Eine virtuelle kontinuierliche Welt als Testbett für KI-Modelle

Günter Bruns[1], Daniel Polani[2] und Thomas Uthmann[1]
[1]Institut für Informatik, Johannes Gutenberg-Universität Mainz
[2]Institut für Neuro- und Bioinformatik, Universität zu Lübeck
polani@inb.mu-luebeck.de, uthmann@informatik.uni-mainz.de

1 Einführung

Virtuelle Umgebungen werden in der KI oft verwendet, um in einer wohldefinierten Umwelt kontrollierter Komplexität die Fähigkeiten künstlicher Agenten zu entwickeln und zu evaluieren. Sie sind immer dann von Nutzen, wenn standardisierte Testsuiten fester Struktur nicht ausreichen, wenn eine hardwaremäßige Realisierung problematisch ist (z.B. bei Untersuchungen zur Evolution von Morphologie [24], Sensoren [16] oder funktionalen Phänotypen [2, 22]) oder wenn sie in Hardware zu aufwendig ist (etwa bei Untersuchung von kollektivem Verhalten).

Eine Simulation kann hierbei eine konkrete Realisierung nicht ersetzen [9]. Sie muß stets dahingehend betrachtet werden, inwieweit die durch sie „nachgewiesenen" Eigenschaften einer Agentensteuerung auf andere simulative oder physikalische Szenarien übertragbar sind [10] und welche davon in konkreten Anwendungen genutzt werden können. Es geht insbesondere nicht um spezialisierte Lösungsmethoden, sondern um abstrahierbare Prinzipien. In [9] wird daher vorgeschlagen, Simulationen als *absichtlich* abstrahierte Versionen realer Systeme aufzufassen, als algorithmisierte Modelle im Sinne des physikalischen Modellbegriffs, wobei die Unterscheidung zwischen Modell und Simulation fallengelassen wird. Solche Simulationsmodelle liefern Voraussagen, die dann am realen System überprüft werden können. Insbesondere ist die Qualität einer Simulation nicht eine eindimensionale Größe, sondern hängt von vielen Faktoren ab, u.a. dem Anwendungszweck. Die erhaltenen Resultate für die Performanz von Agentensteuerungen müssen dementsprechend kritisch beurteilt werden.

Andererseits legitimieren diese Überlegungen die Betrachtung von Simulationsmodellen, die nicht den Anspruch erheben, konkrete physikalische Realisationen möglichst akkurat zu beschreiben, sondern die bestimmte, in der Realität relevante, Aspekte auf einem höheren Niveau untersuchen, bzw. von vorneherein in einer eigenen Modellrealität formulieren. Viele Modelle im Gebiet *Artificial Life* sind von dieser Art.

In diesem Bereich gibt es zum einen spezialisierte Werkzeuge zur Modellierung spezifischer Aspekte der Evolution und von Szenarien zu Artificial Life (z.B. TIERRA [21, 22], oder dessen Weiterentwicklung AVIDA [1, 3]), zum anderen findet man Werkzeuge für die Entwicklung von agentenorientierten Simulationsumgebungen im allgemeinen (etwa CBB [25] oder SWARM [13]).

2 Die XRaptor-Umgebung

2.1 Grundkonzept

Das hier beschriebene System XRaptor dient zur Simulation virtueller Agentenszenarien im Kontinuum des $\mathbb{R}^2$ oder $\mathbb{R}^3$. Es ermöglicht die Modellierung virtueller 3D-Simulationsszenarien und durch die erweiterbare OO-Architektur ist das Konzept von XRaptor allgemeiner als etwa das von Tierra bzw. Avida, dabei aber durch die höhere Spezialisierung effizienter als Swarm. XRaptor unterscheidet sich von den genannten Simulationsumgebungen auch durch die explizite Unterstützung des Konzepts von Agenten*teams*, die von unterschiedlichen Entwicklern implementiert werden und in einem Turnier gegeneinander antreten können [19]. Der Soccer Server des RoboCup-Projektes erlaubt eine ähnliche Teamorientierung [17], wobei er aber auf die Simulation von Roboterfußball eingeschränkt ist.

XRaptor simuliert Populationen von lokalisierten Agenten. Eine Motivation, XRaptor als explizites *Multi*agentensystem zu konzipieren, kam aus der Anforderung, ohne aufwendige Umweltmodelle eine komplexe Testumwelt für adaptive Verfahren zur Verfügung zu stellen; die Komplexität der Umgebung und der dafür erforderliche Grad an Adaptivität sind typischerweise ein Resultat der kollektiven Dynamik der Agentenpopulationen und nicht von detaillierten physikalischen Modellen. Solche Modelle können auf der C++-Implementationsebene aber auch erstellt werden.

Man kann das System auf drei verschiedenen konzeptionellen Anwendungsebenen nutzen: 1. als *Experimentator*, der die Simulations„physik“ verändert; 2. als *Teamentwickler*, der ein Agententeam bei *vorgegebener* Physik, jedoch mit *komplettem* Weltwissen, entwickelt, das dann schließlich gegen andere Agententeams antritt; 3. als *Turnierteilnehmer*, dessen Agenten nur über eingeschränkte Sensor/Aktor-Fähigkeiten verfügen. Wichtig ist insbesondere die Möglichkeit, Turniere zwischen unterschiedlichen Agententeams durchzuführen.

XRaptor erfüllt zentrale Anforderungen an ein Testbett (siehe etwa [4]) mit unterschiedlichen Schwerpunkten: *Externe Ereignisse* lassen sich einfach integrieren. Eigenschaften der Sensoren oder auch Parameter der Simulationswelt können sich über die Zeit ändern, was adaptive Kontrollmechanismen bevorzugt. Das Zeitmodell beinhaltet eine synchrone Abfrage und Ausführung der Agentenaktionen. Der Zeitverbrauch durch die Agentensteuerung wird in Form von *Zeitkosten* direkt in Lebensenergiekosten umgerechnet (s. Kap. 3). Ein Maß für die *Qualität einer Agentensteuerung* ist szenarioabhängig definierbar. Als Basismaß fungiert die Lebensenergie, das bei allen unseren Simulationen das Kriterium für den Erfolg eines Agenten ist.

2.2 Technische Aspekte

XRaptor ist in C++ für UNIX-Plattformen mit dem X-Window-System, Motif 1.2 sowie den 3D-Grafik-Systemen PEX oder OpenGL geschrieben [5]. XRaptor-Agenten besitzen eine Sensor-Aktor-Architektur, in die ein Steuerungsmodul eingefügt wird, das Sensorinformationen auswertet und daraus eine geeignete Aktivierung der Aktoren veranlaßt. Die Steuerung wird hierbei frei implementiert.

Die Agenten sind als C++-Klassen realisiert, und zu jedem konkreten Agententyp existiert eine entsprechende abstrakte Steuerungsklasse. Der Experimentator, der ein Szenario entwirft, muß die entsprechenden Agenten- und Steuerungsklassen, sowie Sensor- und Aktorklassen entwickeln und die dazugehörige Dynamik definieren. Bereits verfügbare Klassen können wiederverwendet bzw. spezialisiert werden. Eine konkrete Agentensteuerung wird dann durch die Teamentwickler

als Spezialisierung der entsprechenden abstrakten Steuerungsklasse in C++ implementiert und mit der Weltsimulation compiliert; die gesamte Simulation läuft innerhalb *eines* Prozesses. Da die individuellen Szenarien z.T. sehr unterschiedliche Untersuchungsaufgaben zum Gegenstand haben, ist der Rahmen für den Eingriff und die Protokollierung von Experimenten sehr allgemein gehalten. Die Benutzungsoberfläche stellt eine generische Textschnittstelle zur Verfügung, über die der Benutzer den Team- bzw. den Agentenklassen Kommandos frei definierbarer Syntax zusenden kann. Die zur Auswertung von Szenarien nötigen Daten werden von den Team- bzw. Agentensteuerungsklassen individuell protokolliert.

Präsentationsebene und Simulator sind durch ein *Model-View-Controller*-Konzept weitgehend getrennt. Die Effizienz der Darstellung wird durch eine *Level-of-Detail*-Verwaltung erreicht, die weiter entfernte Objekte mit geringerer Auflösung zeichnet als nähere. Für das Design der 3D-Objekte können externe Standardwerkzeuge eingesetzt werden (etwa AC3D, [7]), die VRML-Darstellungen erzeugen. Daraus generieren *Level-of-Detail*-Generatoren wie etwa LODESTAR [23] abgestufte Auflösungen, aus denen XRaptor die passende auswählt. Für ein System aus 285 Agenten und Objekten – 150 Früchten, 75 Fliegen, 60 Fledermäusen (zum Szenario siehe Kap. 3) – werden auf einer Sun Ultra 10 Workstation 20 Simulationsschritte pro Sekunde ausgeführt, das Grafiksystem kann 2–12 Bilder pro Sekunde dieses visuell durch ca. $2 \cdot 10^5$ Dreiecke beschriebenen Szenarios darstellen. In der Standardeinstellung werden je Bild 10 Simulationsschritte durchgeführt, was zu einer Gesamtleistung von ca. 1-2 Bildern pro Sekunde führt. Die neueste Fortentwicklung unter Verwendung von OpenGL und hardwarebeschleunigter Grafik erzielt 10–76 Bilder (typisch 25) pro Sekunde.

3 Simulationsszenarien

Dieser Abschnitt gibt einen kurzen Überblick über drei verschiedene Simulationsszenarien, die mit XRaptor realisiert worden sind. Die ersten beiden Szenarien wurden auch in Vorlesungszyklen über adaptive Methoden der KI eingesetzt.

Im ersten Szenario findet die Simulation in einem virtuellen 3D-Raum statt, in dem sich Agenten mit gewissen abstrahierten Eigenschaften von *Fliegen* und *Fledermäusen* frei bewegen können (Abb. 1). Jede Aktion wie etwa *Flügelschlagen*, *Sensoren abfragen*, auch Verbrauch von Rechenzeit für Berechnungen, kostet Energie (nähere Einzelheiten in [19]). Das zweite (2D-)Szenario ist im Bereich Artificial Life angesiedelt. Im Gegensatz zu Modellen für Schwarmverhalten, in denen eher generelle, individuen-unabhängige Bewegungsmuster untersucht werden [13], stehen in Modellen zu Ameisenstaaten oft lokal interagierende Agententypen im Vordergrund (s. [8] zur Selbstorganisation in Insektenkolonien oder [12] zu emergentem Verhalten in Ameisenstaaten). In unserem Szenario geht es um Analysen von Überlebensstrategien für Populationen ameisenähnlicher Agenten mit beschränkten Ressourcen (s. [20] für Details). Eine weitere Zielsetzung wird im dritten Szenario verfolgt: In der Biologie hat die Evolution von Sensoren bei der Entwicklung von Überlebensstrategien eine wesentliche Rolle gespielt (siehe z.B. [11]). Es liegt somit nahe, sich mit der Simulation dieses Phänomens ebenfalls genauer zu beschäftigen. So wurden in [15] „Ohren“ für den Khepera-Roboter entwickelt, ferner gibt es etliche Ansätze, die Sensorparameter genetisch codieren [6, 18]. In unseren Untersuchungen besitzen die Braitenberg-ähnlichen Agenten visuelle Sensoren („Augen“) mit verschiedenen evolvierbaren Parametern, so etwa *Augenunschärfe*, *Augenanzahl* oder *spektrale Empfindlichkeit*. Mit unseren Experimenten haben wir erste erfolgreiche Schritte in Richtung Simulation sensorieller Evolution unternommen [14, 16].

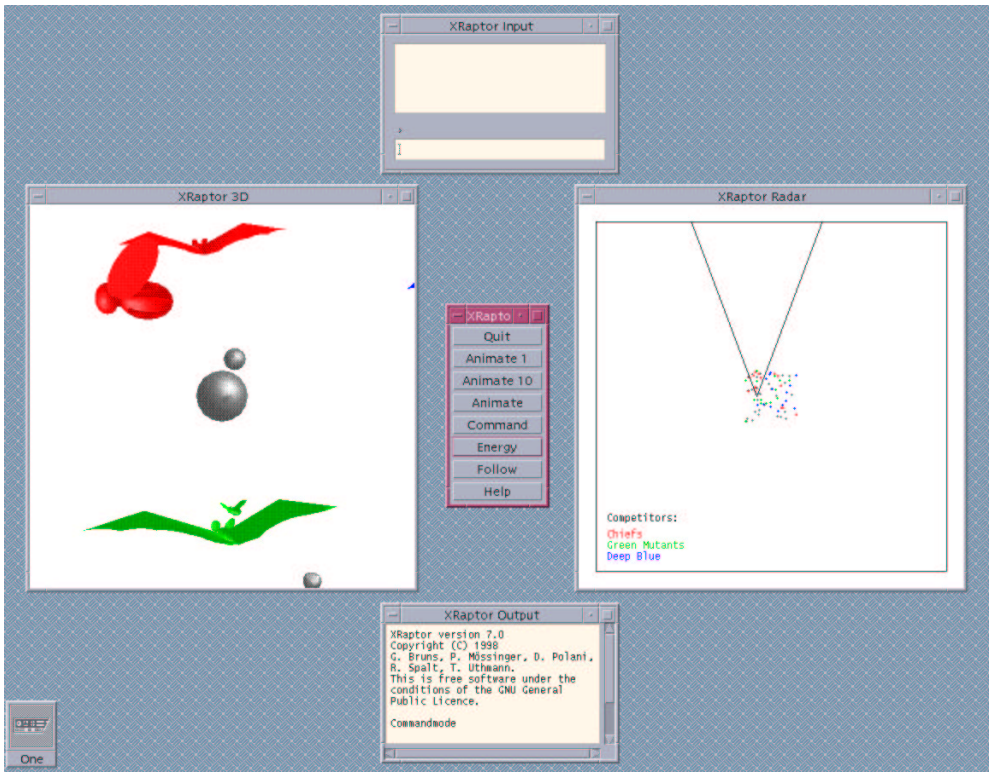

Abbildung 1: Blick auf die XRaptor-Oberfläche. Links das 3D-Sichtfenster in die Simulationswelt (mit zwei Fledermäusen, zwei Fliegen und drei Früchten), rechts eine Übersichtsanzeige (von oben; Sichtkegel des 3D-Fensters ist explizit eingezeichnet), oben ein Kommandoeingabefenster, unten ein Ausgabefenster, in der Mitte das Steuerfeld. Vgl. auch [19]

.

4 Zusammenfassung und Ausblick

XRaptor ist ein Simulationswerkzeug für kontinuierliche Multiagentenwelten. Die Architektur erlaubt eine flexible Erweiterung der repräsentierten Objekte, eine vom eigentlichen Simulationssystem unabhängige Erstellung der Darstellungsobjekte, sowie eine komfortable Integration von zu testenden KI-Steuerungen in eine Reihe von Simulationsszenarien.

XRaptor eignet sich besonders für Simulationen von kontinuierlichen *Multi*agentenwelten und für Turniere zwischen mehreren Agententeams, die von verschiedenen Entwicklergruppen stammen können. Nach unserer Erfahrung ist die Akzeptanz von XRaptor durch die transparenten Schnittstellen für die Agentensteuerungen in KI-Veranstaltungen des Hauptstudiums sehr gut. Die Simulationsumgebung unterstützt zwar nicht primär die detaillierte physikalische Modellierung einer virtuellen Umwelt, diese ist aber auf jeden Fall möglich. Aufgaben, die von den Agenten zeitaufwendige Deliberationsprozesse erfordern, sind für XRaptor nicht so gut geeignet, da diese in der jetzigen Version innerhalb eines Prozesses seriell abgearbeitet werden. XRaptor stellt eine „kontinuierliche“ Alternative zu den verbreiteten Gitterweltsimulatoren dar. Ist die Berücksichtigung von detaillierten Modellannahmen gewünscht, so wäre an ein entsprechend spezialisiertes Simulationswerkzeug zu denken (wie [12] für die realistische Simulation von Ameisenpopulationen). Für generischere Modelle wird man auf ein allgemeineres Baukastensystem (z.B. Swarm, [13]) zurückgreifen. In Planung sind gegenwärtig Konzepte zu einer

komfortableren Entwicklung und einem einfacheren Austausch unterschiedlicher Szenarien.

XRAPTOR ist unter der *General Public License* als freie Software verfügbar [5]. Über Rückmeldungen und Verbesserungsvorschläge würden wir uns freuen. Bei genügendem Interesse ist an die Einrichtung einer Mailingliste gedacht.

Danksagung

An dieser Stelle möchten wir allen danken, die bei der Entwicklung von XRAPTOR mitgewirkt haben: Alexandra Mark, Peter Mössinger, René Spalt, darüberhinaus den Teilnehmern der Kurse *Lernende Systeme* und *Multiagentensysteme* der letzten Jahre. Wir bedanken uns außerdem bei dem/der anonymen Reviewer/in für die konstruktiven und wertvollen Hinweise.

Literatur

[1] Adami, C. Handbook and Publications about Avida. `http://www.krl.caltech.edu/avida/`, 1997.

[2] Adami, C. *Introduction to Artificial Life.* Springer, 1998.

[3] Adami, C. and Brown, C. T. Evolutionary Learning in the 2D Artificial Life System "Avida". In *Proc. Artificial Life IV*, S. 377, MIT Press, 1994.

[4] Al-Badr, B. and Hanks, S. Critiquing the Tileworld: Agent Architectures, Planning Benchmarks, and Experimental Methodology. Technical Report TR 91-11-01, Dept. of Computer Science & Engr., Univ. of Washington, Seattle, 1991.

[5] Bruns, G., Mössinger, P., Polani, D., Schmitt, R., Spalt, R., Uthmann, T., and Weber, S. XRaptor — A Simulation Environment for Continuous Virtual Multi-Agent Systems, November 2000. (Software). `http://www.informatik.uni-mainz.de/~polani/XRaptor/XRaptor.html`, Nov. 2000

[6] Cliff, D., Harvey, I., and Husbands, P. Explorations in Evolutionary Robotics. *Adaptive Behavior*, 2(1):73–110, 1993.

[7] Colebourne, A. AC3D Modeller, 2000.
`http://www.comp.lancs.ac.uk/computing/users/andy/ac3d.html`, Sep. 2000

[8] Deneubourg, J. L. and Goss, S. Collective Patterns and Decision Making. *Ethology, Ecology and Evolution*, I, S. 295–311, 1989.

[9] Gat, E. On the Role of Simulation in the Study of Autonomous Mobile Robots. In *Proc. of AAAI 1995 Spring Symposium*, 1995.

[10] Hanks, S., Pollack, M. E., and Cohen, P. R. Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures. *AI Magazine*, 14(4):17–42, 1993.

[11] Heiligenberg, W. *Neural Nets in Electric Fish.* MIT Press, Cambridge MA, 1991.

[12] Klügl, F., Puppe, F., Raub, U., and Tautz, J. Simulating Multiple Emergent Phenomena – Exemplified in an Ant Colony. In *Proc. Artificial Life VI.* MIT Press, 1998.

[13] Langton, C., Burkhart, R., Lee, I., Daniels, M., and Lancaster, A. The Swarm Simulation System. `http://www.santafe.edu/projects/swarm/`, April 1998.

[14] Achim Liese, Daniel Polani, and Thomas Uthmann. On the development of spectral properties of visual agent receptors through evolution. In *Proc. GECCO 2000*, S. 857–864. Morgan Kaufmann.

[15] Lund, H. H., Hallam, J., and Lee, W.-P. Evolving Robot Morphology. In *Proc. IEEE 4th Int. Conf. on Evolutionary Computation*, NJ, 1997. IEEE Pr.

[16] Mark, A., Polani, D., and Uthmann, T. A Framework for Sensor Evolution in a Population of Braitenberg Vehicle-like Agents. In *Proc. Artificial Life VI*, S. 428–432. MIT Press, 1998.

[17] Matsubara, H., Noda, I., and Hiroaki, K. Learning of Cooperative Actions in Multi-Agent Systems: a case study of pass in Soccer. In *AAAI-96 Spring Symposium on Adaptation, Coevolution and Learning in Multi-Agent Systems*, S. 63–67, 1996.

[18] Menczer, F. and Belew, R. K. Evolving Sensors in Environments of Controlled Complexity. In *Proc. Artificial Life IV*, S. 210–221. MIT Press, 1994.

[19] Mössinger, P., Polani, D., Spalt, R., and Uthmann, T. A virtual testbed for analysis and design of sensorimotoric aspects of agent control. *Simulation Practice and Theory*, 5(7-8):671–687, 1997.

[20] Polani, D. and Uthmann, T. Survival Strategies for Ant-Like Agents in a Competitive Environment. In *Proc. 3rd German Workshop on Artificial Life*, S. 185–196, 1998.

[21] Ray, T. S. An approach to the synthesis of life. In *Proc. Artificial Life II*, S. 371–408, 1991. Addison-Wesley.

[22] Ray, T. S. A network-wide biodiversity reserve for digital organisms. In *Imagina 96 Proc.*, S. 15–26, Bry-sur-Marne, France, 1996. Institut National De L'audiovisuel.

[23] Sainitzer, R. and Buchegger, H. LODESTAR — Level of Detail Generator Release 1.0, 1997. `http://www.cg.tuwien.ac.at/research/vr/lodestar/`, Sep. 2000

[24] Sims, K. Evolving 3D Morphology and Behavior by Competition. In *Proc. Artificial Life IV*, MIT Press, 1994.

[25] Wieczorek, D., Lukowsky, M., and Brückner, S. Agent Development and Control System. In *Proc. Workshop DIMAS '95 (Decentralized Intelligent And Multi-Agent Systems), Krakow, Poland*, 1995.