr eng zusammen stehen. Dies kann bspw. eine Wand sein, wo sich die Agenten stauen und eine hohe Agentendichte herrscht. Da der Bereich der Wand, wo sich keine Agenten befinden können, nicht aus der Referenzfläche heraus gerechnet wird, fällt die Kennzahl in diesen Situationen niedriger aus.

5. Realisierung

Auf Basis des Konzeptes der Simulation wird im Folgenden Beschrieben, wie die Simulation umgesetzt wurde. Dabei wird der Schwerpunkt der Beschreibung auf der Auswahl der Detailstufen sowie die Messmethode gelegt.

5.1. Framework

Die Simulation wurde auf Basis von Java 7 umgesetzt, wobei als Framework OSGi verwendet wurde. Durch die Verwendung von OSGi als Plattform, wurden die Komponenten der Simulation auf die OSGi Module abgebildet. Dabei wurden die Schnittstellen jeweils als eigene OSGi Module umgesetzt und die Implementierungen der Komponenten ebenfalls als eigene OSGi Module. Diese Umsetzung ist in Abbildung 5.1 beispielhaft für drei Komponenten mit zwei Schnittstellen dargestellt. Für eine Einführung in OSGi und weitergehende Informationen sei an dieser Stelle auf OSGi Alliance (2012) oder auf Hall u. a. (2011) verwiesen.

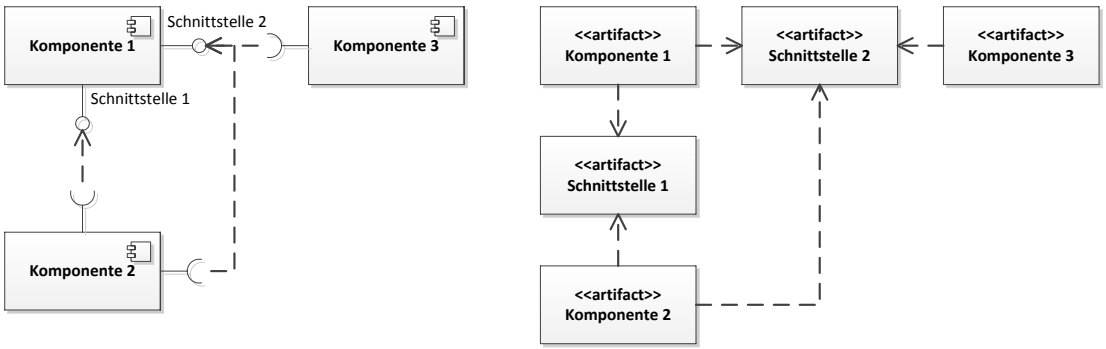


Abbildung 5.1.: Umsetzung der Komponenten auf die OSGi Module

Dabei existieren lediglich Abhängigkeiten auf Module, die Schnittstellen enthalten und nicht auf Module, die die Implementierung beinhalten. Dadurch ist sichergestellt, dass unterschiedliche Implementierungen für die gleiche Schnittstelle vorhanden sein können. Dies wird bspw. bei der Persistierung benutzt, wo alle Module, welche messbar sind, die Schnittstelle *MeasureableAPI* realisieren.

Die Konfiguration der OSGi Module zur Laufzeit wird über OSGi Services (Hall u. a. (2011), Kap. 4) umgesetzt, wobei für jede Komponente jeweils eine Factory existiert. Diese Factory (Gamma (2005), S. 87ff.) wird als OSGi Service veröffentlicht. So wird der Simulationskern bei dem Start einer neuer Simulation komplett neu über die Factories erzeugt. Dies hat den Vorteil, dass kein Zurücksetzen des Simulationskerns von vorherigen Simulationen erfolgen muss, was fehleranfällig ist.

5.2. Auswahl der Detailstufen

Die Auswahl der Detailstufen wird durch den *LoD Controller* durchgeführt. Dieser bietet, wie bereits beschrieben, für die anderen Komponenten eine abstrakte Schnittstelle, um die entsprechende Detailstufe für eine Skala abzufragen.

Dabei werden die Regeln hierbei in XML definiert, wodurch diese unabhängig von der restlichen Simulation modelliert werden können. Diese XML-Regeln werden dann von dem *LoD Controller* in eine eigene interne Repräsentation übertragen. Dadurch ist es möglich, auch andere Repräsentationen der Regeln zu verwenden oder auch ggf. zukünftig die Regeln zur Laufzeit der Simulation zu ändern.

Die Auflistung 5.1 stellt die XML-Repräsentation von einigen Regeln für die Kollisionsskala und die temporalen Skala beispielhaft dar. Die erste Regel ab Zeile 6 gibt an, dass die Detailstufe 0 bei der Kollisionsskala verwendet werden soll, wenn die Simulationszeit größer als -1ms ist. Da die Simulationszeit bei 0ms beginnt, ist diese Regel immer erfüllt. Dadurch wird sichergestellt, dass zu jedem Zeitpunkt und für jeden Ort eine gültige Zuordnung zu einer Detailstufe der Skala gegeben ist.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ns:LoDConfiguration
3    xmlns:ns="http://walk.informatik.haw-hamburg.de/sim/lodcontroller">
4    <Scale id="COLLISION_DETECTION">
5      <LoD id="0">
6        <GE>
7          <Time time="-1"/>
8        </GE>
9      </LoD>
10     <LoD id="1">
11       <AND>
12         <GE>
13           <Time time="50"/>
14         </GE>
```

```
15          <GT>
16            <AgentDensity density="0.2" radius="1"/>
17          </GT>
18        </AND>
19      </LoD>
20    </Scale>
21    <Scale id="TEMPORAL">
22      <LoD id="1">
23        <GE>
24          <Time time="-1"/>
25        </GE>
26      </LoD>
27      <LoD id="2">
28        <AND>
29          <GE>
30            <Time time="50"/>
31          </GE>
32          <AND>
33            <GT>
34              <Point x="13.8" y="0" z="2.07"/>
35            </GT>
36            <LT>
37              <Point x="23.46" y="5" z="9.67"/>
38            </LT>
39          </AND>
40        </AND>
41      </LoD>
42    </Scale>
43  </ns:LoDConfiguration>
```

Auflistung 5.1: Beispiel der XML-Repräsentation von Regeln

Dadurch, dass bei der Auswertung der Regeln jeweils die erste Zuordnung verwendet wird, bei welcher die Bedingung erfüllt ist und die Auswertung bei der numerisch höchsten Detailstufe beginnt, können Bereiche der Simulation mit höheren Detailstufen überschrieben werden. So wird durch die Regel ab Zeile 11 die Detailstufe mit der Nummer 1 für die Kollisionsskala verwendet, wenn die aktuelle Simulationszeit größer oder gleich 50ms ist und die Agentendichte an dem Ort in einem Umkreis von 1m höher als 20% liegt.

Bei der Auswertung der Regeln werden von dem *LoD Controller* automatisch die Variablen aus der Simulation mit den Werten aus den Regeln verglichen. So wird bspw. bei der Auswertung

dieser Detailstufe, bei der Ausführung eines Agenten, die Position des Agenten für den Vergleich verwendet.

Es existiert jedoch auch die Möglichkeit, dass der *LoD Controller* nicht automatisch die Variable extrahieren kann. Bspw. existiert in den Regeln eine Bedingung, die auf die Art einer Aktion, wie z.B. Schubsen, prüft. Wenn jetzt der *LoD Controller* über Detailstufe zum einplanen eines Agenten abgefragt wird, kann der Wert der Variable nicht automatisch extrahiert werden, da ein Agent für das einplanen keine Aktion besitzt. In diesem Fall wird eine solche Bedingung bzw. der Teil der Bedingung, wenn die Variable nicht extrahiert werden kann, weil diese in dem aktuellen Kontext nicht existiert, immer als falsch angenommen.

Die Zuordnung zwischen den Nummern der Detailstufen und den tatsächlichen Detailstufen, wird durch die Effekt-Module definiert. Dabei kann die Reihenfolge der Detailstufen beliebig definiert werden. So ist es möglich, die Genauigkeit der Detailstufen einer Skala mit den Nummern ansteigen zu lassen. So wäre bspw. die Detailstufe mit der Nummer 2 genauer als die Detailstufe mit der Nummer 1. Das Effekt-Modul kann jedoch auch die Reihenfolge umkehren. Dadurch ist es möglich, dass bspw. immer eine hohe Detailstufe verwendet wird, die gezielt durch niedrigere Detailstufen in den Regeln überschrieben wird.

5.3. Messungen

Die Simulation einer Simulationsreihe ist, wie in Abbildung 5.2 dargestellt, in die Teilschritte Simulation und Evaluierung eingeteilt. Bei der Simulation werden das Modell berechnet und die Messwerte gespeichert. Auf diesen gespeicherten Messwerten werden dann in einem zweiten Schritt, welcher auch nach der Simulation ausgeführt werden kann, die Auswertung durchgeführt.

Bei dieser Aufteilung liefert die Persistierung der Daten die Schnittstelle zwischen der Berechnung des Simulationsmodells und der Evaluierung der Simulation.

5.3.1. Messmethode

Zum Ermitteln des Ressourcenbedarfs der Simulation existieren hauptsächlich zwei Ressourcen: Die CPU, sowie der Arbeitsspeicher.

Da es sich bei der Simulationsanwendung um eine Java-Anwendung handelt, wird die Speicherverwaltung durch die JVM (Java Virtual Machine) durchgeführt. Es kann nicht garantiert werden, wann der Garbage Collector läuft und daher kann keine Aussage über den tatsächlichen Bedarf an Arbeitsspeicher eines Algorithmus oder Teils des Codes getroffen werden.

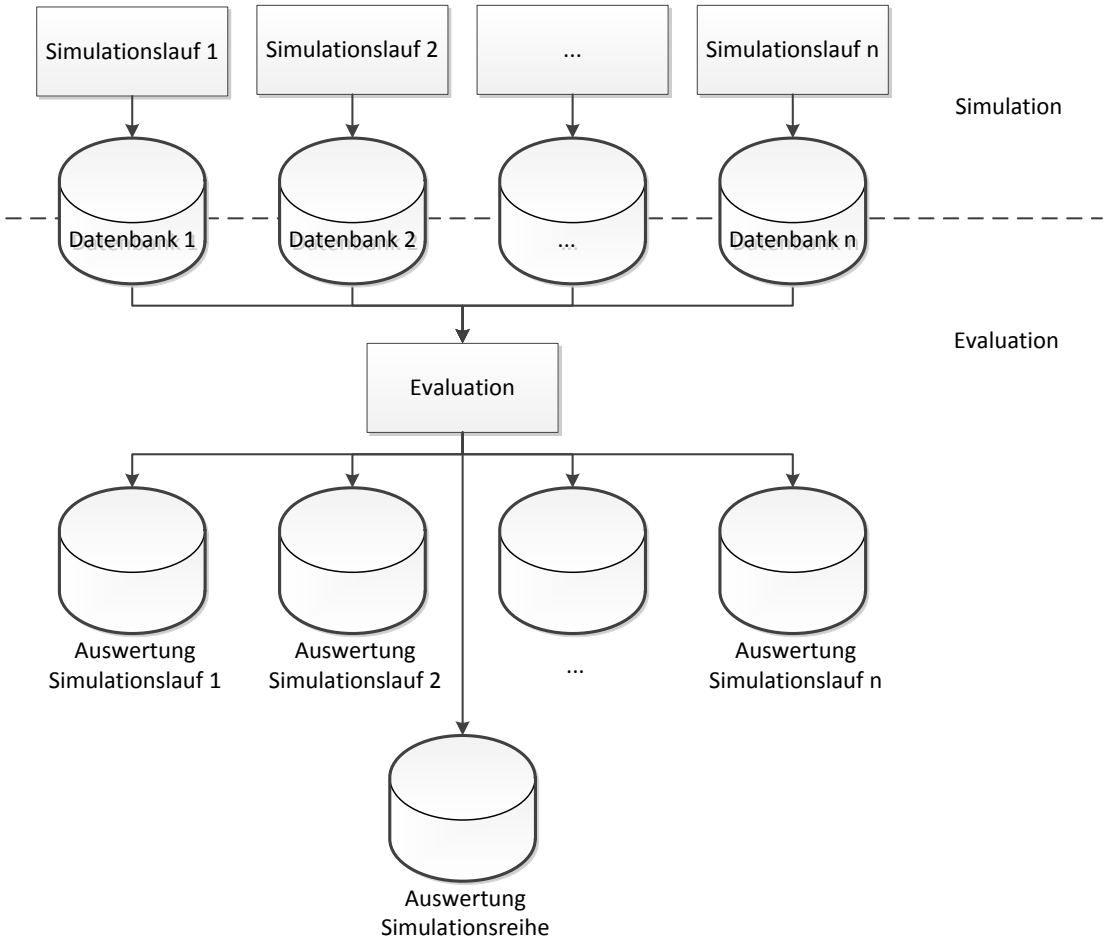


Abbildung 5.2.: Ablauf der Simulation und Auswertung

Für die Ermittlung der CPU-Leistung sind die folgenden drei Methoden denkbar:

- Messen der Laufzeit des Algorithmus anhand der Differenz der Zeit vor und nach der Ausführung des zu messenden Bereichs.
- Messen der aktuellen CPU-Auslastung während der Ausführung des zu messenden Bereichs.
- Messen der Differenz der benötigten CPU-Zeit der JVM aus der CPU-Zeit vor und nach der Ausführung des zu messenden Bereichs.

In dieser Arbeit wird, wie bereits in Twelkemeier (2013b) (S. 22f.), die dritte Methode, das Messen der benötigten CPU-Zeit der JVM verwendet. Dies hat den Vorteil, dass diese Zeit unabhängiger von der aktuellen CPU-Auslastung des Systems durch andere Prozesse als der JVM ist. Des Weiteren enthält diese Zeit ebenfalls die Zeit, wenn mehrere Threads parallel ausgeführt werden, was bei einer reinen Messung der Laufzeit, wie bei der ersten Methode, nicht beachtet wird. Dadurch sind die Ergebnisse von der Parallelisierung der Ausführung des Algorithmus unabhängig.

Jedoch enthält die CPU-Zeit der JVM auch die Zeit, die bspw. durch den Garbage Collector benötigt wird.

Zum Messen wird die Java Management Extension verwendet. Diese liefert über die MxBeans die Möglichkeit, die JVM zu überwachen und Kennzahlen, wie bspw. die benötigte CPU-Zeit, abzufragen (Java SE Documentation (2014), Javadoc getProcessCpuTime (2013)). Über die OperatingSystemMXBean wird die benötigte CPU-Zeit vor und nach dem zu messenden Bereich, wie in der Auflistung 5.2 dargestellt, abgefragt und die Differenz gebildet werden. Diese entspricht der benötigten CPU-Zeit der JVM während der Ausführung des zu messenden Bereichs. Es ist jedoch zu beachten, dass, obwohl die gemessene Zeit im Bereich von Nanosekunden liegt, die Genauigkeit der Messwerte nicht im Bereich von einer Nanosekunde liegen muss. Die Genauigkeit hängt von der JVM und dem Betriebssystem ab (Javadoc getProcessCpuTime (2013)).

```
1 import java.lang.management.ManagementFactory;
2 import com.sun.management.OperatingSystemMXBean;
3
4 public class Example {
5  private final OperatingSystemMXBean mxBean = (OperatingSystemMXBean)
6   ManagementFactory.getOperatingSystemMXBean();
7
8  public void method() {
```

```
9    long cpuTimeBefore = mxBean.getProcessCpuTime();
10   // do something
11   long cpuTime = mxBean.getProcessCpuTime() - cpuTimeBefore;
12   // notify Persistence
13  }
14 }
```

Auflistung 5.2: Beispiel einer Messung der benötigten CPU-Zeit

5.3.2. Persistierung der Daten

Die Daten der Simulation werden in einer Berkeley DB Java Edition Datenbank gespeichert. Diese bietet den Vorteil, dass diese komplett in Java umgesetzt ist, wodurch nicht für jede Operation ein Wechsel aus dem Kontext der JVM nötig ist (Oracle (2014)).

Bei der Berkeley DB handelt es sich jedoch nicht um eine relationale Datenbank, sondern um einen Key / Value Store. Daher wird für die Speicherung der Daten in dieser Datenbank die Struktur, die in Abbildung 5.3 dargestellt ist, verwendet. Dabei wird die Zeit als verkettete Liste abgebildet, wobei durch die Schlüsselwörter „start“ und „finish“ der Anfang bzw. das Ende markiert werden. Für jeden Zeitschritt existiert in dieser vorwärts verketteten Liste jeweils ein Element, was der Simulationszeit entspricht. Jedem dieser Zeitschritte ist dann ein sogenanntes *WorldState* Objekt zugeordnet. Dabei handelt es sich um ein Java-Objekt, welches über die Java-Serialisierung gespeichert wird und den Zustand der Umwelt sowie die Messwerte, wie bspw. die benötigte CPU-Zeit für die Ausführung der Agenten, für diesen Zeitschritt enthält.

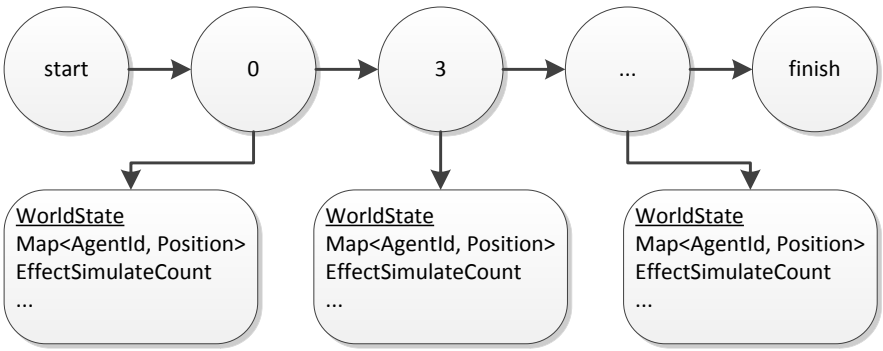


Abbildung 5.3.: Datenstruktur der Persistierung

Durch die Java-Serialisierung existiert zwar an dieser Stelle eine Abhängigkeit auf die Java-Programmiersprache, jedoch verursacht die Java-Serialisierung im Vergleich mit anderen Serialisierungen wie bspw. JSON, XML oder Protocol Buffers von Google einen geringeren

Mehraufwand. Da die benötigte Zeit für die Serialisierung und Persistierung ggf. die Messungen beeinflussen kann, muss dieser Mehraufwand bei der Berechnung der Simulation möglichst gering gehalten werden.

Die Zuordnung der *WorldState*-Objekte zu den Simulationszeiten erfolgt, indem als Schlüssel für diese Werte der Simulationszeit jeweils die Zeichenfolge „worldstate_“ vorangestellt wird. So ist das WorldState-Objekt für die Simulationszeit 3 unter dem Schlüssel „worldstate_3“ in der Datenbank gespeichert.

Werte, welche sich auf den ganzen Simulationslauf beziehen, werden unter separaten Schlüsselwörtern in der Datenbank gespeichert. So wird der Name des Szenarios, welches simuliert wurde unter dem Schlüsselwort „szenario“ gespeichert und die benötigte CPU-Zeit für die komplette Simulation unter „cpuTimeNeeded“.

5.3.3. Auswertung der Messergebnisse

Bei der Auswertung der Simulationsreihen werden jeweils die Simulationsläufe einzeln evaluiert sowie auch alle Simulationsläufe einer Simulationsreihe zusammen. Sowohl bei der Evaluation der einzelnen Simulationsläufe als auch bei der Simulationsreihe werden die benötigten Entfluchtungszeiten und CPU-Zeiten sowie unterschiedliche Linien- und Balkendiagramme erstellt. Die Diagramme visualisieren die unterschiedlichen gemessenen Werte, wie bspw. die benötigte CPU-Zeit für die Agentenausführung oder die Anzahl der eingeplanten Agenten. Lediglich bei der Evaluierung der einzelnen Simulationsläufe werden zusätzlich die Wege eines jeden Agenten sowie ein Video der Simulation erzeugt.

Für die Evaluierung einer Simulationsreihe werden die Kennzahlen entsprechend ihren Semantiken ermittelt. Das bedeutet, dass bspw. bei der Auswertung einer Simulationsreihe mit 10 Simulationsläufen für einen Median-Wert für die benötigte CPU-Zeit pro Simulationsrunde der Median-Wert über die 10 Median-Werte der einzelnen Simulationsläufe gebildet wird. Für die Berechnung der Werte wird jeweils eine Zeitspanne von 500ms Simulationszeit verwendet. Das bedeutet, dass sowohl der Median, als auch der Minimal-, Maximal- und Durchschnittswert jeweils über 500ms berechnet wurde. Lediglich die Kennzahlen, wie die benötigte Entfluchtungszeit sowie die insgesamt benötigte CPU-Zeit werden nicht über 500ms Simulationszeit, sondern über die gesamte Simulationsdauer gebildet.

Für die Auswertung der CPU-Zeiten über die Simulationszeit wird jeweils ein 95% Quantil verwendet. Durch dieses sollen Messfehler, welche bspw. durch den Garbage-Collector oder ähnlichen Aufgaben entstehen, herausgefiltert werden.

6. Evaluierung

In dem vorherigen Kapitel wurde die Realisierung der Simulation beschrieben. Aufbauend auf dieser Simulationsplattform wird in dem folgenden Kapitel die beiden Beispielszenarien umgesetzt, simuliert, gemessen und ausgewertet.

6.1. Modellierung der Szenarien

Die Szenarien, die bereits in dne Grundlagen vorgestellt wurden, sind für die Messung und Evaluierung in dieser Arbeit umgesetzt worden. Dabei existieren in beiden Szenarien einige Aspekte, welche gleich modelliert wurden.

Nach Weidemann (1993) (S. 14f.) kann die Standfläche einer Person als Ellipse mit einer durchschnittlichen Breite von 0,46m und einer Tiefe von 0,23m abgebildet werden. In Kirik u. a. (2007) sind die Agenten dagegen als Rechtecke mit 0,4m x 0,4m Fläche modelliert. In dieser Arbeit sind die Agenten als Zylinder mit einem Durchmesser von 0,4m und einer Höhe von 1,75m modelliert.

Die Umwelt wird als dreidimensionaler Raum modelliert, wovon jedoch einige Algorithmen lediglich eine zweidimensionale Darstellung für die Berechnung verwenden. Um die unterschiedlichen Objekte und Bereiche abbilden zu können, existieren in der Umwelt verschiedene Objekttypen:

Hindernisse Dies sind Bereiche, die der Agent nicht betreten kann. Hiermit werden bspw. Wände modelliert.

Ein- und Ausgänge Diese Objekte definieren lediglich einen Punkt. Die Agenten können an diesen Punkten die Welt betreten oder verlassen.

Agenten Die Agenten werden, wie bereits beschrieben, in der Umwelt als Zylinder dargestellt. Diese Zylinder sind lediglich die Repräsentation der Agenten im physikalischen Raum und enthalten nicht die Logik des Agenten. Der Agent kann diesen Zylinder nur über Aktionen beeinflussen, welche er ausführt.

Treppenbereich Dieser Bereich bildet Treppen ab. Aufgrund der Steigung können sich die Agenten lediglich mit 50% der Geschwindigkeit in diesem Bereich bewegen (vgl. Klüpfel (2003) (S. 67)).

Türbereich In dieser Simulation sind die Türen nicht explizit als Türen, welche geöffnet oder geschlossen werden können, modelliert, sondern wie in Klüpfel (2003) (S. 67) als Bereiche mit verminderter Geschwindigkeit. Das bedeutet hierdurch werden Türbereiche gekennzeichnet. Ein Agent, der sich in diesem Bereich befindet, kann sich lediglich mit 25% seiner Geschwindigkeit bewegen (vgl. Klüpfel (2003) (S. 67)).

6.1.1. Kino-Szenario

Dieses Szenario wird u.a. für eine Validierung der Simulation verwendet. Daher werden die Agenten zum Start der Simulation analog zu Tabelle 2.1 positioniert. Des Weiteren ist für jeden Agenten das Ziel festgelegt. Dafür wird entsprechend der Auswertung der Übung von Klüpfel (2003) den Agenten auch jeweils der Ausgang als Ziel zugewiesen, welchen auch die Person in der Übung verwendet hat.

Die maximale Geschwindigkeit der Agenten ist abweichend von Klüpfel (2003) an Weidemann (1993) (S.43) orientiert. Dabei wird als maximale Geschwindigkeit für jeden Agenten zufällig ein Wert zwischen 0,97 m/s und 1,5 m/s zugeordnet.

Beim Start der Simulation sind alle Agenten bereits im Kinosaal auf ihren Plätzen und wechseln sofort in den Flucht-Zustand. Wie jedoch auch in Klüpfel (2003), besitzt jeder Agent eine zufällige Verzögerung zwischen 0 und 4 Sekunden, bevor dieser beginnt sich zum Ausgang zu bewegen.

Zusätzlich werden die in Tabelle 6.2 beschriebenen Varianten des Szenarios für die Evaluation simuliert.

6.1.2. Nachtclub-Szenario

Bei diesem Szenario werden insgesamt 500 Agenten in der Umwelt sein. Diese betreten die Umwelt in den ersten 50 Sekunden und bewegen sich zu unterschiedlichen Positionen. Nach 50 Sekunden wird etwa in der Mitte der Tanzfläche das Pfefferspray-Ereignis aktiviert. Dieses breitet sich dann entsprechend der Detailstufe aus.

Die Agenten ergreifen die Flucht, sobald diese den Wirkstoff wahrnehmen. Für die Wahrnehmung existiert ein Grenzwert von 0,0000000001. Sobald dieser Grenzwert überschritten wird, ergreifen die Agenten die Flucht. Das bedeutet, die Agenten gehen in den Flucht-Zustand,

sobald diese das Pfefferspray wahrnehmen. Dabei versuchen alle Agenten den Eingang E1 zu erreichen und die Umwelt dort zu verlassen.

Für die maximale Bewegungsgeschwindigkeit besitzt jeder Agent einen zufälligen Wert zwischen 1,2 m/s und 2 m/s (Klüpfel (2003) S.78). Dieser Wert wurde, im Gegensatz zu dem Kino-Szenario, höher gewählt, da hierbei die Personen tatsächlich einer Gefahr ausgesetzt waren und die Personen in der Übung von Klüpfel (2003) dazu angehalten waren, sich vorsichtig zu verhalten (Klüpfel (2003) S.76)

Bei diesem Szenario werden die in Tabelle 6.3 beschriebenen Varianten simuliert.

6.2. Messungen

Für eine Evaluation der Simulation und Überprüfung der Hypothese dieser Arbeit müssen die beschriebenen Szenarien mit unterschiedlichen Konfigurationen simuliert werden. Dafür müssen unterschiedliche Variablen gemessen werden. Im Folgenden wird beschrieben, welche Variablen unter welcher Messumgebung für welche Konfigurationen gemessen werden.

6.2.1. Variablen

Um Aussagen über das vorgestellte Verfahren treffen zu können, müssen möglichst aussagekräftige und genaue Kennzahlen ermittelt werden. Da jedoch durch jede Messung, die zusätzlich durchgeführt wird, das Verhalten und vor allem der Ressourcenverbrauch des Programms verändert wird, sollten möglichst wenige Messungen durchgeführt werden.

In jeder Simulationsrunde werden die folgenden Variablen gemessen und gespeichert:

- Benötigte CPU-Zeit der
  - Simulationsrunde
  - Agentenausführung (alle Agenten)
  - *Event-Manager* Komponente
  - *LoD-Controller* Ausführung
  - *Environment* Ausführung
  - Effekt Module (Berechnung des Zeitpunktes bis zu welchem simuliert werden kann.)
  - Effekt Module (Simulation des Effekts)
  - gesamten Simulation

  - einzelnen Detailstufen der Kollisionserkennung

- Fachliche Variablen
  - Aktuelle Simulationszeit
  - Position der Agenten
  - Anzahl der eingeplanten Aktionen
  - Anzahl der laufenden Effekte
  - Anzahl der ausgeführten Aktionen
  - Anzahl der ausgeführten Agenten
  - Anzahl der *LoD-Controller* Abfragen

Aus diesen Werten kann dann zusätzlich bspw. die Schrittgröße pro Simulationsrunde und pro Agent bestimmt werden.

Zur Auswertung werden dann, aufgrund der vielen Messpunkte, jeweils der minimale und maximale Wert sowie der Median und der Durchschnitt über 500ms Simulationszeit berechnet. Diese Werte werden jeweils als Liniendiagramm dargestellt. Des Weiteren wird jeweils die Summe der Werte über 500ms Simulationszeit als Säulendiagramm dargestellt.

Um einzelne Extremwerte, welche bspw. durch den Garbage Collector der JVM verursacht worden sein können, aus den Daten zu filtern, wird jeweils bei den gemessenen CPU-Zeiten vorher ein 95% Quantil angewendet.

Zusätzlich zu den Liniendigrammen werden jeweils, wie in Klüpfel (2003) (S. 78), die minimale, maximale, der Median sowie der Durchschnitt der benötigten Entfluchtungszeit der Agenten pro Ausgang und über alle Ausgänge berechnet.

6.2.2. Messumgebung

Alle Messungen wurden auf dem in Tabelle 6.1 beschriebenen Rechner ausgeführt.

6.2.3. Konfigurationen

Für die Evaluierung der Simulation werden die beiden Szenarien mit unterschiedlichen Konfigurationen bezüglich der Detailstufen simuliert.

Für das Kino-Szenario werden 12 unterschiediche Konfigurationen simuliert. Diese in Tabelle 6.2 zusammengefassten Konfigurationen können in fünf unterschiedliche Gruppen eingeteilt werden. Alle Konfigurationen des Kino-Szenarios werden zusätzlich mit einer maximalen Schrittweite von 10ms, 50ms und 500ms simuliert.

CPU	Intel Xeon x5660 @ 2,8GHz (2 Prozessoren, 12 Kerne, 24 Kerne mit Hyperthreading)
RAM	48 GB 1333 MHz
Betriebssystem	Windows Server 2012, 64 Bit
JVM	Oracle JVM, 1.7.0_25 b17 64 Bit
Genauigkeit CPU-Zeit	15,625 ms
JVM Parameter	-Xms12800m -Xmx25600m -XX:+UseParNewGC -XX:MaxPermSize=512m

Tabelle 6.1.: Beschreibung der Simulationsumgebung

	Detailskala	
Konfig.	**Temporale Skala**	**Kollisionsskala**
T0C0	Hard-Time-Step-Driven	Deaktiviert
T0C2	Hard-Time-Step-Driven	Hoch
T2C0	Event-Driven	Deaktiviert
T2C2	Event-Driven	Hoch
C1	Soft-Time-Step-Driven im Bereich der Ausgänge A und B, sonst Hard-Time-Step-Driven	Hoch im Bereich der Ausgänge A und B, sonst deaktiviert
C2	Soft-Time-Step-Driven im Bereich der Ausgänge A und B, sonst Hard-Time-Step-Driven	Mittel im Bereich der Ausgänge A und B, sonst deaktiviert
C3	Event-Driven im Bereich der Ausgänge A und B, sonst Hard-Time-Step-Driven	Hoch im Bereich der Ausgänge A und B, sonst deaktiviert
C4	Event-Driven bei einer Agentendichte größer 0,2 in einem Radius von 1m, sonst Hard-Time-Step-Driven	Deaktiviert
C5	Soft-Time-Step-Driven in den Türbereichen, sonst Hard-Time-Step-Driven	Hoch im Bereich der Türen, sonst deaktiviert
C6	Soft-Time-Step-Driven	Mittel im Kinosaal, sonst deaktiviert
C7	Hard-Time-Step-Driven	Mittel im Kinosaal, sonst deaktiviert
C8	Hard-Time-Step-Driven	Hoch im Kinosaal, sonst deaktiviert

Tabelle 6.2.: Simulierte Konfigurationen des Kino-Szenarios bezüglich der Detailstufen der jeweiligen Skalen

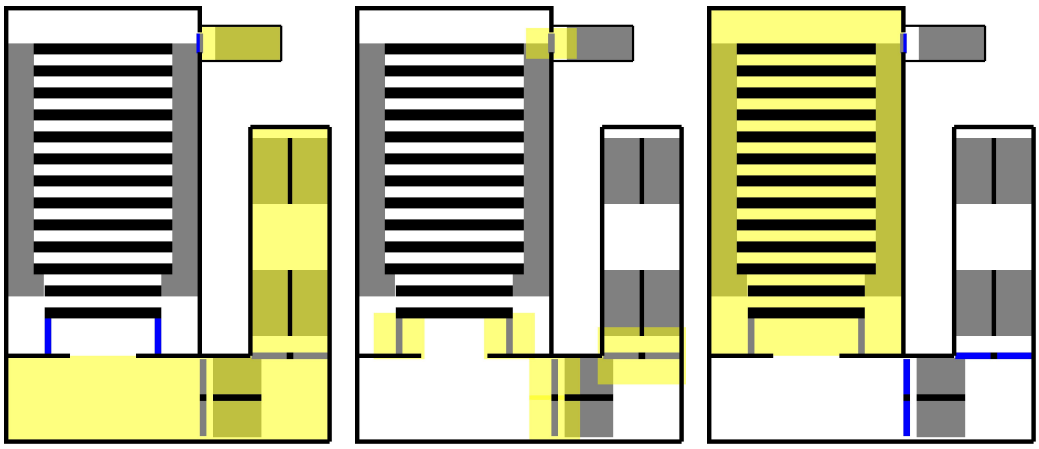
Die erste Gruppe simuliert die Extreme der unterschiedlichen Konfigurationen der Detailstufen. Dazu zählen die Konfigurationen T0C0, T0C2, T2C0 sowie T2C2. Diese Konfigurationen simulieren alle möglichen Kombinationen mit den jeweils höchsten und niedrigsten Detailstufen der temporalen und der Kollisionsskala.

Die zweite Gruppe legt den Schwerpunkt auf die Bereiche der Ausgänge. Dabei werden die Bereiche bei den Ausgängen aus Abbildung 6.1(a) mit unterschiedlichen Detailstufen simuliert. Diese Gruppe umfasst die Konfigurationen C1 bis C3.

Der Schwerpunkt wird bei der dritten Gruppe auf die Bedingung der Agentendichte gelegt. Zu dieser Gruppe gehört die Konfiguration C4.

C5 legt als weitere Gruppe den Schwerpunkt auf die Türbereiche. Diese Bereiche sind in Abbildung 6.1(b) dargestellt.

Bei den Konfigurationen C6 bis C8 wird der Schwerpunkt auf den Kinosaal gelegt. Dieser Bereich wird in Abbildung 6.1(c) dargestellt.



(a) Ausgangsbereiche A und B (b) Türbereiche (c) Bereich des Kinosaals

Abbildung 6.1.: Bereiche der Konfigurationen des Kino-Szenarios (gelb eingefärbt)

Für das Nachtclub-Szenario werden die Konfigurationen aus Tabelle 6.3 mit einer maximalen Schrittweite von 100ms verwendet. Dabei verwendet die Konfiguration C3 für alle Skalen jeweils die höchste Detailstufe. Um die Auswirkungen der Detailstufen evaluieren zu können, werden diese dann sukzessive in den Konfigurationen C1, C2, C4 und C5 gesenkt. Bei diesen Konfigurationen C1 bis C5 ist die Detailstufe für die Skalen jeweils für die komplette Simulation festgelegt und ändert sich nicht.

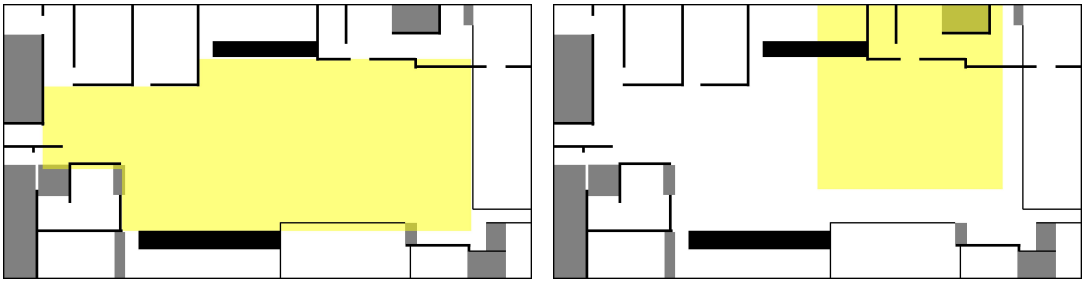
Bei der Konfiguration C6 sind die Detailstufen aller drei Skalen von dem Ort in der Umwelt sowie der Simulationszeit abhängig. Die Kollisionsskala und die temporale Skala sind lediglich

	Detailskala		
Konfig.	**Temporale Skala**	**Kollisionserkennung und -behandlung**	**Gasausbreitung**
C1	Event-Driven	Deaktiviert	Ausbreitung sofort
C2	Event-Driven	Hoch	Ausbreitung sofort
C3	Event-Driven	Hoch	Ausbreitung über die Zeit
C4	Event-Driven	Mittel	Ausbreitung sofort
C5	Soft-Time-Step-Driven	Hoch	Ausbreitung sofort
C6	Ab 49,5 Sekunden Event-Driven im Bereich des Ausgangs, sonst Hard-Time-Step-Driven	Ab 49,5 Sekunden hoch im Bereich des Ausgangs, sonst deaktiviert.	Zwischen 49,5 und 80 Sekunden Ausbreitung über die Zeit auf der Tanzfläche und dem Ausgangsbereich, sonst zwischen 49,5 und 80 Sekunden Ausbreitung sofort. Vor 49,5 Sekunden und nach 80 Sekunden wird keine Ausbreitung berechnet.

Tabelle 6.3.: Simulierte Konfigurationen des Nachtclub-Szenarios bezüglich der Detailstufen der jeweiligen Skalen

im Ausgangsbereich ab der Simulationszeit 49,5 Sekunden hoch, da dort die höchste Agentendichte erwartet wird und dies eine kritische Stelle darstellt. Vor der Simulationszeit 49,5 sowie nach 49,5 in allen Bereichen, außer dem Ausgangsbereich, wird die niedrigste Detailstufe verwendet.

Für die genaue Ausbreitung des Pfeffersprays in dem Szenario wird davon ausgegangen, dass lediglich der Bereich der Tanzfläche, die in Abbildung 6.2 dargestellt ist, sowie des Ausgangs für die Simulation relevant ist. Daher legt die Skala der Gasausbreitung dort den Schwerpunkt. Die Gasausbreitungsskala ist auf der Tanzfläche und dem Ausgangsbereich zwischen den Simulationszeitpunkten 49,5 und 80 Sekunden hoch und simuliert die genaue Ausbreitung. In allen anderen Bereichen wird die Ausbreitung zwischen den Zeitpunkten 49,5 und 80 Sekunden mit der mittleren Detailstufe simuliert, was einer sofortigen Ausbreitung entspricht. Vor 49,5 und nach 80 Sekunden wird in der ganzen Simulation keine weitere Ausbreitung berechnet. Die Zeit 49,5 Sekunden kommt daher, dass zum Zeitpunkt 50 Sekunden die Ausbreitung des Pfeffersprays beginnt. Durch Simulationen hat sich gezeigt, dass die Ausbreitung bis zum Zeitpunkt 80 Sekunden etwa bis zum Ausgang ausgebreitet hat. Da der Fokus bei dieser Konfiguration auf der Tanzfläche sowie dem Ausgang gelegt wurde, wird ab dem Zeitpunkt die weitere Ausbreitung des Gases gestoppt.



(a) Bereich der Tanzfläche (b) Bereich des Ausgangs

Abbildung 6.2.: Bereiche der Konfigurationen des Nachtclub-Szenarios (gelb eingefärbt)

6.3. Validierung

Für die Validierung der Simulation wird das Kino-Szenario verwendet, da für dieses Szenario genaue Entfluchtungszeiten aus Klüpfel (2003) bekannt sind. Des Weiteren wird die Simulation mit den unterschiedlichen Detailstufen der Skalen visuell analysiert.

Wie in Unterabschnitt 6.2.3 bereits beschrieben, existieren für jedes Szenario unterschiedliche Konfigurationen, welche im Folgenden betrachtet werden. In Tabelle 6.4 sind die gemessenen

Werte der Konfigurationen über jeweils zehn Durchläufe dargestellt. Bei den Entfluchtungszeiten entspricht der Maximalwert dem Median der Maximalwerte der einzelnen Durchläufe und der Minimalwert entsprechend dem Median der Minimalwerte der einzelnen Durchläufe. Der Durchschnitt bzw. Median entspricht dem Durchschnitt über die Durchschnitte bzw. dem Median der Median-Werte der einzelnen Durchläufe. Die Spalte Ref gibt, soweit vorhanden, die jeweilige Entfluchtungszeit als Referenzwert aus Klüpfel (2003) (S.78) an.

Die Felder der Tabelle 6.4 sind bei den benötigten CPU-Zeiten grün eingefärbt, wenn die Zeit unterhalb von 500 Sekunden liegt, gelb, zwischen 500 und 1000 Sekunden und rot bei über 1000 Sekunden. Bei der Evakuierungszeit ist die Farbe des Feldes abhängig von der Differenz zwischen dem Referenzwert und dem Wert des Felds. Das Feld ist grün eingefärbt, wenn die Differenz unterhalb von 5 Sekunden liegt, gelb zwischen 5 und 10 Sekunden und rot, wenn die Differenz über 10 Sekunden liegt.

Für die Validierung werden im Folgenden die Konfigurationen T0C0, T0C2, T2C0 und T2C2 aus der Tabelle 6.4 betrachtet.

Hierbei liefert die Konfiguration T0C2 die höchste Genauigkeit. Alle Entfluchtungszeiten, bis auf der Durchschnitt über alle Ausgänge, weichen um weniger als 5 Sekunden von dem Referenzwert aus der Übung von Klüpfel (2003) ab.

Überraschend ist jedoch, dass die Konfiguration T2C2, welche jeweils die höchste Detailstufe für alle Skalen verwendet, etwa gleich bezüglich der beiden Konfigurationen T0C0 und T2C0 ist. Dies bestätigt die Annahme aus Unterabschnitt 3.1.1, dass die aufwändigste Detailstufe bzw. die Detailstufe mit der höchsten Genauigkeit nicht unbedingt die beste Detailstufe in allen Situationen ist.

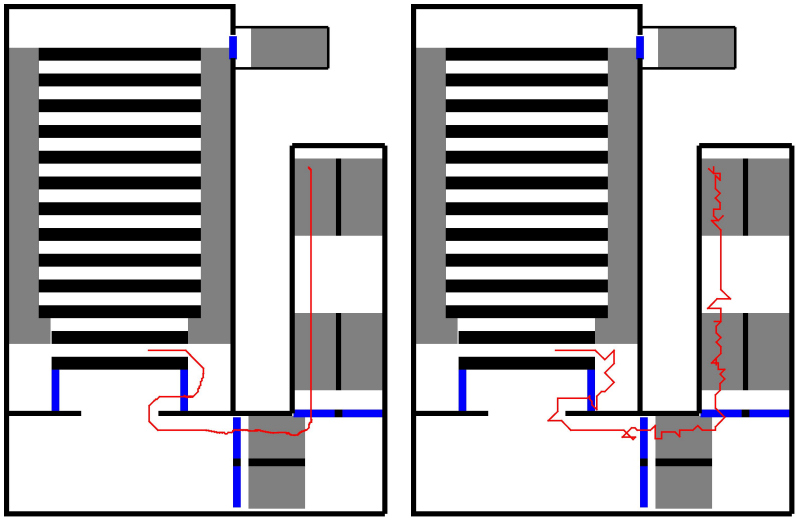
Des Weiteren fällt auf, dass die Durchschnitts- und Median-Werte aller vier Konfigurationen unterhalb der Referenzwerte liegen. Das bedeutet, die Agenten haben das Ziel schneller als in der Übung aus Klüpfel (2003) erreicht. Dies kann mehrere Ursachen haben: Die erste Ursache ist, dass die Agenten zu schnell gelaufen sind. Da die verwendete Geschwindigkeit mit 3,492 bis 5,4 km/h geringer, als die von Klüpfel (2003) (S. 78) genannten 4,32 bis 7,2 km/h ist, wird an dieser Stelle die reine maximale Bewegungsgeschwindigkeit ausgeschlossen. Eine andere Möglichkeit in dieser Simulation ist, dass die Agenten die Richtung innerhalb von wenigen Millisekunden komplett ändern können und auch keine Beschleunigung oder Verzögerung der Geschwindigkeit der Agenten modelliert ist. Des Weiteren kann diese Ungenauigkeit auch durch eine Abweichung bei der Modellierung der Umwelt entstehen. Da für die Abmessungen der Umwelt lediglich die Grafiken aus Klüpfel (2003) zur Verfügung standen, können Abweichungen bei dem Aufbau der Umwelt nicht ausgeschlossen werden.

	Ref	T0C0			T0C2			T2C0			T2C2		
		10ms	50ms	500ms	10ms	50ms	500ms	10ms	50ms	500ms	10ms	50ms	500ms
Alle Ausgänge													
Maximum	66	55,0	54,7	62,0	67,4	111,4	147,5	54,3	53,8	61,1	56,5	65,4	73,1
Minimum		8,6	9,2	11,0	8,2	9,4	11,5	8,1	8,4	10,3	8,6	8,5	9,7
Durchschnitt	45	34,0	34,8	38,0	39,6	58,3	77,0	33,4	33,7	37,5	34,0	34,3	34,9
Median	44	34,7	36,4	38,5	39,4	54,5	74,0	34,3	34,8	37,9	34,4	34,0	35,2
Ausgang A													
Maximum	45	42,3	44,6	43,0	46,9	62,1	92,0	41,3	41,8	44,5	40,6	44,1	42,5
Minimum		8,6	9,2	11,0	8,2	9,4	11,5	8,1	8,4	10,3	8,6	8,5	9,7
Durchschnitt	31,1	25,5	26,3	27,7	27,7	36,4	52,1	25,2	25,1	27,4	24,1	25,0	24,7
Median	31	25,9	26,5	28,5	28,4	37,9	54,0	25,6	25,6	27,7	24,5	24,0	24,9
Ausgang B													
Maximum	66	55,0	54,7	62,0	67,4	111,4	147,5	54,3	53,8	61,1	56,5	65,4	73,1
Minimum		26,5	27,0	28,5	25,8	26,2	31,0	25,0	26,6	25,4	26,5	25,1	27,9
Durchschnitt	53,1	40,6	41,4	46,0	48,8	75,2	96,2	39,7	40,3	45,4	41,7	41,4	42,8
Median	53	41,5	42,5	47,0	49,6	80,6	99,5	39,8	40,5	46,0	42,2	42,5	42,6

Tabelle 6.4.: Benötigte Entfluchtungszeiten in Sekunden [angelehnt an Klüpfel (2003) (S.78)]

Bei der Betrachtung der Konfigurationen bei einer maximalen Schrittweite von 50 ms verändert sich die beste Konfiguration. Hierbei liefert bezüglich der maximalen Entfluchtungszeit die Konfiguration T2C2 die besten Ergebnisse.

Bei den Simulationen mit 500ms aus Tabelle 6.4 ist zu erkennen, dass sich die Konfiguration T2C2 im Vergleich zu der maximalen Schrittweite von 50ms verbessert hat. Jedoch sind die besten Konfigurationen T0C0 und T2C0. Dies ist dadurch zu erklären, dass die Agenten bei 500ms, wie in Abbildung 6.3 dargestellt, nicht mehr den optimalen Weg nehmen. Durch die längere Strecke benötigen die Agenten länger um ihr Ziel zu erreichen. Dies führt dazu, dass bei den Konfigurationen ohne Kollisionserkennung die Zeiten näher an den Referenzwerten sind.



(a) 10ms maximale Schrittweite (b) 500ms maximale Schrittweite

Abbildung 6.3.: Vergleich des Pfads bei der Konfiguration T0C0

Bei der Konfiguration T0C2 führt dies dazu, dass bei einer Kollision der Agent den Rest des Zeitschritts nicht nutzen kann, da dieser durch die Kollision beeinflusst wurde und der Bewegungs-Effekt dadurch abgebrochen wurde. Der Agent muss erst wieder eine neue Aktion einplanen. Da das Einplanen einer neuen Aktion jedoch aufgrund der temporalen Detailstufe erst zu dem nächsten Zeitschritt erfolgt, bleibt der Agent bis zum nächsten Zeitschritt stehen, wodurch die benötigte Zeit des Agenten deutlich über den Referenzwert steigt.

Bei der Konfiguration T2C2 führen Kollisionen nur dazu, dass der Agent sofort neue Aktionen einplant. Das bedeutet, dass die Schrittweite zu dem Zeitpunkt dann für diesen Agenten geringer als 500 ms ist. Da dies aber nicht immer passiert, bewegt sich der Agent bei größeren

Schrittweiten nicht mehr auf dem optimalen Weg und benötigt daher mehr Zeit für die Strecke. Dies gleicht ggf. die Zeitdifferenz durch die fehlende Berücksichtigung von Beschleunigung bzw. Verzögerung und Richtungsänderung in diesem Modell aus.

Bei der Betrachtung der Entfluchtungszeiten aus Abbildung 6.4 ist ebenfalls erkennbar, dass die Agenten etwas früher die Ausgänge erreichen. Der charakteristische Verlauf der Kurve ist jedoch annähernd identisch mit dem Verlauf der Kurve, die von Klüpfel (2003) für die Übung angegeben ist. Das bedeutet, dass die Agenten in dieser Simulation etwa in gleicher Frequenz die Umwelt bzw. den Kinosaal verlassen.

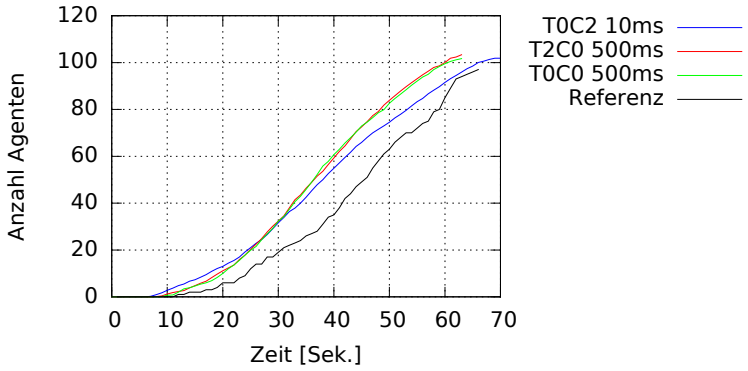


Abbildung 6.4.: Vergleich der Entfluchtungszeiten für alle Ausgänge [angelehnt an Klüpfel (2003) S. 81f.]

Zusammengefasst erzeugt die Simulation Werte, welche nah an den Referenzwerten liegen oder lediglich einen konstanten Fehler aufweisen. Die Ergebnisse sind jedoch nicht nur von der Konfiguration der Detailskalen abhängig, sondern auch stark von der gewählten, maximalen Schrittweite der Simulation. Aus diesem Grund müssen diese separat betrachtet werden.

6.4. Messergebnisse

Für die Analyse und dem Vergleich der Konfigurationen werden im Folgenden die beiden Szenarien separat betrachtet.

6.4.1. Kino-Szenario

Bei der Evaluierung des Kino-Szenarios werden unter anderem der CPU-Bedarf und die Qualität der Ergebnisse betrachtet. Des Weiteren wird untersucht, wie sich die unterschiedlichen maximalen Schrittgrößen auf die unterschiedlichen Konfigurationen und Detailstufen auswir-

ken. Zusätzlich wird überprüft, wie der Einfluss der Anzahl der Simulationsrunden auf die Simulation sind sowie wie der CPU-Bedarf der Kollisionsskala ist.

CPU-Bedarf der Konfigurationen

Bei dem Bedarf der CPU-Zeit existieren deutliche Unterschiede zwischen den einzelnen Konfigurationen sowie auch zwischen den Messungen mit unterschiedlicher maximaler Schrittgröße. Hierbei können drei unterschiedliche Gruppen von Konfigurationen bezüglich der benötigten CPU-Zeit identifiziert werden: Die Konfigurationen mit dem höchsten, mittleren und niedrigsten CPU-Zeit Bedarf.

Beim Vergleich der unterschiedlichen Messreihen mit 10, 50 und 500 ms ist zu erkennen, dass es bei einzelnen Konfigurationen bei steigender maximaler Schrittgröße zu einer Verringerung des Bedarfs an CPU-Zeit kommt. Dies sind die Konfigurationen T0C0, T2C0, C2, C4, C6 und C7, welche von 214,7 bis 266,9 Sekunden bei 10 ms auf 13,3 bis 25,2 Sekunden bei 500 ms maximaler Schrittgröße gefallen sind. Dies ist auch gleichzeitig die Gruppe der Konfigurationen mit dem geringsten Bedarf an CPU-Zeit.

Bei den Konfigurationen C1, C3, C5 und C8 verringert sich mit steigender maximaler Schrittweite ebenfalls die benötigte CPU-Zeit, wobei dies jedoch nicht so deutlich ist, wie bei den vorherigen Konfigurationen. Diese Konfigurationen gehören zu der Gruppe mit dem mittleren CPU-Zeit Bedarf.

Während die vorherigen Konfigurationen noch von 461,5 bis 752,6 Sekunden bei 10 ms auf 320,6 bis 478,8 Sekunden bei 500 ms maximaler Schrittweite gefallen sind, hat sich der Bedarf der CPU-Zeit der Konfigurationen T0C2 und T2C2 nicht erheblich verändert. Diese Konfigurationen benötigen mit 871 bis 1147,3 CPU-Sekunden auch unabhängig von der maximalen Schrittgröße am meisten CPU-Zeit.

Qualität der Ergebnisse

Für die Bewertung der Qualität der Ergebnisse der Messungen werden die Abweichungen der gemessenen Entfluchtungszeiten von den Referenzwerten aus Klüpfel (2003) verwendet.

Dafür werden für die Werte des Maximums, Durchschnitts und Medians in Abbildung 6.5(a), 6.6(a) sowie 6.8(a) jeweils die Beträge der Differenzen zwischen den gemessenen Werten und dem Referenzwert auf Klüpfel (2003) gebildet und als Histogramm dargestellt. Dabei ist die Summe der Differenzen lediglich ein künstlicher Wert, wodurch jedoch verglichen werden kann, wie hoch die Differenz insgesamt über alle Ausgänge ist.

Für die CPU-Zeit aus den Abbildungen 6.5(b), 6.6(b) sowie 6.8(b) wird als Wert der Median verwendet, da dieser weniger von einzelnen Simulationen der Messreihe abhängig ist.

10ms maximale Schrittgröße

In Abbildung 6.5(a) ist zu sehen, dass die Konfiguration T0C2, von den Konfigurationen mit 10 ms maximaler Schrittweite, die geringste Abweichung sowohl bezüglich dem Maximalwert als auch bezüglich dem Median und Durchschnitt zu dem Referenzwert hat. Diese beträgt im Einzelnen lediglich zwischen 1,4 (2,1%) und 5.6 Sekunden (12,4%). Jedoch benötigt diese Konfiguration, wie auch in Abbildung 6.5(b) sowie Tabelle A.1 dargestellt, mit 1147,3 CPU-Sekunden die meiste CPU-Zeit.

Eine etwas größere Abweichung weist die Konfiguration C7 für die Maximalwerte auf. Lediglich der Ausgang A hat eine größere Abweichung, welche bei 5,5 Sekunden (12,2%) liegt. Die anderen Werte liegen mit 0,2 Sekunden (0,3%) Differenz nahezu genau bei dem Referenzwert. Die Differenz der Durchschnitts- und Medianwerte liegt für den Ausgang B und über alle Ausgänge über 10 Sekunden bzw 22%, was der Abweichung der meisten anderen Konfigurationen entspricht. An CPU-Zeit benötigt diese Konfiguration mit 230,4 Sekunden deutlich weniger als die Konfiguration T0C2. Jedoch liefert diese Konfiguration auch lediglich für die Maximalwerte, bezogen auf die Entfluchtungszeit, ähnlich gute Ergebnisse wie die Konfiguration T0C2.

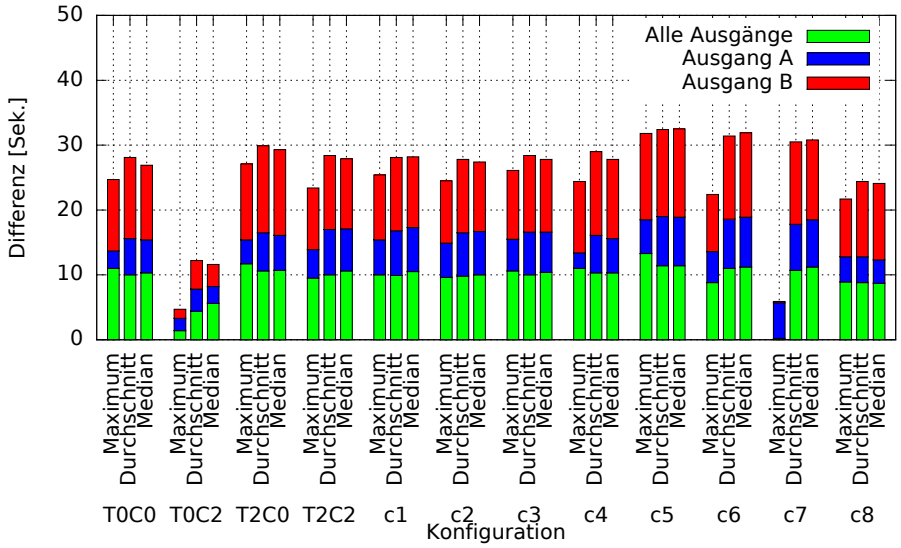
Die Konfiguration C8 liefert ebenfalls noch recht geringe Abweichungen von den Referenzwerten, wobei der Unterschied zu der Konfiguration T0C0 recht gering ist und T0C0 im Gegensatz zu C8 etwa dreimal weniger CPU-Zeit benötigt.

50ms maximale Schrittgröße

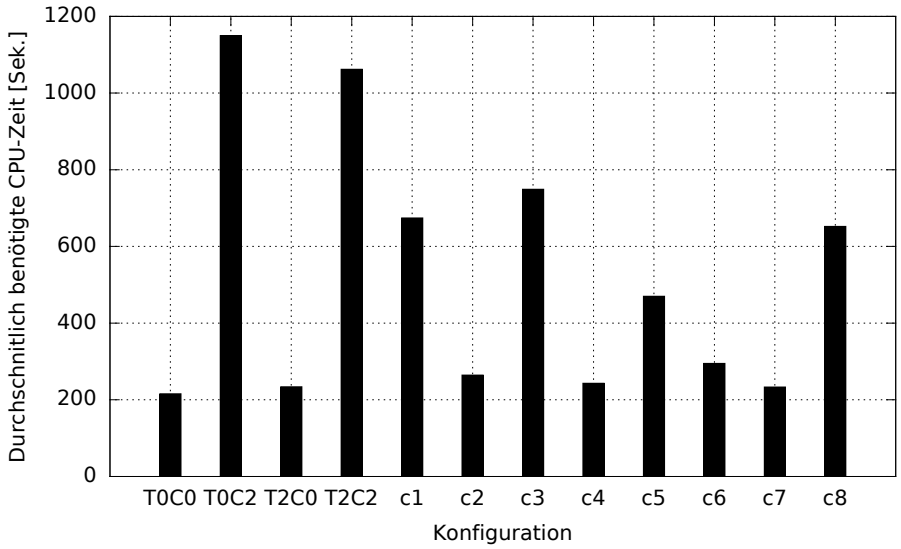
Bei der maximalen Schrittweite von 50 ms ist in Abbildung 6.6(a) zu erkennen, dass die Abweichungen von dem Referenzwert bei der Konfiguration T0C2 im Vergleich zu der Messung mit 10 ms maximaler Schrittweite deutlich angestiegen sind.

Die Konfiguration T2C2 hat sich im Vergleich zu der Messung mit 10 ms bezüglich dem Maximalwert verbessert und weist eine sehr geringe Abweichung von 0,6 (0,9%) bis 0,9 Sekunden (2,1%) auf. Jedoch ist die benötigte CPU-Zeit bei dieser Konfiguration durch die Erhöhung der maximalen Schrittgröße von 10 auf 50 ms vergleichsweise gering gesunken, wie im Vergleich von Abbildung 6.5(b) und Abbildung 6.6(b) deutlich wird. Dadurch ist dies mit 871 CPU-Sekunden die Konfiguration, welche die meiste CPU-Zeit benötigt hat.

Ebenfalls eine geringe, wenn auch mit 5,3 (8,1%) und 6,2 Sekunden (13,8%) Differenz eine höhere Abweichung, liefert die Konfiguration C6 für die Maximalwerte. Diese Konfiguration benötigt auch, im Gegensatz zu T2C2, lediglich 20,5 Sekunden mehr CPU-Zeit, als die Konfiguration T0C0, welche die minimale CPU-Zeit darstellt.

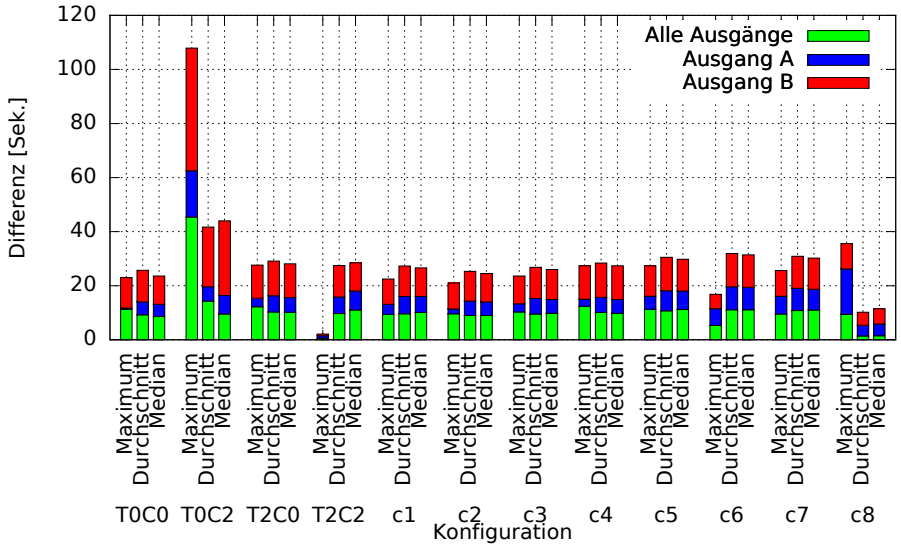


(a) Differenz der Konfigurationen

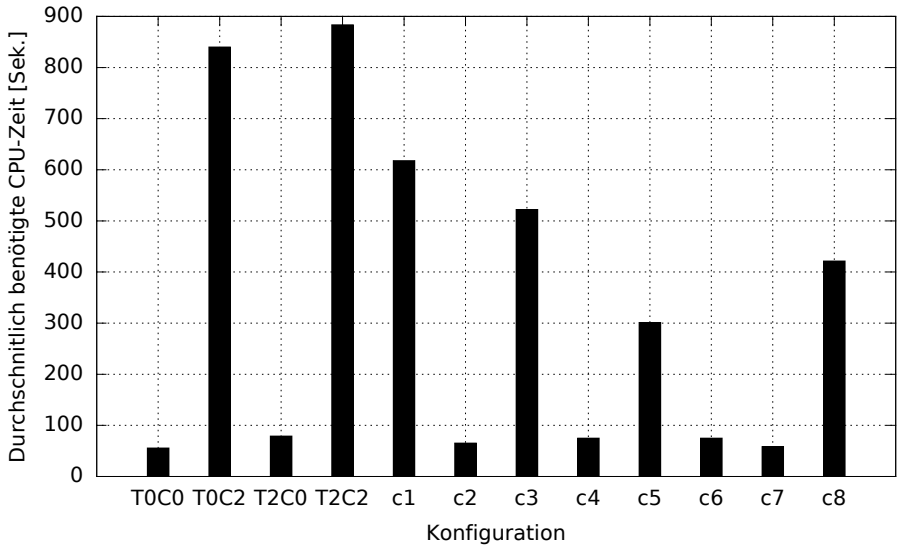


(b) CPU-Zeit der Konfigurationen

Abbildung 6.5.: Differenz und benötigte CPU-Zeit der Konfigurationen bei 10ms maximaler Schrittgröße



(a) Differenz der Konfigurationen



(b) CPU-Zeit der Konfigurationen

Abbildung 6.6.: Differenz und benötigte CPU-Zeit der Konfigurationen bei 50ms maximaler Schrittgröße

Geringe Abweichungen, bezogen auf den Durchschnitt und Median, liefert die Konfiguration C8, die sich bezogen auf die Messung mit 10 ms maximaler Schrittweite verbessert hat. Bei dieser liegen die Abweichungen zwischen 1,4 und 5,7 Sekunden bzw 3,3% und 13,9%. Diese Konfiguration benötigt dabei mit 425,3 Sekunden deutlich weniger CPU-Zeit, als die Konfiguration T2C2 und 222,3 Sekunden weniger als bei der Simulation mit 10 ms maximaler Schrittweite. Dabei hat sich die Abweichung bezüglich dem Maximalwert jedoch erhöht und bezüglich der Durchschnitts- und Medianwerte verringert. Die Agenten verlassen dadurch die Umwelt etwa genauso, wie in der Übung von Klüpfel (2003) (Abbildung 6.7).

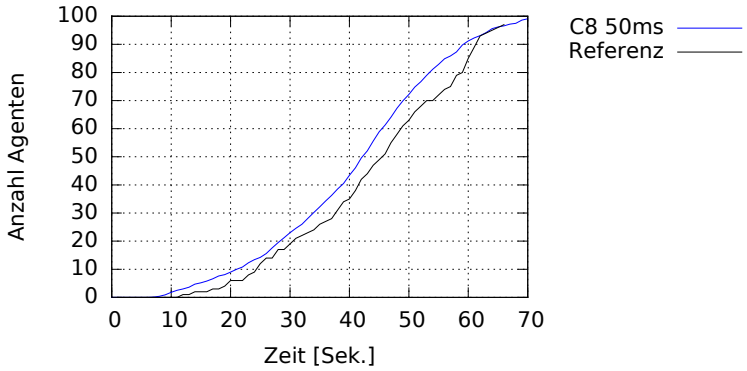


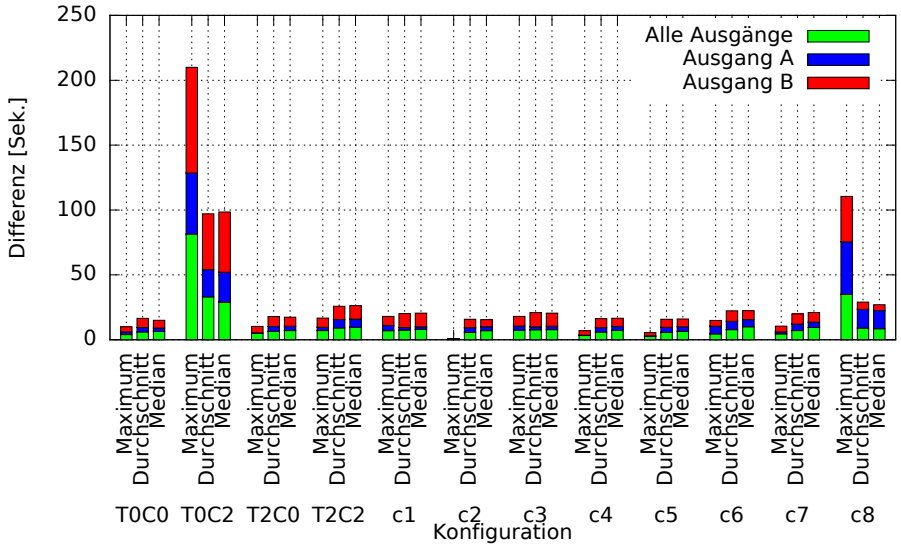
Abbildung 6.7.: Entfluchtungszeit der Konfiguration C8 bei 50ms maximaler Schrittweite für alle Ausgänge [angelehnt an Klüpfel (2003) S. 81f.]

500ms maximale Schrittgröße

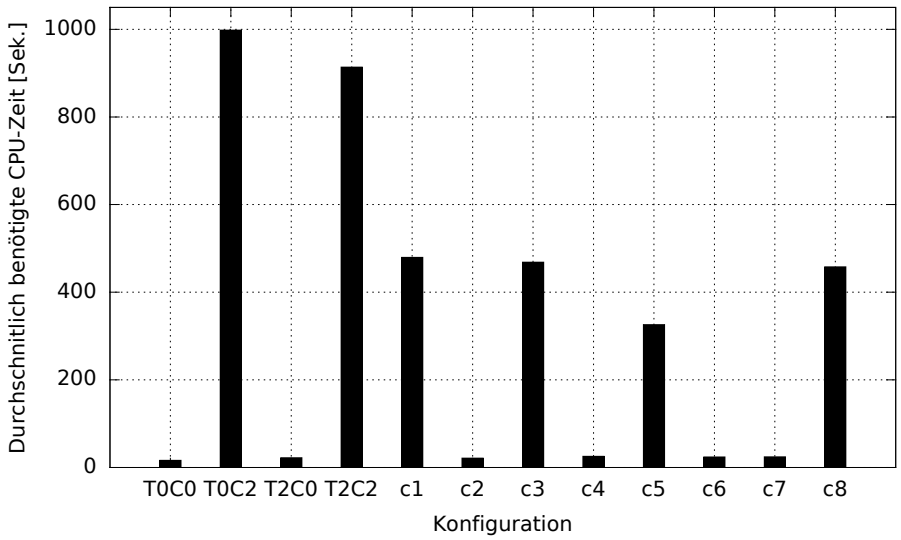
Bei einer maximalen Schrittweite von 500 ms sind die Abweichungen der Konfiguration T0C2, wie bereits bei der Erhöhung der maximalen Schrittweite von 10 auf 50 ms, deutlich angestiegen. Dabei hat sich jedoch nicht, wie aus Abbildung 6.8(b) entnommen werden kann, die benötigte CPU-Zeit verringert.

Ebenfalls haben sich bei der Konfiguration C8, wie auch bereits bei der Erhöhung der maximalen Schrittweite auf 50 ms, die Abweichungen des Maximalwerts sowie die Durchschnitts-, als auch die Medianwerte auf bis zu 40,5 Sekunden bzw. 90% erhöht.

Die geringste Abweichung bei den Maximalwerten liefert die Konfiguration C2 mit einer Abweichung von lediglich 0 bis 0,5 Sekunden (0% bis 0,8%). Des Weiteren benötigt C2 lediglich 17,8 Sekunden CPU-Zeit, was nur 4,5 Sekunden mehr als T0C0 bei 500ms maximaler Schrittweite ist.



(a) Differenz der Konfigurationen



(b) CPU-Zeit der Konfigurationen

Abbildung 6.8.: Differenz und benötigte CPU-Zeit der Konfigurationen bei 500ms maximaler Schrittgröße

Die Konfigurationen C4 und C5 liefern ebenfalls Maximalwerte, welche mit einer Abweichung von 0 bis 3,5 Sekunden (0 bis 5,3%) bzw. mit 0,5 bis 2,5 Sekunden (1,1% bis 3,8%) Abweichung nahe an dem Referenzwert liegen. Dabei benötigt C4 mit 23,3 Sekunden CPU-Zeit deutlich weniger als C5 mit 320,6 Sekunden und gleichzeitig ähnlich viel wie C2. C2 liefert jedoch geringere Abweichungen als C4.

Insgesamt hat sich die Abweichung von allen Konfigurationen, bis auf T0C2 und C8, fast durchgehend durch die Erhöhung der maximalen Schrittweite von 50 auf 500 ms verringert.

Einfluss der Anzahl an Simulationsrunden auf den CPU-Bedarf

Die Anzahl der Simulationsrunden, die durchgeführt werden, ist zu einem Teil ein Faktor bei der benötigten CPU-Zeit für die Simulation. Dies wird an der benötigten CPU-Zeit pro simulierter Sekunde (Abbildung 6.9 sowie 6.10) bei der Konfiguration T0C0 mit 10ms, 50ms und 500ms maximaler Schrittweite deutlich. Mit steigender Schrittgröße nimmt die benötigte CPU-Zeit für das Simulieren von einer Sekunde Simulationszeit ab.

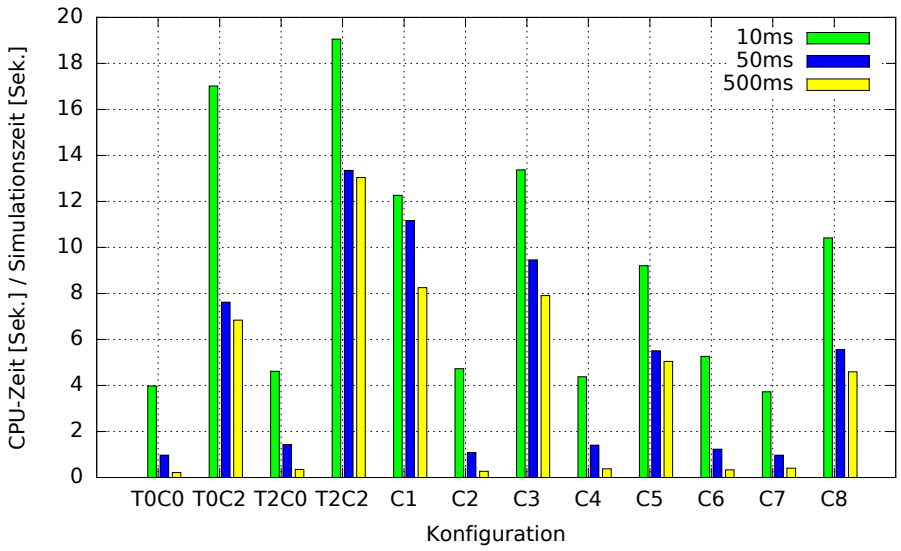


Abbildung 6.9.: Benötigte CPU-Zeit zum Simulieren einer Sekunde Simulationszeit

In Abbildung 6.9 ist auch deutlich zu erkennen, dass bei allen getesteten Konfigurationen die benötigte CPU-Zeit pro Simulationssekunde bei einer größeren Schrittweite geringer ist. Dabei ist bei fast allen Konfigurationen, bis auf bei der Konfiguration C1, die Ersparnis bei einer Erhöhung der maximalen Schrittgröße von 50ms auf 500ms geringer, als der Erhöhung

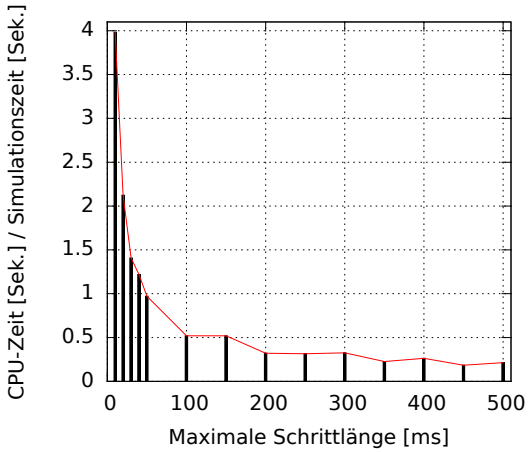


Abbildung 6.10.: Benötigte CPU-Zeit zum Simulieren einer Sekunde Simulationszeit für die Konfiguration T0C0

der maximalen Schrittgröße von 10ms auf 50ms. Lediglich bei der Konfiguration C1 ist dies umgekehrt.

Für die Konfiguration T0C0 ist dies genauer in Abbildung 6.10 dargestellt. Dort bestätigt sich die Beobachtung aus Abbildung 6.9, dass sich die benötigte CPU-Zeit bei einer Erhöhung der maximalen Schrittweite von 50ms auf 500ms nur geringfügig ändert. Die Einsparung der benötigte CPU-Zeit fällt ab einer maximalen Schrittgröße von etwa 50 bis 100ms bei der Konfiguration T0C0 deutlich ab. Dieses Verhalten ist jedoch u.a. von der Konfiguration der Detailskalen und der Anzahl der Agenten abhängig.

In Abbildung 6.11 sind die tatsächlichen Schrittweiten über jeweils 500ms Simulationszeit dargestellt. Hierbei ist deutlich die Phase der ersten vier Sekunden zu erkennen, in der sich noch nicht alle Agenten bewegen. In dieser Zeitspanne liegt die durchschnittliche tatsächliche Schrittweite bei den Simulationen mit 10 und 50ms maximale Schrittgröße unterhalb der späteren durchschnittlichen, tatsächlichen Schrittweite, wenn die Agenten die Umwelt verlassen. Lediglich bei der Konfiguration mit 500ms maximaler Schrittgröße sind die beiden Durchschnittswerte etwa gleich.

Zwischen dem Zeitpunkt, ab dem alle Agenten sich zu den Ausgängen bewegen (ab etwa 4 Sekunden) und dem Zeitpunkt, ab dem die ersten Agenten die Simulation verlassen (etwa acht bis neun Sekunden), ist die tatsächliche Schrittgröße die maximale mögliche Schrittgröße von 10, 50 oder 500ms.

Wie aus Abbildung 6.11(a) entnommen werden kann, beträgt die tatsächliche Schrittweite bei der Konfiguration T0C0 mit einer maximalen Schrittweite von 10ms im Durchschnitt etwa

10ms. Bei den höheren Schrittweiten beträgt die durchschnittliche tatsächliche Schrittweite nicht die maximalen 500ms, sondern etwa 200 bis 300ms (Abbildung 6.11(c)). Die geringere durchschnittliche, tatsächliche Schrittweite ist durch die Aktionen zum Verlassen der Umwelt der Agenten begründet. Diese Aktionen, welche von den Agenten ausgeführt werden, dauern genau eine Millisekunde und führen, nachdem die Aktion beendet wurde, zu einer neuen Simulationsrunde. Ein Verlassen der Umwelt durch einen Agent muss nach der Ausführung dieser Aktion durch die Simulation zu einer neuen Simulationsrunde führen, da der Agent, der die Umwelt verlassen hat, direkt wieder eingeplant werden müsste, wenn dieser Event-Driven simuliert werden sollte.

Der Unterschied bei der tatsächlichen Schrittweite zwischen den Schrittweiten von 10 und 500ms bei der Konfiguration T0C0 bedeutet, dass bei der Konfiguration mit der maximalen Schrittlänge von 10ms im Schnitt etwa 20 bis 30 mal so viele Simulationsrunden durchgeführt wurden, wie bei der Simulation mit 500ms maximaler Schrittweite. Die benötigte CPU-Zeit pro simulierter Sekunde beträgt bei der Konfiguration mit 10ms maximaler Schrittweite 3,3 Sekunden und bei der Konfiguration mit 500ms 0,3 Sekunden. Dies entspricht etwa dem Faktor 11.

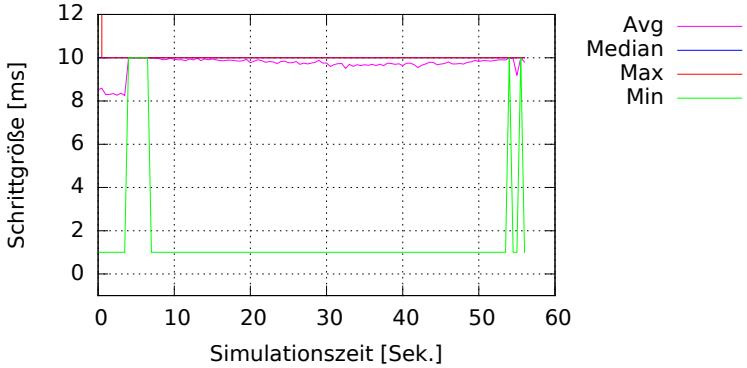
CPU-Bedarf der Kollisionsskala

Ebenfalls hat sich bestätigt, dass die Kollisionserkennung einen nicht vernachlässigbaren Bedarf an CPU-Zeit hat. Dieser Bedarf ist jedoch aufgrund der Messungenauigkeit von 15,625 ms nur schwer direkt messbar. In der Konfiguration T2C2 mit einer maximalen Schrittweite von 10ms Abbildung 6.12 sind jeweils die Sprünge von 15,625 ms, welche durch die Messungenauigkeit verursacht werden, zu sehen. Diese Messungenauigkeit hat die Auswirkung, dass bei vielen Messungen eine Zeit von 0 ms gemessen wurde.

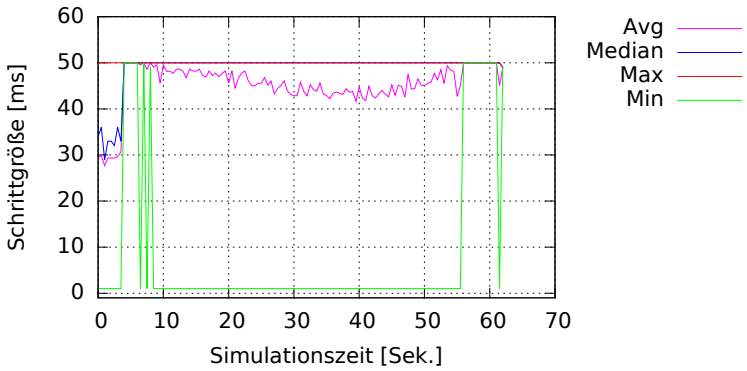
Eine Annäherung an die benötigte CPU-Zeit der Kollisionsskala bei der Konfiguration liefert Abbildung 6.13. Hierbei ist jeweils die gemessene CPU-Zeit über 500ms summiert dargestellt.

Dieser Bedarf ist jedoch von der Anzahl der Agenten in der Umwelt abhängig. So ist in Abbildung 6.14 die Anzahl der Agenten, die sich in der Umwelt befinden, dargestellt.

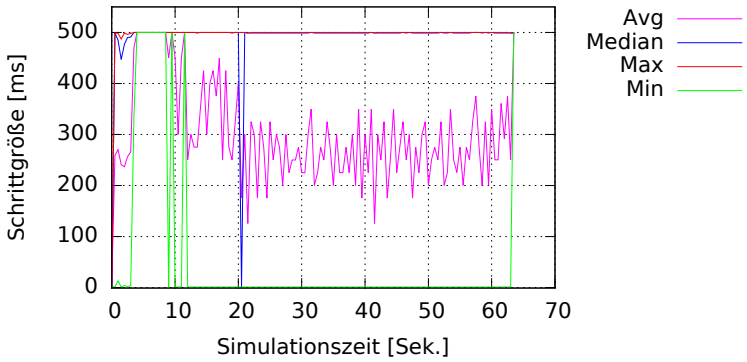
Im Vergleich zwischen der Anzahl der Agenten die sich in der Umwelt befinden aus Abbildung 6.14 und der benötigten CPU-Zeit für die Kollisionserkennung und -behandlung in Abbildung 6.13 ist zu erkennen, dass die benötigte CPU-Zeit mit der Anzahl der Agenten, die die Umwelt verlassen haben, sinkt. Dies hat als Ursache, dass bei weniger Agenten weniger Objekte auf eine Kollision geprüft werden müssen. Des Weiteren kann es sein, dass es bei weniger Objekten zu weniger Kollisionen kommt, die tatsächlich aufgelöst werden müssen.



(a) 10ms maximale Schrittweite



(b) 50ms maximale Schrittweite



(c) 500ms maximale Schrittweite

Abbildung 6.11.: Tatsächliche Schrittweite der Simulation (Konfiguration T0C0, 95% Quantil)

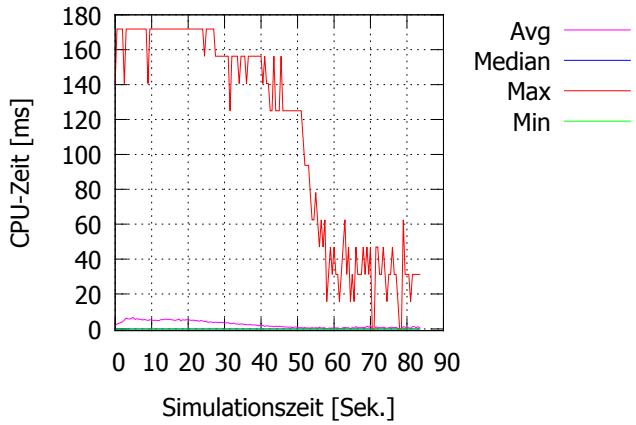


Abbildung 6.12.: Benötigte CPU-Zeit für die Kollisionserkennung und -behandlung (Konfiguration T2C2, 10ms maximale Schrittweite, 95% Quantil)

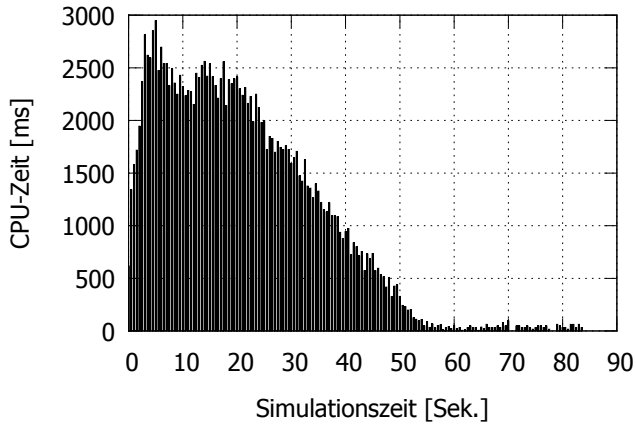


Abbildung 6.13.: Summe der benötigte CPU-Zeit für die Kollisionserkennung und -behandlung über 500ms (Konfiguration T2C2, 10ms maximale Schrittweite, 95% Quantil)

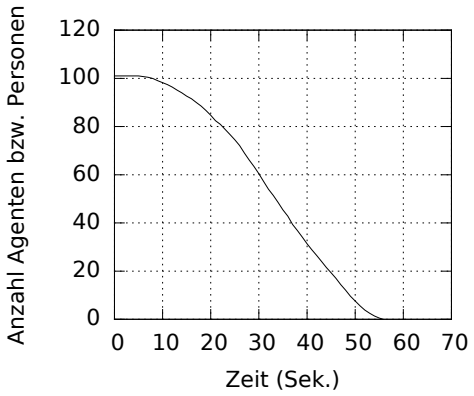


Abbildung 6.14.: Anzahl an Agenten in der Simulation (Konfiguration T2C2, 10ms maximale Schrittlänge)

Zusammenfassung

Es hat sich gezeigt, wie bereits auch in der Analyse vermutet, dass nicht unbedingt die höchsten und aufwändigsten Detailstufen in jeder Situation die besten Ergebnisse liefern. So liefert die Konfiguration T2C2, welche jeweils die höchsten Detailstufen verwendet, bei den maximalen Schrittweiten 10 und 500ms lediglich Ergebnisse, bei denen die Abweichung der Kennzahlen zur Entfluchtungszeit lediglich durchschnittlich waren. Einzig bei der Schrittweite von 50ms liefert diese Konfiguration für die maximal benötigte Entfluchtungszeit geringe Abweichungen. Jedoch ist dies auch die Konfiguration, welche am meisten CPU-Zeit benötigt.

So hat für die Schrittgröße von 10ms die Konfiguration T0C2 die geringste Abweichung von den Entfluchtungszeiten geliefert, benötigt jedoch lediglich geringfügig weniger CPU-Zeit als die Konfiguration T2C2. Eine Alternative ist die Konfiguration C7. Diese benötigt deutlich weniger CPU-Zeit und liefert bezüglich der maximal benötigten Entfluchungszeit ebenfalls eine geringe Abweichung.

Bei der Schrittgröße von 50ms liefert, wie bereits beschrieben, die Konfiguration T2C2 für die maximal benötigte Entfluchtungszeit die geringsten Abweichungen. Eine Alternative ist hier die Konfiguration C8 für die durchschnittliche und im Median benötigte Entfluchtungszeit. Für den Maximalwert lieferte auch die Konfiguration C6 geringe Differenzen.

Für die Schrittgröße von 500ms bietet die Konfiguration C2 für die maximal benötigte Entfluchtungszeit geringe Abweichungen. Eine Alternative stellen hier die Konfigurationen C4 und C5 dar.

	C1	C2	C3	C4	C5	C6
CPU-Zeit						
Maximum	1885,1	131609,6	234019,6	1998,6	119154,5	111465,3
Minimum	1422,4	60354,8	128177,1	1537,7	69423,3	55023,1
Durchschnitt	1515,8	83757,9	155193,1	1683,7	90159,2	73306,1
Median	1485,8	82781,5	148095,2	1671,3	89314,9	69875,6
Ausgang E1						
Maximum	169,7	207,1	215,3	156,9	226,6	208,8
Minimum	50,0	50,0	62,7	50,1	50,1	62,0
Durchschnitt	99,0	125,0	131,9	89,9	133,8	132,5
Median	94,7	133,8	133,1	85,0	147,9	135,6

Tabelle 6.5.: Benötigte CPU- und Entfluchtungszeiten in Sekunden

Des Weiteren hat sich gezeigt, dass die Verwendung der hohen Detailstufe der Kollisionsskala, zusammen mit der Event-Driven Detailstufe der temporalen Skala, zu vielen Simulationsschritten führt, welche aufgrund der Kollisionsskala einen relativ hohen CPU-Zeit Bedarf besitzen.

6.4.2. Nachtclub-Szenario

Bei dem Nachtclub-Szenario wird im Folgenden die Konfiguration C3 als Referenz verwendet, da keine Werte von dem echten Vorfall bekannt sind. Die Konfiguration C3 wird gewählt, da diese jeweils die höchsten Detailstufen verwendet.

Konfiguration C3

Bei der Konfiguration C3 haben, wie in Tabelle 6.5 dargestellt, nach etwa 215,3 Sekunden im Median alle Agenten die Umwelt verlassen. In dieser Zeit sind jedoch die 50 Sekunden zu Beginn der Simulation enthalten, in der die Agenten die Umwelt betreten und sich verteilen.

Für die Durchführung eines Simulationslaufs in der Messreihe hat der Simulationsrechner im Median 148.095,2 CPU-Sekunden bzw 41,14 CPU-Stunden benötigt. Dabei hat im Median die Hälfte der Agenten in 133,1 Sekunden die Umwelt verlassen. Dies weist auf einen gleichmäßigen Abfluss von Agenten aus der Umwelt hin, da abzüglich der 50 Sekunden, ab wann die Agenten begonnen haben die Simulation zu verlassen, dieser Wert etwa die Hälfte der Zeit ist, bis alle Agenten die Umwelt verlassen haben. Dies wird auch durch die Darstellung der Anzahl der Agenten, die die Umwelt bereits verlassen haben, über die Zeit in Abbildung 6.15 deutlich.

Pro Simulationsschritt benötigte diese Konfiguration, wie aus Abbildung 6.16(a) entnommen werden kann, bis zu etwa 1,9 CPU-Sekunden und im Durchschnitt bis etwa 1,4 CPU-Sekunden.

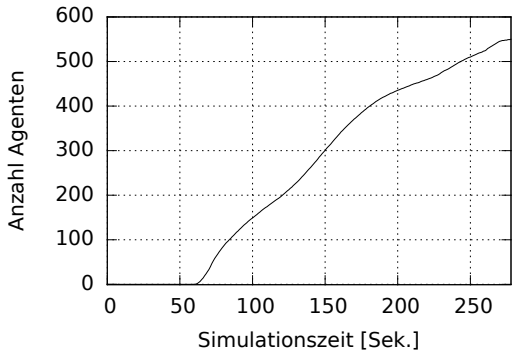


Abbildung 6.15.: Anzahl der Agenten, die die Umwelt verlassen haben (Konfiguration C3)

Dies summiert sich durch die Anzahl an Schritten auf bis zu etwa 600 CPU-Sekunden pro 0,5 Sekunden Simulationszeit (siehe Abbildung 6.16(b)).

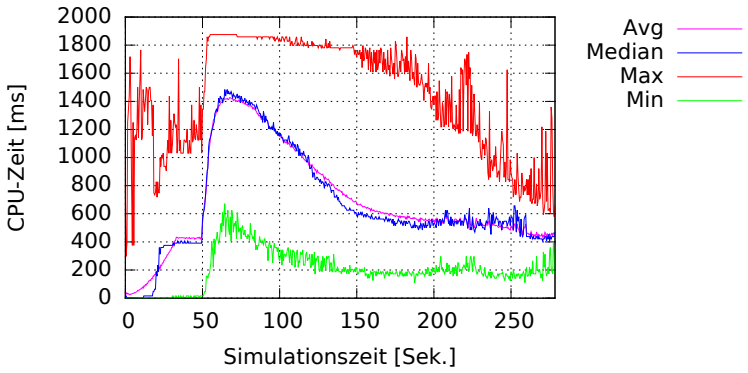
In den Abbildung 6.16(a) und 6.16(b) sind auch die Phasen der Simulation zu erkennen. Bis etwa zum Zeitpunkt t=30 Sekunden betreten neue Agenten die Umwelt, was zu einem Anstieg der CPU-Zeit führt. Bis zum Zeitpunkt t=50 Sekunden bewegen sich die Agenten frei in der Simulation, wodurch die benötigte CPU-Zeit pro simulierter Sekunde Simulationszeit konstant bei fast 400ms CPU-Zeit pro simulierter Simulationsrunde bleibt.

Zum Zeitpunkt t=50 Sekunden beginnt die Ausbreitung des Reizstoffs, was bei den Agenten, die dies wahrnehmen, zu einem Fluchtverhalten führt. Dadurch steigt die benötigte CPU-Zeit pro Simulationsrunde auf den Maximalwert. Dieser ist etwa zum Zeitpunkt t=60 Sekunden erreicht. Ab dann nimmt die benötigte CPU-Zeit wieder ab. Diese langsame Verringerung der benötigten CPU-Zeit hängt damit zusammen, dass die ersten Agenten die Umwelt zum Zeitpunkt t=60 Sekunden verlassen.

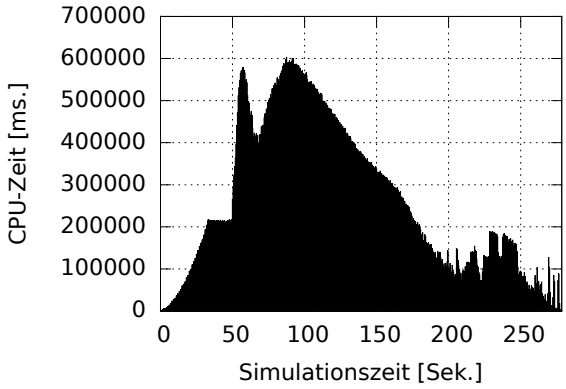
Der Anstieg der benötigten CPU-Zeit pro Simulationsrunde ab dem Zeitpunkt t=50 Sekunden besteht aus etwa 500ms für die Reizstoffausbreitung (Abbildung 6.17) und etwa 400ms für die Berechnung der Kollisionsskala (Abbildung 6.18). Der CPU-Bedarf der Kollisionsskala erhöht sich, da es zu mehr Kollisionen zwischen den Agenten kommt.

Konfiguration C2

Die Konfiguration C2 hat mit 82.781,5 CPU-Sekunden bzw. 22,99 CPU-Stunden im Median 44,1% weniger CPU-Zeit als die Referenzkonfiguration C3 benötigt, wobei über alle Simulationen der Konfiguration im Median, wie in Abbildung 6.19 dargestellt, 383,8 CPU-Sekunden pro Simulationssekunde benötigt werden. Dabei lag die Abweichung der maximalen Entfluchtungszeit zu C3 lediglich bei 8,2 Sekunden (3,8%), zur durchschnittlichen Entfluchtungszeit bei



(a) CPU-Zeit pro Simulationsrunde (95% Quantil)



(b) Summe über 500ms der CPU-Zeit pro Simulationsrunde (95% Quantil)

Abbildung 6.16.: CPU-Zeit pro Simulationsrunde der Konfiguration C3

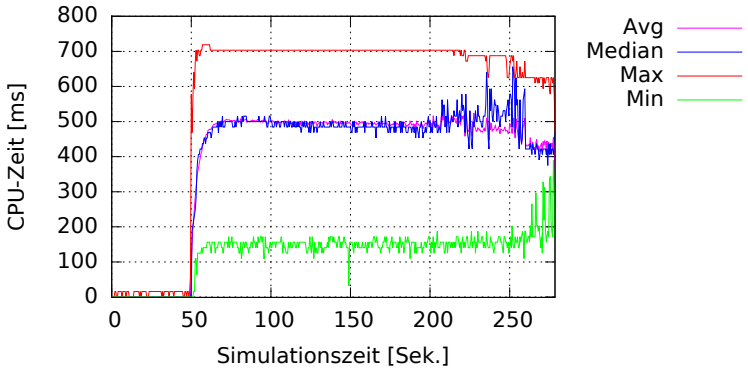


Abbildung 6.17.: CPU-Zeit pro Simulationsrunde der Gasausbreitungsskala (Konfiguration C3, 95% Quantil)

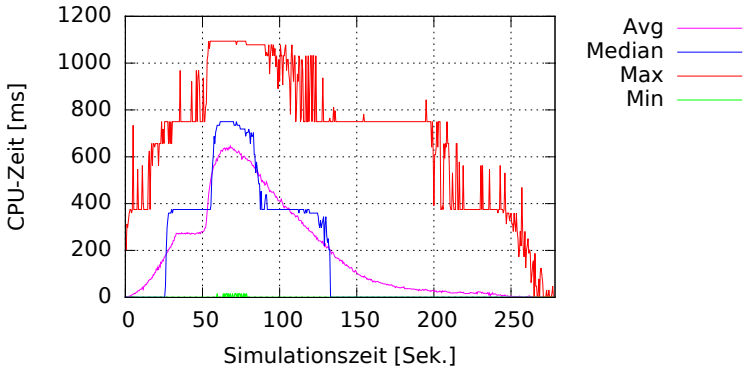


Abbildung 6.18.: CPU-Zeit pro Simulationsrunde der Kollisionsskala (Konfiguration C3, 95% Quantil)

6,9 Sekunden (5,3%) und die Abweichung des Medians der Entfluchtungszeit beträgt lediglich 0,7 Sekunden (0,6%). Damit ist C2 die Konfiguration mit der geringsten Abweichung bezüglich dem Median zur Konfiguration C3.

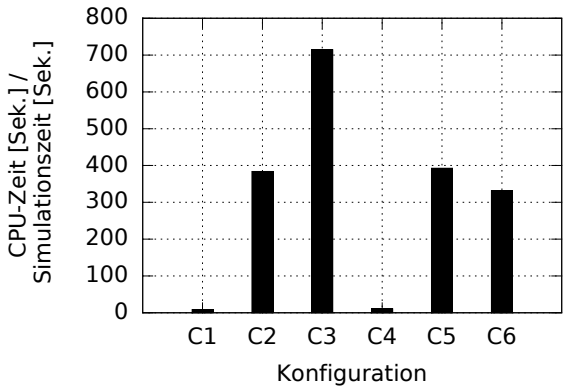


Abbildung 6.19.: Median der durchschnittlichen benötigten CPU-Zeit pro Simulationszeit

Der geringere CPU-Zeit Bedarf pro Simulationsrunde bei dieser Konfiguration aus Abbildung 6.20, der beim Vergleich mit der Konfiguration C3 aus Abbildung 6.16(a) deutlich wird, ist durch die verwendete mittlere Detailstufe für die Gasausbreitungsskala begründet. Dadurch wird zum Zeitpunkt t=60 Sekunden etwa 400ms CPU-Zeit pro Simulationsschritt gespart.

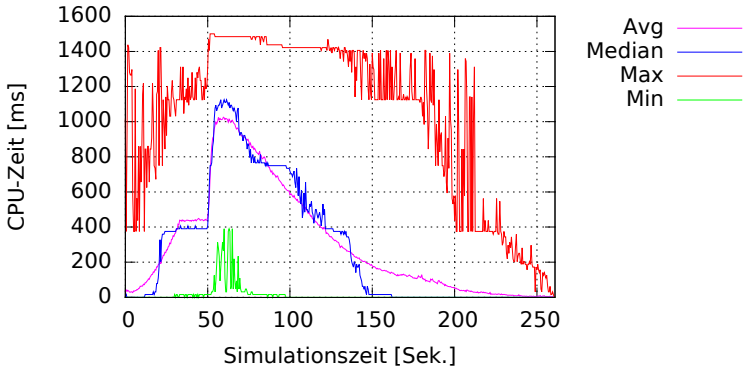


Abbildung 6.20.: CPU-Zeit pro Simulationsschritt (Konfiguration C2, 95% Quantil)

Konfiguration C5

Bei der Konfiguration C5, welche, anders als C3, für die temporale Skala die Soft-Time-Step-Driven Detailstufe und für die Gasausbreitungsskala die Detailstufe mit der sofortigen Ausbreitung verwendet, wurde mit 89.314,9 CPU-Sekunden bzw. 24,81 CPU-Stunden 39,7% weniger CPU-Zeit als bei der Konfiguration C3 verwendet.

Bei der maximalen benötigten Entfluchtungszeit bzw. dem Median der Entfluchtungszeit weicht diese Konfiguration um 11,3 Sekunden (5,2%) bzw. 14,8 Sekunden (11,1%) von dem Referenzwert von C3 ab. Jedoch liefert diese Konfiguration für die durchschnittliche Entfluchtungszeit mit einer Abweichung von 1,9 (1,5%) ebenfalls Ergebnisse, die nahe an dem Referenzwert liegen.

Dabei beträgt die durchschnittlich benötigte CPU-Zeit pro Simulationsrunde zum Zeitpunkt t=60 Sekunden im Maximum, wie in Abbildung 6.21 dargestellt, fast 1000ms und damit etwa 400ms weniger als bei der Konfiguration C3. Diese Ersparnis kommt durch die Reduzierung der Detailstufe der Gasausbreitungsskala auf die mittlere Detailstufe zustande. Das führt zu einer Einsparung von 500ms pro Simulationsrunde. Von diesen 500ms werden jedoch, wie in Abbildung 6.22 dargestellt, 100ms mehr für die Ausführung der Agenten benötigt. Dieser höhere Bedarf an CPU-Zeit für die Ausführung der Agenten kommt durch die Umstellung der temporalen Skala auf die Soft-Time-Step-Driven Detailstufe zustande. Da bei dieser die Agenten möglichst zu festen Zeitschritten gemeinsam ausgeführt werden, werden im Durchschnitt mehr Agenten pro Simulationsrunde ausgeführt, was mehr CPU-Zeit benötigt.

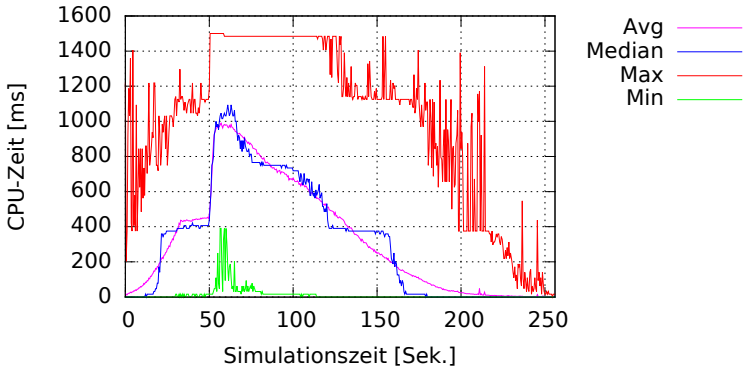


Abbildung 6.21.: CPU-Zeit pro Simulationsschritt (Konfiguration C5, 95% Quantil)

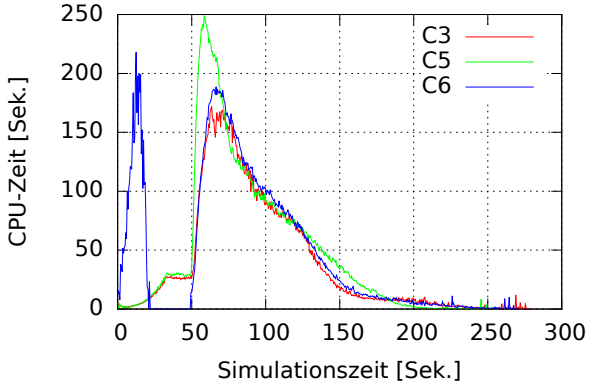


Abbildung 6.22.: CPU-Zeit pro Simulationsrunde der Agentenausführung (95% Quantil)

Konfigurationen C4 und C1

Die Konfigurationen C4 und C1, welche, neben der Detailstufe mit der sofortigen Ausbreitung für die Gasausbreitungsskala, die niedrige bzw. mittlere Detailstufe für die Kollisionsskala verwenden, benötigten lediglich 1.671,3 bzw. 1.485,8 CPU-Sekunden. Damit sind dies die Konfigurationen, welche den geringsten Bedarf an CPU-Zeit aufweisen. Dabei ist der Bedarf der Konfiguration C1 mit 8,73 CPU-Sekunden pro simulierter Sekunde Simulationszeit noch geringer, als bei der Konfiguration C4 mit 10,71 CPU-Sekunden pro Simulationssekunde.

Diese Konfigurationen weisen jedoch gleichzeitig die höchste Abweichung der Entfluchtungszeiten mit bis zu 45,6 bzw. 58,4 Sekunden auf.

Konfiguration C6

Eine etwas komplexere Auswahl der Detailstufen besitzt die Konfiguration C6, welche abhängig vom Ort und der Zeit die Detailstufe auswählt. Der Fokus wurde bei dieser Konfiguration auf den Bereich der Tanzfläche sowie des Ausgangs ab 49,5 Sekunden gelegt. Lediglich die Gasausbreitungs-Skala verringert nach 80 Sekunden in dieser Konfiguration wieder die Detailstufe.

Dabei benötigt diese Detailstufe mit 69.875,6 CPU-Sekunden oder 19,41 CPU-Stunden 52,82% weniger CPU-Zeit als C3. Die Abweichung von den Referenzwerten liegt hierbei zwischen 0,6 und 6,5 Sekunden, was 0,5 bis 3% entspricht.

Beim Vergleich der benötigten CPU-Zeit pro Simulationsrunde zwischen C3 (Abbildung 6.16(a)) und dieser Konfiguration in Abbildung 6.23 ist zu erkennen, dass vor dem Zeitpunkt 50 Sekunden die benötigte CPU-Zeit deutlich höher liegt. Nach dem Zeitpunkt 50 Sekunden ändert sich dies und diese Konfiguration benötigt etwa 300ms weniger CPU-Zeit pro Simulationsrunde als die Konfiguration C3.

Beim Vergleich der Anzahl der Simulationsschritte in Abbildung 6.24 wird deutlich, dass, durch die geänderte temporale Skala, diese Konfiguration deutlich weniger Simulationsschritte durchführt, als die Konfiguration C3. Dadurch wird, wenn man die aufsummierte CPU-Zeit pro 500ms Simulationszeit zwischen der Konfiguration C3 (Abbildung 6.16(b)) und C6 in Abbildung 6.25 vergleicht, deutlich, dass diese Konfiguration durch die Reduktion der temporalen Skala bis zum Zeitpunkt 49,5 Sekunden die Anzahl der Simulationsschritte und dadurch die benötigte CPU-Zeit deutlich reduzieren kann.

Bei der Betrachtung der benötigten CPU-Zeit der Gasausbreitungs-Skala in Abbildung 6.26 führt ebenfalls die Fokussierung dieser Konfiguration zu einem verringerten Bedarf der CPU-Zeit. So steigt die benötigte CPU-Zeit pro Simulationsrunde für die Berechnung der Gasaus-

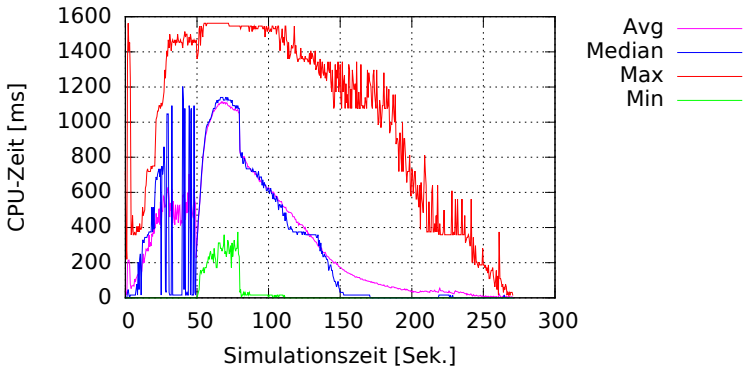


Abbildung 6.23.: CPU-Zeit pro Simulationsrunde (Konfiguration C6, 95% Quantil)

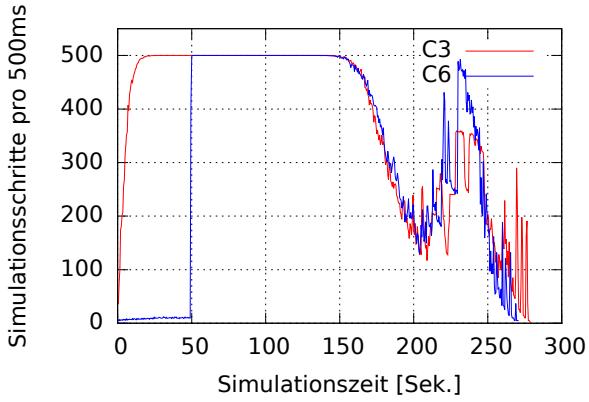


Abbildung 6.24.: Vergleich der Anzahl der Simulationsschritte

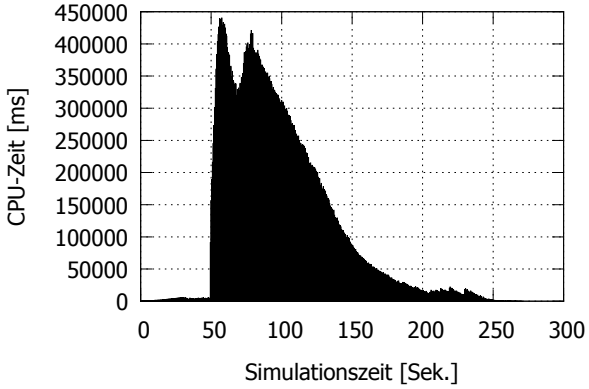


Abbildung 6.25.: Summe über 500ms der CPU-Zeit pro Simulationsrunde (Konfiguration C6, 95% Quantil)

breitung bei dieser Konfiguration lediglich auf etwa 250ms, anstatt, wie bei der Konfiguration C3, auf 500ms. Nach 80 Sekunden wird bei dieser Konfiguration die geringste Detailstufe der Skala verwendet, was dazu führt, dass nahezu keine CPU-Zeit mehr benötigt wird.

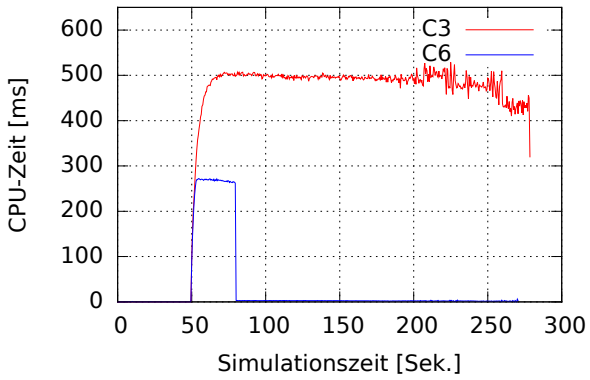


Abbildung 6.26.: Vergleich der durchschnittlich benötigten CPU-Zeit pro Simulationsrunde für die Berechung der Gasausbreitungs-Skala (95 % Quantil)

Für die Kollisionsskala benötigt diese Konfiguration, im Vergleich zu der Konfiguration C3, ebenfalls bis zum Zeitpunkt 49,5 Sekunden, mit bis zu 400ms CPU-Zeit pro Simulationssekunde deutlich weniger CPU-Zeit. Ab 49,5 Sekunden benötigt die Konfiguration C6, wie in Abbildung 6.27 dargestellt, im Maximum etwa 100ms weniger als die Konfiguration C3. Der Unterschied zwischen den beiden Konfigurationen in Abbildung 6.27 nimmt mit der Simulationszeit ab. Das kommt daher, dass immer weniger Agenten sich nicht in dem Bereich des Ausgangs befinden und dadurch mit einer geringeren Detailstufe simuliert werden.

Zusammenfassung

Die Entfluchtungszeiten der Konfiguration C3 wurden aufgrund der, in jeder Skala verwendeten, höchsten Detailstufe als Referenzwerte für die weitere Auswertung verwendet. Bei der Auswertung wurde für die Bewertung der Qualität der Ergebnisse, wie bereits auch bei dem Kino-Szenario, die Abweichungen der Entfluchtungszeiten von den Referenzwerten verwendet.

Es hat sich gezeigt, dass die beiden Konfigurationen C1 und C4 zwar den geringsten Ressourcenbedarf aufweisen, jedoch zu ungenaue Ergebnisse liefern.

Die anderen drei Konfigurationen waren je nach betrachtetem Entfluchtungswert unterschiedlich gut. So ist die Konfiguration C2 bezogen auf den Median der Entfluchtungszeit am nächsten an dem Referenzwert von C3. Für die durchschnittliche und maximale Entfluchtungszeit liefert die Konfiguration C6 die Werte mit den geringsten Abweichung zu C3. Die

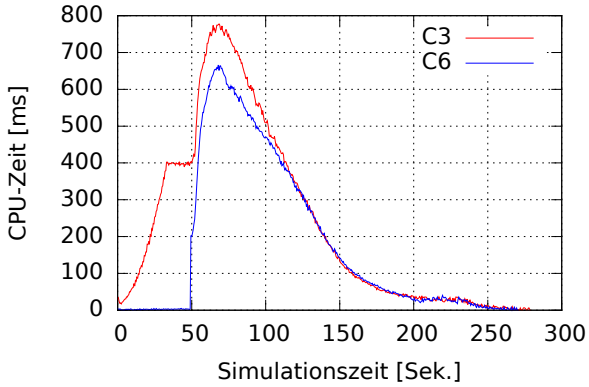


Abbildung 6.27.: Vergleich der durchschnittlich benötigten CPU-Zeit für die Kollisionsskala (95 % Quantil)

Konfiguration C5 liefert bei einem ähnlichen Bedarf an CPU-Zeit etwas bessere Werte für die durchschnittliche Entfluchtungszeit.

Alle drei Konfigurationen benötigen dafür weniger CPU-Ressourcen als C3, wobei C5 von den drei Konfigurationen die meisten und C6 die wenigsten CPU-Ressourcen benötigt hat.

Durch die Konfiguration von C6, wo der Fokus auf der Tanzfläche und dem Ausgang gelegt wurde, bietet C6 die geringsten Abweichungen bezogen auf die Entfluchtungszeit bei dem gleichzeitig geringsten Bedarf an CPU-Zeit. Dadurch konnten in dieser Simulation durch die Konfiguration C6 bei einer maximalen Abweichung von 3% 52,82% CPU-Zeit eingespart werden.

6.4.3. Betrachtung der Detailskalen

Nachdem in den vorherigen beiden Kapitel der Schwerpunkt auf der Evaluierung der Konfigurationen lag, wird im folgenden Kapitel der Schwerpunkt auf der Betrachtung der einzelnen Skalen gelegt.

Temporale Skala

Durch die Betrachtung der Ergebnisse der beiden Szenarien wird deutlich, dass die temporale Skala eine starke Auswirkung sowohl auf die Entfluchtungszeit als auch auf die Genauigkeit der Simulation aufweist.

Bei der Betrachtung der niedrigsten, der Hard-Time-Step-Driven Detailstufe, wird deutlich, dass die Simulation pro Zeitschritt im Durchschnitt mehr CPU-Zeit benötigt, als in der höchsten Detailstufe. Der Grund dafür ist, dass die Agenten nicht, wie bei der hohen Event-Driven

max. Schrittweite	Einsparung (T2C0 - T0C0)
10ms	0,63 CPU-Zeit / Simulationszeit
50ms	0,46 CPU-Zeit / Simulationszeit
500ms	0,14 CPU-Zeit / Simulationszeit

Tabelle 6.6.: Einsparung der benötigten CPU-Zeit pro Simulationszeit von T0C0 gegenüber T2C0

Detailstufe, verteilt über mehrere Simulationsrunden ausgeführt werden, sondern gesammelt zu einer Simulationsrunde.

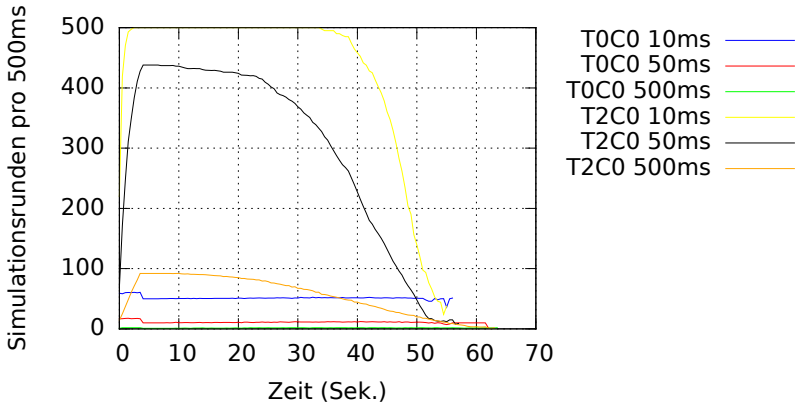


Abbildung 6.28.: Durchschnittliche Anzahl Simulationsschritte pro 500ms

Dadurch verringert sich auch die Anzahl der Simulationsrunden, die ausgeführt werden. Diese Verringerung beträgt, wie in Abbildung 6.28 dargestellt, zwischen der Konfiguration T0C0 und T2C0 etwa den Faktor 10 pro 500ms bei 10 ms maximaler Schrittweite. Bei den Konfigurationen mit der hohen Detailstufe ist zu erkennen, dass die benötigte Anzahl der Simulationsschritte von der Anzahl der Agenten abhängig ist. Der genaue Faktor hängt dabei u.a. von den verwendeten Agenten, den anderen Skalen sowie der maximalen Schrittweite ab.

So hat die verwendete Detailstufe der Kollisionsskala zusammen mit dieser Skala einen nicht zu vernachlässigenden Einfluss auf die Simulation. Im Folgenden wird das Verhalten der benötigten CPU-Zeit pro Simulationszeit zwischen der Hart-Time-Step-Driven und der Event-Driven Detalstufe bei der Verwendung der hohen und niedrigen Detailstufen der Kollisionsskala betrachtet.

Bei der Verwendung der niedrigen Detailskala der Kollisionsskala nimmt die Einsparung der CPU-Zeit pro Simulationszeit mit der maximalen Schrittweite der Simulation ab. So beträgt

max. Schrittweite	Einsparung (T2C2 - T0C2)
10ms	2,03 CPU-Zeit / Simulationszeit
50ms	5,72 CPU-Zeit / Simulationszeit
500ms	6,2 CPU-Zeit / Simulationszeit

Tabelle 6.7.: Einsparung der benötigten CPU-Zeit pro Simulationszeit von T0C2 gegenüber T2C2

die Differenz der benötigten CPU-Zeit pro Simulationszeit bei 10ms noch 0,63 und bei 500ms lediglich noch 0,14 CPU-Zeit pro Simulationszeit (Tabelle 6.6).

Bei der Verwendung der hohen Detailskala der Kollisionsskala steigt die Einsparung der CPU-Zeit pro Simulationszeit mit der maximalen Schrittweite der Simulation. Während bei einer Schrittweite von 10ms die Differenz noch 2,03 beträgt, steigt diese bei 500ms maximaler Schrittweite auf 6,2 CPU-Zeit pro Simulationszeit an.

Der Unterschied, dass bei der niedrigen Detailstufe der Kollisionsskala die Einsparung sinkt und bei der hohen Detailstufe die Einsparung steigt, wird in dem Grund vermutet, dass die benötigte CPU-Zeit pro Simulationsrunde bei der hohen Detailstufe deutlich höher ist, als bei der niedrigen Detailstufe der Kollisionsskala. Aus diesem Grund „profitiert“ eine Konfiguration mit der hohen Detailstufe der Kollisionsskala durch die Einsparung von Simulationsschritten, wie dies durch die Hard-Time-Step-Driven Detailstufe geschieht, stärker, als eine Konfiguration, die die niedrige Detailstufe der Kollisionsskala verwendet. Diese These lässt sich auch auf andere Konfigurationen bzw. Detailstufen übertragen, die die benötigte CPU-Zeit eines Simulationsschritts unabhängig von der Länge des Simulationsschritts erhöhen.

Kollisionsskala

Bei der Betrachtung der Kollisionsskala wird deutlich, dass das Ergebnis bezüglich der Qualität stark von der Temporalen Skala abhängig ist. Das bedeutet, es hat sich gezeigt, dass bei einer hohen Detailstufe der Kollisionsskala auch eine hohe Detailstufe der temporalen Skala verwendet werden sollte, da ansonsten Artefakte, in Form von Agenten die stehen bleiben, auftreten können.

Bei der höchsten Detailstufe der Kollisionsskala ist es, wie erwartet, nicht zu Überschneidungen zwischen Agenten gekommen und die Agenten haben sich auch zu keiner Zeit durch Objekte, wie bspw. Wände, bewegt. Dagegen bewegten sich bei der geringen Detailstufe die Agenten, wie in Abbildung 6.29 dargestellt, auch durch statische Objekte, wie Wände, hindurch.

Der Aufwand für die höchste Detailstufe der Kollisionsskala hängt auch von der Anzahl an Agenten ab. So steigt die durchschnittlich benötigte CPU-Zeit pro Simulationsrunde für

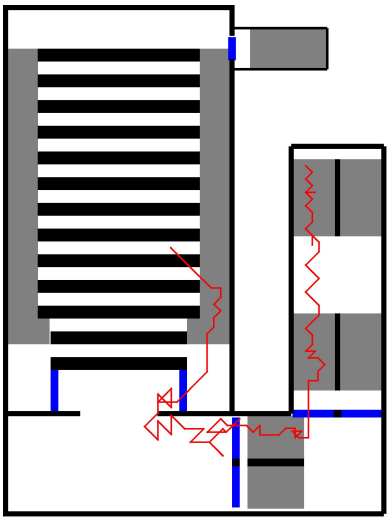


Abbildung 6.29.: Darstellung des Wegs eines Agenten bei der Konfiguration T0C0 und 500ms maximaler Schrittweite

die Ausführung der Kollisionsskala bei dem Kino-Szenario, wie in Abbildung 6.30 dargestellt, mit 101 Agenten auf etwa 8ms bei der Konfiguration T2C2 mit einer maximalen Schrittweite von 10ms. Bei dem Nachtclub-Szenario mit 500 Agenten steigt die benötigt CPU-Zeit, bis alle Agenten die Umwelt betreten haben, wie in Abbildung 6.31 zu sehen ist, auf etwa 400ms an. Im Maximum steigt diese Zeit auf etwa 785ms bei der Konfiguration C3 des Nachtclub-Szenarios an, nachdem das Pfefferspray-Ereignis zum Zeitpunkt 50 Sekunden eingetreten ist und die Agenten flüchten. Durch die Fluchtreaktion kommt es zu einem Stau der Agenten am Ausgang, wodurch es ebenfalls vermehrt zu Kollisionen kommt.

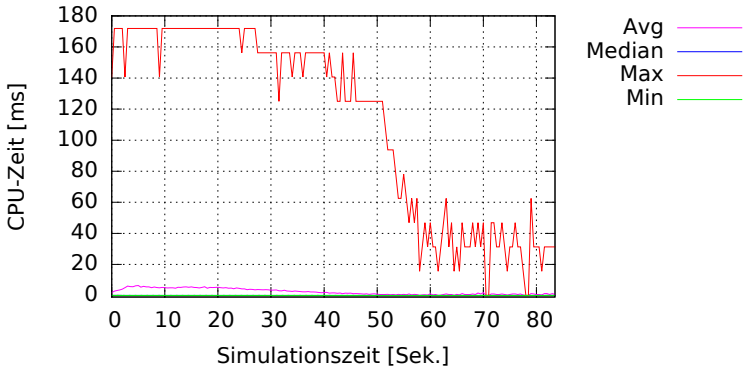


Abbildung 6.30.: Zeit für die Kollisionsskala bei der Konfiguration T2C2 mit 10ms maximaler Schrittweite und 101 Agenten. (95% Quantil)

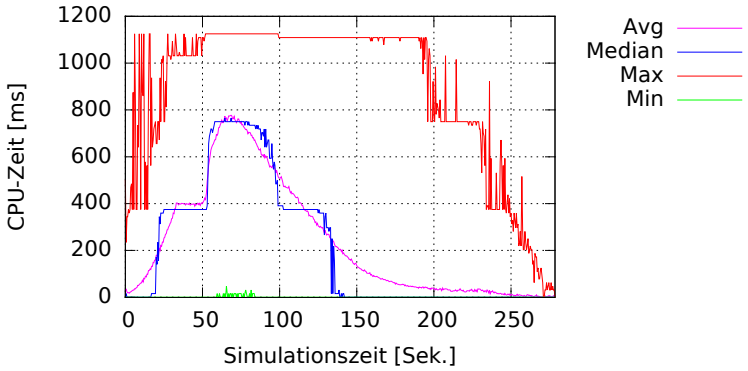


Abbildung 6.31.: Zeit für die Kollisionsskala bei der Konfiguration C3 des Nachtclub-Szenarios mit 10ms maximaler Schrittweite und 500 Agenten. (95% Quantil)

Bei dem Vergleich der Konfigurationen T0C0 und T0C2 ist zu sehen, dass die Konfiguration T0C0 einen höheren CPU-Bedarf pro Simulationsschritt als die Konfiguration T0C2 aufweist. Dagegen weist die Konfiguration T0C2 gegenüber der Konfiguration T0C0, wie in Abbildung 6.32 zu sehen ist, bei einer maximalen Schrittweite von 10ms eine etwa um den Faktor 10 höhere Anzahl an Simulationsschritten auf. Der Grund hierfür ist, dass bei jeder Kollision eine neue Simulationsrunde, unabhängig davon, ob die Agenten in dieser sofort wieder ausgeführt werden oder nicht, eingefügt wird.

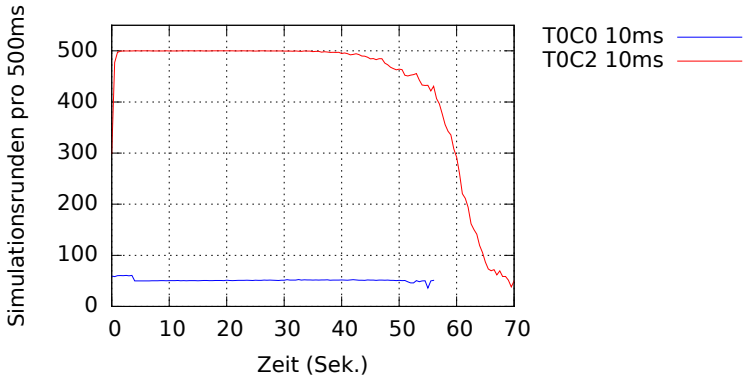


Abbildung 6.32.: Vergleich der Anzahl an Simulationsrunden pro 500ms Simulationszeit zwischen den Konfigurationen T0C0 und T0C2 mit jeweils 10ms maximaler Schrittweite

In Tabelle 6.8 ist die Einsparung der durchschnittlich benötigten CPU-Zeit pro Simulationszeit von der Konfiguration T0C2 gegenüber T0C0 dargestellt. Die Einsparung der Konfiguration

Max. Schrittweite	Einsparung (T0C2 - T0C0)
10ms	13,04 CPU-Zeit / Simulationszeit
50ms	6,64 CPU-Zeit / Simulationszeit
500ms	6,63 CPU-Zeit / Simulationszeit

Tabelle 6.8.: Einsparung der benötigten CPU-Zeit pro Simulationszeit von T0C2 gegenüber T0C0

T0C0 hat dabei mit steigender Schrittgröße abgenommen, wobei der Unterschied zwischen der Schrittweite von 50ms und 500ms nur marginal ist.

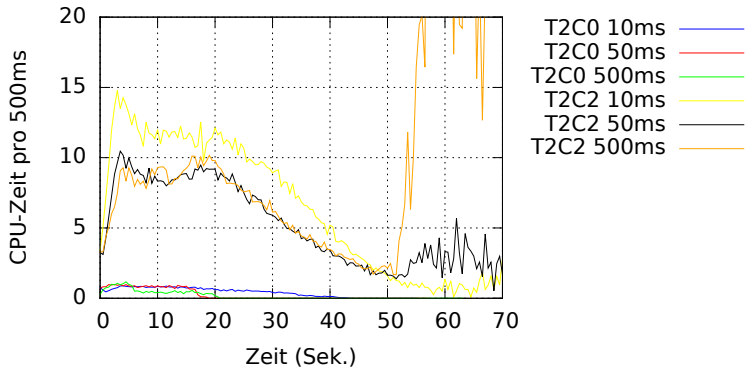


Abbildung 6.33.: Vergleich der Durchschnittlichen CPU-Zeit pro 500ms der Konfigurationen T2C0 und T2C2 (95% Quantil)

Beim Vergleich der Konfigurationen T2C0 und T2C2 in Abbildung 6.33 ist zu sehen, dass die benötigte CPU-Zeit pro Simulationsrunde bei der Konfiguration T2C0 wie erwartet geringer ist, als bei der Konfiguration T2C2. Die Anzahl der Simulationsschritte ist bei der maximalen Schrittweite von 10ms nahezu gleich. Mit steigender maximaler Schrittweite nimmt der Unterschied bei der Anzahl an Simulationsschritten zwischen den beiden Konfigurationen zu, sodass, wie in Abbildung 6.34 zu sehen, bei der maximalen Schrittweite von 500ms die Konfiguration T2C0 die Schrittweite etwa um den Faktor 5 geringer ist, als bei der Konfiguration T2C2. Der Grund ist, dass bei der Konfiguration T2C0 die Agenten immer Schritte von 500ms durchführen, welche zu beliebigen Zeitpunkten starten. D.h. es wird innerhalb von 500ms jeder der 100 Agenten genau einmal eingeplant. Daher beträgt die durchschnittliche Zeit zwischen zwei Simulationsschritten unter der Annahme, dass keine zwei oder mehr Agenten in einem Simulationsschritt eingeplant sind, $\frac{500ms}{100Agenten} = 5ms$. Bei der Konfiguration T2C2 wird durch jede Kollision zwischen Agenten oder Agenten und Objekten ein neuer Simulationsschritt

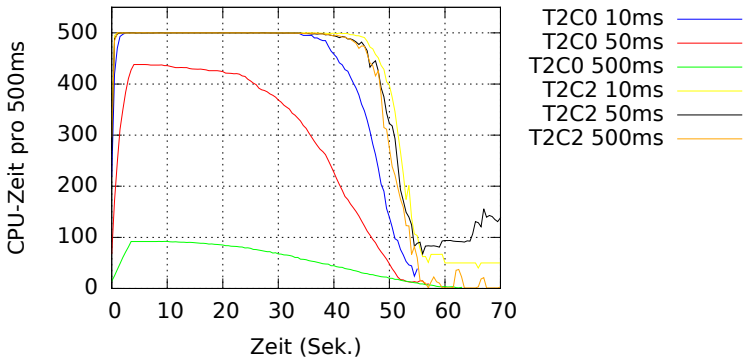


Abbildung 6.34.: Vergleich der durchschnittlich benötigten Anzahl Simulationsschritte pro 500ms zwischen T2C0 und T2C2

Max. Schrittweite	Einsparung (T2C2 - T2C0)
10ms	14,44 CPU-Zeit / Simulationszeit
50ms	11,92 CPU-Zeit / Simulationszeit
500ms	12,69 CPU-Zeit / Simulationszeit

Tabelle 6.9.: Einsparung der benötigten CPU-Zeit pro Simulationszeit von T2C2 gegenüber T2C0

eingefügt, was die Anzahl der Simulationsschritte pro 500ms von maximal 100 auf maximal 500 erhöht.

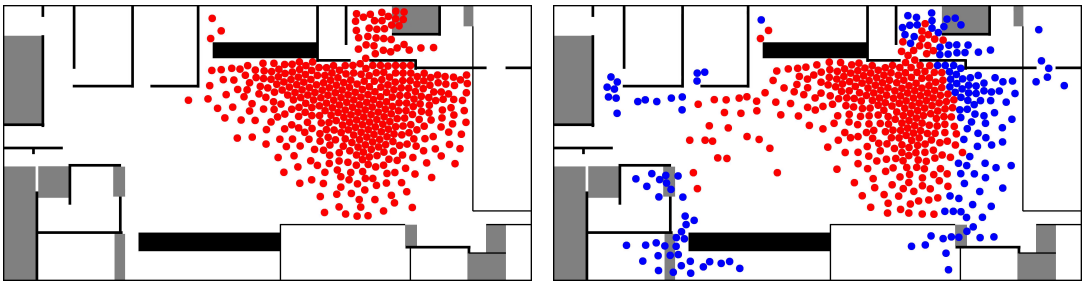
Beim Vergleich der Einsparung der CPU-Zeit pro Simulationszeit der Konfiguration T2C0 gegenüber der Konfiguration T2C2 in Tabelle 6.9 ist zu sehen, dass bei der maximalen Schrittweite von 10ms 14,44 CPU-Sekunden pro Simulationssekunde weniger benötigt wurden. Diese Ersparnis verringert sich bei größeren Schrittweiten auf 11,92 bei 50ms bzw. 12,69 CPU-Sekunden pro Simulationssekunde bei 500ms maximaler Schrittweite. Es ist jedoch kein Trend erkennbar, dass diese Einsparung der CPU-Zeit mit noch größeren maximalen Schrittweiten weiter abnimmt.

Gasausbreitungsskala

Die Gasausbreitungsskala besitzt ebenfalls eine hohe Auswirkung auf die benötigte CPU-Zeit der Simulation.

Dies wird beim Vergleich der Konfiguration C3 mit C2 des Nachtclub-Szenarios deutlich. Der Unterschied zwischen diesen beiden Konfigurationen ist, dass die Konfiguration C2 die Gasausbreitungsskala anstatt mit der hohen, mit der mittleren Detailstufe simuliert. Dadurch

weist die Simulation mit der Konfiguration C2 einen etwa 400ms geringeren CPU-Bedarf pro Simulationsrunde auf. Dieser geringere CPU-Bedarf summiert sich auf etwa 330,44 CPU-Sekunden pro Simulationssekunde im Durchschnitt auf.



(a) Screenshot der Konfiguration C2 (b) Screenshot der Konfiguration C3

Abbildung 6.35.: Screenshots der Simulation zum Zeitpunkt 60 Sekunden. Agenten sind als Kreise dargestellt. Agenten die flüchten, sind rot eingefärbt und Agenten, die sich willkürlich bewegen, sind blau eingefärbt.

Dabei ist die Auswirkung mit den in dieser Arbeit verwendeten Parametern auf die Entfluchtungszeiten recht gering. Ein Effekt im Ausgangsbereich wurde dadurch bei der Auswertung der Simulationsergebnisse nicht beobachtet. So kam es bei der Konfiguration C3 im Ausgangsbereich dazu, dass, während die ersten Agenten von der Tanzfläche versucht haben den Ausgang zu erreichen, noch Agenten aus dem Ausgangsbereich versucht haben, sich entgegengesetzt auf die Tanzfläche zu bewegen. Der Grund ist, dass die Agenten aus dem Ausgangsbereich noch nicht flüchteten, da diese noch nichts von dem Pfefferspray-Ereignis wussten. Dies Verhalten ist zum Zeitpunkt 60 Sekunden in der Abbildung 6.35 dargestellt.

Diese Skala ist von der Kollisionsskala nicht beeinflusst. Lediglich die temporale Skala hat einen Einfluss auf diese Skala. So hängt bspw. der Zeitpunkt, wann ein Agent durch die Wahrnehmung der Effekte dieser Skala flüchtet, von der temporalen Skala ab. Wenn die Simulation mit der Hard-Time-Step-Driven Detailstufe für die temporale Detailstufe simuliert wird, kann ein Agent die Wahrnehmung auch nur zu den Zeitpunkten durchführen, zu denen dieser eingeplant wird. Dies ist unabhängig von der verwendeten Detailstufe der Gasausbreitungsskala.

7. Vergleich

In den Grundlagen in Abschnitt 2.5 wurden unterschiedliche, bereits existierende Ansätze für die Umsetzung von Multi-Level Simulationen beschrieben. Ein Vergleich der unterschiedlichen Ansätze ist auch in Tabelle 7.1 zu finden.

Bei dem Holon-basierten Ansatz wird eine Hierarchie verwendet, wobei jedes Element in der Hierarchie entscheidet, ob jeweils die Kind-Elemente ausgeführt werden (Gaud u. a. (2008b), S.1664 sowie Gaud u. a. (2008a), S.184). Dadurch, dass es möglich ist, dass ein Holon in mehr als einer Hierarchie ist, können auch mehrere Skalen gleichzeitig umgesetzt werden (Cossentino u. a. (2010), S.272). Dabei kann bspw. eine Hierarchie eine Skala abbilden. Durch diese Aufteilung und unterschiedliche Rollen wird jedoch auch gleichzeitig das Konzept des Simulationsmodells komplexer.

Im Gegensatz dazu bietet der in dieser Arbeit vorgestellte Ansatz eine modulare Möglichkeit der Auswahl der Detailstufe, indem dies in einer Komponente gekapselt ist. Durch diese Kapselung wird die Komplexität des Modells verringert und es ist ein einfacher Austausch der konkreten Umsetzung der Auswahl möglich.

Ein weiterer Unterschied ist die Möglichkeit der temporalen Skala. Eine solche Skala wird lediglich von der in dieser Arbeit vorgestellten Simulation umgesetzt. Durch diese Skala ist, je nach Schrittgröße, eine Verringerung der benötigten CPU-Zeit möglich, indem sowohl die Ausführung von Agenten als auch die Ausführung von Aktionen verzögert werden können. Diese Verringerung der benötigten CPU-Zeit hat jedoch eine Verringerung der Genauigkeit zur Folge, da es zu nicht genutzten Zeitspannen kommen kann.

Die Methode, wie die Detailstufe einer Skala bestimmt wird, unterscheidet sich in jeder betrachteten Simulation. In der Holon-basierten Simulation ist eine konkrete Methode nicht definiert. Es ist lediglich definiert, dass die Holone selber bestimmen, ob die Sub-Holone ausgeführt werden, was der Ausführung einer höheren Detailstufe entspricht (Gaud u. a. (2008a), S.184). Dagegen besitzt der agentenzentrierte Ansatz sowohl eine physische als auch psychische Distanz, die in einer Affinitätsfunktion zusammengefasst sind. Ein Wechsel der Detailstufe in einer Skala wird durch ein unter- bzw. überschreiten eines Schwellenwertes ausgelöst (Navarro u. a. (2013), S.163). Der Proxy-basierte Ansatz besitzt als einzige Methode

die Sichtbarkeit der Objekte (Chenney u. a. (2001)). Dies ist dadurch begründet, dass dieser Ansatz aus dem Bereich der Grafik kommt. Diese Bedingung sollte jedoch auch bei diesem Ansatz gegen eine andere Bedingung austauschbar sein. In dieser Arbeit wird ein regelbasierter Ansatz verwendet, wodurch es möglich ist, den Fokus auf bestimmte Bereiche oder Zeiten zu lenken. Es ist jedoch durch die Modularisierung der Detailstufenauswahl möglich diese zu ändern und ggf. über Methoden des maschinellen Lernens die Detailstufen für bestimmte Situationen zu erlernen.

	Holon-basiert	Agentenzentriert	Proxy-basiert	Dieser Ansatz
Diverse				
Schwerpunkt des Ansatzes	Multi-Level Simulation	Multi-Level Simulation	Darstellung	Multi-Level Simulation
Aktionen	Influences-Reaction Modell	Keine Angaben	Keine Angaben	Influences-Reaction Modell
Mehrere Skalen gleichzeitig	Ja	Ja	Nein	Ja
Unterschiedliche Skalen				
Umwelt	Ja	Nein	Nein	Ja
Agenten	Ja	Ja	Ja	Nein
Zeit	Nein	Nein	Nein	Ja
Auswahl der Detailstufe				
Methode	Nicht definiert	Affinitätsfunktion und Schwellenwert	Sichtbarkeit	Regelbasiert, Austausch möglich
Kann durch Domänenexperten definiert werden	Nein	Nein	Nein	Ja
Kombination unterschiedlicher Bedingungen	Ja	Ja	Nein	Ja

Tabelle 7.1.: Vergleich der Ansätze

8. Fazit

In dieser Arbeit wurde untersucht, ob es möglich ist, mit adaptiven Detailstufen in einer Fußgängersimulation den Ressourcenbedarf zu senken. Als Ressource wurde die CPU-Zeit betrachtet.

Dafür wurden die Anforderungen an die Simulation analysiert und definiert. Anhand von zwei Szenarien wurden dann vier unterschiedliche Detailskalen mit jeweils drei unterschiedlichen Detailstufen entwickelt. Die Detailstufen wurden jeweils entsprechend dem erwarteten Bedarf an CPU-Zeit sortiert, sodass die höchste Detailstufe die mit dem erwarteten höchsten CPU-Zeitbedarf darstellt.

Von den entwickelten Skalen wurden die temporale, Kollisions-und Gasausbreitungsskala in einer Fußgängersimulation umgesetzt. Zur Auswahl der jeweiligen Detailstufe einer Skala in einer Situation wurde ein regelbasierter Ansatz verwendet. Durch diesen kann jeweils eine Situation einer Detailstufe und Skala zugeordnet werden.

Für die beiden Szenarien wurden unterschiedliche Konfigurationen der Detailskalen erstellt. Für das Aufstellen der Konfigurationen hat sich gezeigt, dass sowohl Fachwissen bezüglich dem Modell nötig ist, sodass alle gewünschten Effekte auftreten und nicht durch die Konfiguration vernachlässigt werden, als auch technisches Wissen über den Ablauf und die Funktionsweise der jeweiligen Detailskalen. Durch dieses wird sichergestellt, dass möglichst Kombinationen von Detailskalen verwendet werden, sodass die benötigte CPU-Zeit gering bleibt. Für ein Szenario wurde mit vier Konfigurationen eine Validierung durchgeführt. Als Konfigurationen wurden die Vier ausgewählt, welche jeweils einmal die höchste und niedrigste Detailstufe der Skala verwenden.

Mit den Konfigurationen wurden dann jeweils 10 Simulationen durchgeführt. Bei diesen Simulationen wurden unterschiedliche Zeiten und Werte gemessen. Aus diesen Werten wurden dann jeweils die Durchschnitts- und Median-Werte berechnet.

Bei der Auswertung dieser Konfigurationen hat sich gezeigt, dass, wie bereits in der Analyse vermutet, nicht unbedingt die höchste Detailstufe immer die besten Ergebnisse in jeder Situation liefert. Das bedeutet, dass durchaus in bestimmten Situationen eine Kombination von Detailstufen, welche nicht die höchsten sind, bessere Ergebnisse liefern kann. Dies hat

den Grund, dass in dem Modell nicht alles modelliert ist. So ist bspw. in diesem Modell die Beschleunigung und Verzögerung beim Laufen der Agenten nicht modelliert. Durch eine gezielte Kombination von Detailstufen der temporalen und Kollisionsskala kann diese Ungenauigkeit bei der Bewegungszeit ausgeglichen werden, indem die Agenten sich bei einer Kollision für eine kurze Zeit nicht bewegen können.

Des Weiteren hat sich gezeigt, dass es unterschiedliche Kombinationen von Detailstufen gibt, welche zu einem hohen CPU-Bedarf führen oder auch welche, die zu schlechten Ergebnissen führen, sobald die maximale Schrittweite der Simulation zu hoch ist. Dies ist dadurch begründet, dass sich die Kollisions- und die Gasausbreitungsskala jeweils mit der temporalen Skala beeinflussen.

Außerdem hat sich gezeigt, dass sich die Simulationsergebnisse bei den Kennzahlen nicht gleichmäßig verändern. So wurde in dieser Arbeit für die Evaluierung der Qualität der Ergebnisse als Kennzahl die Abweichung der maximalen sowie der durchschnittlichen und im Median benötigten Entfluchtungszeit der Agenten von einem Referenzwert verwendet. So liefern manche Konfigurationen für die maximale Entfluchtungszeit Ergebnisse, welche nahe an dem Referenzwert liegen und andere Konfigurationen geringe Abweichungen für die durchschnittliche und im Median benötigte Entfluchtungszeit.

Es hat sich gezeigt, dass mit der Verwendung von adaptiven Detailskalen CPU-Ressourcen eingespart werden können, ohne dass die Ergebnisse dabei stark von den Referenzwerten abweichen. Dabei muss jedoch beachtet werden, dass diese Optimierung immer mit einem Schwerpunkt auf bestimmte Kennzahlen erfolgen muss. So ist eine Konfiguration, die für einen bestimmten Aspekt ein gutes Ergebnis liefert, nicht unbedingt für andere Kennzahlen aussagekräftig. Daher muss auch bei jeder Konfiguration überprüft werden, ob das Modell noch plausibel ist bzw. valide Ergebnisse liefert.

Das in dieser Arbeit beschriebene und verwendete System eignet sich besonders für Szenarien, in welchen ein eindeutiger Bereich als Fokus definiert werden kann. Dies war in dieser Arbeit bei dem Nachtclub-Szenario der Fall. Dieser Ansatz hat jedoch den Nachteil, dass eine Validierung erschwert wird, da bei jeder Änderung der Detailstufe, welche verwendet wird, sich die Simulationsergebnisse deutlich ändern können.

8.1. Ausblick

In dieser Arbeit gibt es einige mögliche Erweiterungen und Betrachtungen, die aufbauend durchgeführt werden können.

So könnte eine Betrachtung der Auswirkung der Gerüchteskala sowohl auf den Ressourcenbedarf, als auch auf die Entfluchtungszeiten durchgeführt werden. Dabei kann untersucht werden, in wie weit sich die vorhandenen Konfigurationen durch das Hinzukommen einer neuen Detailstufe ändern müssen, um gleich bleibende oder bessere Ergebnisse zu erzielen.

Mit dieser Arbeit kann auch eine weitergehende Analyse des Zusammenspiels der unterschiedlichen Skalen und Detailstufen durchgeführt werden. So kann überprüft werden, ob ein Katalog mit einer Zuordnung zwischen Situationen und zu verwendenden Detailstufen von Skalen aufgestellt werden kann. Dadurch wäre es möglich, den Aufwand für die Erstellung der Konfigurationen zu verringern.

Diese Arbeit hat sich auf Detailstufen zusammen mit der Umwelt fokussiert. Es wäre jedoch auch denkbar, das System um Detailstufen der Agenten zu erweitern, sodass je nach Situation die Agenten unterschiedlich komplex handeln oder auch, wie in dem Holon-basierten Ansatz aus Gaud u. a. (2008a,b), Cossentino u. a. (2007, 2010), Moujahed u. a. (2007) sowie Rodriguez u. a. (2007), die Agenten teilweise zusammen zu fassen.

Darauf aufbauend wäre es auch denkbar, den vorhandenen regelbasierten Ansatz zur Auswahl der Detailstufen zu ersetzen. So kann überprüft werden, ob es möglich ist, die Konfiguration über Lernverfahren zu erlernen und dadurch auch für komplexe Situationen die Zuordnung zu der zu verwendenden Detailstufe durchzuführen.

Momentan wird die Ausführung der Regeln ohne spezielle Optimierungen durchgeführt. Da diese jedoch ggf. sehr häufig evaluiert werden müssen, sollte der Overhead möglichst gering ausfallen. Sobald die Regeln jedoch komplexer werden, sodass dies nicht nur Bereichabfragen sind, sondern, wie bspw. bei der Berechnung der Agentendichte, Werte sind, die aufwändig berechnet werden müssen, steigen die benötigten Ressourcen an. Daher sollte untersucht werden, ob es möglich ist, für die Ausführung der Regeln Strategien zu finden, sodass die Regeln mit möglichst wenig Ressourcen evaluiert werden können. So ist es denkbar, dass diese, ähnlich bei einer Datenbank, über einen Ausführungsplan verfügen.

Auch bei der Reduzierung der Ressourcen durch die adaptive Auswahl der Detailstufe, kann eine Simulation so aufwändig sein, dass diese nicht oder nicht in akzeptabler Zeit auf einem einzelnen Rechner ausgeführt werden kann. Dann muss diese Simulation horizontal skaliert werden. Dabei können jedoch ggf. auch durch die Verwendung von adaptiven Detailstufen die benötigten Ressourcen für die Simulation verringert werden. Dieser Aspekt, wie die adaptive Auswahl der Detailstufen vor allem auch bei komplexeren Bedingungen für die Auswahl bei einer verteilten Simulation möglich ist, muss weitergehend betrachtet werden.

Ein weiterer Aspekt, welcher bei der Evaluierung in dieser Arbeit aufgefallen ist, ist dass die benötigte CPU-Zeit von der Anzahl an Simulationsrunden abhängt. Da es bei der aktuellen

Struktur der Simulation je nach Konfiguration sehr viele Simulationsrunden geben kann, in welchen kein Agent für die Planung ausgeführt wird, steigt bei diesen Konfigurationen die benötigte CPU-Zeit deutlich an. Diese „leeren“ Simulationsrunden sind momentan in der Simulation vorhanden, da die Detailstufe in diesen überprüft werden müssen. Diese „leeren“ Simulationsrunden sollten jedoch möglichst vermieden werden.

Für die Möglichkeit mehr unterschiedliche Szenarien zu simulieren, müssen weitere Skalen und ggf. Detailstufen implementiert werden. Dabei sollte überprüft werden, ob es möglich ist, für die Umsetzung der Detailstufen ggf. auch bereits fertige Bibliotheken oder ähnliches einzubinden, wodurch die Umsetzung der Detailstufe lediglich die Umsetzung eine Adapters für die entsprechende Bibliothek wäre. Der Vorteil ist, dass ggf. die bereits umgesetzten Lösungen eine deutlich bessere Performance aufweisen können. Dadurch würde sich der benötigte CPU-Bedarf der Simulation weiter reduzieren.

A. Anhang

A.1. Numerische Lösung der Diffusionsgleichung

Im Folgenden wird die Herleitung der numerischen Lösung aus der Diffusionsgleichung für den zweidimensionalen Raum beschrieben. Hierfür wurden die Forward und Rearward Difference Methoden aus Laurien und Oertel (2009) (S.63ff.) sowie Wendt (2009) (S.88ff.) verwendet, wobei die Fehler der numerischen Lösung im Folgenden vernachlässigt sind. Die Differentialgleichung ist für einen konstanten Diffusionskoeffizienten D ebenfalls in Shankar (2012) sowohl für den ein- als auch für den zweidimensionalen Fall in Matlab umgesetzt.

Gleichung A.2 zeigt nochmal die Differentialgleichung 4.2 aus Abschnitt 4.2.3 für die Berechnung der Diffusion im zweidimensionalen Raum.

$$\frac{\delta c}{\delta t} = \frac{\delta}{\delta x} * \left(D * \frac{\delta c}{\delta x} \right) + \frac{\delta}{\delta y} * \left(D * \frac{\delta c}{\delta y} \right) \tag{A.1}$$

$$= \frac{\delta}{\delta x} * A + \frac{\delta}{\delta y} * B \tag{A.2}$$

Gleichung A.4 löst die partielle Ableitung von dem Teilterm A nach x mit einem Forward Difference Differenzenquotienten von dem Teilterm A.3 aus A.2. Der Diffusionskoeffizient wird abhängig von der Zeit sowie dem Quell- und Zielpunkt gemacht.

$$A = D * \frac{\delta c}{\delta x} \tag{A.3}$$

$$A(x, y, t) \approx D(x + dx, y, x, y, t) * \frac{c(x + dx, y, t) - c(x, y, t)}{dx} \tag{A.4}$$

Gleichung A.5 bzw. A.6 zeigen die Auflösung der zweiten partiellen Ableitung nach x aus A.2 mit A.4 eingesetzt mit einem Rearward Difference Differenzenquotient.

$$\frac{\delta A(x, y, t)}{\delta x} \approx \frac{D(x + dx, y, x, y, t) * \frac{c(x+dx,y,t) - c(x,y,t)}{dx}}{dx} - \frac{D(x, y, x - dy, y, t) * \frac{c(x,y,t) - c(x-dy,y,t)}{dx}}{dx} \tag{A.5}$$

$$\begin{aligned}\frac{\delta A(x,y,t)}{\delta x} \approx & D(x+dx,y,x,y,t) * \frac{c(x+dx,y,t)-c(x,y,t)}{(dx)^2} \\ & - D(x,y,x-dx,y,t) * \frac{c(x,y,t)-c(x-dx,y,t)}{(dx)^2}\end{aligned} \tag{A.6}$$

Analog zu dem Auslösen des Teilterms A zeigt die Gleichung A.7 den Teilterm B aus Gleichung A.2. Gleichung A.8 löst die erste partielle Ableitung von B nach y mit einem Forward Difference Differenzenquotient. Der Diffusionskoeffizient wird abhängig von der Zeit sowie dem Quell- und Zielpunkt gemacht.

$$B = D * \frac{\delta c}{\delta y} \tag{A.7}$$

$$B(x,y,t) \approx D(x,y+dy,x,y,t) * \frac{c(x,y+dy,t)-c(x,y,t)}{dy} \tag{A.8}$$

Die Gleichung A.9 bzw. Gleichung A.10 zeigt die Auflösung der zweiten partiellen Ableitung nach y aus der Gleichung A.2 mit A.8 und einem Rearward Difference Differenzenquotient.

$$\begin{aligned}\frac{\delta B(x,y,t)}{\delta y} \approx & \frac{D(x,y+dy,x,y,t) * \frac{c(x,y+dy,t)-c(x,y,t)}{dy}}{dy} \\ & - \frac{D(x,y,x,y-dy,t) * \frac{c(x,y,t)-c(x,y-dy,t)}{dy}}{dy}\end{aligned} \tag{A.9}$$

$$\begin{aligned}\frac{\delta B(x,y,t)}{\delta y} \approx & D(x,y+dy,x,y,t) * \frac{c(x,y+dy,t)-c(x,y,t)}{(dy)^2} \\ & - D(x,y,x,y-dy,t) * \frac{c(x,y,t)-c(x,y-dy,t)}{(dy)^2}\end{aligned} \tag{A.10}$$

Gleichung A.12 zeigt die eingesetzten Gleichungen A.6 und A.10 in Gleichung A.2. Die partielle Ableitung nach t in Gleichung A.2 wurde durch einen Forward Difference Differenzenquotienten, wie in Gleichung A.11 dargestellt, aufgelöst. Lösen der partiellen Ableitung nach t durch einen Forward Difference Differenzenquotienten.

$$\frac{\delta c}{\delta t} \approx \frac{c(x,y,t+dt)-c(x,y,t)}{dt} \tag{A.11}$$

$$\begin{aligned}\frac{c(x,y,t+dt)-c(x,y,t)}{dt} =& D(x+dx,y,x,y,t) * \frac{c(x+dx,y,t)-c(x,y,t)}{(dx)^2} \\ &- D(x,y,x-dx,y,t) * \frac{c(x,y,t)-c(x-dx,y,t)}{(dx)^2} \\ &+ D(x,y+dy,x,y,t) * \frac{c(x,y+dy,t)-c(x,y,t)}{(dy)^2} \\ &- D(x,y,x,y-dy,t) * \frac{c(x,y,t)-c(x,y-dy,t)}{(dy)^2}\end{aligned} \tag{A.12}$$

Nach der Umformung von A.12 nach $c(x,y,t+1)$ entsteht die Gleichung A.13.

$$\begin{aligned}c(x,y,t+dt) =& c(x,y,t) + dt \\ &\Bigg(D(x+dx,y,x,y,t) * \frac{c(x+dx,y,t)-c(x,y,t)}{(dx)^2} \\ &- D(x,y,x-dx,y,t) * \frac{c(x,y,t)-c(x-dx,y,t)}{(dx)^2} \\ &+ D(x,y+dy,x,y,t) * \frac{c(x,y+dy,t)-c(x,y,t)}{(dy)^2} \\ &- D(x,y,x,y-dy,t) * \frac{c(x,y,t)-c(x,y-dy,t)}{(dy)^2} \Bigg)\end{aligned} \tag{A.13}$$

A.1.1. Stabilitätsanalyse der numerischen Lösung der Diffusionsgleichung

Im Folgenden wird die Stabilitätsanalyse der numerischen Lösung durchgeführt. Diese orientiert sich an der Stabilitätsanalyse von Wendt (2009) (S. 97ff.), MIT Courseware (2009) sowie Friedrich und Gurevich (2009).

Wenn der Diffusionskoeffizient in der Gleichung A.12 als konstant angenommen wird, ergibt sich die Gleichung A.14.

$$\begin{aligned}\frac{c(x,y,t+dt)-c(x,y,t)}{dt} =& D\Bigg(\frac{c(x+dx,y,t)-2c(x,y,t)+c(x-dx,y,t)}{(dx)^2} \\ &+ \frac{c(x,y+dy,t)-2c(x,y,t)+c(x,y-dy,t)}{(dy)^2} \Bigg)\end{aligned} \tag{A.14}$$

Zur besseren Lesbarkeit wurde die Schreibweise der Funktionen in Gleichung A.15 abgekürzt.

$$\frac{c^{t+dt}_{x,y}-c^t_{x,y}}{dt} = D\left(\frac{c^t_{x+dx,y}-2c^t_{x,y}+c^t_{x-dx,y}}{(dx)^2} + \frac{c^t_{x,y+dy}-2c^t_{x,y}+c^t_{x,y-dy}}{(dy)^2} \right) \tag{A.15}$$

Wenn der jeweilige Fehler, welcher durch die numerische Lösung mit dem Computer eingearbeitet wird, ergibt sich nach Wendt (2009), S.98 die Gleichung A.16

$$\begin{aligned}&\frac{c_{x,y}^{t+dt} + e_{x,y}^{t+dt} - c_{x,y}^{t} - e_{x,y}^{t}}{dt}\\ =&D\Bigg(\frac{c_{x+dx,y}^{t} + e_{x+dx,y}^{t} - 2c_{x,y}^{t} - 2e_{x,y}^{t} + c_{x-dx,y}^{t} + e_{x-dx,y}^{t}}{(dx)^2}\\ &+\frac{c_{x,y+dy}^{t} + e_{x,y+dy}^{t} - 2c_{x,y}^{t} - 2e_{x,y}^{t} + c_{x,y-dy}^{t} + e_{x,y-dy}^{t}}{(dy)^2}\Bigg)\end{aligned} \tag{A.16}$$

Wenn davon ausgegangen wird, dass die Gleichung A.15 die genaue Lösung liefert, ergibt eine Divison der beiden Gleichungen A.15 und A.16 wie in Wendt (2009) S. 98 den Fehler.

$$\frac{e_{x,y}^{t+dt} - e_{x,y}^{t}}{dt} =D\Bigg(\frac{e_{x+dx,y}^{t} - 2e_{x,y}^{t} + e_{x-dx,y}^{t}}{(dx)^2} + \frac{e_{x,y+dy}^{t} - 2e_{x,y}^{t} + e_{x,y-dy}^{t}}{(dy)^2}\Bigg) \tag{A.17}$$

Angelehnt an MIT Courseware (2009) sowie Wendt (2009) S. 98 wird der Fehler in Gleichung A.18 mit $e^{at}e^{ik_mx}e^{il_my}$ approximiert.

$$\begin{aligned}&\frac{e^{a(t+dt)}e^{ik_mx}e^{il_my} - e^{at}e^{ik_mx}e^{il_my}}{dt}\\ =&D\Bigg(\frac{e^{at}e^{ik_m(x+dx)}e^{il_my} - 2e^{at}e^{ik_mx}e^{il_my} + e^{at}e^{ik_m(x-dx}e^{il_my}}{(dx)^2}\\ &+\frac{e^{at}e^{ik_mx}e^{il_m(y+dy)} - 2e^{at}e^{ik_mx}e^{il_my} + e^{at}e^{ik_mx}e^{il_m(y-dy)}}{(dy)^2}\Bigg)\end{aligned} \tag{A.18}$$

Die Gleichung A.18 wird zur Gleichung A.19 umgeformt.

$$\begin{aligned}e^{adt} =&1 + dt * D\\ &\Bigg(\frac{e^{ik_m(x+dx)} - 2 + e^{ik_m(x-dx}}{(dx)^2}\\ &+\frac{e^{il_m(y+dy)} - 2 + e^{il_m(y-dy)}}{(dy)^2}\Bigg)\end{aligned} \tag{A.19}$$

Nach Wendt (2009) S. 99 gilt die Gleichung A.20:

$$cos(k_mdx) =\frac{e^{ik_mdx} + e^{-ik_mdx}}{2} \tag{A.20}$$

Durch einsetzen von Gleichung A.20 in A.19 ergibt sich Gleichung A.21:

$$\begin{aligned}e^{adt} =&1 + 2dt * D\\ &\Bigg(\frac{cos(k_mdx) - 1}{(dx)^2}\\ &+\frac{cos(l_mdy) - 1}{(dy)^2}\Bigg)\end{aligned} \tag{A.21}$$

Der Fehler der numerischen Lösung darf nach Wendt (2009) S. 98 nicht mit der Zeit größer werden, sondern darf maximal gleich bleiben oder sinken. Dies ist in der Gleichung A.22 von Wendt (2009) S.98f. beschrieben:

$$\left|\frac{e^{a(t+dt)}e^{ik_mx}e^{il_my}}{e^{at}e^{ik_mx}e^{il_my}}\right| = \left|a^{adt}\right| \leq 1 \tag{A.22}$$

Durch einsetzen von Gleichung A.21 in A.22 erhält man Ungleichung A.23:

$$\left|1 + 2dt * D\left(\frac{cos(k_mdx) - 1}{(dx)^2} + \frac{cos(l_mdy) - 1}{(dy)^2}\right)\right| \leq 1 \tag{A.23}$$

Bei Ungleichung A.23 müssen, aufgrund des Betrags und der Ungleichheit, wie in Wendt (2009) S.99f., die zwei Fallunterscheidungen in Ungleichung A.24 und A.25 behandelt werden:

$$1 + 2dt * D\left(\frac{cos(k_mdx) - 1}{(dx)^2} + \frac{cos(l_mdy) - 1}{(dy)^2}\right) \leq 1 \tag{A.24}$$

$$1 + 2dt * D\left(\frac{cos(k_mdx) - 1}{(dx)^2} + \frac{cos(l_mdy) - 1}{(dy)^2}\right) \geq -1 \tag{A.25}$$

Der erste Fall in Ungleichung A.24 ist, wie in MIT Courseware (2009) und Wendt (2009) S.99f. beschrieben, immer wahr. Dazu wird diese Ungleichung zu A.26 umgestellt. Die Kosinus-Funktion besitzt den niedrigsten Wert bei 2π mit 1. Dadurch würde, wenn $k_mdx = l_mdy = \pi$ gilt, für die beiden Teilterme aus Ungleichung A.26 $cos(k_mdx)-1 = 0$ sowie $cos(l_mdy)-1 = 0$ gelten.

$$2dt * D\left(\frac{cos(k_mdx) - 1}{(dx)^2} + \frac{cos(l_mdy) - 1}{(dy)^2}\right) \geq 0 \tag{A.26}$$

Der zweite Fall wird, wie in MIT Courseware (2009), zu Ungleichung A.27 umgestellt.

$$2dt * D\left(\frac{1 - cos(k_mdx)}{(dx)^2} + \frac{1 - cos(l_mdy)}{(dy)^2}\right) \leq 2 \tag{A.27}$$

Dabei ist der schlechteste Fall für die Ungleichung A.27 wenn, wie in MIT Courseware (2009), $k_mdx = l_mdy = \pi$ gilt. Dieser Wert eingesetzt in Ungleichung A.27 ergibt die Ungleichung A.28.

$$dt * D\left(\frac{1}{(dx)^2} + \frac{1}{(dy)^2}\right) \leq \frac{1}{2} \tag{A.28}$$

Ungleichung A.28 nach dt umgestellt, ergibt die Ungleichung A.29.

$$dt \leq \frac{1}{2D\left(\frac{1}{(dx)^2} + \frac{1}{(dy)^2}\right)} \tag{A.29}$$

A.2. Messdaten Kino-Szenario

	Ref	T0C0	T0C2	T2C0	T2C2	C1	C2	C3	C4	C5	C6	C7	C8
CPU-Zeit													
Maximum		230,3	1227,4	257,9	1101,9	706,7	293,6	797,4	269,3	496,7	317,7	263,7	692,6
Minimum		206,7	1080,8	176,7	973,1	599,1	250,0	705,5	216,7	443,2	274,3	212,4	620,1
Durchschnitt		215,9	1149,9	233,9	1062,1	674,4	264,6	749,1	243,3	470,0	295,0	233,4	652,5
Median		214,7	1147,3	246,0	1076,1	685,9	266,9	752,6	239,0	461,5	295,0	230,4	647,6
Alle Ausgänge													
Maximum	66	55,0	67,4	54,3	56,5	56,0	56,4	55,4	55,0	52,7	57,2	65,8	74,9
Minimum		8,6	8,2	8,1	8,6	8,4	8,5	9,0	9,5	8,2	8,2	8,2	9,3
Durchschnitt	45	34,0	39,6	33,4	34,0	34,1	34,2	34,0	33,7	32,6	33,0	33,3	35,2
Median	44	34,7	39,4	34,3	34,4	34,5	35,0	34,6	34,7	33,6	33,8	33,8	36,3
Ausgang A													
Maximum	45	42,3	46,9	41,3	40,6	39,6	39,7	40,1	42,6	39,8	40,2	39,5	48,9
Minimum		8,6	8,2	8,1	8,6	8,4	8,5	9,0	9,5	8,2	8,2	8,2	9,3
Durchschnitt	31,1	25,5	27,7	25,2	24,1	24,2	24,4	24,5	25,3	23,5	23,5	24,0	27,1
Median	31	25,9	28,4	25,6	24,5	24,2	24,3	24,8	25,7	23,5	23,3	23,7	27,4
Ausgang B													
Maximum	66	55,0	67,4	54,3	56,5	56,0	56,4	55,4	55,0	52,7	57,2	65,8	74,9
Minimum		26,5	25,8	25,0	26,5	25,7	26,1	25,5	26,3	27,5	26,0	25,4	25,5
Durchschnitt	53,1	40,6	48,8	39,7	41,7	41,8	41,8	41,3	40,2	39,7	40,3	40,4	41,5
Median	53	41,5	49,6	39,8	42,2	42,1	42,3	41,8	40,8	39,4	40,0	40,7	41,2

Tabelle A.1.: Benötigte CPU Zeiten und Entfluchtungszeiten in Sekunden bei einer maximalen Schrittweite von 10ms [angelehnt an Klüpfel (2003) (S.78)]

	Ref	T0C0	T0C2	T2C0	T2C2	C1	C2	C3	C4	C5	C6	C7	C8
CPU-Zeit													
Maximum		80,0	869,5	109,9	995,5	681,1	108,5	548,7	95,2	366,3	104,5	75,3	452,8
Minimum		47,0	773,9	70,3	829,2	552,5	54,2	481,5	63,0	272,2	61,7	50,8	352,7
Durchschnitt		55,6	840,1	79,0	883,4	617,9	65,3	522,3	74,9	301,4	75,0	58,4	421,4
Median		53,8	852,2	75,6	871,0	612,2	61,1	529,2	76,1	297,6	73,3	55,6	425,3
Alle Ausgänge													
Maximum	66	54,7	111,4	53,8	65,4	56,6	56,4	55,7	53,6	54,7	60,7	56,5	75,4
Minimum		9,2	9,4	8,4	8,5	9,0	8,5	9,0	9,2	8,7	8,7	8,7	9,6
Durchschnitt	45	34,8	58,3	33,7	34,3	34,4	35,0	34,5	33,9	33,3	32,9	33,2	42,6
Median	44	36,4	54,5	34,8	34,0	34,9	36,0	35,2	35,2	33,8	33,9	34,0	43,5
Ausgang A													
Maximum	45	44,6	62,1	41,8	44,1	41,3	43,2	42,0	42,4	40,2	38,8	38,4	61,8
Minimum		9,2	9,4	8,4	8,5	9,0	8,5	9,0	9,2	8,7	8,7	8,7	9,6
Durchschnitt	31	26,3	36,4	25,1	25,0	24,7	25,8	25,3	25,5	23,7	22,6	22,9	35,1
Median	31,1	26,5	37,9	25,6	24,0	25,1	26,0	25,9	25,9	24,2	22,7	23,3	35,3
Ausgang B													
Maximum	66	54,7	111,4	53,8	65,4	56,6	56,4	55,7	53,6	54,7	60,7	56,5	75,4
Minimum		27,0	26,2	26,6	25,1	25,1	26,5	26,3	24,9	27,1	25,7	26,4	27,0
Durchschnitt	53	41,4	75,2	40,3	41,4	41,8	42,1	41,6	40,4	40,7	40,8	41,2	48,3
Median	53,1	42,5	80,6	40,5	42,5	42,4	42,5	41,9	40,5	41,2	41,0	41,5	47,3

Tabelle A.2.: Benötigte CPU Zeiten und Entfluchtungszeiten in Sekunden bei einer maximalen Schrittweite von 50ms [angelehnt an Klüpfel (2003) (S.78)]

	Ref	T0C0	T0C2	T2C0	T2C2	C1	C2	C3	C4	C5	C6	C7	C8
CPU-Zeit													
Maximum		47,0	1053,2	31,8	1002,9	582,8	39,6	512,8	61,1	377,8	52,4	43,7	550,3
Minimum		8,4	953,0	12,8	846,8	431,4	9,5	426,2	13,5	299,6	9,6	14,1	409,3
Durchschnitt		16,2	998,2	22,1	913,6	479,5	21,2	468,6	25,6	325,9	24,2	24,4	457,8
Median		13,3	1005,8	21,4	927,5	472,8	17,8	462,3	23,3	320,6	20,4	25,2	458,1
Alle Ausgänge													
Maximum	66	62,0	147,5	61,1	73,1	59,0	65,5	58,5	62,5	63,5	61,5	61,5	101,0
Minimum		11,0	11,5	10,3	9,7	11,0	10,0	11,0	10,5	11,5	9,5	10,5	12,5
Durchschnitt	45	38,0	77,0	37,5	34,9	36,6	38,3	36,4	38,1	38,3	36,2	36,9	53,0
Median	44	38,5	74,0	37,9	35,2	37,0	38,0	37,2	37,7	38,5	35,0	35,5	53,5
Ausgang A													
Maximum	45	43,0	92,0	44,5	42,5	49,0	45,0	48,0	45,0	44,5	39,1	43,5	85,5
Minimum		11,0	11,5	10,3	9,7	11,0	10,0	11,0	10,5	11,5	9,5	10,5	12,5
Durchschnitt	31	27,7	52,1	27,4	24,7	29,1	27,5	28,8	27,7	27,2	24,7	26,2	45,5
Median	31,1	28,5	54,0	27,7	24,9	29,0	28,0	28,4	28,0	27,5	25,5	27,0	45,0
Ausgang B													
Maximum	66	62,0	147,5	61,1	73,1	59,0	65,5	58,5	62,5	63,5	61,5	61,5	101,0
Minimum		28,5	31,0	25,4	27,9	25,6	26,0	25,0	27,0	26,0	25,0	24,0	29,0
Durchschnitt	53	46,0	96,2	45,4	42,8	42,3	46,6	42,2	46,1	46,9	45,0	45,2	58,7
Median	53,1	47,0	99,5	46,0	42,6	42,5	47,5	42,9	46,7	47,0	46,1	45,5	57,5

Tabelle A.3.: Benötigte CPU Zeiten und Entfluchtungszeiten in Sekunden bei einer maximalen Schrittweite von 500ms [angelehnt an Klüpfel (2003) (S.78)]

Literaturverzeichnis

[Abschlussbericht Loveparade 2010] Stadt Duisburg; Heuking Kühn Lüer Wojtek: *Bericht zur Untersuchung des Verwaltungshandelns auf Seiten der Stadt Duisburg anlässlich der Loveparade.* 2010. – URL https://www.duisburg.de/ratsinformationssystem/bi/vo0050.php?__kvonr=20056110&voselect=20049862. – Zugriff: 20.02.2013

[Baehr und Stephan 2008] Baehr, Hand D. ; Stephan, Karl: *Wärme- und Stoffübertragung.* 6. neu bearbeitete Auflage. Springer-Verlag, 2008. – ISBN 978-3-540-87689-2

[Beeler u. a. 2012] Beeler, M.F. ; Aleman, D.M. ; Carter, M.W.: A large simulation experiment to test influenza pandemic behavior. In: *Simulation Conference (WSC), Proceedings of the 2012 Winter*, 2012, S. 1–7. – ISSN 0891-7736

[Brenner u. a. 1998] Brenner, Walter ; Zarnekow, Rüdiger ; Wittig, Hartmut: *Intelligente Softwareagenten - Grundlagen und Anwendungen.* Springer-Verlag Berlin Heidelberg New York, 1998. – ISBN 3-540-63431-2

[Chenney 2004] Chenney, Stephen: Flow Tiles. In: *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2004)

[Chenney u. a. 2001] Chenney, Stephen ; Arikan, Okan ; Forsyth, D: Proxy simulations for efficient dynamics. In: *Proceedings of EUROGRAPHICS* Bd. 2001 Citeseer (Veranst.), 2001

[CNN 2003a] CNN: *Judge blocks charges against E2 owners.* 2003. – URL http://www.cnn.com/2003/US/Midwest/02/18/chicago.nightclub/index.html

[CNN 2003b] CNN: *Survivor: Standing on people's heads.* 2003. – URL http://www.cnn.com/2003/US/Midwest/02/17/club.death.scene.ap/. – Quelle nach Schneider (2011), Zugriff nicht mehr möglich, da diese Quelle nicht mehr existiert.

[Cossentino u. a. 2007] Cossentino, Massimo ; Gaud, Nicolas ; Galland, Stéphane ; Hilaire, Vincent ; Koukam, Abderrafiâa: A Holonic Metamodel for Agent-Oriented Analysis and Design. In: Mařík, Vladimír (Hrsg.) ; Vyatkin, Valeriy (Hrsg.) ; Colombo, ArmandoW. (Hrsg.): *Holonic and Multi-Agent Systems for Manufacturing* Bd. 4659. Springer Berlin Heidelberg, 2007, S. 237–246. – URL http://dx.doi.org/10.1007/978-3-540-74481-8_23. – ISBN 978-3-540-74478-8

[Cossentino u. a. 2010] Cossentino, Massimo ; Gaud, Nicolas ; Hilaire, Vincent ; Galland, Stéphane ; Koukam, Abderrafiâa: ASPECS: an agent-oriented software process for engineering complex systems. In: *Autonomous Agents and Multi-Agent Systems* 20 (2010), Nr. 2, S. 260–304. – URL http://dx.doi.org/10.1007/s10458-009-9099-4. – ISSN 1387-2532

[Crank 1975] Crank, John: *The Mathematics of Diffusion*. 2nd Edition. Oxford University Press, 1975. – ISBN 0 19 853411 6

[Demtröder 2013] Demtröder, Wolfgang: *Experimentalphysik 1 - Mechanik und Wärme*. 6., neu bearbeitete und aktualisierte Auflage. Springer-Ver, 2013. – ISBN 978-3-642-25466-6

[DIVAs Projekt Webseite 2012] *DIVAs Projekt Webseite*. 2012. – URL http://mavs.utdallas.edu/node/4. – Zugriff: 21.07.2012

[Eberly 2008] Eberly, David: Intersection of Convex Objects: The Method of Separating Axes. (2008). – URL http://geometrictools.com/Documentation/MethodOfSeparatingAxes.pdf. – Zugriff: 06.02.2014

[Ferber 1999] Ferber, Jacques: *Mulit-Agent Systems - An Introduction to Distributed Artifical Intelligence*. English Edition. Addison Wesley Longman, 1999

[Ferber und Müller 1996] Ferber, Jacques ; Müller, Jean-Pierre: Influences and Reaction: a Model of Situated Multiagent Systems. In: *Proceedings of the Second International Conference on Multiagent Systems*, AAAI Press, 1996, S. 72–79. – URL http://aaaipress.org/Papers/ICMAS/1996/ICMAS96-009.pdf

[Friedrich und Gurevich 2009] Friedrich, R. ; Gurevich, S.: *Skript: Numerische Methoden für komplexe Systeme, Diffusionsgleichung*. 2009. – URL http://pauli.uni-muenster.de/tp/fileadmin/lehre/NumMethoden/WS0910/ScriptPDE/Heat.pdf. – Zugriff: 31.01.2014

[Gamma 2005] Gamma, Erich: *Design Patterns: elements of reusable object-oriented software*. 32nd printing. Pearson Education Inc., 2005. – ISBN 0-201-63361-2

[Gaud u. a. 2008a] Gaud, N. ; Galland, S. ; Koukam, A.: Towards a Multilevel Simulation Approach Based on Holonic Multiagent Systems. In: *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*, 2008, S. 180–185

[Gaud u. a. 2008b] Gaud, Nicolas ; Galland, Stèphane ; Gechter, Franck ; Hilaire, Vincent ; Kouka, Abderrafiâa: Holonic multilevel simulation of complex systems: Application to real-time pedestrians simulation in virtual urban environment. In: *Simulation Modelling Practice and Theory* 16 (2008), Nr. 10, S. 1659 – 1676. – URL http://www.sciencedirect.com/science/article/pii/S1569190X08001780. – <ce:title>The Analysis of Complex Systems</ce:title>. – ISSN 1569-190X

[Gong und Xiao 2007] Gong, Xiaoguang ; Xiao, Rebin: Research on Multi-Agent Simulation of Epidemic News Spread Characteristics. In: *Journal of Artificial Scoieties and Social Simulation* 10 (2007), Nr. 3, S. 1

[Görz u. a. 2003] Görz, Günther (Hrsg.) ; Rollinger, Claus-Rainer (Hrsg.) ; Schneeberger, Josef (Hrsg.): *Handbuch der Künstlichen Intelligenz*. 4. korrigierte Auflage. Oldenbourg Wissenschaftsverlag GmbH, 2003. – ISBN 3-486-27212-8

[Guo und Tay 2008] Guo, Zaiyi ; Tay, Joc C.: Multi-timescale event-scheduling in multi-agent immune simulation models. In: *Biosystems* 91 (2008), Nr. 1, S. 126 – 145. – URL http://www.sciencedirect.com/science/article/pii/S0303264707001281. – ISSN 0303-2647

[Hall u. a. 2011] Hall, Richard S. ; Pauls, Karl ; McCulloch, Stuart ; Savage, David: *OSGi in action*. Manning Publications Co., 2011. – ISBN 978-1-933988-91-7

[Helbing 1992] Helbing, Dirk: A Fluid-Dynamic Model for the Movement of Pedestrians. In: *Complex Systems* 6 (1992), S. 391–415. – URL http://www.complex-systems.com/pdf/06-5-1.pdf

[Helbing und Moinar 1995] Helbing, Dirk ; Moinar, Peter: Social force model for pedestrian dynamics. In: *PHYSICAL REVIEW E* 51 (1995), Nr. 5

[Java SE Documentation 2014] *Java SE Documentation: Overview of Java SE Monitoring and Management*. 2014. – URL http://docs.oracle.com/javase/7/docs/technotes/guides/management/overview.html. – Zugriff: 30.01.2014

[Javadoc getProcessCpuTime 2013] *Javadoc zu OperatingSystemMxBean.getProcessCpuTime()*. 2013. – URL http://docs.oracle.com/javase/7/docs/jre/api/management/extension/com/sun/management/OperatingSystemMXBean.html#getProcessCpuTime(). – Zugriff: 03.01.2013

[Kirik u. a. 2007] Kirik, Ekaterina ; Yurgel'yan, Tat'yana ; Krouglov, Dmitriy: An intelligent floor field cellular automation model for pedestrian dynamics. In: *Proceedings of the 2007 Summer Computer Simulation Conference*. San Diego, CA, USA : Society for Computer Simulation International, 2007 (SCSC '07), S. 21:1–21:6. – URL http://dl.acm.org/citation.cfm?id=1357910.1358127. – ISBN 1-56555-316-0

[Klüpfel 2003] Klüpfel, Hubert L.: *A Cellular Automaton Model for Crowd Movement and Egress Simulation*, Universität Duisburg-Essen, Dissertation, 2003. – URL http://duepublico.uni-duisburg-essen.de/servlets/DerivateServlet/Derivate-5477/Disskluepfel.pdf

[Laurien und Oertel 2009] Laurien, Eckart ; Oertel, Herbert: *Numerische Strömungsmechanik*. View, 2009. – ISBN 978-3-8348-0533-1

[Macal u. a. 2012] Macal, C.M. ; North, M.J. ; Collier, N. ; Dukic, V.M. ; Lauderdale, D.S. ; David, M.Z. ; Daum, R.S. ; Shumm, P. ; Evans, J.A. ; Wilder, J.R. ; Wegener, D.T.: Modeling the spread of community-associated MRSA. In: *Simulation Conference (WSC), Proceedings of the 2012 Winter*, 2012, S. 1–12. – ISSN 0891-7736

[Mertens u. a. 2006] Mertens, Koenraad ; Holvoet, Tom ; Berbers, Yolande: An Adaptive Distributed Layout for Multi-agent Applications. In: Garcia, Alessandro (Hrsg.) ; Choren, Rciardo (Hrsg.) ; Lucena, Carlos (Hrsg.) ; Giorgini, Paolo (Hrsg.) ; Holvoet, Tom (Hrsg.) ; Romanovsky, Alexander (Hrsg.): *Software Engineering for Multi-Agent Systems IV*. Springer-Verlag Berlin Heidelberg, 2006, S. 35 – 52. – ISBN 3-540-33580-3

[Metanet Software Inc 2011] Metanet Software Inc: 2011. – URL http://www.metanetsoftware.com/technique/tutorialA.html. – Zugriff: 06.02.2014

[MIT Courseware 2009] MIT Opencourseware (Veranst.): *Stability for ODE and von Neumann stability analysis*. 2009. – URL http://ocw.mit.edu/courses/mathematics/18-336-numerical-methods-for-partial-differential-equations-spring-2009/lecture-notes/MIT18_336S09_lec14.pdf. – Zugriff: 02.02.2014, http://ocw.mit.edu/terms/

[Müller 1996] Müller, Jörg P.: *The design of intelligent agents: a layered approach.* Vol. 1177. Springer-Verlag Berlin Heidel, 1996. – ISBN 3-540-62003-6

[Moujahed u. a. 2007] Moujahed, S. ; Gaud, N. ; Meignan, D.: A Self-Organizing and Holonic Model for Optimization in Multi-Level Location Problems. In: *Industrial Informatics, 2007 5th IEEE International Conference on* Bd. 2, 2007, S. 1053–1058. – ISSN 1935-4576

[Navarro u. a. 2013] Navarro, Laurent ; Corruble, Vincent ; Flacher, Fabien ; Zucker, Jean-Daniel: A flexible approach to multi-level agent-based simulation with the mesoscopic representation. In: *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems.* Richland, SC : International Foundation for Autonomous Agents and Multiagent Systems, 2013 (AAMAS '13), S. 159–166. – URL http://dl.acm.org/citation.cfm?id=2484920.2484948. – ISBN 978-1-4503-1993-5

[Navarro u. a. 2011] Navarro, Laurent ; Flacher, Fabien ; Corruble, Vincent: Dynamic level of detail for large scale agent-based urban simulations. In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2.* Richland, SC : International Foundation for Autonomous Agents and Multiagent Systems, 2011 (AAMAS '11), S. 701–708. – URL http://dl.acm.org/citation.cfm?id=2031678.2031717. – ISBN 0-9826571-6-1, 978-0-9826571-6-4

[Oracle 2014] Oracle: *Berkeley DB Java Edition Website.* 2014. – URL http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html. – Zugriff: 05.02.2014

[OSGi Alliance 2012] *Webseite OSGi Alliance.* 2012. – URL http://www.osgi.org/Main/HomePage. – Zugriff: 13.11.2012

[Park u. a. 2012] Park, Seung I. ; Quek, F. ; Cao, Yong: Modeling social groups in crowds using Common Ground Theory. In: *Simulation Conference (WSC), Proceedings of the 2012 Winter*, 2012, S. 1–12. – ISSN 0891-7736

[Povh 2011] Povh, Bogdan: *Anschauliche Physik für Naturwissenschaftler.* Springer-Verlag, 2011. – ISBN 978-3-642-17787-3

[Rao und Georgeff 1991] Rao, Anand S. ; Georgeff, Michael P.: Modeling rational agents within a BDI-architecture. In: *KR* 91 (1991), S. 473–484

[Rao u. a. 1995] Rao, Anand S. ; Georgeff, Michael P. u. a.: BDI Agents: From Theory to Practice. In: *ICMAS* Bd. 95, 1995, S. 312–319

[Reid u. a. 1987] Reid, Robert C. ; Prausnitz, John M. ; Poling, Bruce E.: *The Properties of Gases & Liquids*. Fourth Edition. McGraw-Hill, 1987. – ISBN 0-07-051799-1

[Rodriguez u. a. 2007] Rodriguez, Sebastian ; Gaud, Nicolas ; Hilaire, Vincent ; Galland, Stéphane ; Koukam, Abderrafiâa: An Analysis and Design Concept for Self-organization in Holonic Multi-agent Systems. In: Brueckner, SvenA. (Hrsg.) ; Hassas, Salima (Hrsg.) ; Jelasity, Márk (Hrsg.) ; Yamins, Daniel (Hrsg.): *Engineering Self-Organising Systems* Bd. 4335. Springer Berlin Heidelberg, 2007, S. 15–27. – URL http://dx.doi.org/10.1007/978-3-540-69868-5_2. – ISBN 978-3-540-69867-8

[Russel und Norvig 2010] Russel, Stuart ; Norvig, Peter: *Artificial Intelligence - A Modern Approach*. Third Edition. Pearson Education Inc., 2010. – ISBN 0-13-604259-7

[Schneider 2011] Schneider, Bernhard: *Die Simulation menschlichen Panikverhaltens*. Bd. 1. Auflage. Vieweg+Teubner Verlag, 2011. – ISBN 978-3-8348-1544-6

[Shankar 2012] Shankar, Suraj: *Matlab Code zur Simulation der 2D Diffusionsgleichung*. 2012. – URL http://www.mathworks.com/matlabcentral/fileexchange/38088-diffusion-in-1d-and-2d/content/Diffusion_2D.m. – Zugriff: 30.09.2013

[Socolofsky und Jirka 2005] Socolofsky, Scott A. ; Jirka, Gerhard H.: *Environmental Fluid Mechanics 1: Mixing and Transport Processes in the Environment*. 5th Edition. URL https://ceprofs.civil.tamu.edu/ssocolofsky/cven489/Downloads/Book/Socolofsky_Jirka.pdf, 2005. – Zugriff: 02.02.2014

[Spiegel Online 2003] Spiegel Online: *Panik in Chicago: 21 Disco-Besucher totgetrampelt*. 2003. – URL http://www.spiegel.de/panorama/panik-in-chicago-21-disco-besucher-totgetrampelt-a-236472.html. – Zugriff: 18.07.2013

[Steel u. a. 2010a] Steel, T. ; Kuiper, D. ; Wenkstern, R.Z.: Context-Aware Virtual Agents in Open Environments. In: *Autonomic and Autonomous Systems (ICAS), 2010 Sixth International Conference on*, march 2010, S. 90 –96

[Steel u. a. 2010b] Steel, T. ; Kuiper, D. ; Wenkstern, R.Z.: Virtual Agent Perception in Multi-agent Based Simulation Systems. In: *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on* Bd. 2, 31 2010-sept. 3 2010, S. 453 –456

[Steel und Wenkstern 2010] Steel, T. ; Wenkstern, R. Z.: Simulated event propagation in distributed, open environments. In: *Proceedings of the 2010 Spring Simulation Multiconference*. New York, NY, USA : ACM, 2010 (SpringSim '10), S. 17:1–17:8. – URL http://doi.acm.org/10.1145/1878537.1878555. – ISBN 978-1-4503-0069-8

[Tannenbaum und van Steen 2007] Tannenbaum, Andrew S. ; Steen, Maarten van: *Distributed Systems: Principles and Paradigms*. Second Edition. Pearson, 2007. – ISBN 0-13-613553-6

[Thiel 2013] Thiel, Christian: *Analyse von Partitionierungen und partieller Synchronisation in stark verteilten multiagentenbasierten Fußgängersimulationen*, HAW Hamburg, Diplomarbeit, April 2013. – URL http://opus.haw-hamburg.de/volltexte/2013/2059/pdf/MA_christian_thiel.pdf

[Twelkemeier 2012] Twelkemeier, Christian: *Konzept zur Behandlung von Ereignissen in Multi-Agenten Simulationen*. 2012. – Unveröffentlicht

[Twelkemeier 2013a] Twelkemeier, Christian: *Adaptive Ereignisbehandlung in MAS Simulationen*. 2013. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master12-13-seminar/twelkemeier/bericht.pdf. – Zugriff: 06.04.2014

[Twelkemeier 2013b] Twelkemeier, Christian: *Analyse von Detailstufen in ereignisorientierten Multi-Agenten Simulationen*. 2013

[Weidemann 1993] Weidemann, Ulrich: *Transporttechnik der Fussgänger*. Bd. zweite, ergänzte Auflage. Institut für Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau an der ETH-Zürich, März 1993

[Wendt 2009] Wendt, John F. (Hrsg.): *Computational Fluid Dynamics - An Introduction*. Spring, 2009. – ISBN 978-3-540-85055-7

[Weyns u. a. 2005] Weyns, Danny ; Van Dyke Parunak, H. ; Michel, Fabien ; Holvoet, Tom ; Ferber, Jacques: Environments for Multiagent Systems State-of-the-Art and Research Challenges. In: Weyns, Danny (Hrsg.) ; Van Dyke Parunak, H. (Hrsg.) ; Michel, Fabien (Hrsg.): *Environments for Multi-Agent Systems* Bd. 3374. Springer Berlin / Heidelberg, 2005, S. 1–47. – URL http://dx.doi.org/10.1007/978-3-540-32259-7_1. – 10.1007/978-3-540-32259-7_1. – ISBN 978-3-540-24575-9

[Wirth und Strauch 2012] Wirth, Ingo ; Strauch, Hansjürg: *Rechtsmedizin - Grundwissen für die Ermittlungspraxis*. 3. Auflage. Kriminalistik Verlag, 2012. – ISBN 978-3-7832-0021-8

[Wooldridge 2002] Wooldridge, Michael: *An Introduction to MultiAgent Systems*. John Wiley & Sons, Ltd., 2002. – ISBN 0-471-49691-X

[Zhou u. a. 2010] Zhou, Suiping ; Chen, Dan ; Cai, Wentong ; Luo, Linbo ; Low, Malcolm Yoke H. ; Tian, Feng ; Tay, Victor Su-Han ; Ong, Darren Wee S. ; Hamilton, Benjamin D.: Crowd modeling and simulation technologies. In: *ACM Trans. Model. Comput. Simul.* 20 (2010), November, S. 20:1–20:35. – URL http://doi.acm.org/10.1145/1842722.1842725. – ISSN 1049-3301

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 10. Februar 2014 Christian Twelkemeier