XRaptor

Konzepte der Präsentation für Multiagentenszenarien in einer virtuellen kontinuierlichen Welt

Günter Bruns, Daniel Polani und Thomas Uthmann
Johannes Gutenberg-Universität Mainz, Institut für Informatik
Staudingerweg 9, D-55099 Mainz
{polani,uthmann}@informatik.uni-mainz.de

1 Einführung

Virtuelle Umgebungen haben in der letzten Zeit rapide an Bedeutung gewonnen. Ihre Anwendungen erstrecken sich über viele Bereiche, in denen sie etwa dazu beitragen, die Mensch-Maschine-Kommunikation zu verbessern, indem die imaginativen und Gestalt-aufnehmenden Fähigkeiten der Menschen dazu genutzt werden, Datenrepräsentation im Vergleich zu rein numerisch-symbolischen Kanälen zu verbessern. Eine dazu gewissermaßen duale Anwendung virtueller Umgebungen ist es, eine wohldefinierte Umwelt kontrollierter Komplexität zur Verfügung zu stellen, in der die Fähigkeit künstlicher Agenten zur sensormotorischen Verarbeitung, zur Adaption, zur Kommunikation und Koordination mit anderen Agenten entwickelt und evaluiert werden kann.

Eine virtuelle Welt zur Untersuchung von KI-Ansätzen ist immer dann von Nutzen, wenn standardisierte Testsuiten fester Struktur (wie sie zum Beispiel zur Leistungsbewertung von Neuronalen Netzen häufig verwendet werden) nicht ausreichen, wenn eine hardwaremäßige Realisierung problematisch ist (z.B. bei Untersuchungen zu Evolution von Morphologie (Sims 1994), Sensoren (Mark et al. 1998) oder funktionalen Phänotypen (Ray 1996; Adami 1998)) oder wenn sie zu aufwendig ist (etwa, weil das kollektive Verhalten sehr vieler einzelner Agenten untersucht werden soll).

In diesem Bereich gibt es unterschiedliche Ansätze verschiedener Spezialisierungsebenen. Zum einen gibt es spezialisierte Werkzeuge zur Modellierung spezifischer Aspekte der Evolution und von Szenarien zu künstlichem Leben (z.B. Tierra (Ray 1991, 1996), oder auch als Fortentwicklung davon das System Avida (Adami und Brown 1994; Adami 1997)). Auf der anderen Seite findet man Werkzeuge für die generelle Entwicklung von agentenorientierten Simulationsumgebungen (etwa CBB (Wieczorek et al. 1995; Wieczorek, D. and Lukowsky, M. and Brückner, S. 1995) oder Swarm (Langton et al. 1998)).

Mit dem in diesem Papier beschriebenen System XRaptor wird ein Bereich zwischen hochspezifischen und gänzlich allgemein gehaltenen Simulationsumgebungen eingenommen. XRaptor dient zur Simulation virtueller Agentenszenarien im Kontinuum des $\mathbb{R}^2$ oder $\mathbb{R}^3$. Durch die Möglichkeit der Modellierung eines virtuellen 3D-Simulationsszenarios und die durch die objektorientierte Architektur unterstützte Erweiterbarkeit ist das Konzept von XRaptor deutlich allgemeiner als etwa das von Tierra oder Avida. Auf der anderen Seite kann durch die höhere Spezialisierung im Vergleich etwa zu dem völlig generalistischen Ansatz von Swarm die Simulation effizienter und die Präsentation effektiver gestaltet werden.

Eine weitere Eigenschaft unterscheidet XRaptor von den anderen genannten Simulationsumgebungen. XRaptor unterstützt explizit die Implementation von Agententeams, die auch von unterschiedlichen Entwicklergruppen implementiert werden und in einem sogenannten *Turniermodus* – in einer gemeinsamen Welt – gegeneinander antreten können (Mössinger et al. 1995). Der Soccer Server im Rahmen des RoboCup-Projektes erlaubt eine ähnliche Teamorientierung (Noda 1995; Matsubara et al. 1996), wobei er allerdings im Gegensatz zu XRaptor auf die Simulation von Roboterfußball eingeschränkt ist. XRaptor eignet sich hingegen für die Simulation eines breiten Spektrums unterschiedlicher Szenarien.

Im Papier sollen einige generelle Eigenschaften von XRaptor vorgestellt werden, insbesondere einige der für die Präsentation wichtigen Mechanismen.

2 Generelles

XRaptor ist in C++ geschrieben und auf UNIX-Plattformen mit dem X-Window-System und Motif 1.2 lauffähig (Bruns et al. 1999). Die Simulationswelt von XRaptor ist das 2- oder 3-dimensionale reelle Kontinuum. XRaptor simuliert Agenten mit einer Sensor-Aktor-Architektur, in die ein Steuerungsmodul eingefügt werden kann, das die eingehenden Sensorinformationen auswertet (ggf. unter Berücksichtigung eines internen Weltmodelles) und daraus eine geeignete Aktivierung der Aktoren veranlaßt.

Eine wesentliche Designentscheidung bei XRaptor bestand in der Implementation einer expliziten Zugangskontrolle zu den Weltdaten, auf die sowohl Agenten wie Agentenentwickler zugreifen können und der Option, diesen Zugriff in unterschiedlichem Maße zuzulassen. Die Motivation für dieses Konzept war die Verwendung von XRaptor als Werkzeug für Lehrzwecke. Außerdem sollte XRaptor als Testumgebung fungieren, mit der die Wirksamkeit unterschiedlicher Agentensteuerungskonzepte erprobt werden sollte. Die Steuerung kann hierbei frei implementiert werden, sei es z.B. in Form wissensbasierter Ansätze, Neuronaler Netze oder auf andere Weise (Uthmann und Polani 1996).

Bei der Entwicklung von XRaptor wurde besonders auf eine hohe Flexibilität geachtet. Diese wird insbesondere deutlich durch die drei verschiedenen konzeptionellen Anwendungsebenen, in der Anwender das System nutzen können:

- als *Forscher*, der die Simulations„physik“ verändert, um Eigenschaften des simulierten Agentensystems zu studieren;
- als *Teamentwickler*, der ein Agententeam bei *vorgegebener* Physik, jedoch mit komplettem Weltwissen, entwickelt, das dann schließlich in dieser Simulationswelt gegen andere Agententeams antreten kann;
- als *Turnierteilnehmer*, dessen Agenten nur über eine sensoriell eingeschränkte Sicht auf die Welt verfügen.

3 Die Präsentationsebene

Die ursprünglich implementierte Präsentationsebene von XRaptor (Mössinger et al. 1997) ist komplett neu entwickelt worden und diese neue Ebene wird in diesem Papier erstmalig vorgestellt. Das Konzept wurde wesentlich verbessert und modernisiert, ohne das ursprüngliche „look-

and-feel“ zu verändern. Da die Kürze des zur Verfügung stehenden Raumes keine ausführliche Beschreibung der Architektur erlaubt, sollen hier nur die wichtigsten Aspekte dargestellt werden.

Die zentralen Anforderungen an die Neuimplementation der Präsentationsebene waren:

1. Eine weitgehende Trennung von Oberfläche und Problembereich (dem Simulator selbst). Eine Erweiterung um weitere zu simulierende Objekttypen oder eine Modifikation ihrer graphischen Darstellung soll möglichst isoliert vom restlichen Code der Oberfläche und ohne dessen genaues Verständnis durch den Entwickler des Simulators möglich sein;

2. eine einfache Erweiterung um andere Darstellungen und Sichtweisen der simulierten Welt;

3. eine simultane Darstellung der Simulation auf mehreren Bildschirmen zur Demonstration und Durchführung von Turnieren. Die Darstellungsformen in den verschiedenen *Display*-Fenstern sollen unabhängig voneinander manipuliert werden können;

4. die bisherige Darstellung durch Drahtmodelle soll durch 3-dimensionale Oberflächen ersetzt werden;

5. eine hinreichend effiziente Darstellung von Objekten, um den Hauptanteil der Rechenzeit der Simulation zur Verfügung stellen zu können.

Um die ersten drei Anforderungen zu erfüllen, wurde das Design der Präsentationsebene nach dem aus Smalltalk-Implementationen bekannten Model-View-Controller-Paradigma (Smalltalk User's Guide 1992) konzipiert. Simulierte Objekte und auch die ganze simulierte Welt gehören der Klasse *Model* an. Jeweils ein *Model* erhält durch mehrere unabhängige *Controller* Eingaben und wird durch mehrere unabhängige *Views* beobachtet. Ändert sich der Zustand eines Models, so benachrichtigt es alle bei ihm angemeldeten Views, damit diese ihre Darstellung aktualisieren können. Die zur Implementation von XRaptor verwendeten *Widgets* fassen Eingabe und Ausgabe in einem Objekt zusammen, daher modifiziert XRaptor MVC zu einem Model-View-Konzept, welches Controller und View in einer Klasse vereinigt (Collins 1995).

Diverse View-Controller-Unterklassen können verschiedene Aspekte des Modells auf unterschiedliche Art und Weise darstellen. So läßt sich wahlweise die Teamzugehörigkeit oder die Lebensenergie der Agenten, ein wichtiger Parameter in den Simulationen, über ihre Farbe visualisieren. Eine spezielle Klasse liefert eine Übersichtskarte der Welt, um eine bessere Orientierung bei der Wahl der Perspektive für die detailierte 3D-Darstellung in Zentralprojektion mittels einer anderen Klasse zu ermöglichen. Eine Texteingabeklasse erlaubt dem Benutzer, Anweisungen an die Welt und ihre Agenten einzugeben, während eine Textausgabeklasse genaue Messungen von Zustandsparametern der Agenten ermöglicht. Eine vierte Klasse steuert schließlich die Simulation der virtuellen Welt. Objekte dieser View-Controller-Klassen können in beliebiger Vielfalt und Anzahl ganz zwanglos koexistieren und bilden so die Bausteine zu einer Erweiterung oder Anpassung der Präsentationsebene an besondere Problemstellungen.

Dies Konzept schafft auch einen Rahmen für die simultane Betrachtung auf mehreren Bildschirmen. Jedes View-Controller-Objekt öffnet dazu sein eigenes X-Window-Display. Obwohl das X-Window-System die gleichzeitige Verwendung mehrerer *Displays* erlaubt, nutzen dies bisher nur wenige Anwendungen. Auch gibt X11 dazu keine explizite Unterstützung und einige Besonderheiten (Jones 1992) müssen beachtet werden; manche Toolkit-Implementationen ignorieren die Möglichkeit, mehrere unterschiedliche Displays zu öffnen und führen so zu Inkonsistenzen und Fehlern. Die Darstellung auf verschiedenen Bildschirmen ist wiederum völlig voneinander unabhängig und kann so den Wünschen der einzelnen Benutzer angepaßt werden. Auch ist es bei

der Durchführung von Turnieren wünschenswert, daß nur der Turnierleiter an seinem Bildschirm die Simulation beeinflußen kann, während die rein beobachtenden Funktionen allen Benutzern offenstehen.

Die bisherige Darstellung der Agenten durch Drahtmodelle führte zu einer nur unzureichenden räumlichen Wahrnehmung der Benutzer. Durch die Verwendung von plastisch wirkenden beleuchteten Oberflächenmodellen soll dem abgeholfen werden. Änderungen des inneren Zustandes und Aktionen der Agenten (z.B. Fressen eines Beuteobjektes) lassen sich durch Gestaltsänderungen oder farblich visualisieren.

Zur Präsentation von 3D-Welten verwendet XRaptor das zur Standarddistribution des X-Window Systems gehörende PEX. Oberflächen werden aus einer Vielzahl von Polygonen (Facetten) zusammengesetzt und der Beleuchtung entsprechend schattiert gezeichnet. Eine der Anforderungen an XRaptor war ein vergleichsweise geringer Rechenaufwand der grafischen Darstellung relativ zur eigentlichen Simulation. Dies ist insbesondere bei der 3D-Darstellung wichtig. Da typischerweise mehrere hundert Objekte und Agenten in einem Simulationsexperiment vorkommen können, ist es wichtig, die Darstellung effizient zu gestalten, um auch bei nicht grafikbeschleunigten Systemen den Hauptanteil der Rechenleistung auf die Dynamik des Systems konzentrieren zu können. Eine vergleichbare Problematik tritt bei der Bearbeitung komplexer statischer Modelle durch CAD-Systeme (Silicon Graphics 1998) auf, dort soll die Interaktivität erhalten bleiben. Die Effizienz der Darstellung von 3D-Szenarien wird durch verschiedene Optimierungsmaßnahmen erreicht:

- Eine *Level-of-Detail*-Verwaltung stellt weiter entfernte Objekte mit geringerer Auflösung dar als nähere und vereinfacht dazu die Oberflächen durch Verwendung von weniger Facetten. Für das Design der 3D-Objekte können externe Standardwerkzeuge eingesetzt werden, die VRML-Repräsentationen erzeugen. Aus diesen Repräsentationen generieren *Level-of-Detail*-Generatoren wie etwa LODESTAR (Sainitzer und Buchegger 1999) VRML-Darstellungen mit abgestuften Auflösungen, unter denen XRaptor die zur jeweiligen Perspektive passende auswählt und zum Zeichnen verwendet (siehe z.B. Abb. 1 für eine solche Darstellung). Die Komplexität der darzustellenden Objekte unterliegt keinen a priori Beschränkungen.

- *View-Frustrum-Culling* übergibt PEX nur Objekte, die innerhalb des Gesichtsfeldes des 3D-Darstellungs-Fensters liegen.

- *Backface-Culling* zeichnet nur Facetten mit zum Betrachter hin orientierter Normale und vermeidet so bei geschlossenen Oberflächen die Belastung von PEX mit dem Zeichnen ohnehin verdeckter Flächen, das Einsparpotential beträgt typischerweise 50%.

Die Darstellung von komplexen 3D-Welten – im Überblick oder ausschnittsweise wie etwa in Abb. 1 – benötigt auf dem lokalen Display einer Sun Ultra 1 Workstation ohne Grafikbeschleunigung in der gegenwärtigen Implementation ca. 0,1–0,3 s, was im angestrebten Bereich liegt. Eine weitere Steigerung der Bildfrequenz auf über 20 Bilder/Sekunde durch die Anwendung von Hardware-Graphikbeschleunigern erscheint möglich.

Die *Remote*-Anzeige von Simulationen benötigt wegen der über das Netz zu transportierenden Daten u.U. mehr Zeit. Der genaue Wert hängt von dem zur Verfügung stehenden PEX-Protokoll ab. Da die PEX-Implementationen z.T. voneinander abweichen, unterstützt XRaptor verschiedene Formen des PEX-Protokolls, die unterschiedlich sparsam in der Verwendung der Netzbandbreite sind. Eine derartige *Remote*-Darstellung ist insbesondere bei einer Multi-Display-Anzeige

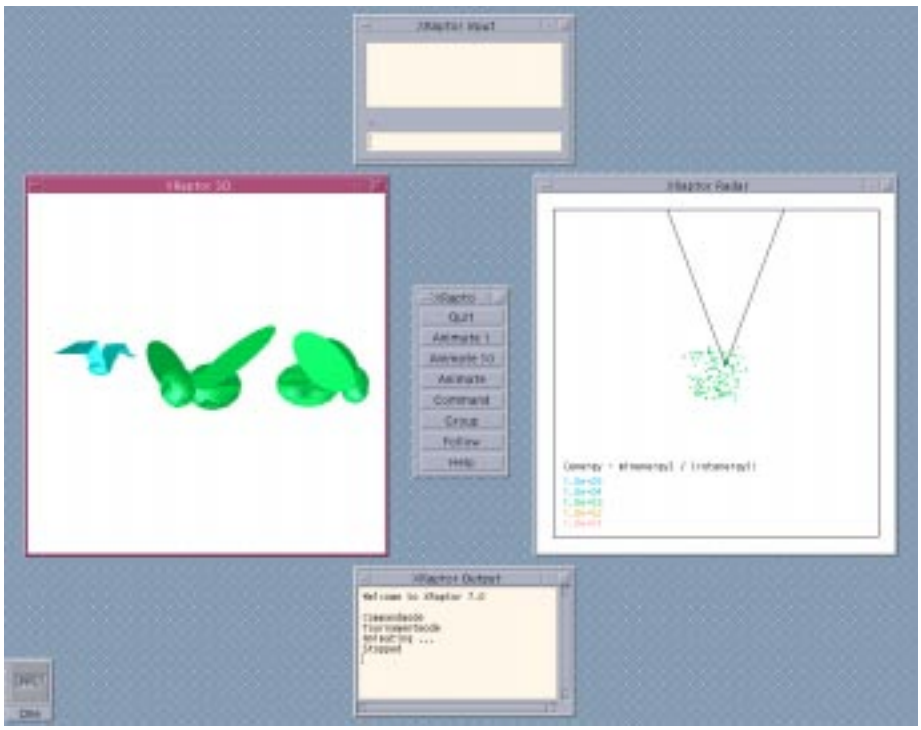

Abbildung 1: Blick auf die XRaptor-Oberfläche. Links das 3D-Sichtfenster in die Simulationswelt, rechts eine Übersichtsanzeige (von oben; Sichtkegel des 3D-Fensters ist explizit eingezeichnet), oben ein Kommandoeingabefenster, unten ein Ausgabefenster, in der Mitte das Steuerfeld. Vgl. auch (Mössinger et al. 1997). Das 3D-Fenster zeigt drei Agenten eines Räuber-Beute-Szenarios (Polani und Uthmann 1999).

(Anforderung 3) von Interesse. Hier wirkt sich das Client-Server-Konzept von PEX positiv auf die der Simulation zur Verfügung stehenden Rechenleistung aus – XRaptor muß nur die Vertices der Oberflächenfacetten zu anderen Workstations tranportieren, während der Aufwand für die Grafikberechnungen von diesen Fremdrechnern getragen wird.

Falls kein PEX zur Verfügung steht, schaltet XRaptor automatisch auf eine Drahtmodellrepräsentation der Objekte und Agenten um, mit der zumindest noch eine abstrahierte Darstellung möglich bleibt.

4 Zusammenfassung und Ausblick

In dem Papier ist das Simulationswerkzeug XRaptor für Multiagentenwelten und insbesondere seine Präsentationsschicht dargestellt worden. Der vorgestellte Präsentationsmechanismus erlaubt gleichzeitig eine flexible Erweiterung der in XRaptor repräsentierten Objekte wie eine vom eigentlichen Simulationssystem abgekoppelte Möglichkeit der unabhängigen Erstellung der Darstellungsobjekte. Multidisplayfähigkeit sowie eine effiziente und an die Erfordernisse der Simulation angepaßte Darstellung durch eine flexible *Level-of-Detail*-Organisation sind weitere Eigenschaften der Präsentationsebene.

PEX war als 3D-Darstellungssystem wegen seiner im X-Window-Standard festgelegten universellen Verfügbarkeit gewählt worden. Wegen der zunehmenden Verfügbarkeit von OpenGL und kompatiblen Produkten für viele Plattformen, insbesondere wegen ihrer Unterstützung von Grafikbeschleunigern, ist zusätzlich zu PEX auch die Implementation mittels OpenGL geplant. Das für die Struktur der Präsentationsebene gewählte Konzept bleibt hierbei erhalten, was eine Begrenzung des Umstellungsaufwandes erwarten läßt.

Literatur

Adami, C., (1997). Handbook and Publications about Avida. `http://www.krl.caltech.edu/avida/`.

Adami, C., (1998). *Introduction to Artificial Life*. Springer.

Adami, C., und Brown, C. T., (1994). Evolutionary Learning in the 2D Artificial Life System "Avida". In Brooks, R., und Maes, P., editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 377.

Bruns, G., Mössinger, P., Polani, D., Spalt, R., Uthmann, T., und Weber, S., (1999). XRaptor: A Simulation Environment for Continuous Virtual Multi-Agent Systems. `http://www.informatik.uni-mainz.de/ polani/XRaptor/XRaptor.html`, Juni 1999

Collins, D., (1995). *Designing Object-Oriented User Interfaces*. Benjamin Cummings.

Jones, O., (1992). Multi-user Application Software Using Xt. *The X Resource*, 3 (Summer 1992).

Langton, C., Burkhart, R., Lee, I., Daniels, M., und Lancaster, A., (1998). The Swarm Simulation System. `http://www.santafe.edu/projects/swarm/`.

Mark, A., Polani, D., und Uthmann, T., (1998). A Framework for Sensor Evolution in a Population of Braitenberg Vehicle-like Agents. In Adami, C., Belew, R., Kitano, H., und Taylor, C., editors, *Proc. of Artificial Life VI, Los Angeles, June 26-29*, 428–432. MIT Press.

Matsubara, H., Noda, I., und Hiraki, K., (1996). 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*, 63–67.

Mössinger, P., Polani, D., Spalt, R., und Uthmann, T., (1995). XRaptor – A synthetic Multi-Agent Environment for Evaluation of Adaptive Control Mechanisms. In Breitenecker, F., und Husinsky, I., editors, *Proc. EUROSIM '95*. Elsevier.

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

Noda, I., (1995). Soccer Server: a simulator for RoboCup. In *JSAI AI-Symposium 95*.

Polani, D., und Uthmann, T., (1999). XRaptor – Eine flexible Umgebung für die Simulation virtueller zeit- und raumkontinuierlicher 2D- und 3D-Agentenszenarien. In Szczerbicka, H., und Uthmann, T., editors, *Modellierung, Simulation und Künstliche Intelligenz. State of the Art*. SCS European Publishing House. (in preparation).

Ray, T. S., (1991). An approach to the synthesis of life. In Langton, C., Taylor, C., Farmer, J. D., und Rasmussen, S., editors, *Artificial Life II*, vol. XI of *Santa Fe Institute Studies in the Sciences of Complexity*, 371–408. Redwood City, CA: Addison-Wesley.

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

Sainitzer, R., und Buchegger, H., (1999). LODESTAR — Level of Detail Generator Release 1.0. `http://www.cg.tuwien.ac.at/research/vr/lodestar/`, Juni 1999

Silicon Graphics, Inc., (1998). OpenGL Optimizer White Paper. `http://www.sgi.com/software/optimizer/whitepaper.html`, Juli 1999

Sims, K., (1994). Evolving 3D Morphology and Behavior by Competition. In Brooks, R., und Maes, P., editors, *Proc. Artificial Life IV*.

Smalltalk User's Guide, (1992). *Objectworks/Smalltalk User's Guide*. ParcPlace Systems, Sunnyvale, CA.

Uthmann, T., und Polani, D., (1996). Verteilte Künstliche Intelligenz in einer virtuellen Welt mit selbstorganisierenden Karten und Genetischen Algorithmen. In Haugeneder, H., Kraetzschmar, G., Müller, J., Weiß, G., und Wrobel, S., editors, *Proc. des Workshops "Lernen, Adaption und Selbstorganisation in verteilten intelligenten Systemen" der KI-96*, Nr. FKI-217-96 der Forschungsberichte Künstliche Intelligenz, 52–67. TU München.

Wieczorek, D., Lukowsky, M., und Brückner, S., (1995). Agent Development and Control System. `http://www.informatik.hu-berlin.de/ lukowsky/cbb.html`.

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