PartSim - Ein System zur Unterstützung interaktiver Simulationen in der Lehre

Giovanni Falcone, Nicolai Scheele, Wolfgang Effelsberg, Colin Atkinson
University of Mannheim, Germany
{*falcone, scheele, effelsberg, atkinson*}@informatik.uni-mannheim.de

Abstract: Im Rahmen des PartSim-Projektes wurde ein Java-basiertes Softwaresystem entwickelt, das die Erstellung interaktiver Simulationen für die Lehre erleichtert. Diese Arbeit stellt zunächst die Anwendung solcher partizipatorischer Simulationen im Ausbildungssektor vor und geht auf bestehende Systeme ein. Im zweiten Teil wird die PartSim-Software besprochen, sowie eine beispielhafte Anwendung mit den ersten evaluatorischen Ergebnissen.

1 Einleitung

Die rasante Entwicklung neuer Technologien in den letzten Jahren hat dazu geführt, dass auch die Lehre mehr und mehr mit multimedialen Elementen angereichert wird. Neben der heute in vielen Fachbereichen üblichen Verwendung elektronisch unterstützter Präsentationsformen ist es sowohl in den Vorlesungen an Universitäten als auch im Schulunterricht schon üblich, zu bestimmten Themen Bilder, Videos oder Animationen zu zeigen.

Simulationen sind ein weiteres multimediales Element, das zunehmend Beachtung gewinnt. Für Simulationen wird ein streng definierter Bereich der Realität so reproduziert, dass das entstehende Modell innerhalb seiner Grenzen möglichst wirklichkeitsgetreu reagiert. Entsprechend können diese insbesondere dann sinnvoll eingesetzt werden, wenn die Lehre gerne mit Experimenten fundiert wird, entweder als Unterstützung oder gegebenenfalls sogar als Ersatz für das eigentliche Experiment. Doch auch in anderen Bereichen, in denen Experimente bisher nicht oder nur sehr schwer durchführbar waren, können Simulationen für ein besseres Verständnis bei Studenten bzw. Schülern sorgen.

So müssen sich die Parameter einer Simulation nicht zwangsläufig an realistischen Richtlinien orientieren. Zusätzlich zu einem realen Experiment können die Ergebnisse per Simulation auf andere Umgebungen oder Einflüsse reflektiert werden, die nicht ohne weiteres dargestellt werden können. Beispielsweise können Messungen eines Ballwurfes mit simulierten Ergebnissen desselben auf anderen Planeten verglichen werden.

Weiterhin haben Simulationen keinen Einfluss auf die Realität, so dass die Möglichkeit gegeben, wird gefährliche Experimente zu substituieren. Ebenso ist es möglich, fehlgeschlagene Simulationen ohne zusätzliche Kosten beliebig häufig zu wiederholen, wodurch eine experimentellere Herangehensweise an bestimmte Themen gefördert wird. Darüber hin-

aus sind die Ergebnisse von Simulationen nicht auf die Darstellung normal wahrnehmbarer Phänomene (Töne, sichtbares Licht) beschränkt, sondern können durch andere, messbare Werte in beliebig feiner Auflösung ergänzt werden, wie beispielsweise Temperatur, Verteilung mikroskopisch kleiner Partikel oder nicht ohne Hilfsmittel wahrnehmbare Frequenzbereiche.

Ein insbesondere für die Lehre interessanter Aspekt von Simulationen ist, dass die Komplexität des Modells beinahe beliebig angepasst werden kann. Dadurch ist es möglich, zunächst nur mit dem nötigsten Regelsatz zu beginnen und diesen dann Schritt für Schritt zu ergänzen. Bei dem Ballbeispiel, könnte zunächst nur mit dem Wurfwinkel, der Wurfkraft und der Gravitationskonstante begonnen werden, um in späteren Durchläufen dann Luftwiderstand in Abhängigkeit von Ballgröße, -form und -gewicht, Windeinflüsse und Auswirkungen der Balloberfläche hinzuzufügen.

Simulationen können statisch oder dynamisch sein. Während statische Simulationen anhand eines vorher festgelegten Parametersatzes ein beliebig komplexes Szenario durchrechnen und die Ergebnisse präsentieren, erlaubt die dynamische Variante bei begrenzt komplexen Modellen die Veränderung der Parameter zu jeder Zeit.

Interaktive - oder partizipatorische - Simulationen erweitern die dynamischen Modelle um eine Gruppenkomponente mit rollenspielähnlichem Charakter. Den beteiligten Personen wird eine Facette des Modells zugewiesen, die sie selbstverantwortlich regeln können. Sie haben jedoch keinen Zugriff auf die Parameter der anderen Facetten, mit denen ihr Bereich oft sehr starke Abhängigkeiten aufweist. Der aktuelle Zustand des gesamten Modells kann je nach Bedarf für alle Teilnehmer sichtbar gemacht werden.

Wir sehen partizipatorische Simulationen als ein neues interessantes Werkzeug an, bei Schülern bzw. Studenten ein sehr tiefes Verständnis für bestimmte, simulierbare Sachverhalte zu erzeugen. Mit den Simulationen verbundene Zielsetzungen lassen sich nur in der Gruppe lösen, wobei alle Beteiligten gleichermaßen gefordert sind. Ausgiebige Diskussionen über Beobachtungen und daraus abzuleitende Strategien sind dazu notwendig. Die im Modell gewonnenen Erfahrungen lassen sich anschließend leicht auf die Realität übertragen [CHB01].

2 Vorhandene Ansätze

In den vergangenen Jahren wurden einige Ansätze veröffentlicht, die eine gemeinsame Nutzung einer simulierten Umwelt zum Ziel hatten. Als eines der ersten Projekte in diesem Bereich sind die „Thinking Tags“ des MIT zu nennen. „Thinking Tags“ waren kleine, elektronische Geräte mit einem Infrarotsender und -empfänger, zwei Tasten und fünf LEDs.

Diese Geräte wurden von den Teilnehmern an einem Band um den Hals getragen; trafen sich zwei „Tag Bearer“, so wurden zwischen den Geräten automatisch bestimmte Daten übertragen. Mit Hilfe der Tasten und LEDs konnten die gespeicherten Informationen abgerufen und einprogrammierte Routinen gestartet werden.

Die „Tags“ wurden für verschiedene Anwendungen getestet; eine der bekanntesten war eine Simulation, mit der die Ausbreitung einer Seuche verdeutlicht werden sollte. Den Teilnehmern wurden zufällig die Zustände „normal“, „immun“ oder „infiziert“ zugewiesen. Daraufhin sollten sie sich mit den angelegten Tags ganz normal für eine bestimmte Zeit untereinander unterhalten. Immer, wenn ein „normaler“ Teilnehmer auf einen „infizierten“ traf, wurde zufällig bestimmt, ob es zu einer Infizierung gekommen ist; je länger der Kontakt bestand, desto höher war die Wahrscheinichkeit einer Infizierung. Am Ende der Simulation wurden die Daten ausgewertet und diskutiert.

Vom MIT wurden später noch weitere partizipatorische Simulationen veröffentlicht, die meistens auf *PDAs*, die den Teilnehmern ausgehändigt wurden, liefen [Klo05]. Dabei wurden auch weitere sensorische Daten, wie beispielsweise GPS verwendet: In einer Beispielsimulation „Environmental Detectives“ sollten die Studenten auf dem Campus die Quelle einer Wasserverunreinigung herstellen. Mit den *PDAs* konnten sie virtuelle Messungen vornehmen; die Daten erhielten die Geräte von einem zentralen Server, der diese anhand der Geokoordinaten berechnete.

All diese Simulationen jedoch waren einzelne Anwendungen ohne einheitliches Rahmenwerk oder andere Unterstützung für interessierte Programmierer. Ein solches wurde einige Zeit später von der Northwestern University Evanston in Illinois geliefert: *NetLogo* bot eine einheitliche, einfach zu programmierende Oberfläche für die Gestaltung einfacher, dynamischer Simulationen [WHF99], [TW04]. Da *NetLogo* komplett in *Java* implementiert wurde, war es möglich, die Simulationen als Applet auf Webseiten zu veröffentlichen.

Allerdings untersützte *NetLogo* alleine noch keine Mehrbenutzerszenarien. Diese wurden erst durch die Erweiterung *HubNet* möglich [WS99]. *HubNet* erlaubte zwei verschiedene Modi. Der erste sah eine gemeinsame Darstellung der modellierten Welt vor und gestattete den Teilnehmern die Beeinflussung ihres Bereiches mit Hilfe von infrarotfähigen Taschenrechnern (wie bei einer Fernbedienung). Der zweite Modus stellte für jeden Teilnehmer individuell auf einem eigenen Notebook die jeweilige Sicht auf das Modell dar.

Eine sehr bekannte, beim *HubNet*/*NetLogo*-Paket beiliegende Simulation ist *GridLock*. Die Teilnehmer sind hier aufgefordert, die Ampelschaltungen in einem Verkehrsknoten so vor zu nehmen und zu synchronisieren, dass sich ein stabiler, staufreier Zustand einpegelt.

Bedauerlicherweise werden weder *NetLogo* noch *HubNet* seit einigen Jahren weiterentwickelt. Entsprechend basieren beide Systeme auf einer veralteten Technologie. In zwei Projekten mit Studenten konnten wir zwar feststellen, dass sich mit NetLogo sehr leicht und intuitiv einfache Simulationen zusammenstellen lassen, ohne dass dafür fortgeschrittene Programmierkenntnisse notwendig wären, für komplexere Simulationen jedoch sind weder die verwendete Skriptsprache, die auf *LOGO* basiert, noch die zur Verfügung stehenden Benutzeroberflächenkomponenten sonderlich geeignet [WK04].

3 Technische Vorraussetzungen

Resultierend aus diesen Untersuchungen haben wir beschlossen, mit *PartSim* ein neues Rahmenwerk zur Programmierung partizipatorischer Simulationen zu schaffen, dessen

Realisierung wir in den nachfolgenden Abschnitten beschreiben wollen. Zunächst sollen hierzu die technischen und softwaretechnischen Vorraussetzungen und die damit verbundenen Einschränkungen erörtert werden.

3.1 Hardware

Um bereits im Vorfeld eine große Akzeptanz des Systems zu erzielen, entschieden wir uns, basierend auf den Erfahrungen des *WILD (Wireles Interactive Learning Device)*-Projekts [SWE04], als mobile Endgeräte der Lernenden *PocketPCs* bzw. *PDAs* einzusetzen. Dieser Entscheidung lag die Erwartung zu Grunde, dass sich in den kommenden Jahren *PDAs* bzw. *PocketPCs* und mobile Telefone sowohl im Aussehen als auch in den Fähigkeiten annäheren und in einem Gerät münden werden. Ansätze für diese Annahme sind bereits heute bei *Smartphones* erkennbar.

Auf der Seite der Serveranwendung kann ein handelsüblicher PC aktueller Generation zum Einsatz kommen, an den keine weiteren Vorraussetzungen geknüpft sind, außer dass er, ebenso wie die mobilen Endgeräte, über eine Wireless LAN (*IEEE 802.11*) Anbindung zur Kommunikation untereinander verfügen muss. Bei mobilen Endgeräten ist für die Beispielsimulationen der Routing-Protokolle, die nachfolgend beschrieben werden, zudem noch eine Positionsbestimmung mittels Global Positioning System notwendig.

3.2 Software

Die Erfahrungen aus dem *WILD* Projekt, bei dem als Programmiersprache *Java* auf der Serverseite und *PersonalJava*, basierend auf dem *JDK1.1* (*Java Development Kit 1.1* [SUN05]), auf der Clientseite zum Einsatz kam, waren durchaus positiv, so dass diese auch in diesem Projekt zum Einsatz kommen sollten. Lediglich die „End of Lifetime“-Lizenz von *PersonalJava* zwang zu einer Evaluation vergleichbarer Umgebungen.

Die Entscheidung fiel hierbei auf die *EWE* Runtime Umgebung [EWE05], da diese ebenfalls auf *PersonalJava* basiert, jedoch zusätzlich noch Möglichkeiten zur Verfügung stellt, mit einfachen Mitteln ansprechende graphische Benutzeroberflächen zu erzeugen. Auf der Seite des Servers hingegen kommt *J2SDK* (*Java Standard Development Kit*) in der Version 1.4 oder höher zum Einsatz.

Im nachfolgenden Abschnitt soll nun, basierend auf den technischen und theoretischen Voraussetzungen, das Design des entstandenen Frameworks erläutert werden.

4 Design

Die beiden wichtigsten Ziele, die als Maßgabe für das Framework dienen sollten, sind

1. dem Anwender eine standardisierte Oberfläche zur Verfügung zu stellen, um damit seine Konzentration auf den zu erlernenden Sachverhalt zu lenken und nicht auf ständig wechselnde oder zu komplexe GUIs und
2. dem Anwendungsentwickler eine Umgebung zur Verfügung zu stellen, bei der er sich nicht um Basisfunktionalitäten des Systems, wie beispielsweise die Kommunikation zwischen den Clients und dem Server, kümmern muss.

Eine Trennung in einen als Kern bezeichneten Teil und einen modul-, simulationsspezifischen Teil schien daher sinnvoll. Der Kern stellt dem System hierzu Basisfunktionalitäten, wie einem Authentifizierungsmechanismus zur Anmeldung am System, zur Verfügung. Des weiteren werden Logging Mechanismen bereitgestellt, die auf verschiedenen Ebenen zum Einsatz kommen können. Zum einen handelt es sich hierbei um ein Logging auf Systemebene, zum anderen um ein Logging auf Modulebene, bei der modulspezifische Daten zur späteren Analyse gespeichert werden können. Beispiele dafür wären Antworten der Anwender auf bestimmte Ereignisse. Die Resultate eines Loggings auf Modulebene können auch dazu genutzt werden, einen Überblick über den Leistungsstand der einzelnen Anwender zu bekommen. Diese Ergebnisse sind dabei weit detaillierter und spiegeln den Wissensstand der einzelnen Teilnehmer deutlich besser wider, als es in einer herkömmlichen Vorlesung oder Unterrichtseinheit an Schulen möglich wäre.

Im modulspezifischen Teil werden hingegen Basiskomponenten zur Verfügung gestellt, die eine einfache Entwicklung neuer Module ermöglichen sollen:

Server: Auf der Seite des Servers sind dies die beiden Komponenten `AbstractModule` und `GUIModule`. `AbstractModule` stellt hierbei die Schnittstellen zur Kommunikation zu den einzelnen Clients oder einen ungerichteten Versand zu allen Teilnehmern zur Verfügung. Auch die Basisfunktionalitäten wie der Start, das Stoppen, oder Pausieren eines Moduls werden von dieser Komponente angeboten. `GUIModule` stellt einen einheitlichen Aufbau der Oberfläche auf Serverseite zur Verfügung, um dem Dozenten das Zurechtfinden innerhalb eines neuen Moduls zu erleichtern.

Client: Auf der Clientseite entschieden wir uns dafür, die beiden Teile `AbstractModule` und `GUIModule` in einer einzelnen Komponente zur Verfügung zu stellen (`AbstractModule`), da die Vielfältigkeit der möglichen Simulationen ein einheitliches Oberflächenaussehen kaum zulässt. Auch die Kommunikation lässt sich auf die beiden Basisfälle der Kommunikationen mit dem systemspezifischen Teil auf der einen und dem Server auf der anderen Seite reduzieren.

Netzwerkkommunikation: Die Kommunikation zwischen den einzelnen Parteien erfolgt hierbei mittels eines textbasierten Protokolls, wobei der Austausch von Nachrichten im systemspezifischen Teil zu finden ist und auf Byte-Ströme als Kommunikationsbasis setzt. Zur Reduzierung der Nachrichtengröße entschieden wir uns, eindeutige Kennzahlen für die einzelnen Zielgruppen (System, ModulID,...) einzuführen, die auf *short*-Werten basieren. Die Nachrichtentexte selbst werden als Klartext im Message Body übertragen.

Sicherheit: Da eine Übertragung über eine unsichere Netzwerkleitung stattfinden sollte, entschieden wir uns zudem, noch eine (optionale) Verschlüsselung der Nachrichten durchzuführen. Hierzu diente als Ausgangspunkt ein Schlüsselaustausch zwischen einem Client und dem Server mittels des *Diffie-Hellman* Protokolls. Es werden zwei verschiedene Schlüssel erzeugt. Zum einen ein modul- oder simulationsspezifischer-Schlüssel, der als Grundlage zur sicheren Kommunikation von Moduldaten dient, zum anderen einen anwenderspezifischen Schlüssel, mit dem die Daten eines einzelnen Anwenders und dem System ausgetauscht werden können. Die Schlüssel selbst basieren hierbei auf dem *Data Encryption Standard (DES)* [Sch02]. Die Möglichkeit der Verschlüsselung wird ab der nächsten Version des Systems zur Verfügung gestellt werden.

5 Beispielsimulation - Routing Protokolle

Der Einsatz des Systems wurde anhand einer Beispielsimulation getestet und anschließend in einem Informatikkurs der Jahrgangsstufe 12 evaluiert. Nachfolgend soll zunächst die Beispielanwendung selbst und deren Aufbau erläutert und anschließend die Ergebnisse der Evaluation dargestellt werden.

Bei einer Simulation von Routing Protokollen kann auf natürliche Weise die Position der einzelnen Anwender in die Simulation selbst integriert werden und damit ein weiterer Sinn des Anwenders angesprochen werden, wie dies bereits in der Einleitung erläutert wurde. Diese Integration erfolgt dabei dadurch, dass man die Distanz zwischen zwei Teilnehmern nicht mit einem festen Wert annimmt, sondern die reale Distanz und damit verbunden die Übertragungszeit einer Nachricht in einem Netzwerk misst.

Auch die Reihenfolge und die Auswahl der zu erlernenden Protokolle spielt bei dem zu erwartenden Erfolg einer Simulation eine große Rolle. Hierbei entschieden wir uns, mit dem einfachsten Routing Protokoll, dem *Flooding Protokoll* die Simulation zu beginnen. Die Teilnehmer sollten zunächst die Nachteile des Protokolls in drei Phasen erkennen. Darüber hinaus sollten sie erkennen, wie diese Nachteile das *Routing Information Protocol (RIP)* [Mal94] beeinflussen. Dies sollte im nächsten Schritt erarbeitet werden. Hierbei erfolgt eine Unterteilung in drei einzelne Phasen, in denen die Teilnehmer die Problematik des Protokolls verstehen sollen. Am Ende sollten sie ein Gefühl für die Problematik bekommen, welches sie in der aus zwei Phasen bestehenden *Open Shortest Path First (OSPF) Protokoll* [Moy98] weiter vertiefen können. Nachfolgend sollen die einzelnen Stufen der Erarbeitung geschildert werden.

Die Struktur des Netzwerks und damit, mit wem ein Teilnehmer virtuell verbunden ist, wird durch das System beim Start festgelegt. Man ist jedoch immer mit dem am nächsten gelegenen Nachbarn verbunden, und mit den weiteren zu einer bestimmten Wahrscheinlichkeit, die abhängig von der Entfernung ist.

5.1 Flooding Protokoll

1. Phase: In dieser Einstiegsphase sollen dem Anwender zum einen die graphische Oberfläche näher gebracht werden, die sich auch in den folgenden Phasen zu den verschiedenen Protokollen nur sehr wenig in Ihrer Struktur und Handhabung ändert. Hierzu wurde das Protokoll soweit vereinfacht, dass die Konzentration zunächst auf die Gewöhnung an das System ausgerichtet wurde. Die Teilnehmer schicken hierzu eine initiale Nachricht an einen der Nachbarn. Im weiteren Verlauf werden bei Ihnen zur Weiterleitung bestimmte Nachrichten eintreffen, die sie wiederum an alle Nachbarn, mit denen sie virtuell verbunden sind, schicken müssen (Abbildung 1).

Abbildung 1: Screenshots bei der Erarbeitung des *Flooding Protokolls*

2. Phase: In dieser mittleren Phase sollen nun erste Restriktionen beim Versenden von Nachrichten die eintreffende Flut von Nachrichten bei den einzelnen Teilnehmern verringern, in dem eine Nachricht nur an die Nachbarn verschickt wird von der die Nachricht nicht erhalten wurde. Die sich dadurch einstellende Reduktion der Nachrichten ist aber verschwindend gering und auch die Masse an Nachrichten, die sich durch das „Fluten“ im Netzwerk befinden, ist dadurch nicht reduzierbar gewesen. Diese Erkenntnis sollte die Teilnehmer zur dritten Phase führen, in der das komplette Protokoll simuliert werden soll.

3. Phase: In dieser letzten Phase der Simulation des *Flooding Protokolls*, soll nun das eigentliche Protokoll simuliert werden, bei der eine Nachricht einen Hop-Counter enthält, der die Anzahl an Stationen zählt und nur eine gewisse Anzahl an Hops zulässt. Da bei der Simulation Positionsdaten und damit auch die direkten Distanzen zwischen den Teilnehmern zur Verfügung stehen, wird hier die mittlere Länge des Netzwerks als Schwellwert für den Weiterversand einer Nachricht verwendet.

Die Nachteile des *Flooding Protokolls* sollten nach Ablauf dieser Phase verstanden worden sein und die Teilnehmer sollten festgestellt haben, dass der Mehraufwand in einem Protokoll, welches einen direkten Versand einer Nachricht über einen bestimmten Pfad zum Empfänger zulässt, durchaus gerechtfertigt ist. Ein solches Protokoll stellt das *RIP*

zur Verfügung, bei der ein Knoten darüber informiert ist, über welchen Nachbarn eine Nachricht zu einem bestimmten Empfänger zu verschicken ist. Dies wird anhand von Routingtabellen auf den einzelnen Knoten festgehalten, die sich dynamisch an die Umgebung des Netzwerks anpassen. In den folgenden 3 Phasen sollen die Teilnehmer zunächst die Vorteile gegenüber dem *Flooding Protokoll* verstehen und die Auswirkungen direkt durch die reduzierte Datenflut spüren. Ein Beispiel der Benutzeroberfläche während einer *RIP* Simulation ist Abbildung 2 zu entnehmen.

5.2 Routing Information Protokoll

1. Phase: Diese erste Phase soll wieder zum Eingewöhnen an die geänderte Benutzeroberfläche dienen, wobei die Teilnehmer eine Routingtabelle zu den einzelnen Empfängern und den Weg, über welchen direkten Nachbarn eine Nachricht versandt werden soll, erstellen müssen.

2. Phase: An dieser Stelle sollen sie nun merklich die Verbesserung durch das eingesetzte Protokoll spüren, indem sie, ähnlich wie bei der Simulation des *Flooding Protokolls*, eine Nachricht an einen Empfänger schicken und im weiteren Verlauf als Vermittler für eingehende Nachrichten dienen sollen. Dabei werden diese, basierend auf der erstellten Routingtabelle, an ihren Bestimmungsort weitergeleitet.

3. Phase: Diese letzte Phase soll nun den Nachteil durch das Nutzen von Tabellen erläutern, der darin besteht, dass es nach Ausfall eines Knotens im Netzwerk eine gewisse Zeit dauert bis die Tabellen korrigiert worden sind. Dabei kann es dazu kommen, dass eine Nachricht einen sehr langen Weg zu Ihrem Bestimmungsort zurücklegen muss. Die dadurch entstehende Verzögerung beim Ausliefern einer Nachricht kann diese für einen Empfänger allerdings unbrauchbar machen, wie dies beispielsweise bei dem Versand eines Videostroms der Fall ist. Zu diesem Zweck werden zufällig einige virtuelle Verbindungen gelöscht und nur den direkt beteiligten Knoten eine Nachricht zugeschickt, die sie über diesen Verlust informiert.

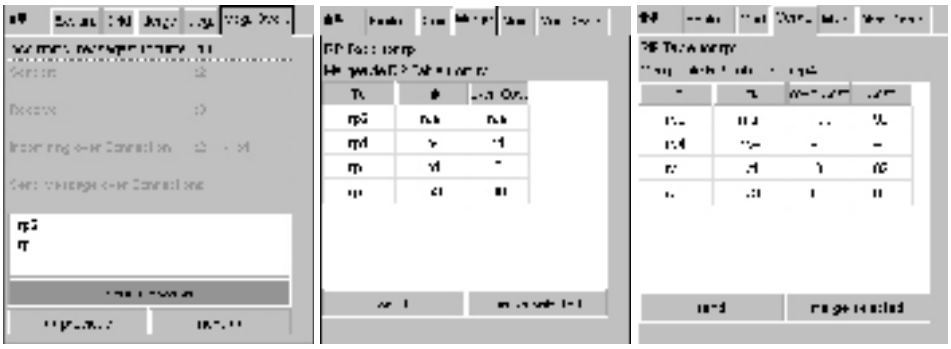

Abbildung 2: Beispiele der Clientanwendung bei der Erarbeitung des *RIP*

Der Verzögerungsnachteil soll im abschließenden *OSPF Protokoll* behandelt und erläutert werden und wie durch das Wissen über die komplette Struktur eines Netzwerkes in sehr kurzer Zeit eine alternative Route für ein Paket gefunden werden kann.

5.3 OSPF Protokoll

1. Phase: Da es sich bei der Oberfläche um eine leichte Abwandlung der im *RIP* Protokoll verwendeten handelt, ist eine Eingewöhnungsphase wie in den vorigen Simulationen hier nicht notwendig. Vielmehr sollen sich die Teilnehmer mit der Suche nach dem kürzesten Pfad in einem Graphen mittels des *Dijkstra Algorithmus* beschäftigen. Diesen können sie anhand einer graphischen Darstellung der Netzwerkstruktur erstellen. Durch das Versenden von Nachrichten und der Aufgabe der Weiterleitung an einen Empfänger wird dies gefestigt, wie es bereits aus den vorigen Protokollsimulationen bekannt ist.

2. Phase: Die zweite und letzte Phase soll zeigen wie einfach es ist, nach Verlust einer Verbindung eine alternative Route durch das Netz zu finden, wenn man über die Struktur des Netzes informiert ist. Hierzu wird wieder der Verlust einer Leitung simuliert. Da der Wiederaufbau der korrekten Struktur für eine einführende Simulation zu komplex erscheint, werden alle Knoten über den Verlust informiert, so dass sich der Wiederaufbau der neuen Netzstruktur massiv vereinfacht.

Am Ende dieser Simulationen sollen die Teilnehmer ein gutes Gefühl für die Problematik der Routingprotokolle bekommen haben. Durch eine Evalution sollten erste Erfahrungen im Umgang mit dem entwickelten System gesammelten werden, die im nachfolgenden Abschnitt kurz vorgestellt werden.

6 Evaluation, Curriculum

Die teilnehmenden Schüler des Informatikkurses der Klassenstufe 12 am Auguste-Pattberg Gymnasium, Mosbach (Baden) hatten im Vorfeld keinerlei Erfahrungen und minimales Wissen im Bereich der Rechnernetze und deren Problematik, so dass diese Evaluation nur als erster Funktionstest der Software und des Einsatzes dieser neuen Lehrmethode dienen sollte. Um das zu erarbeitende Themengebiet weiter zu festigen, wurde zudem noch ein Fragenkatalog entwickelt, der von den Schülern zwischen den einzelnen Phasen ausgefüllt werden sollte. Dieser dient als Grundlage für eine Kontrolle über das in den Simulationen erlangte Wissen.

Den Funktionstest bestand die Software dabei ohne auftretende Fehler; lediglich Probleme mit der eingesetzten Hardware verzögerten den Ablauf der Simulationen ein wenig. Die abgegebenen Antworten wiesen auf durchaus positives Lernverhalten der Schüler hin, so dass wir weitere Evaluationen in nächster Zukunft durchführen möchten, um ein breiteres Verständnis für den Lernerfolg zu bekommen. Der durchweg positiven Resonanz der Teil-

nehmer folgte die Entscheidung, für dieses Problemgebiet ein Curriculum zu entwickeln, in dem wir unsere Erfahrungen im Umgang mit der Software und der Evaluation einfließen ließen [Fal04].

7 Zusammenfassung

Nach einer kurzen theoretischen Einführung in die Vorteile multimedialer Systeme, insbesondere Gruppenarbeit-basierter Systeme, die in der Lehre an Schulen und Hochschulen genutzt werden, gaben wir einen Überblick über bestehende partizipatorische Simulations-Umgebungen. Aufgrund der Erfahrungen mit diesen Systemen bestand die Notwendigkeit, ein neues System zu entwickeln, bei dessen Design und Entwicklung besonders auf die einfache Erweiterung durch neue Simulationsmodule geachtet wurde. Einer Übersicht der technischen Voraussetzungen der *PartSim* Umgebung folgte ein Einblick in das Design des Systems. Eine erste Beispielanwendung zum Themengebiet der Netzwerkprotokolle und deren Evaluierung an einer Schule lieferte uns dabei einen ersten Einblick in die Akzeptanz dieser Lehrmethode durch Schüler und Lehrer.

Weitere Evaluationsphasen in Schulen und Hochschulen sollen dabei in den nächsten Monaten neuere Erkenntnisse des Lernverhaltens der Teilnehmer gegenüber herkömmlichen Lehrmethoden geben. Hierzu gilt es, zunächst weitere Themengebiete aus verschiedenen Fachgebieten als Simulationen umzusetzen um eine breitere und repräsentativere Aussage über die allgemeinen Vorteile dieser Lehrmethode zu erhalten. Typische Anwendungen, die hierbei in Betracht kommen, sind zum einen die bereits in der referenzierten Umgebung *NetLogo/ HubNet* umgesetzten Simulationen zum Ausbreitungsverhalten von Seuchen bzw. die zur Steuerung des Vehrkehrsverhaltens in einer Stadt, zum anderen ist eine Simulation zum Thema Börse von besonderem Interesse. Bisherige Simulationen, wie beispielsweise das „Planspiel Börse“ der Sparkassen für Schüler, bieten zwar die Möglichkeit, sich mit diesem Themengebiet zu beschäftigen, geben aber keine Aussage darüber, inwieweit die Transaktionen der Teilnehmer den aktuellen Kurs einer Aktie beeinflussen. Damit kein kooperatives bzw. kompetetives Gruppenspiel erlauben.

Literatur

[CHB01] G.K.W.K. Chung, T.C. Harmon und E.L. Baker. The impact of a simulationbased learning design project on student learning. In *IEEE Transactions on Education*, Vol. 44 (No. 4), Seiten 390–398, San Jose, CA, Nov. 2001.

[EWE05] EWE Soft. http://www.ewesoft.org/. WWW, last visited: Jan. 2005.

[Fal04] G. Falcone. Positionsbasierte partizipierende Simulationen. Master thesis, University of Mannheim, Faculty of Computer Science, Aug. 2004.

[Klo05] E. Kloepfer. http://education.mit.edu/ar/. WWW, last visited: Jan. 2005.

[Mal94] G. Malkin. RIP Version 2 Protocol Analysis. *RFC 1387, IETF*, Nov 1994.

[Moy98] J. Moy. OSPF Version 2. *RFC 2328, IETF*, Apr 1998.

[Sch02] Bruce Schneier. *Angewandte Kryptographie . Protokolle, Algorithmen und Sourcecode in C*. Addison-Wesley, Bonn u.a., Deutschland, 5. Auflage, 2002.

[SUN05] SUN Inc. Java. http://java.sun.com/, last visited: June 2005.

[SWE04] Nicolai Scheele, Anja Wessels und Wolfgang Effelsberg. Die Interaktive Vorlesung in der Praxis. In *DeLFI 2004*, Seiten 283–294, Paderborn, Germany, Sept. 2004.

[TW04] S. Tisue und U. Wilensky. NetLogo: Design and Implementation of a MultiAgent Modeling Environment. In *Agent '04*, Chicago, IL, U.S.A., Oct. 2004.

[WHF99] U. Wilensky, E. Hazzard und R. Froemke. An Extensible Modeling Toolkit for Exploring Statistical Mechanics. In *Seventh European Logo Conference (EUROLOGO'99)*, Sofia, Bulgaria, 1999.

[WK04] Lilli Winschel und Stephan Kopf. Entwicklung einer Börsensimulation der multiagentenbasierten Entwicklungsumgebung NetLogo. Bericht TR04007, University of Mannheim, Faculty of Computer Science, Mannheim, Germany, 2004.

[WS99] U. Wilensky und W. Stroup. Learning through Participatory Simulations: NetworkBased Design for Systems Learning in Classrooms. In *International Conference on Computer Supported Collaborative Learning (CSCL'99)*, Stanford University, CA, U.S.A., Dec. 1999.