Philipps Universität Marburg

Agentbasierte Modellierung

Inhaltsverzeichnis

Christoph Reudenbach

Erstellt: 15.04.09, Update: 09.11.23
Netlogo_Challenge1.odt

1 Einführung

Das Ableiten wissenschaftlicher Konzepte, bzw. allgemeiner Ideen in eine angemessene Modellstruktur ist ein komplexer und nicht schematisch zu erlernender Vorgang. Die Zusammenhänge der realen Welt in der Regel so komplex, dass sie nur in einer generalisierten Form verständlich darstellbar oder analysierbar sind. Schon im Alltag konstruieren wir kontinuierlich sogenannte mentale oder kognitive Modelle zur Vereinfachung unserer Weltwahrnehmung (Rasch 2006). Dieser Umstand ist auch in den Wissenschaften bekannt. So notiert der Physiker Bridgman 1927: *" believe that the model is a useful and indeed unescapable tool of thought, in that it enables us to think about the unfamiliar in terms of the familiar"* (Bridgman 1927), während 1972 der Modellierer Rivet schlicht behauptet: *"The History of mankind is the history of model building."* (Rivett 1972).

Die Perzeption und Interpretation der „realen Welt" sowie die Entwicklung geeigneter Strategien für den praxistauglichen Umgang mit dieser Welt, findet mit dem Hilfsmittel der Abstraktion und Kommunikation (= Modellbildung) statt. Die Strategien der Abstraktion sind widerstreitend und vielfältig, da Kontextabhängigkeit und Zielsetzung des Abstrahierenden einen wesentlichen Einfluss auf die Resultate ausüben ("Methode Götterblick" s.a. Eckmüllner 2007). Das heißt die gewählte Abstraktion der (räumlichen) Welt ist, wissenschaftlich betrachtet, bestenfalls nachvollziehbar und transparent aber niemals wahr. Auch garantiert die logische Validität der Abstraktion weder die Gültigkeit abgeleiteter noch allgemeiner Aussagen. Ob das konstruierte Modell der Wirklichkeit entspricht, also richtig ist, lässt sich daher nicht beweisen. Bestenfalls lässt sich die Gültigkeit für den definierten Zweck nachweisen, nie aber Wahrheit (Bossel 2004).

Prinzipiell kann man das Erlernen und Implementierenden von Werkzeugen zur Implementierung von Modellkonzepten in bottom up bzw. top down Ansätze unterscheiden. Als bottom up Ansatz kann das schrittweise Erlernen der Modellierungssprache (in diesem Kurs Netlogo) durch Implementierung konzeptioneller Modellideen (z.B. positive oder negative Rückkopplungsschemata etc.) bezeichnet werden. Dieser Ansatz hat den Vorzug, schrittweise elementare Bausteine zur Verfügung zu stellen um so dann zu guter Letzt auch komplexere Modelle erstellen und verstehen zu können. Er ist mit dem konventionellen Ansatz eine Fremdsprache durch systematisches Lernen von Vokabel und Grammatik zu erlernen vergleichbar.

Der umgekehrte Weg top down hingegen verfolgt eher den Ansatz „Spring ins kalte Wasser und schwimm' (oder geh' unter)" oder um beim zuvor angeführten Beispiel zu bleiben, dem Erlernen einer Fremdsprache durch sprechen und interagieren in dieser Sprache von Beginn an.

Beide Konzepte haben didaktisch und pragmatisch bekanntermaßen Vor- und Nachteile. In unserem Fall ist der Versuch mit einem existierenden recht komplexen Modell zu beginnen, der Erfahrung geschuldet, dass mit dem bottom up Ansatz unverhältnismäßig viel Zeit benötigt wird um erste inhaltlich auch nur einigermaßen interessante Modelle verstehen und entwickeln zu können.

Für die *geographische Lehrpraxis* erscheint mir ein kombinierter Ansatz von bottom up und top down als ein geeigneter Weg um effizient sowohl die inhaltlichen wie technischen Kompetenzen zu erwerben. Daher wird im zweiten Übungsteil auch intensiv der bottom up Ansatz verfolgt.

2 ChallengeOne – “Top of the World”

2.1 Bottom Up oder wie erstelle ich ein Modell in Netlogo?

Modelle sind -wie bereits wiederholt diskutiert wurde- spezifische, vereinfachte Ausschnitte der Wirklichkeit. Diese erste Programmierübung **ChallengeOne soll anhand eines einfachen Beispiels vermitteln, wie dieser recht komplexe Vorgang praktisch zu realisieren ist. Zunächst geht es um die Motivation und Fragestellung einer Modellentwicklung. Nach Festlegung einer Zielsetzung geht es dann vorrangig um die Frage „Wie setzte ich meine Frage in Regeln oder Algorithmen (formale Lösungsbeschreibungen) um?“ und schließlich gilt es die drückende Frage „Wie programmiere ich das Ganze?“ zu beantworten. Leider ist es nicht nur zu Beginn einer „Modellierer-Laufbahn“ schwierig diese Ebenen zu trennen und in effizienter Weise zu bearbeiten. Es gibt also kein Patentrezept.Die Übung ChallengeOne versucht eine aus meiner Sicht zielorientierte Vorgehensweise aufzuzeigen. Lernziel ist es durch geschicktes Anpassen von Beispielmodellen (=Netlogo-Skripten) und geeigneten Anpassungen ein eigenes Modell zu erstellen und es schrittweise zu erweitern und verbessern. Dabei ist es, zumindest zu Beginn, nicht zwingend erforderlich den Programmcode vollständig zu verstehen (das kommt hoffentlich mit der Übung...), sondern zu folgende Frage beantworten zu können „Wie beurteile ich das Resultat des Programmcodes hinsichtlich meiner Zielsetzung?“ In der anschließenden Übung ChallengeTwo** werden diese Kenntnisse vertieft.

Beginnen wir also.

2.2 Schneller, besser, weiter, höher...

Motivationsgrundlage unseres Modellierungsvorhabens ist das alte Spiel: alle wollen nach Möglichkeit besser sein als die Anderen oder zumindest für sich das Beste herausholen – ganz nach dem in gewissen Kreisen berühmten Ausspruch von von Bülow „***Mit einem Worte: wir wollen niemand in den Schatten stellen, aber wir verlangen auch unseren Platz an der Sonne“***[1]. In unserem Falle ist der „Platz an der Sonne“ nicht Namibia oder Togo, sondern alle wollen frei nach Chris Stangl[2] und Kollegen die „seven summits“ packen – ohne jede weitere Motivation nur auf einem der höchsten Berggipfel[3] stehen.

Lernziele ChallengeOne

Verstehen, dass zufällige (stochastische) Raummuster natürlichen/anthropogenen Raummustern („Landschaften“) vergleichbar sind

Herausfinden, dass Raum das Verhalten der Turtles beeinflusst

Wahrnehmen, dass nicht der Gipfel das Ziel ist, sondern die Analyse und Interaktion mit der Nachbarschaft

Erkunden der Auswirkungen unterschiedlicher räumlicher Strukturen

Ziel dieser Übung ist es alle beweglichen Agenten (turtles) auf den höchsten (erreichbaren) Gipfel zu bekommen. So einfach es auf den ersten Blick erscheint, alle Turtles auf einen Berg zu bekommen, so viele unterschiedliche Abstraktions- und Implementierungs-Konzepte werden berührt. Was soll das heißen? Nun wenn keine genaueren oder klareren Angaben gemacht werden (als gerade geschehen) kann kaum von einer Zielsetzung geschweige denn von einem Modellkonzept gesprochen werden (vgl. Bossel 2004). Versuchen wir es daher mit einer etwas klareren Abstraktion. Die übergeordneten Lernziele sind

---

[1]Bernhard von Bülow in einer Reichtagsdebatte am 6. Dezember 1897.
Quelle: http://de.wikisource.org/wiki/Deutschlands_Platz_an_der_Sonne. Zugriff 05.05.2009

[2]Chris Stangl. http://readyfornature.com/magazin-artikel.pdf?no_cache=1&L=0&tx_ttnews%5Btt_news%5D=10881&tx_wccategorytree_pi1%5Bcategory%5D=29. Zugriff 05.05.2009

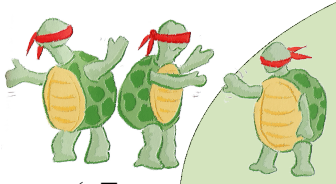
[3]Die Turtles Grafiken sind verändert und bearbeitet aus dem empfehlenswerten Buch von: Colella, V., E. Klopfer, and M. Resnick. 2001. Adventures in Modeling: Exploring Complex, Dynamic Systems with StarLogo. Teachers College Press.

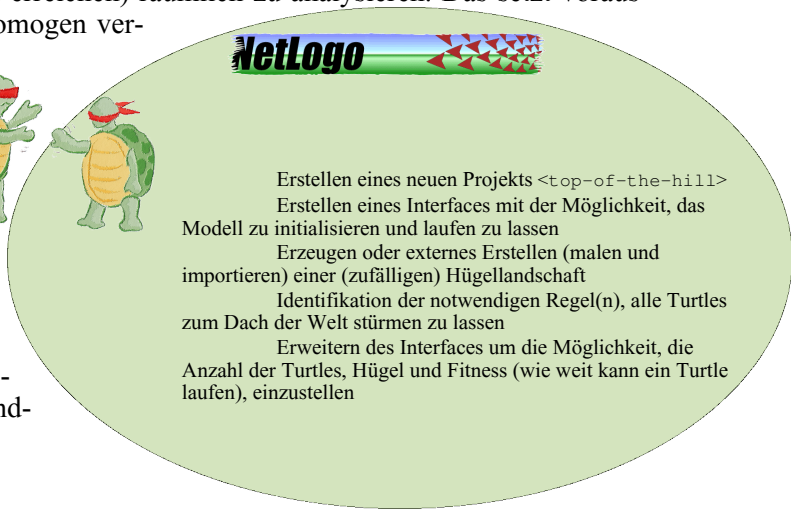
Christoph Reudenbach
Erstellt: 15.04.09, Update: 09.11.23
Netlogo_Challenge1.odt

im Kasten Lernziele ***ChallengeOne*** einzusehen

2.2.1 ChallengeOne - Beschreibung des Problems

Ziel unseres Modellierungsvorhabens ist es die Auswirkungen der behaupteten Neigung von Individuen ihren Vorteil zu optimieren (= den höchsten Punkt zu erreichen) räumlich zu analysieren. Das setzt voraus dass im Raum das zu optimierende Gut (Vorteil) inhomogen verteilt ist. Im Gebirge ist die Höhe inhomogen verteilt sonst wäre es nämlich eine Ebene. Das heißt der Raum besteht aus einzelnen Zellen die eine inhomogene Verteilung eines Parameters (Höhe, Nahrung, Geld, Liebe...) aufweisen. Weiterhin seien unsere Agenten (=Turtles) Einzelgänger die nur nach eigener Wahrnehmung des Raumes (also ohne Kommunikation mit anderen Turtles) das Ziel verfolgen, einen Ort (=Patch) möglichst hohen bzw. den höchsten Ort zu erreichen. Realisiert werden soll dieser Ansatz in einer hügeligen Landschaftsstruktur (=Hügellandschaft).





2.2.2 Netlogo Implementierungskonzept

Die Implementierung beginnt mit der Identifikation der zentralen Anforderungen aus der Problemstellung. Wie in dem ausführlich diskutierten Beispiel Anasazi sind diese jedoch immer noch unscharf formuliert. Was heißt „Hügellandschaft“? Wie sieht diese aus? Wie viele Hügel gibt es? Wie viele Turtles bevölkern die Welt „Top of the Hill“? Was heißt genau „die Turtles sind Einzelgänger“? Darüber hinaus müssen technische Fragen geklärt werden: Welche Reihenfolge in der Vorgehensweise ist sinnvoll. Was sind die notwendigen Einzelschritte etc.. Im ***Netlogo-Kasten*** sind in einer Art Pseudo-Programmcode konkrete Hinweise für einen sinnvollen Ablauf gegeben. Allerdings reicht dies noch nicht aus um los zulegen.

2.2.2.1 Die Welt

Es ist durchaus sinnvoll (wie bei jeder guten Schöpfungsgeschichte), zunächst die Welt zu erschaffen (=den Modellraum festzulegen). Mit dem Anlegen eines neuen Projekts wird eine Standard-Umgebung generiert (vgl. Netlogo Tutorial 1-3). Diese sollte nach dem Modellzweck (in diesem Fall Stadard / Default) angepasst werden (Tutorial 1->Controlling the View).

Hinweis: Da in Netlogo nicht direkt (wie bei einem rasterbasierten Grafikprogramm oder auch Starlogo) die Patches „angemalt“ werden können, muss man dies entweder programmieren oder es muss eine bereits existierende „Welt“ eingeladen werden[4].

Vorgehensweise: Für die hier gestellte Aufgabe erscheint die Programmierung einer einfachen Setup-Prozedur sinnvoll. Betrachtet man die Anforderungen müssen folgende Kriterien berücksichtigt werden:

Festlegung einer bestimmten Anzahl von Gipfelpunkten

Zuweisen von Höhenwerten für diese Punkte

Erzeugen von Talstrukturen zwischen diesen Gipfeln (unter Verwendung der Höhenwerte)

Visualisierung der „Landschaft“

Umsetzung: Natürlich kann man jetzt sofort ins Handbuch schauen und los programmieren. Einfacher und meistens ziel orientierter ist es sich anzuschauen was es an Problemlösungen gibt. In Netlogo werden eine Vielzahl von Beispiel-Modellen und Code-Schnipsel in der Model Library angeboten. Alle

[4]Hierzu stehen etliche Möglichkeiten bereit. Im Anasazi Modell werden Textdateien eingeladen. Es gibt die GIS Extension und es besteht die Möglichkeit, über Datei Import Pixelbilder als Hintergrundwelten einzuladen.

Christoph Reudenbach
Erstellt: 15.04.09, Update: 09.11.23
Netlogo_Challenge1.odt

sind ausgezeichnet dokumentiert und im Quellcode zusätzlich kommentiert. Als Newbie sollte man unbedingt hier stöbern gehen. Am besten (weil am einfachsten aufgebaut) beginnt man mit der Suche in den Code-Beispielen. Dort springt das *Hill Climbing Example ins Auge. Im Intro steht „This example shows how to make turtles climb hills -- or descend into valleys -- using the UPHILL, UPHILL4, DOWNHILL, and DOWNHILL4 commands. The same technique is useful for modeling any kind of creature that follows a gradient in its environment“. Offensichtlich eine ideale Vorlage für unser Problem. Also folgt die konkrete Aufgabe:*

Aufgabe: Analyse des Quellcodes des *Hillclimbing Examples.*

m Prinzip liefert das ***Hillclimbing Examples sowohl Welt als auch Turtles. Wir müssen nur einige Anpassungen vornehmen Nach der Analyse sollten die folgenden Fragen relativ leicht beantwortet werden können:***

- Wie funktioniert der Befehl *diffuse*? Können damit kontrolliert Höhenwerte erzeugt werden?
- An welcher Stelle des Scripts muss eine Variable eingesetzt werden um mit Hilfe eines Sliders die Anzahl der Turtles einstellen zu können?
- An welcher Stelle muss dies für die Anzahl der Hügel geschehen?
- Wie kann man über das Interface einstellen ob die Turtles ihren Weg markieren (*pen-up, pen-down*)?I

2.2.2.2 Die Turtles

Eigentlich geht es erst jetzt daran die Turtles in diese Welt zu setzten. Es gilt ein möglichst einfaches Regelwerk zu finden, das die Turtles veranlasst, stets nach den Höhen zu streben. Folgende Kriterien müssen berücksichtigt werden:

Festlegung einer bestimmten Anzahl von Turtles

Implementieren einer geeigneten Wahrnehmung des Kriteriums „Höhe des Patches“

Implementieren, mit Hilfe dieser Information den höchsten Punkt zu erreichen

Visualisierung

Vertiefung: Weiterhin steht das *Hillclimbing Example Pate. Allerdings wurde im Lösungsscript ChallengeOneBasic (Kap. 2.5) gewollt (u.a.) auf die Funktion uphill* verzichtet. Warum? Netlogo verfügt über eine oft unübersichtliche Vielzahl von bereits implementieren Funktionen und Primitiven. Anhand der Unterschiede von *uphill* und dem verwendeten *max-one-of neighbors* soll bereits zum Einstieg das der Netlogo Programmierung zugrunde liegende Konzept von Befehlen, Funktionen und Primitiven[5] besser verständlich gemacht werden. *Uphill* (Hillclimbing Example) ist ein Beispiel für eine komfortable Funktion weil es, wie in der Hilfe zu lesen ist, in einem Schritt mehrere Befehle ausführt:„[it] moves the turtle to the neighboring patch with the highest value ***for patch-variable“*. Das heißt *uphill*l identifiziert die (1) „Was ist benachbart? (2) Was ist der höchste Wert in der Nachbarschaft? Und (3) bewegt das Turtle dorthin. Da sich Agenten (Turtles) häufig entlang einer Gradientenkraft bewegen sollen ist es nahe liegend eine derartige Funktion in Netlogo verfügbar zu haben. Die gleiche Funktion ist allerdings nicht hilfreich, falls z.B. differenziert entschieden werden muss ob in Richtung des höchsten oder zweit höchsten (etc.) Wertes und ob dann 1, 2 oder *n Schritte weit gegangen werden soll oder ob etwa sofort zu diesem Punkt gegangen wird.***

Die Unterschiede zu *max-one-of neighbors* werden durch eine nähre Betrachtung des Beispielprogramms *Neighborhoods Example* deutlicher.Für solche Ziele ist der Befehl *max-one-of neighbors* geeigneter. Er analysiert beliebige

[5]Für eine gute Übersicht der Netlogo Struktur sei das Netlogo Tutorial von René Doursat genannt. Second Annual French Complex Systems Summer School http://iscpif.csregistry.org/Summer+School+2008 . Zugriff 2.5.2009.

Raummerkmale in der Nachbarschaft flexibler. *Max-one-of bzw. min-one-of* sucht nach dem höchsten bzw. niedrigsten Wert eines sog. **agentsets. Das Agenset kann eine beliebige (auch frei definierte) Auswahl (Gruppe) von Agenten sein (z.B. *neighbors* oder spezielle Gruppen von Turtles etc.).**

Neighbors ist, vergleichbar mit *uphill*, eine spezialisierte Funktion, die die Werte in der vollständigen Patch-Nachbarschaft (von Neumann'sche *neighbors4*, Moor'sche *neighbors*) mit einem Befehl„erfragt". Da die Nachbarn patches sind (=unbewegliche Agenten =ein agentset) kann *neighbors* einfach an *max-one-of* angehangen werden. Als Resultat liest sich der Befehl wie in der Menschensprache. Diese Vorgehensweise, sowohl beim Zusammensetzten von Befehlen als auch der schrittweisen Substitution allgemeiner Befehle/Funktionen durch komplexere Funktionen, ist LOGO-spezifisch und sollte verstanden und geübt werden.

Natürlich zeigt das ChallengeOneBasic-Beispiel, dass es nicht sinnvoll ist (wie es oben allerdings aus didaktischen Gründen durchgeführt wurde) Turtles und Patches während der Implementierung getrennt voneinander zu betrachten. Auch hier sei für das bessere Verständnis auf die Beispielprogramme hingewiesen In Kontext von ChallengeOne sind die folgenden Code Examples sowohl für spezifische Lösungen als auch Programmierkonzepte in Netlogo von besonderer Bedeutung:

Rasternachbarschaft erkunden: Neighborhoods, Moore & Von Neumann, Vision Cone

Visualisierung der Daten: Plot Axis

Direkte Kommunikation von Agenten: Communication-T-T, Communication-T-P

Fragen und Untersuchungen

Welchen Weg nehmen die Turtles? Sieht der Weg je nach unter 1.4 genannter Implementierungen unterschiedlich aus ?

Warum irren die Turtles immer weiter um die Gipfel herum? Wie kann man das stoppen?

Welche Variable (Grafikausgabe) sind geeignet den Systemzustand zu beschreiben?

Welche Variable müsste eingeführt werden um den Weg unter Punkt 2 untersuchen zu können?

Wäre das Ergebnis anders wenn sich die Turtles über den Weg verständigen könntenv

2.3 Gipfelstürmer und was nun?

Schneller, besser, weiter, höher... ist nur der Anfang. Es drängen sich eine Reihe von noch nicht gelösten Fragen und Optimierung auf. So z.B. ist die Visualisierung der Hügel und ihre Formgebung noch wenig ansprechend (oder gar naturnah...). Man könnte auch Informationen über den Zustand der Turtles sammeln und visualisieren oder speichern. Eingedenk der Einleitung kann z.B. die aktuelle minimale, mittlere bzw. maximale Höhe der Agenten eine Maßzahl für seinen Glückszustand sein (=je höher das Turtle je glücklicher). Auch ist bei der vorliegenden Implementierung die Frage ungelöst ob und wie es erreicht werden kann, dass alle Turtles irgendwann auf dem höchsten Gipfel stehen (bleiben). Im Kasten ***Fragen und Untersuchungen*** sind einige Anregungen für Modellerweiterungen und Fragen aufgelistet.

Im Script **ChallengeOneAdvanced** sind die meisten dieser Fragen bearbeitet. Es bietet eine gute Grundlage um sich mit der Programmierung und den unterschiedlichen Auswirkungen der Suchstrategien vertraut zu machen. Das Skript ist recht ausführlich kommentiert. Es sollte unbedingt durchgearbeitet werden da es die Grundlage für **ChallengeTwoBoatpeople bildet.**

2.4 Verwendete Strukturen und Befehle im Skript ChallengeOneBasic

Beispielhaft für das unter Kap.2.5 gezeigte Lösungsskript sind in Tabelle1 die Befehle aufgelistet. Unter beabsichtigte Aktion ist in normaler Sprache skizziert was mit dem Befehl erreicht werden soll während unter Rolle das unbedingt zu berücksichtigende Akteurskonzept von Netlogo dargestellt ist (vgl. auch Tutorial #2: Commands). Für eigene Aktivitäten um etwa die Bearbeitung der Anregungen umsetzten zu können, hilft nur das konsequente Nutzen des NetLogo Programming Guides und NetLogo Dictionaries. Ganz wichtig für das Erlernen von Netlogo ist natürlich das „Abschauen“ von Lösungswegen aus Modellen der Model Library. Gerade als Anfänger darf nicht verwirren, dass viele Befehle miteinander kombinierbar sind. So ist z.B. ***max-one-of neighbors*** aus den einzelnen Primitiven ***max-one-of*** und ***neighbors*** zusammengesetzt. Viele Primitive sind mit ***with*** erweiterbar (Achtung mit oder ohne Bindestrich erzielt völlig unterschiedliche Resultate). Also einfach im Command Center ausprobieren was bei Eingabe eines Befehls passiert – anders als in der Wirklichkeit kann nichts kaputtgehen. Eine aus meiner Sicht gelungene Zusammenstellung des Programmierkonzepts von Netlogo inkl. der Begriffsdefinition von Agenten, Prozeduren Funktionen und Primitive hat René Doursat für die Second Annual French Complex Systems Summer School erstellt. Hier werden alle notwendigen Konzepte klar strukturiert erläutert. Auf der Seite der Summerschool finden sich noch einige weitere interessante Beiträge unterschiedlicher Autoren[6].

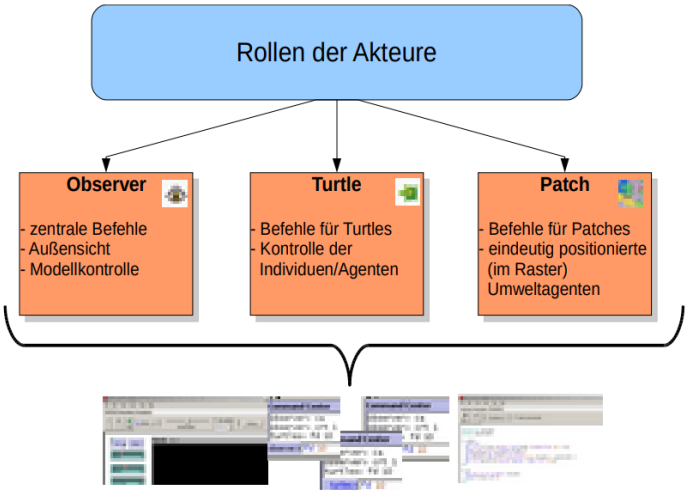


Abbildung 1: Akteurskonzept in Netlogo

Beabsichtigte Aktion	Befehl	Rolle
Alle Variablen löschen	***clear-all***	
Agenten irgendwas fragen, Agenten irgendetwas ausführen lassen	***ask***	
Beliebige Variable definieren	***set, let***	
Die Aktionsrichtung eines Akteurs festlegen	***face***	
Turtles auf dem Patch-Raster entstehen lassen	***sprout***	
Irgendwas wiederholen	***repeat***	
Werte auf dem Patchraster über Nachbarschaften verteilen	***diffuse***	

Tabelle 1: Hinweise zu den verwendeten Befehlen

2.5 Lösungen

Die Skripte **ChallengeOneBasic und ChallengeOneAdvanced sind auf github verfügbar. Das Basic Script implementiert die Grundfragestellung ohne viel Schnörkel. Im Advanced sind die meisten der im Kasten *Fragen und Untersuchungen*** angeregten Punkte bearbeitet. Vor allem sollte man sich mit der unterschiedlichen Wirkungsweise der Befehle und damit dem Raumverhalten der Turtles auseinandersetzen. Hierzu dient Import turtleposition. Auch die Programmierlogik ist nun erweitert um ifelse abfragen und die

[6]Second Annual French Complex Systems Summer School http://iscpif.csregistry.org/Summer+School+2008 . Zugriff 2.5.2009.

Möglichkeit Dateien auszuschreiben und wieder einzulesen.

2.6 Zusammenfassung

In dieser ersten Programmierübung sollte ein erster Einstieg in die Programmierung von Mensch-Umweltsystemen erreicht werden. Es wurde gezielt eine etwas zugespitzte Motivationsgrundlage gewählt um zu zeigen dass die wirklichen Fragen sich erst ergeben wenn eine erste Abstraktion implementiert ist und die Resultate interpretiert werden müssen. Gleichzeitig sollte an einem sehr übersichtlichen Beispiel das Grundgerüst der Netlogo Entwicklungsumgebung, diesmal Bottom up verständlich gemacht werden. In der Advanced Version kommen vor allem weitere technische Programmierfähigkeiten hinzu. Hier geht es um bedingte Abfragen und Dateneingabe bzw und -ausgabe.

2.7 Bearbeiten Sie bitte folgende Aufgaben:

- Dokumentieren Sie die beiden Modelle ChallengeOneBasic und ChallengeOneAdvanced gemäß der Vorgaben von Grimm (2006). Benutzen Sie die Vorlagen von Janssen (Janssen, 2008a).
- Führen Sie eine Sensitivitätsstudie mit dem Modell ChallengeOneAdvanced durch. Dokumentieren Sie diese gemäß Janssen (2008b).

3 Literatur

Axtell et al. 2002 Population Growth and Collapse in a Multi-Agent Model of the Kayenta Anasazi in Long House Valley. PNAS 99(3): 7275-7279.

Bartelme, N., 2005. Geoinformatik: Modelle, Funktionen. Berlin: Springer.

Bolzen 2009. Das Flüchtlingsdrama und die Hilflosigkeit der EU. Quelle: http://www.welt.de/politik/article3485575/Das-Fluechtlingsdrama-und-die-Hilflosigkeit-der-EU.html. Zugriff: 21.5.2009

Bossel, H., 2004. Systeme, Dynamik, Simulation – Modellbildung, Analyse und Simulation komplexer Systeme. Norderstedt/Germany: Books on Demand.Norderstedt/Germany, 2004

Bridgman, P.W., 1927. The Logic of Modern Physics. New york: Macmillan .

Dean J S, Gumerman G J, Epstein J M, Axtell R L, Swedlund A C, Parker M T, and McCarroll S. 2000 Understanding Anasazi Culture Change Through Agent Based Modeling in Dynamics in Human and Primate Societies: Agent Based Modeling of Social and Spatial Processes, T. Kohler and G. Gumerman (eds.), Santa Fe Institute. New York & London: Oxford University Press.

Dienelt 2006': Bootsflüchtlinge Ceuta, Melilla, Lampedusa, Malta, EU-Kommission Hampton Court http://www.migrationsrecht.net/nachrichten-asylrecht/410-bootsfluechtlinge-ceuta-melilla-lampedusa-malta-eu-kommission-hampton-court.html. Zugriff: 21.5.2009

Eckmüllner, O., 2007. Götterblick – oder was?. Östereichische Forstzeitung, 11, 16.

Grimm V. Berger U. Bastiansen F. Eliassen S. Ginot V. Giske J. Goss-Custard J. Grand T. Heinz SK. Huse G. Huth A. Jepsen JU. Jorgensen C. Mooij WM. Muller B. Pe'er G. Piou C. Railsback SF. Robbins AM. Robbins MM. Rossmanith E. Ruger N.(2006): A standard protocol for describing individual-based and agent-based models. Ecological Modelling. v198. 115-126. Ecological Modelling. Vol. 198, Issues 1-2, 2006, p. 115-126. http://www.bio.uib.no/inc/pdffiles/Pub/pub3126.pdf. Zugriff: 1.9.2007.

Gumerman G J, Swedlund A C, Dean J S, and Epstein J M (2003) The Evolution of Social Behavior in the Prehistoric American Southwest. Artificial Life 9: 435-444 #

Janssen, M., 2008a: ODD of Netlogo implementation of Artificial Anasazi. https://www.openabm.org/site/download/ArtificialAnasazi/version1/doc. Zugriff: 20.01.2009

Janssen, M., 2008b: NetLogo 4.02 Code of Replication of the well known Artificial Anasazi model that simulate the population dynamics between 800 and 1350 in the Long House Valley in Arizona. https://www.openabm.org/site/download/ArtificialAnasazi/version1/code. Zugriff: 20.01.2009

Lutterbeck 2006:Policy Migration in the Mediterranean. http://www.gcsp.ch/e/publications/Issues_Institutions/ME_Med/Academic_Papers/Lutterbeck-Med_Politics-March06.pdf
Rasch, T., 2006. Verstehen abstrakter Sachverhalte: Semantische Gestalten in der Konstruktion mentaler Modelle . Berlin: Wissenschaftlicher Verlag.
Rivett, P., 1972. Principles of model building. The construction of models for decision analysis. London: Wiley.
Tobler, W.R., 1970. A Computer Movie Simulating Urban Growth in the Detroit Region. Economic Geography, 46, 234. (Herunterladen)
Werlen, B., 1993. Gibt es eine Geographie ohne Raum? Zum Verhältnis von traditioneller Geographie und zeitgenössischen Gesellschaften.. Erdkunde, 47, 241. (Herunterladen)

Christoph Reudenbach
Erstellt: 15.04.09, Update: 09.11.23
Netlogo_Challenge1.odt