Master-Thesis

zur Erlangung des Grades eines

M.Sc. Wirtschaftsinformatik

Segregation bei komplexen Migrationstrategien

Sebastian Geiermann
Matrikelnummer 209211008
sgeiermann@uni-koblenz.de

UNIVERSITÄT KOBLENZ · LANDAU

Kurzfassung des Dokuments

Diese Masterarbeit beschreibt und evaluiert die im Rahmen der Arbeit entwickelten Erweiterungen zur Schellings Segregationsimulation.

Der Nobelpreisträger Thomas C. Schelling simulierte als erster ab 1969 die Segregation. Die dabei entstandenen Ergebnisse erlaubten eine genauere Analyse der auftretenden Phänomene. Uri Wilensky, Mitarbeiter der „Northwestern University", entwickelte seit 1999 eine Segregationssimulation in NetLogo. Im Rahmen meiner Masterarbeit habe ich das NetLogo-Model von Uri Wilensky in sechs unterschiedlichen Szenarien weiterentwickelt:

Im ersten Szenario „**Basisszenario oder betterworld**" wird das NetLogo-Modell um eine nicht vorhandene, aber von Schelling geforderte Eigenschaft erweitert. Mittels dieser Erweiterung segregieren nahezu alle Simulationen schneller.

Im zweiten Szenario „**Multikulturalismus**" wird die Anzahl der zur Simulation gehörenden Klassen auf bis zu zehn erweitert. Schelling sowie Wilensky simulierten ausschließlich mit zwei unterscheidbaren Klassen.

Im dritten Szenario „**Integrationsbeauftragter**" wird der Umzug der Turtle durch spezialisierte Turtle koordiniert. Dadurch kann die Segregation aufgehalten werden.

Im vierten Szenario kommen „**Intelligente Patches**" zum Einsatz. Hierbei wird ein Teil der künstlichen Intelligenz auf die Patches transformiert. Diese übernehmen schließlich integrations- sowie segregationsfördernde Koordinationsaufgaben.

Im fünften Szenario „**Ökonomische Variation**" verfügt die Simulation über eine zusätzliche monetäre Umzugsrestriktion. Dadurch wird die Anzahl der zur Wahl stehenden freien Plätze variiert.

Im sechsten Szenario „**Populationsaustausch**" wird zur Laufzeit die

Population durch eine dritte Art inkrementell ersetzt. Dies lässt eine Vielzahl von Evaluierungsmöglichkeiten zu.

Alle Szenarien wurden von mir ausführlich dokumentiert und evaluiert.

Abstract

This thesis describes and evaluates my extensions to Schellings Segregation Model.

Nobel prize winner Thomas C. Schelling started to simulate segregation in 1969. His results allowed specific analysis of this occurring phenomena. Since 1999, Uri Wilensky has been developing similar simulations using NetLogo. In this thesis I describe six different scenarios:

In the first scenario, Wilensky's extentions of the simulation caused faster segregation in almost all instances..

In the second scenario, the number of different populations used in the simulation is increased to ten, while the original scenario used by Schelling and Wilensky used only two.

In the third scenario, some of the normal “turtles,” or agents, are promoted into leaders and allowed to coordinate the movements of the remaining population.

In the fourth scenario, the turtles transfer some artificial intelligence to the patches. With this information, the patches can also coordinate the movements of the remaining population.

In the fifth scenario, a rudimentary economical environment is introduced.

In the sixth scenario, several turtles are replaced with a new, distinct kind of turtle.

All scenarios were documented in detail and evaluated.

Inhaltsverzeichnis

1 Einleitung

Im Wintersemester 2010/2011 besuchte ich die Veranstaltung „Simulation und Agenten-basierte Systeme“ von Professor Troitzsch an der Universität Koblenz. Die Veranstaltung lehrte grundlegende und fortgeschrittene Simulationsthemen. Die in der Agenten-basierten Simulation implementierte kollektive Intelligenz und deren Möglichkeiten weckten in mir den Wunsch, auch selbst mit diesen zu arbeiten.

Die zur Veranstaltung gehörende Prüfungsleistung war die Ausarbeitung einer Simulation und deren Weiterentwicklung. Die Wahl der Agenten-basierten Simulation aus NetLogo zu Schellings Modell war schnell getroffen. Die in NetLogo vorhandene Adaption des Schellingschen-Modells ist eine gute Demonstration der Agenten-basierten Simulation. Meine erstellte Erweiterung beruhte auf dem Ausbau der vorhandenen Intelligenz zur Bestimmung von neuen Plätzen. Das Ergebnis dieser Erweiterung ist eine starke Beschleunigung der Segregation. Nach der Präsentation meines Themas legte mir Professor Troitzsch nahe, weiterhin an der Thematik zu arbeiten.

Folglich fokussiere ich in der vorliegenden Arbeit die Fortführung dieser Erweiterung auf vielfältige Weise. Das Ziel meiner Arbeit ist es nicht, die Segregation zu verhindern, sondern meine Ideen experimentell zu erforschen und zu dokumentieren.

Zu Beginn meiner Thesis gebe ich eine **Einführung** zu den bedeutsamen Begrifflichkeiten. Im Verlauf der Arbeit beschreibe ich mein **Basismodell,** welches in den Folgekapiteln unterschiedlich verändert wird. Diese Kapitel enthalten jeweils die Beschreibung und die Evaluation meiner **Erweiterungen**. Abschließend stelle ich in einer **Zusammenfassung** meine Erkenntnisse übersichtlich dar und ziehe ein **Fazit**. Mit dem **Ausblick** nenne ich weitere Entwicklungsmöglichkeiten

2 Praeambulum

Dieses Praeambulum vermittelt dem Leser die Grundlagen der Arbeit.

2.1 Segregation

Segregation ist ein auftretendes Phänomen, welches entsteht, wenn mindestens zwei heterophobe[1] Klassen aus mehreren homophilen[2] Individuen (o.a. Klassen von Individuen) in einem begrenzten Raum existieren. Das Bestreben, das die Individuen dabei verfolgen, führt zu Aufspaltung der Gesamtgemeinschaft und damit zur Segregation.

Der Difu-Bericht[3] von 2006 definiert:

> ***„Segregation ist nichts anderes als eine räumliche Abbildung sozialer Ungleichheit in einer Gesellschaft. Alle Bewohner einer Stadt kennen das Phänomen, dass sich soziale Gruppen unterschiedlich auf Wohnstandorte verteilen. Die Qualität des Wohnstandortes korrespondiert häufig mit dem sozialen Status der Gruppe." [Difu]***

Der Bericht bezieht sich dabei stark auf den Aspekt der sozialen Ungleichheit, jedoch muss Segregation nicht ausnahmslos im Kontext von sozialen Ungleichheiten stehen. Kultur und Religion bieten ebenso polarisationsförderliche Themen und vermögen dadurch als Segregationstreiber zu agieren.

Dr. Klaus Peter Strohmeier unterscheidet folgende Segregationen in seinem Gutachten „Segregation in den Städten" [Strohmeier]

- demographische Segregation zwischen Familien und Singles
- soziale Segregation zwischen Armen und Reichen
- ethnische / religiöse Segregation zwischen Migranten und Einheimischen

1 Im Sinne von: Abneigung gegenüber unterscheidbare Andere: Polarisierend und segregierend
2 Im Sinne von: Liebend, Suchend der unterscheidbaren Anderen: Integrierend
3 DIFU = Deutsches Institut für Urbanistik

und schreibt weiter:

> ***„In allen großen Städten gibt es Segregation, sie ist ein Merkmal städtischen Lebens. Heterogenität von Lebenslagen, Lebensformen und Lebensstilen der Bevölkerung ist ein Merkmal des Städtischen, und in Städten finden wir erhebliche sozialräumliche Disparitäten im Hinblick auf diese Merkmale.“ [Strohmeier] (Seite 13)***

Folglich ist Segregation allgegenwärtig und somit Bestandteil der Städte. Strohmeier fasst im weiteren Verlauf zusammen, dass Segregation grundsätzlich kein Problem darstellt:

> ***„Ethnische Segregation allein z.B. halten die wenigsten für ein Problem. Ethnisch segregierte Quartiere werden dann als problematisch bewertet, wenn gleichzeitig auch ein hohes Maß an sozialer Segregation zu beobachten ist.“ [Strohmeier](Seite 34)***

Daher lässt sich Segregation in zwei abweichenden Formen beobachten:

Zum einen die freiwillige, gewollte Segregation. (aktive Segregation) Diese zeichnet sich dadurch aus, dass der Zug der Bewohner ungezwungen ist. Die dadurch entstehenden Wohnungsviertel verfügen zumeist über gute soziale Netzwerke sowie ausgeprägte Solidarität unter Gleichen. [Difu]
Erscheinungsformen dieser Art sind sogar gewünscht und bilden:

- Studentenviertel
- Familienviertel
- Künstlerviertel
- Migrantenviertel

Die nicht-freiwillige Segregation (passive Segregation) ist geprägt durch:

- Chancenungleichheit
- Ausgrenzung
- Ghettoisierung
- Diskriminierung

Diese Form der Segregation ist nicht freiwillig, sondern durch nicht ausreichende Wahlmöglichkeiten (z.B. Wohnungswahl) bedingt. [Difu]

Bertram unterstreicht in „Ethnische Segregation in Deutschland – freiwillig oder erzwungen?“ den negativen Aspekt als:

> ***„[...] dauerhaftem Scheitern der Systemintegration“. [Bertram]***

Sie verweist damit auf die primäre Interpretation der Bevölkerung, die besagt, dass Segregation grundsätzlich schlecht ist und somit gemieden werden soll.

Der Begriff Segregation steht simultan für den Prozess der Entmischung, Polarisation und Bildung von homophilen Nachbarschaften sowie deren Ergebnis. Segregation ist somit der Prozess und das zugehörige Ergebnis, welches bedingt durch unterschiedliche Indikatoren dazu führt, dass sich eine Gesamtbevölkerung aufteilt.

Eine nicht freiwillige und somit negative Segregation sollte im Sinne einer sozialen Gemeinschaft vermieden werden.

2.2 Schellings Grundmodell

Thomas C. Schelling entwickelte um 1970 ein Modell, mit dem die Segregation besser verstanden werden konnte. Er verwendete ein Schachbrett mit zwei unterschiedlichen Münzarten: Die anfänglich zufällig verteilten Münzen wurden infolge, analog der Präferenz zur eigenen Art, nacheinander entmischt[4]. Dabei wurden nur Münzen verschoben, die mehr als das Doppelte an „nicht-Gleichen" in ihrer Nachbarschaft hatten. Nach einigen iterativen Durchläufen wurde die Segregation ersichtlich. Aus diesem Versuchsaufbau ließ sich folgende Hauptaussage formulieren: [Schelling, 1969]

„Der einzelne Wunsch eines Individuums wirkt sich erheblich auf die Gesamtzufriedenheit auf."

Die Einzelwerte der Individuen erzeugen signifikant höhere Werte in der Gesamtheit. Eine Ähnlichkeitsrelation von zwei Eigenen auf einen Fremden bewirkt eine gesamte Ähnlichkeit von rund 60%. Dieses Phänomen reicht aus, damit eine Gesellschaft ohne ausgeprägten Rassismus in der Gesamtheit segregiert.

Das Schachbrett und die Münzen werden heute durch moderne Computer abgelöst. Simulationsmodelle ersetzen die Münzen durch sogenannte Agenten (o. a. Turtle). Diese Agenten besitzen ein gewisses Maß an künstlicher „Intelligenz" (Vgl.: 2.3) und können definierte Entscheidungen selbst determinieren.

4 o.a. sortiert

Die händische Simulation ist im Verhältnis zu den modernen Computersimulationen sehr zeit- und arbeitsintensiv. Die heutige Technologie erlaubt es, die gewünschte Simulation innerhalb kürzester Zeit in einer Vielzahl unterschiedlicher Parametrisierungen durchzuführen.

2.3 Simulation

Die in dieser Arbeit eingesetzte Methodik der Computer-Simulation[5] ist ein noch junges Forschungsgebiet. Die Anfänge der Simulation sind nur schwierig zu determinieren. Eine Simulation ist im Grunde nicht mehr als die computergestützte programmatische Lösung einer Aufgabe, die bedingt durch den Umfang kaum händisch zu lösen wäre. Dabei wird nicht versucht, die gänzliche Komplexität der Realität nachzubilden. Simulation nutzt Modelle, die in Umfang und Komplexität reduziert sind. [Gilbert, Troitzsch] (Seite 1 ff.)

Simulationen erfordern keine grundlegenden neu-entwickelten Programme oder Verfahren, sondern können mit vorhandener Software[6] erfolgen. Mit dem Voranschreiten der Leistungsfähigkeit der ersten PC'S in den 60er Jahren des vergangenen Jahrhunderts entwickelten sich auch die ersten Simulationssprachen. In den 80er Jahren wurden die Ergebnisse erstmals optisch ansprechend dargestellt.

Die in diesem Fall genutzte Agenten-basierte Simulation (ABS)[7] grenzt sich dahingehend von vielen Simulation ab, so dass ein Teil der Entscheidungs- und Handlungsmöglichkeit auf einen Teil der Programmierung, den Agenten, übergeht. Ein Agent kann aus einer Menge gegebener Alternativen selbstständig entscheiden und somit den Fortgang der Simulation aktiv beeinflussen.

Das zur ABS gehörige Forschungsgebiet der künstlichen Intelligenz (KI) bzw. der verteilten KI ist eine relativ junge Entwicklung.

5 Anmerkung: Im Folgenden bezeichne ich Computer-Simulation nur noch als Simulation
6 Tabellenkalkulation, bzw. einfach zu erlernende Programmiersprachen
7 Anmerkung: Im Folgenden bezeichne ich Agenten-basierte Simulation nur noch als ABS

Die **Vorteile** der ABS können in drei Aussagen erfolgen: [Bonabeau]

- ABS modelliert eine natürliche Beschreibung des Systems:

  Die Formulierung des Systems erfolgt in einer realitätsnahen Formulierung und ist somit mühelos adaptierbar. Zum Beispiel ist die Beschreibung „Wie verhält sich ein Fluggast im Flughafen" verständlicher als eine komplexe mathematische Gleichung, die dieses zu berechnen versucht.

- ABS ist flexibel

  ABS können vielfältig verändert werden, z.B. die Anzahl der Agenten, die Größe der Simulation, die intrinsische sowie extrinsische Interaktion, die Hierarchien und die Kommunikation.

- ABS kann emergente Phänomene nachbilden.

2.3.1 Emergente Phänomene[8]

„Das Ganze ist mehr als die Summe seiner Teile." [Aristoteles]

Von dem lateinischen Verb „emergere" für „auftreten" „sich herausarbeiten" abgeleitetes Wort. Emergenz bezeichnet das Charakteristikum von der Erscheinung neuer Eigenschaften, die aus der ursprünglichen Information nicht abgeleitet bzw. erklärt werden kann. [Stein]

Beispiel: Treten Soldaten im Gleichschritt über eine Brücke, kann es dazu kommen, dass diese einstürzt. Der Einsturz ist in diesem Fall nicht auf das Gewicht aller Soldaten zurückzuführen, sondern auf das Resultat der Resonanz[9]. Das gleichmäßige Treten der Soldaten erzeugt eine gesteigerte Amplitude, welche den Einsturz herbeiführen kann.

8 o.a.: Fulguration http://de.wikipedia.org/wiki/Fulguration

9 Die Minimierung des Deltas zwischen Erreger- und Erzeugeramplitude führt dazu, dass die Amplitude des angeregten Systems stark steigt.

Das emergente Phänomen ist somit der Einsturz, da dieser aus der Ausgangslage nicht direkt abgeleitet werden kann. ABS ermöglicht ähnliche Erscheinungen in Simulationen nachzubilden.

2.4 NetLogo

NetLogo ist eine Entwicklungsumgebung für ABS. Uri Wilensky, Direktor des *„Center for Connected Learning (CCL) and Computer-Based Modeling" an der „Northwestern University",* entwickelt und verbessert seit 1999 das LOGO[10]-Derivat. NetLogo bietet unerfahrenen Entwicklern einen mühelosen Einstieg und eröffnet gleichzeitig Experten fortgeschrittene Modellierungsmöglichkeiten.

Die für diese Arbeit genutzte Version ist NetLogo 5.0beta3. Die Auswahl dieser Version erfolgte primär aus dem Grund, dass ab Version 5.0 die eigentliche Programmierlogik von dem, für die Aufbereitung der Ausgabe verantwortlichen Teil getrennt wurde. Diese Trennung fördert die Verständlichkeit, was meiner Meinung nach sehr sinnvoll ist.

3 Ausgangsmodell aus NetLogo

Grundlage für die folgende Beschreibung ist das in NetLogo 5.0b3 vorhandene Segregationmodell von Uri Wilensky.

Uri Wilensky beschreibt den Aufbau im Original so:

> ***„This project models the behavior of two types of turtles in a mythical pond. The red turtles and green turtles get along with one another. But each turtle wants to make sure that it lives near some of "its own." That is, each red turtle wants to live near at least some red turtles, and each green turtle wants to live near at least some green turtles. The simulation shows how these individual preferences ripple through the pond, leading to large-scale patterns. This project was inspired by Thomas Schelling's writings about social systems (such as housing patterns in cities)."[Wilensky 1997]***

10 Einfach zu erlernende, funktionale Programmiersprache

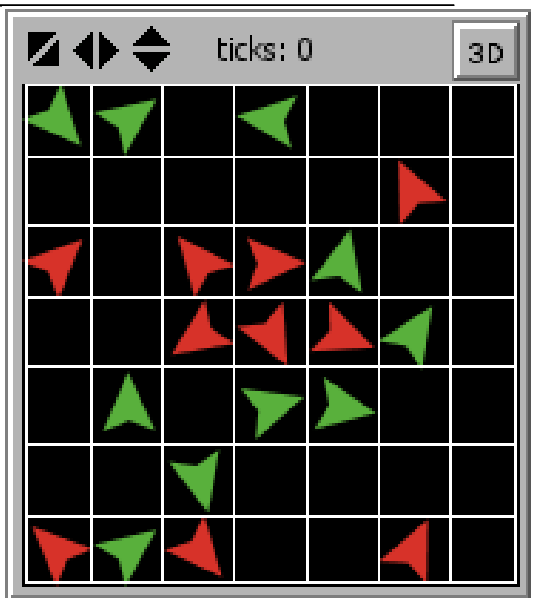

Abbildung 1: Modellwelt

Wilenskys Modell (siehe: Abbildung 1) beinhaltet zwei verschiedene Agenten o.a. Turtle, in Form von roten und grünen Pfeilen. Diese verkörpern unterschiedliche Rassen bzw. Arten von Akteuren. Die Akteure agieren in einem geschlossenen System miteinander. Ziel der Simulation ist die Zufriedenstellung alle Turtle. Jeder Turtle maximiert seine Zufriedenheit dadurch, dass er die Anzahl **nicht gleichrassiger** Nachbarn bis zu einem Schwellenwert meidet.

Für die Bestimmung der Nachbarn werden alle direkt anliegenden Felder, o.a. Patches, (siehe: Abbildung 2) betrachtet. Liegt die Anzahl der fremdem Turtle oberhalb eines definierten Schwellenwertes, wird der Turtle unglücklich (siehe: Quellcodeauflistung 1) und versucht umzuziehen.

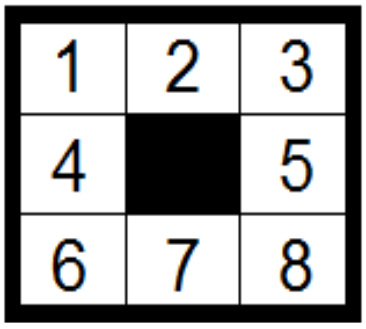

Abbildung 2: Nachbarfelder

```
1   set similar-nearby count (turtles-on neighbors)
2   with [color = [color] of myself]
3
4   set other-nearby count (turtles-on neighbors)
5   with [color != [color] of myself]
6
7   set total-nearby similar-nearby + other-nearby
8   set happy? similar-nearby >= ( %-similar-wanted *
    total-nearby / 100 )
```

Quellcodeauflistung 1: Bestimmung unglücklicher Turtle

Der Umzug kann jedoch nur durchgeführt werden, wenn genügend alternative freie Plätze vorhanden sind. (siehe Quellcodeauflistung 2)

```
1   to find-new-spot
2     rt random-float 360
3     fd random-float 10
4     if any? other turtles-here
5       [ find-new-spot ]
6     move-to patch-here
7   end
```

Quellcodeauflistung 2: Bestimmung eines neuen Patches

Der Umzug erfolgt zufällig. Der Turtle bestimmt wahllos einen Winkel und eine Entfernung. Ist das gewählte Feld belegt, muss die Prozedur wiederum durchlaufen werden. (siehe: Quellcodeauflistung 2)

Die Simulation endet, sobald alle Turtle zufrieden sind. Dies bedeutet, dass die Nachbarschaftsverteilung jedes Turtles unterhalb des Schwellenwerts bleibt. Ist jedoch mindestens ein Turtle weiterhin unglücklich, muss die Simulation fortgeführt werden. Dies kann mitunter dazuführen, dass die Simulation niemals endet. Abbildung 3 visualisiert den programmatischen Ablauf.

Die von *Uri Wilensky* erstellte Simulation ist eine Anlehnung an Thomas Schellings Modell. Festhalten lässt sich jedoch, dass ein neu gewähltes Feld rein zufällig bestimmt wird. Es ist damit nicht klar, ob das neue Feld eine bessere oder schlechtere „Nachbarschaft“ bietet. Ferner ist die Bestimmung eines neuen Ortes ineffizient.

In diesem Punkt unterscheidet sich das NetLogo-Modell von Schellings Überlegungen. Schellings Akteure ziehen nicht zufällig um. Sie bestimmen den neuen Aufenthaltsort durch Identifizierung einer neuen „geeigneteren“ Nachbarschaft und werden erst danach aktiv.

3.1 Flussdiagramm

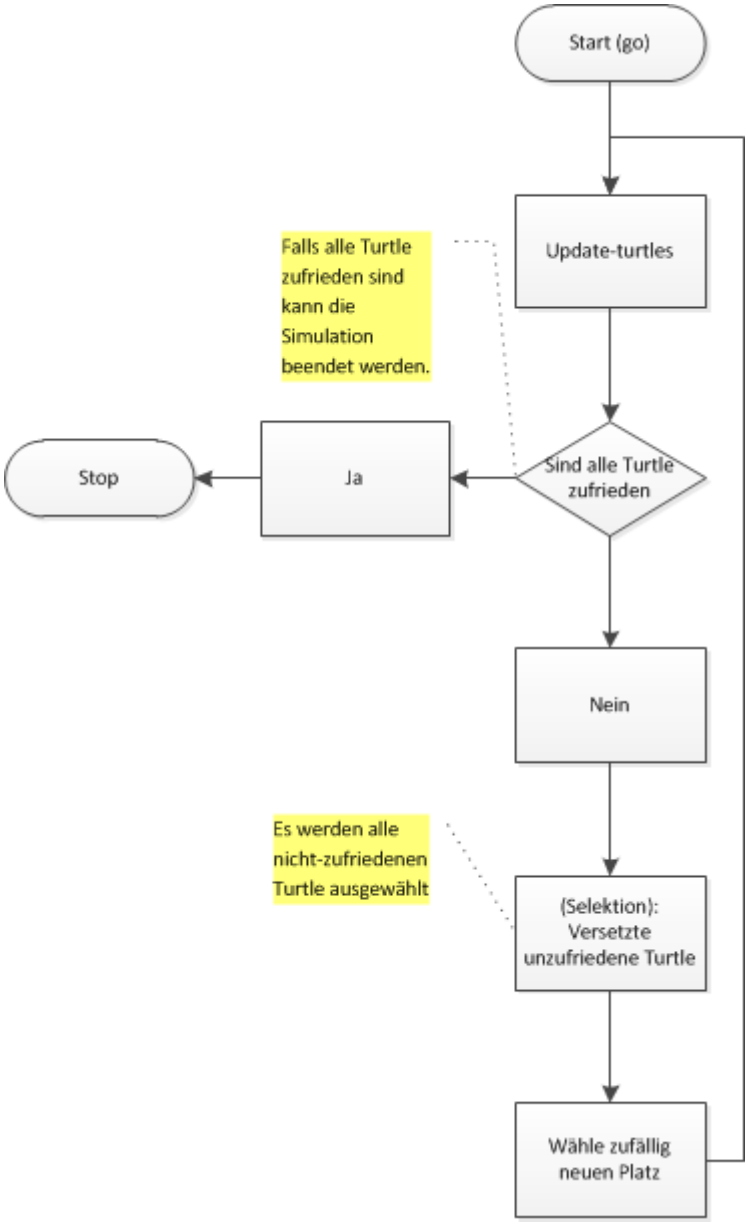

Abbildung 3: Flussdiagramm NetLogo-Original

4 Erweiterungen des Modells

4.1 Szenario 1 - Basismodell

Wie in Kapitel 5 beschrieben erfolgt der Umzug eines Turtles akzidentiell, d.h. es ist nicht sichergestellt, dass der neugewählte Platz die Zufriedenheit des Turtles garantiert. Dies führt mitunter zu langen Simulationsläufen. Die Anzahl der Simulationsschritte steigt mit den Versuchen, die ein Turtle benötigt, um einen „besseren" Platz zu finden. Bedingt durch die Zufallsauswahl unterscheiden sich die Ergebnisse der Simulation stark.

Die implementierte Platzfindung führt nur durch eine große Anzahl von zufälligen Ereignissen zur Segregation. Die eigentliche Intention der Turtle „finde einen besseren Platz" wird nur über Umwege erreicht.

Schelling wählt fortlaufend Plätze, die eine höhere Zufriedenheit garantieren.

> ***„If fewer than half are his color, he moves in either direction to the nearest point (measured in the number he passes on the way) at which half his eight nearest neighbors are the same color as he." [Schelling, 1969](Seite 4)***

Daher muss ein neuer Platz den Turtle zufriedenstellen.
Grundlage der späteren Szenarien wird das nun vorgestellte Basismodell sein. Das Basismodell wird als Option ***betterworld*** als Funktion bieten. ***betterworld*** versucht für alle Turtle die Zufriedenheit nach einem Umzug sicher zu stellen. (Siehe: 4.1.3)
Ebenso werden die in Tabelle 1 zusammengefassten Änderungen der Eingabeparameter zur Basissimulation gehören.

Parameter	*NetLogo-Original*	*Basismodell*
***Anzahl** der vorhandenen Akteure (**Turtle**) auf dem Spielfeld*	***Number*** *absolute Anzahl an Turtle auf dem Feld. Entspricht der absolute Bevölkerungsdichte* *Eine Absolutzahl steht im Konflikt mit einem variablen Simulationsfeld*	***population density (V)*** *Bevölkerungsdichte ist eine prozentuale Angabe. Sie definiert die freien Plätze relativ zum Simulationsfeld. Die absolute Zahl berechnet sich entsprechend. Siehe Formel 2*
Zufriedenheitsvariable *Ab einer definierten Anzahl von gleichen Nachbarn ist der Turtle zufrieden.*	***%-similar-wanted*** *Falls die Anzahl der Gleichen in Relation zu allen anderen Turteln in der Nachbarschaft größer ist als %-similar-wanted, ist der Turtle zufrieden. (siehe: Formel 1)* *Die Berechnung erlaubt eine Vielzahl von redundanten Einstellmöglichkeiten.*	***comfortability threshold (T)*** *Der comfortability threshold ist eine natürliche Zahl und ist damit direkt interpretierbar. Die Zahl gibt an, wie viele Turtle mindestens vorhanden sein müssen, damit der Turtle zufrieden wird.*
Abbruchkriterium	***Zufriedenheit aller Turtle*** *Die Simulation endet nur dann, wenn alle Turtle zufrieden sind.*	***Zufriedenheit aller Turtle*** *Die Erweiterungen der Simulation bedingen neue Stop-Kriterien. Daher verwenden einzelne Szenarien unterschiedliche Stopgrenzen, z.B. eine gewisse Anzahl unglücklicher wird toleriert, nach n-Ticks endet die Simulation.*

Tabelle 1: Vergleich: NetLogo original – Basismodell

4.1.1 Zufriedenheitsdeterminierung (Exkurs)

Abhinav Singh, Dmitri Vainchtei und **Howard Weiss** nutzen in ihrem Artikel *„Schelling's Segregation Model: Parameters, Scaling and Aggregation"* zur NetLogo-Simulation unterschiedliche Parameter.
Tabelle 2 zeigt die unterschiedliche Definition und die zugehörige Normierung. Die Tabelle macht deutlich, dass diese Parameter besser geeignet sind. Aus diesem Grund habe ich sie auch in mein Basismodell integriert.

$$zufrieden = \begin{cases} \text{true, falls } SN \geq \left(\frac{\%\text{-similar-wanted} * TN}{100}\right) \\ \text{false, falls } SN < \left(\frac{\%\text{-similar-wanted} * TN}{100}\right) \end{cases}$$
$$mit\ SN = \text{Anzahl gleicher Turtle i. d. Nachbarschaft}$$
$$mit\ TN = \text{Anzahl nicht-gleicher Turtle i. d. Nachbarschaft}$$

Formel 1: Zufriedenheitsdeterminierungen im ursprünglichen NetLogo-Modell

$$v = \frac{V}{N^2}$$
$$mit: V = freien\ Plätze$$

Formel 2: Bevölkerungsdichte Basismodell

Die Bestimmung der Zufriedenheit steht in Abhängigkeit von der Beurteilung der freien Plätze. Die unterscheidbaren Interpretationen habe ich in Tabelle 2 als Zufriedenheitsdeterminierungen festgehalten.

sympathetic	***unsympathetic***	***indifferent***
Die benachbarten freien Plätze sind dem Turtle sympathisch, d.h. für die Berechnung sind gleiche Nachbarn sowie freie Felder kombiniert positiv zu betrachten.	*Die benachbarten freien Plätze sind dem Turtle unsympathisch, d.h. für die Berechnung sind nur gleiche Nachbarfelder relevant.*	*Der Turtle ist der Entscheidung sympathisch oder unsympathisch, gegenüber indifferent, d.h. der Turtle relativiert die freien Plätze und bezieht diese in die Berechnung nicht mit ein.*
$SN + FN \geq CT$ *mit* : $SN =$ similar-nearby $FN =$ free-nearby $CT =$ comfortability_threshold	$SN \geq CT$ *mit* : $SN =$ similar-nearby $CT =$ comfortability_threshold	$(\frac{SN}{8}) \geq (\frac{CT}{8}) * (\frac{TN}{8})$ *mit* : $SN =$ similar-nearby $TN =$ total-nearby $CT =$ comfortability_threshold
Formel 3: sympathische, freie, Nachbarfelder	*Formel 4: unsympathische, freie Nachbarfelder*	*Formel 5: indifferenter Turtle*

Tabelle 2 Übersicht – Zufriedenheitsbestimmung

4.1.2 Segregation Messungen (Exkurs)

Vielen Menschen erkennen die Segregation unaufgefordert als Trennung der Gesamtheit. Abbildung 4 zeigt ein gutes Beispiel für die komplette Trennung der beiden Individuen.

Abbildung 4: Segregationsbeispiel

Um die visualisierten Ergebnisse auch mathematisch nutzen zu können, bedarf es einer Messung der Segregation. Zur Messung können verschiedene Verfahren angewandt werden.

Schelling nutzte zwei voneinander verschiedene statistische Maßzahlen, um die Segregation zu messen:

- Der Quotient von gleichen zu ungleichen Nachbarn bezogen auf die Nachbarschaft über die Gesamtheit aller Turtle. (Percent Similar)
- Die Anzahl der Turtle, die ausschließlich über gleiche Nachbarn auf ihren Nachbarfelder verfügen. (Completely Surrounded)

In meiner Simulation protokolliere ich, sofern für die Auswertung relevant, beide Zahlen fortlaufend und stelle sie übersichtlich in einem Diagramm dar. (Siehe Abbildung 4)

Des Weiteren messen und dokumentiere ich die Anzahl der unzufriedenen Turtle. (Percent Unhappy)

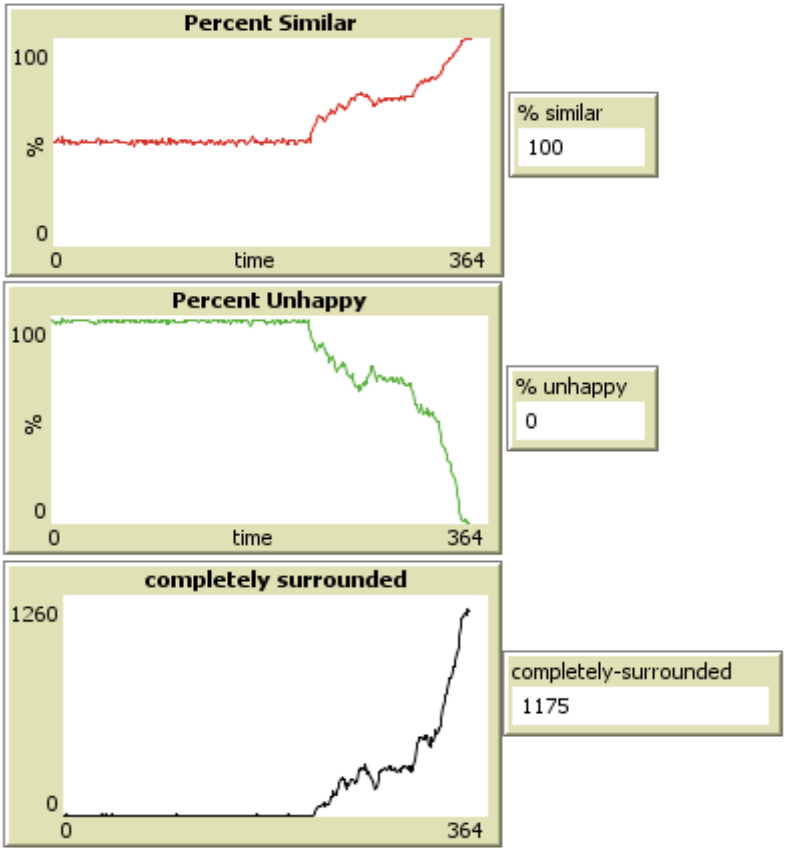

Abbildung 5: Segregationsbeispiel - Diagramme

4.1.3 Konzeption

Die verbesserte Platzfindung ist in der Simulation als Option wählbar. (Siehe: Tabelle 18) Abbildung 7 zeigt den stark vereinfachten programmatischen Ablauf des Basismodells. Nach der Entscheidung, ob die Erweiterung ***betterworld*** angewandt wird oder nicht, entstehen zwei unterschiedliche Abläufe:

Falls die Option nicht gewählt wird, verhält sich diese Simulation identisch zur ursprünglichen Simulation der NetLogo-Bibliothek: Alle unzufriedenen Turtle suchen zufällig einen neuen Platz.

Sofern die Option gewählt wird, versucht der Turtle einen Platz zu finden, auf dem er zufrieden ist. Ist kein Platz vorhanden bzw. kann kein geeigneter in einer bestimmen Anzahl von Durchläufen gefunden werden, entscheidet der Turtle, ob er seinen alten Platz einnimmt oder zufällig einen neuen wählt.

4.1.4 Implementation

Hinzufügte Elemente

Bezeichnung	***Typ***	***Erläuterung***
Eingabe		
population_density	*Slider (%)*	*Prozentuale Angabe über die mit Turtles bevölkerten Felder zum Beginn der Simulation.*
comfortability_threshold	*Slider*	*Anzahl der zur Zufriedenheit benötigten „korrekt" besetzten Nachbarfelder. Muss in Kombination mit free-spaces-happy betrachtet werden.*
free-spaces-happy	*Chooser*	*Interpretationensauswahl für freie Plätze.*
random-seeds	*Put*	*Eine Eingabe ungleich 9999 setzt den gewählten Wert als Startwert für die Bestimmung von Zufallswerten ein.*
Betterworld?	*Switch*	*Aktiviert oder deaktiviert die Funktionalität betterworld.*
rount_to_find_bw%	*Slider (%)*	*Prozentuale Angabe der maximal zu prüfenden Plätze. Die Angabe bezieht sich auf die in der Simulation befindlichen freien Plätze.*
stay_leave	*Slider*	*Auswahl, ob der Turtle nach dem Erreichen der maximalen Wiederholungsrunden den aktuellen Platz oder den ursprünglichen Platz wählt.*
Ausgabe		
Percent Similar	*Plot*	*Diagramm zum Verlauf der „Percent Similar"*
Percent Unhappy	*Plot*	*Diagramm zum Verlauf der „Percent Unhappy"*
Completely_surrounded	*Plot*	*Diagramm zum Verlauf der „completely surrounded"*
Number	*Monitor*	*Anzahl aller Turtle innerhalb der Simulation*

Tabelle 3: Szenarien 1 – Szenario 1 – Hinzugefügte Elemente

Tabelle 18 listet alle Eingabe bzw. Ausgabeelemente der Basissimulation auf. Die ersten beiden Eingabeelemente sind vergleichbar mit den Eingabeparameter der NetLogo-Ausgangssimulation. „free-spaces-happy“ definiert die Einstellung der Turtle gegenüber freier Plätze. Wahlweise ist „sympathetic“, „unsympathetic“ und „indifferent“ die Variable, die im Verlauf der Funktion „happy-status“ evaluiert wird. (Vgl. Tabelle 2)

Wird im Eingabeparameter „randoms-seeds“ ein Wert ungleich „9999“ eingetragen, so wird dieser Wert als Startwert des Zufallsgenerators gewählt. Durch diese Eigenschaft kann ein eigentlich zufälliges Experiment determiniert werden. Durch diese Eigenschaft lassen sich einzelne Parameterveränderungen besser analysieren. (Vgl. 4.1.6)

Die drei zur Ausgabe gehörenden Diagramme zeichnen über den zeitlichen Verlauf die Veränderung der drei Variablen „Percent-Similar“, „Percent-Unhappy“ und „Completely surrounded“ auf.

Der Monitor „Number“ gibt „population_density“ als natürliche Zahl aus.

betterworld

Für die Bestimmung, ob ein Turtle zufrieden ist, kann eine leicht modifizierte „update-turtles“ Funktion genutzt werden. Sofern nach dem Umzug die Aktualisierung der Werte einen Turtle zufrieden werden lässt, war der Umzug den Anforderungen entsprechend. Ist der Turtle unglücklich, kann diese Operation bis zur definierten Grenze wiederholt werden.

„update-turtles“ aktualisiert die Eigenschaften aller Turtles, dies ist rechenintensiv und vermeidlich. Die eingeführte Funktion *„update-turtle“* aktualisiert ausschließlich die Werte des aufrufenden Turtles.

```
1 to update-turtle
2     set similar-nearby count (turtles-on neighbors)
3     with [color = [color] of myself]
4     set other-nearby count (turtles-on neighbors)
5     with [color != [color] of myself]
6     set total-nearby similar-nearby + other-nearby
7
8     ifelse similar-nearby >= comfortability_threshold   [
9       set happy? true
10        ]
11        [
12          set happy? false
13        ]
14    end
```

Quellcodeauflistung 3

Die Aktualisierung erfolgt aus der Funktion ***find-new-spot*** heraus. Quellcodeauflistung 4, Zeile 11 definiert das Abbruchkriterium.
Nach n-Iterationen wird der Versuch, einen besseren Platz zu finden, abgebrochen. Ist die Anzahl der Durchläufe höher als die des Abbruchkriteriums, wird per Zufall bestimmt, ob der Turtle am aktuellen Ort verbleibt oder den ursprünglichen Platz einnimmt.

Das Abbruchkriterium ***rounds_to_find_bw%*** ist eine prozentuale Angabe aller freien Plätze. 100% bedeutet, dass der Turtle die Möglichkeit hat, alle freien Plätze zu überprüfen.

Sofern die Anzahl der Iterationen noch innerhalb der Grenze liegt, wird der Code unterhalb Zeile 25 ausgeführt.

Zeile 26 und 27 bestimmen einen neuen zufälligen Platz. Entsprechend der Abbildung 6 erfolgt der Programmablauf ab Zeile 28ff.

Falls die zufällige Auswahl einen nicht-zufriedenstellenden Platz wählt, muss die Funktion wiederholt werden.

In Zeile 29 aktualisiert der aufrufende Turtle seinen Status. Ist der Turtle nun zufrieden, wählt der Turtle diesen Platz.

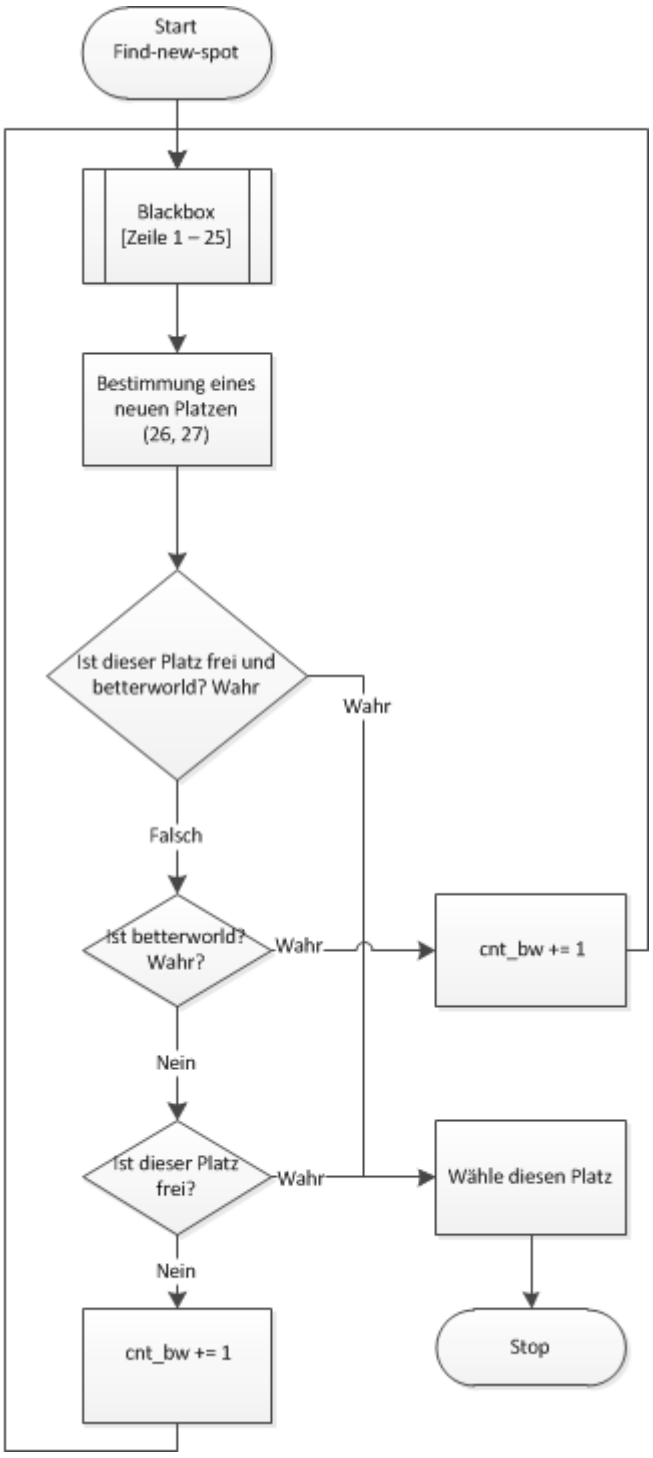

Abbildung 6: Flussdiagramm, find-new-spot-szenario1

```
1 to find-new-spot
2   let freex 0
3   let freey 0
4
5   if cnt_bw = 0 [
6     set oldx xcor
7     set oldy ycor
8     set cnt_bw ( cnt_bw + 1 )
9   ]
10
11      if cnt_bw > rounds_to_find_bw [
12        ifelse random 10 > stay_leave [
13         ask patches with [ not any? other turtles-here]
14          [
15            set freex pxcor
16            set freey pycor
17          ]
18        ][
19          set freex oldx
20          set freey oldy
21        ]
22        set happy? true
23        move-to patch freex freey
24        stop
25      ]
26      rt random-float 360
27      fd random-float 10
28      if not any? other turtles-here and betterworld? [
29        update-turtle
30        if happy? [
31          move-to patch-here
32          stop
33        ]
34        set cnt_bw cnt_bw + 1
35        find-new-spot
36        stop
37      ]
38      ifelse any? other turtles-here
39        [
40          set cnt_bw cnt_bw + 1
41          find-new-spot
42          stop
43        ]
44        [
45          if not any? other turtles-here
46          [ move-to patch-here
47            stop
48        ] ]
  end
```

Quellcodeauflistung 4: find-new-spot-szenario1

4.1.5 Flussdiagramm

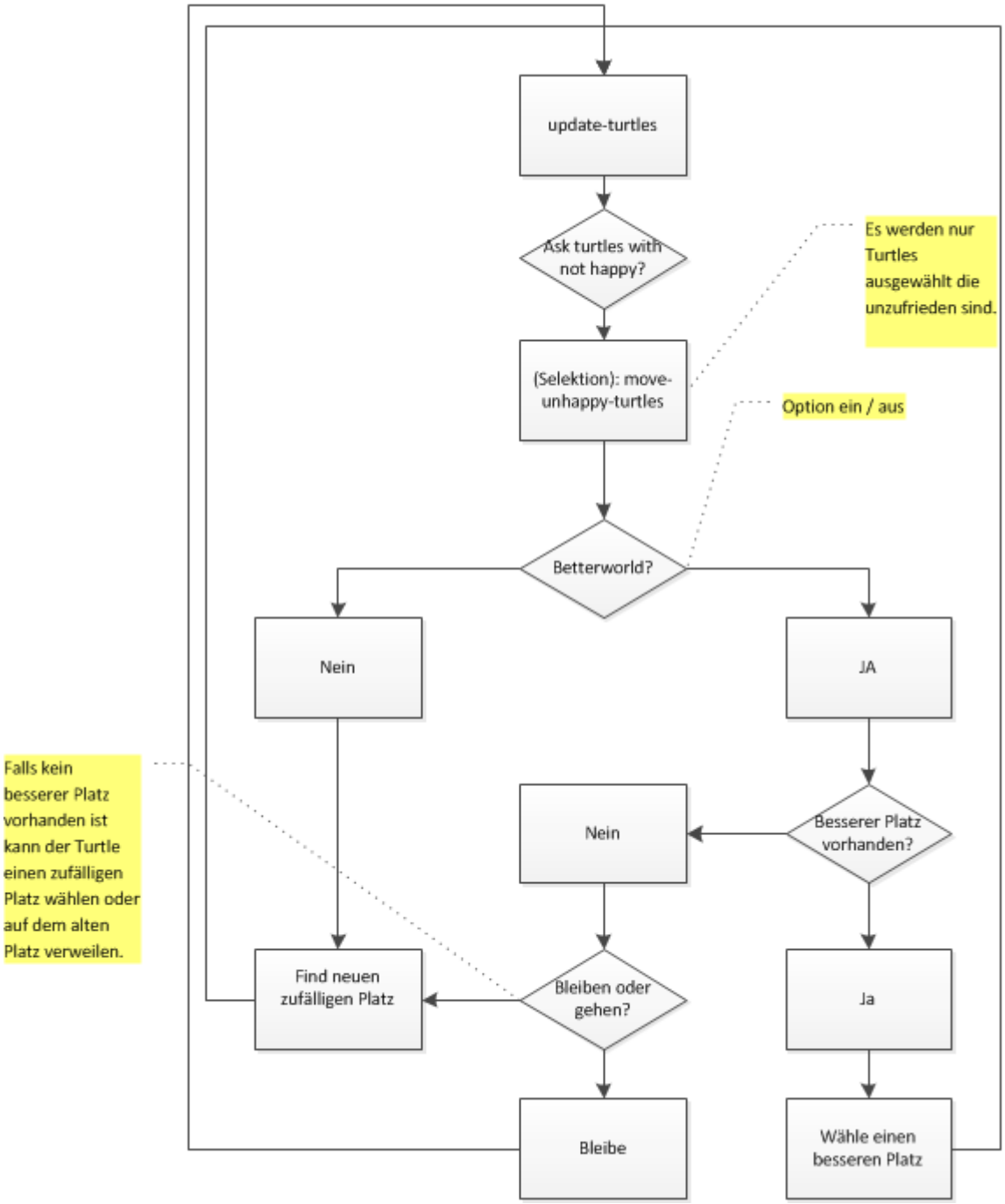

Abbildung 7: Flussdiagram Basismodell

4.1.6 Evaluation

Bedingt durch die Zufallsprozesse ergeben sich bei gleichbleibendem Input unterschiedliche Ergebnisse. Dieser Effekt sollte durch mehrmalige Wiederholung geglättet werden[11]. NetLogo bietet jedoch auch die Möglichkeit, die Initialisierung des Zufallsgenerators festzulegen, d.h. die Zufallsprozesse werden durch einen Parameter initialisiert und können somit beliebig oft repliziert werden.

Als Folge dessen ist es mir möglich, spezielle Experimente mit reduzierten Durchlaufen exakt zu analysieren. Die Entscheidung, ob zufällig oder nicht, muss individuell für das Experiment entschieden werden. In der folgenden Evaluation werden beide Fälle durchgeführt.

Interessante Fragestellungen bezüglich dieser Simulation sind:

- Wie wirkt sich ***betterworld*** auf das Ergebnis aus?
- Wie wirkt sich eine Erhöhung des **round_to_find_bw%** auf das Ergebnis aus?

Die Option ***betterworld*** wird in verschiedenen Einzelexperimenten ausgewertet.

```
1["comfortability_threshold" 6]
2["random-seeds" "9999"]
3["stay_leave" 5]
4["rounds_to_find_bw%" 0 20 40 60]
5["betterworld?" true]
6["population_density" 40 65 90]
7["free-spaces-happy" "indifferent"]
```

Experiment 1 - Szenario 1: betterworld rounds

Das erste Experiment dient der Identifizierung der optimalen **round-to-find-bw%**. Experiment 1 zeigt eine typische Beschreibung von Experimenten in NetLogo. Jede Zeile definiert einen eigenständigen Eingabeparameter nebst Werten.

11 Vgl.: Gesetz der großen Zahlen

Die anhängige Aufzählung der Zahlen bildet jeweils einzelne Parameter. Folgt dem Eingabeparameter eine Klammer, setzen sich die Werte wie folgt zusammen: Die erste Zahl definiert den **Startwert**, die letzte den **Endwert**. Die mittlere Zahl definiert einen Wert, der bis zum Erreichen des Endwerts auf den Startwert addiert wird (**Inkrement**). Hierdurch lassen sich aufsteigende Folgen definieren, die nicht einzeln aufgelistet werden müssen. Abbildung 8 zeigt eine typische Definition der Eigenschaften zu einem Experiment.

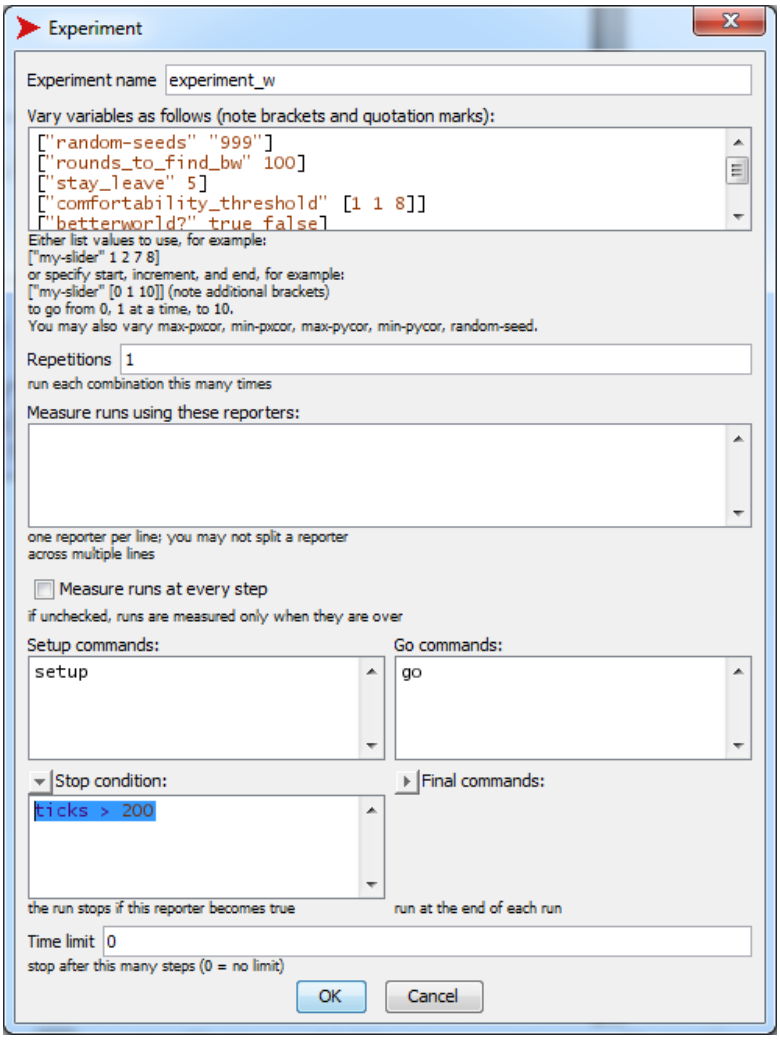

Abbildung 8:BehaviorSpace – Experimentfenster

Für dieses Experiment wird der Zufallsgenerator festgesetzt. Mittels der unter Experiment 1 aufgeführten Konfiguration kann der Einfluss der Wiederholungen auf die Segregation genauer geprüft werden.

Durch die in Experiment 1 beschriebenen Parameter-Variationen entstehen 600 einzigartige Testläufe, die in Tabelle 4 zusammengefasst sind. Die Ergebnisse[12] aus 50 Durchläufen sind auf einen Mittelwert abgebildet. Erkennbar ist, dass mit steigender Anzahl von ***rounds_to_find_bw%***

12 Anzahl der ***Ticks*** bis zu Segregation

die Anzahl der ***Ticks*** vorwiegend sinkt. Jedoch geht aus diesem Experiment nicht definitiv hervor, ob mit steigender Anzahl von ***rounds_to_find_bw%*** die Anzahl der ***Ticks*** konstant sinkt.

Es kann nicht mit Sicherheit gesagt werden, dass mehr Informationen zur schnelleren Segregation führt. Die grau hinterlegte Zelle zeigt eine typische Auffälligkeit, die in allen vergleichbaren Experimenten auftritt: Die Steigerung von 20% auf 40% führt nicht zu einem Sinken der ***Ticks***. Die auffällige Regelmäßigkeit ist kein Einzelfall und ist selbst in Alternativexperimenten (Tabelle 5) erkennbar.

OLAP-Würfel

		round_to_find_bw%			
Statistik=Mittelwert		0	20	40	60
population_density	40	3,60	3,66	3,60	3,56
	65	6,24	4,66	4,58	4,50
	90	158,46	114,72	116,08	109,54

Tabelle 4: Szenario1 - Ergebnis: Experiment 1

OLAP-Würfel

Statistik=Mittelwert

population_density	rounds_to_find_bw						
	0	10	20	30	40	50	60
60	5,25	4,42	4,48	4,34	4,37	4,52	4,45
65	6,24	4,63	4,62	4,67	4,51	4,57	4,57
80	20,79	6,32	5,85	5,96	5,93	5,77	5,87

Tabelle 5: Szenario 1 - Ergebnis: Experiment 1 – 1

Tabelle 6 zeigt die zusammengefassten Ergebnisse von 7200 Simulationen. Durch die zusätzlichen Kennzahlen habe ich eine starke Streuung der Ergebnisse festgestellt. Anhand der zusätzlichen Kennzahlen lässt sich ebenso die Streubreite der Experimente ablesen. Abbildung 9 visualisiert diese Streubreite.

Deskriptive Statistik

density	rtfbw		Statistiken										
			Mittelwert		95% Konfidenzintervall des Mittelwerts		5% getrimmtes Mittel	Median	Varianz	Standardabweichung	Minimum	Maximum	Spannweite
			Statistik	Standardfehler	Untergrenze	Obergrenze							
			Statistik		Statistik	Statistik	Statistik	Statistik	Statistik	Statistik	Statistik	Statistik	Statistik
ticks	65	0	6,12	,058	6,00	6,23	6,06	6,00	1,013	1,007	4	10	6
		20	4,62	,050	4,52	4,72	4,60	4,00	,752	,867	3	8	5
		40	4,55	,048	4,45	4,64	4,51	4,00	,697	,835	3	8	5
		60	4,52	,047	4,42	4,61	4,49	4,00	,665	,816	3	9	6
		80	4,57	,045	4,48	4,66	4,55	4,00	,607	,779	3	9	6
		100	4,60	,051	4,50	4,70	4,57	4,00	,789	,888	3	8	5
	75	0	11,24	,124	10,99	11,48	11,11	11,00	4,603	2,145	7	21	14
		20	4,89	,060	4,77	5,01	4,83	5,00	1,095	1,046	3	10	7
		40	4,83	,059	4,72	4,95	4,77	5,00	1,036	1,018	3	10	7
		60	4,80	,061	4,68	4,92	4,73	5,00	1,102	1,050	3	10	7
		80	4,81	,052	4,70	4,91	4,75	5,00	,805	,897	3	8	5
		100	4,81	,062	4,68	4,93	4,76	5,00	1,160	1,077	3	10	7
	85	0	53,87	1,086	51,74	56,01	52,27	50,00	353,549	18,803	24	157	133
		20	27,91	,807	26,32	29,50	27,10	25,00	195,497	13,982	7	86	79
		40	29,58	,940	27,73	31,43	28,29	25,00	265,160	16,284	7	106	99
		60	29,01	,905	27,23	30,79	27,89	26,00	245,799	15,678	7	87	80
		80	31,70	1,048	29,64	33,77	30,55	28,00	329,420	18,150	6	106	100
		100	30,74	1,008	28,76	32,73	29,41	28,00	304,599	17,453	7	107	100
	95	0	495,73	1,514	492,75	498,71	500,94	501,00	687,730	26,225	266	501	235
		20	456,57	4,884	446,96	466,18	467,67	501,00	7155,062	84,588	115	501	386
		40	461,56	4,442	452,82	470,30	472,17	501,00	5918,153	76,930	141	501	360
		60	454,35	5,048	444,41	464,28	466,30	501,00	7643,284	87,426	145	501	356
		80	469,16	4,261	460,78	477,55	481,58	501,00	5447,348	73,806	102	501	399
		100	467,43	4,343	458,88	475,97	479,11	501,00	5658,908	75,226	167	501	334

Tabelle 6: Szenario 2 - Auswertung 7200 Experimente

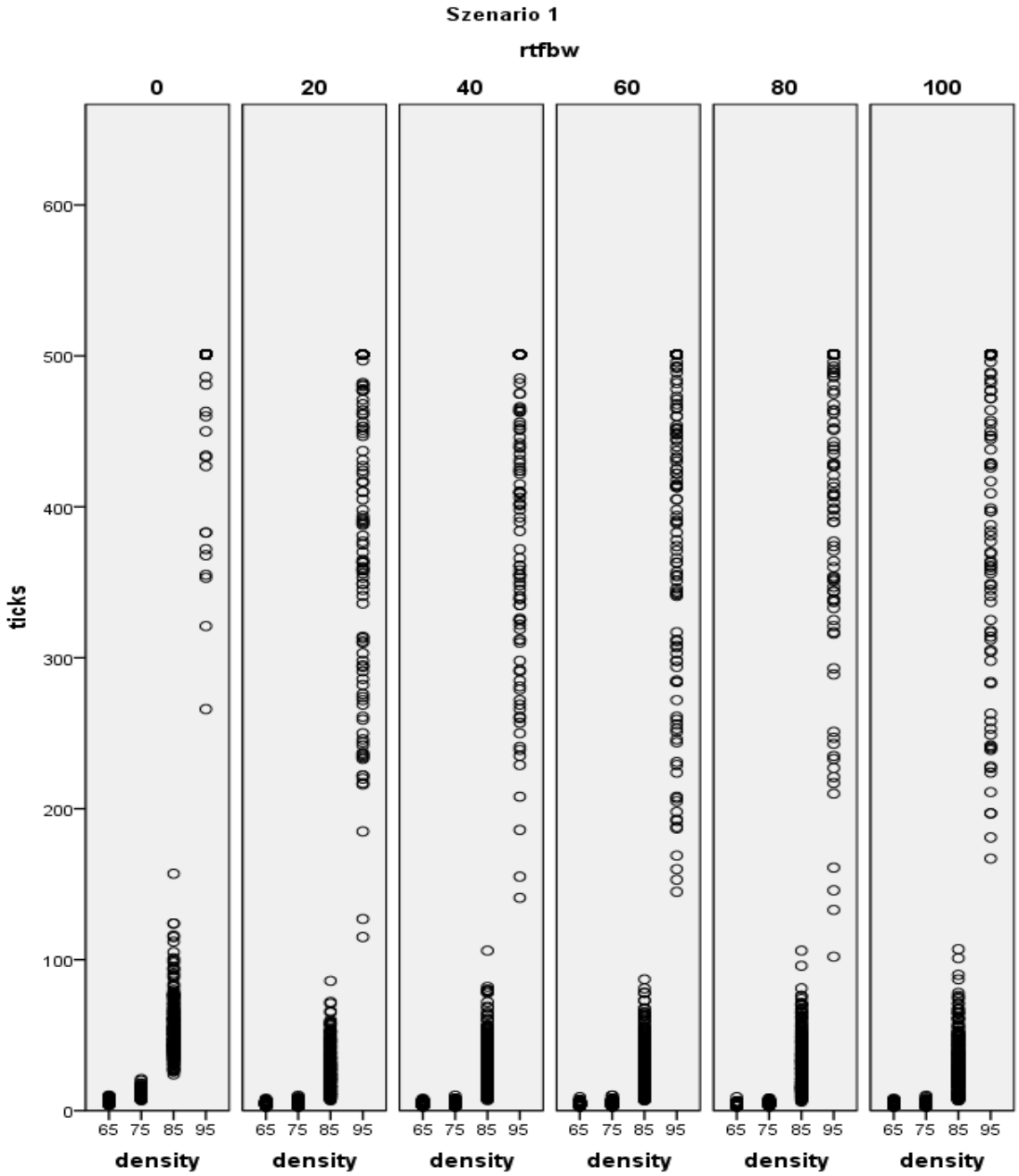

Abbildung 9: Streubreite bei 7200 Simulationen

Ich kann daher nicht mit Bestimmtheit sagen, in wie weit eine höhere Anzahl an ***round-to-find-bw%*** zu einem besseren Ergebnis führt.

Im Verhältnis zur Simulation aus der NetLogo-Bibliothek lässt sich jedoch mit Sicherheit festhalten, dass die Simulation schneller segregiert. Bedingt durch den Informationsvorsprung wählt der Turtle zumeist die optimierte Möglichkeit und verringert somit die Anzahl der Platzwechsel bis zur Zufriedenheit.

4.1.7 Konklusion

betterworld beschleunigt durch die optimierte Suche die Segregation. Die Bestimmung der besten Anzahl für ***rounds_to_find_bw%*** ist nicht eindeutig mathematisch lösbar. Bedingt durch die hohe Streubreite kann nur eine statistisch Signifikaten-Relation dokumentiert werden. Das Experiment mit 7200 Simulationen benötigte jedoch mehr als eine gesamte Woche Rechenzeit.

4.2 Szenario 2 – Multikulturalismus

Schelling schreibt zu seinen Simulationsversuchen:

> ***„The only requirement in my model is that the distinction be twofold, exhaustive, and recognizable."***
> ***[Schelling, 1969] (Seite 488)***

Daher beschränkt er seine Experimente lediglich auf zwei Ausprägungstypen. Nur selten lassen sich Menschen anhand einer einzigen binären Variablen ausreichend klassifizieren. Daher wird in diesem Szenario die Anzahl der Ausprägungstypen erhöht.

Die Festlegung der Anzahl an Turtle-Klassen für eine möglichst realitätsnahe Simulation ist schwierig. Die im Kapitel 2.1 vorgestellten Segregationsklassifizierungen erlauben keine direkte Bestimmung. Relevante demographische Merkmale sind einfach festzulegen, die verfügbaren Status[13] sind gering. Soziale, religiöse und vor allem ethnische Unterschiede lassen eine hohe Zahl an Möglichkeiten zu. Beobachtet man alle

13 Plural

Unterscheidungsmerkmale kombiniert, entsteht eine unüberschaubare Anzahl an Charakteristika bzw. Klassen zur Kategorisierung.

Die dadurch entstehenden Kombinationsmöglichkeiten sind unrealistisch. Die folgende Konzeption bestimmt eine taugliche Anzahl an Kulturen.

4.2.1 Konzeption

Die neu-eingeführten Turtle-Klassen werden im Folgenden ***Tribes*** genannt. In der Simulation dient stets die Farbe als Unterscheidungsmerkmal. Daher wird auch hier die Farbe den unterscheidbaren Charakter der ***Tribes*** definieren. NetLogo bietet standardmäßig 140 Farben und somit maximal 140 unterscheidbare ***Tribes***.

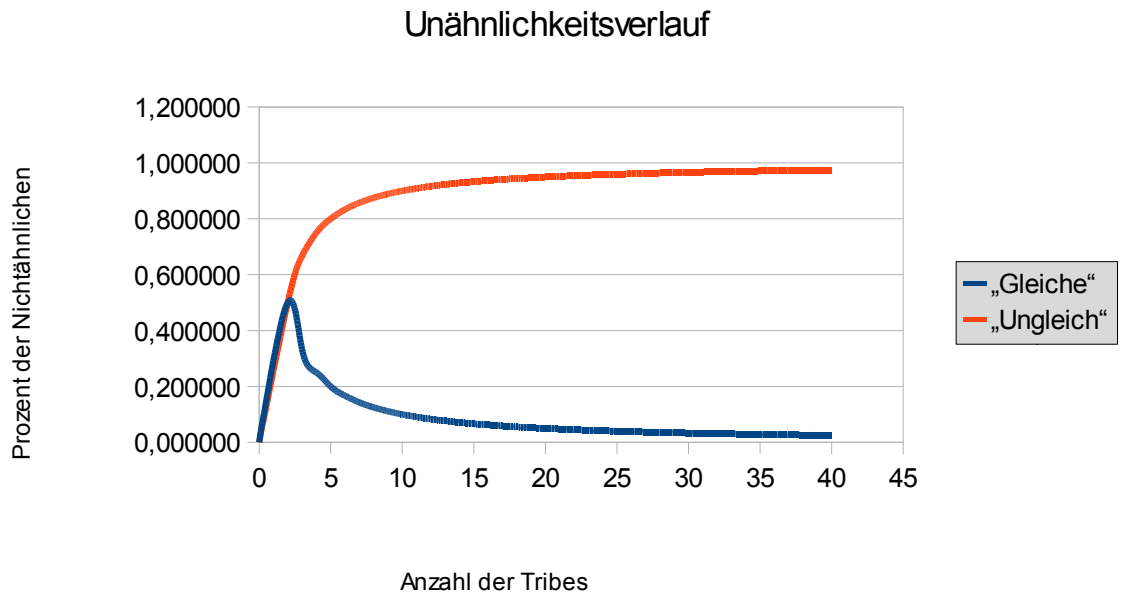

Abbildung 10: Unähnlichkeitsverlauf

Jedoch lassen sich die 140 Farben, die auf dem kleinen Simulationsfeld Platz finden müssen, mit dem menschlichen Auge nicht ausreichend gut unterscheiden.

Des Weiteren ist zu beachten, dass mit steigender Anzahl neuer ***Tribes*** die Anzahl der nicht-gleichen Turtle exponentiell steigt. Abbildung 10

visualisiert den Verlauf der Unähnlichkeit. Bei mehr als zehn unterschiedlichen Tribes fällt der prozentuale Anteil an Gleichen unter 10%. Die Anzahl der Kombinationsmöglichkeiten steht in Relation zur Wahrscheinlichkeit, dass die Segregation eintritt.

Bedingt durch beide Einschränkungen stelle ich den Nutzen mit mehr als zehn unterschiedlichen ***Tribes*** in Frage. Daher setzte ich die maximale Anzahl unterscheidbarer ***Tribes*** auf 10.

4.2.2 Implementation

Die Verteilung der Klassen ist beliebig einstellbar. Jede Klasse kann einen Anteilswert aufnehmen und erhält entsprechend der Gesamtzahl Turtle. Eine grundlegende Plausibilitätsüberprüfung verbietet eine nicht korrekte Verteilung.

Hinzugefügte Elemente

Bezeichnung	***Typ***	***Erläuterung***
Eingabe		
cnt-tribes	*Slider*	*Anzahl verschiedener Turtle-Klassen. Wertebereich zwei bis 10.*
[n]th-c	*Slider*	*Natürliche Zahl bis 100, die in Kombination mit den anderen Tribes eine relative Verteilung der Turtle erlaubt.*

Tabelle 7: Szenario 2 – Hinzugefügte Elemente

Die Berechnung der Verteilung erfolgt auf diese Weise: (Quellcodeauflistung 5)

Der Quotient (first_cv; Zeile 3) aus first_c und der Summe alle Color-Verteilungen bildet den prozentualen Anteil einer Klasse an der Gesamtbevölkerung. Daher muss sichergestellt werden, dass die Anzahl für cnt-tribes mit der Anzahl der eingestellten Verteilung übereinstimmt.

```
1   ifelse ( first_c != 0 )
2 [
3   set first_cv   ( ( first_c /  pop_dist ) * number )
4 ]
5  [
6  set first_cv  0
7 ]
```

Quellcodeauflistung 5 - Bestimmung relativer Anteil eines Turtle-klasse.

Flussdiagramm

Da für dieses Szenario keine entscheidungsrelevanten Quellcodeveränderungen durchgeführt werden, gilt der Entscheidungsbaum der Basissimulation. Siehe Abbildung 7

4.2.3 Evaluation

Erwartete Veränderungen

Prognostizierbar ist, dass mit jedem hinzugefügten Tribe die Simulation später segregiert. Abbildung 10 unterstützt diese Erwartung. Ein weiterer Indiz liefert die Berechnung der Kombinationen durch Repetition an:

Bei zwei unterscheidbaren ***Tribes*** und acht möglichen Nachbarfeldern können maximal 45 unterschiedliche Kombinationsmöglichkeiten eintreten. Zu beachten ist, dass ein freier Platz kombinationsrelevant ist. (Siehe: Formel 7) Für drei unterschiedliche ***Tribes*** steigt die Anzahl der Kombinationen um das 5-fache . Für zehn unterscheidbare ***Tribes*** entstehen 43758 Kombinationsmöglichkeiten. (Siehe: Formel 6).

$$mit: n=3\, k=8$$

$$\binom{n+k-1}{k} = \binom{3+8-1}{8} = 45\, Kombinationen$$

Formel 6: Repetition für 2;8

$$mit: n=11\,;\, k=8$$

$$\binom{n+k-1}{k} = \binom{11+8-1}{8} = 43758\, Kombinationen$$

Formel 7: Repetition für 10;8

Eingetretene Veränderungen

```
["random-seeds" "9999"]
["stay_leave" 5]
["population_density" 60 70 80 90]
["free-spaces-happy" "indifferent"]
["comfortability_threshold" 5]
["cnt-tribes" 4]
["betterworld?" true false]
["rounds_to_find_bw%" 100]
    ["first_c" 100]
    ["second_c" 100]
    ["third_c" 100]
    ["fourth_c" 100]
    ["fifth_c" 0]
    ["sixth_c" 0]
    ["seventh_c" 0]
    ["eighth_c" 0]
    ["tenth_c" 0]
    ["ninth_c" 0]
```

Experiment 2 - Szenario 2: N-Kulturen CT=5

Experiment 2 sammelt die NetLogo Einstellungen zu den folgenden Tabellen. Wird eine Farbe hinzugefügt (cnt-tribes + 1) muss der zugehörige Anteilswert ebenso größer null definiert werden.

Die Tabellen 7 – 11 dokumentieren die nötigen ***Ticks*** bis das entsprechende Szenario segregiert. Die frühere Segregation der Simulation ist durch die Nutzung von ***betterworld*** bedingt. (Siehe Tabelle 8ff) Ohne ***betterworld*** können die Experimente mit mehr als 6 Farben nicht vor dem Abbruchkriterien (1000 Ticks) entmischt werden.

OLAP-Würfel

Statistik:Mittelwert

CT=4		betterworld?		
		false	true	Insgesamt
Density	60	15,86	6,22	11,04
	70	17,20	6,82	12,01
	80	19,70	7,64	13,67
	90	21,18	9,58	15,38
	Insgesamt	18,49	7,57	13,03

Tabelle 8: Szenario 2 - 2 Farben; CT=4

OLAP-Würfel

Statistik=Mittelwert

CT=4		betterworld?		
		false	true	Insgesamt
Density	60	55,66	6,94	31,30
	70	61,54	7,84	34,69
	80	67,60	8,74	38,17
	90	80,16	11,94	46,05
	Insgesamt	66,24	8,87	37,55

Tabelle 9: Szenario 2 - 4 Farben; CT=4

OLAP-Würfel

Statistik:Mittelwert

CT=4		betterworld?		
		false	true	Insgesamt
Density	60	141,48	7,20	74,34
	70	203,80	7,60	105,70
	80	291,54	8,56	150,05
	90	365,68	16,44	191,06
	Insgesamt	250,63	9,95	130,29

Tabelle 10: Szenario 2 - 6 Farben; CT=4

OLAP-Würfel

Statistik=Mittelwert

CT=4		betterworld?		
		false	true	Insgesamt
Density	60	551,98	7,40	279,69
	70	987,68	7,20	497,44
	80	1001,00	8,68	504,84
	90	1001,00	24,56	512,78
	Insgesamt	885,42	11,96	448,69

Tabelle 11: Szenario 2 - 8 Farben; CT=4

OLAP-Würfel

Statistik:Mittelwert

CT=4		betterworld?		
		false	true	Insgesamt
Density	60	1001,00	6,92	503,96
	70	1001,00	7,56	504,28
	80	1001,00	9,44	505,22
	90	1001,00	37,46	519,23
	Insgesamt	1001,00	15,35	508,17

Tabelle 12: Szenario 2 - 10 Farben; CT=4

Abbildung 11 enthält einen ähnlichen Verlauf wie Abbildung 10. (Zu beachten ist die Deaktivierung von ***betterworld***.) Damit wird folgende Aussage gestützt:

Mit steigender Anzahl der ungleichen Turtle sinkt die Wahrscheinlichkeit zur Segregation.

Bedingt ist diese Behauptung dadurch, dass die Wahrscheinlichkeit, einen nicht gleichen Tribe zu treffen, mit jedem hinzugefügtem Tribe steigt.

In Abbildung 12 ist die Option ***betterworld*** aktiv. Für 60%, 70% und 80% Dichte entsteht ein annähernd paralleler Verlauf zur x-Achse. Lediglich bei 90% Dichte steigt die Anzahl der Ticks signifikant.

Szenarien mit einer Dichte von 90% benötigen jedoch auffallend mehr ***Ticks*** pro hinzugefügter Farbe.

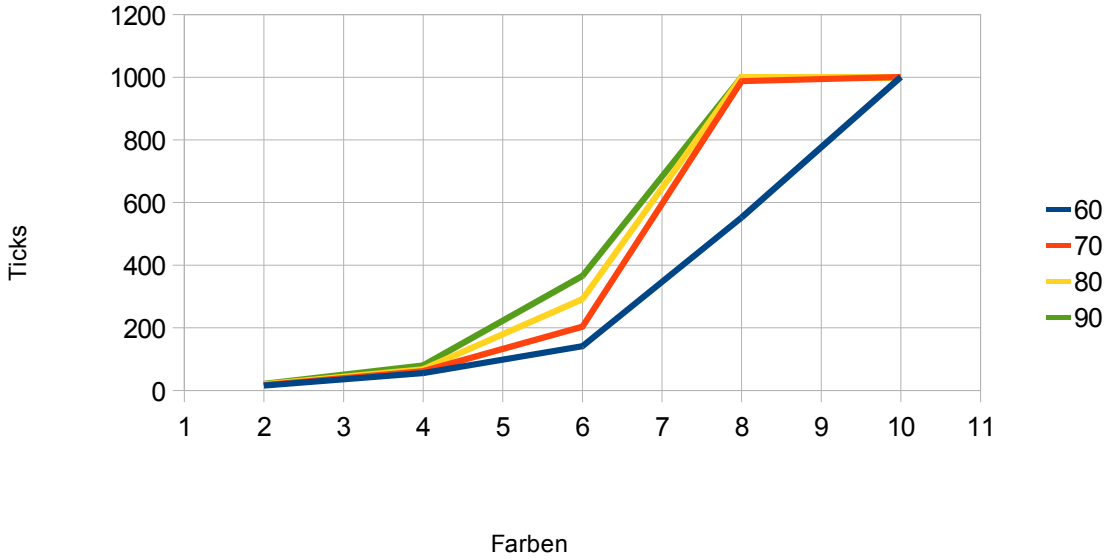

*Abbildung 11: Szenario 2 - Diagramm **Ticks** je Farbe * Dichte (bw? = false)*

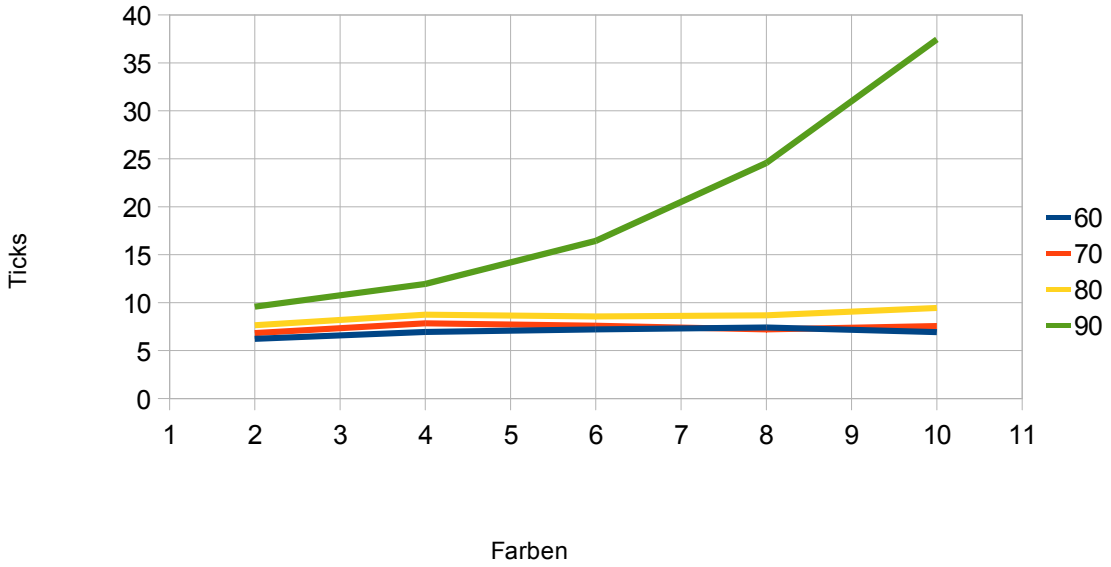

*Abbildung 12:Szenario 2- Diagramm **Ticks** je Farbe * Dichte (bw? = true)*

4.2.4 Konklusion

Für den Fall ***betterworld*** = false:

Die Anzahl der benötigten ***Ticks*** steigt in einem ähnlichen Verhältnis, wie die berechneten Kombinationsmöglichkeiten. Für Dichten über 60% konnte keine meiner Simulationen innerhalb des Abbruchkriteriums segregieren.

Für den Fall ***betterworld*** = true:

Bedingt durch die geschickte Auswahl des nächsten Platzes konnte bis zu einer gewissen Dichte die höhere Anzahl an nicht-zufriedenstellenden Nachbarn ausgeglichen werden. In meinen Simulationen erfolgte eine Segregation innerhalb des Abbruchkriteriums. ***betterworld*** erlaubt also die Segregation bei komplexen Szenarien.

4.3 Szenario 3 - Integrationsbeauftragter

Durch die Einführung von ***betterworld*** ist es möglich, dass Turtle alle zufriedenstellenden Plätze finden. Die in 4.1.6 beschriebene Erweiterung kann die annähernd perfekte Situation darstellen:

> Die Turtle besitzen eine unendliche Sichtweite.

Dies ist nur bedingt realitätsnah und fördert darüber hinaus die Segregation.

4.3.1 Konzeption

In diesem Szenario führe ich eine segregationshinderliche Funktionalität ein. Mit sogenannten Integrationsbeauftragten versuche ich einer Segregation vorzubeugen. Diese sollen die Umzüge der Turtle koordinieren. Das Ziel des Beauftragten ist es, die Segregation zu verhindern. Die Turtle müssen jedoch bis zu einem gewissen Teil segregieren, damit sie zufrieden sind. Mit Blick auf die Maximierung der lokalen Wünsche(**%-**

similar-wanted), ist der Zweck dieses Szenarios, die globale Segregation zu minimieren.

Um die globale Segregation zu bestimmen, bedarf es einem geeigneten Evaluationsinstrument. Da ich in diesem Szenario das allgemein geltende Stopkriterium (Zufriedenheit aller Turtle) nicht nutzt kann, muss diese Simulation eine gewisse Anzahl an ***Ticks*** durchlaufen. Die Protokollierung der Diagramme erlaubt mir die Evaluation der Simulation.

4.3.2 Implementation

Für dieses Szenario führe ich die neuen Funktionen ***„heck-neighbors*** (Siehe: Quellcodeauflistung 6) und ***find-new-spot*** *(Siehe: Quellcodeauflistung 7)* ein.

Zu Beginn der Funktion ***check-neighbors*** werden alle alten Koordinatoren (Leader) zurückgesetzt. Alle zufriedenen Turtle, die mindestens ***leader_limit*** (Vgl.: Tabelle 13) freie Plätze in der Nachbarschaft besitzen, werden zu Leadern (Zeile 6).

```
1 to check-neighbors
2   ask turtles [ set leader? false ]
3   ask turtles with [ happy? = true ] [
4     if ( 8 - count turtles-on neighbors ) > leader_limit
5     [
6       set leader? true
7     ]
8   ]
9 end
```

Quellcodeauflistung 6 - check-neighbors

Quellcodeauflistung 7 beschreibt die Erweiterung zu ***find-new-spot***. Zeile 1 selektiert ausschließlich „Leader-Turtle“ der gleichen Farbe. Ein zufällig gewählter Leader wählt einen freien Platz aus der Nachbarschaft und vermittelt diesen. (Zeile 5, 6)

```
1 ask turtles with [ leader? = true
  and  color = [color] of myself ]
2 [
3  ask neighbors with [ not any? other turtles-here]
4   [
5     set to-x pxcor
6     set to-y pycor
7   ]
8 ]
```

Quellcodeauflistung 7 - Erweiterung find-new-spot Leader

Hinzugefügte Elemente

Bezeichnung	***Typ***	***Erläuterung***
Eingabe		
leader_limit	*Slider*	*Bestimmt die Anzahl der freien Plätze in der Nachbarschaft. Besitzt ein Turtle mehr als die Anzahl entsprechender Nachbarn, kann dieser Leader werden.*
Ausgabe		
mark leaders	*Button*	*Markiert alle Leader mit einem „L“*

Tabelle 13: Szenario 3 – Hinzugefügte Elemente

4.3.3 Flussdiagramm

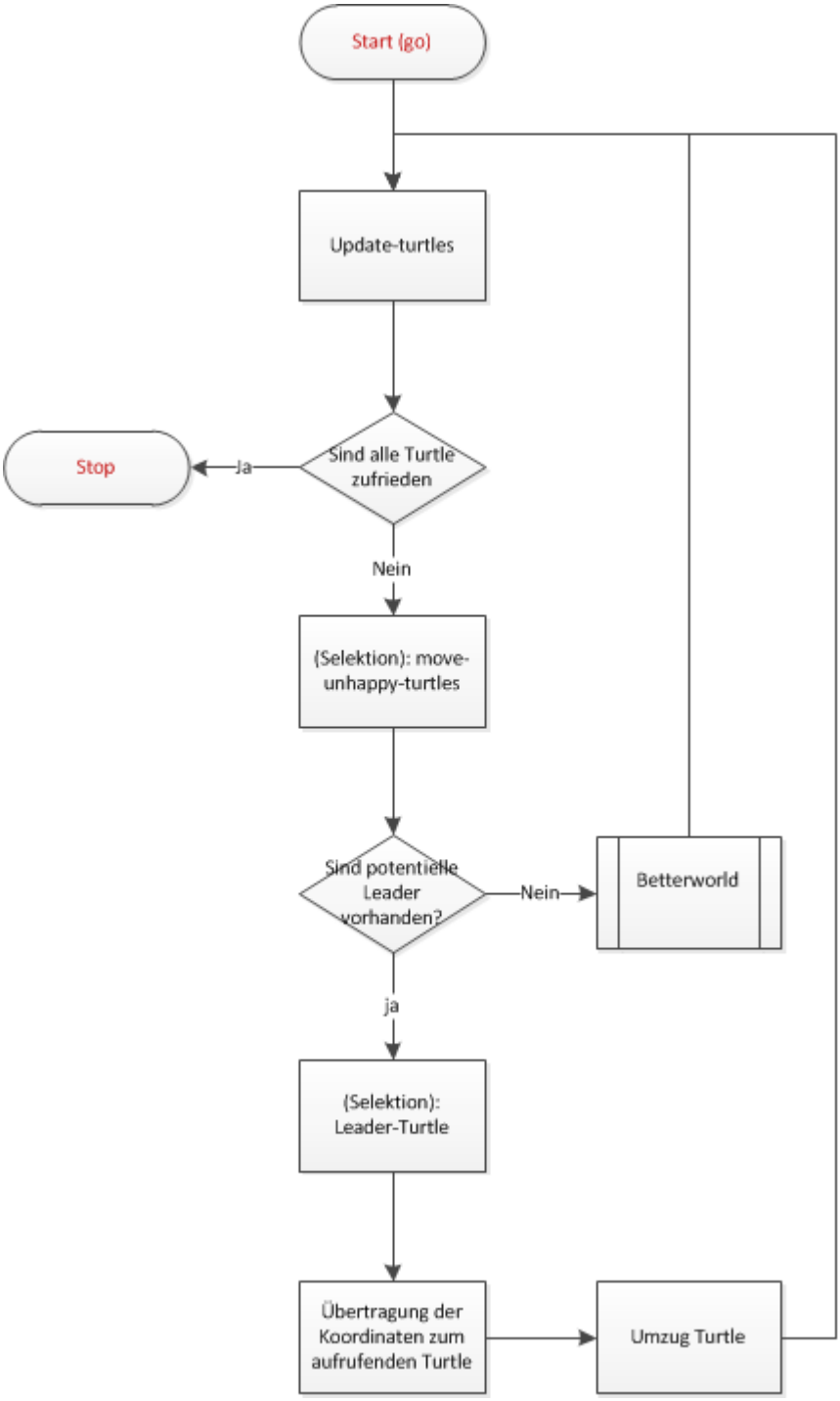

Abbildung 13: Szenario 3 - Flussdiagramm

4.3.4 Evaluation

Erwartete Veränderungen

Die Integrationsbeauftragten führen bewusst Ungleiche zusammen. Deshalb gehe ich davon aus, dass die Simulation niemals enden wird. Die zwei unterschiedlichen Akteursklassen arbeiten gegeneinander. Der gemeine[14] Turtle versucht seinen **comfortability_threshold (CT)** zu erreichen. Gleichzeitig bindet der Leader nicht-gleiche Turtle an sich. ***percent-unhappy*** sowie ***percent-similar*** werden zeigen, welcher Akteur mehr Einfluss auf die Gesellschaft hat. Wenn ***percent-unhappy*** steigt, kann ich davon ausgehen, dass Leader die Gesellschaft mehr fragmentieren, als Turtle sie defragmentieren. Steigt ***percent-similar***, ist der Einfluss der Leader wahrscheinlich nur gering.

Eingetretene Veränderungen

```
1     ["comfortability_threshold" 7]
2     ["stay_leave" 4]
3     ["rounds_to_find_bw%" 100]
4     ["population_density" 75]
5     ["random-seeds" "999"]
6     ["leader_limit" 2]
7     ["betterworld?" false]
8     ["free-spaces-happy" "indifferent"]
```

Experiment 3 - Szenario 3 – stabil

Experiment 3 ist ein gutes Beispiel für eine Simulation, die niemals endet. Abbildung 14 zeigt den Verlauf der Diagramme nach 100 Ticks. Integrationsbeauftragte arbeiten gegen die übrigen Turtle. Zwar ist eine Entmischung für *betterworld=true* erkenntbar, jedoch tritt keine Segregation[15] ein. Der Verlauf der grauen Linie protokolliert die Anzahl der Leader. Der sprunghafte Verlauf in den linken Diagrammen (bw=true) ist ein Indiz für die wechselseitige Interdependenz von Leadern und ***betterworld.*** Die von mir erwartete Veränderung bei ***percent_similar*** ist somit

14 Auf die Gesamtheit, Allgemeinheit bezogen
15 Im Sinne von: Entmischung der Gesamtheit, Der Prozess der Segregation ist erkennbar.

eingetreten; es erfolgte keine Segregation in beiden Experimenten. ***Betterworld*** erreicht folglich eine höhere Gesamtähnlichkeit.

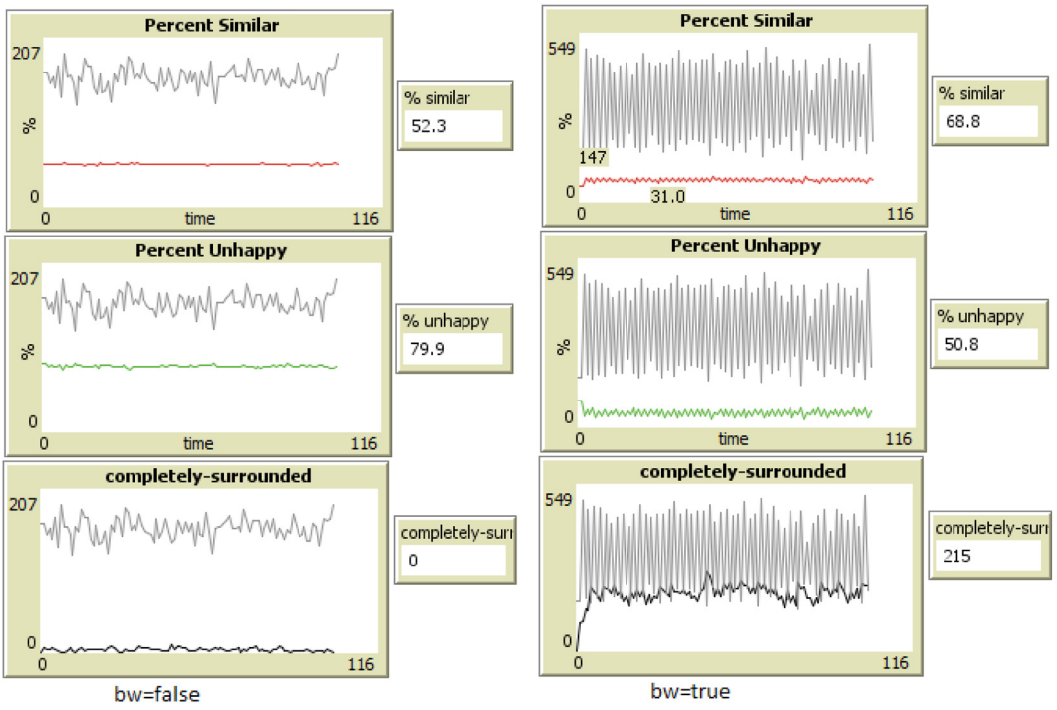

Abbildung 14: Szenario 3 betterworld gegen nicht betterworld

```
1      ["comfortability_threshold" 6]
2      ["stay_leave" 5]
3      ["rounds_to_find_bw%" 100]
4      ["population_density" 85]
5      ["random-seeds" "999"]
6      ["leader_limit" 6]
7      ["betterworld?" false]
8      ["free-spaces-happy" "indifferent"]
```

Experiment 4 - Szenario 4 - Integrationsbeauftragte

Experiment 4 analysiert die Veränderung, die eintritt, wenn der CT[16] in regelmäßigen Abständen reduziert wird. Mein Ziel ist es; den CT soweit zu verringern bis die Simulation segregiert.

Die ersten 150 Ticks verlaufen ungleichmäßig und die Turtle sind gut durchmischt. Durch die kontinuierliche Verringerung von CT steigt die

16 CT = comfortability_threshold

Toleranz und die Wahrscheinlichkeit der Segregation an. Die eigentliche integrationsförderende Maßnahme führt dazu, dass sich die Turtle entmischen. Nach den ersten Umzügen entstehen erste aufgeteilte Bereiche. Die zufällige Platzierung nutzt den geringeren CT-Wert, um die Gesamtheit zu segregieren.

Abbildung 15 zeigt die Ergebnisse von Experiment 4. Den CT-Wert habe ich jeweils nach 1000 Ticks um eine Einheit reduziert.

Erst ab einem CT von 5 ändert sich die Ausgabe der Diagramme. Die aktive Integrationshilfe sorgt dafür, dass passive Segregation entsteht. Sinkt der CT unter 5, kann die Population segregieren, da die Toleranzschwelle niedrig genug ausfällt. Meine erwarteten Veränderung bei ***percent_similar*** sind gleichermaßen eingetreten

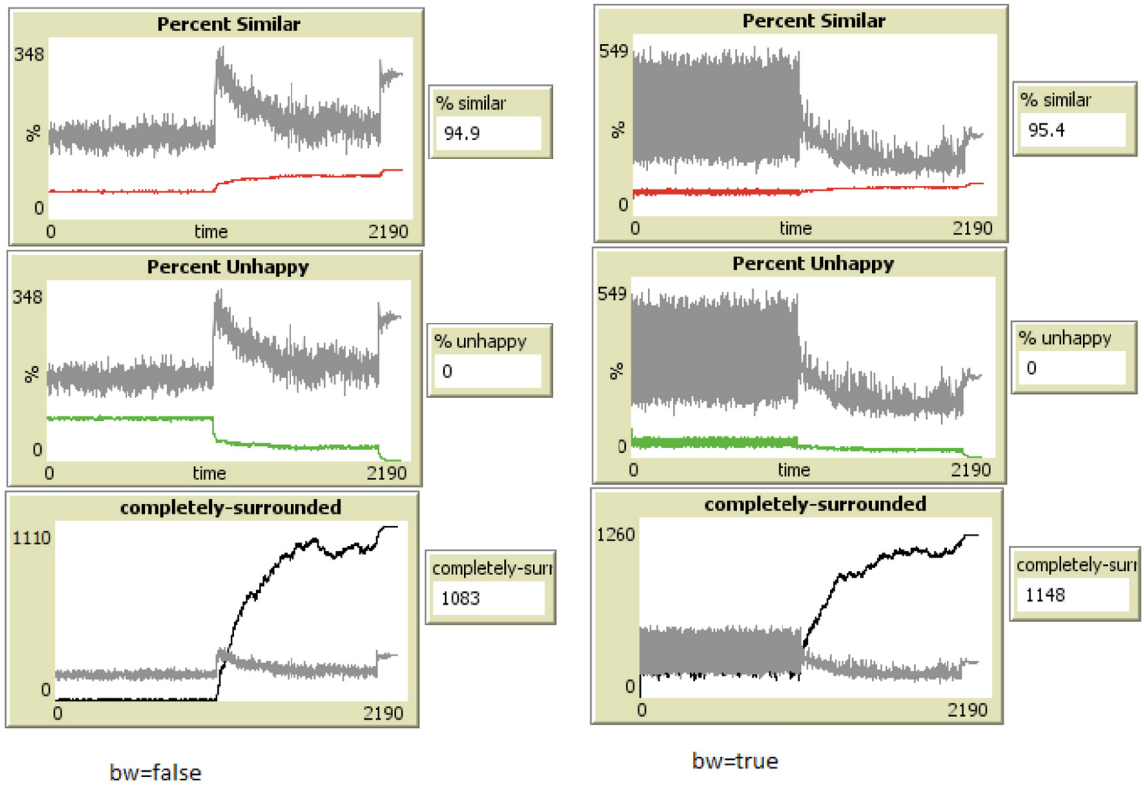

Abbildung 15: Szenario 3: Reduzierung CT [1000 ticks]

4.3.5 Konklusion

Es hat sich bewahrheitet, dass die von mir eingeführten Integrationsbeauftragten die Segregation aufhalten. Es entsteht eine wechselseitige Beeinflussung, welche zu keinem eindeutigen Ergebnis führt.

Ein kontinuierliches Absenken des CT-Wertes hat zur Folge, dass die Simulation trotz Integrationsbeauftragten segregiert. ***betterworld*** erreicht in allen Experimenten eine stärkere Gruppenbildung. (Completely_Surrounded jeweils höher bei bw=false)

4.4 Szenario 4 - Intelligente Patches

Die vorherigen Szenarien implementieren die programmatische Intelligenz der Simulation komplett in den Turtle. Im diesem Szenario wird ein zusätzlicher Teil der Intelligenz auf die eigentlich passiven Patches übertragen. Die Patches bestimmen selbstständig und entsprechend der gewählten Methode, ob diese segregationsfördernd oder intergrationsfördernd agieren.

4.4.1 Konzeption

Alle Patches analysieren die anliegenden Nachbarfelder. Bestimmt sich die Nachbarschaft eines Feldes durch eine qualifizierte Mehrheit einer Art, so kann diese positive Ausgangslage für den Zuzug einer Turtleklasse markiert werden. Im nächsten Tick wählen nun „interessierte“ Turtle dieses Feld. Hierbei kann eine segregations- oder integrationsfördernde Wirkung initiiert werden. Analog zu den Kapiteln 4.1 und 4.3 werden beide Wirkungen in einem Modell vereint.

4.4.2 Implementation

Um die zu diesem Szenario gehörende Funktionalität bereit zu stellen, habe ich eine neue Funktion programmiert. Diese Funktion heißt ***check-patches*** und übernimmt die Markierung der qualifizierten Felder. (Siehe Abbildung 16) Entsprechend der Vorgabe aus ***free-spaces-happy*** werden unterschiedliche Werte berechnet. Erreicht eine Farbe die Mehrheit in einer betrachteten Nachbarschaft, so wird der Patch artenspezifisch eingefärbt.

Darüber hinaus benötige ich die Funktion ***intelligent-patches*** (IP). Diese unterscheidet zu Beginn anhand des Wertes von ***Segregation?***, ob segregationsfördernd oder integrationsfördernd fortgefahren wird. Für den Fall, dass segregationsfördernd entschieden wird, suchen die Turtle ein für die Klasse reservierten Patch.

Wird jedoch integrationsfördernd entschieden, wird ein Feld der anderen Klasse gewählt. Somit zieht ein Turtle auf ein Feld, welches mit der alternativen Farbe gekennzeichnet ist.

Hinzugefügte Elemente

Bezeichnung	***Typ***	***Erläuterung***
Eingabe		
intelligent_patches?	*Switch*	*Ermöglicht die Funktion, intelligente Patches einzuschalten.*
Segregation?	*Switch*	*Schaltet zwischen segregationfördernder und integrationfördernder Simulation.*
Ausgabe		

Tabelle 14: Szenario 4 - Hinzugefügte Elemente

4.4.3 Flussdiagramm

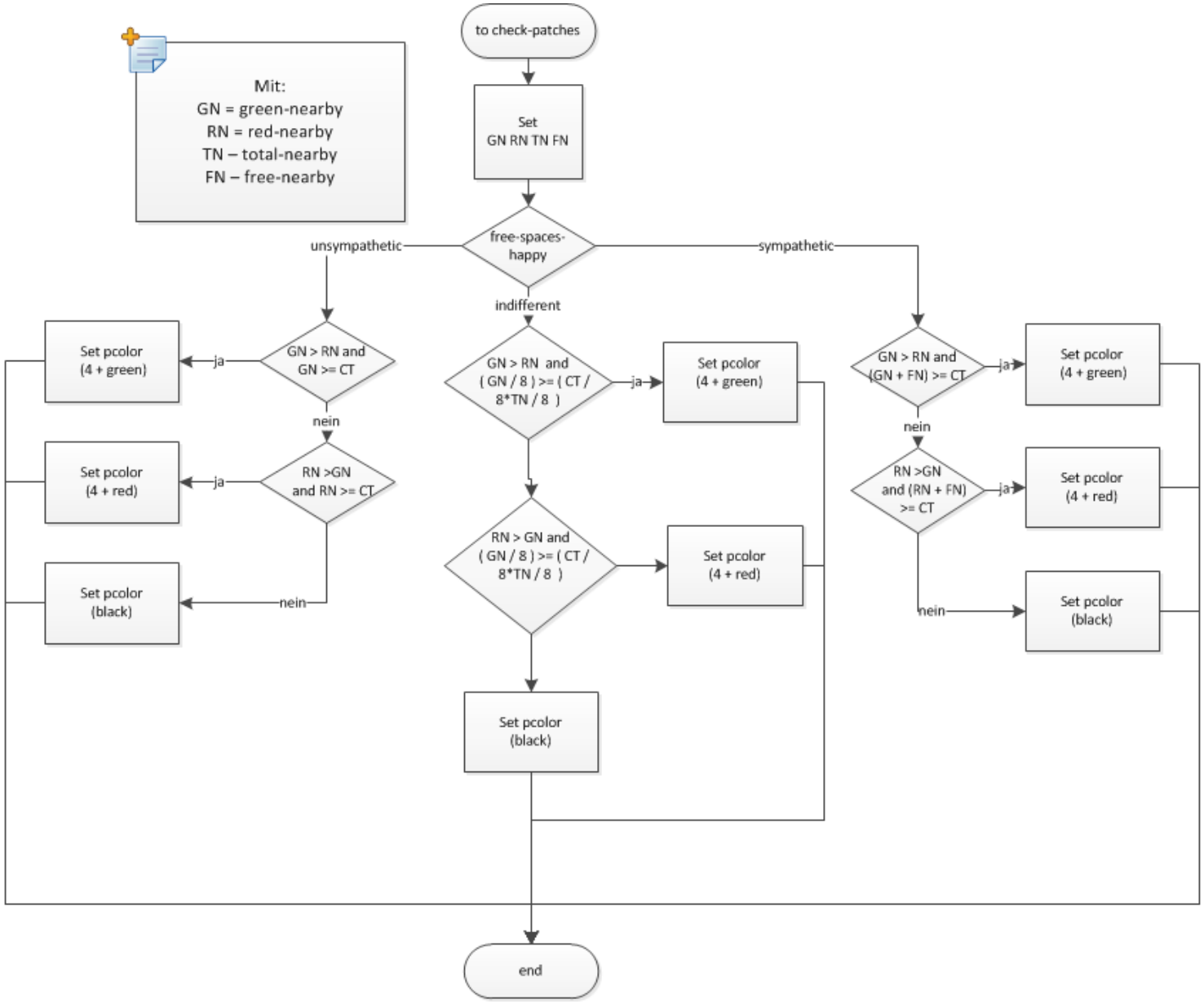

Abbildung 16: Szenario 4 – Flussdiagramm

4.4.4 Evaluation

Erwartete Veränderungen

Bei diesem Szenario vermute ich, dass im segregationsfördernden Fall die Simulation früher segregiert.

Im Gegensatz dazu sollte bei der integrationsfördernden Simulation die Segregation nicht eintreten. Analog zum Kapitel 4.3 müsste die Simulation unendlich fortlaufen.

Eingetretene Veränderungen

Experiment 5 zielt darauf ab, die unterschiedlichen Einstellungsparameter vergleichbar zu evaluieren. Bedingt durch die drei wählbaren Schalter ***betterworld***, ***intelligent-patches*** und ***Segregation?*** entstehen acht unterscheidbare Experimentesammlungen. Jede Experimentsammlung besteht aus 100 Iterationen für die in Experiment 5; Zeile 5 aufgeführten Dichten. Es entstehen insgesamt 3200 Einzelexperimente in acht Sammlungen. Endet eine Simulation nicht nach 1000 ***Ticks***, wird diese abgebrochen.

```
1 ["stay_leave" 5]
2 ["Segreation?" true false]
3 ["free-spaces-happy" "indifferent"]
4 ["random-seeds" "1" […] "100"]
5 ["population_density" 60 70 80 90]
6 ["comfortability_threshold" 6]
7 ["intelligent-patches?" false true]
8 ["betterworld?" true false]
9 ["rounds_to_find_bw%" 100]
```

Experiment 5 - Szenario 4: intelligente Patches

Tabelle 15 zeigt das Ergebnis von Experiment 5 für ***Segregation?=true***. Die jeweilige Unterschrift beschreibt die vorhandenen Zustände. Die oberste Tabelle (F1) zeigt, dass komprimierte Ergebnis [Alle Parameter=false]. In diesem Fall verhält sich die Simulation entsprechend der Originalversion. Die Turtle wählen einen neuen und zufälligen Platz.

In der zweiten Tabelle (F2) endet keine Simulation. ***IP=true*** in Kombination mit ***Segregation?=true*** führt dazu, dass die Simulation nicht segregiert. Nach 1000 ***Ticks*** wurde jede Simulation vermischt beendet.

In der dritten Tabelle (F3) entmischt ***betterworld*** die Turtle. Angesichts der Tatsache, dass keine intelligenten Patches verfügbar sind, wählen die Turtle die zweite Alternative ***betterworld.***

Segregation = false

OLAP-Würfel *(F1)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	70,58	,856	69,00	70,10
	70	104,17	1,172	105,00	103,51
	80	179,34	2,374	176,00	177,83
	90	288,16	5,110	278,00	284,05
	=	160,56	4,426	136,00	138,36

a. Segregation? = fals, betterworld? = false, int_patch? = false

OLAP-Würfel *(F2)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	1001,00	,000	1001,00	1001,00
	70	1001,00	,000	1001,00	1001,00
	80	1001,00	,000	1001,00	1001,00
	90	1001,00	,000	1001,00	1001,00
	=	1001,00	,000	1001,00	1001,00

a. Segregation? = fals, betterworld? = false, int_patch? = true

OLAP-Würfel *(F3)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	4,46	,076	4,00	4,40
	70	4,86	,098	5,00	4,77
	80	5,63	,120	5,00	5,51
	90	121,95	5,615	114,00	109,93
	=	34,23	2,896	5,00	10,62

a. Segregation? = fals, betterworld? = true, int_patch? = false

OLAP-Würfel *(F4)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	1001,00	,000	1001,00	1001,00
	70	1001,00	,000	1001,00	1001,00
	80	1001,00	,000	1001,00	1001,00
	90	580,59	39,760	469,00	409,89
	=	895,90	13,458	1001,00	800,74

Tabelle 15: Szenario 4 – OLAP-Würfel - false

In der untersten Tabelle (F4) enden die Simulationen in aller Regel nie. Die Funktionalität von ***betterworld*** und ***intelligent_Patches*** arbeiten gegeneinander. Mit steigender Dichte wird es schwierig, neue intelligente Patches zu finden. Daher kann ***betterworld*** mehr Einfluss erzielen.

In Tabelle 16 habe ich ***Segregation=true*** gesetzt.

Erkennbar ist das an der deutlich geringer ausfallenden Anzahl an ***Ticks*** bis zur Segregation.

Die oberste Tabelle (T1) ist mit dem Ergebnis der ursprünglichen Simulation aus der NetLogo Bibliothek identisch

Segregation = true

OLAP-Würfel *(T1)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	70,58	,856	69,00	70,10
	70	104,17	1,172	105,00	103,51
	80	179,34	2,374	176,00	177,83
	90	288,16	5,110	278,00	284,05
	=	160,56	4,426	136,00	138,36

a. Segregation? = true, betterworld? = false, int_patch? = false

OLAP-Würfel *(T2)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	7,19	,116	7,00	7,10
	70	10,02	,121	10,00	9,95
	80	17,58	,207	17,00	17,47
	90	57,89	2,832	48,50	53,95
	=	23,17	1,243	13,50	16,06

a. Segregation? = true, betterworld? = false, int_patch? = true

OLAP-Würfel *(T3)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	4,46	,076	4,00	4,40
	70	4,86	,098	5,00	4,77
	80	5,63	,120	5,00	5,51
	90	121,95	5,615	114,00	109,93
	=	34,23	2,896	5,00	10,62

a. Segregation? = true, betterworld? = true, int_patch? = false

OLAP-Würfel *(T4)*

		Mittelwert	std. Fehler des Mittelwertes	Median	geom. Mittel
steps	60	5,14	,096	5,00	5,05
	70	5,30	,093	5,00	5,22
	80	5,94	,126	6,00	5,82
	90	312,56	41,115	77,50	130,38
	=	82,24	12,214	6,00	11,89

a. Segregation? = true, betterworld? = true, int_patch? = true

Tabelle 16: Szenario 4 – OLAP-Würfel true

Die zweite und dritte Tabelle (T2,T3) stellen ***betterworld*** und ***intelligent-patches*** gegenüber. Es ist deutlich erkennbar, dass für geringe und hohe Dichten ***intelligent-patches*** nicht schnellstmöglich segregieren. Bei hoher Dichte ist die Anzahl der intelligenten Patches zu gering.

Bei geringer Dichte ist die Anzahl der freien Plätze weniger und die Anzahl der unglücklichen Turtle größer. (Bei zufälliger Verteilung zum Simulationsstart.) Ein weiterer Grund für die Überlegenheit von ***betterworld*** ist, dass jeder Turtle jedes Feld „selbst" prüft, d.h. die gewonnen Informationen sind aktuell. Intelligente Patches werden einmal pro Tick bestimmt, daher kann nur diese Anzahl an intelligenten Umzügen erfolgen.

Anmerkung: NetLogo definiert, dass kein Turtle alle Patches „selektieren" darf. Daher ist es den Turtle unmöglich alle Patches zu aktualisieren.

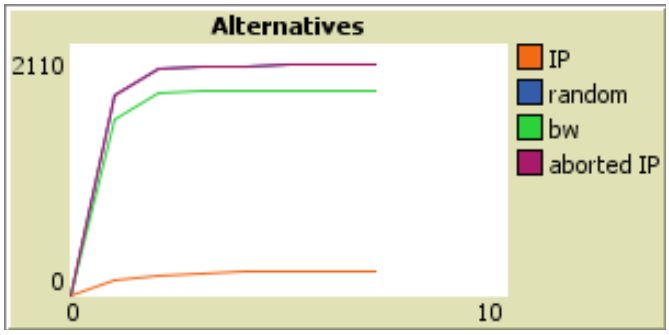

Abbildung 17: Szenario 4 - Alternativen Wahl

In der untersten Tabelle (T4) wirken beide Effekte simultan. Nach Auswertung der Graphen wirkt ***betterworld*** stärker als ***intelligent_patches.*** (Siehe Abbildung 2)

4.4.5 Konklusion

Meine erwarteten Veränderungen sind eingetreten. ***betterworld*** führt im allgemeinen zur schnellsten Segregation. Intelligente Patches können die Segregation stoppen. Werden beide Funktionen kombiniert genutzt, kann weder Segregation noch Integration bestmöglich erreicht werden.

4.5 Szenario 5 – Ökonomische Variation

In diesem Szenario untersuche ich die zwei verschiedene Prozesse der Segregationsbildung:

- **Aktive Segregationsbildung** – Der freiwillig angestrebte Umzug führt dazu, dass der Turtle Zufriedenheit erreicht.
- **Passive Segregationsbildung** – Entmischt sich ein Beobachtungsgebiet aktiv, so entsteht eine neue Segregation passiv. (Der aktive Wegzug beliebiger Turtle führt zur passiven Segregation im vorherigen Betrachtungsgebiet.)

Mit diesem Szenario versuche ich die ökonomischen Randbedingungen der Realität zu simulieren. Mit der Adaption einer einfachen ökonomischen Wirklichkeit zeige ich aktive und passive Segregation auf. Dabei setze ich Mietausgaben (fee) und Einkommen (money) als Zusatzeigenschaften in die Simulation ein. Ebenso werden beiden Parameter zur Laufzeit variabel sein, d.h. die Mietpreise sowie das Einkommen steigen während der Simulation dynamisch an. Einnahme und Ausgabe können von mir individuell initialisiert werden.

Aktiv kann der Turtle einen neuen Platz wählen. Er ist jedoch bei der Platzsuche durch sein Einkommen passiv eingeschränkt.

4.5.1 Konzeption

Die Turtle erhalten zwei neue Variablen: ***money*** und ***upgrowth. money*** enthält eine gamma-verteilte[17] (Abbildung 18) Zufallszahl mit einem Wert zwischen 0 und 25000. Zwecks Harmonisierung werden Ausreißer größer 25000 auf 25000 abgerundet. Die Patches erhalten eine variable ***fee***. Diese enthält eine gleich-verteilte Zufallszahl zwischen

17 Vereinfachte simul. Gesellschaft. (Wenig Arme, breite Mitte, wenige Reiche) Siehe: Abbildung 18

dem definierbaren Startwert ***society*** und dem durch die Anzahl der unterscheidbaren Mietpreise berechenbaren Endwert (***society*** + ***patch_colors***). Die Veränderung von ***society*** erlaubt es unterschiedliche Verläufe für Einnahme und Ausgabe zu generieren. ***money*** wird in jedem *tick* um einen prozentualen Wert ***upgrowth*** gesteigert. ***fee*** um den kumulierten ***upgrowth***. Im Unterschied zu den vorherigen Simulationen kann der Turtle in dieser Simulation nur dann umziehen, wenn ***money*** abzüglich ***fee*** positiv bleibt.

Die Simulation soll keinesfalls versuchen, die ökonomische Realität nachzubilden. Im Gegensatz zu Flávia F. Feitosa, die in ihrer Dissertation „MASUS: A Multi-Agent Simulator for Urban Segregation" die Situation auf Basis von empirischen Daten nachbilden konnte:

> ***„Despite the increasing attention towards ABM models of segregation, there have been only a few examples of models that rely on empirical data and methods."***
> ***[Feitosa (short Version), 2009] Seite 2***

Im Gegensatz dazu habe ich die Verteilungen und Startparameter selbst festgelegt, um aktive und passive Segregationsbildung zu simulieren.

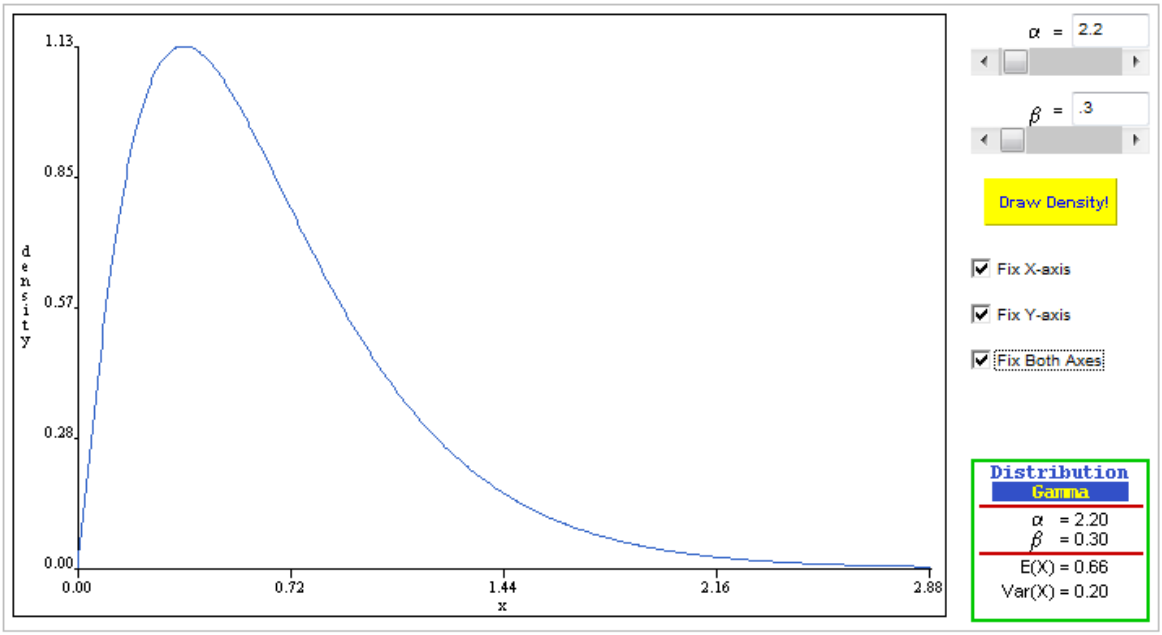

Abbildung 18: Szenario 5 - Gamma-Verteilung der Einkünfte

4.5.2 Implementation

Nach der „Geburt“ werden den Turtle-Variablen ***money*** und ***upgrowth*** Werte zugewiesen. Ebenso werden zu Simulationsbeginn den Patchfeldern zufällig Farben harmonisiert[18] zugewiesen. Diese Farben dienen der visuellen Unterscheidung sowie der Bestimmung des Mietpreises. Die Funktion ***monetary*** kapselt die Berechnungen zur Bestimmung der ***money***-Variablen. Zu Beginn von ***monetary*** wird überprüft, ob der Turtle seine aktuelle Miete zahlen kann. Ist das nicht der Fall, wird er unbeweglich und muss auf dem Platz verharren.

Abbildung 19 zeigt den programmatischen Ablauf der Entscheidungsfindung. Nachdem die Miete gezahlt wurde, wird diese um die globale Steigerungsrate erhöht. Ein Turtle kann kein negatives Kapital erreichen. Grundsätzlich handelt ein Turtle innerhalb von fünf Aktionen bedingt-rational:

Seine oberste Prämisse ist es, günstig und zufrieden zu leben. Gelingt ihm das nicht, ersetzt günstiges Wohnen seine oberste Prämisse. Falls keine dieser Prämissen gewählt werden kann, wählt der Turtle einen

18 Eine harmonisierte Randomisierung weist stadtteilähnliche Strukturen auf.

neuen und zufriedenstellenden Platz mittels ***intelligent-patches***.

→ ***Aktive Segregation***

Die letzte Wahl ist eine zufällige Bestimmung des neuen Ortes unter Risiko. (Unzufrieden zu bleiben und oder mehr Miete zu zahlen) bzw. den ursprünglichen Platz beizubehalten.

→ ***Passive Segregation***

Die Risikoentscheidung kann mittels ***risk_val*** (Siehe Tabelle 17) beeinflusst werden.

Die fünfte und letzte Möglichkeit ist ***unmoveable***. Besitzt der Turtle nicht ausreichend Kapital, um einen neuen Platz zu wählen, verbleibt er auf seinem alten Platz.

Hinzugefügte Elemente

Bezeichnung	***Typ***	***Erläuterung***
Eingabe		
patch_colors	*Slider*	*Bestimmt die Anzahl der Patchfarben ≙ Anzahl der unterschiedlichen Mietpreise.*
min-percent-unhappy	*Slider (%)*	*Prozentualer Anteil an der Gesamtbevölkerung. Die Unzufriedenheit bleibt, die Simulation wird jedoch beendet.*
Neigh4?	*Chooser*	*Die Harmonisierung der Mietpreise kann in 4er-Nachbarschaften sowie 8er-Nachbarschaften berechnet werden.*
society	*Slider*	*Mietpreisfaktor; Beeinflusst den Startwert der Mietpreise.*
Ausgabe		
Min-unhappy	*Monitor*	*Minimale Anzahl der unzufriedenen Turtle zur Laufzeit.*
Distribution Patches	*Plot*	*Histogramm über die Mietpreisverteilung. Der rote Punkt zeigt die Anzahl der „zufriedenen" Turtle.*
Turtle classes with happy	*Plot*	*Histogramm über die Verteilung auf die Klassen. Der rote Punkt zeigt die Anzahl der „zufriedenen" Turtle.*
Money – Fee	*Plot*	*Diagramm zum Verlauf der Mietpreise sowie die Einnahmen der Turtle.*

Tabelle 17: Szenario 5 – Hinzugefügte Elemente

4.5.3 Flussdiagramm

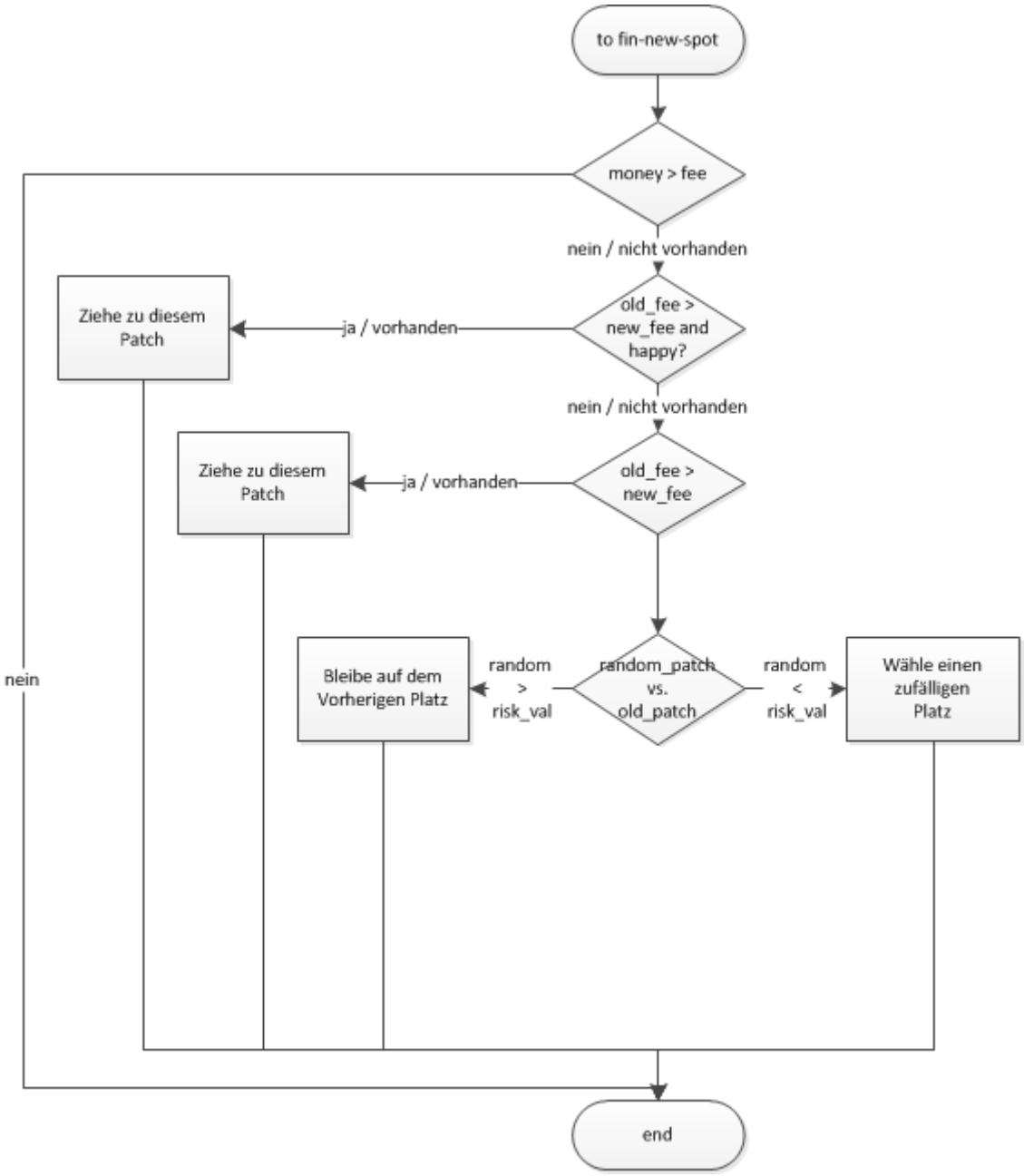

Abbildung 19: Szenario 5 - Entscheidungsbaum

4.5.4 Evaluation

Erwartete Veränderungen

Ich nehme an, dass die Turtle analog zum Entscheidungsbaum wählen. Wird eine Prämisse bewusst benachteiligt, z.B. die Steigerung der Dichte führt zur Senkung der Anzahl an intelligenten Patches, wird diese Alternative gemieden. Sobald eine gewisse Anzahl an Turtle ***unmoveable*** ist, erwarte ich eine Verlangsamung des Segregationsprozesses.

Eingetretene Veränderungen

```
1["comfortability_threshold" 6]
2["population_density" 90]
3["risk_val" 5]
4["neigh4?" true]
5["patch_colors" 2 10]
6["society" 2]
7["free-spaces-happy" "indifferent"]
```

Experiment 6 - Szenario 5: 2/10 Patches

Abbildung 20 zeigt ein Diagramm zu den getroffenen Entscheidungen der Turtle für das Experiment 6 Bedingt durch die geringe Anzahl an freien Alternativplätzen, konnten die besten Entscheidungsmöglichkeiten nur selten gewählt werden. Die ersten ***Ticks*** sind davon geprägt, dass die Turtle ihren Mietpreis durch Umzug reduzieren. Bedingt dadurch entstehen neue freie Plätze, die in den nachfolgenden Ticks als oberste Prämisse besetzt werden.

Im Verlauf kann die Alternative mit einer geringeren Miete und einem garantiert zufriedenstellenden Platz (schwarze Linie) zwar kurzzeitig eine beachtliche Steigung erreichen, flacht jedoch in der Folge ab.

Die orangefarbene Linie protokolliert die Anzahl der insolventen Turtle, die keine Alternativen wählen können. Der starke Anstieg führt meist dazu, dass die Anzahl der unzufrieden Turtle identisch mit der Anzahl

der unbeweglichen[19] Turtle ist. In diesem Fall endet die Simulation.

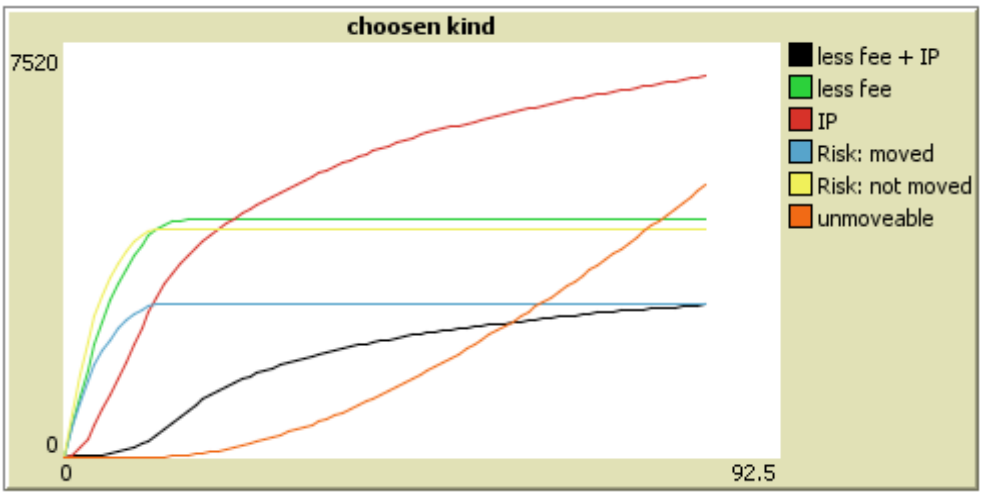

Abbildung 20: Szenario 5 - Diagramm Entscheidung

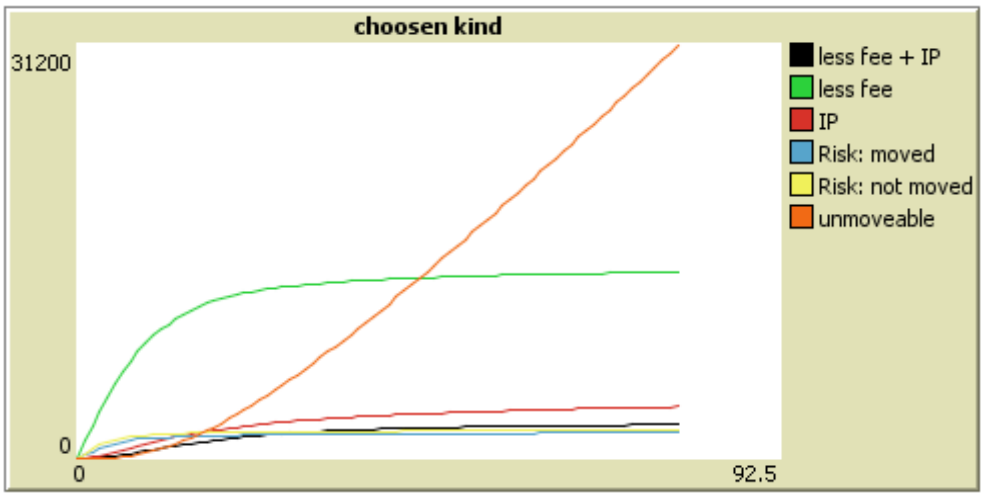

Abbildung 21: Szenario 5 - Diagramm Entscheidung, 10 Patches

Die zur Abbildung 20 und Abbildung 21 gehörigen Experimentdefinitionen unterscheiden sich im Aufbau, bis auf die Variation der Patchanzahl von 2 auf 10, nicht. Die Ergebnisse der beiden Simulationen sind jedoch stark unterschiedlich. Einzig die Verläufe von „less fee“ und „unmoveable“ sind ähnlich. Alle anderen Alternativen werden im unteren Experiment nahezu niemals gewählt. Bedingt durch die geringer Anzahl an optimalen Patches sinken die Alternativen mit ***intelligent_patches*** im unteren Verlauf. Nach gleicher Anzahl an Ticks sind im unteren Experiment deutlich mehr Turtle insolvent.

19 Unbeweglich im Sinne von insolvent.

4.5.5 Konklusion

Meine Erwartung, dass die Turtle ihre Intelligenz nutzen und sich analog verhalten, hat sich bestätigt. Mit der Veränderung der Preisstruktur agieren die Turtle differenziert. Entsprechend der Parameterwahl konnte ich aktive sowie passive Segregationsprozesse nachbilden.

4.6 Szenario 6 – Populationsaustausch

Die oben beschriebenen Szenarien beinhalten keinerlei Fluktuation. Die Verteilung der Turtle innerhalb der Klassen bleibt über die gesamte Simulationsdauer konstant. Eine realitätsnahe Simulation sollte jedoch auch eine dynamische Variabilität in den Klassen ermöglichen. Meine Intention in diesem Szenario ist, die drei verfügbaren Zustände (Nachbar ist rot, Nachbar ist grün, kein Nachbar) um einen weiteren Zustand zu ergänzen. Dieser Nachbartyp verhält sich in vier unterschiedlichen Weisen gegenüber den ursprünglichen Populationen. Dabei verändert sich die Einstellung der ursprünglichen Turtle nicht.

Ich möchte nicht versuchen, einzelne Turtle zu *bekehren, v*ielmehr ist das Ausscheiden und die Ersetzung durch tolerantere Turtle vorgesehen.

Im Unterschied zu den vorherigen Szenarien ändert sich dieses Modell dynamisch zur Laufzeit.

4.6.1 Konzeption

Die Eigenschaften der Turtle werden um das Alter (***age***) erweitert. Pro Simulationsschritt steigt das Alter der Turtle um eine Einheit an. Erreicht das Alter eine definierbare Schwelle, wird ein prozentualer Anteil entfernt. Die Anzahl der eliminierten Turtle wird im Folgeschritt durch gelbe (yellow) Turtle, die nicht umsiedeln werden, ersetzt. Die gelben Turtle bilden eine optisch und logisch unterscheidbare neue Klasse. Das Verhalten dieser Klasse gegenüber Rot und Grün ist in vier Anwendungsfälle einstellbar:

Tabelle 18 Szenario 6 - „all-like-yellow“ zeigt die Beziehungsverhältnisse des ersten Anwendungsfalls an. Gelbe Turtle verhalten sich immer positiv. Jede Klasse ist sympathisch mit gelb.

Negiert man das Verhalten (Tabelle 19Szenario 6 - „nobody-like-yellow“) der gelben Turtle, entsteht ein Verhalten analog einer dritten Rasse mit der einzigen Ausnahme, dass in diesem Szenario die neu hinzugefügten gelbe Turtle nicht umziehen können.

Eine Verschiebung der Verhältnisse wird in Tabelle 20 Szenario 6 - „red-like-yellow“ definiert. Dabei ist gelb mit rot sympathisch. In diesem Fall verschiebt sich das ursprüngliche Verhältnis von 50:50 Turtle. Die im Verlauf hinzugefügten gelben Turtle sind faktisch rote Turtle.

Et vice versa Tabelle 21 Szenario 6 - „green-like-yellow“. (Abbildung 22 visualisiert den zugehörigen Entscheidungsbaum)

„all-like-yellow“	Rot	Grün	Gelb
Rot	*Zufrieden*	*Unzufrieden*	*Zufrieden*
Grün	*Unzufrieden*	*Zufrieden*	*Zufrieden*
Gelb	*Zufrieden*	*Zufrieden*	*Zufrieden*

Tabelle 18 Szenario 6 - „all-like-yellow“

„nobody-like-yellow“	Rot	Grün	Gelb
Rot	*Zufrieden*	*Unzufrieden*	*Unzufrieden*
Grün	*Unzufrieden*	*Zufrieden*	*Unzufrieden*
Gelb	*Unzufrieden*	*Unzufrieden*	*Zufrieden*

Tabelle 19 Szenario 6 - „nobody-like-yellow“

„red-like-yellow“	Rot	Grün	Gelb
Rot	*Zufrieden*	*Unzufrieden*	*Zufrieden*
Grün	*Unzufrieden*	*Zufrieden*	*Unzufrieden*
Gelb	*Zufrieden*	*Unzufrieden*	*Zufrieden*

Tabelle 20 Szenario 6 - „red-like-yellow“

„green-like-yellow“	Rot	Grün	Gelb
Rot	*Zufrieden*	*Unzufrieden*	*Unzufrieden*
Grün	*Unzufrieden*	*Zufrieden*	*Zufrieden*
Gelb	*Unzufrieden*	*Unzufrieden*	*Zufrieden*

Tabelle 21 Szenario 6 - „green-like-yellow“

4.6.2 Implementation

Hinzugefügte Element

Bezeichnung	*Typ*	*Erläuterung*
Eingabe		
die-age	*Slider*	*Bestimmt das Alter, ab dem die Turtle sterben können.*
%-die	*Slider (%)*	*Prozentualer Anteil der sterbenden Turtle an der Gesamtbevölkerung.*
yellow_likes	*Chooser*	*Auswahl für das Verhalten der gelben Turtle. (Vergleiche Tabelle 18 - 21).*
Ausgabe		
Dying Turles	*Monitor*	*Anzahl der sterbenden Turtle pro tick.*
Yellow Turtles	*Monitor*	*Anzahl der gelben Turtle.*
Red's	*Monitor*	*Anzahl der roten Turtle.*
Green's	*Monitor*	*Anzahl der grünen Turtle.*
%-unhappy-red	*Monitor*	*%-ualer Anteil der nicht zufriedenen Turtle an Roten.*
%-unhappy-green	*Monitor*	*%-ualer Anteil der nicht zufriedenen Turtle an Grünen.*
Percent Unhappy	*Plot*	*Diagramm zum zeitlichen Verlauf der Zufriedenheit [aller, roter, grüner] - Turtle.*

Tabelle 22: Szenario 6 – Hinzugefügte Elemente

Die Hauptfunktionalität implementiere ich mit der neu-eingeführten Funktion ***live-and-die*** (Siehe Quellcodeauflistung 8). Zu Beginn meiner Simulation (bzw. pro tick) werden alle Turtle um eine Zeiteinheit altern. Zeile 3 prüft, ob ausreichend [n] Turtle („Dying Turtles“) zur Verfügung stehen. Falls die Prüfung zutrifft, leben in Zeile 3 n-Turtle ab (Zeile 5). In Zeile 6 (in Verbindung mit Zeile 8) werden dann die neuen, konnivenTen[20] Turtle eingesetzt. Anhand des gewählten Anwendungsfalls wird in ***update-turtle*** ein Pfad gewählt. Zu beachten ist, dass lediglich die Variablen ***similar-nearby*** und ***other-nearby*** unterschiedlich befüllt werden. Anhand dieser Variablen kann die Funktion ***happy-status*** den Zustand der einzelnen Turtle bestimmen.

20 Im Sinne von besonders duldend, nachsichtig.

```
1 to live-and-die
2   ask turtles [ set age age + 1 ]
3   if  count turtles with [ color = red or color = green
       and age > die-age and die-age != -1 ] >  ( %-die *
       number * 0.01 ) [
4     ask  n-of ( %-die * number * 0.01 )  turtles with [ (
       color = red or color = green)  and age > die-age ]
5     [ die  ]
6      ask n-of ( %-die * number * 0.01 )    patches with
  [ not any? other turtles-here ]
7     [
8       sprout 1
9         [
10              set color yellow
11              set age 1
12              set happy? true ]
13            ]
14    ] end
```

Quellcodeauflistung 8 - live-and-die

4.6.3 Entscheidungsbaum

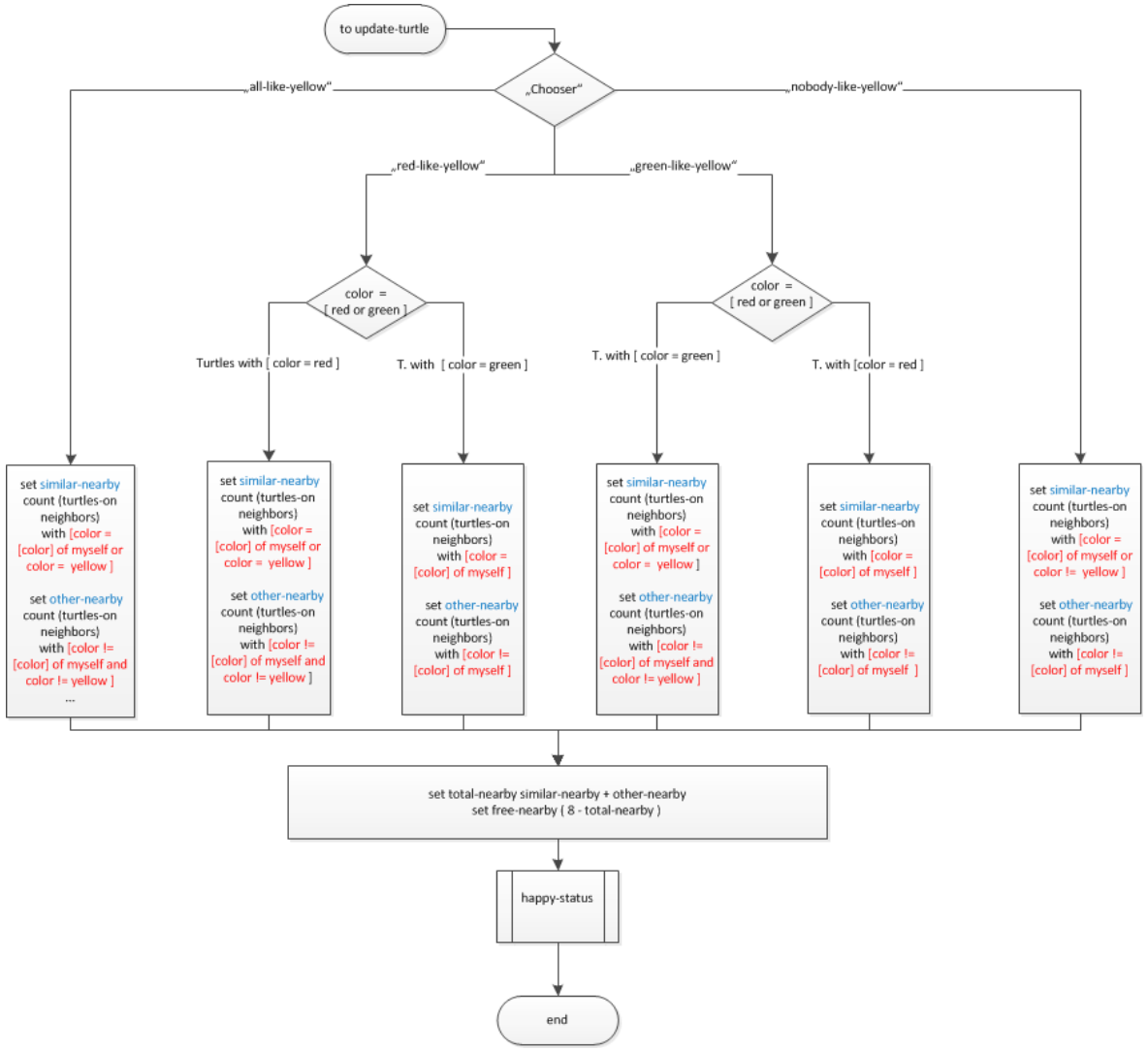

Abbildung 22: Szenario 6 - Populationsaustausch

4.6.4 Evaluation

Erwartete Veränderungen:

Anwendungsfall 1 - Szenario 6 - „all-like-yellow“:

Meiner Meinung nach wirken sich die gelben Turtle segregationsbeschleunigend aus, d.h. durch die Reduzierung der störenden Turtle kann die Segregation schneller eintreten. Ebenso können Simulationen, die niemals segregieren, mit steigender Anzahl der gelben Turtle segregieren. Folgende interessante Fragestellung entsteht dabei für mich:

Wie viele gelbe Turtle müssen der Segregation zugesetzt werden, damit eine Simulation, die ursprünglich nicht segregiert, segregieren kann?

Anwendungsfall 2 - Szenario 6 - „nobody-like-yellow“

Durch das Hinzufügen der Gelben steigt die Anzahl der ungleichen Turtle an. (siehe Abbildung 10 - Unähnlichkeitsverlauf) Erschwerend kommt hinzu, dass die eingesetzten Turtle immobil sind. Daher kann die Segregation sich nicht um diese Turtle herum bilden. Zumal die Turtle jedoch zufällig gesetzt werden und die Anzahl derer pro Runde ansteigt, erwarte ich keine Segregation.

Anwendungsfall 3 - Szenario 6 - „red-like-yellow“

Da die hinzugefügten gelben Turtle positiv mit den roten Turtle agieren, werden diese schneller zufrieden sein. Um diesen Effekt zu messen, habe ich den Plot ***percent-unhappy*** erweitert. Zusätzlich zu allen Turtle wird nun klassenspezifisch unterschieden. Ich gehe davon aus, dass die roten Turtle vor den Grünen gänzlich zufrieden sein werden.

Anwendungsfall 4 - Szenario 6 - „green-like-yellow“

Die simulierte Logik ist mit der aus dem Anwendungsfall 3 identisch.

Eingetretene Veränderungen

Anwendungsfall 1 - Szenario 6 - „all-like-yellow“:

Gegeben ist folgendes Szenario:

```
1["comfortability_threshold" 7]
2["population_density" 90]
3["free-spaces-happy" "indifferent"]
4["betterworld?" false]
5["stay_leave" 5]
6["random-seeds" "999"]
7["yellow_like" "all_like_yellow"]
8["die-age" 300]
9["%-die" 0.95]
```

Experiment 7 - Szenario 6: „all_like_yellow"

Wird die Simulation mit einem ***die-age*** von -1 (Funktionalität ist ausgeschaltet) gestartet, zeigt sich nach kurzer Zeit, dass keine Segregation eintritt. Wird im Verlauf der Simulation das Alter auf einen „sterbefähigen“ Wert angehoben, so steigt ***percent-similar*** stark an. Weiterhin sinkt die Anzahl der nichtzufriedenen Turtle. Die Diagramme in Abbildung 23 sind nach 376 ***Ticks*** festgehalten. Abbruchkriterien 0% unhappy-Rot

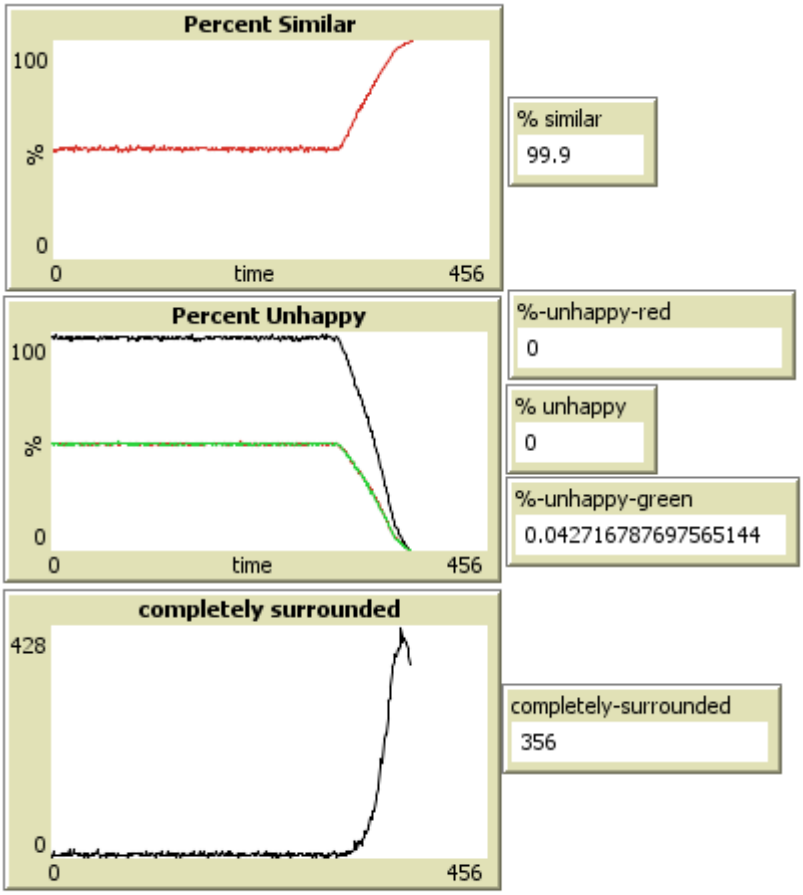

Abbildung 23: Szenario 6 - Diagramme zum Anwendungsfall: "all_like_yellow"

Anwendungsfall 2 - Szenario 6 - „nobody-like-yellow“

Geben ist folgendes Szenario:

```
1["comfortability_threshold" 7]
2["population_density" 90]
3["free-spaces-happy" "indifferent"]
4["betterworld?" false]
5["stay_leave" 5]
6["random-seeds" "999"]
7["yellow_like" "nobody_like_yellow"]
8["die-age" 300]
9["%-die" 0.95]
```

Experiment 8 - Szenario 6: „nobody-like_yellow“

Die ersten 300 ***Ticks*** verlaufen verständlicherweise (ticks < die_age) gleich mit denen des vorherigen Experiments. (Siehe: Abbildung 24) Ebenso fällt die Anzahl der nicht-zufriedenen Turtle im Verlauf des Diagramms ab. Diese Simulation konnte nach 471 ***Ticks*** beendet werden. Erst dann führt ausschließlich die hohe Anzahl an gelben Turtle (2332) zur Segregation. Das Diagramm ***Percent-Similar*** unterstreicht diese Beobachtung. Das anfängliche Einsacken ist ein Indiz für die Unbequemlichkeit, die dadurch entsteht, dass die gelben Turtle in die Simulation eintreten. Mit Verlauf der Simulation steigt dann jedoch die Anzahl der gelben Turtle bis zu rund 99.5% der Gesamtbevölkerung an.

Anwendungsfall 3 - Szenario 6 - „red-like-yellow“

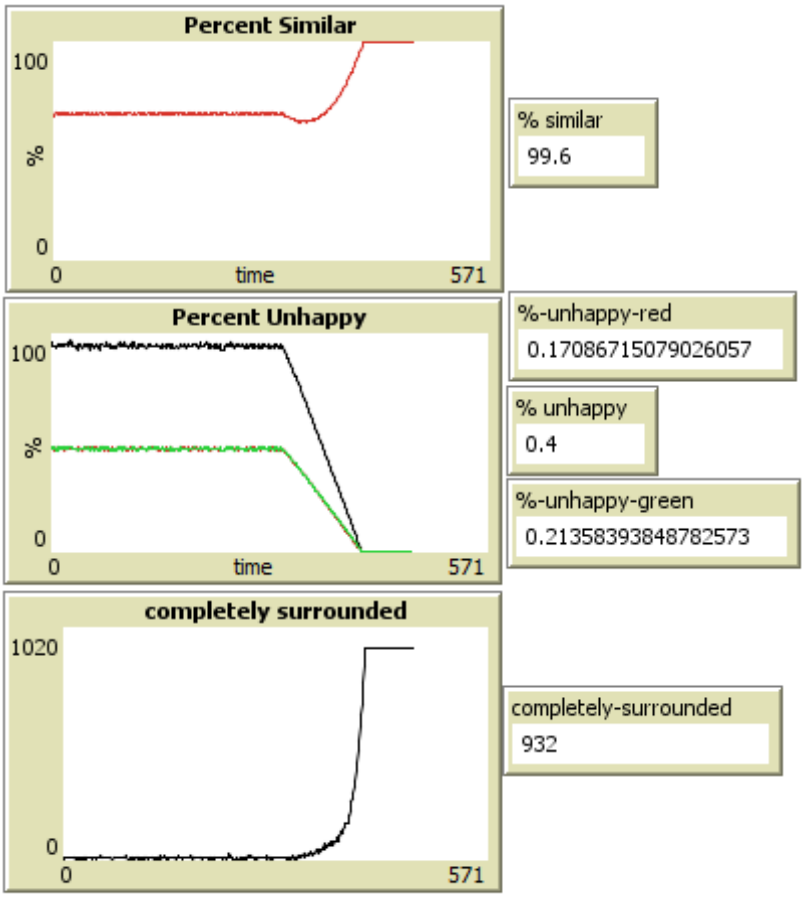

Abbildung 24: Szenario 6 - Diagramme zum Anwendungsfall: "nobody_like_yellow"

```
["comfortability_threshold" 7]
["population_density" 90]
["free-spaces-happy" "indifferent"]
["betterworld?" false]
["stay_leave" 5]
["random-seeds" "999"]
["yellow_like" "red_like_yellow"]
["die-age" 300]
["%-die" 0.20
```

Experiment 9 - Szenario 6: „red_like_yellow“

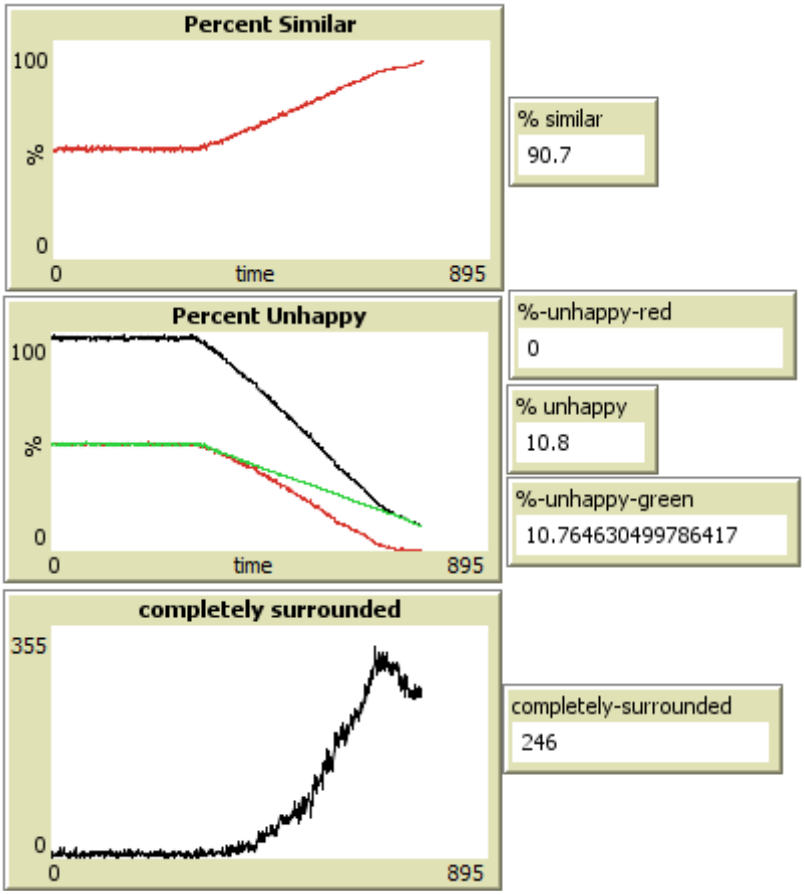

Abbildung 25: Szenario 6 - Diagramme zum Anwendungsfall: "red_like_yellow"

Die beiden nun folgenden Experimente unterscheiden sich von den beiden vorherigen darin, dass die Anzahl der sterbenden Turtle pro Runde geringer ist. Bedingt dadurch lässt sich das auftretende Phänomen besser visualisieren. Erwartungsgemäß sind die roten Turtle schneller zufrieden gestellt als ihre gelb-meidenden Nachbarn (Abb. 25).

Die global erreichten 90,7% ***percent_similar*** lassen sich dadurch erklären, dass die Segregation dann endet, sobald eine oder beide Klassen komplett zufrieden sind.

Anwendungsfall 4 - Szenario 6 - „green-like-yellow“

Entspricht dem Anwendungsfall 3 mit umgekehrten Farben.

4.6.5 Konklusion

Meine Ergebnisse dokumentieren, dass Simulationen aufgrund weniger „unpassender“ Turtle nicht segregieren. In diesen Fällen kann der variable Austausch aufschlussreiche Effekte bewirken. Weiterhin habe ich festgestellt, dass die Variante „**nobody-likes-yellow**“ zu einem nicht erwarteten Ergebnis führt. Die Veränderung der Gesellschaft durch die Gelben erzeugt kein endendes Szenario. Erst nach dem Tod nahezu aller nicht-gelben Turtle stoppt die Segregation.

5 Zusammenfassung

Die sechs ausgearbeiteten Szenarien, die ich thematisiert und programmiert habe, möchte ich an dieser Stelle übersichtlich zusammenfassen.

Die in Szenario 1 als *Basissimulation* entwickelte Erweiterung ***betterworld*** verbesserte die NetLogo Simulation. Die Frage nach der „besten" Anzahl an Versuchen konnte ich zwar nur bedingt beantworten; jedoch segregieren alle Simulationen signifikant schneller.

Im zweiten Szenario, dem *Multikulturalismus-Szenario*, erfolgte die Segregation später, nachdem ich die Anzahl der Kulturen erhöht habe. ***betterworld*** konnte diesen Effekt nur bedingt verhindern.

Der *Integrationsbeauftragte*, den ich in Szenario 3 mit der Absicht eingeführt habe, die Segregation aufzuhalten, hatte Erfolg. In diesem Fall endet die Simulation nie.

Die in Szenario 4 transformierte kollektive Intelligenz konnte sowohl integrations- sowie segregationsfördernd wirken. Die Einführung dieser sogenannten *intelligenten Patches* ermöglicht die Übertragung der Intelligenz von den Turtle auf die Patche.

Im *ökonomisch* geprägten Szenario 5 ist es mir gelungen, aktive sowie passive Segregation zu simulieren. Analog zum entwickelten Entscheidungsbaum wählen die Turtle bedingt rational und erzeugen in der Folge diese beiden Segregationsprozesse.

Im letzten Szenario 6 habe ich einen *variablen Austausch* der Populationen programmiert, der es erlaubt, die Gesamtbevölkerung durch eine neue und einstellbare Kultur zu verändern.

Die vorgestellten Szenarien beleuchten allesamt interessante Erweiterungen zu Schellings Modell. Diese sind geprägt durch die intensive Nutzung der Agenten-basierten Simulation.

6 Fazit

Mit dieser Arbeit ist es mir gelungen, einen schnellen Einstieg in die Simulation der Agenten-basierten Systeme zu finden. Dabei konnte ich die vorgestellten Vorteile der ABS in allen Szenarien nutzen. NetLogo ermöglichte es mir hierbei innerhalb kürzester Zeit kollektive Intelligenz zu programmieren. Durch unterschiedliche Evaluationen konnte ich die entstandenen Informationen individuell dokumentieren und aufschlussreich nutzen, wobei die größtenteils sehr zeit- und arbeitsintensive Beschaffung der Daten durch Experimente nicht außer Acht gelassen werden darf.

Weiterhin zeigt die Ausarbeitung, auf welch vielfältige Weise meine ausgewählten Erweiterungsmodelle das Schelling-Modell ergänzen. Sicherlich sind nicht alle Szenarien in der Realität direkt anwendbar, jedoch liefern die Erkenntnisse neue Anregungen für eine mögliche Weiterarbeit.[21]

Die von mir erstellten Evaluationen der Szenarien konnten einen kleinen Teil der möglichen Variablenkombinationen abbilden. Meine Auswahl der Tests zeigte hierbei die Basismöglichkeiten der Szenarien auf.

Als Ergebnis dieser Arbeit möchte ich zum einen die erfolgreiche Erweiterung der Agenten-basierten Simulation im allgemeinen festhalten. Zum anderen habe ich es geschafft, sechs unterschiedliche Simulationen zu programmieren, die jeweils als Grundlage weiterer Evaluationen und Simulationen dienen können.

21 Siehe dazu Kapitel 7 Ausblick

7 Ausblick

Im folgenden Kapitel gebe ich einen kleinen Ausblick auf weitere Möglichkeiten der Simulation und Evaluation.

Meine vorgestellten Szenarien nutzen lediglich die aktive Segregation. Dabei streben die Turtle nach einer zufriedenstellenden Nachbarschaft und ziehen aktiv in eine bessere Region um. Ich konnte die passive Segregation in den unterschiedlichen Szenarien erzeugen (Bsp. Szenario 5 – Ökonomische Variation), jedoch zielte keins primär auf die Erzeugung dieser Varietät ab. Passive Segregation ist das Ergebnis aus Simulationen, in denen der Umzug Restriktionen unterliegt. Sofern ein Turtle nicht umziehen kann, verbleibt dieser in einer Region. Trifft mindestens eine der Restriktionen auf mehrere der gleichen Art, so entsteht passive Segregation. Daher sollten in zukünftigen Simulationen mittels „**geplantem** Wegzug" einiger Turtle weitere Erkenntnisse entstehen können.

Eine zusätzliche interessante Weiterentwicklung lässt das Multikulturalismus-Szenario zu. In meinem Szenario bestehen zwischen allen Tribes gleiche Beziehungsstrukturen. *„Rote verhalten sich gleich zu Grünen, Orangen oder Gelben."* Eine Beziehungsmatrix würde unterschiedliche Präferenzen bzw. Abneigungen integrieren: *„Rote tolerieren mehr Gelbe und Orange als Grüne in der Nachbarschaft; sie meiden jedoch Blaue komplett."* Dabei könnten interessante Ergebnisse entstehen.

Eine weitere Möglichkeit wäre zu untersuchen, in wie weit die künstliche Anhebung bzw. die Reduzierung der ökonomischen Bedingungen Einfluss auf die Simulation hat. Meine Idee ist dabei, die Mietpreise zur Laufzeit zu variieren. Dadurch könnte eine staatliche Instanz simuliert werden, die aktiv Mietpreise steuern kann. Mittels dieser Erweiterung könnte in der Folge passive Segregation beeinflusst werden, da diese meist zu Integrationsschwierigkeiten führt.

8 Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, den 26. August 2011

Sebastian Geiermann

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Literaturverzeichnis

Strohmeier: Klaus Peter Strohmeier, Segregation in den Städten, 2006

Bertram: Jenny Bertram, Ethnische Segregation in Deutschland - freiwillig oder erzwungen?, 2008

Schelling, 1969: Thomas C. Schelling, Dynamic Models of Segregation, 1969

Gilbert, Troitzsch: Nigel Gilbert, Klaus Troitzsch, Simulation for the Social Scientist, 2005

Bonabeau: Eric Bonabeau, Agent-based modeling: Methods and techniques for simulating human systems, 2002

Stein: Sebastian Stein, Emergenz in der Softwareentwicklung –bereits verwirklicht oder Chance?, 2004

Wilensky 1997: Uri Wilensky, Information to the Segregation Simulation on Netlogo, 1997

Feitosa (short Version), 2009: Flávoa da Fonseca Feitosa, Urban Segregation as a Compley System: An Agent-Based Approach, 2009

12 Weitere Quellen

[Difu] http://www.difu.de/publikationen/difu-berichte-12006/segregation.html