Bachelorarbeit

Malte Eckhoff

Ein Agenten-basiertes Gepardenmodell - Konzept und räumliche Wahrnehmung

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Malte Eckhoff

Ein Agenten-basiertes Gepardenmodell - Konzept und räumliche Wahrnehmung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

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

Eingereicht am: 25. September 2013

Malte Eckhoff

Thema der Arbeit
Ein Agenten-basiertes Gepardenmodell - Konzept und räumliche Wahrnehmung

Stichworte
Multi-Agenten-Simulation, Gepard

Kurzzusammenfassung
Dieses Arbeit behandelt die Entwicklung und Implementierung eines ersten Konzepts einer Multi-Agenten-Geparden-Simulation. Das Ergebnis dieser Arbeit ist ein erster Prototyp in Unity.
Malte Eckhoff

Title of the paper
An agent based cheetah model - Concept and spatial perception

Keywords
multi-agent-simulation, cheetah

Abstract
This document covers the development and implementation of a first concept for a multi-agent-cheetah-simulation. The result of this work is a Unity prototype.

Inhaltsverzeichnis

Listings

1 Einführung

1.1 Motivation

Geparden sind eine gefährdete Spezies. Laut Durant *et al.* (2013) gibt es zurzeit auf der Welt noch weniger als 10000 Tiere.

Der größte Feind dieser Tiere ist der Mensch. Geparden wurden von 76% ihrer historischen Lebensräume durch den Menschen vertrieben (Durant *et al.*, 2013). In Südafrika sterben die meisten Geparden nicht durch natürliche Ursachen, sondern werden von Farmern aus Angst um ihre Nutztiere getötet Houser (2008).
Neue Erkenntnisse über das Verhalten der Tiere könnten entscheidend dazu beitragen den Fortbestand der Geparden zu sichern. Könnte man beispielsweise den Farmern in einer anschaulichen Weise beweisen, dass die Geparden für ihre Nutztiere keine Gefahr darstellen, würden vermutlich weitaus weniger Tiere sinnlos erschossen werden.

Geparden in der freien Wildbahn zu beobachten hat sich jedoch als äußerst schwierig herausgestellt. Die derzeitigen Studien haben folgende Probleme:

1. Die Studien beziehen sich meist nur auf einzelne Habitate in denen Geparden leben.
2. Die Anzahl der betrachteten Geparden ist gering. Statistische Aussagen zu treffen ist also, aufgrund der geringen Menge der Daten, problematisch.
3. Die erhobenen Daten sind aufgrund technischer Begrenzungen sehr grob. Ein gutes Beispiel dafür sind GPS-Daten. Typischerweise steht nur eine Positionsangabe innerhalb mehrerer Stunden zur Verfügung. Eine Interpolation der Bewegungen des Tieres fällt also entsprechend ungenau aus.
4. Das Verhalten von Geparden ist nur dort bekannt wo sie auch gut beobachtet werden können. Es können momentan nur begrenzt Aussagen darüber getroffen werden, wie sich ein Gepard in anderen Umgebungen verhält.

5. Die Ergebnisse der Forschungen sind meist nicht sonderlich anschaulich.

Umfassende Informationen über das Verhalten von Geparden im allgemeinen sind also schwierig zu erheben. Eine Fusion der Informationen der Erkenntnisse der einzelnen Studien fehlt gänzlich.

Eine Simulation von Geparden, die realitätsnah genug ist um Prognosen über das Verhalten der Tiere zu treffen, würde diese Probleme lösen.

Das Ziel dieser Bachelorarbeit ist es den Grundstein für diese Simulation zu legen. In der finalen Version sollen von dem Jagdverhalten bis zur Aufzucht der Jungtiere alle Aspekte des Lebens und Verhaltens eines Geparden abgebildet werden.
Die Untersuchung emergenter Effekte großer Geparden-Populationen können mit einer Simulation einfacher und vollständiger, als durch Beobachtung, untersucht werden.
Anschaulich visualisierte Simulationen von Geparden Populationen könnten dazu dienen, das öffentliche Interesse für den Erhalt dieser Tiere, zu wecken.
Strategien zur Auswilderung in Gefangenschaft geborener Geparden könnten untersucht und optimiert werden, ohne echte Geparden in Gefahr zu bringen.

In dieser Arbeit werde ich mich hauptsächlich auf das Jagdverhalten der Tiere konzentrieren.

2 Analyse

Die folgende Analyse der Fachdomäne ist der erste Schritt auf dem Weg zu einer gültigen Modellierung.

Als erstes interessieren uns hauptsächlich die Eigenschaften und das Verhalten von Geparden. Andere, in der Simulation vorkommende, Tiere dienen den Geparden nur als Beutetiere. Das Verhalten und die Eigenschaften dieser Tiere sind also momentan nur von Interesse, solange sie die Funktion als Beutetier betreffen.

2.1 Über Geparden - Eine Zusammenfassung der für die Simulation relevanten Eigenschaften von Geparden

Um einen Gepard modellieren zu können müssen wir wissen, welche grundlegenden Eigenschaften dieser hat. Wie groß ist ein Gepard? Was für Habitate bevorzugen Geparden? Sind Geparden Tag oder Nachtaktiv? Wie jagen Geparden ihre Beutetiere?

Diese und weitere Fragen werden in diesem Kapitel geklärt.

Soweit nicht anders angegeben wurden die Informationen aus diesem Abschnitt aus folgender Quelle entnommen: Wikipedia (2013).

2.1.1 Physische Eigenschaften

Ein durchschnittlicher ausgewachsener Gepard hat eine Kopf-Rumpf-Länge von 150 cm und eine Schulterhöhe von 80 cm. Der Schwanz ist 70 cm lang.

Er hat ein Gewicht von 60 kg und kann ein Alter von 15 Jahren erreichen. Im vollen Lauf können Geparden eine Geschwindigkeit von 112 km/h erreichen, und diese über 400 Meter aufrecht erhalten.

2.1.2 Lebensraum

Geparden lebten einst in fast ganz Afrika mit Ausnahme der zentralafrikanischen Waldgebiete, Vorderasien, der indischen Halbinsel und in teilen Zentralasiens. Heute sind Geparden fast nur noch in Afrika südlich der Sahara anzutreffen.

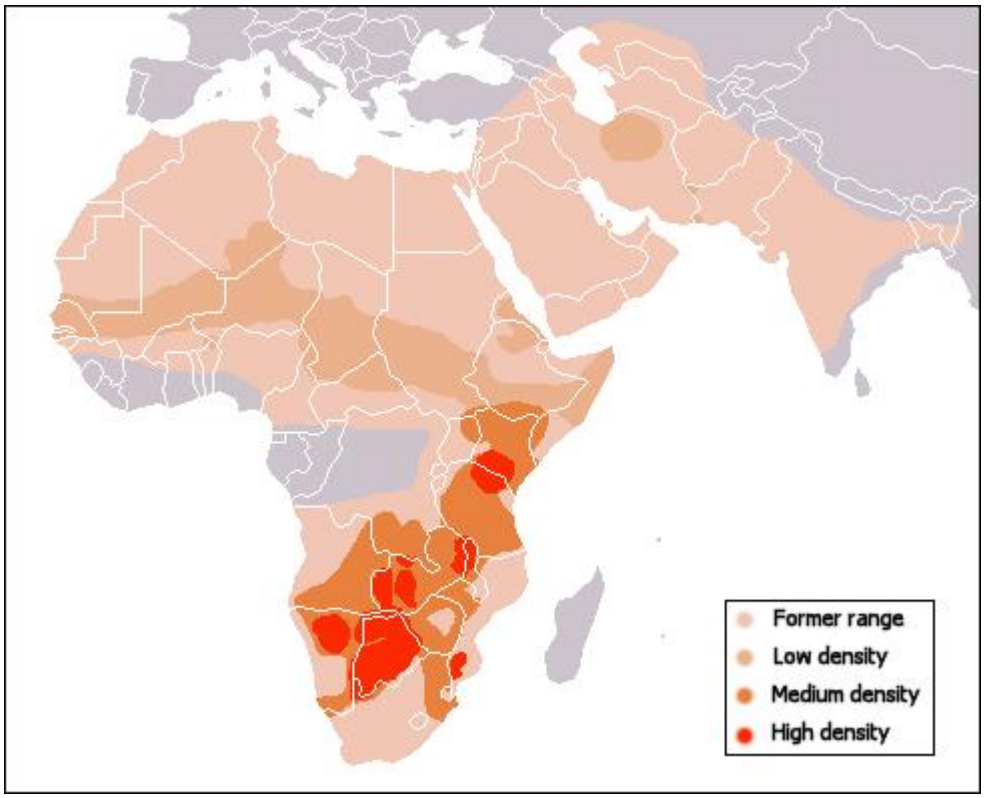


Abbildung 2.1: Übersicht der derzeitigen noch existierenden Lebensräume der Geparden. Wikipedia (2013)

Laut Wikipedia (2013) sind Geparden reine Savannen- und Steppentiere. Sie bevorzugen Bereiche mit hohem, Deckung bietendem Gras und Hügeln als Ausschaupunkten. Zu viele Bäume und Sträucher machen eine Landschaft für Geparde ungeeignet, da sie dort ihre Schnelligkeit nicht ausnutzen können. In Halbwüsten kommen Geparde dagegen gut zurecht, wenn sie genügend Beutetiere finden.

Nach Cheetah.org (2013) gibt es weltweit vermutlich noch insgesamt 10000 Tiere. Die größte lokale Population, mit unter 2500 Tieren, wird in Namibia vermutet.

2.1.3 Lebensweise

Geparden sind tagaktive Tiere. Innerhalb der Spezies unterscheidet sich die Lebensweise stark zwischen Geparden-Männchen und Weibchen. Geparden-Weibchen leben meistens allein mit Ausnahme der Zeit, in der sie Junge führen. Männchen hingegen formen Verbände, in denen sie (meistens Wurfbrüder) zu zweit oder dritt leben. Selten gibt es größere Gepardgruppen von bis zu 15 Individuen.

Zur Abgrenzung ihres Reviers nutzen Geparden-Männchen Urinmarkierungen. Männchen und Weibchen kommen nur zur Paarung zusammen und trennen sich gleich darauf wieder.

2.1.4 Bewegungsverhalten

Einzelne Geparden bevorzugen Gelände mit einer möglichst guten Deckung (beispielsweise bewaldete Gebiete). Eine erhöhte Chance Beutetiere anzutreffen, Deckung vor rivalisierenden Raubtieren (Löwen und Hyänen) und eine gute Deckung bei der Jagd sind die Vorteile dieses Habitats.

Broomhall & Lynne (2002) postuliert das eine erhöhte Deckung für die Geparden den Nachteil besitzt, dass diese ihre hohe Geschwindigkeit nicht ausnutzen können. Sie präferieren daher sich am Rand von bewaldeten Gebieten zu bewegen, um die Deckung des bewaldeten Gebietes nutzen zu können, aber gleichzeitig Beutetiere auf dem anliegenden offeneren Gelände jagen zu können.

Methode zur Berechnung des Aktionsraums	Aktionsraum einzelnes Geparden Männchen (km^2)	Aktionsraum Geparden Männchen Koalition (km^2)	Aktionsraum Geparden Weibchen (km^2)
95% MCP	316	598	409.9
95% Kernel	579(2)	849(1)	668,7(2)

Abbildung 2.2: Aktionsräume von Geparden. Houser (2008)

Zeit	Distanz in km Mittelwert $\pm$ Standard Abweichung	Min	Max
Geparden Männchen			
02:00 - 08:00	3.46 ± 0.26	0	16.18
08:00 - 14:00	1.14 ± 0.12	0	7.84
14:00 - 20:00	2.16 ± 0.15	0	8.46
20:00 - 02:00	2.44 ± 0.22	0	14.47
Geparden Weibchen			
01:00 - 13:00	0.88 ± 0.08	0	7.65
13:00 - 01:00	1.24 ± 0.08	0	6.63

Abbildung 2.3: Durchschnittlich zurückgelegte Strecken nach Tageszeit. Houser (2008)

Jahreszeit	Distanz in km Mittelwert $\pm$ Standard Abweichung	Min	Max
Regenzeit	2.23 ± 0.13	0	13.85
Trockenzeit	1.97 ± 0.10	0.03	7.63

Abbildung 2.4: Die durchschnittlich zurückgelegte Distanz von Geparden-Weibchen nach Jahreszeit. Houser (2008)

Geparden-Männchen

Geparden-Männchen haben einen Aktionsraum von 492 km^2 (für einzelne Tiere) bis zu 849 km^2 (für Gepardengruppen). Houser (2008) Sie achten dabei darauf, dass dieser eine möglichst hohe Zahl an Geparden-Weibchen enthält. Broomhall & Lynne (2002)

Geparden-Männchen legen am Tag im Durchschnitt eine Strecke von 6,13 Kilometern zurück (0 - 39 Km pro Tag) Houser (2008).

Wenn ein Männchen ein Territorium besitzt wird es dieses nur verlassen, wenn ihn eine zu geringe Zahl an Beutetieren dazu zwingt. Er wird normalerweise nicht den jährlichen Wanderungen seiner Beutetiere folgen.

Ein Männchen ohne Territorium hingegen wird genau dies tun, bis er ein eigenes Territorium gefunden hat. Durant *et al.* (1988)

Geparden, die sich zu einer Gruppe zusammen geschlossen haben, scheinen eine höhere Präferenz für offenes Gelände zu haben. Nach Broomhall & Lynne (2002) bevorzugen die Geparden dieses Gelände, weil Konkurrenten, wie beispielsweise Löwen oder Hyänen, besser abgewehrt werden können. In der Gruppe können größere Beutetiere gejagt werden, die eher im offenen Gelände anzutreffen sind. Außerdem können andere Geparden-Männchen beim eindringen in das Territorium der Gruppe besser ausgemacht werden.

Geparden-Weibchen

Geparden-Weibchen haben einen Aktionsraum von 241 km^2 bis zu 306 km^2 Houser (2008) Geparden-Weibchen wählen ihren Aktionsraum so, dass dieser eine möglichst hohe Zahl an Beutetieren enthält Houser (2008).

Sie legen pro Stunde im Durchschnitt eine Strecke von 2,16 +/- 0.07 Kilometern zurück (0 - 20 Km pro Tag) .

Geparden-Weibchen folgen den jährlichen Bewegungen ihrer Haupt-Beutetiere (beispielsweise Thomson-Gazellen). Bewegen sich die Beutetiere jedoch in Gebiete, die zu wenig Deckung bieten, werden die Geparden-Weibchen ihnen nicht oder nur in geringer Zahl folgen. Die Geparden-Weibchen ziehen in diesem Fall Gebiete vor, die weniger Beutetiere aber mehr Deckung bieten. Durant *et al.* (1988)

2.1.5 Jagdverhalten

Geparden jagen am frühen Morgen und späten Nachmittag. Cooper *et al.* (2007) Zuerst pirschen sie sich auf mindestens 50 bis 100 m an ihr Beutetier heran, um dann mit hoher Geschwindigkeit anzugreifen (dieser Vorgang wird als Hetzjagd bezeichnet). Wenn ein Gepard das anvisierte Beutetier nicht nach einigen Hundert Metern Sprint erreicht hat, muss der Gepard aufgeben.

Beim erreichen seiner Beute läuft der Gepard in die Beine seines Opfers und bringt es damit zu Fall. Anschließend erstickt er es indem er die Kehle des Tieres mit den Zähnen zudrückt. Danach muss sich der Gepard erst einmal ausruhen, da eine Jagd zu einer Überhitzung seiner Muskeln führt. Nun muss der Gepard schnell fressen, da er seine erlegte Beute nicht gegen andere Raubtiere wie Hyänen oder Leoparden verteidigen kann.

Geparden werden durch die Präsenz von Löwen und oder Hyänen in ihrem Jagdverhalten gestört. Sie werden diesen aktiv ausweichen und der Jagderfolg wird abnehmen, wenn sie der Meinung sind, dass eine oder mehrere dieser Tiere sich in der Nähe befinden. Durant (1998)

Nach Cooper *et al.* (2007) jagt ein Geparden Weibchen mit Jungen öfter als andere Geparden, um ihre Jungen zu versorgen. Die durchschnittliche Erfolgsquote bei Jagdversuchen liegt bei 70%. Wikipedia (2013)

2.1.6 Beutetiere

Normalerweise bevorzugen die Geparden Beutetiere unter 60 kg Körpergewicht. Meistens sind das kleinere Huftiere wie Gazellen und Böckchen. So ernähren sich beispielsweise in Ostafrika lebende Geparden fast ausschließlich von Thomson-Gazellen, Grant Gazellen und Impalas.

Größere Tiere, wie ausgewachsene Zebras oder Gnus, sind für Geparden keine geeigneten Beutetiere, da er diesen im Kampf unterlegen ist. Jungtiere werden jedoch gelegentlich von in der Gruppe jagenden Geparden erlegt. In Notzeiten jagt ein Gepard auch Hasen, Kaninchen und Vögel.

2.2 Entscheidung zur Jagd

Ob sich ein Gepard entscheidet zu jagen kann nach Durant (1998) als Funktion weniger Einflussgrößen modelliert werden. Entscheidend ist, ob der Gepard Junge hat, ob es einen Überfluss and Thomson-Gazellen gibt und wie viele Löwen im gleichen Gebiet operieren. Andere Faktoren, wie z.B. die Anzahl an Hyänen im Gebiet oder wieviel Hunger der Gepard hat, scheinen keinen Einfluss auf die Entscheidung für oder gegen eine Jagd zu haben.

2.3 Jagderfolg

Hilborn *et al.* (2012) modelliert den zu erwartenden Jagderfolg als eine Funktion folgender Größen:

1. Das Alter des Gepards. - Erwachsen (Alter > 44 Monate), Jungtier (Alter 14 - 44 Monate)
2. Der Bauchumfang des Gepards als Indikator für dessen Sättigung. - Skala von 1 bis 14, wobei bei 1 der Gepard am verhungern ist und bei 14 der Bauch des Tieres extrem voll ist.
3. Die Größe des Beutetiers. - Die Größe des Beutetiers: Beutetiere werden in drei Größenkategorien nach deren Gewicht eingeteilt:
   a) Klein (<10kg): Hasen (Lepus), Thomson Gazellen Kitze, Grants Gazellen Kitze
   b) Mittel (10-30kg): Thomson Gazellen, junge Grant-Gazellen, Gnu Kälber, Zebra Fohlen, Kuhantilopen und Steinböckchen
   c) Groß (>30kg): Grant-Gazellen, Gnu-Jungtiere, ausgewachsene Kuhantilopen, ausgewachsene Impalas, ausgewachsene Warzenschweine und der Riedbock (Redunca redunca)
4. Die Distanz in Metern zum nächsten Kopje.
5. Die Distanz in Metern zum nächsten Fluss.
6. Die Art des Habitats in dem die Jagd stattfindet. - Akazien Waldgebiet/Grassland
7. Das Vorhandensein von Gepardenjungen.
8. Die Jahreszeit. - Feucht/Trocken

Dieser funktionale Zusammenhang vereinfacht die Modellierung des Jagderfolges, in der Simulation, erheblich. Die meisten Einflussgrößen die Hilborn *et al.* (2012) beschreibt sind in der Simulation einfach quantifizierbar. Für das Habitat in dem die Jagd stattfindet gilt dies nicht. Das Habitat kann nur durch Gelände Kategorien abstrakt beschrieben werden.

2.4 Fachliches Datenmodell

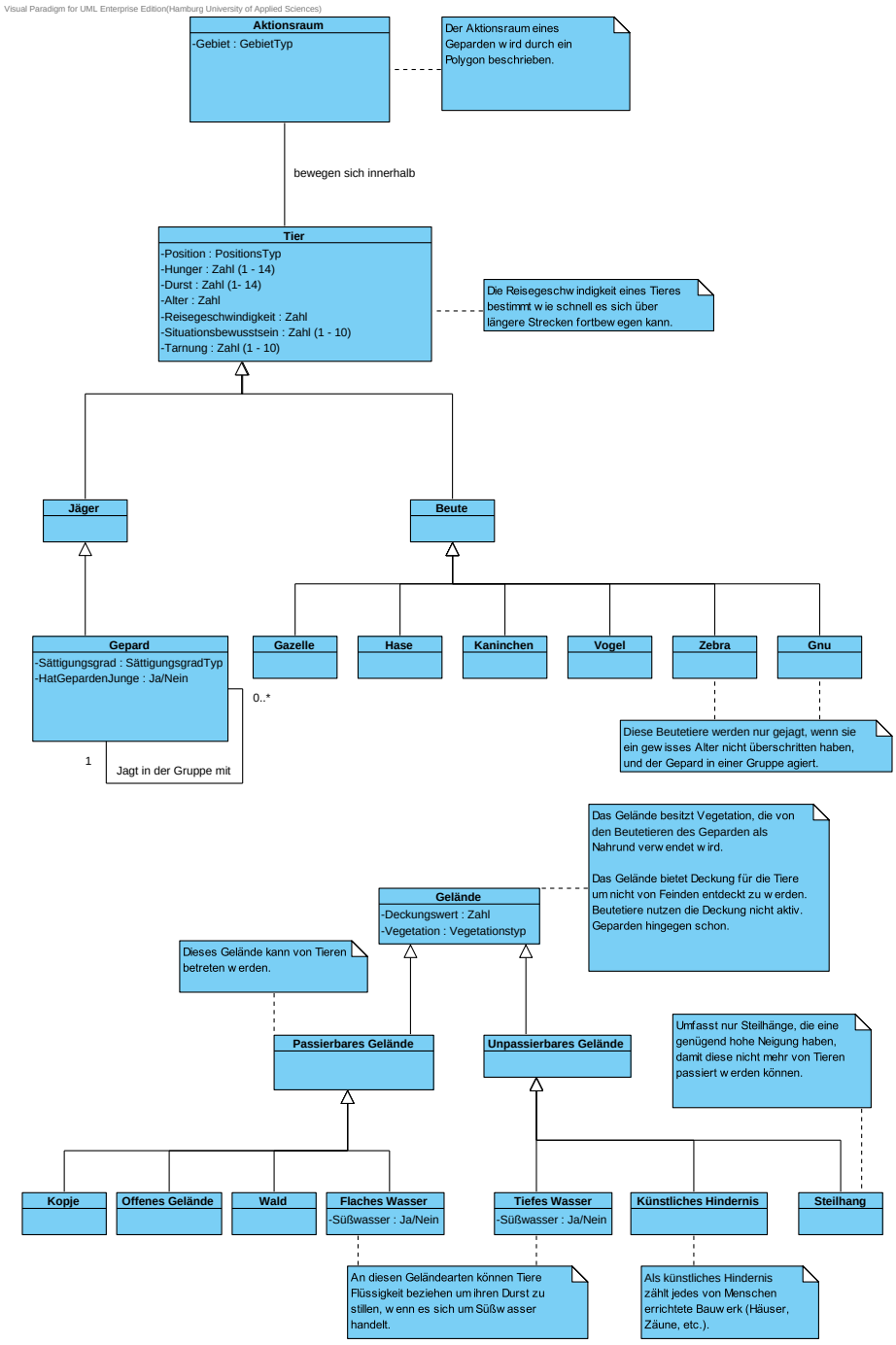


Abbildung 2.5: Fachliches Datenmodell der Domäne.

2.5 Funktionale Anforderungen

1. Das System simuliert Geparden und ihre Beutetiere in einem vorher designierten Gebiet (z.B. ein Teilgebiet eines Nationalparks).
2. Das System simuliert die Bewegung und die Jagd der einzelnen Geparden und Geparden-Gruppen auf Beutetiere in dem designierten Gebiet.
3. Das System simuliert für Geparden und Beutetiere folgende Bedürfnisse:
   a) Hunger
   b) Durst
4. Das System simuliert die Vegetation als Nahrungsquelle für Beutetiere im designierten Gebiet.
5. Das System simuliert Wasser als Flüssigkeitsquelle im designierten Gebiet.
6. Das System simuliert die Auswirkungen des Geländes auf die Reisegeschwindigkeit von Geparden und Beutetieren.
7. Das Gelände auf dem eine Jagd stattfindet beeinflusst den Jagderfolg eines Geparden.
8. Das System simuliert die gegenseitige Wahrnehmung der Beutetiere und Geparden. Beispielsweise werden Beutetiere einem Geparden nur ausweichen, wenn sie ihn erst wahrgenommen haben. Ebenso kann ein Gepard nur Beutetiere verfolgen und jagen, die er zuerst wahrgenommen hat.
9. Das System simuliert folgende Entscheidungsprozesse der Geparden:
   a) Der Gepard entscheidet sich, wohin er sich in dem designierten Gebiet bewegen will.
   b) Der Gepard priorisiert seine Bedürfnisse (Hunger und Durst).
   c) Der Gepard entscheidet durch welche Aktion er seine Bedürfnisse (Hunger und Durst) am besten befriedigen kann.
   d) Der Gepard entscheidet sich, ob er ein Beutetier, dass er wahrgenommen hat, jagen möchte oder nicht.
10. Das System simuliert folgende Entscheidungsprozesse der Geparden:

   a) Das Beutetier entscheidet sich, wohin es sich in dem designierten Gebiet bewegen möchte.
   b) Das Beutetier priorisiert seine Bedürfnisse (Hunger und Durst).
   c) Das Beutetier entscheidet durch welche Aktion es seine Bedürfnisse (Hunger und Durst) am besten befriedigen kann.

11. Geparden sterben, wenn sie entweder verhungern oder verdursten.
12. Beutetiere sterben, wenn sie entweder verhungern, verdursten oder von einem Geparden erlegt werden.
13. Der Benutzer konfiguriert folgende Parameter des Systems:
   a) Position der Geparden im Gebiet
   b) Position der Beutetiere im Gebiet
   c) Anfangszustand und Verhaltensparameter der einzelnen Geparden
   d) Anfangszustand und Verhaltensparameter der einzelnen Beutetiere
14. Der Benutzer konfiguriert für Geparden folgende Parameter:
   a) Alter des Geparden in Monaten
   b) Initialier Sättigungsgrad des Tieres (1-14 wobei 1 nahe dem verhungern und 14 vollgefressen ist)
   c) Hat der Gepard Junge die er versorgen muss?
   d) Zugehörigkeit zu einer Gruppe von anderen Geparden
   e) Reisegeschwindigkeit eines Geparden.
15. Der Benutzer konfiguriert für Beutetiere folgende Parameter:
   a) Spezies des Beutetiers
   b) Alter des Beutetiers
   c) Zugehörigkeit zu einer Gruppe von anderen Beutetieren
   d) Reisegeschwindigkeit der Beutespezies.
16. Durch die Manipulation der vom System bereitgestellten Parameter durch den Benutzer wird der Jagderfolg oder Misserfolg der Geparden bestimmt.

17. Der Benutzer startet, stoppt oder resetted die Simulation zu einem beliebigen Zeitpunkt.
18. Das System zeigt den Verlauf der Simulation in Echtzeit an.
19. Das System zeigt eine Liste aller erfolgten Jagden die während eines Simulationsdurchlaufes stattgefunden haben.
20. Geparden jagen koordiniert in der Gruppe oder einzeln.

2.6 Nicht funktionale Anforderungen

1. Die Simulation der Geparden und deren Beutetiere erfolgt in Echtzeit (d.h. kein Playback von vorausberechneten Daten).

2.7 Ausschlüsse - Was soll das System explizit nicht leisten?

1. Während die Simulation läuft kann der Benutzer die Parameter der Geparden und deren Beutetiere nicht ändern.
2. Außer dem Jagdverhalten des Geparden werden keine anderen Aspekte des Lebens oder des Verhaltens simuliert.
3. Beutetiere und Geparden können sich nicht fortpflanzen. Gepardenjungen gehen nur als Parameter für das Jagdverhalten in die Simulation ein. Sie werden nicht explizit simuliert.
4. Die Akteure des Modells sind ausschließlich die Geparden und ihre Beutetiere, andere Tiere werden nicht simuliert.
5. Während die Simulation berechnet wird findet nur eine zweckmäßige Visualisierung statt. Die Visualisierung soll bewusst einfach gehalten werden, um mehr Rechenleistung für die Simulation zur Verfügung zu haben.

2.8 Erweiterungen - Wie könnte das System in zukünftigen Iterationen erweitert werden?

1. Die Simulationsdurchläufe können aufgezeichnet und in einer grafisch aufwendigen und ansprechenden Form wiedergegeben werden. Denkbar ist z.B. das aufbereiten der Daten in einer 3D-Engine.

2. Das System kann von nicht Informatikern benutzt und konfiguriert werden. Das bedeutet, alle Anforderungen die den Benutzer als Akteur betreffen müssen so umgesetzt werden, dass auch nicht technik-affine Benutzer diese bedienen können.

3. Das System ist auf mehrere im Netzwerk/Internet/Cloud verbundene Computer verteilt.

2.9 Verfügbare Frameworks

Es gibt sehr viele Frameworks für Multi-Agenten-Simulationen. Die meist verwendeten sind folgende:

1. NetLogo - http://ccl.northwestern.edu/netlogo/

2. Repast - http://repast.sourceforge.net/

3. GAMA - https://code.google.com/p/gama-platform/

Das entscheidende Kriterium für die Auswahl eines Frameworks ist in diesem Fall Performance. Die Ausführung der Simulation, mit einer sehr großen Anzahl an Agenten, soll von dem Benutzer "live"beobachtet werden können.

Des weiteren sollen eine große Anzahl an unterschiedlichen Szenarien, zur Erforschung des Verhaltens der Geparden, erstellt werden können.

Eine hochwertige Visualisierung und der Export der Simulationsergebnisse zur weiteren Auswertung werden ebenfalls, in der Zukunft, eine große Rolle spielen. Flexibilität und Erweiterbarkeit sind also zwei weitere Kernanforderungen.

Damit ein Framework für die Simulation nutzbar ist muss es also folgende Kriterien erfüllen:

1. Das System simuliert eine mehrere Tausend Agenten mit mindestens 30 Simulationsupdates pro Sekunde.

2. Das System nutzt die vorhandene Hardware effizient aus und skaliert möglichst gut mit mehr Hardwareleistung.

3. Das System ist einfach erweiterbar um es an neue Szenarien anzupassen.

4. Das System stellt einfach anpassbare Schnittstellen für Nachbarsysteme zur Verfügung.

5. Das System greift bei der Generierung des Szenarios auf beliebige Datenquellen zu. Die Adapter für neue Datenquellen werden durch Plugins geliefert.

Das Problem der bereits vorhandenen Frameworks ist, dass keines eine ausreichend hohe Anzahl an komplexen Agenten simulieren kann. GAMA bietet als einziges Framework die Möglichkeit einen Simulationsdurchlauf zu verteilen. Dabei werden jedoch nur mehrere Simulationen parallel ausgeführt. Die einzelne Simulation wird dadurch nicht schneller. Die Anzahl der Agenten in einer Simulation ist also auch in diesem Fall immer noch nicht ausreichend.

Eines der vorhandenen Frameworks zu verwenden ist also keine Option. Ein an der HAW in der Entwicklung befindendes Framework ist da schon interessanter. WALK setzt den Fokus auf die effiziente Ausnutzung der Hardware. Damit ist eine sehr hohe Anzahl an Agenten möglich.

2.9.1 WALK

Vorteile:

1. Sehr gute Vierteilbarkeit!
2. Quellcode steht zur Verfügung.
3. Hoch modular aufgebaut. Alle Komponenten können beliebig ausgetauscht werden.
4. Plattform Unabhängig.

Nachteile:

1. Noch keine DSL.
2. Kaum Tools vorhanden.
3. Teilweise erst prototypisch implementiert.
4. Noch kein Support für GIS-Daten.
5. Befindet sich noch in der Entwicklung.

WALK ist der beste Kandidat für die Umsetzung der Simulation weil es das Hauptproblem der hohen Anzahl an Agenten löst. Der frühe Entwicklungsstand bringt in diesem Fall sogar Vorteile.

Die Entwicklung der Simulation erfordert mit jedem Framework einen hohen Arbeitsaufwand für Anpassungen. Der frühe Entwicklungsstand macht es tendenziell einfacher das Framework den Anforderungen anzupassen.

Aus diesen Gründen werde ich WALK für die Entwicklung der Simulation im Master verwenden.

3 Konzept

3.1 Modellierung der Umwelt

Die Umwelt in der sich die Agenten bewegen bildet ein vom Benutzer gewähltes spatial begrenztes fiktives oder echtes Gebiet. Die Simulation der Umwelt wird auf die für die Simulation der Geparden relevanten Eigenschaften begrenzt.

Es wird davon ausgegangen, dass das Gebiet durch natürliche oder künstliche Barrieren begrenzt ist, die von Tieren nicht überwunden werden können.

Die Veränderungen des Geländes die uns interessieren sind relativ großflächiger Natur. In der Simulation wird beispielsweise nicht jeder einzelne Baum simuliert. Veränderungen des Szenarios die durch Tiere verursacht werden sind also zu gering um für die Simulation von Interesse zu sein. Das bedeutet also, dass Agenten keine Möglichkeit haben ihre Umwelt zu verändern.

Die Lebenszyklen der Simulation der Vegetation im Gebiet werden nur sehr rudimentär simuliert. Für die Beutetiere ist es nur wichtig zu wissen, in welchem Gebiet zu welcher Jahreszeit wie viel Nahrung zu finden ist. Geparden nutzen die Vegetation nur als Deckung bei der Jagd.

Die Simulation der Jahreszeiten fällt ähnlich simpel aus. Hauptsächlich ist der Einfluss der Jahreszeiten auf die Vegetation und die verfügbaren Süßwasserquellen interessant.

Welche Jahreszeiten genau unterschieden werden ist von der geographischen Position des Gebiets abhängig. In Afrika wird beispielsweise nur zwischen der Trocken- und Regenzeit unterschieden.

Die Umwelt wird für die Simulation in Quadranten aufgeteilt. Ein Quadrant ist für die Berechnung der Simulation atomar. Damit kann die Auflösung, in der die Simulation berechnet wird, fast beliebig skaliert werden. Die Größe eines Quadranten wird nach unten durch die Größe des größten vorkommenden Tieres beschränkt. Es ist ebenfalls möglich das gesamte Gebiet mit genau einem Quadranten abzudecken, was aber natürlich nicht zielführend ist.

Die Größe eines Quadranten sollte von der verfügbaren Rechenleistung abhängig gemacht werden. Wird ein relativ kleines Gebiet abgebildet, oder steht viel Rechenleistung zur Ver-

fügung, kann die Größe eines Quadranten verhältnismäßig klein gewählt werden. Kleinere Quadranten resultieren in einer größeren Auflösung und damit in einer genaueren Simulation. Eine möglichst kleine Quadranten Größe ist also wünschenswert.

3.1.1 Zeit

Die Zeit wird in der Simulation als diskrete Zeitschritte behandelt, sogenannte Ticks. Ein Tick ist dabei für die Simulation atomar. In einem Tick führen alle Agenten in einer festgelegten Reihenfolge ihre Aktionen aus.

Zur Veranschaulichung kann man die Simulation mit einem Brettspiel vergleichen. Ein Tick entspricht dabei einer Runde. Die Agenten sind die Spieler. Während jeder Runde (Tick) führen also alle Spieler (Agenten) ihrer Aktionen aus.

Die Zeit (in Tagen, Stunden, Sekunden usw.) die während eines Ticks in der Simulation vergeht kann konfiguriert werden.

3.1.2 Gelände

Quadranten können passierbar oder unpassierbar sein. Passierbare Quadranten können von Agenten betreten werden, unpassierbare nicht. Ein typisches Beispiel für passierbares Gelände ist ein Stück Wald oder Savanne. Ein typisches unpassierbares Gelände ist ein tiefes Gewässer, oder ein von Menschen erbautes Hindernis (z.B. ein Zaun oder ein Haus).

Höhenunterschiede im Gelände werden nur dann modelliert, wenn diese von einem Tier nicht überwunden werden können. Ein Steilhang (http://de.wikipedia.org/wiki/Steilhang) muss beispielsweise eine gewisse Steigung und einen gewissen Höhenunterschied aufweisen um als unpassierbar eingestuft zu werden.

Die Anzahl der Agenten die sich in einem Quadranten gleichzeitig aufhalten können wird durch dessen Größe bestimmt.

3.2 Modellierung der Agenten

Alle Agenten (in diesem Fall Geparden und Beutetiere) teilen sich bestimmte Eigenschaften, Fähigkeiten und Verhaltensweisen. Diese können für alle Agenten modelliert werden. Die jeweilige Agentenspezies erweitert dann diese Attribute in einer für die Spezies geeigneten Weise.

3.2.1 Bewegung

Große Distanzen werden von den Agenten in ihrer jeweiligen Reisegeschwindigkeit zurückgelegt. Diese Reisegeschwindigkeit ist die Geschwindigkeit die der Agent über größere Strecken (mehrere Kilometer) aufrecht erhalten kann.

Agenten können sich ebenfalls entschließen zu sprinten. Diese Geschwindigkeit ist bedeutend höher als die Reisegeschwindigkeit, kann aber nur für kurze Strecken aufrecht erhalten werden.

Die Geschwindigkeit wird in Metern pro Tick gemessen. Um einen Quadranten zu durchqueren benötigt ein Agent t Ticks $t = \sqrt{k}/r$ wobei r die Reisegeschwindigkeit des Tieres und k die Kantenlänge eines Quadranten ist. Wenn ein Agent eine genügend hohe Geschwindigkeit besitzt kann er in einem Tick mehrere Quadranten durchqueren.

3.2.2 Wahrnehmung

Jeder Agent nimmt seine Umwelt und andere Agenten (mit einer bestimmten Wahrscheinlichkeit) in einem bestimmten Radius um sich herum wahr. Dies ist der Wahrnehmungsbereich des Agenten. Der Wahrnehmungsbereich wird wie folgt berechnet: $w = s/k$ wobei w der Wahrnehmunsbereich, s das Situationsbewustsein eines Agenten und k die Kantenlänge eines Quadranten ist.

Ein Agent A entdeckt einen anderen Agenten B in seiner Wahrnehmungsbereich nur mit einer gewissen Wahrscheinlichkeit x. Sofern ein Quadrant groß genug ist können sich beide auch in einem Quadranten befinden.

Ob ein Tier ein anderes wahrnimmt hängt von vielen Faktoren ab. Folgend werde ich diese aufzählen und deren Relevanz erläutern. Die hier angenommenen Beeinflussungen der Wahrnehmungswahrscheinlichkeit gehen vom Normalfall aus. Es mag beispielsweise sein, dass es Tiere gibt, deren Wahrnehmung mit steigender Entfernung immer besser wird. Das System unterstützt eine entsprechende Parametrisierung in diesen speziellen Fällen.

Das **Situationsbewusstsein** ist ein abstrakter Wert dafür wie gut ein Agent seine Umgebung wahrnimmt. Dieser Wert ist also eine abstrakte Repräsentation der Leistungsfähigkeit der Sinnesorgane des jeweiligen Tieres.

Er bestimmt also ob und auf welche Reichweite ein Agent andere Agenten wahrnimmt. Dies wird besonders interessant, wenn Agenten aktiv versuchen eine Entdeckung zu vermeiden.

Die **Tarnung** eines Agenten repräsentiert die Qualität der passiven Schutzmechanismen eines Tieres um nicht wahrgenommen zu werden. Die in dieser Simulation vorkommenden Tiere erreichen diesen Effekt durch spezielle Tarnmuster ihrer Felle.
Selbst ein Tier das sich nicht aktiv versteckt mag durch diese Mechanik einer Entdeckung entgehen.

Das **Alter** der involvierten Agenten spielt eine entscheidende Rolle. Im Normalfall werden ältere Tiere mehr Erfahrung haben andere Tiere zu entdecken oder sich zu verstecken. Größere Erfahrung erhöht die Chance ein anderes Tier zu entdecken oder sich erfolgreich zu verstecken deutlich.

Der **Deckungswert** eines Quadranten sagt aus, wie stark die lokalen Umstände eine Entdeckung negativ beeinflussen. Ein hoher Deckungswert kann beispielsweise durch ein hügeliges unübersichtliches Gelände bedingt sein. Ein bewaldetes Gebiet würde ebenso wie eine urbane Umgebung einen hohen Deckungswert liefern.

Mit einer höheren **Entfernung** nimmt die Qualität der Wahrnehmung von Objekten ab. Die Chance für eine erfolgreiche Wahrnehmung eines Agenten wird also mit dessen Entfernung sinken. Da die Entfernungen in Quadranten gemessen werden ist also auch die **Kantenlänge** eines Quadranten entscheidend.

Folgend werde ich eine Funktion zur Berechnung der Wahrnehmungswahrscheinlichkeit beschreiben. Eine Beschreibung der Variablen gibt es darunter.

$detectionSkill_A = situationalAwareness_A * f_0(situationalAwareness_A) + age_A * f_1(age_A)$

$hidingSkill_B = stealth_B * f_2(stealth_B) + hiding_B * (age_B * f_3(age_B))$

$distanceModifier_{AB} = (distance_{AB} * edgeLength) * f_5(distance_{AB} * edgeLength)$

$coverModifier_B = cover_B * f_4(cover_B)$

$$positiveEvents = detectionSkill_A$$

$$negativeEvents = hidingSkill_B + coverModifier_B + distanceModifier_{AB}$$

$$\boxed{Chance(agentDetected) = \frac{positiveEvents}{positiveEvents+negativeEvents}}$$

Liegen optimale Bedingungen für die Entdeckung von Agent B vor ($negativeEvents = 0$) wird Agent B mit einer 100% Wahrscheinlichkeit entdeckt werden.

Variable	Beschreibung
$agentDetected$	Das Ereignis, dass Agent B von Agent A entdeckt wird.
$Chance(agentDetected)$	Die Wahrscheinlichkeit mit der Agent A Agent B entdeckt.
$situationalAwareness_A$	Das Situationsbewusstsein von Agent A wobei gilt: $situationalAwareness_A \in \mathbb{N} \wedge situationalAwareness_A \geq 1 \wedge situationalAwareness_A \leq 10$.
$stealth_B$	Die Tarnung von Agent B wobei gilt: $stealth_B \in \mathbb{N} \wedge stealth_B \geq 1 \wedge stealth_B \leq 10$.
age_A	Das Alter von Agent A in Monaten.
age_B	Das Alter von Agent B in Monaten.

$hiding_B$	Ein Parameter dafür ob Agent B versucht die vorhandene Deckung aktiv zu nutzen. Es gilt: $hiding_B = 1 \Rightarrow$ Der Agent nutzt aktiv die vorhandene Deckung. $hiding_B = 0 \Rightarrow$ Der Agent nutzt die vorhandene Deckung nicht.
$cover_B$	Der Deckungswert des Quadranten in dem sich Agent B befindet wobei gilt: $cover_B \in \mathbb{N} \wedge cover_B \geq 1 \wedge cover_B \leq 10$.
$distance_{AB}$	Die Entfernung in Quadranten zwischen Agent A und Agent B.
$edgeLength$	Die Kantenlänge eines Quadranten.
$positiveEvents$	Die Summe aller Ereignisse die zu einer Entdeckung von Agent B durch Agent A führen.
$negativeEvents$	Die Summe aller Ereignisse in denen Agent B durch Agent A nicht entdeckt wird.

$f_0(x_0)...f_n(x_n)$	Koeffizienten zur Gewichtung der einzelnen Parameter. Diese Koeffizienten werden aus einer Polynomfunktion berechnet, die in Abhängigkeit zum jeweiligen Parameter ($x_0 = s_A, x_1 = t_B$ usw.) steht. Im einfachsten Fall verhält sich diese Funktion linear. Aber es sind auch komplexere Verhaltensweisen möglich. Beispielsweise könnte ein Gepard bis zu einem gewiss hohen Alter immer effizienter darin werden Beute aufzuspüren, bis, ab einem gewissen Alter, diese Fähigkeit wieder rapide abnimmt. Für alle Koeffizienten gilt: $f_n(x_n) \in \mathbb{R} \wedge f_n(x_n) >= 0$

Abbildung 3.1: Beschreibung der Variablen, die in der Wahrnehmungsberechnung verwendet werden.

3.2.3 Hunger und Durst

Ein Tier muss für alle Aktivitäten Energie aufwenden. Diese Ausgaben müssen durch Nahrungs- und Flüssigkeits-Aufnahme gedeckt werden.

Die Zunahme von Hunger und Durst über die Zeit wird von einer, vom Benutzer anzugebende, Polynom-Funktion bestimmt. Körperliche Anstrengungen, wie Jagd oder Flucht, beschleunigen die Zunahme von Hunger und Durst.

Berechnung des Hungers

Für die Berechnung des Hungers benötigen wir eine Funktion die den Hungerwert eines Tieres in Abhängigkeit zu den folgenden Faktoren ausgibt.

Der **Hungerwert in diesem Tick** ist vom **Hungerwert im vorherigen Tick** abhängig. Der Hunger eines Tieres nimmt jeden Tick um einen bestimmten **Basiswert** zu. Diese Zunahme wird durch den Energieaufwand zur Aufrechterhaltung der Lebensfunktionen bedingt.

Körperliche Anstrengungen wie die Zurücklegung großer Strecken oder die Jagd bringen einen signifikanten Energieaufwand mit sich.

Zusätzlich mag es sein, dass durch **ungewöhnliche Umstände** der Energieaufwand in diesem Tick noch weiter erhöht wird. Das Tier könnte beispielsweise durch eine Krankheit geschwächt werden.

$$\boxed{h_A(t_n) = h_A(t_{n-1}) + basicHi_A(h_A(t_{n-1})) + stressHi_A(t) + exceptionalHi_A(t) - hd_A(t)}$$

Variable	Beschreibung
$h_A(t_n)$	Hunger des Tieres A zum Zeitpunkt t_n wobei gilt: $h_A \in \mathbb{N} \wedge h_A \geq 1 \wedge h_A \leq 14 \wedge t \geq 0$
$h_A(t_{n-1})$	Der Hunger des Tieres A zum Zeitpunkt des letzten Ticks (t_1).
$basicHi_A(h_A(t_{n-1}))$	Eine Polynomfunktion die den Zuwachs an Hunger, in Abhängigkeit zum letzten Hungerwert, für den derzeitigen Tick bestimmt. Hi steht hierbei für "hunger increase". Über diese Funktion kann die Basis-Zunahme des Hungers nach belieben durch den Benutzer parametrisiert werden.
$stressHi_A(t)$	Funktion die die Zunahme des Hungers durch körperliche Anstrengungen in diesem Tick zurückliefert. Hi steht hierbei für "hunger increase".
$exceptionalHi_A(t)$	Funktion durch die ungewöhnliche Umstände in diesem Tick dargestellt werden können, durch die sich der Hungerwert erhöht.

$hd_A(t)$	Funktion die die Abnahme des Hungers durch Aufnahme von Nahrung in diesem Tick zurückliefert. Die Variable hd steht hierbei für "hunger decrease". Beutetiere erhalten Nahrung durch das fressen der Vegetation. Geparden erhalten Nahrung durch das Erlegen und Fressen von Beutetieren.

Abbildung 3.2: Beschreibung der Variablen, die in der Berechnung des Hungers eines Tieres verwendet werden.

Berechnung des Durstes

Der Durst eines Tieres verhält sich analog zum Hunger. Alle Aktivitäten des Tieres führen zu einem Flüssigkeitsverlust der ausgeglichen werden muss.

$$\boxed{thirst_A(t_n) = thirst_A(t_{n-1}) + basicTi_A(d_A(t_{n-1})) + stresTi_A(t) + exceptionalTi_A(t) - td_A(t)}$$

Variable	Beschreibung
$thirst_A(t_n)$	Durst des Tieres A zum Zeitpunkt t_n wobei gilt: $thirst_A(t_n) \in \mathbb{N} \wedge thirst_A(t_n) \geq 1 \wedge thirst_A(t_n) \leq 14 \wedge t \geq 0$
$thirst_A(t_{n-1})$	Der Durst des Tieres A zum Zeitpunkt des letzten Ticks (t_1).
$basicTi_A(thirst_A(t_{n-1}))$	Eine Polynomfunktion die den Zuwachs an Durst, in Abhängigkeit zum letzten Durstwert, für den derzeitigen Tick bestimmt. Über diese Funktion kann die Zunahme des Durstes nach belieben durch den Benutzer parametrisiert werden.

$stressTi_A(t)$	Funktion die die Zunahme des Durstes durch körperliche Anstrengungen in diesem Tick zurückliefert.
$exceptionalTi_A(t)$	Funktion durch die ungewöhnliche Umstände in diesem Tick dargestellt werden, durch die sich der Durstwert erhöht.
$td_A(t)$	Funktion die die Abnahme des Durstes durch Aufnahme von Flüssigkeit in diesem Tick zurückliefert.

Abbildung 3.3: Beschreibung der Variablen, die in der Berechnung des Durstes eines Tieres verwendet werden.

3.2.4 Modellierung der Geparden

Verhalten von alleine jagenden Geparden

Das Verhalten von alleine jagenden Geparden wird in der Simulation relativ einfach gehalten. Der Fokus dieser Arbeit liegt auf dem Jagdverhalten der Geparden. Alle Aktivitäten die der Gepard ausführt zielen darauf ab Beutetiere zu erlegen. Wenn der Gepard nicht jagt, ist er auf der Suche nach einem Beutetier. Andere Verhaltensweisen und Ziele die nicht der Jagd dienen werden (noch) nicht berücksichtigt.

Das Verhalten eines Geparden-Agenten lässt sich mit einem einfachen endlichen Automaten beschreiben. Wenn der Gepard momentan kein Beutetier jagt, sucht er eines (**Roaming**). Dafür wird er sich auf die nächste Kopje bewegen, um durch die erhöhte Position einen besseren Sichtradius zu erhalten.

Sobald der Gepard ein Beutetier entdeckt hat, das momentan nicht auf der Flucht ist, wird er anfangen sich an dieses anzuschleichen (**Stalking**).

Hat sich der Gepard dem Beutetier weit genug genähert, ohne das dieses auf ihn aufmerksam geworden ist, wird er in einen Sprint übergehen, um das Beutetier zu erlegen (**Charging**).

Nachdem der Gepard seinen Sprint beendet hat muss er sich erst ausruhen um seine überhitzten Muskeln zu kühlen (**Resting**).

Sollte er das Beutetier erlegt haben wird er dieses nach dem ausruhen verspeisen (**Eating**) sonst wird er direkt dazu übergehen das nächste Beutetier zu suchen (**Roaming**).

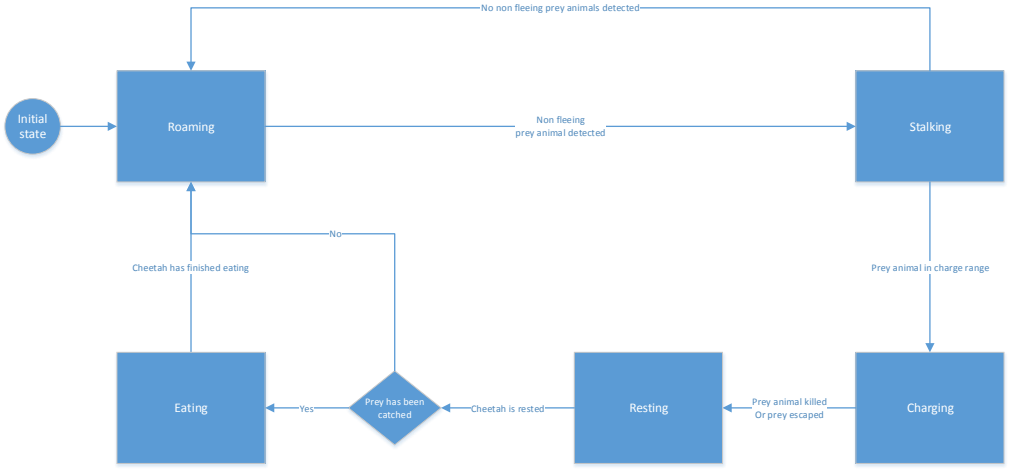


Abbildung 3.4: Beschreibung des Jagdverhaltens eines Geparden mit Hilfe eines endlichen Automaten.

Verhalten von in der Gruppe jagenden Geparden

In der Gruppe jagende Geparden bilden das Verhalten aus Shah (2013) ab.

Die Gruppe wird von einem leitendem Tier angeführt. Falls die Gruppe nicht jagt wird sie nach einem geeigneten Beutetier suchen (**Roaming**). Dafür wird sie, genauso wie alleine agierende Geparden, die nächste Kopje aufsuchen.

Ist ein Beutetier ausgemacht bleibt der Anführer zurück. Die anderen Geparden versuchen in eine flankierende Position zu kommen, die, vom anführenden Geparden, hinter dem Beutetier liegt (**Travel to flanking position, Follow the prey at a certain distance**).

Nimmt das Beutetier die Geparden frühzeitig wahr wird es fliehen und die Jagd ist gescheitert.

Sobald alle Geparden ihre Positionen erreicht haben gehen alle Geparden in einem Sprint über, um das Beutetier zu erlegen (**Charging**). Die Geparden versuchen dabei das Beutetier in Richtung Anführer zu treiben.

Nach dem Sprint müssen sich die Geparden ausruhen um ihre Muskeln abzukühlen (**Resting**).

War die Jagd erfolgreich wird die Beute verspeißt (**Eating**). Sonst machen sich die Geparden auf die Suche nach dem nächsten Beutetier (**Roaming**).

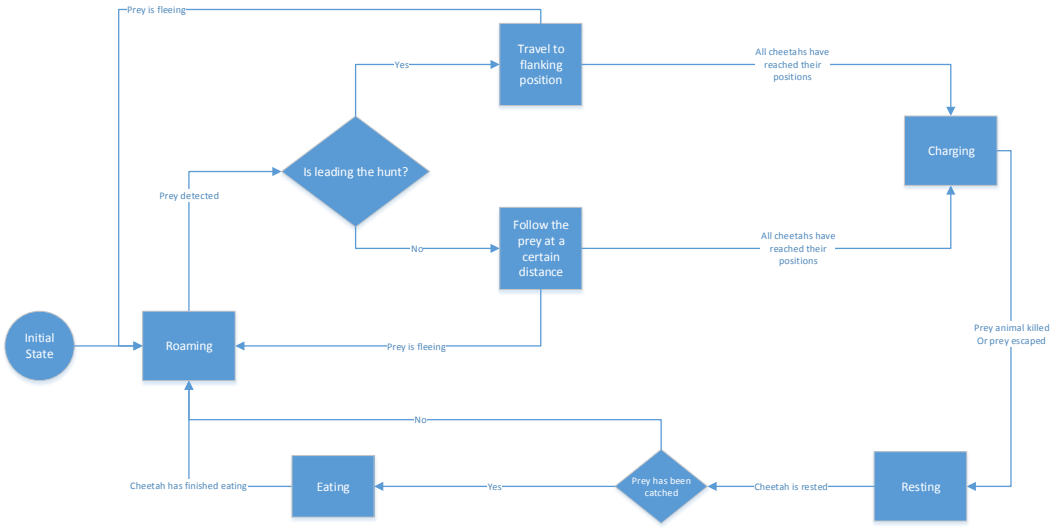


Abbildung 3.5: Beschreibung des Jagdverhaltens eines in der Gruppe agierenden Geparden mit Hilfe eines endlichen Automaten.

Berechnung des erwarteten Jagderfolgs

Um das Modell zu validieren kann der erwartete Erfolg einer Jagd mit dem tatsächlichen Ergebnis der Simulation abgeglichen werden. Ergibt die Ausführung der Simulation beispielsweise eine 70% Erfolgswahrscheinlichkeit für die Jagd ist das ein gutes Zeichen. Daraus lässt sich aber noch nicht entnehmen, ob die korrekten 70% der Jagden erfolgreich verlaufen.

Die Funktion zur Voraussage der Jagd erlaubt es dem Domänenexperten anzugeben, in welcher Situation er einen Jagderfolg erwartet. Die verwendeten Parameter sind aus 2.3 entnommen.

Zur Voraussage des Jagderfolgs wird ein einfacher Laplace-Experiment Ansatz verwendet. Jeder Parameter der für die Jagd entscheidend ist wird mit eine Koeffizienten versehen, der

bestimmt, ob der entsprechende Summand die Chance für einen Jagderfolg positiv, negativ oder gar nicht beeinflusst.
Die Summe der Parameter, multipliziert mit deren zugehörigen Koeffizienten, ist die Anzahl der Ergebnisse, in denen ein Jagderfolg erzielt wird.
Die Berechnung ist in diesem Fall bewusst simpel gehalten, um Fehler zu vermeiden und die Verständlichkeit des Modells zu erhöhen.

$$cheetahAge = age_{cheetah} * f_0(age_{cheetah})$$
$$cheetahHunger = hunger_{cheetah} * f_1(hunger_{cheetah})$$
$$preySize = size_{prey} * f_2(size_{prey})$$
$$distanceToKopje = distance_{kopje} * f_3(distance_{kopje})$$
$$distanceToRiver = distance_{river} * f_4(distance_{river})$$
$$availableCover = terrainCover * f_5(terrainCover)$$
$$cheetahHasCubs = hasCubs_{cheetah} * f_6(hasCubs_{cheetah})$$
$$huntTimeOfYear = timeOfYear * f_7(timeOfYear)$$

$$successfullHuntEvents = cheetahAge + cheetahHunger + preySize + distanceToKopje + distanceToRiver + availableCover + cheetahHasCubs + huntTimeOfYear$$

$$allPossibleEvents = |cheetahAge| + |cheetahHunger| + |preySize| + |distanceToKopje| + |distanceToRiver| + |availableCover| + |cheetahHasCubs| + |huntTimeOfYear|$$

$$\boxed{P(J) = \frac{successfullHuntEvents}{allPossibleEvents}}$$

Variable	Beschreibung
J	Das Ereignis, dass die Jagd erfolgreich verläuft.

$successfullHuntEvents$	Die Summe der Ereignisse in denen die Jagd erfolgreich verlaufen wird. Ist $successfullHuntEvents >= 100$ wird die Jagd garantiert erfolgreich sein. Ist $successfullHuntEvents <= 0$ wird die Jagd in jedem Fall fehlschlagen.
$P(J)$	Die Wahrscheinlichkeit mit der die Jagd erfolgreich verläuft.
$age_{cheetah}$	Alter des Geparden in Monaten
$hunger_{cheetah}$	Bauchumfang des Geparden (1 - 14 wobei bei 1 der Gepard verhungert und bei 14 der Gepard extrem gesättigt ist.)
$size_{prey}$	Die Größe des Beutetieres
$distance_{kopje}$	Die Distanz in Metern zur nächsten Kopje
$distance_{river}$	Die Distanz in Metern zum nächsten Fluss
$terrainCover$	Der Deckungswert des Terrains in der die Jagd stattfindet
$hasCubs_{cheetah}$	Das Vorhandensein von Gepardenjungen
$timeOfYear$	Die Jahreszeit (-1 für Trockenzeit +1 für Regenzeit)

$f_0(x_0)...f_n(x_n)$	Koeffizienten zur Gewichtung der einzelnen Parameter. Diese Koeffizienten werden aus einer Polynomfunktion berechnet, die in Abhängigkeit zum jeweiligen Parameter ($x_0 = s_A, x_1 = t_B$ usw.) steht. Im einfachsten Fall verhält sich diese Funktion linear. Aber es sind auch komplexere Verhaltensweisen möglich. Beispielsweise könnte ein Gepard bis zu einem gewiss hohen Alter immer effizienter darin werden Beutetiere zu jagen bis ab einem gewissen Alter diese Fähigkeit wieder rapide abnimmt. Für die Koeffizienten gilt: $f_n(x_n) \in \mathbb{R} \wedge f_n(x_n) >=$ $-1 \wedge f_n(x_n) <= 1$ $f_n(x_n) > 0 \Rightarrow$ Beeinflusst den Erfolg der Jagd positiv. $f_n(x_n) = 0 \Rightarrow$ Beeinflusst den Erfolg der Jagd nicht. $f_n(x_n) < 0 \Rightarrow$ beeinflusst den Erfolg der Jagd negativ.

Abbildung 3.6: Beschreibung der Variablen, die in der Vorhersage des Jagderfolgs verwendet werden.

3.2.5 Modellierung der Beutetiere

Verhalten eines Impalas

Das Verhalten von Impalas ist momentan sehr einfach gehalten. Die Hauptaufgabe der Impalas, in der Simulation, ist es den Geparden als Beute zu dienen. Daher sind nur wenige Aspekte des Verhaltens der Tiere von Interesse.

Impalas ziehen auf der Suche nach Nahrung und Wasser durch das Gelände (**Roaming**). Sie werden dabei versuchen mit anderen Imapalas, die sie wahrnehmen, eine Herde zu bilden.

Entfernen sich die Impalas einer Herde zu weit voneinander, werden sie versuchen die Kohäsion wieder herzustellen (**IncreasingHerdCohesion**).

Sobald die Tiere einen Geparden wahrnehmen werden sie vor diesem fliehen (**Fleeing**). Nach einer erfolgreichen Flucht gehen die Tiere wieder auf Nahrungssuche (**Roaming**).

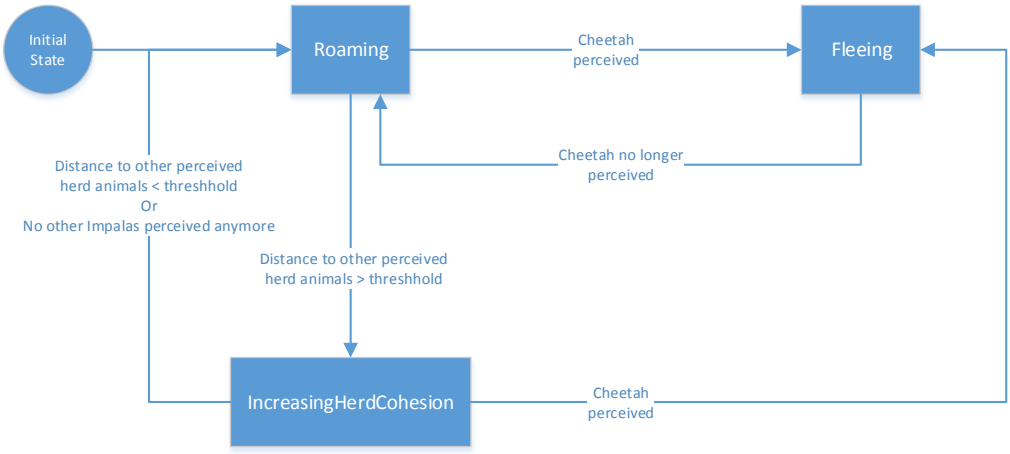


Abbildung 3.7: Deterministischer endlicher Automat der das Verhalten eines Impalas, in der Simulation, beschreibt.

Nährwert der Vegetation

Beutetiere ernähren sich von der Vegetation des Quadranten auf dem sie sich befinden. Der Typ der Vegetation bestimmt dessen Nährwert. Wegen der Annahme, dass die Agenten ihre Umgebung nicht ändern können, können die Tiere weder den Typ noch den Nährwert der Vegetation beeinflussen.

Der vorherrschende Typ der Vegetation in einem Quadranten ist von der Jahreszeit abhängig. Der Basis-Nährwert dieses Vegetationstyps kann durch spezielle externe Umstände modifiziert werden. Während einer Dürreperiode könnten Bäume beispielsweise ungewöhnlich wenig Früchte tragen, was ihren Nährwert reduzieren würde.

$$\boxed{n_C(D) = n_{C_base} + exceptionalnmod(D,t)}$$

Variable	Beschreibung

$n_C(D)$	Der Nährwert des Vegetationstyps C in Quadrant D.
n_{C_base}	Der Basis Nährwert des Vegetationstyps C.
$exceptionalnmod(D,t)$	Modifiziert den Nährwert der Vegetation in Quadrant D in Abhängigkeit zur Zeit. Dieser Modifikator ist dafür gedacht ungewöhnliche Umstände in einem Quadranten zu einer bestimmten Zeit darzustellen, die den Nahrungswert der Vegetation beeinflussen.

Abbildung 3.8: Beschreibung der Variablen, die in der Berechnung des Nährwertes der Vegetation verwendet werden.

Nährwert eines Beutetiers

Der Nährwert eines Beutetieres kommt dann zum tragen, wenn es von einem anderen Tier getötet und verspeist wird. Es wird vereinfacht angenommen, dass der Nährwert eines Tieres in direkter Abhängigkeit zu seinem letzten Hungerwert steht. Ausgehungerte Tiere besitzen weniger Masse als satte. Weniger Masse bedeutet weniger essbares Fleisch und damit einen niedrigeren Nährwert.

Die Geschwindigkeit mit der der Gepard seine Beute verzehrt ist in diesem Fall von besonderem Interesse. Da der Gepard seine erlegte Beute nicht vor konkurrierenden Raubtieren verteidigen kann, muss er diese möglichst schnell verspeisen.

Da es momentan in der Simulation keine Konkurrenten gibt, zieht eine niedrige Fressgeschwindigkeit direkt keine negativen Konsequenzen nach sich. Trotzdem ist das Fressverhalten ein integraler Bestandteil des Jagdverhaltens und damit von Interesse für die Simulation.

$$\boxed{n_B(t) = n_{B_basis} - nd_B(h_B(t))}$$

Variable	Beschreibung
$n_B(t)$	Nährwert eines Tieres B zum Zeitpunkt t.
n_{B_basis}	Der Basis Nährwert der Beutetierspezies von Tier B.
$eatSpeed_A(B)$	Eine Funktion, die zurückgibt, wie viele Ticks Tier A benötigt, um Tier B zu verzehren. Tier A muss dabei ein Gepard, und Tier B ein Beutetier sein.
b_B	Der Grundnährwert eines Tieres B. Wenn das Tier A ein Gepard ist, und dieser Tier B erlegt hat und dies verspeist, dann gilt: $n_B/eatSpeed_A(B) = hd_A(t)$
$nd_B(h_B(t))$	Eine vom Benutzer definierte Funktion, die die Abnahme des Nährwertes des Beutetiers in Abhängigkeit der Zunahme des Hungers des Beutetiers zurückgibt.

Abbildung 3.9: Beschreibung der Variablen, die in der Berechnung des Nährwertes eines Beutetiers verwendet werden.

3.3 Aufteilung der Simulation in Layer

Die Simulation ist in Layer aufgeteilt. Jeder Layer bietet Schnittstellen um den Zustand des Layers abzufragen oder zu verändern. Ein Layer kann andere Layer referenzieren. Layer können dabei beliebig komplex sein.
Dieses Konzept bietet eine sehr große Freiheit bei der Modellierung der Domäne.

Die Umwelt in der sich die Agenten bewegen ist eine Menge von Layern. Jeder dieser Layer beschreibt einen bestimmten Aspekt der Umwelt. Agenten können Informationen über ihre Umwelt erfragen oder ihre Umwelt verändern, indem sie die Schnittstellen der entsprechenden Layer verwenden.

Die Umwelt speißt sich aus einer Aggregation mehrerer Datenquellen. Zu jedem Zeitpunkt ist aber nur eine Untermenge dieser Informationen von Interesse.
Daher bietet jeder Layer eine gefilterte Sicht auf die Daten. Beispielsweise den Niederschlag oder die Vegetation in einem bestimmten Gebiet.
Für jeden Layer kann für jede Position und jeden Zeitpunkt ein exakter Wert für dessen Informationen abgefragt werden. Agenten nutzen diese Funktionalität um Informationen über ihre Umwelt zu beziehen.
Der Abschnitt Verfügbare Datenquellen enthält einen Überblick über die verschiedenen Quellen die für die Befüllung der Layer verwendet werden können. Im Abschnitt Benötigte Layer werde ich dann die Layer, und damit die Informationen, herausarbeiten, die für die Simulation von Interesse sind. Der Abschnitt Kommunikation zwischen den Layern befasst sich mit der Kommunikation der Layer der Fachdomäne untereinander.

3.3.1 Verfügbare Datenquellen

Um die Layer mit Daten zu versorgen, können verschiedene, teilweise offen verfügbare, Datenquellen genutzt werden. Die Daten die für die Simulation benötigt werden sind so vielfältig, dass nicht alle Informationen aus einer einzigen Quelle bezogen werden können.

Die SRTM-Mission bietet beispielsweise nur Höheninformationen. Wenn diese Daten nun mit denen der Global Land Cover Characterization angereichert werden, ergibt sich ein sehr viel aussagekräftigeres Bild. Die Idee ist es also, die Informationen der verschiedenen Datenquellen zusammenzuführen und durch die Layer-Struktur abfragbar zu machen. Für diesen Zweck stehen folgende Datenquellen zur Verfügung:

1. Globale Karten
    a) **SRTM** (Shuttle Radar Topography Mission) http://www2.jpl.nasa.gov/srtm/cbanddataproducts.html - Eine Karte der Höhenwerte des Geländes für die gesamte Erdoberfläche.
    b) **SRTM30Plus** http://topex.ucsd.edu/WWW_html/srtm30_plus.html - Eine Erweiterung des SRTM Datensatzes.
    c) **GLOBAL LAND COVER CHARACTERIZATION** http://edc2.usgs.gov/glcc/glcc.php - Globale Karte der Landnutzung (Dokumentation: http://edc2.usgs.gov/glcc/globdoc2_0.php).
    d) **GlobCover** http://due.esrin.esa.int/globcover/ - Globale Karte der Landnutzung.

e) **BIO DIVERSITY GIS** http://bgis.sanbi.org/index.asp?screenwidth=1920 - GIS-Daten über Tier und Pflanzenwelt.

f) **RedList Geparden Lebensräume** http://maps.iucnredlist.org/map.html?id=219 - Derzeitige Lebensräume von Geparden

2. Südafrika

   a) **Departement of Water Affairs - South Africa 1:500 000 Rivers** http://www.dwaf.gov.za/iwqs/gis_data/river/rivs500k.html - Alle Flüsse in Südafrika inklusive reichhaltiger Meta Informationen (bspw. Flussrichtung).

3. Nationalparks

   a) **Kruger National Park**

      i. Scientific Services http://www.sanparks.org/parks/kruger/conservation/scientific/gis/ - Sammlung verschiedenster GIS-Daten für den Krüger National-Park.

      ii. Biodiversity Summary - Kruger National Park Municipality http://bgis.sanbi.org/municipalities/show-muni-summaries.asp?muni=LIMDMA33 - Vegetation im Krüger National Park

   b) **Tanzania National Park** http://serengetidata.org/ (Passwort geschützt) - Sammlung verschiedenster GIS-Daten für die Serengeti

3.3.2 Benötigte Layer

Alle aus dem Fachlichen-Datenmodell bekannten Entitäten müssen auf den Layern der Simulation untergebracht werden. Wir benötigen daher folgende Layer:

1. **Tiere** - Auf diesem Layer leben die Geparden und ihre Beutetiere.

2. **Geländehöhe** - Die Höhe einer Position des Gebiets über Normalhöhennull (https://de.wikipedia.org/wiki/Normalh%C3%B6hennull) in Metern. Ist der Höhenunterschied zwischen zwei Quadranten A und B groß genug, können Agenten sich nicht von A nach B bewegen.

3. **Kopjes** (http://en.wikipedia.org/wiki/Monadnock) - An welcher Stelle gibt es im untersuchten Gebiet ein Kopje. Diese werden von Geparden gerne als Aussichtspunkte für die Jagd verwendet und sind damit für den Jagderfolg von entscheidender Bedeutung.

4. **Vegetation** - Welches ist der vorherrschende Vegetationstyp an welcher Stelle zu welcher Zeit im Jahr. Wichtig um das Verhalten und die Bewegungen der pflanzenfressenden Beutetiere zu bestimmen.

5. **Bauwerke** - Alle Bauwerke die von Menschen erschaffen wurden. In diesem Layer befinden sich beispielsweise Häuser, Zäune oder Straßen.

6. **Gewässer und Flüsse** (inklusive Wassertiefe) - Welche Gebiete sind mit Wasser bedeckt, und wie tief ist dieses? Alle Agenten benötigen Wasser um nicht zu verdursten. Ist das Wasser zu tief kann es nicht von den Agenten überquert werden.

7. **Deckung** - Wieviel Deckung bietet ein Gebiet einem Tier an einem bestimmten Punkt zu einer bestimmten Zeit im Jahr. Ein Quadrant kann einem Tier auf viele Arten Deckung bieten. Denkbar wären beispielsweise: Vegetation (hohes Gras, Bäume usw.), hügeliges Terrain, ausgetrocknete Flussläufe oder von Menschen errichtete Bauwerke oder Felder. Der Wert der Deckung hängt größtenteils stark von der jeweiligen Jahreszeit ab. Über das Jahr hinweg wird sich die Vegetation stark ändern. Flüsse die vorher ausgetrocknet waren können zu bestimmten Zeiten im Jahr Wasser führen. Felder werden über das Jahr neu bestellt und abgeerntet.
   Das zurückgegebene Datum ist ein abstrakter Wert, der aus der Abfrage des Vegetations-, Bauwerke- und Gewässer und Flüsse-Layers durch eine Bewertungsfunktion errechnet wird.

Die Entitäten aus dem Fachlichen Datenmodell werden wie folgt auf diese Layer verteilt:

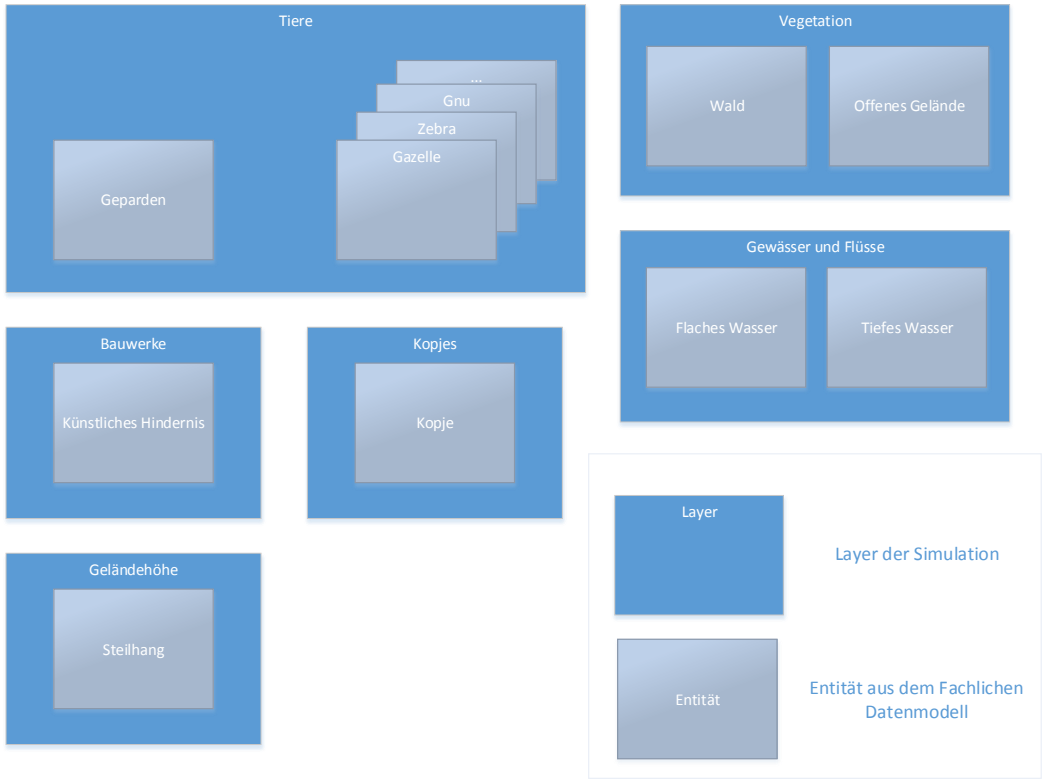


Abbildung 3.10: Verteilung der fachlichen Entitäten auf die Layer.

Geparden und Beutetiere befinden sich auf einem Layer. Eine Aufteilung der Tiere auf mehrere einzelne Layer ist, in diesem Fall, nicht zielführend. Beide Agentengruppen sind in ihrer Interaktion und Kommunikation stark voneinander abhängig. Eine Aufteilung auf mehrere Layer würde daher zu zyklomatischen Abhängigkeiten führen.

Einige der benötigten Layer für die Umwelt sind relativ simpel. Die Layer Geländehöhe und Kopjes können ihre Daten direkt aus den GIS-Daten beziehen. Die Geländehöhen werden sich, ebenso wie die Positionen der Kopjes, im Verlauf der Simulation nicht ändern.
Bei dem Vegetations-, Bauwerke- und Fluss-Layer ist die Situation komplexer. Die Daten die von diesen Layern an einer bestimmten Position ausgegeben werden, sind von der Zeit abhängig.
Die Vegetation wird sich, durch die Jahreszeiten, drastisch ändern. Zur Trockenzeit wird es sehr viel weniger Vegetation geben als zur Regenzeit. Bauwerke können abgerissen, versetzt oder

neu erbaut werden. Flüsse können unterschiedlich viel Wasser führen, oder ganz austrocknen. Der derzeit komplizierteste Layer der Simulation ist der Deckungs-Layer. Dieser Layer aggregiert seine Daten aus anderen Layern.

3.3.3 Kommunikation zwischen den Layern

Usecases

Ein Layer A benötigt eine Referenz auf einen anderen Layer B, wenn A Informationen von B benötigt. Um diese Abhängigkeiten herauszufinden verwenden wir UseCases. Diese beschreiben einen Anwendungsfall der Kommunikation zwischen mehreren Layern während eines Simulationsdurchlaufes.

Aufgrund dieser UseCases können dann im nächsten Schritt die Abhängigkeiten generiert werden.

1. **Agenten-Layer (Geparden und Beutiere)**:
    a) Tiere fragen die Position von Süßwasserquellen (Bspw. Flüssen) in ihrer Umgebung ab, um bei Bedarf zu trinken.
    b) Tiere fragen die Höhe ihres umliegenden Geländes ab, um zu bestimmen welche anderen Quadranten sie von ihrem derzeitigen Quadranten erreichen können.
    c) Tiere fragen die Deckungswerte an den Positionen der Agenten in ihrer Umgebung ab, um zu berechnen, ob sie diese wahrnehmen.

    a) **Geparden**:
        i. Geparden suchen nach Beutetieren in ihrer Umgebung, um diese zu jagen.
        ii. Geparden fragen den Nährwert von Beutetieren ab, wenn sie ein erlegtes Beutetier verzehren, um ihren Hunger zu reduzieren.
        iii. Geparden fragen die Position von Kopjes in ihrer Umgebung ab, um auf diesen nach Beutetieren Ausschau zu halten.
        iv. Geparden fragen die Deckung in ihrer Umgebung ab, um sich an Beutetiere heranzuschleichen.

    b) **Beutetiere (kleine, mittlelgroße und große)**:
        i. Beutetiere fragen die Position von Geparden in ihrer Umgebung ab, um ggf. vor ihnen zu fliehen.

ii. Beutetiere fragen die Art der Vegetation in ihrer Umgebung ab, um Nahrung zu finden.

iii. Beutetiere fragen den Nährwert der in der Umgebung befindlichen Vegetation ab, um ihren Hunger zu stillen.

2. **Deckungs-Layer**

a) Der Deckungs-Layer fragt die Vegetation, die Bauwerke und das Vorhandensein von Wasserquellen an einer Position ab, um, aufgrund dieser Daten, einen Deckungswert für diese Position zu ermitteln.

Referenzen der Layer

Aus den Usecases lassen sich nun die Abhängigkeiten der Layer herleiten.
Tiere müssen sehr viel über ihre Umgebung wissen. Daher benutzt der Tiere-Layer fast alle anderen Layer. Bauwerke sind für die Tiere nur soweit von Interesse wie sie Deckung bieten. Dies ist über den Deckungs-Layer abgedeckt.
Der Deckungs-Layer verwendet die Vegetation, Bauwerke und Flüsse um die Deckung an einem bestimmten Punkt zu errechnen.
Alle anderen Layer benötigen keine Informationen von außen und haben daher auch keine Referenzen auf andere Layer.

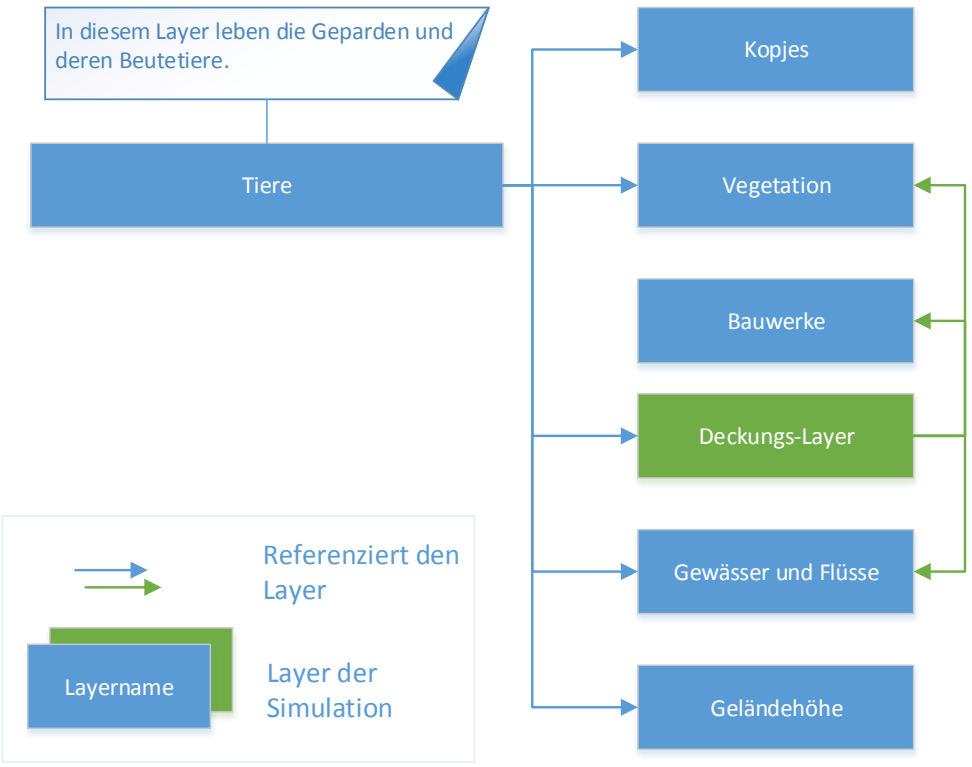


Abbildung 3.11: Abhängigkeiten der Layer untereinander.

4 Realisierung

4.1 Platzierung der Geparden und ihrer Beutetiere

Es muss eine große Anzahl von Geparden und deren Beutetieren im Gebiet platziert werden (mehrere Tausend). Diese manuell zu platzieren würde zu viel Zeit in Anspruch nehmen.
Es ist also wünschenswert die Platzierung der Geparden und Beutetiere in einer Datei abzuspeichern, um diese dann beliebig oft für Simulationsdurchläufe zu laden und ggf. zu modifizieren.
Hierfür bietet sich eine GIS-Datei an, in der die Platzierung der Tiere abgespeichert ist.

4.2 Speicherung der Simulationsdaten mit einem Datacube

Der aktuelle Zustand der Simulation ist definiert als die Menge der Zustände aller Layer. Um den Verlauf der Simulation später nachvollziehen zu können, muss am Ende jedes Ticks der neue Zustand aller Layer gespeichert werden.
Der Zustand eines Layers kann dabei beliebig komplex werden.

Um die anfallenden Datenmengen effizient speichern und abfragen zu können benötigen wir eine Relationale-Datenbank. Das Problem mit Relationalen-Datenbanken ist, dass diese normalerweise nur schlecht mit temporal veränderlichen Daten umgehen können. Diese Einschränkung lässt sich jedoch durch die Verwendung eines speziellen Datenbankschemas, auf Kosten der Normalisierung, umgehen.
Ein weiteres Problem ist die verschiedenartige Skalierung der zur Verfügung stehenden Daten. Daten über Flussläufe mögen beispielsweise bis auf 100 Meter genau sein, während die Höheninformationen in dem Gebiet nur mit einer Auflösung von 800 Metern zur Verfügung stehen.

Es ist nun aber erforderlich diese Daten, an einem beliebigen Ort und zu einer beliebigen Zeit, abfragen zu können. Thiel-clemen (2013) schlägt vor, das aus Data-Warehouse-Anwendungen bekannte Stern-Datenbankschema auf ökologische Daten anzuwenden.

4.2.1 Datenbankschema

Um den Zustand der Layer in der Datenbank zu speichern verwende ich eine an dieses Szenario angepasste Version des von Herrn Prof. Dr. Thiel-Clemen beschriebenen Datenbankschemas Thiel-clemen (2013):

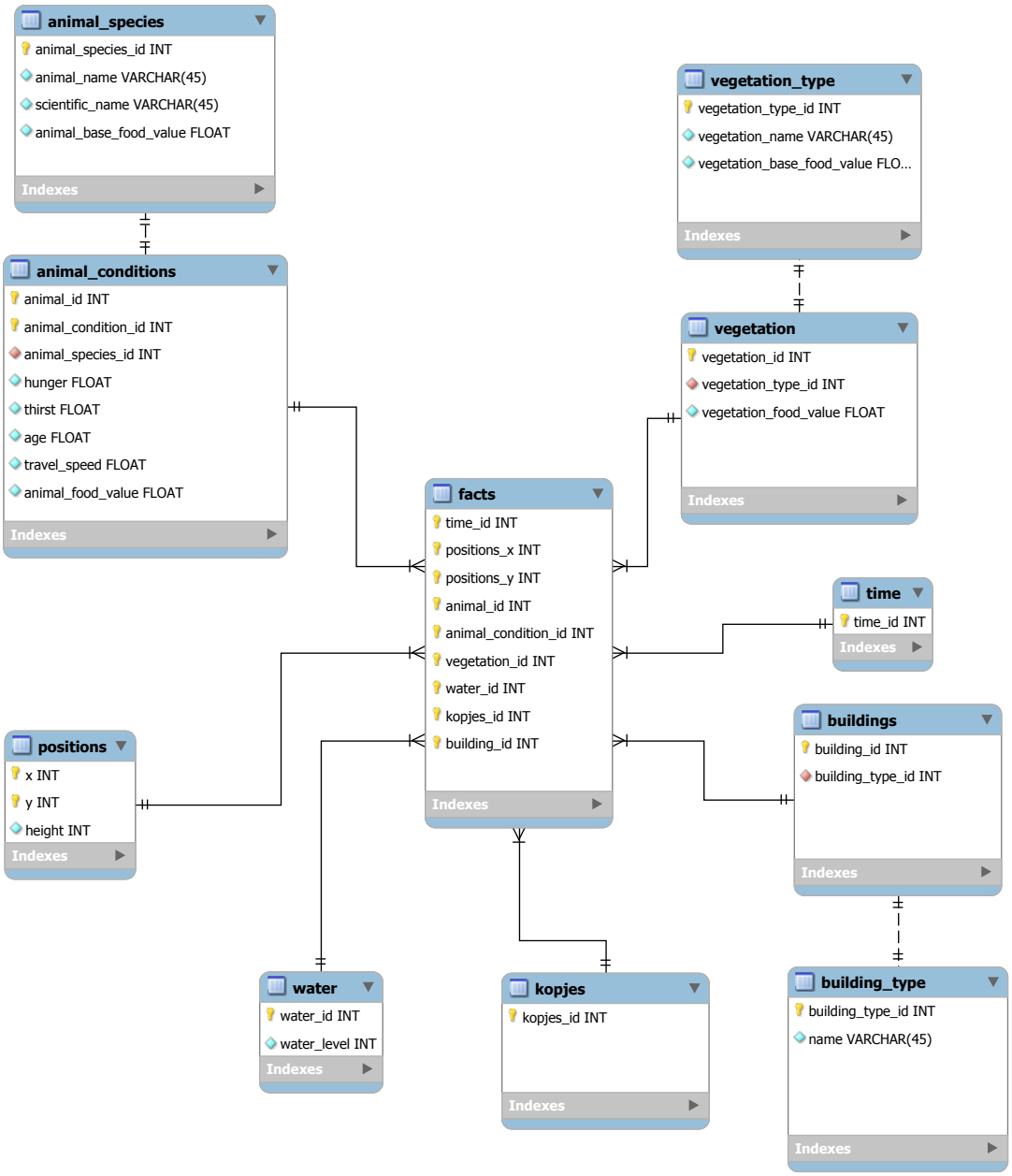


Abbildung 4.1: Datacube Datenbankschema zur Speicherung des Simulationsverlaufs.

Die **time-Tabelle** stellt die grundlegende Funktionalität zur Identifikation eines Ticks bereit. Ein Tick ist die atomare Zeiteinheit der Simulation. Jeder Eintrag in dieser Tabelle beschreibt genau einen Simulations-Tick. Über geschickte Datenbankabfragen, auf die ich später näher eingehen werde, kann damit der Zustand der gesamten Simulation (also den Zustand aller Layer) zu jedem Zeitpunkt (Tick) bestimmt werden.

Die **position-Tabelle** gibt uns die Möglichkeit die Position (Koordinaten des Quadranten) unserer Entitäten der Simulation auf dem Gelände zu speichern. Zusätzlich wird in dieser Tabelle die Höhe des Geländes gespeichert.
Mit Hilfe der **time-** und **positions** Tabellen können wir also die Bewegung von Entitäten über den Verlauf der Simulation speichern. Dies ist beispielsweise notwendig für die Speicherung der Bewegung der Geparden und ihrer Beutetiere.

Die **animal_conditions-** und **animal_species-** Tabellen speichern den Zustand und die Spezies der jeweiligen Tiere. Die animal_conditions-Tabelle besitzt einen zusammengesetzten Schlüssel. Dieser setzt sich aus der ID des Tieres und einer ID für dessen Zustand zusammen. Damit lässt sich der Zustand eines bestimmten Tiers über den Simulationsverlauf hinweg verfolgen. Alle anderen Entitäten der Simulation bewegen sich nicht, und können daher über ihren Standort eindeutig identifiziert werden.

Die **vegetation-** und **vegetation_type-** Tabellen kümmern sich um die Speicherung des Zustands und der Art der Vegetation. Es genügt hier nicht nur die Art der Vegetation zu speichern, da sich der Nährwert der Vegetation an einer Position durch besondere Umstände verändern kann.

Die **water-**Tabelle wird verwendet, um die Wassertiefe an einer bestimmten Position zu einer bestimmten Zeit abzuspeichern.

Die **kopjes-** und **buildings-** Tabellen speichern die Positionen der Kopjes und Gebäude. Mit der **building_type-**Tabelle kann die Art eines Gebäudes bestimmt werden.

Über geschickte Datenbankabfragen kann nun der Zustand der Simulation, an einer beliebigen Position zu einem beliebigen Zeitpunkt, abgefragt werden.

An folgendem Beispiel-Szenario werde ich zeigen, dass das Datenbankschema für das geforderte Szenario gültig ist.

Da es momentan nur darum geht, zu zeigen, dass das Datenbankschema für die Fachdomäne gültig ist, können wir folgende Vereinfachungen treffen:

1. Obwohl Technische Aspekte wie Zugriffszeiten, Speicherbedarf, Skalierbarkeit, Verteilbarkeit etc. eine entscheidende Rolle spielen, sind diese für die Validierung des Datenbankmodells erst einmal nicht von Interesse. Wie wir beispielsweise sehen werden, könnte die große Menge an zu speichernden Daten in einem Tick zu einem echten Problem werden.

2. Datenbankabfragen in diesem Abschnitt sind nicht optimiert. Schnelle Leseoperationen sind bereits eine große Stärke dieser Lösung. Eine Optimierung der Schreiboperationen ist in diesem Fall sehr viel interessanter.

3. Eine kleine Anzahl von Sektoren genügt. Die Überprüfung der Gültigkeit des Datenbankmodells für die Modellierung wird durch eine Erhöhung der Quadranten-Anzahl nicht beeinflusst.

4. Eine kleine Anzahl unterschiedlicher Tierarten ist ausreichend. Wenn das Datenbankmodell zwei Tierarten darstellen kann, kann es auch beliebig mehr abbilden.

5. Eine biologisch gültige Parametrisierung des Modells ist nicht notwendig.

6. Eine Generierung des Geländes aus GIS-Daten ist nicht notwendig.

4.2.2 Beispiel-Szenario

Dies ist der initiale Zustand der Simulation:

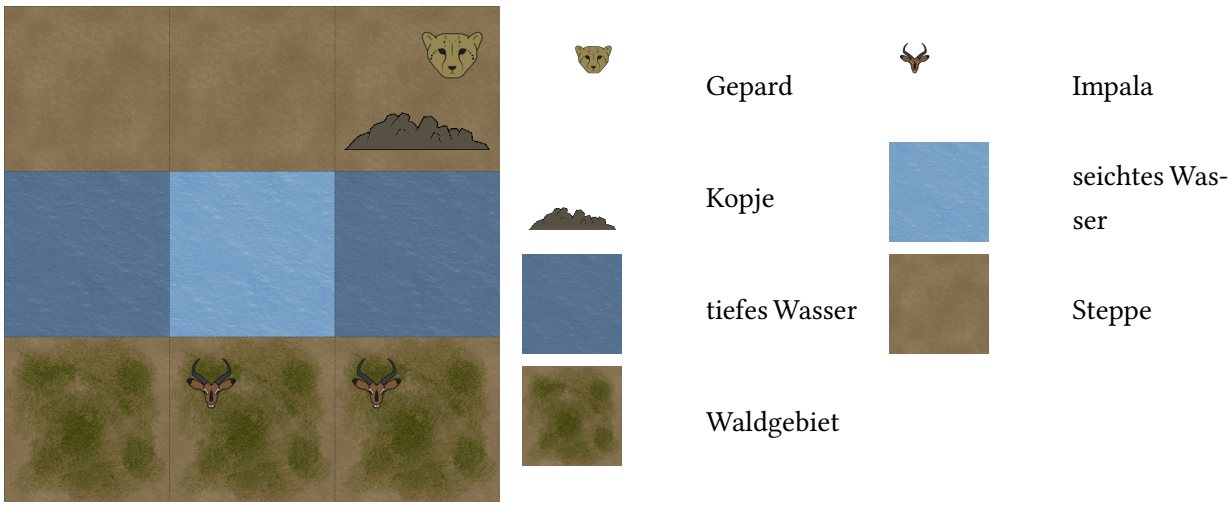


Abbildung 4.2: Initialer Zustand der Simulation des Beispiel-Szenarios.

In diesem Beispiel sehen wir einen Geparden der auf einer Kopje grasende Impalas beobachtet. Die Impalas und der Gepard sind durch einen Fluss getrennt, der nur an einer Stelle flach genug ist, um überquert zu werden.
Ein Abschnitt der Karte repräsentiert einen Quadranten. In diesem Beispielszenario gibt es also neun Quadranten. Das Koordinatensystem beginnt beim unteren linken Quadranten mit x = 0 und y = 0 und endet beim oberen rechten Quadranten mit x = 2 und y = 2. Die x-Achse beschreibt dabei die horizontale und die y-Achse die vertikale Komponente des Positionsvektors.

Mit folgender SQL-Abfrage erhalten wir alle Informationen die derzeit in der Datenbank zu dieser Simulation vorliegen:

```
SELECT
        t.time_id,
        p.X, p.Y,
        ac.animal_id,
        ac.animal_condition_id,
        ac.animal_species_id,
        ac.hunger,
        ac.thirst,
        ac.age,
        ac.travel_speed,
        ac.animal_food_value,
        v.vegetation_id,
        v.vegetation_type_id,
        v.vegetation_food_value,
        w.water_id, w.water_level,
```

```
        k.kopjes_id,
        b.building_id, b.building_type_id
FROM
        facts f,
        time t,
        positions p,
        animal_conditions ac,
        vegetation v,
        water w,
        kopjes k,
        buildings b
WHERE
        f.time_id = t.time_id
        and f.positions_x = p.x
        and f.positions_y = p.y
        and f.animal_id = ac.animal_id
        and f.animal_condition_id = ac.animal_condition_id
        and f.vegetation_id = v.vegetation_id
        and f.water_id = w.water_id
        and f.kopjes_id = k.kopjes_id
        and f.building_id = b.building_id
```

Abbildung 4.3: Query1: Gibt alle Informationen zum bisherigen Simulationsverlauf aus.

Diese Abfrage liefert folgendes Ergebnis:

time_id	X	Y	animal_id	animal_condition_id	animal_species_id	hunger	thirst	age	travel_speed	animal_food_value	vegetation_id	vegetation_type_id	vegetation_food_value	water_id	water_level	kopjes_id	building_id	building_type_id
0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	2	0	2	2	0	15	30	10	2	1	1	0	0	0	0	0
0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2	0	3	0	2	3	0	16	30	10	3	1	1	0	0	0	0	0
0	2	2	1	0	1	10	5	26	50	2.5	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	30	0	0	0
0	2	1	0	0	0	0	0	0	0	0	0	0	0	1	30	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	2	300	0	0	0

Abbildung 4.4: Ergebnis von Query1

Als erstes fällt die hohe Anzahl der Nullen auf. Eine Null bedeutet per Konvention, dass es an dieser Position zu diesem Zeitpunkt keine Entität des entsprechenden Typs gibt. Die Zeit und die Position bilden hier semantische Ausnahmen.

Ein Wert von 0 in der Zeit-Tabelle hat die Bedeutung des initialen Zustands der Simulation. Es wurde noch kein Tick berechnet.

Eine Eintrag von 0 in der X-Achse beschreibt die am weitesten links (westlich) gelegene Position eines Quadranten. 0 in der Y-Achse beschreibt die am weitesten unten (südlich) gelegene Position eines Quadranten. Ein Quadrant auf Position X = 0 und Y = 0 ist also der linke unterere Quadrant.

In der facts-Tabelle gibt es einen oder mehrere Einträge für den Zustand jedes Quadranten zu jedem Zeitpunkt (Tick). Die Anzahl der vorhandenen Zeitpunkte wächst mit der Ausführung der Simulation.
Jeder dieser Einträge ist ein zusammengesetzter Schlüssel aus Fremdschlüsseln. Daraus resultiert, dass in der Facts-Tabelle keine null-Werte auftauchen dürfen.

Nur in Ausnahmefällen sind in einem Quadranten, zu einem Zeitpunkt alle Entitäten vertreten. Konkret würde dieser Fall eintreten, wenn sich ein Gepard in einem sumpfigen Gebiet (Wasser und Vegetation) mit einem Gebäude und einer Kopje befindet. In diesem Fall wären alle Fremdschlüssel für die Entitäten != 0.
Es wäre semantisch schöner gewesen, für nicht vorhandene Entitäten null anzugeben. Fremdschlüssel dürfen jedoch aus offensichtlichen Gründen niemals den Wert null annehmen.

Die Menge der Einträge in der facts-Tabelle, pro Tick, wächst linear mit der Menge der Quadranten in der Simulation. Die Menge der Quadranten wächst quadratisch mit der Vergrößerung des Szenarios (unter der Voraussetzung, dass jede Vergrößerung gleich viele Spalten und Zeilen an Quadranten hinzufügt). Die Datenmenge wird also sehr schnell sehr groß. Da die schnelle Speicherung und der performante Zugriff auf diese Daten kritisch ist, kommt nur die Speicherung in einer Datenbank in Frage.

4.2.3 Beweis

Beweisidee

Wenn das Datenbankschema den initialen Zustand der Simulation und den Zustand im folgenden Tick darstellen kann, dann funktioniert dies auch für alle darauf folgenden Zustände.

Der Zustand der Simulation, zu einem Zeitpunkt, ist die Menge der Zustände der Entitäten der Simulation. Der Zustand einer Entität setzt sich aus einer Menge von primitiven Datentypen zusammen.

Die Datenbank bildet den Zustand der Simulation also korrekt ab, wenn jedem Zeitpunkt die richtige Menge an Entitäts-Zuständen zugewiesen wird.
Um den Beweis einfach zu halten genügt es also den Verlauf einer Eigenschaft einer Entität über die Zeit zu verfolgen.

Das Beweisschema welches sich hierfür anbietet ist die vollständige Induktion.

Induktionsanfang

Als erstes überprüfen wir, ob das Datenbankschema den initialen Zustand der Simulation darstellen kann. Die gewählte Entität ist der Gepard unseres Beispiels. Die folgende Abfrage liefert das Tier welches sich an Position X = 2 und Y = 2 zu Zeitpunkt t_0 befindet:

```
SELECT
        r.animal_id, a.animal_name, a.scientific_name
FROM
        (
        -- Query 1
        ...
        ) r,
        animal_species a
WHERE
        r.X = 2 and r.Y = 2
        and r.time_id = 0
        and a.animal_species_id = r.animal_species_id
```

Abbildung 4.5: QueryInduktionsanfang

Diese Abfrage liefert das folgende Ergebnis:

animal_id	animal_name	scientific_name
1	Cheetah	Acinonyx jubatus

Abbildung 4.6: Ergebnis: QueryInduktionsanfang

An der Position X = 2 und Y = 2 befindet sich zu Zeitpunkt t_0 also der Gepard mit der ID 1. Ein Vergleich mit dem Szenario bestätigt die Ausgabe der Query:

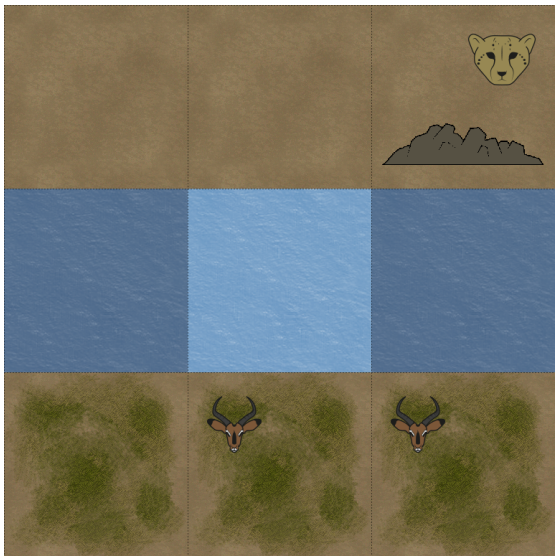


Abbildung 4.7: Der Zustand der Simulation zu Zeitpunkt t_0

Das Datenbankschema ist also in der Lage den Zustand der Simulation zu einem bestimmten Zeitpunkt (Tick) zu speichern.

Induktionsbehauptung

Wenn das Datenbankschema zur Darstellung des Zustandes der Simulation zu einem Zeitpunkt t funktioniert, so wird dies auch für den Zeitpunkt $t + 1$ gelingen.

Induktionsschluss

Im nächsten Simulationsschritt ($t_0 + 1$) bewegt sich der Gepard einen Quadranten nach links.

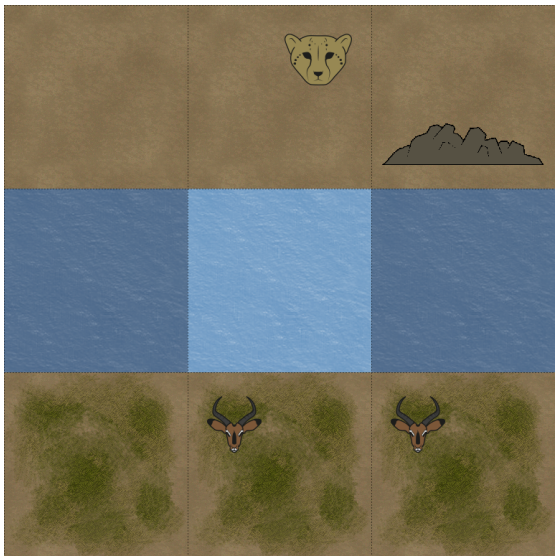


Abbildung 4.8: Zustand der Simulation zum Zeitpunkt $t_0 + 1$.

Eine weitere Abfrage des Datenbankinhaltes ergibt folgendes Ergebnis:

time_id	X	Y	animal_id	animal_condition_id	animal_species_id	hunger	thirst	age	travel_speed	animal_food_value	vegetation_id	vegetation_type_id	vegetation_food_value	water_id	water_level	kopjes_id	building_id	building_type_id
0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2	2	1	0	1	10	5	26	50	2.5	0	0	0	0	0	1	0	0
1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	2	1	1	1	10	5	26	50	2.5	0	0	0	0	0	0	0	0
1	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	30	0	0	0
0	2	1	0	0	0	0	0	0	0	0	0	0	0	1	30	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	30	0	0	0
1	2	1	0	0	0	0	0	0	0	0	0	0	0	1	30	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	2	300	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	2	300	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	1	0	2	0	2	2	0	15	30	10	2	1	1	0	0	0	0	0
1	1	0	2	1	2	2	0	15	30	10	2	1	1	0	0	0	0	0
0	2	0	3	0	2	3	0	16	30	10	3	1	1	0	0	0	0	0
1	2	0	3	1	2	3	0	16	30	10	3	1	1	0	0	0	0	0

Abbildung 4.9: Inhalt des Datacubes zum Zeitpunkt $t_0 + 1$.

Der komplette Zustand der Simulation zum neuen Zeitpunkt ist in den Zeilen mit der time_id = 1 ($t_0 + 1$) gespeichert.

Wir müssen nun überprüfen, ob das Datenbankschema die Position des Geparden immer noch richtig angibt. Wir können dazu die bereits bekannte Abfrage verwenden:

```
SELECT
        r.animal_id, a.animal_name, a.scientific_name
FROM
        (
        ...
        ) r,
        animal_species a
WHERE
        r.X = 1 and r.Y = 2
        and r.time_id = 1
        and a.animal_species_id = r.animal_species_id
```

Abbildung 4.10: QueryInduktionsschluss1

In der Abfrage muss natürlich die abgefragte Position und Zeit entsprechend geändert werden.

Die Abfrage liefert das richtige Tier zurück:

animal_id	animal_name	scientific_name
1	Cheetah	Acinonyx jubatus

Abbildung 4.11: Ergebnis: QueryInduktionsschluss1

Nun müssen wir nur noch sicherstellen, dass sich an der alten Position des Geparden kein Tier mehr befindet:

```
SELECT
        r.animal_id, a.animal_name, a.scientific_name
FROM
        (
        ...
        ) r,
        animal_species a
WHERE
        r.X = 2 and r.Y = 2
        and r.time_id = 1
        and a.animal_species_id = r.animal_species_id
```

Abbildung 4.12: QueryInduktionsschluss2

Diese Abfrage liefert keine Ergebnisse zurück. An der alten Position des Geparden befindet sich zu Zeitpunkt $t_0 + 1$ also kein Tier. Das ist korrekt.

Das Datenbankschema erfüllt also unsere Anforderungen.

4.3 Ein erster Prototyp

Um die einige Kernaspekte der Modellierung zu testen habe ich einen Prototypen entwickelt.

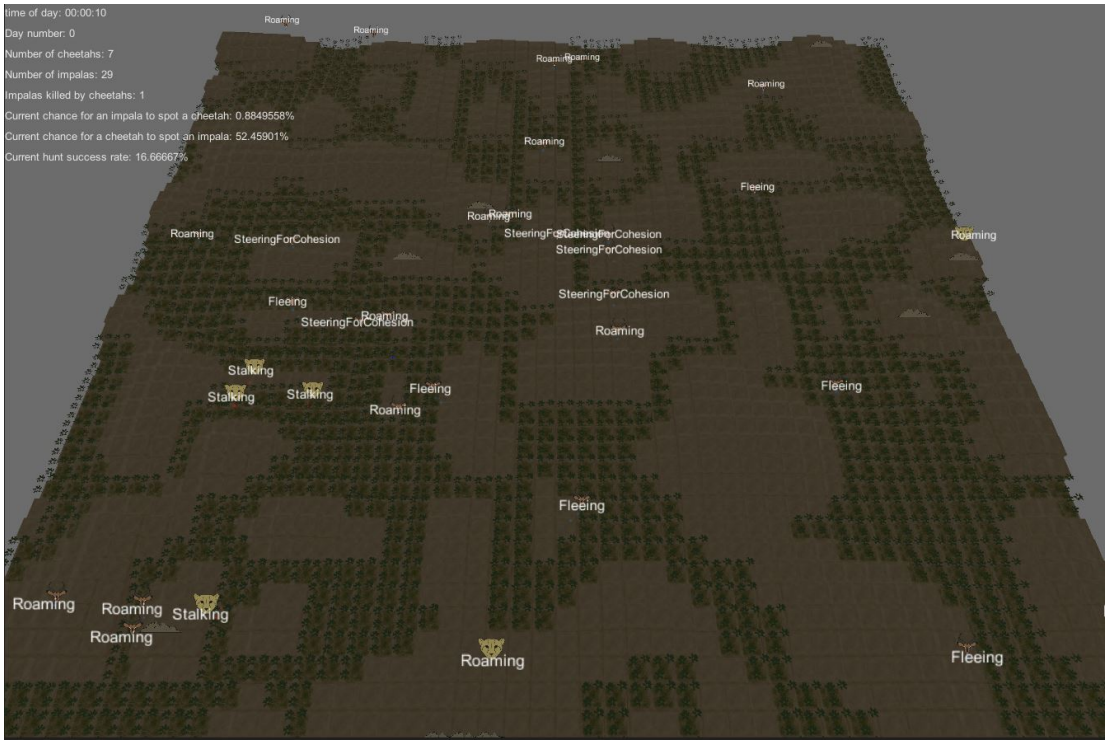


Abbildung 4.13: Screenshot des Prototypen während eines Simulationsdurchlaufes.

Das erste Hauptziel dieses Prototypen ist es eine anschauliche Vision für die weitere Entwicklung der Simulation in WALK erstellen. Quasi also eine Vision in welche Richtung ich mit der Simulation gehen möchte.
Fehler in der Modellierung die auf das Zusammenspiel von verschiedenen Komponenten zurückzuführen sind, sind nur schwer beim Entwurf zu entdecken. Solche Fehler haben das Potential, in der späteren Entwicklung, viel Arbeit zu verursachen. Das zweite Ziel ist es also die Modellierung auf Fehler und Unstimmigkeiten zu überprüfen und diese frühzeitig zu beseitigen. Die Entwicklung eines Prototypen lohnt sich also. Momentan umfasst der Prototyp folgende Funktionalitäten:

1. Aufteilung des Geländes in Quadranten wie in Abschnitt 3.1 beschrieben.
2. Simulierung diskreter Zeitschritte wie in Abschnitt 3.1.1 beschrieben. In jedem Zeitschritt führen alle Agenten ihre Aktionen aus.

3. Gelände mit Höhenunterschieden.
4. Tiere nehmen andere Tiere nur mit einer bestimmten Wahrscheinlichkeit wahr. Diese Wahrscheinlichkeit errechnet sich wie in Abschnitt 3.2.2 beschrieben.
5. Die Jagd von Beutetieren durch Geparden verläuft nur mit einer gewissen Wahrscheinlichkeit erfolgreich. Die Wahrscheinlichkeit für eine erfolgreiche Jagd wird wie in Abschnitt 3.2.4 berechnet.
6. Implementierung der Layer-Struktur wie in Abschnitt 3.3 beschrieben.
7. Agenten nehmen Informationen über ihre Umwelt über die Layer-Struktur wahr.

Der Prototyp hat folgende Einschränkungen:

1. In der Modellierung sind die Koeffizienten als Funktionen vorgesehen (siehe Abschnitt 3.2.2 und 3.2.4). In dem Prototyp sind diese Werte als einfache feste Werte implementiert. Es gibt momentan in Unity keine benutzerfreundliche Möglichkeit Funktionen zu editieren.
   Der eingebaute AnimationGraph-Editor würde diese Funktionalität theoretisch erfüllen, ist aber eine Usability-Technische Katastrophe.
   Die mir bekannten Frameworks zum parsen von Funktionstermen sind leider nicht mit der in Unity verwendeten Version von Mono kompatibel.

Folgende Funktionalitäten sind noch nicht im Prototypen implementiert:

1. Generierung des Geländes aus GIS-Daten. Das Gelände wird zufällig aus einer Noise-Funktion generiert.
2. Die Berechnung der Simulation erfolgt nicht verteilt.
3. Eine Anbindung an den Datacube ist nicht implementiert. Die Daten werden transient im Speicher gehalten. Die Ergebnisse können nicht gespeichert und der Verlauf einer Simulation nicht im Nachhinein nachvollzogen werden.

Durch die Verwendung von Unity entstehen einige Einschränkungen. Diese werden aber durch die schnelle Entwicklungsgeschwindigkeit ausgeglichen.
Die größte Einschränkung für die Performance ist die API. Unitys API ist nicht Thread-Safe und alle Zugriffe eines, nicht Unity eigenen, Threads lösen eine Exception aus. Alle Zugriffe auf die API müssen vom Unity eigenen Thread aus erfolgen. Durch diesen Flaschenhals wird die

Anzahl der Agenten deutlich beschränkt. Da eine hohe Anzahl an Agenten für den Prototyp nicht von Relevanz ist, kann diese Einschränkung hingenommen werden. Die sehr alte Mono-Version schränkt uns in diesem Fall auch nicht weiter ein, da keine externen Bibliotheken verwendet werden.

4.3.1 Implementierung

Für die Implementierung habe ich einige grundlegende Design-Entscheidungen getroffen, die sich durch das gesamte System ziehen. Solange kein Return-Wert erforderlich ist erfolgt die Kommunikation über das verschicken und Empfangen von Nachrichten über ein zentrales Messaging-System. Dadurch wird die Kopplung der einzelnen Komponenten stark reduziert und neue Komponenten können leicht hinzugefügt werden.
Alle zentralen Bestandteile des Systems sind als Singletons implementiert. Hierzu zählen unter anderem die Layer. Für eine generische Lösung würde man die Abhängigkeiten der Layer per Dependency-Injection auflösen. Für einen Prototypen ist dieser Aufwand unnötig.
Grundsätzlich gelten die StyleCop Konventionen. Außerdem sind alle Felder, die mit dem Präfix "cfg"beginnen, zur Konfiguration durch den Benutzer vorgesehen. Die Werte für diese Felder dürfen vom Benutzer nicht, obwohl Unitys Editor das zulassen würde, zur runtime verändert werden. Cfg-Felder sind immer private und werden durch das [SerializeField] Attribut im Editor editierbar gemacht.

Infrastruktur

Der TickTimer und das MessagingSystem bilden die zentrale Infrastruktur des Systems.
Der TickTimer ist der zentrale Zeitgeber der Simulation. Die Hauptaufgabe des TickTimers ist es in regelmäßigen Abständen ein TickEvent abzuschicken um die Berechnung eines neuen Ticks zu veranlassen. Die derzeitige Tageszeit, Tag und Jahreszeit können hier ebenso abgefragt werden wie die verbleibende Zeit zum nächsten Tick.
Das MessagingSystem ist die zentrale Implementierung des Observer-Patterns. Es bietet die Möglichkeit sich für bestimmte Nachrichtentypen einzuschreiben, auszuschreiben und Nachrichten per Broadcast zu verschicken.

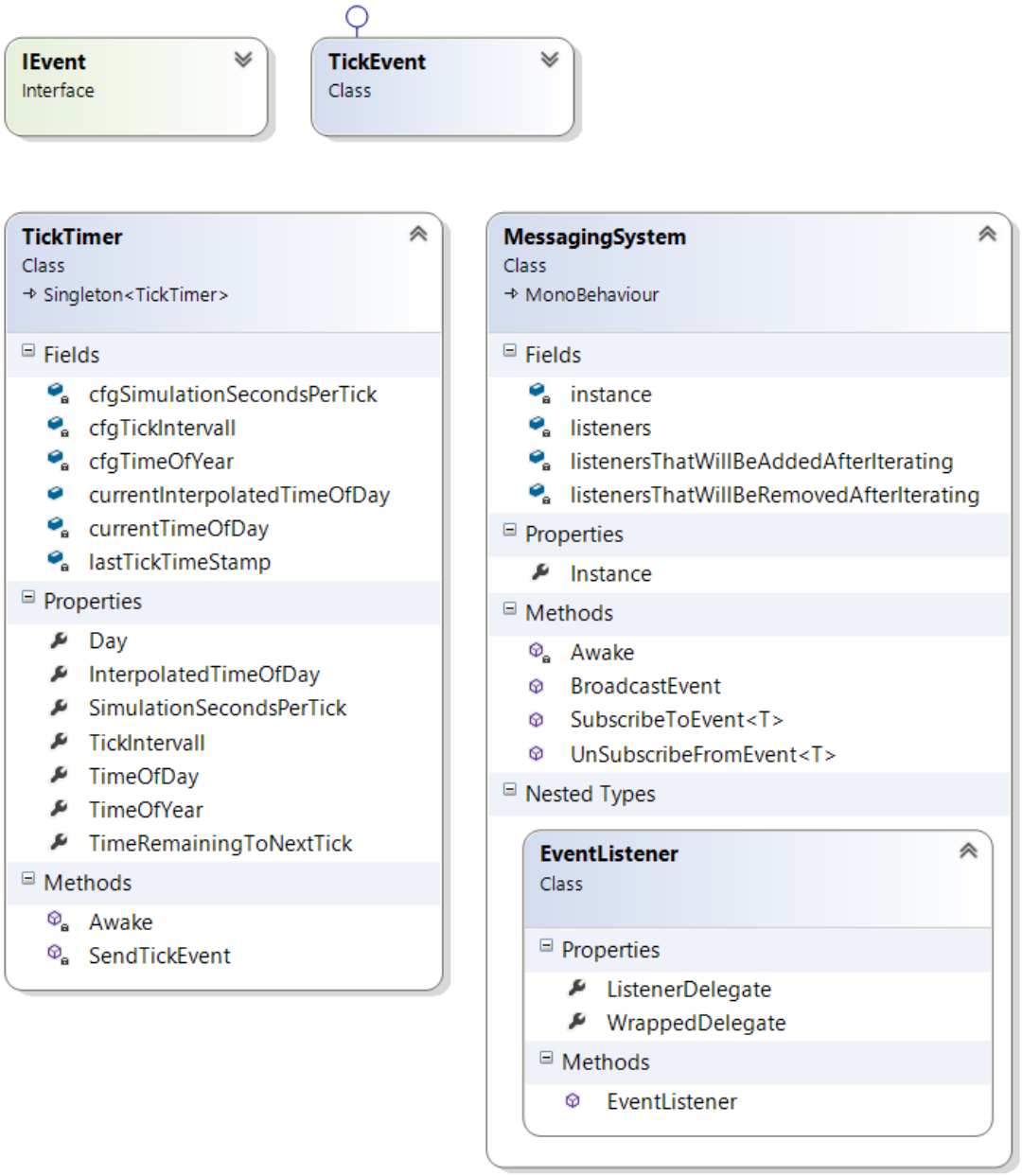
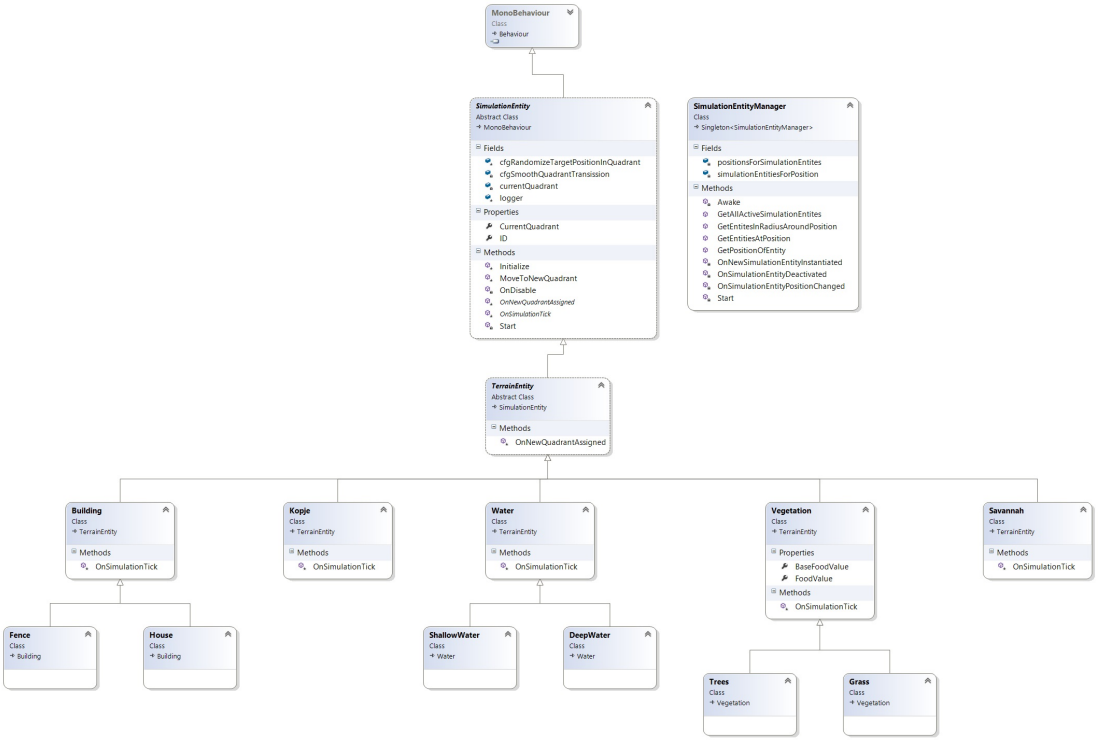


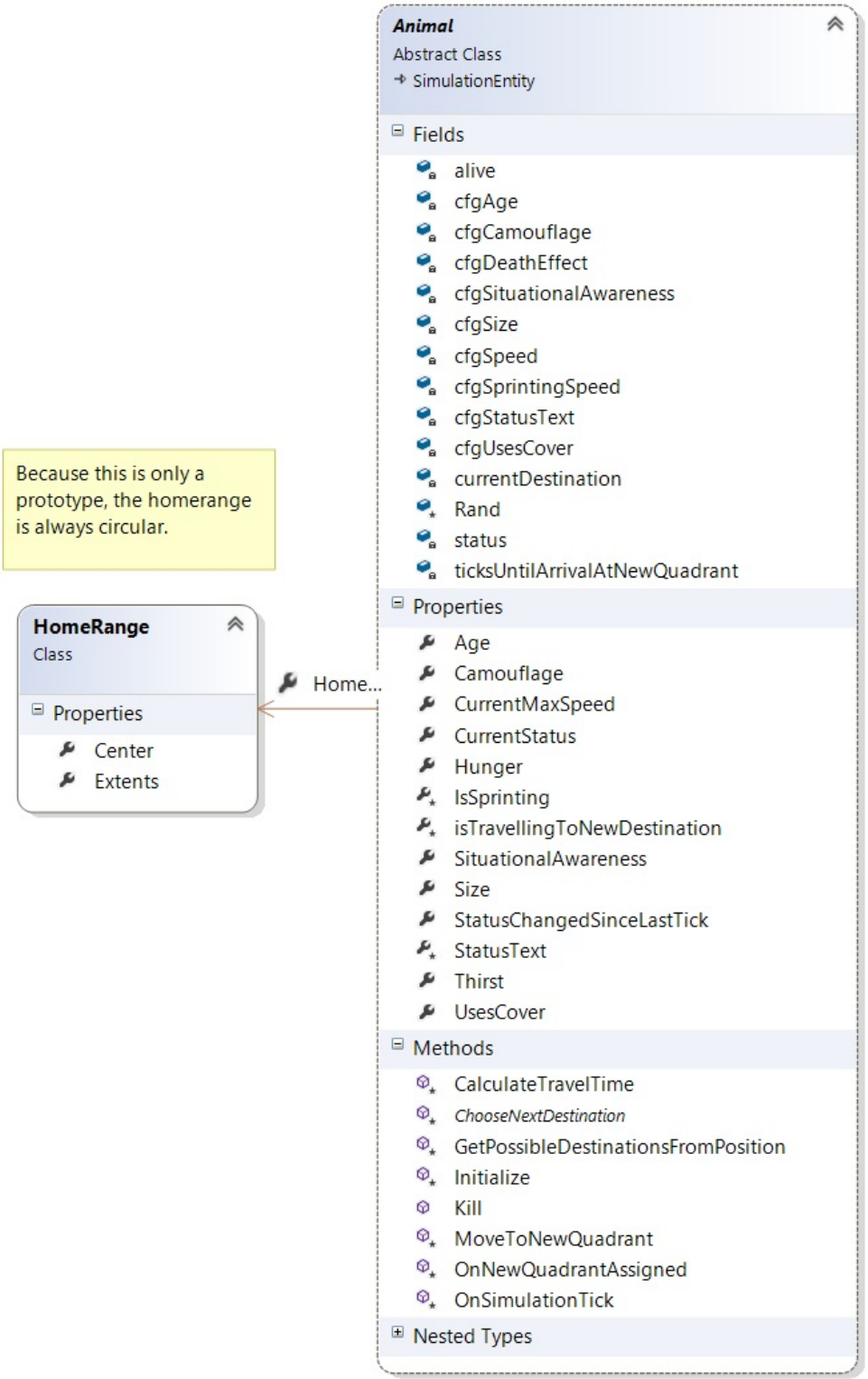
Abbildung 4.14: Infrastruktur des Prototypen.

Simulation-Entities

Jede Entität in der Modellierung die für die Simulation von Interesse ist, ist eine SimulationEntity. Hierzu gehören Agenten genauso wie die Vegetation oder Kopjes. Alle SimulationEntities werden am Anfang eines neuen Ticks benachrichtigt.

Subklassen von SimulationEntity implementieren ihr Verhalten während eines Ticks in der Methode OnSimulationTick. Geparden müssen sich beispielsweise, während der Jagd, entscheiden ob sie das Anschleichen fortsetzen, oder einen Sprint zur Erlegung der Beute initiieren.
Agenten erhalten die Informationen, auf deren Basis sie ihre Entscheidungen treffen, aus den Layern. Animals können beispielsweise alle anderen Animals, die sie in diesem Tick wahrnehmen können, über den Animals-Layer beziehen.
Jede SimulationEntity hat einen eigenen Zustand, der durch Instanzvariablen implementiert ist. Alle aus dem fachlichen Datenmodell (siehe Abschnitt 2.4) bekannten Entitäten finden sich hier als SimulationEntites wieder.




Animal
Abstract Class
SimulationEntity
Fields
alive
cfgAge
cfgCamouflage
cfgDeathEffect
cfgSituationalAwareness
cfgSize
cfgSpeed
cfgSprintingSpeed
cfgStatusText
cfgUsesCover
currentDestination
Rand
status
ticksUntilArrivalAtNewQuadrant
Properties
Age
Camouflage
CurrentMaxSpeed
CurrentStatus
Hunger
IsSprinting
isTravellingToNewDestination
SituationalAwareness
Size
StatusChangedSinceLastTick
StatusText
Thirst
UsesCover
Methods
CalculateTravelTime
ChooseNextDestination
GetPossibleDestinationsFromPosition
Initialize
Kill
MoveToNewQuadrant
OnNewQuadrantAssigned
OnSimulationTick
Nested Types
Because this is only a prototype, the homerange is always circular.
HomeRange
Class
Properties
Center
Extents
Home...

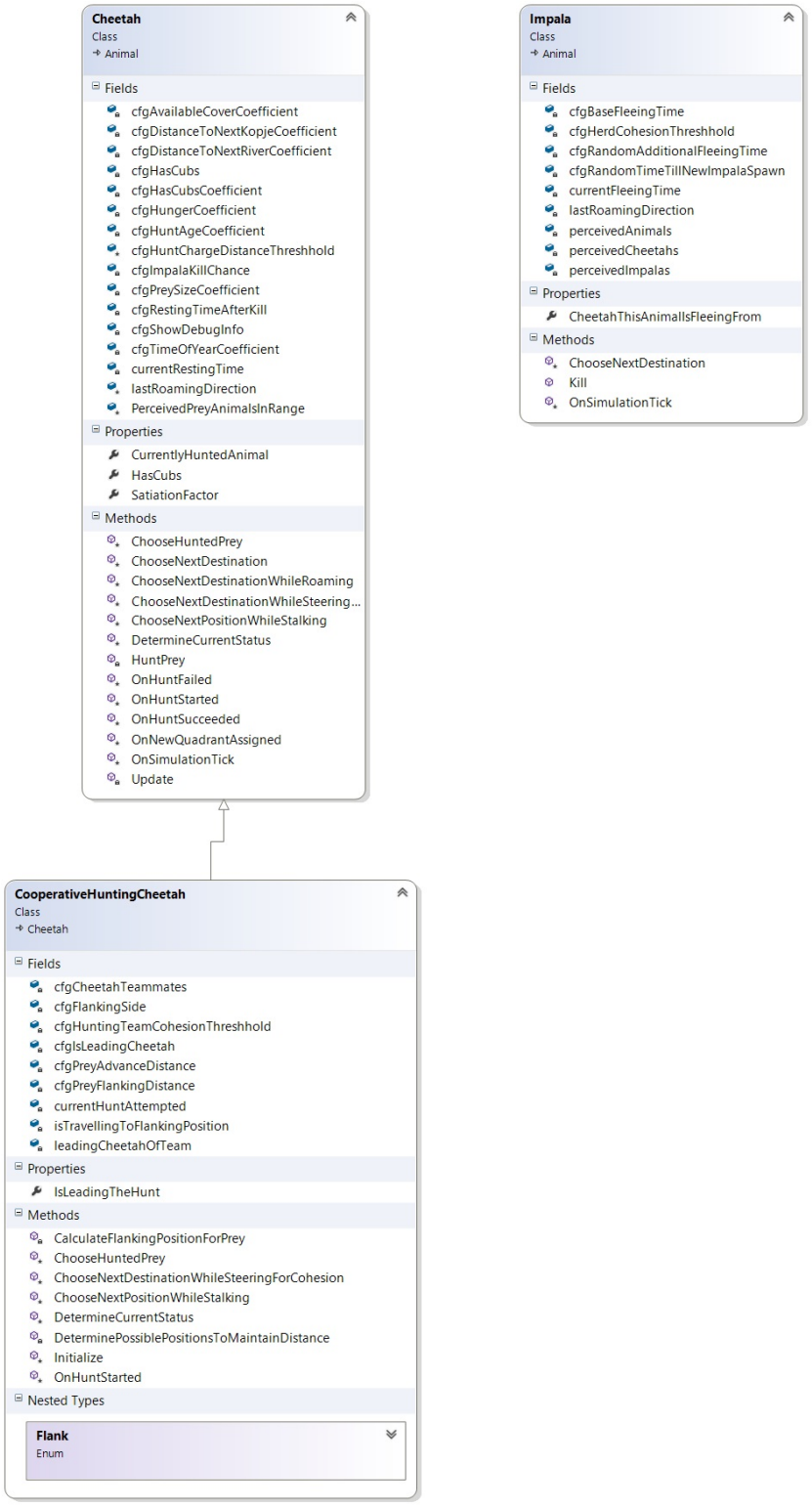

Cheetah
Class
Animal
Fields
cfgAvailableCoverCoefficient
cfgDistanceToNextKopjeCoefficient
cfgDistanceToNextRiverCoefficient
cfgHasCubs
cfgHasCubsCoefficient
cfgHungerCoefficient
cfgHuntAgeCoefficient
cfgHuntChargeDistanceThreshhold
cfgImpalaKillChance
cfgPreySizeCoefficient
cfgRestingTimeAfterKill
cfgShowDebugInfo
cfgTimeOfYearCoefficient
currentRestingTime
lastRoamingDirection
PerceivedPreyAnimalsInRange
Properties
CurrentlyHuntedAnimal
HasCubs
SatiationFactor
Methods
ChooseHuntedPrey
ChooseNextDestination
ChooseNextDestinationWhileRoaming
ChooseNextDestinationWhileSteering...
ChooseNextPositionWhileStalking
DetermineCurrentStatus
HuntPrey
OnHuntFailed
OnHuntStarted
OnHuntSucceeded
OnNewQuadrantAssigned
OnSimulationTick
Update
Impala
Class
Animal
Fields
cfgBaseFleeingTime
cfgHerdCohesionThreshhold
cfgRandomAdditionalFleeingTime
cfgRandomTimeTillNewImpalaSpawn
currentFleeingTime
lastRoamingDirection
perceivedAnimals
perceivedCheetahs
perceivedImpalas
Properties
CheetahThisAnimalIsFleeingFrom
Methods
ChooseNextDestination
Kill
OnSimulationTick
CooperativeHuntingCheetah
Class
Cheetah
Fields
cfgCheetahTeammates
cfgFlankingSide
cfgHuntingTeamCohesionThreshhold
cfgIsLeadingCheetah
cfgPreyAdvanceDistance
cfgPreyFlankingDistance
currentHuntAttempted
isTravellingToFlankingPosition
leadingCheetahOfTeam
Properties
IsLeadingTheHunt
Methods
CalculateFlankingPositionForPrey
ChooseHuntedPrey
ChooseNextDestinationWhileSteeringForCohesion
ChooseNextPositionWhileStalking
DetermineCurrentStatus
DeterminePossiblePositionsToMaintainDistance
Initialize
OnHuntStarted
Nested Types
Flank
Enum

Abbildung 4.15: SimulationEntites repräsentieren die Entitäten aus dem fachlichen Datenmodell.

Layer-Struktur

Die Umwelt der Simulation ist, wie in Abschnitt 3.1 beschrieben in Quadranten aufgeteilt. Diese Quadranten werden vom QuadrantManager verwaltet. Der QuadrantManager bietet außerdem die Möglichkeit ein neues randomisiertes, auf einer Noise-Funktion basierendes, Terrain zu erzeugen.
Die Layer teilen sich in normale und SpatialLayer auf. SpatialLayer bieten einen gefilterten Zugriff auf ihre jeweiligen SimulationEntities. Diese Filterung erfolgt entweder Aufgrund anhand der Position der SimulationEntity. Es ist beispielsweise möglich sich alle Entites in einem Umkreis um einen Punkt zu erhalten. Layer bieten Informationen über die Umwelt, die nicht auf SimulationEntites bezogen sind.

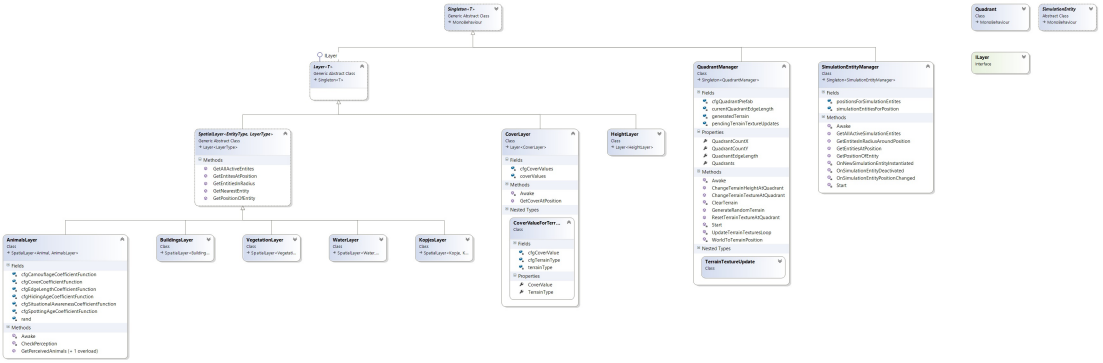


Abbildung 4.16: Implementierung der Layer im Prototypen.

Verhalten der Tiere

Geparden implementieren eine echte Untermenge der in 3.2.4 beschriebenen Verhaltensweisen. Geparden jagen alleine oder in der Gruppe nach Beutetieren, müssen im Prototypen aber weder fressen noch sich nach einer Jagd ausruhen.
Impalas implementieren das in 3.2.5 beschriebene Verhalten.

Parametrisierung des Modells

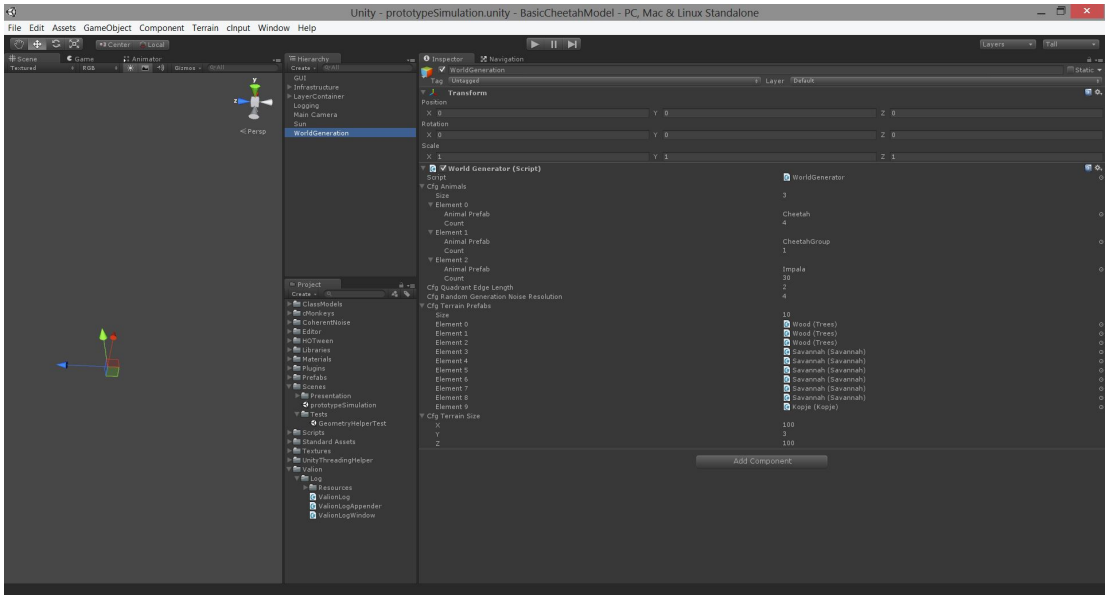


Abbildung 4.17: Generierungsoptionen der Umwelt.

Die Parametrisierung des Prototypen erfolgt über den Unity Editor. Die Klassen der Infrastruktur liegen in der Simulations-Szene auf eigenen GameObjects. Die SimulationEntites sind als Prefabs implementiert. Beide lassen sich komfortabel mit dem Unity-Inspektor konfigurieren.

5 Diskussion

5.1 Bewertung der Modellierung

Der Großteil der Modellierung hat sich als gut herausgestellt. Durch den Bau des Prototypen sind mir jedoch einige grundlegende Schwächen aufgefallen. Im folgenden werde ich die Schwächen, und mögliche Lösungsansätze, einzeln behandeln.

5.1.1 Einteilung des Geländes in Quadranten

Die Verwendung von Quadranten hat sich für die Darstellung des Terrains bewährt. Durch dieses Konzept wird die Speicherung des Simulationszustandes in einem Datacube praktikabel. Auch ermöglicht es die Verwendung von Geschwindigkeitsoptimierungen für den Zugriff auf die Daten (bspw. Sortierung der Quadranten in einem Quadtree).

Ein weiterer Vorteil ist die Trennung des Geländes von den Quadranten. Ein Quadrant dient nur als Container für das Gelände an dessen Position. Er hat kein Wissen über seinen Inhalt. So kann der Inhalt eines Quadranten beliebig komplex werden, ohne das dessen Implementierung geändert werden muss.

5.1.2 Begrenzung der Bewegung von Tieren auf Quadranten

Wie im Konzept beschrieben befinden sich Tiere immer in genau einem Quadranten. Die Quadranten-Struktur an sich hat sich für die Speicherung und Abfrage des Simulationszustandes im Datacube bewährt. Probleme gibt es bei der Darstellung der Bewegung von Tieren.
Wie in 3.2.1 beschrieben verbleiben Tiere so lange in ihrem Ausgangsquadranten, wie sie für die Bewegung zum Zielquadranten benötigen.
In der Visualisierung wird die Position der Tiere bei der Bewegung zwischen den Quadranten interpoliert.
Werden die Quadranten zu groß dimensioniert kommt es zu erheblichen Abweichungen an welcher Position ein Tier in der Visualisierung dargestellt wird, und wo es sich tatsächlich befindet.

Diese Diskrepanz zwischen Visualisierung und Simulation kann zu, durch einen Beobachter, nicht nachvollziehbaren Ergebnissen führen.
Die Granularität der Simulation der Bewegung der Tiere wird durch die Quadrantengröße bestimmt. Dadurch wird die Simulation von komplexen Bewegungsmustern, beispielsweise während der Jagd, unnötig kompliziert. Die Implementation der Simulation unterschiedlicher Geschwindigkeiten von Tieren hat sich mit der jetzigen Modellierung als sehr kompliziert und unflexibel erwiesen.

Die Position der Tiere muss also von der Position der Quadranten entkoppelt werden. Durch die Verwendung einer Physik-Engine, für die Simulation der Bewegung der Tiere, kann dies erreicht werden. Die Tiere sind dadurch nicht mehr an die Quadranten Struktur gebunden, und können sich frei im Raum bewegen.
Die Bestimmung des Geländes in dem sich ein Tier befindet kann weiterhin über die Quadrantenstruktur erfolgen.
Die Grenzen jedes Quadranten können mit Hilfe der Position und Kantenlänge des Quadranten leicht berechnet werden.
Da die Position des Tieres bekannt ist, lässt sich der Quadrant, in dem sich das Tier befindet, nun leicht feststellen.

5.1.3 Entscheidungsfindung der Agenten mit Hilfe eines endlichen Automaten

Die Implementierung der Entscheidungsfindung der Agenten mit Hilfe eines endlichen Automaten hat sich als einfach und flexibel genug erwiesen. Dies ist jedoch auf die relative Simplizität der derzeit nötigen Entscheidungen zurück zu führen.

Für die nächste Iteration plane ich Behaviour-Trees zur Modellierung der Entscheidungsfindungsprozesse zu verwenden. Mit diesen lassen sich komplexe Entscheidungsprozesse übersichtlicher, als mit endlichen Automaten, darstellen.

5.1.4 Deckungssystem

Die Modellierung der Wahrnehmung der Tiere hat sich als zu abstrakt herausgestellt.

In der derzeitigen Modellierung ist es möglich, dass sich Tiere ohne vorhandene Deckung (also auf freier Fläche) vor anderen Tieren verstecken können. Dieser Fehler lässt sich leicht dadurch korrigieren, dass die Chance der Entdeckung eines Tieres auf 100% steigt, sobald dieses keine Deckung besitzt.

Sichtlinien lassen sich mit der derzeitigen Modellierung sehr schlecht darstellen. Falls ein Tier A eigentlich durch ein Hindernis vor einem anderen Tier B verborgen ist, wird dieses nicht in die Berechnung mit einbezogen. Den höchsten Deckungswert der Quadranten zwischen den Tieren für die Deckung von Tier A zu verwenden verbessert die Situation erheblich. Dies ist aber keine zufrieden stellende Lösung.

Mit einer steigenden Komplexität der Simulation wird es nötig werden dieses System massiv zu erweitern. Beispielsweise wird es nötig werden den Tieren einen Geruchssinn zu geben. Geparden müssen dann beim An schleichen auf die Windrichtung achten, um nicht von der Beute frühzeitig bemerkt zu werden.

5.1.5 Einfluss des Geländes auf die Bewegung der Tiere

Momentan beeinflusst das Gelände in dem sich ein Tier bewegt nicht dessen Bewegungsgeschwindigkeit.

In der Modellierung ist dieses Feature vorgesehen, im Prototypen jedoch noch nicht implementiert. Dies führt dazu, dass die Simulation, in bestimmten extremen Geländekonfigurationen, nicht die erwarteten Ergebnisse liefert.

Beispielsweise wird der Jagderfolg von Geparden in stark bewaldeten Gebieten unerwartet groß sein, weil dieses Gelände eine sehr gute Deckung zum anschleichen bietet. Man würde jedoch erwarten, dass die Bäume den Geparden beim sprinten erheblich behindern sollten.
Der Einfluss des Geländes auf die Bewegung ist also von entscheidender Bedeutung. Für die nächste Iteration sollte dieses Feature aber wie folgt verfeinert werden:

Sobald eine Physik-Engine für die Bewegung der Tiere genutzt wird, wird jedes Hindernis einzeln modelliert werden. Hierdurch wird dann die Bewegungsgeschwindigkeit, ohne einen abstrakten und von der Physik-Simulation losgelösten Faktor, limitiert.

5.1.6 Nährwert der Vegetation

Beutetiere haben in der derzeitigen Modellierung keine Möglichkeit den Nährwert der Vegetation eines Quadranten zu beeinflussen. Diese Entscheidung könnte sich in der Zukunft als problematisch herausstellen.
Ein Quadrant mit der entsprechenden Vegetation ist momentan einen unerschöpfliche Nahrungsquelle. Solange der Quadrant groß genug ist, ist diese Abstraktion vertretbar. Wird ein Quadrant jedoch zu klein dimensioniert wird diese Abstraktion zu einem Problem.
Nehmen wir folgende Situation an. Ein Quadrant ist so groß dimensioniert, dass dieser genau einem Früchte tragendem Baum Platz bietet. Für die gesamte Jahreszeit, in der der Baum Früchte trägt, wird dieser konstant dieselbe Nahrungsmenge liefern. Der Vorrat an Früchten ist quasi unbegrenzt. Theoretisch könnte also eine gesamte Impala-Herde, innerhalb kurzer Zeit, von diesem einen Baum satt werden.
Das System wird vermutlich, aus diesem Grund, nicht gut auf extrem große Konzentrationen von Impalas, in einem Gebiet, reagieren. Die Erwartung wäre, dass diese die lokalen Ressourcen erschöpfen und weiterziehen müssten. Diese Ressourcen unterliegen in der Simulation jedoch effektiv keinerlei Beschränkungen. Die Impalas wären also nicht gezwungen weiter zu ziehen.
Um dieses Problem zu lösen wird die Simulation der Beutetiere und der Vegetation deutlich komplexer werden müssen. Die simulierten Beutetiere werden lernen müssen, neue Nahrungsquellen zu erschließen. Die Lebenszyklen der Vegetation müssen ebenfalls modelliert werden.

6 Fazit und Ausblick

Der Großteil der Modellierungsentscheidungen hat sich im Test als gut herausgestellt. Schwächen konnten durch den Prototypen schnell erkannt und beseitigt werden. Der Bau eines Prototypen hat sich sehr gelohnt. Kritische Systembestandteile werde ich auch in Zukunft zuerst prototypisch umsetzen, um Fehler schnell zu erkennen.
Die nächste Iteration der Simulation wird in WALK implementiert werden. Diese Version wird hauptsächlich die Künstliche Intelligenz und die Performance verbessern.

Ich bin sehr zufrieden mit dem bisherigen Verlauf des Projekts. Es ist mir gelungen Biologen die Vision des Projekts, durch den Prototypen, nahe zu bringen. Das Interesse ist geweckt und das Feedback ist äußert positiv. Ich freue mich auf die Weiterentwicklung des Prototypen im Master-Studium.

Literaturverzeichnis

Broomhall, & Lynne, Susan. 2002. *Cheetah Acinonyx jubatus ecology in the Kruger National Park : a comparison with other studies across the grassland-woodland gradient in African savannas.* Ph.D. thesis, University of Pretoria.

Cheetah.org. 2013. *Cheetah Conservation Fund.*

Cooper, Andrew B, Pettorelli, Nathalie, & Durant, Sarah M. 2007. Large carnivore menus: factors affecting hunting decisions by cheetahs in the Serengeti. *Animal Behaviour*, **73**(4), 651–659.

Durant, S., Marker, L., Purchase, N., Belbachir, F., Hunter, L., & Packer, C. BreitenmoserWursten, C. Sogbohossou. 2013. *Acinonyx jubatus (Cheetah, Hunting Leopard).*

Durant, S. M. 1998. Living with the enemy: avoidance of hyenas and lions by cheetahs in the Serengeti. *Behavioral Ecology*, **11**(6), 624–632.

Durant, S.M., Caro, T.M., Collins, D.A., ALAWI, R.M., & Fitzgibbon, C.D. 1988. Migration patterns of Thomson's gazelles and cheetahs on the Serengeti Plains. *Afr. J. Ecol.*, **26**, 257–268.

Hilborn, A, Pettorelli, N, Orme, C D L, & Durant, S M. 2012. Stalk and chase: How hunt stages affect hunting success in Serengeti cheetah. *Animal Behaviour*, **84**(3), 701–706.

Houser, Annmarie. 2008. *Spoor density , movement and rehabilitation of cheetahs in Botswana By.* Ph.D. thesis, University of Pretoria.

Shah, M A N O J. 2013. catch & release. *Africa Geographic*, **18**, 20.

Thiel-clemen, Th. 2013. Information Integration in Ecological Informatics and Modelling.

Wikipedia. 2013. *Gepard – Wikipedia.*

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

Hamburg, 25. September 2013 Malte Eckhoff