

---

# **Vernetzte grafische Simulationen für den schulischen Informatikunterricht**

-

**Eine qualitativ-empirische Studie zu Möglichkeiten und  
Hindernissen bei der Vermittlung von Modellierungs- und  
Programmierungskompetenzen**

Schriftliche Hausarbeit im Rahmen der Ersten Staatsprüfung für das Lehramt  
an Gymnasien und Gesamtschulen

dem Landesprüfungsamt I NRW – Geschäftsbereich Duisburg-Essen

vorgelegt von

Martin Tilmans

Essen, im März 2010

Themensteller: Professor Dr. Michael Goedicke

Specification of Software Systems

Universität Duisburg-Essen, Campus Essen

## Inhaltsverzeichnis

1 Einleitung.....	4
1.1 Motivation.....	4
1.2 Vorgehen und Ausblick .....	4
1.3 Einordnung der eigenen Arbeit.....	5
2 Beschreibung der verwendeten Software.....	7
2.1 Greenfoot.....	7
2.1.1 Funktionsweise .....	7
2.1.2 Bewertung .....	9
2.1.3 Verteilte Simulation mit Greenfoot.....	10
2.2 Konzeption und Programmierung des Spiels.....	10
2.2.1 Spielidee.....	10
2.2.2 Steuerung .....	11
2.2.3 Modellierung durch endliche Automaten .....	12
2.2.4 Kommunikation mit dem MOSES-Server .....	15
3 Theoretische Grundlagen .....	17
3.1 Zur qualitativen Empirie.....	17
3.2 Neue Medien in der Schule .....	18
3.2.1 Mediendidaktik und medienpädagogische Arbeit .....	20
3.2.2 Integration Neuer Medien in den Unterricht.....	22
3.2.3 Exkurs: Behaviorismus, Kognitivismus, Konstruktivismus .....	22
3.2.4 Die veränderte Lehrerrolle .....	26
3.2.5 Didaktische Funktionen von Computersimulationen.....	26
3.3 Kollaboratives Lernen.....	27
3.3.1 Kollaboratives Lernen – eine Begriffsbestimmung .....	29

3.3.2 Vorteile von kollaborativem Unterricht.....	33
4 Die Unterrichtsreihe.....	36
4.1 Über die Schule.....	36
4.2 Beschreibung der Lerngruppe.....	37
4.2.1 Einschätzung des fachlichen Niveaus.....	37
4.3 Einrichtung der Infrastruktur.....	40
4.4 Die Durchführung der Unterrichtsreihe.....	42
4.4.1 Erste Doppelstunde.....	43
4.4.2 Zweite Doppelstunde.....	44
4.4.3 Dritte Doppelstunde.....	45
4.4.4 Vierte Doppelstunde.....	46
4.4.5 Fünfte Doppelstunde.....	48
4.4.6 Sechste Doppelstunde.....	48
4.5 Beobachtungen und Besonderheiten.....	49
4.5.1 Erste Doppelstunde.....	49
4.5.2 Zweite Doppelstunde.....	49
4.5.3 Dritte Doppelstunde.....	51
4.5.4 Vierte Doppelstunde.....	52
4.5.5 Fünfte Doppelstunde.....	53
4.5.6 Sechste Doppelstunde.....	53
4.6 Evaluation der Unterrichtsreihe.....	54
4.6.1 Persönliche Bewertung.....	54
4.6.2 Blitzlicht und zweiter Erhebungsbogen.....	56
5 Fazit.....	61
Anhang.....	63
Literaturverzeichnis.....	81

## Abbildungsverzeichnis

Abbildung 1: Die Greenfoot-Oberfläche .....	8
Abbildung 2: Kontextmenü eines Objekts .....	8
Abbildung 3: Szene aus dem Spiel .....	11
Abbildung 4: Automat "Click" .....	12
Abbildung 5: Automat "Farmer" .....	12
Abbildung 6: Automat "Field" .....	13
Abbildung 7: Automat "FieldRecovery" .....	14
Abbildung 8: Automat "FieldImage" .....	14
Abbildung 9: Automat "Farmhouse" .....	14
Abbildung 10: Automat "Harvester" .....	15
Abbildung 11: Mediendidaktik und medienpädagogische Arbeit .....	21
Abbildung 12: Lerntheorie des Behaviorismus .....	23
Abbildung 13: Lerntheorie des Kognitivismus .....	24
Abbildung 14: Lerntheorie des Konstruktivismus .....	25
Abbildung 15: Modell kollaborativer Unterrichtsarbeit .....	32
Abbildung 16: Ergebnisse der Selbsteinschätzung .....	38
Abbildung 17: Beispiel einer Schülerarbeit.....	45
Abbildung 18: Bewertung der Reihe .....	58
Abbildung 19: Bewertung des Lernzuwachses.....	59

# 1 Einleitung

Die in dieser Arbeit vorgestellte Studie testet und bewertet Möglichkeiten für den Einsatz vernetzter grafischer Simulationen im schulischen Informatikunterricht. Dafür wurde auf wissenschaftstheoretischer Basis eine sechswöchige Unterrichtsreihe im Fach Informatik geplant und in einem Leistungskurs einer 13. Stufe durchgeführt. Der Unterrichtsschwerpunkt lag dabei auf der Erweiterung von Schülerkompetenzen in den Bereichen der Modellierung und Programmierung. Als Basis diente eine mehrspielerbasierte Simulation, mit der die Schülerinnen und Schüler (SuS) im vernetzten Klassenraum gemeinsam arbeiteten.

## 1.1 Motivation

Der heutige moderne Unterricht ist auf das eigenverantwortliche und kollaborative Arbeiten von SuS ausgelegt, die sich ihr Wissen durch aktives Handeln erschließen und konstruieren. Allzu häufig findet dies allerdings in der Praxis noch keine Anwendung. Das liegt zum einen an einem Mangel an Forschungsarbeiten und Einsichten auf diesem Themengebiet und zum anderen an den höheren Anforderungen solcher Unterrichtskonzepte an Lehrerinnen und Lehrer, die deshalb häufig auf traditionelle Ansätze und Methoden zurückgreifen. Gerade der Informatikunterricht bietet jedoch zahlreiche Möglichkeiten moderne Lehr- und Lernmethoden zu implementieren und somit wichtige Kompetenzen zu fördern. Diese Arbeit soll einen Beitrag zur Beseitigung der bestehenden Defizite leisten und Möglichkeiten beleuchten, den Informatikunterricht an heutige Anforderungen der Gesellschaft anzupassen.

## 1.2 Vorgehen und Ausblick

Nach einer Einordnung der eigenen Arbeit in bereits vorhandene Forschungsergebnisse in Kapitel 1.3 werde ich im zweiten Kapitel zunächst die für diese Studie verwendete Software beschreiben. Hierzu gehören die Vorstellung der genutzten Entwicklungsumgebung (Kapitel 2.1) und der daraus entstandenen mehrspielerbasierten grafischen Simulation (Kapitel 2.2).

In Kapitel 3 werde ich dann auf theoretisch relevante Grundlagen aus der Pädagogik eingehen. Insbesondere werden dort der Einsatz Neuer Medien im Schulunterricht (Kapitel 3.2) sowie Formen und Eigenschaften kollaborativer Arbeitsformen untersucht werden (Kapitel 3.3).

Das vierte Kapitel dieser Arbeit beschäftigt sich mit der Vorstellung der Unterrichtsreihe selbst. Ich werde hier meine Vorgehensweise (Kapitel 4.1 - 4.4) sowie gemachte Beobachtungen beschreiben (Kapitel 4.5) und die gewonnenen Ergebnisse kritisch bewerten (Kapitel 4.6).

Die Schlussbemerkungen in Form eines Fazits der hier vorgestellten Studie finden sich in Kapitel 5.

### **1.3 Einordnung der eigenen Arbeit**

Um den Fokus der eigenen Arbeit genau zu beschreiben, soll an dieser Stelle zunächst ein Vergleich und eine Abgrenzung zu bereits veröffentlichten und thematisch ähnlichen Projekten vorgenommen werden.

Ziel dieser Arbeit ist es, Möglichkeiten und Hindernisse im schulischen Informatikunterricht, der auf einer vernetzten grafischen Simulation basiert, sichtbar zu machen. Zum Einsatz von Neuen Medien und spielerischem Lernen gibt es bereits eine Reihe von Forschungsergebnissen. Diese machen jedoch deutlich, dass die am häufigsten genutzte Komponente Neuer Medien reine Übungsprogramme sind, die nach dem *drill-and-practice-Prinzip* funktionieren und somit auf einer behavioristisch orientierten Sichtweise des Lernens beruhen (vgl. Stadtfeld 41 ff). Der diese Programme kennzeichnende Mangel an Interaktivität soll in der hier vorgestellten Unterrichtsreihe vermieden werden. Die SuS bekommen auf verschiedene Weise ständiges Feedback und können darauf reagieren. Dies wird zum einen durch eine grafische Simulationsumgebung realisiert, welche auf Fehler im Programmcode aufmerksam macht und andererseits alle erfolgreichen Programmiersuche sofort am Bildschirm sichtbar macht. Des Weiteren fungiert die Lehrperson während des Unterrichts als unterstützender Tutor. Die SuS sollen ihre Aufgaben zwar mit großer Selbständigkeit bearbeiten, sind aber nicht allein gelassen. Der dritte Aspekt, welcher Interaktivität während der schulischen Arbeit ermöglicht, ist durch die kollaborative Arbeitsform gegeben. Durch die Vernetzung des Klassenraums und das gemeinsame Spielen und Lernen arbeitet

nicht jedes Klassenmitglied isoliert, sondern tritt mit den Mitschülerinnen und Mitschülern in eine gemeinsame Welt ein. Dieser Aspekt ist als äußerst wichtig anzusehen, da hier der größte Unterschied zu vielen anderen Arbeiten begründet liegt. Während sich computergestütztes Lernen nicht nur im Informatikunterricht größer werdender Beliebtheit erfreut und die Forschungsarbeiten hierzu vielfältig sind, so handelt es sich jedoch in den meisten Fällen um Lernprogramme, die in Einzelarbeit und manchmal – besonders im Bereich des E-Learning – mit Unterstützung von Tutoren zu bewältigen sind. Eine Ausnahme stellt beispielsweise die Arbeit von Donna Teague und Paul Roe dar, die die Wichtigkeit von Partnerarbeit bei der Programmierung untersucht haben. Sie stellten fest, dass die Erfolgsquote von Abschlussprüfungen bei den Probanden deutlich höher ausfiel, wenn sie konsequent in Paaren gearbeitet hatten (vgl. Teague, Roe 151 ff). Während der hier vorgestellten Unterrichtsreihe soll kollaboratives Arbeiten auf zwei Ebenen stattfinden. Einerseits soll genau der gerade beschriebene positive Effekt der Partnerarbeit genutzt werden. Andererseits soll jedoch auch kollaborativ auf Klassenebene gearbeitet werden, was durch das gemeinsame Partizipieren in einer virtuellen Welt ermöglicht wird.

Die motivierende Eigenschaft und andere Vorteile von Spielen als Lehr- und Lernmittel sowie eine ganze Reihe weiterer Aspekte im Zusammenhang mit spielerischem Lernen wurden im Rahmen der *International Conference on E-Learning and Games* vorgestellt, so zum Beispiel in der Arbeit *Motivational Factors in Educational MMORPGs: Some Implications for Education* von Hung, Kinzer und Chen (vgl. Chang et al. 174 ff). Die in der hier vorliegenden Arbeit beschriebene Studie untersucht das kollaborative und handlungsorientierte Erlernen von curricularen Bestandteilen des gymnasialen Informatikunterrichts – wie dem Modellieren und Programmieren – mittels einer spielerischen grafischen Simulation und einem Multiplayer-Aspekt. Diese Fokussierung findet in der Veröffentlichung der Konferenz kein Pendant. Auch Rabenstein und Reh bestätigen, dass die Frage nach den Erträgen selbständiger und kollaborativer Unterrichtsformen durch bisherige empirische Untersuchungen noch nicht beantwortet werden konnte (vgl. Rabenstein, Reh 35).

## 2 Beschreibung der verwendeten Software

Das Spiel *Farmer's Valley*, welches als Grundlage der Unterrichtsreihe diente, ist komplett in Java (vgl. Gosling et al.) programmiert. Hierfür wurde die durch Greenfoot bereitgestellte Entwicklungsumgebung genutzt. Greenfoot ist ein Werkzeug, das für den schulischen Unterricht entwickelt wurde. Es eignet sich insbesondere für Programmieraufgaben mit grafischen Elementen. Es erlaubt das Lehren und Lernen von objektorientiertem Programmieren mittels zweidimensionaler Mikrowelten, die auf einfache Art und Weise erstellt werden können. In diesen Welten werden alle erstellten Objekte visualisiert und können miteinander interagieren (vgl. Henriksen und Kölling). Ein zentraler Server, der die Spieldaten aller Spieler in einer Datenbank speichert, beruht auf einem Application Server der Java Enterprise Platform (vgl. Bien). Nachdem er einmal zentral gestartet wurde, greifen alle Nutzer der verteilten Simulation auf ihn zu. Ohne dies ist das Starten des Spiels nicht möglich, da es rein auf den Multiplayer-Modus ausgelegt ist.

### 2.1 Greenfoot

Greenfoot ist eine komplett in und für Java geschriebene Kombination aus einer Entwicklungsumgebung und einem Gerüst, mit dessen Hilfe virtuelle Simulationen in Form von zweidimensionalen Spielwelten erstellt werden können. Besonders gut eignet es sich für die Programmierung von Welten mit visuellen Elementen. Hierfür steht neben den mitgelieferten Basisklassen und -methoden die komplette Java-Sprache zur Verfügung (Henriksen und Kölling „Greenfoot“).

#### 2.1.1 Funktionsweise

Greenfoot ist durch sein spielerisch erscheinendes Äußeres und durch mit der Installation bereitgestellte Beispiele, Tutorien sowie vordefinierte Basisklassen und Methoden für Programmieranfänger geeignet. Es vereint eine grafische Oberfläche, einen Quellcode-Editor und die Möglichkeit, den geschriebenen Code sofort zu kompilieren und auszuführen. Dadurch wird die Komplexität des Erstellens insbesondere von visuellen Welten reduziert. Die Anwendungsoberfläche ist in Abbildung 1 dargestellt.

Per Mausklick lassen sich neue Klassen erstellen und nutzen. Zuvor muss jedoch die Spielwelt einmalig erstellt werden, die immer aus einem Zellen-Design besteht. Die Größe der Welt ist dabei frei wählbar und hängt von der Anzahl der Zellen ab. Deren Maße sind pixelgenau anzugeben.

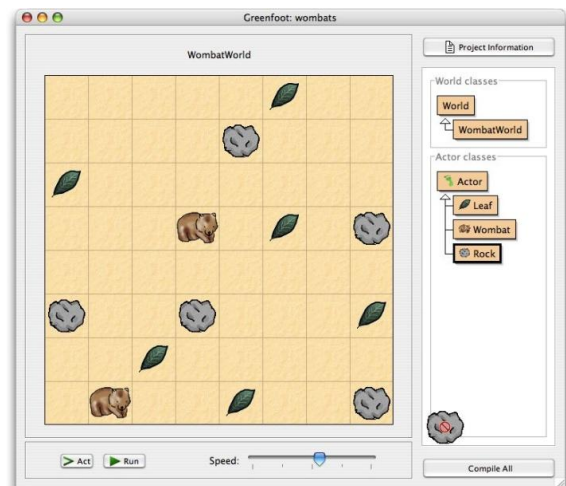


Abbildung 1: Die Greenfoot-Oberfläche

Greenfoot stellt die Basisklassen *World* und *Actor* zur Verfügung.

Diese vererben ihre Eigenschaften an alle Objekte, die im Spiel erstellt werden. Objekte vom Typ *World* spezifizieren dabei die Spielwelt, in welche Objekte des Typs *Actor* eingefügt werden können. Eine zentrale Methode, die alle Aktoren erben, ist die *act()*-Methode. Wird ein Szenario gestartet, verhalten sich die erstellten Objekte, wie es in ihren *act()*-Methoden spezifiziert wurde. Startet man das Spiel, werden alle *act()*-Methoden in regelmäßigen Zeitintervallen aufgerufen. Per Doppelklick auf einen Klassennamen öffnet sich ein Texteditor, um den Quelltext bearbeiten zu können. Hierfür steht eine spezielle Greenfoot-API mit zahlreichen nützlichen Methoden zur Verfügung, wie zum Beispiel dem Abrufen einer Liste aller Objekte der Spielwelt oder dem Festlegen der Koordinaten und Ausrichtung eines bestimmten Objekts.

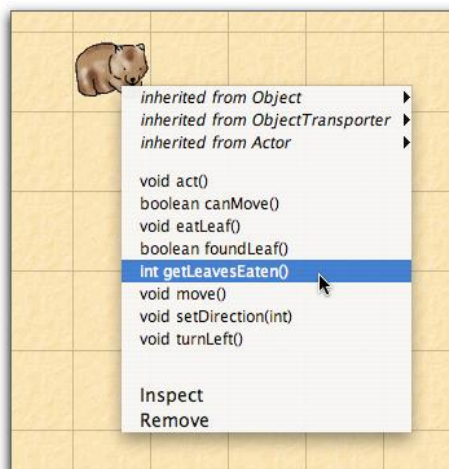


Abbildung 2: Kontextmenü eines Objekts

Ebenfalls per Mausklick lässt sich jedes Objekt genauer inspizieren. Dabei werden Beziehungen zu übergeordneten Objekten dargestellt und alle vorhandenen Methoden aufgelistet. Sie können von dieser Stelle aus einzeln aufgerufen werden. Wird die Option *Inspect* gewählt, so erhält man eine genaue Übersicht über alle Variablen und ihre aktuellen Belegungen. So ist eine punktgenaue Zustandsabfrage eines jeden Objekts jederzeit möglich. Bevor jedoch

einzelne Methoden oder das gesamte Spiel getestet werden können, muss der Quellcode selbstverständlich fehlerfrei sein und kompiliert werden. Beides ist ebenfalls aus der geöffneten Greenfoot-Instanz heraus zu leisten.

### 2.1.2 Bewertung

Greenfoot bietet einige gravierende Vorteile, gerade in Bezug auf die Anwendung im Schulunterricht.

Sehr positiv sehe ich die Aufmachung und Ausrichtung. Greenfoot eignet sich in besonderem Maße zur Erstellung kleiner zweidimensionaler Spiele, die in jedem Entwicklungsstadium direkt am Bildschirm ausgeführt und getestet werden können. Dies ermöglicht den Schülerinnen und Schülern den Erfolg ihrer eigenen Arbeit sehr schnell zu sehen, nämlich wenn sich die von ihnen erstellten Objekte auf der Spielfläche bewegen. Diese Tatsache wird die intrinsische Motivation der meisten SuS zur intensiven Beschäftigung mit der Materie enorm steigern. Beim Programmieren von Datenstrukturen, beispielsweise von verketteten Listen oder Baumstrukturen, die über keine native Repräsentation auf dem Bildschirm verfügen, ist dies häufig nicht der Fall und wird dementsprechend meist als "trocken" angesehen. Es ist gerade für SuS von großer Wichtigkeit zu sehen, dass sie etwas Funktionierendes erschaffen haben, was zudem auch außerhalb des Klassenraums bei Freunden und in der Familie vorgeführt werden kann.

Vereinfacht wird das schnelle Erreichen von Erfolgen durch die mitgelieferten Beispiel-Szenarien. Sie lassen sich per Mausklick laden und daraufhin nach Belieben erweitern. So lässt sich sehr kleinschrittig auf einem bereits bestehenden Gerüst aufbauen, was häufig einfacher ist als eigenständig eins zu erstellen. Es lassen sich vom Lehrer leicht kleine Aufgaben finden, wie die Erstellung einfacher Methoden, die den Objekten ein verändertes Verhalten ermöglichen.

Nicht zu vernachlässigen ist auch die Tatsache, dass Greenfoot – genau wie Java – kostenlos erhältlich ist. Hierdurch fällt der Einsatz im schulischen Umfeld leichter, da keine Finanzierung getätigt werden muss, um Greenfoot als Unterrichtswerkzeug testen zu können.

### 2.1.3 Verteilte Simulation mit Greenfoot

Die Unterrichtsreihe wurde auf der Basis einer verteilten grafischen Simulation geführt, welche mit Greenfoot realisiert wurde. Das für diesen Zweck erstellte Szenario mit dem Namen *Farmer's Valley* ermöglicht es den SuS in einer gemeinsamen virtuellen Welt anzutreten. Dies geschieht im Stile eines Massively Multiplayer Online Role-Playing Game (MMORPG) (vgl. Cole und Griffiths). Als zentraler Server wurde MOSES genutzt, ein experimentelles Projekt des Lehrstuhls *Specification of Software Systems* an der Universität Duisburg-Essen. MOSES liefert einen Adapter, der die gemeinsame Verwendung mit Greenfoot ermöglicht.

## 2.2 Konzeption und Programmierung des Spiels

Im Folgenden werde ich das eigentliche Spiel vorstellen. Dabei gehe ich zunächst auf die grundsätzliche Spielidee ein, die ich dann durch UPPAAL-Automatenmodelle verdeutlichen werde. Ebenfalls werde ich auf die Besonderheit der MOSES-Anbindung eingehen.

### 2.2.1 Spielidee

In *Farmer's Valley* geht es darum, der erfolgreichste Bauer zu sein. Jeder Spieler befehligt einen Bauern, der eine Farm bewirtschaftet. Das Farmland erstreckt sich über ein Tal und besteht aus vielen fruchtbaren Getreidefeldern. Das Getreide muss von den Bauern möglichst schnell geerntet und zur eigenen Farm gebracht werden (Abb. 3).

Für jede heimgebrachte Getreideladung erhält der Bauer Geld auf sein Konto. Hat er genügend Geld verdient, kann er sich einen Mähdrescher kaufen. Mit diesem kann er größere Ladungen auf einmal transportieren, wodurch er sich Wege zur Farm und zurück spart.

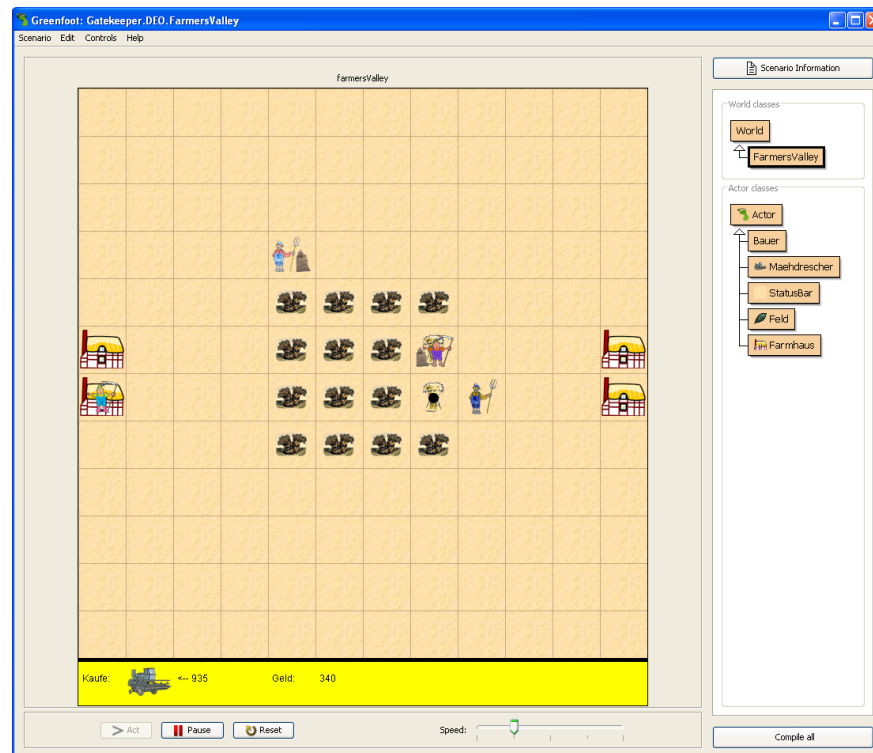


Abbildung 3: Szene aus dem Spiel

Die Felder sind für alle Bauern zugänglich, private Ländereien gibt es nicht. Es darf sich jedoch zu jedem Zeitpunkt nur ein Bauer auf einem Feld befinden. Ein Feature in dieser Spielversion ist die Fruchtbarkeit der Ackergründe. Einmal komplett abgeerntet, regeneriert sich ein Feld selbständig und neues Getreide beginnt zu sprießen. Dieses kann jedoch erst wieder geerntet werden, wenn es ausgereift ist. So lange ist das Feld zwar begehbar (bzw. befahrbar), bringt jedoch keinen Ertrag.

### 2.2.2 Steuerung

Die Steuerung funktioniert in *Farmer's Valley* ausschließlich per Maus. Ist das Spiel einmal gestartet, wird durch einen Mausklick die Zelle ausgewählt, zu der sich die Spielfigur bewegen soll. Wählt man ein Feld aus, das sich nicht in der Regenerierungsphase befindet, startet der Bauer, nachdem er es erreicht hat, sofort mit dem Erntevorgang. Durch das Abliefern der Ladung an der heimischen Farm wird Geld verdient. Ist genügend Geld auf dem Konto, kann – ebenfalls per Mausklick – der Mähdrescher gekauft werden.

### 2.2.3 Modellierung durch endliche Automaten

Die Modellierung des Szenarios wurde mit UPPAAL realisiert. UPPAAL ist ein Werkzeug, mit dem mittels endlicher Automaten Echtzeitsysteme validiert und verifiziert werden können (vgl. Bengtsson et al.). Die grafische Oberfläche und die Fähigkeit, Kommunikation zwischen Automaten mit Hilfe von *Channels* zu modellieren, ermöglichen es, alle Komponenten von *Farmer's Valley* vor der Programmierung zu veranschaulichen und zu testen.

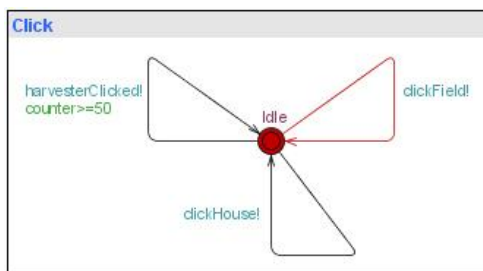


Abbildung 4: Automat "Click"

Gesteuert wird das Spiel – wie bereits erwähnt – komplett mit der Maus (Abb. 4). Jederzeit kann eine beliebige Zelle, ein Feld oder die heimische Farm angeklickt werden, woraufhin sich die Spielfigur dorthin bewegen wird. Falls der Bauer sich zu diesem Zeitpunkt im Bewe-

gungs- oder Erntevorgang befindet, wird dieser abgebrochen und der neue Befehl umgesetzt. Dies ist im Automaten *Farmer* (Abb. 5) modelliert. Hat der Spieler genügend Geld erwirtschaftet – hier gespeichert in der Variable *counter* – kann er sich einen Mähdrescher (Abb. 10) per Klick zulegen.

Es kann auch ein Bereich des Spielfelds angeklickt werden, auf dem weder eine Farm noch ein Feld vorhanden sind. In diesem Fall bewegt sich die Spielfigur dorthin und wartet auf den nächsten Befehl. Dies ist hier nicht modelliert.

Das Herzstück selbst, der Farmer, ist in Abbildung 5 modelliert. Gut zu sehen ist die Möglichkeit, zu jeder Zeit einem Mausbefehl zu folgen. Beispielsweise modelliert durch Clocks ist die Zeit, die benötigt wird, um ein angeklicktes Ziel zu erreichen. Hier beträgt sie fünf oder

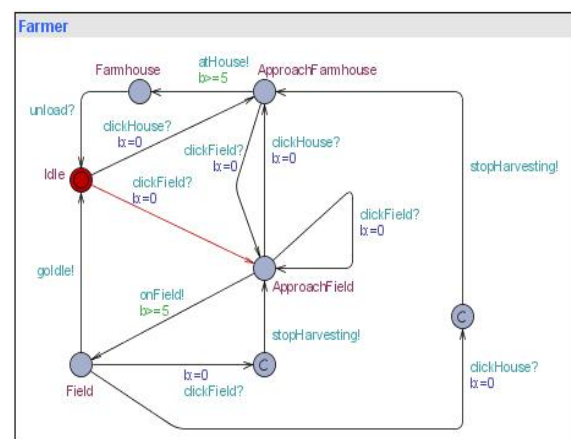


Abbildung 5: Automat "Farmer"

mehr Zeiteinheiten. Die Methoden für die tatsächliche Bewegung sind hier durch *ApproachField* und *ApproachFarmhouse* nur angedeutet.

Falls die heimische Farm oder ein Feld erreicht wird, werden weitere Methoden aufgerufen, die durch die folgenden Modelle deutlicher werden.

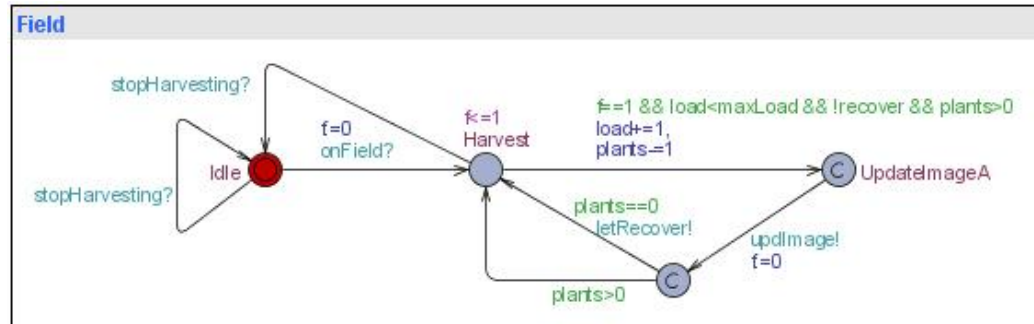


Abbildung 6: Automat "Field"

In Abbildung 6 ist das Modell eines Getreidefeldes zu sehen, auf dem geerntet werden kann. Sobald es erreicht ist, wird der Erntevorgang durch den *Farmer* (Abb. 5) in Gang gesetzt (Sync: *onField*). Auch hier ist die Erntegeschwindigkeit beispielhaft modelliert. Ein Feld besitzt 10 Einheiten Getreide, gespeichert in der Variablen *plants*. Diese wird pro Zeiteinheit um 1 verringert und die *load* des Farmers entsprechend erhöht. Dies ist aber nur möglich, wenn sich das Feld gerade nicht im Recovery-Zustand befindet, repräsentiert durch die boolesche Variable *recover*. Der Bauer und auch der Mähdrescher besitzen ein Limit an Getreide, das mitgeführt werden kann (*maxLoad*). Außerdem wird das angezeigte Bild des Feldes ständig durch *updlmage* aktualisiert. Umgesetzt sind drei verschiedene bildliche Zustandsanzeigen – gut bewachsenes Feld, halb bewachsenes Feld und leeres Feld (Abb. 8). Außerdem wird gekennzeichnet, wenn ein Feld gerade regeneriert. Dies ist in diesem Modell jedoch nicht erfasst.

Wenn das Feld abgeerntet ist, stoppt der Erntevorgang automatisch und der Recovery-Prozess wird durch *letRecover* in Gang gesetzt.

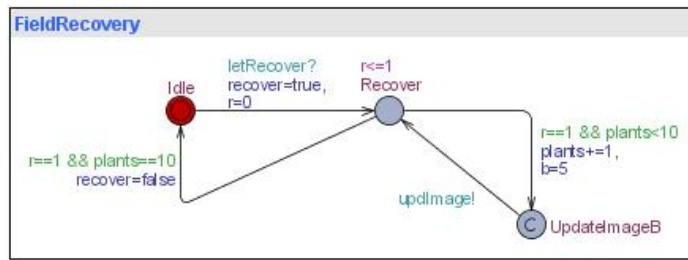


Abbildung 7: Automat "FieldRecovery"

Erholt sich ein Feld, wächst dort neues Getreide, modelliert im Diagramm *Field-Recovery* (Abb. 7). Während dieser Zeit ist das Ernten nicht möglich. Es dauert

eine festgelegte Zeit, bis es wieder voll bewachsen ist und zur erneuten Aberntung freigegeben wird. Während dieser Zeit wird ebenfalls bildlich dargestellt, in welchem Zustand sich das Feld befindet. Ist der Recovery-Vorgang abgeschlossen, wird die boolesche Variable *recover* wieder auf „false“ gesetzt und das Feld damit wieder freigegeben.

In Abbildung 8 ist das Modell zum Aktualisieren des Feld-Bildes dargestellt, welches in der Welt angezeigt wird. Zusätzlich zur Getreidemenge auf dem Feld wird im Spiel ebenfalls grafisch dargestellt, wenn es sich im Erholungsprozess befindet.

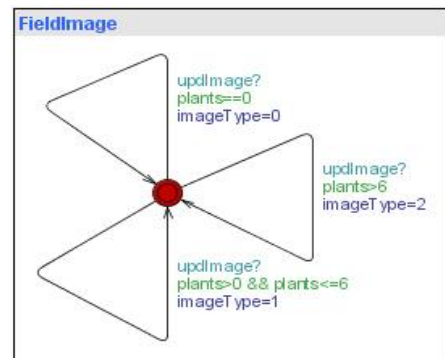


Abbildung 8: Automat "FieldImage"

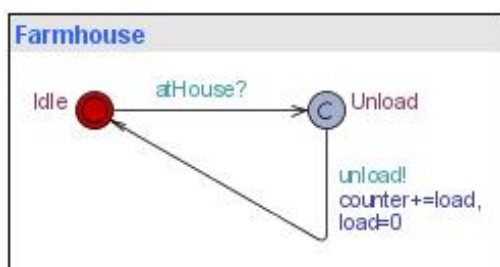


Abbildung 9: Automat "Farmhouse"

Wurde die heimische Farm angefahren, entlädt der Bauer das gesammelte Getreide (Abb. 9). Die Variable *load* wird dementsprechend wieder auf „0“ gesetzt. Gleichzeitig erhöht sich das Guthabenkonto, gespeichert in *counter*.

In Abbildung 10 ist der Kauf eines Mähdreschers beispielhaft dargestellt. In diesem Falle wird die maximale Tragkraft von 10 auf 40 Einheiten erhöht, da ein Mähdrescher mehr Getreide transportieren kann, als ein Bauer auf seinem Rücken.

Zusätzlich ist natürlich ein anderes Bild der Figur auf dem Spielfeld implementiert. Jeder Spieler behält dabei immer seine eigene unverwechselbare Spielerfarbe.



Abbildung 10: Automat "Harvester"

## 2.2.4 Kommunikation mit dem MOSES-Server

Da *Farmer's Valley* ein Multiplayer-Spiel ist, müssen alle Spieler zu jedem Zeitpunkt die gleiche Konfiguration auf dem Bildschirm sehen können. Dies wird dadurch realisiert, dass alle Objekte der Welt zentral vom Server in einer Datenbank gespeichert werden. Das geschieht, indem zuerst alle zu erstellenden Objekte lokal in der eigenen Greenfoot-Instanz erstellt und anschließend zur Welt hinzugefügt werden. Nachdem dies geschehen ist, werden sie in MOSES veröffentlicht. Es werden also die Objekte samt Koordinaten und einer eindeutigen ID, die automatisch von MOSES vergeben wird, in der Datenbank gespeichert. Anhand dieser ID können einzelne Objekte jederzeit abgerufen werden. Nur die eigene Spielfigur, also der Bauer, wird zusätzlich eingeloggt. Per Admin-Interface ist es möglich, sowohl die maximale Anzahl von Logins als auch die Art der Objekte, die sich einloggen dürfen, festzulegen. Gerade bei den Feldern ist darauf zu achten, dass sie nicht mehrfach erstellt werden.

In jedem `act()`-Zyklus werden dann die Objekte, an denen Veränderungen stattgefunden haben, auf dem Server per Update aktualisiert. Schließlich werden Objekte aus der lokalen Welt gelöscht und eine aktuelle Liste von MOSES abgerufen. Nacheinander werden sie dann wieder an ihren jeweiligen Positionen in die Welt eingefügt. Bei der Programmierung stellte sich zu diesem Zeitpunkt heraus, dass die Reihenfolge der vom Server gelieferten Objekte nicht immer der gewünschten entsprach und zufällig zu sein schien. So kam es vor, dass Bilder der Farmhäuser oder Felder vor die der Bauern rutschten und letztere verdeckten. Um hier Abhilfe zu schaffen, wird die gelieferte Liste nun drei Mal durchlaufen. Zunächst werden alle Objekte des Typs "Feld" eingefügt. Danach folgen alle Objekte des Typs "Farmhaus" und schließlich die des Typs „Bauer“.

Je nach Leistungsfähigkeit der eingesetzten technischen Infrastruktur ist die maximale Spielgeschwindigkeit durch die notwendige Kommunikation mit MOSES eingeschränkt. Sollen diese Geschwindigkeitseinbußen im Spiel vermieden werden, ist darauf zu achten, dass ein leistungsfähiger Server mit guter Anbindung, sowie leistungsfähige Client-Rechner eingesetzt werden. Zudem sollte die Server-Client-Kommunikation minimal gehalten werden.

### 3 Theoretische Grundlagen

In diesem Kapitel sollen die theoretischen Grundlagen aus der Pädagogik thematisiert werden, auf denen diese Arbeit beruht. Zunächst werde ich meine Wahl einer qualitativ-empirischen Untersuchungsmethode begründen. Im Anschluss daran werden einige Aspekte zur Arbeit mit Neuen Medien im Schulunterricht aufgegriffen. In diesem Zusammenhang sollen zudem durch einen kurzen Exkurs die drei wichtigsten neuzeitlichen Lerntheorien vorgestellt werden. Dies stellt gleichzeitig den Übergang zur näheren Betrachtung von kollaborativem Lernen im Schulunterricht dar.

#### 3.1 Zur qualitativen Empirie

Das Wort *Empirie* stammt aus dem Griechischen und bedeutet „auf Erfahrung beruhend“. In der empirischen Forschung geht es darum, Erkenntnisse durch systematische Auswertung von Erfahrungen zu gewinnen. Dabei gibt es grundsätzlich zwei zu unterscheidende Ansätze, nämlich den qualitativen und den quantitativen Ansatz. Meine Untersuchung bedient sich hauptsächlich qualitativ-empirischer Elemente. Diese Wahl möchte ich hier kurz begründen.

Die quantitative Empirie stellt das traditionellere Paradigma dar. Sie ist stärker formalisiert und stellt stets die Frage nach Möglichkeiten der Operationalisierung bzw. Quantifizierung der zu erhebenden Merkmale (vgl. Bortz, Döring 137 ff). Untersuchungen dieser Art finden in kontextfreien Situationen statt und gelten als reliabel und wiederholbar (vgl. Storey). Dabei werden im Vorhinein die Merkmale isoliert, die von Interesse sind, und dann Ausprägungen sowie Veränderungen im Laufe der Untersuchung numerisch festgehalten und im Anschluss ausgewertet.

Liegt der Fokus jedoch stärker auf Interpretationen von Beobachtungsmaterial als auf dessen quantitativer Auswertung, so befindet man sich auf dem Gebiet der qualitativ-empirischen Forschung. Hier interagiert der Forschende mit dem Untersuchungsobjekt. Dementsprechend stehen alle auf diese Weise gewonnenen Erkenntnisse in einem Kontext und sind weniger stark formalisiert. Qualitative Methoden bieten sich insbesondere dann an, wenn es eine große Anzahl von Faktoren gibt und sich der Forschende nicht auf die Untersuchung von nur wenigen operationalisierten Variablen festlegen möchte. Storey

schreibt in ihrem Artikel über Theorien, Methoden und Werkzeuge im Forschungsfeld des Programmverständnisses (*program comprehension*), dass die anzutreffenden Bedingungen gerade in diesen Forschungsgebieten häufig sehr komplex seien. Viele Forscher halten hier somit qualitative Ansätze für die interessantesten und wertvollsten.

Obwohl die Methoden in den beiden beschriebenen Paradigmen sehr unterschiedlich sind, werden in der qualitativen Forschung häufig auch quantitative Elemente genutzt und umgekehrt. Extrempositionen, die einen Absolutheitsanspruch für einen der beiden Ansätze fordern, werden immer seltener (vgl. Bortz, Döring 301 ff).

Aufgrund der Fokussierung meiner eigenen Arbeit habe ich mich ebenfalls für einen qualitativ-empirischen Ansatz entschieden, da ich diesen aus den eben genannten Gründen für den geeignetsten halte. Die vorrangig genutzte Methode ist die der teilnehmenden Beobachtung (vgl. Lamnek 239 ff). Teilnehmend allein schon deswegen, da ich die zugrunde liegende Unterrichtsreihe selber geplant, vorbereitet und auch durchgeführt habe. Die gesammelten Ergebnisse sind somit niemals völlig frei von Subjektivität und die daraus entwickelten Hypothesen induktiv gebildet. Dies sind jedoch grundsätzliche Merkmale der qualitativen Empirie und stellen die wissenschaftliche Bedeutung und Haltbarkeit nicht in Frage (vgl. Bortz, Döring 295 ff). Durch die Nutzung von Fragebögen findet sich auch in dieser Untersuchung ein Element aus dem Methodenkanon der quantitativ-empirischen Forschung. Dies soll dazu beitragen, zu weiteren – objektiven – Untersuchungsergebnissen zu gelangen, um diese anschließend in einen Vergleich mit den qualitativ erworbenen Daten und Eindrücken zu stellen und letztendlich differenzierte Schlussfolgerungen ziehen zu können.

### 3.2 Neue Medien in der Schule

„People can learn more deeply from words and pictures than from words alone.“<sup>1</sup> (Mayer 1) Diese einfache Hypothese kann als Ausgangspunkt für die Beschäftigung mit Neuen Medien in Bildung und Ausbildung angesehen werden. Einige bedeutende Fragen sind, wie man Neue Medien nutzen kann,

---

<sup>1</sup> Frei übersetzt: Menschen können durch Worte und Bilder besser lernen als durch Worte allein.

um damit eine Aufwertung des modernen Unterrichts zu erreichen und welcher Art diese Aufwertungen sein können. Dabei geht es auch stets darum, welche Unterrichtsformen und Lernarrangements für einen produktiven Einsatz Neuer Medien am besten geeignet sind.

Gerade moderner Informatikunterricht nutzt Neue Medien in besonderem Maße. Dieser Umstand macht es notwendig, sich mit dem Thema auseinanderzusetzen und ihren Nutzen sowie ihre Einsatzmöglichkeiten deutlich zu machen. Eine Unterrichtsreihe, wie sie im Rahmen dieser Arbeit durchgeführt wurde, die sich vornehmlich mit Themen der Modellierung und Programmierung beschäftigt, kann vielfältig gestaltet werden. Bewusst wurden jedoch in meiner Umsetzung Neue Medien, nämlich eine mehrspielerbasierte grafische Simulation als Basis und als wichtigstes Lernwerkzeug, in den Mittelpunkt gerückt.

Eine umfassende Definition von Neuen Medien und insbesondere eine Abgrenzung von Neuen Medien zu (klassischen) Medien zu finden, ist kein einfaches Unterfangen. In einer Stellungnahme der Kultusministerkonferenz über Neue Medien im Bildungswesen der Bundesrepublik Deutschland wurde sich auf folgende Definition geeinigt:

„Unter den Begriffen Neue Medien und Telekommunikation, insbesondere Multimedia, versteht man diejenigen Medien (mit Texten, Bildern und Tönen) und Informations- und Kommunikationstechniken, die durch die Nutzung von Digitalisierung, Speicherung und algorithmischer Verarbeitung die Verknüpfung und Übermittlung beliebig großer Datenmengen in kürzester Zeit erlauben.“ (zitiert in: Kron, Sofos 33)

Mit dieser Definition gehen Überlegungen zu notwendigen Veränderungen von Lehr- und Lernformen im Unterricht einher, wie z.B. die Selbstorganisation des Wissens und kooperative Arbeitsformen.

Stadtfeld versucht diese noch wenig konkreten Aussagen spezifischer zu betrachten und nimmt eine Klassifizierung Neuer Medien vor (vgl. Stadtfeld 41 ff). Dabei unterscheidet er insgesamt sechs Formen:

- Übungsprogramme
- Tutorielle Systeme
- Simulationen
- Hypermedia
- Anwendungsprogramme
- Computerunterstützte Kommunikationswerkzeuge

Die Grenzen der hier aufgeführten Kategorien sind fließend, weshalb eine eindeutige Zuordnung einer Lehr- bzw. Lernmethode häufig schwer möglich ist. Dennoch lässt sich festhalten, dass Greenfoot am ehesten zur Kategorie der Simulationen gehört. Die Eigenschaften, die Stadtfeld Simulationen zuordnet, umfassen das Auftreten von Interaktion, das Vorhandensein einer Spielkomponente und das Vorkommen von Wettbewerb (vgl. Stadtfeld 44). Alle diese Komponenten sind Bestandteile der Unterrichtsreihe mit *Farmer's Valley*. Erstere und Letztere werden insbesondere durch die kollaborativen Arbeitsformen und die Auslegung des Spiels auf das Mehrspieler-Erlebnis gewährleistet. Bevor wir jedoch zu einer differenzierteren Betrachtung der vorgenommenen Integration eines Neuen Mediums in den Unterricht kommen, sollten zunächst einige grundlegende Begriffe eingeführt und voneinander abgegrenzt werden.

### **3.2.1 Mediendidaktik und medienpädagogische Arbeit**

Mediendidaktik ist eine Teildisziplin der Medienpädagogik (s. Abb. 11). Letztere beschäftigt sich mit Entwicklungs-, Erziehungs- und Bildungsprozessen in der Medienwelt. Sie stellt dabei beispielsweise Fragen nach der Rolle des Medienkonsums in der Freizeitgestaltung junger Menschen und wie Jugendliche Fernsehsendungen wahrnehmen und verstehen.

Die Mediendidaktik ist speziell an den Auswirkungen medienunterstützter Lehr- und Lernprozesse interessiert. Fragestellungen umfassen dabei die Auswirkungen des Einsatzes von technischen Medien auf Lernprozesse, die Veränderung der Lehrerrolle durch Medieneinsatz sowie Chancen und Gefahren, die mit Lehren und Lernen in Computernetzen verbunden sind.

In der medienpädagogischen Arbeit werden die entwickelten Theorien praktisch umgesetzt und kritisch reflektiert. Sie lässt sich in zwei Teilgebiete

unterteilen, nämlich in die didaktische Medienverwendung und -gestaltung sowie die Medienerziehung.

Folgendes Schaubild (Abb. 11) verdeutlicht die vorherigen Ausführungen (Sacher 14):

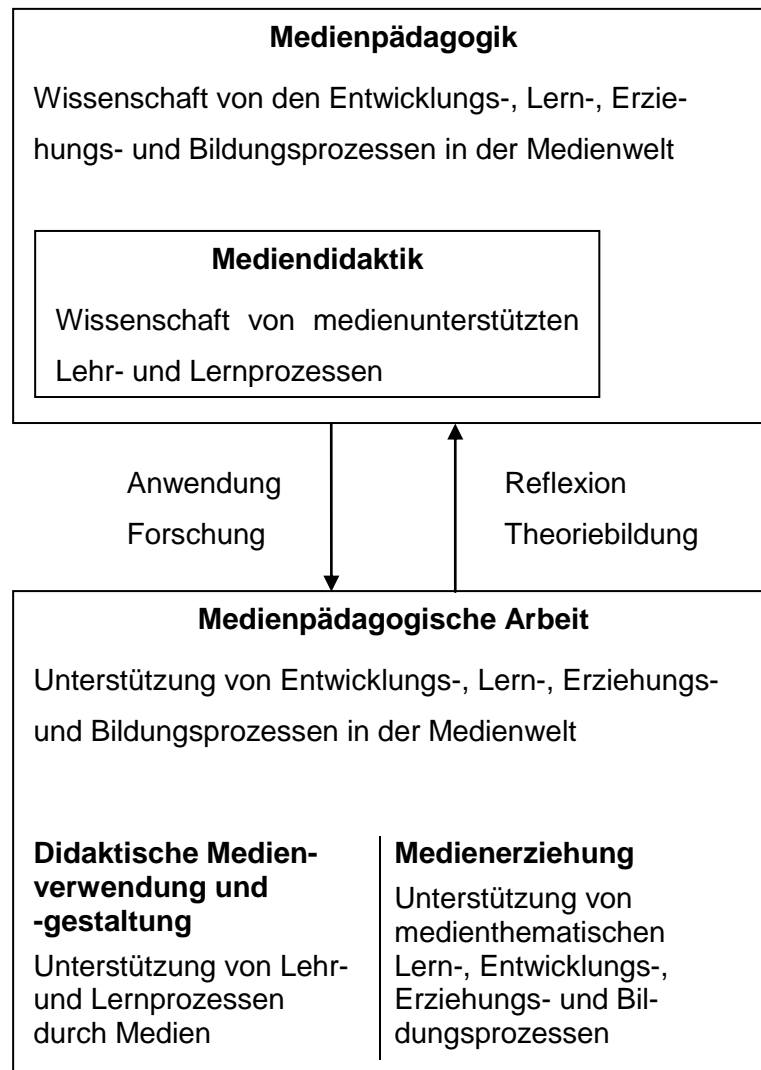


Abbildung 11: Mediendidaktik und medienpädagogische Arbeit

Im Rahmen dieser Arbeit habe ich also – u. a. ausgehend von vorhandenen mediendidaktischen Erkenntnissen, wie sie im theoretischen Teil vorgestellt werden – medienpädagogisch in der Schule gearbeitet. Durch die Reflexion der Ergebnisse, welche im dritten Teil der Arbeit vorgestellt werden, werden schließlich Rückschlüsse auf die am Anfang vorgestellte Fragestellung möglich sein und somit zur weiteren Theoriebildung beigetragen.

Die Medienerziehung stellt laut den Empfehlungen der Kultusministerkonferenz ebenfalls einen wichtigen Teil schulischer Arbeit dar (vgl. Stadtfeld 13), spielt aber in dieser Studie lediglich eine nachgeordnete Rolle. Deswegen wird darauf in dieser Arbeit nicht weiter eingegangen.

### **3.2.2 Integration Neuer Medien in den Unterricht**

Werden Neue Medien in den Schulunterricht integriert, so bringt dies eine Reihe von Veränderungen mit sich. Oftmals wird ein Film, eine Folie oder ein Übungsprogramm als „Extra“ oder gar als Belohnung angesehen, nicht jedoch in das Stundenkonzept eingebettet. Es ist darauf zu achten, sich Medien nicht verselbständigen zu lassen – so auch bei der Arbeit mit dem Computer. „Der Computer ist nur Knecht. Er darf nicht zum Schulmeister werden.“ (Gutheil, Mügge, 28) Stadtfeld verdeutlicht ebenfalls, dass Neue Medien nicht allein deshalb wirksam sind, da sie ein modernes Etikett tragen (vgl. Stadtfeld 143 ff). Vielmehr sind sie ein eigenständiges Strukturelement des Lehr- und Lernprozesses. Unterricht lässt sich schließlich auch durchaus ohne Neue Medien vorstellen. Gleichwohl kann der Gewinn durch ihren wohlüberlegten Einsatz sehr groß sein. Hierzu dürfen sie im Unterricht nicht rein additiv genutzt werden, sondern müssen systemisch implementiert werden. Es setzt sich mehr und mehr die Einsicht durch, dass Neue Medien den klassischen Unterricht nicht vollständig ersetzen können, da computergestütztes Lernen nicht für jedes Qualifikationsziel ideal ist (vgl. Stadtfeld 154). Es wirkt jedoch stark förderlich bei dem Vorhaben, Unterricht auf Grundlage moderner Lerntheorien konstruktivistisch, handlungsorientiert und kollaborativ zu gestalten.

### **3.2.3 Exkurs: Behaviorismus, Kognitivismus, Konstruktivismus**

Behaviorismus, Kognitivismus und Konstruktivismus beschreiben die klassischen Lerntheorien. Dabei ist Letzterer der modernste der drei Ansätze und eine Weiterentwicklung des Kognitivismus. Dieser ist wiederum aus dem Behaviorismus entstanden, welcher den ältesten Ansatz darstellt.

#### *Behaviorismus*

Der Behaviorismus entstand in den 60er Jahren und kann vereinfacht als „Lernen am Erfolg“ (Hamann 8) bezeichnet werden. Es wird dabei davon

ausgegangen, dass auf einen Reiz (*Stimulus*) stets eine bestimmte Reaktion (*Response*) gezeigt wird. Handelt es sich bei der Reaktion um erwünschtes Verhalten, wird dieses positiv verstärkt und somit die Wahrscheinlichkeit erhöht, dass dieses Verhalten erneut gezeigt wird (vgl. Kron, Sofos 87). Negativ gewertete Reaktionen werden hingegen nicht belohnt oder gar sanktioniert, was dazu beiträgt, dieses Verhalten zu löschen. Man spricht deswegen beim Behaviorismus auch von der *Reiz-Reaktions-Theorie* und bezeichnet einen auf diese Weise strukturierten Lernprozess als *operantes Lernen* (Kron, Sofos 88).

Bei dieser Theorie steht der Lerner im Hintergrund. Er erhält lediglich Reize aus seiner Umwelt und zeigt daraufhin eine bestimmte Reaktion. Die Vorgänge, die dabei im Lerner selbst stattfinden, sind nicht von Interesse – das Gehirn stellt die sogenannte *black box* dar (s. Abb. 12 Quelle: Hamann 9).

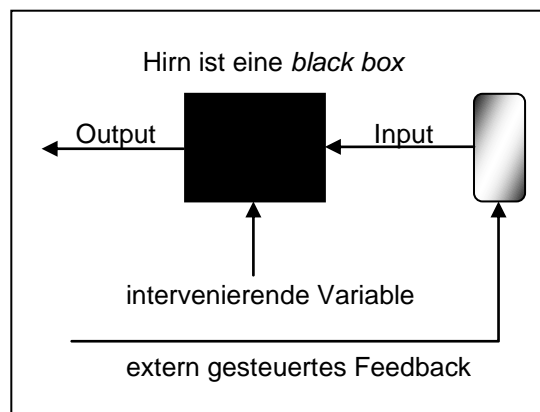


Abbildung 12: Lerntheorie des Behaviorismus

Bei der Arbeit mit Neuen Medien finden behavioristische Ansätze ihre Anwendung besonders in Vokabeltrainern, Rechtschreib- oder Rechenprogrammen.

### *Kognitivismus*

Im Gegensatz zum Behaviorismus wird hier der Lerner nicht mehr als rein passives Objekt gesehen. Stattdessen wird davon ausgegangen, dass er aktiv Informationen verarbeitet. Diese Verarbeitungsprozesse sind dabei von Interesse, weswegen der Lerner im Kognitivismus nicht mehr als eine *black box* dargestellt wird (s. Abb. 13 Quelle: Hamann 12).

Nach Piaget, der einer der führenden Vertreter kognitiver Entwicklungstheorien war, gibt es zwei verschiedene Arten des Lernens, das in diesem Kontext

als „Austauschprozesse des Menschen mit seiner Umwelt“ (Hamann 11) verstanden wird. Dies ist zum einen die *Akkommodation*, bei der bereits vorhandene kognitive Schemata durch neue Erfahrungen verändert werden und die *Assimilation* zum anderen, welche die Anpassung von Erfahrungen beschreibt, so dass sie in vorhandene kognitive Schemata eingefügt werden können.

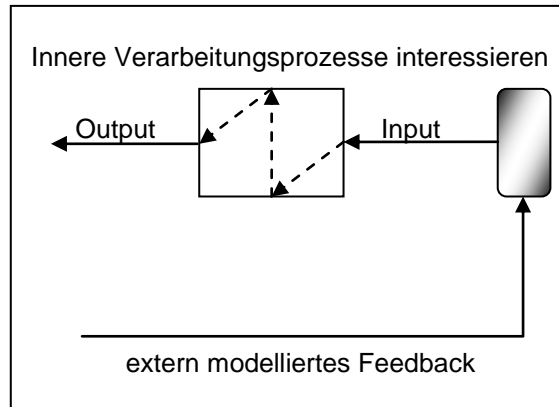


Abbildung 13: Lerntheorie des Kognitivismus

In Bezug auf die Arbeit mit Neuen Medien werden kognitivistische Lernstrategien angewandt, wenn die Gestaltung der Lernumgebung reichhaltig ist und es verschiedene Zugangsmöglichkeiten und Wege gibt.

### *Konstruktivismus*

Der Konstruktivismus ist eine Weiterentwicklung des Kognitivismus. In ihm tritt der Lerner als aktives Subjekt hervor, welcher sich die Wirklichkeit selbst erschließt. „Der Konstruktivismus beschreibt das Lernen nicht als eine Folge des Lehrens, sondern als eigenständige Konstruktionsleistung des Lernenden.“ (Jank, Hilbert 286) Radikale Vertreter dieser Lerntheorie gehen soweit und behaupten, dass jegliches Wissen tatsächlich nur in den Köpfen der Menschen existiert und dort allein durch eigene Erfahrungen und eigenes Handeln konstruiert werden kann. Dieser radikale Standpunkt hätte jedoch zur Folge, dass die Lehre, wie sie in institutionellen Lehranstalten wie der Schule stattfindet, so überhaupt nicht geschehen könnte, da es keinen *Lehrer* geben kann, der in irgendeiner Weise Einfluss auf das Lernen nimmt (vgl. Jank, Hilbert 300). Diese radikale Position wird allerdings selten vertreten. Eine Grundannahme der etwas moderateren konstruktivistischen Sicht ist aber,

dass der Lehrer nicht bestimmen kann, was die Schüler tatsächlich lernen, sondern lediglich die Möglichkeiten zur Wissenskonstruktion schaffen kann. Dabei wird betont, dass es kein *bestes* oder objektiv *wahres* Wissen geben kann, da auch allgemein anerkannte Aussagen – sogenannte *hard facts* – nur solange Gültigkeit haben, bis der Forschungsstand sie überholt (vgl. Reich 75). Der Wissenserwerb ist somit aus Sicht des Konstruktivismus ein selbstbezüglicher Prozess (s. Abb. 14 Quelle: Hamann 14).

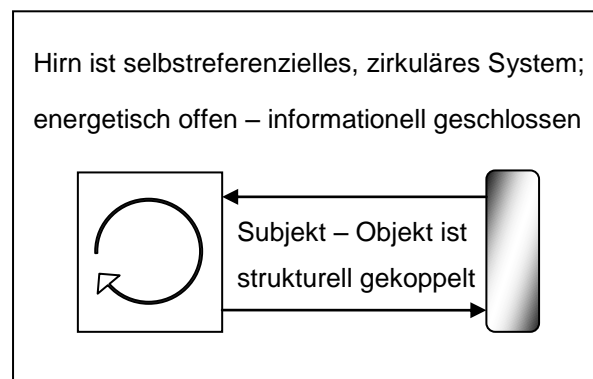


Abbildung 14: Lerntheorie des Konstruktivismus

Es wird deutlich, dass die konstruktivistische Didaktik nur eine praxisorientierte Didaktik sein kann, bei der die Lernenden zu eigenständigem Handeln befähigt werden. Dabei sind die Lerner in drei Dimensionen tätig. Die erste Dimension ist die der Konstruktion, in der neues Wissen und neue Bedeutungen erschaffen werden und mit individuell vorhandenem Vorwissen und kognitiven Strukturen verknüpft werden (*Erfinder der Wirklichkeit*). Dabei muss aber nicht das gesamte Wissen von jedem Individuum erneut entwickelt werden. Vielmehr wird bereits vorhandenes Wissen in der Dimension der Rekonstruktion lediglich neu entdeckt (*Entdecker der Wirklichkeit*). Es ist jedoch wichtig, eigene Standpunkte zu finden und jegliche Informationen kritisch zu hinterfragen, um eine unreflektierte Aneignung von Wissen zu vermeiden. Hier befindet sich der Lerner in der Dimension der Dekonstruktion (*Enttarnen der Wirklichkeit*) (vgl. Gutheil, Mügge 38 f und Reich 138 ff).

### 3.2.4 Die veränderte Lehrerrolle

In der Praxis ist es schwierig, eine reine konstruktivistische Didaktik anzuwenden. Stattdessen fließen häufig auch Methoden in den Unterricht ein, die dem kognitivistischen Ansatz zugeschrieben werden und mit denen die SuS stärker geleitet oder angeleitet werden. Dennoch macht moderner Unterricht eine Veränderung der klassischen Lehrerrolle, in der er als Instrukteur angesehen wird, nötig. So wie die SuS zu Forschern werden, so wird die Lehrkraft mehr und mehr zum Berater (vgl. Kron, Sofos 113). Sie versorgt die Lerner lediglich mit Material und Anreizen und lässt sie sich damit erst einmal selbständig beschäftigen. Aebli nennt dies das „Prinzip der minimalen Hilfe“ (Aebli 300). Eingreifend tätig wird der Lehrer erst dann, wenn es offensichtlich kein Fortkommen mehr gibt. Aber auch in diesem Falle werden lediglich neue Denkanstöße gegeben, beispielsweise durch zielführende Fragen.

Der Lehrer muss bei dieser Unterrichtsform, in der die Schüler-Arbeitsphasen sehr lang sind, die Lernfortschritte und -schwierigkeiten seiner SuS genau beobachten. Die Ergebnisse dieser Beobachtungen müssen in die zur Verfügung gestellten Aufgaben und Lernangebote einfließen, um Möglichkeiten einer inneren Differenzierung zwischen stärkeren und schwächeren SuS zu gewährleisten (vgl. Gutheil, Mügge 42).

### 3.2.5 Didaktische Funktionen von Computersimulationen

Die Verwendung von Computersimulationen eröffnet dem Lehrer zahlreiche neue Möglichkeiten bei der Unterrichtsgestaltung (vgl. Sacher 163 f). So schreibt Sacher, dass sie die Motivation und die Konzentration heben und damit einen Anreiz zum selbständigen Lernen liefern.

Des Weiteren ermöglichen sie eine handelnd-experimentierende Auseinandersetzung mit dem Unterrichtsstoff im Sinne einer konstruktivistischen Didaktik. Computersimulationen haben dabei eine Veranschaulichungsfunktion inne. Das bedeutet, dass jegliche Änderungen, die innerhalb der Simulation durchgeführt werden, einen sofortigen Effekt haben, der für die SuS sichtbar ist. So lassen sich Erfolge und Misserfolge direkt auf das eigene Handeln zurückführen.

Mit der Veranschaulichungsfunktion einher geht eine weitere Stärke von Computersimulationen. Sie liegt darin, dass sie komplexe Systeme verständlicher machen. Dadurch werden Zusammenhänge für die SuS deutlich, die sonst nur schwer oder gar nicht nachvollziehbar wären. Als Folge der Beschäftigung mit komplexen Themen können sowohl die Problemlösekompetenz als auch die analytisch-synthetische Denkfähigkeit verbessert werden.

Weiterhin schreibt Sacher, dass Computersimulationen sowohl geeignet sind, deklaratives Wissen (Begriffs- und Faktenwissen) als auch prozedurales Wissen (Fertigkeiten und Handlungswissen) und metakognitives Wissen (Kenntnis von Strategien zum Wissenserwerb und heuristischen Strukturen) zu vermitteln. Sie ermöglichen außerdem eine sinnvolle Behandlung von Modellbildung im Unterricht. Dabei können Modelle einerseits aus vorhandenen Simulationsprogrammen entstehen oder aber als Vorlage für einen zu simulierenden Algorithmus dienen.

Ein Punkt, der zum nächsten Themenbereich im Verlauf dieser Arbeit führt, ist der, dass Computersimulationen aufgrund ihres stark interaktiven Charakters kollaboratives Lernen begünstigen. In der hier vorgestellten Unterrichtsreihe geschieht dies auf zwei Ebenen. Zum einen durch die gewählten Sozialformen, da der größte Anteil der Unterrichtszeit aus Schülerarbeitsphasen in Partner- oder Kleingruppenarbeit besteht. Zum anderen wird durch den MOSES-Server und die dadurch erfolgte Ausrichtung auf das Mehrspieler-Erlebnis der gesamte Klassenraum vernetzt und gruppenübergreifende Interaktion und Kollaboration ermöglicht und unterstützt.

### **3.3 Kollaboratives Lernen**

Kollaborative Unterrichtsformen haben in den letzten Jahren immer stärkeren Aufwind bekommen. Einer der Gründe hierfür ist in der Veröffentlichung der Ergebnisse der PISA-Studien zu finden, die den pädagogischen Diskurs in Deutschland erschüttert haben (vgl. Rabenstein, Reh 23). Aufgrund der ernüchternden Ergebnisse im internationalen Vergleich sieht man sich gezwungen, grundsätzliche Änderungen in der Schul- und Unterrichtsstruktur zu vollziehen. Selbstständigkeitsfördernde und kollaborative Arbeitsformen gelten dabei weithin als eine geeignete Möglichkeit, kognitiv anregendes fachliches Lernen zu etablieren. Unterstützt wird diese Ansicht durch moderne kognitivis-

tisch-konstruktivistische Lerntheorien, die das Lernen als selbständige und aktive Tätigkeit des Subjekts begreifen, wie weiter oben bereits beschrieben.

Neben den steigenden fachlichen Anforderungen, die an SuS gestellt werden, trägt eine sich rasch verändernde Umwelt zur Aufwertung kollaborativer Unterrichtsmethoden bei. Der enorme gesellschaftliche, soziale, kulturelle und wirtschaftliche Wandel der letzten Jahrzehnte hinterlässt seine Spuren auch im Schulsystem (vgl. Weidner 18 ff). So bedeutet das fortschreitende Aufbrechen von traditionellen Familienstrukturen einen Zuwachs an pluralistischen und individuellen Lebensstilen. Die Kernfamilie, bestehend aus Vater, Mutter und Kindern, ist nicht mehr das unangefochtene Familienmodell. Die Schule erhält durch diese Entwicklung einen wachsenden erzieherischen Auftrag, welcher Defizite aus der Erziehung und der Lebenssituation, die die SuS in ihren Elternhäusern erfahren, ausgleichen muss.

Zudem geht die fortschreitende Individualisierung der Menschen unserer Gesellschaft konträr mit vermehrten Forderungen der Arbeits- und Berufswelt nach Teamfähigkeit und sozialen Kompetenzen einher. Sie stellen zwei der zentralen Schlüsselqualifikationen dar, die die Wirtschaft heute unmissverständlich einfordert. In Zeiten der Globalisierung wird somit kollaboratives Teamlernen immer wichtiger. „In unserer pluralistischen Gesellschaft ist die Fähigkeit, zusammenzuarbeiten und Unterschiede zu erkennen und zu akzeptieren, eine Grundvoraussetzung, um bei den Lernenden ein Gefühl der Zusammengehörigkeit, der Toleranz und des gegenseitigen Respekts zu entwickeln.“ (Green, Green 32) Die Industrie- und Handelskammer Nordrhein-Westfalen stellt die Bedeutung kollaborativen Lernens für den Einstieg in die Berufswelt heraus: „Nicht Eigenbrötler, auch nicht einsame Tüftler sind in der Regel gefragt, sondern auf Kooperation, auf den Austausch von Informationen, Erfahrungen, Verbesserungsvorschlägen ausgerichtete Mitarbeiter. Zusammenarbeit im Betrieb ist zwingend. Vor allem die neuen betrieblichen Organisationsformen sind wesentlich auf Kooperation ausgelegt.“ (zitiert in: Weidner 21)

Die Allgegenwart der Medien stellt heutzutage eine große Herausforderung an SuS dar. Fernseher, Spielkonsole, Computer, MP3-Player und Handy sind zu ständigen Begleitern geworden, die in Massen Einzug in das Leben der Jugendlichen gehalten haben. Tatsächliche Sinneserfahrungen werden zunehmend von Bildern ersetzt. Dies geschieht zudem meist unbegleitet und

unreflektiert, sodass Jugendliche schon bald abgestumpft gegenüber den dargestellten Inhalten sind (vgl. Weidner 20). Während die Schule Fairness und Toleranz propagiert, zeigen die Medien häufig Gewalt und Brutalität. In der Schule muss somit gelernt werden, dass Kollaboration zum Erfolg führt und nicht Rücksichtslosigkeit und Egoismus. Sie muss lehren, was das Fernsehen nicht kann: „Askese, Disziplin, Konzentration, Stillsitzen, Durchhaltevermögen, Abstrahieren, Forderungen ertragen, Stillstand aushalten, ohne sich zu langweilen, abwarten können, sachlich sein können, Grenzsetzungen ertragen können.“ (Weidner 20)

Weidner betont, dass kollaboratives Lernen eine hervorragende pädagogische Antwort auf alle geschilderten Anforderungen bietet. Von fachlicher Seite bestätigt Bruce Joyce in seinem Buch „Unterrichtsmodelle“, in dem er 80 Unterrichtsstrategien vergleichend vorstellt, dass der Hauptgrund für von ihm beobachtete zunehmende Schülerleistungen kollaboratives Lernen gewesen sei (vgl. Green, Green 32).

### **3.3.1 Kollaboratives Lernen – eine Begriffsbestimmung**

Im Rahmen dieser Arbeit wird stets von kollaborativem Lernen und kollaborativen Unterrichtsmethoden gesprochen. In der Literatur wird meist nicht zwischen den Begriffen der Kooperation und der Kollaboration unterschieden – sie werden synonym verwendet. Manchmal jedoch werden verschiedene Kooperationstypen unterschieden. Rabenstein und Reh, beispielsweise, stellen das „Nebeneinanderher-Arbeiten, Helfen und Kollaborieren“ (Rabenstein, Reh 32) vor. Dabei kommt es beim Nebeneinanderher-Arbeiten nur zu punktuellm Austausch zwischen den SuS, während sich das Kollaborieren durch eine intensive, aufgabenbezogene Interaktion und Zusammenarbeit auszeichnet. Ich habe mich deshalb dazu entschieden, in dieser Arbeit einheitlich den Begriff der Kollaboration zu verwenden, um die Wichtigkeit des gemeinsamen Arbeitens und der Interaktion zwischen den SuS zu betonen.

Der Grundgedanke kollaborativen Arbeitens lässt sich weiter entfalten (vgl. Konrad, Traub 5). Er zeichnet sich dadurch aus, dass Lernen stets in einem Kontext stattfindet, der den Einzelnen zur Kollaboration anregt, um Probleme zu identifizieren und gemeinsam zu lösen. Durch die Heterogenität der Lerngruppe werden der Gruppe ein breit gefächertes Spektrum an Vorerfahrungen und Kompetenzen zur Verfügung gestellt, die es gemeinsam produktiv zu

nutzen gilt. Diese stellen eine Bereicherung des Lernens dar. Der interaktive Austausch und Diskussionen innerhalb der Lerngruppe optimieren das Verstehen und die Reflexion des jeweiligen Lerngegenstands und trainieren gleichzeitig soziale Kompetenzen der Beteiligten. Beim kollaborativen Lernen werden zudem affektive Dimensionen und das subjektive Erleben mit eingeschlossen und umfassen somit neben sozialen auch emotionale Herausforderungen.

Wie bereits weiter oben erwähnt, wird kollaboratives Lernen als ein aktiver und konstruktiver Prozess verstanden. In diesem Zusammenhang bezeichnet handelndes Lernen eine „aktive, aus einem Verständigungsprozess erwachsende selbständige und selbstverantwortete Auseinandersetzung mit einer als bedeutsam eingeschätzten Situation. Sie kann umwelt- oder selbstgerichtet sein und hat bildende Funktion.“ (Konrad, Traub 26)

Damit einher geht das Abrücken von *Lehrzielen*, bei denen der Lehrer im Mittelpunkt des Geschehens steht. Lehrziele sind all die Ziele, bei denen der Lehrer einen Lernstoff – meist im Auftrag der Bildungsinstitutionen und der Gesellschaft – mit seinem Unterricht vermitteln möchte: „Die Lernenden sollen...!“ Ziel ist hierbei der sogenannte Wissenstransfer. Davon unterschieden werden heute *Lernziele*. Diese sind die konkreten Fähigkeiten und Kompetenzen, die sich die Lernenden durch den Unterricht selbst aneignen sollen. Die Lehrkraft bemüht sich also primär, Lernen zu ermöglichen und dafür eine reichhaltige Umgebung zu schaffen, die dann kollaborativ entdeckt werden kann (vgl. Buchegger et al. 39). Dabei ist nicht nur das Produkt der Arbeit von Interesse. Weidner schreibt: „Die Gruppenprozesse beim kollaborativen Lernen sind mindestens genauso wichtig wie das Arbeitsprodukt.“ (Weidner 29)

Kollaborative Unterrichtsphasen werden häufig in Phasen frontalen Unterrichts mit Unterrichtsgespräch eingebettet. Diese sind wichtig, um Vorwissen zu aktivieren, die Aufgabenstellungen mitzuteilen und zu spezifizieren und um nach den Schülerarbeitsphasen die Ergebnisse vorzustellen und zu sichern. Das Gerüst einer Unterrichtseinheit der dieser Arbeit zugrunde liegenden Unterrichtsreihe ist modellhaft in Abbildung 15 dargestellt.

Die Aktivierung des Vorwissens ist wichtig, um die SuS auf die Erarbeitung des neuen Themas vorzubereiten, die Motivation zu steigern und die Einord-

nung neuen Wissens in bereits vorhandenes zu ermöglichen. Die Gruppenbildung kann zu Beginn oder aber direkt vor der Gruppenarbeitsphase stattfinden. Das Vorstellen von Schülerergebnissen im Plenum ist wichtig, um SuS die Möglichkeit zu geben, die Arbeit von anderen Gruppen genau zu betrachten und Rückmeldungen geben zu können. So können mögliche Unterschiede der Lösungen herausgestellt und bewertet werden. Gute Ergebnisse können im Anschluss daran gesichert werden. Verbesserungswürdige Lösungen gehen mit dem Feedback der Klasse an die jeweiligen Gruppen zurück.

Der genaue Unterrichtsablauf wird von der Lehrkraft inhaltsbedingt von Stunde zu Stunde angepasst.

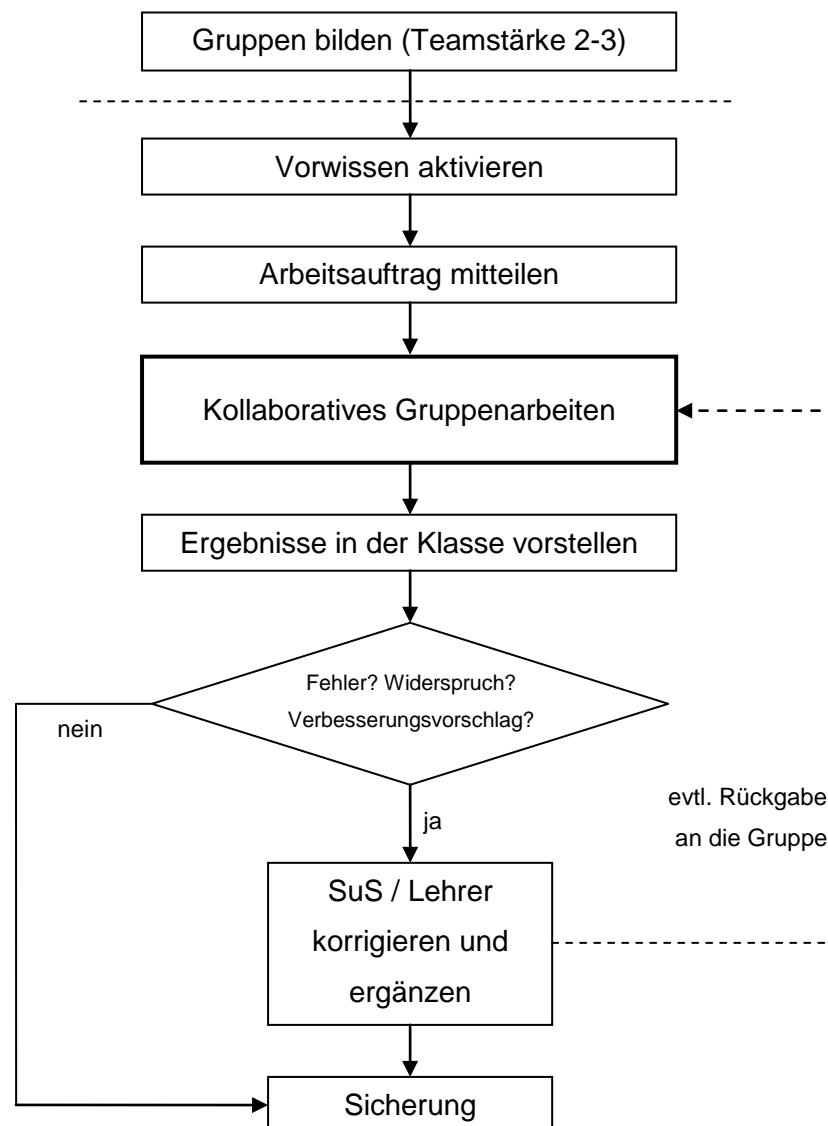


Abbildung 15: Modell kollaborativer Unterrichtsarbeit

Zentraler Bestandteil des kollaborativen Unterrichtsmodells sind die Gruppenarbeitsphasen. Häufig werden fünf Basiselemente genannt, die nötig sind, um produktives kollaboratives Arbeiten zu ermöglichen (vgl. Green, Green 76 ff und Weidner 35 ff). Der Lehrer muss den SuS helfen, diese Elemente zu kennen, zu verstehen und anzuwenden. Letztlich werden sie Werkzeuge zum Problemlösen und zum erfolgreichen Arbeiten in Gruppen:

- **Positive gegenseitige Abhängigkeit**  
Jedes Gruppenmitglied fühlt sich in der Erreichung des Gruppenziels miteinander verbunden. Dazu muss jeder Einzelne erfolgreich sein. Dabei werden Ressourcen geteilt und gegenseitige Unterstützung gewährleistet.
- **Persönliche Übernahme von Verantwortung**  
Jedes Gruppenmitglied trägt Verantwortung für sich selbst und für die Gruppe. Man ist daran interessiert, dass die Lernresultate jedes Mitglieds maximiert werden.
- **Interaktion von Angesicht zu Angesicht**  
Die besten Ergebnisse werden durch rege Interaktion erreicht. Hierfür sollten sich alle teilnehmenden SuS räumlich nahe sein. Unterstützend kann dabei zusätzlich eine Vernetzung des Klassenraums wirken, der gruppenübergreifende Interaktion ermöglicht.
- **Sozial- und Teamkompetenz**  
Hierunter fallen Interaktionsfertigkeiten, die es Gruppen ermöglichen, produktiv zu arbeiten, wie z.B. sich abwechseln, ermutigen, einander zuhören, Hilfe anbieten, sich gegenseitig loben und Differenzen oder Konflikte konstruktiv klären.
- **Evaluation der Gruppenprozesse**  
Die gemeinsamen Anstrengungen und Ergebnisse werden bewertet und Verbesserungen angestrebt. Hierfür muss ausreichend Zeit während des Unterrichts zur Verfügung gestellt werden.

### **3.3.2 Vorteile von kollaborativem Unterricht**

Einige der Vorteile von kollaborativem Arbeiten im Schulunterricht wurden bereits in den vorhergehenden Ausführungen genannt. Diese sollen hier noch einmal aufgegriffen, vervollständigt und zusammenfassend dargestellt werden.

Kollaborative Unterrichtsformen unterstützen eine individuelle Förderung der SuS im Klassenraum. Dies stellt bei Regelklassengrößen von etwa 30 SuS häufig eine Herausforderung dar (vgl. Brüning, Saum 83). Durch das Aufbrechen der Klasse in Gruppen können die einzelnen Lerner unterschiedliche Herangehensweisen wählen und die gestellten Aufgaben verschieden voneinander bearbeiten. Dies kommt dem individuellen Lernstil und Lerntempo

jedes Schülers entgegen. Zeitgleich findet durch den ständigen Austausch der Gruppenmitglieder untereinander eine Förderung der schwächeren Schüler statt. Die stärkeren Schüler hingegen profitieren vom eigenen Erklären und gelangen so zu einer tieferen Verarbeitung des Unterrichtsgegenstandes. Zusätzlich hat der Lehrer die Möglichkeit, einzelnen Gruppen verschiedene Aufgaben zu geben, beispielsweise mit unterschiedlichem Schwierigkeitsgrad. Außerdem können Hilfsmittel vorbereitet werden, die nur an die SuS ausgegeben werden, die auch nach längerer Beschäftigung mit einem Thema in der Bearbeitung einer Aufgabe keinen Fortschritt erzielen. Die Gestaltung eines solchen Unterrichts erweist sich als besonders erfolgreich, wenn eine entsprechende Lernumgebung verwendet wird, die kollaboratives Arbeiten begünstigt (vgl. Bräu 177 ff). Dies wird in der hier beschriebenen Studie durch die grafische Simulation mit Greenfoot und dem zugrunde liegenden Server für das Mehrspielererlebnis geleistet.

Konrad und Traub heben den positiven Effekt hervor, den kollaboratives Lernen auf die Vermeidung von tragem Wissen hat (vgl. Konrad, Traub 159 ff). Als *träges* Wissen bezeichnet man solches Wissen, das zwar prinzipiell vorhanden ist, in konkreten Situationen aber nicht abgerufen und zum Einsatz gebracht werden kann. Der Grund dafür liegt hauptsächlich in der Unterrichtsstruktur und den angewandten Lern- und Lehrmethoden. Häufig dominiert die Vermittlung von reinem Faktenwissen, welches nicht in bestehendes Vorwissen integriert und durch mangelnde Handlungsmöglichkeiten der Schüler nicht tief genug durchdrungen wird. Damit ist es wenig anwendungsbezogen und zu abstrakt. Träges Wissen bleibt somit auf den Kontext beschränkt, in dem es erworben wurde und kann nur in diesem Rahmen genutzt werden. Aktive und konstruktive Lernprozesse hingegen, wie sie mit kollaborativen Arbeitsformen einhergehen, wirken dem Entstehen von tragem Wissen entgegen. Die SuS entdecken ihr Wissen selbst und probieren es zugleich eigenverantwortlich und selber handelnd aus. Dies führt gleichzeitig zur Entwicklung von wertvollen Problemlösekompetenzen und zur Fähigkeit, auf höherem Niveau zu denken.

Durch die aktive Auseinandersetzung mit dem Unterrichtsgegenstand erreicht man eine stärkere Schülerbeteiligung. Das Erlebnis, Dinge selbst erarbeitet zu haben, führt zu einer Steigerung des Selbstwertgefühls (vgl. Green, Green 33 ff). Dies wiederum bedingt häufig eine höhere Lernzufriedenheit der SuS.

Sie entwickeln eine positive Haltung gegenüber den behandelten Themen, was den weiteren Lernprozess begünstigt.

Durch die ständige Interaktion mit anderen Klassenmitgliedern und dem Lehrer werden kommunikative und soziale Kompetenzen geschult und das Vertrauen zueinander gestärkt. Die Fähigkeit, miteinander im Team zu arbeiten, wird trainiert. All dies sind Schlüsselqualifikationen, die in der Schule vermittelt und eingeübt werden sollten, da sie in der Arbeits- und Berufswelt von zunehmender Wichtigkeit sind.

## 4 Die Unterrichtsreihe

Dieser Teil der Arbeit dient der Beschreibung und Evaluation der durchgeführten Unterrichtsreihe. Nach einer kurzen Vorstellung der Schule werde ich zunächst die Lerngruppe charakterisieren. Im Folgenden gebe ich einige Hinweise zum Einrichten der nötigen Infrastruktur, die auf meinen eigenen Erfahrungen beruhen. Daraufhin werden der Ablauf aller Doppelstunden und die Beobachtungen, die ich machen konnte, erläutert. Abschließend folgt die Bewertung der gesammelten Ergebnisse.

### 4.1 Über die Schule

Das „Lessing“ in Düsseldorf bietet zwei Schulen unter einem Dach<sup>2</sup>. Es ist sowohl Gymnasium als auch Berufskolleg. Das außergewöhnliche Profil der Schule wird insbesondere durch die „sieben Wege zum Abitur“ geprägt. Vier Wege davon können auf dem gymnasialen Zweig eingeschlagen werden und bezeichnen jeweils bestimmte Schwerpunktsetzungen in der Fächerwahl. So ist bereits vor dem Eintritt in die Oberstufe zwischen den Lernfeldern Gesellschaft und Geschichte, Sprache und Kultur, Sport und Physik sowie Erziehung und Ethik zu wählen.

Das Berufskolleg bietet drei weitere Wege zum Abitur. Der erste Bildungsgang führt mit dem Schwerpunkt Mathematik und Informatik zur allgemeinen Hochschulreife. Mathematik und Informatik sind dabei die ersten beiden Abiturfächer und werden ab Eintritt in die Oberstufe fünfständig unterrichtet. Als drittes Abiturfach kann später Deutsch oder Englisch gewählt werden. Das vierte (mündliche) Abiturfach muss aus den Fächern Philosophie und Religion oder aus dem gesellschaftswissenschaftlichen Bereich gewählt werden. Der zweite Bildungsgang führt zum Abitur mit Ausbildung zum staatlich geprüften Freizeitsportleiter. Hierbei liegt der Fokus auf der breitensportlichen Förderung. Das „Lessing“ ist seit 2007 die erste anerkannte NRW-Sportschule. Im dritten Bildungsgang wird das Abitur mit gleichzeitiger Berufsausbildung zum biologisch-technischen Assistenten erreicht.

---

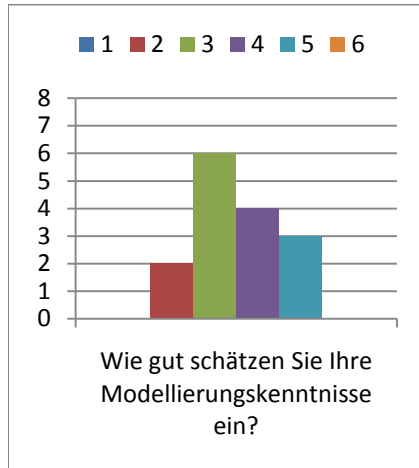
<sup>2</sup> Schulhomepage mit zahlreichen Informationen: <http://www.bk-lessing.eschool.de/>

## 4.2 Beschreibung der Lerngruppe

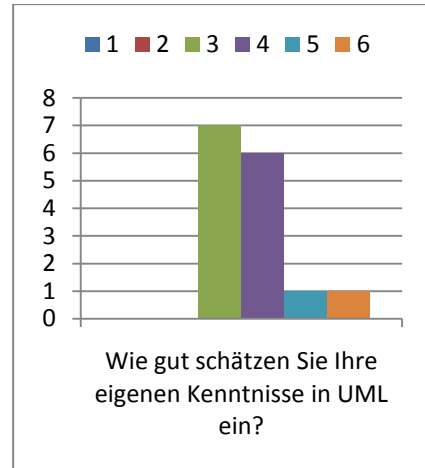
Die Unterrichtsreihe wurde in einem Leistungskurs der Stufe 13 durchgeführt. Dieser Kurs ist Teil des aktuellen Abiturjahrgangs, der den Bildungsgang mit Schwerpunkt Mathematik und Informatik auf dem Berufskolleg verfolgt. Er besteht aus fünf Schülerinnen und zwölf Schülern. Ein Schüler war jedoch nur zu einer Sitzung anwesend, weswegen nur die Antworten von 16 statt 17 SuS in die Auswertung der Erhebungsbögen weiter unten eingehen. Der Unterricht mit *Farmer's Valley* wurde in sechs aufeinanderfolgenden Doppelstunden jeweils freitags durchgeführt – also in zwei der fünf wöchentlichen Informatikstunden.

### 4.2.1 Einschätzung des fachlichen Niveaus

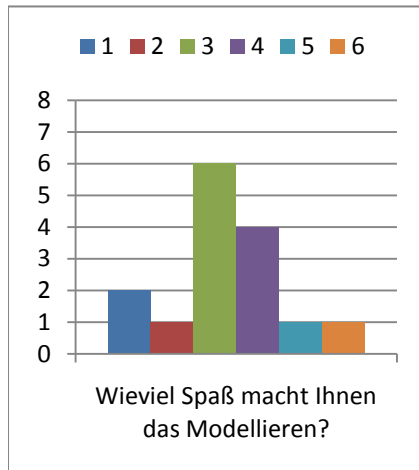
Um das fachliche Leistungsniveau der SuS besser einschätzen zu können, wurde zu Beginn der ersten Doppelstunde ein Erhebungsbogen ausgeteilt, der zur Ermittlung grundlegender Modellierungs- und Programmierkenntnisse diente (s. Anhang A). Die ersten sechs Fragen, die mit einer numerischen Bewertung von eins bis sechs (sehr gut/sehr viel bis sehr schlecht/sehr wenig) beantwortet werden sollten, dienten der Selbsteinschätzung der SuS. Drei der Fragen beschäftigten sich mit dem Thema der Modellierung, die anderen drei mit dem der Programmierung. Die Skala wurde dabei absichtlich so gewählt, damit es keine „goldene Mitte“ gab und selbst bei Unentschlossenheit in eine Richtung tendiert werden musste. Im Anschluss daran folgten Fragen, die sich ebenfalls mit Themen der Modellierung und Programmierung beschäftigten und textlich beantwortet werden mussten. Hier sollten Aufgaben gelöst werden, wie beispielsweise die Deklaration einer `for`-Schleife in Java und das Durchlaufen dieser. Dies sollte objektive Schlüsse über das Fachwissen der SuS zulassen und eine bessere Interpretation der Ergebnisse aus der Selbsteinschätzung ermöglichen.



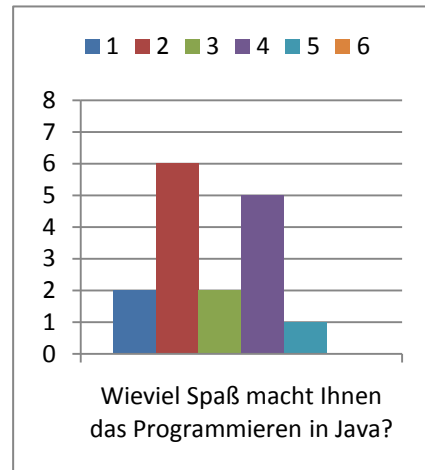
a



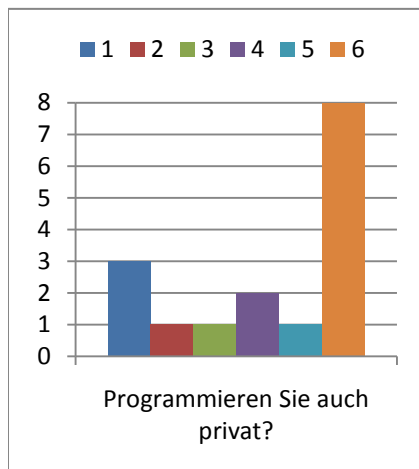
b



c



d



e



f

Abbildung 16: Ergebnisse der Selbsteinschätzung

Die Frage nach der Einschätzung der eigenen Modellierungskennntnisse wurde von den SuS mit mittelmäßig bis schlecht bewertet. Nur zwei SuS hielten ihre Kennntnisse für gut (s. Abb. 16 a). Dieses Ergebnis bestätigte sich in der geforderten Auflistung von bekannten Modellierungstechniken, bei der mit endlichen Automaten bereits ein Beispiel vorgegeben war. Sechs SuS gaben überhaupt keine Antwort auf die Frage, genauso viele Antworteten mit „Automaten“ und eine(r) nannte den Namen eines Programms, mit welchem verschiedene Diagrammtypen gezeichnet werden können (DIA<sup>3</sup>). Nur drei SuS nannten beispielsweise die *Unified Modeling Language* (UML), eine Sprache zur Modellierung von Software und anderen Systemen (vgl. Fowler). Dies ist insofern verwunderlich, als dass UML bereits einmal Unterrichtsgegenstand gewesen war. Der abiturrelevante Diagrammtyp des Klassendiagramms war bereits eingeführt worden und den meisten SuS ein Begriff. Jedoch wurde in vielen Antworten auf die Frage, was UML sei und wofür man es nutzen könne, das Klassendiagramm mit UML gleichgesetzt und behauptet, dass UML (ausschließlich) zum Zwecke der übersichtlichen Darstellung von Klassen mit ihren Attributen und Funktionen sowie ihren Beziehungen untereinander – wie sie beispielsweise durch Vererbung angelegt werden – entwickelt worden sei. Die SuS schätzten ihre eigenen Kennntnisse in UML ebenfalls mittelmäßig ein, fast auf dem gleichen Niveau wie ihre allgemeinen Modellierungskennntnisse (s. Abb. 16 b). Die Frage, wieviel Spaß das Modellieren machen würde, erhielt eine breit gefächerte Bewertung. Die meisten Bewertungen fanden sich jedoch auch hier im mittleren Bereich der Skala wieder (s. Abb. 16 c). Ein(e) Schüler(in) enthielt sich bei den ersten drei Fragen mit der Begründung, dass ihr/ihm Modellierung kein Begriff sei.

Bei den Fragen bezüglich der Programmierung bot sich ein anderes Bild. Die Ergebnisse waren hier weiter gestreut.

Spaß am Programmieren in Java hatten etwa die Hälfte der Kursteilnehmer (Bewertungen von eins bis zwei). Die andere Hälfte empfand eher keine Freude daran (Bewertungen von vier bis fünf). Es deutete sich bei den Ergebnissen eine Tendenz zur Aufteilung in zwei Lager an (s. Abb. 16 d). Als gute Voraussetzung für die anstehenden Unterrichtsstunden war zu interpretieren,

---

<sup>3</sup> DIA ist das zentrale Modellierungsprogramm im Informatikunterricht des Lessing-Berufskollegs. Es ist ein Freeware-Programm zum Zeichnen von strukturierten Diagrammen. Weitere Informationen finden sich unter: <http://dia-installer.de> (deutsch) und <http://live.gnome.org/Dia> (englisch).

dass nur ein(e) Schüler(in) eine Bewertung von fünf gegeben hatte und es keine Person gab, die überhaupt keinen Spaß am Programmieren hatte. Eine relativ geringe Anzahl von SuS programmierte allerdings auch regelmäßig außerhalb des schulischen Unterrichts. Nur drei Schüler taten dies sehr häufig. Acht Personen gaben an, überhaupt nicht privat zu programmieren (s. Abb. 16 e).

Die größte Streuung aller Ergebnisse ergab die Frage nach der Einschätzung der eigenen Java-Kenntnisse (s. Abb. 16 f). Durch einen umfangreichen Teil im Erhebungsbogen sollten diese genauer ermittelt werden. So wurde beispielsweise verlangt, Variablen und for-Schleifen zu deklarieren sowie mit Bedingungen und Konsolenausgaben zu arbeiten. Am Ende der Unterrichtsreihe wurden dann dieselben Programmieraufgaben noch einmal gestellt, um mögliche Unterschiede in der Qualität der Antworten feststellen zu können. Die Aufgaben fragten nur grundlegendes Wissen ab, wie beispielsweise Methodenaufrufe, die Deklaration von Variablen sowie das Arbeiten mit Arrays und Kontrollstrukturen. Sechs SuS gaben dabei mangelhafte oder gar keine Antworten. Hier schien auch Grundlagenwissen nicht verfügbar zu sein und wurde deswegen vor der Arbeit mit *Farmer's Valley* noch einmal wiederholt. Dieser Auswertung nach hätte die Einschätzung eigener Java-Kenntnisse im Klassendurchschnitt schlechter ausfallen müssen. Einige SuS erreichten jedoch exzellente Ergebnisse und bezeugten fundiertes Wissen. Bei einem Teil des Kurses spiegelte sich lückenhaftes Wissen wider. Das bekannteste der abgefragten Elemente war die for-Schleife, die den Großteil der SuS weder semantisch noch syntaktisch vor Probleme stellte. Die explizite Frage nach Kontrollstrukturen in Java wurde sehr schwach beantwortet. Nur vier SuS waren überhaupt in der Lage, Kontrollstrukturen zu nennen. In Anbetracht der soeben beschriebenen Ergebnisse lag dies wahrscheinlich hauptsächlich daran, dass zwar mit solchen bereits gearbeitet wurde, der Fachbegriff aber unbekannt geblieben war.

Alle ausgefüllten Erhebungsbögen sind in Anhang C zu finden.

### 4.3 Einrichtung der Infrastruktur

Vor Beginn der Unterrichtsreihe wurde der Computerraum für den anstehenden Unterrichtsbetrieb vorbereitet. Dabei stellte sich rasch heraus, dass die

Einrichtung des Servers und der Client-PCs eine größere Herausforderung bieten würde als zunächst angenommen. Trotz aller vorhergegangenen Tests mit unterschiedlichen Hard- und Software-Konfigurationen bedurfte es dreier Vormittage bis das System zufriedenstellend funktionierte. Die aufgetretenen Schwierigkeiten sollen hier kurz skizziert werden, um sie bei zukünftigen Arbeiten einplanen oder verhindern zu können.

Die vorhandene Infrastruktur der Computerräume des Lessing Gymnasiums und Berufskollegs wird von der Stadt Düsseldorf gewartet. Dies geht unter anderem mit eingeschränkten Rechten für die Nutzer des Systems einher. Auch mit einem Lehrer-Zugangskonto können nur Daten auf einem fest zugewiesenen Laufwerk abgelegt und dauerhaft gespeichert werden. Installationen, die Änderungen auf dem Systemlaufwerk tätigen, sind nicht möglich, da sämtliche Veränderungen nach einem Neustart des Computers rückgängig gemacht werden und der Computer somit automatisch wieder in seinen Ursprungszustand versetzt wird. Dasselbe gilt für alle Schüler-Zugangskonten. Daher ist es von großem Vorteil, dass sowohl MOSES als auch Greenfoot inklusive des Szenarios auf die Festplatte kopiert werden können und keine Installation benötigen. Die erforderliche SQL-Datenbank kann ebenfalls ohne Installation vollständig konfiguriert auf das Lehrer-Laufwerk kopiert und vor Inbetriebnahme des Servers gestartet werden. Dies wurde mit MoWeS<sup>4</sup> realisiert, einem Programm, welches die Distribution einer Datenbank enthält.

Eine weitere Schwierigkeit, bedingt durch die externe Verwaltung der Computerräume, ergab sich dadurch, dass alle nutzbaren Lehrer- und Schüler-Laufwerke nicht lokal verfügbar waren, sondern dass es sich hierbei um Netzlaufwerke handelte. Dies hatte zur Folge, dass die Performanz des Systems sowohl unter der Server-Client-Kommunikation als auch unter Festplattenzugriffen der Client-Computer litt. In der vorbereiteten Version, die unter verschiedenen Test-Konfigurationen äußerst zügig gearbeitet hatte, war die Leistung auf den Computern der Schule zu niedrig, um annehmbar zu sein. Es war also nötig, jegliche Kommunikation auf das Nötigste zu reduzieren. Das Szenario wurde dahingehend geändert, dass fortan nur noch Objekte vom Server abgefragt wurden, an denen seit dem letzten Abruf Veränderungen stattgefunden hatten. Der Aufwand, den die einzelnen Clients zu leisten hatten

---

<sup>4</sup> MoWeS steht abkürzend für Modulares Webserver System. Weitere Informationen unter: <http://www.chsoftware.net/de/useware/mowes/>

wurde verringert, indem auch nur noch diese empfangenen Elemente in der Welt neu gezeichnet wurden. Nur so war es möglich, acht Client-Computer zeitgleich in zufriedenstellender Geschwindigkeit laufen zu lassen. Die einzigen Verzögerungen im Spiel, die sich nicht verhindern ließen, traten auf, wenn mehrere Felder zeitgleich regenerierten und acht Bauern zur selben Zeit ihre Dienste verrichteten. Dies führte zu kurzzeitigen Überlastungen des Systems, welche jedoch mit den abgeschlossenen Regenerationszyklen der Felder wieder beseitigt waren.

Beim Programmieren von Szenarien ist darauf zu achten, dass jeder Nutzer seine Arbeit kompilieren können muss, ohne dass dies negativen Einfluss auf den Rest des Netzwerks hat. Der aktuelle Zustand des Spiels soll hierbei nicht verloren gehen. Deshalb ist es notwendig, dass beim Kompilieren und nachdem die eigene Spielfigur aus der Welt gelöscht wurde, dasselbe gespeicherte Objekt vom Server abgerufen wird, um erneut den Verweis der eigenen Spielfigur darauf setzen zu können. Dies wird am besten dadurch erreicht, dass beim Erstellen des Objekts die automatisch von MOSES erstellte eindeutige ID durch eine manuell gewählte ID ersetzt und in einer Variablen gespeichert wird. Das eigene Spielobjekt kann somit stets per Angabe der ID vom Server abgerufen werden. Hierfür wurden die entsprechenden Methoden auf dem Server implementiert. Beim manuellen Setzen der IDs ist darauf zu achten, dass diese nie doppelt vergeben werden, da dies im Betrieb zum Absturz des Servers führt.

Eine Auflistung serverseitig verfügbarer Methoden findet sich in Anhang E.

#### **4.4 Die Durchführung der Unterrichtsreihe**

Hier soll nun zunächst der genaue Ablauf aller Unterrichtsstunden der Reihe beschrieben werden. Insgesamt wurden sechs Doppelstunden an aufeinanderfolgenden Freitagen durchgeführt. In den ersten zweieinhalb Doppelstunden wurde ein neuer UML-Diagrammtyp eingeführt und angewendet. Der Rest der Reihe beschäftigte sich mit Programmieraktivitäten.

Ein Großteil der Unterrichtsstunden bestand aus kollaborativen Schülerarbeitsphasen. Hierfür arbeiteten meist zwei SuS in Partnerarbeit. Zweiergruppen bieten die größten Interaktionsmöglichkeiten innerhalb einer Gruppe und eignen sich somit sehr gut, um kollaborative Kompetenzen zu nutzen und zu

verbessern (vgl. Green, Green 103). Dabei wurde darauf geachtet, dass zu keinem Zeitpunkt zwei leistungsschwache Schüler eine Gruppe bildeten. Gab es in einer Unterrichtsstunde eine ungerade Anzahl von SuS oder kam jemand nach Abwesenheit in einer vorherigen Doppelstunde zu einem bereits bestehenden Arbeitsauftrag mit festen Gruppen, so wurden Gruppen aus drei SuS gebildet.

#### 4.4.1 Erste Doppelstunde

Die erste Stunde begann mit der Vorstellung meiner Person und Intention. Im Zuge dessen wurde auch Greenfoot per Beamer gezeigt, die grundsätzliche Funktionsweise erklärt und das Szenario *Farmer's Valley* präsentiert. Den SuS wurde außerdem die Möglichkeit gegeben, Fragen über mich, das Spiel und die kommenden Unterrichtsstunden zu stellen. In der zweiten Hälfte der Stunde wurde der erste Erhebungsbogen ausgeteilt (s. Anhang A) und von den SuS ausgefüllt.

Die zweite Stunde stellte den Einstieg in die Thematik der Modellierung dar. Es wurde zunächst mit dem Aktivitätsdiagramm ein bis dahin unbekannter UML-Diagrammtyp vorgestellt. Dazu diente eine Folie, mit der ein einfaches Beispiel-Diagramm per Overhead-Projektor gut sichtbar an die Wand projiziert wurde (s. Anhang F). Gemeinsam sollten alle sichtbaren Elemente gesammelt und benannt werden. Da die Namen der meisten Elemente nicht bekannt waren, wurden aus der Semantik des Beispiels heraus Vermutungen angestellt, welche Funktion sie erfüllten. Anschließend benannte ich sie auf der Folie. In der nächsten Stunde wurde das Ergebnis für alle SuS als Informationsblatt ausgeteilt (s. Anhang G).

Im Anschluss daran begann die erste kollaborative Schülerarbeitsphase. Die Klasse erhielt den Arbeitsauftrag, sich ein eigenes Realwelt-Beispiel ausdenken und dies mit Hilfe eines Aktivitätsdiagramms zu modellieren. Die SuS hatten dabei die freie Wahl ihrer Themen. Ich gab ihnen lediglich mit der Funktionsweise eines Getränkeautomaten oder mit einer Kontoeröffnung einige Beispiele zu möglichen Szenarien. Als Programm für alle Modellierungsarbeiten wurde DIA genutzt. In den letzten 15 Minuten wurden per Beamer zwei Schülerarbeiten gezeigt. Die entsprechenden SuS stellten ihre Ergebnisse selbst vor und der Rest der Klasse kommentierte diese. Stärken

und Schwächen wurden herausgearbeitet und Verbesserungsvorschläge gesammelt.

Alle Ergebnisse befinden sich in Anhang H.

#### 4.4.2 Zweite Doppelstunde

Die zweite Doppelstunde begann mit der erneuten Vorstellung eines Arbeitsprodukts aus der vorangegangenen Woche. Dies sollte der Anknüpfung an die vorherige Stunde und der Aktivierung der erworbenen Kenntnisse dienen.

Danach durften die SuS zum ersten Mal selbständig *Farmer's Valley* spielen und alle Funktionen ausprobieren. Dabei wurden die Elemente von Greenfoot und die Arbeitsweise noch einmal am Beispiel erklärt, Fragen gesammelt und für alle beantwortet. Kurz vor Ende der ersten Stunde wurde der Abschluss dieser Entdeckungsphase eingeläutet.

Daraufhin bekamen die SuS einen neuen Arbeitsauftrag. Der Ablauf einer Methode aus dem Spiel sollte durch ein Aktivitätsdiagramm modelliert werden. Hierzu diente die Methode `act()` aus der Klasse *Feld* (s. Anhang I). Diese Methode sorgt dafür, dass sich ein abgeerntetes Feld wieder regeneriert und zur Verdeutlichung seines Zustands das auf dem Spielfeld angezeigte Bild angepasst wird. Als Hilfestellung hatte ich eine DIA-Datei vorbereitet, die das Skelett eines möglichen Modells ohne jegliche Beschriftungen enthielt. Diese Datei sollte den Gruppen zur Verfügung gestellt werden, die mit der Bearbeitung der gestellten Aufgabe auch nach hinreichender Beschäftigung nicht zurechtkamen. Das Skelett müsste dann verstanden und ausgefüllt, nicht jedoch selbst erstellt werden. Diese Hilfe musste jedoch nur von einer Gruppe in Anspruch genommen werden, nachdem sie der Lösung auch nach 20-minütiger Arbeitszeit noch nicht näher gekommen war.

Am Ende der Doppelstunde wurden wieder zwei Lösungen per Beamer vorgestellt und von der Klasse kommentiert. Die meisten Gruppen hatten zu diesem Zeitpunkt ein fertiges oder beinahe fertiggestelltes Modell.

Die Ergebnisse sind in Anhang J zu finden.



(s. Anhang K). Alle Gruppen erhielten zu Beginn der Stunde Feedback zu ihrer Arbeit.

Der nun folgende Arbeitsauftrag bestand darin, im Klassengespräch Möglichkeiten zur Umsetzung einer Erweiterung der `act()`-Methode zu sammeln. Die veränderte Methode zur Regeneration der Felder sollte von einer Wettervariablen abhängig sein. Dabei sollte das Wetter zwei Ausprägungen annehmen können. Bei schlechtem Wetter würde das Regenerieren länger dauern als bei gutem Wetter. Die entsprechenden Bilder für das Szenario hatte ich vorbereitet und stellte sie den SuS zur Verfügung. Die Ideen der SuS wurden zuerst unkommentiert an der Tafel gesammelt. Außer einiger ordnender Bemerkungen meinerseits wurde die Bewertung der gesammelten Beiträge den SuS überlassen, die daraufhin in die nächste Gruppenarbeitsphase gingen, in der sie eine mögliche Umsetzung der Wetter-Erweiterung in ihr vorliegendes Modell integrierten.

Zu Beginn der zweiten Stunde wurden einige verschiedenartige Lösungen vorgestellt und ihre jeweiligen Vor- und Nachteile im Plenum herausgearbeitet. Die SuS wurden daraufhin mit der ersten Programmieraufgabe bedacht, welche die Implementierung der eigens modellierten Änderungen im Quellcode des Szenarios darstellte. An diesem Auftrag arbeiteten die Gruppen bis zum Ende der Stunde, welche mit der kurzen Präsentation eines vielversprechenden Zwischenergebnisses abgeschlossen wurde.

#### **4.4.4 Vierte Doppelstunde**

Doppelstunde Vier begann mit der Rekapitulation der letzten Unterrichtseinheit. Nach dieser kurzen Wiederholung, die von mehreren SuS geleistet wurde, setzten die Gruppen ihre Arbeiten fort und beendeten die Programmieraufgabe. Für den Fall, dass eine Gruppe viel früher als die anderen fertig geworden wäre, hatte ich Material für eine Zusatzaufgabe vorbereitet. Ich hätte mir von dieser Gruppe gewünscht, dass sie mit Hilfe des gesamten Szenarios und einem von mir vorbereiteten Informationsblatt, welches die wichtigsten Server-Methoden übersichtlich darstellt und beschreibt (s. Anhang E), einen kleinen Vortrag zur Server-Client-Kommunikation für eine der folgenden Stunden vorbereitet. Es stellte sich jedoch heraus, dass die von mir veranschlagte Zeit auch für die schnellen Gruppen keine großen zusätzlichen Spielräume zuließen. Deshalb wurde diese Aufgabe nicht verwendet.

Zum Ende der Gruppenarbeit bekamen die SuS die Möglichkeit, noch einmal mit dem Szenario zu spielen, um die Effekte ihrer Arbeit und die unterschiedlichen Lösungen erfahrbar zu machen. Zwei differierende Lösungen wurden daraufhin mit dem Beamer an die Wand projiziert und nach ein paar Minuten Bedenkzeit von jeweils gruppenfremden SuS erläutert. Das heißt, dass die SuS fremden Quelltext lesen, verstehen und erklären mussten. Dabei wurden auch Vor- und Nachteile der Ergebnisse zur Sprache gebracht. Alle Lösungen befinden sich in Anhang M.

Mit der zweiten Stunde begannen die Vorbereitungen zur zweiten und letzten Programmieraufgabe dieser Unterrichtsreihe. Der Arbeitsauftrag wurde zunächst von mir nur mündlich präsentiert. Die Aufgabe bestand darin, eine Methode in der Klasse *Bauer* zu erstellen, die verschiedene Konsolenausgaben erzeugen sollte. Der Aufruf dieser Methode sollte durch einen Mausklick auf ein in die Statusleiste einzufügendes Dollarzeichen geschehen. Das Bild wurde später von mir zur Verfügung gestellt. Die SuS sollten sich nun vorstellen, dass ihre Spielfigur – der Bauer – eine Familie hätte, bestehend aus ihm selbst, seiner Frau und zwei Kindern. Mit der Methode sollte nun zu jedem Zeitpunkt das bis dahin verdiente Geld mit einem festgelegten Schlüssel unter den Familienmitgliedern für verschiedene Zwecke aufgeteilt und die Ergebnisse ausgegeben werden. Zusätzlich sollte eine Ausgabe in der Konsole erfolgen, die mögliche Ersparnisse berechnete und je nach Höhe dieses Betrages eine natürlichsprachliche Aussage über die Höhe traf. Das Arbeitsblatt mit der genauen Aufgabenstellung ist in Anhang N zu finden. Es wurden nun zunächst im Klassengespräch Ideen zur Umsetzung dieses Auftrags unbewertet an der Tafel gesammelt. Auch konkrete programmiersprachliche Konstrukte, die zur Implementierung nötig sein würden oder könnten, wurden notiert. Folgende Stichpunkte standen am Ende der Ideensammlung an der Tafel:

- Dollar-Button einfügen
- Methode definieren
- Array für Familienmitglieder und Geldwerte
- for-Schleifen
- System.out.println
- Mausklick Abfrage
- if-Abfragen

Im Folgenden wurde damit begonnen, vorhandenes Vorwissen zu Schleifen, Verzweigungen und dem Konsolenausgabe-Befehl zu aktivieren und durch gezielte Fragestellungen zu erweitern. Dies stellte auch den Abschluss der Doppelstunde dar.

#### 4.4.5 Fünfte Doppelstunde

Die ersten 20 Minuten der fünften Doppelstunde wurden darauf verwendet, die Ergebnisse der letzten Stunde rekapitulieren zu lassen und die theoretische Vorarbeit zu beenden.

Danach teilte ich das Arbeitsblatt mit der konkreten Aufgabenstellung aus und die SuS traten ihre nächste Gruppenarbeit an, in der sie sich wieder der Programmierung widmeten. Diese umfangreiche Aufgabe stellte zugleich die längste Schülerarbeitsphase dar. Die Gruppen arbeiteten den Rest der ersten und die gesamte zweite Unterrichtsstunde daran. Ich trat lediglich unterstützend in Erscheinung, beantwortete Fragen und behielt ihr Vorankommen im Auge. Zu Beginn der zweiten Stunde hatten fast alle Gruppen den notwendigen neuen *Actor* (das Dollarzeichen) erstellt und eine Methode implementiert, die durch Mausklick auf die entsprechende Schaltfläche aufgerufen wurde.

#### 4.4.6 Sechste Doppelstunde

Die letzte Doppelstunde der Unterrichtsreihe war in zwei Abschnitte unterteilt. In den ersten 45 Minuten bekamen die Gruppen noch einmal Zeit, an ihren Aufgaben zu arbeiten. Nur eine Gruppe schaffte es, alle drei Aufgaben vollständig zu bearbeiten. Die meisten Teams erreichten den Abschluss von Aufgabe Eins, welche allerdings auch den größten Arbeitsaufwand darstellte. Alle Ergebnisse befinden sich in Anhang O. Dennoch wurde die Programmierung nach der Stunde für beendet erklärt, da die letzte Unterrichtsstunde im Zeichen des Abschlusses der Reihe stehen sollte.

Sie begann mit einem *Blitzlicht*, um ein kurzes und spontanes Feedback von allen SuS zu erhalten. Das *Blitzlicht* ist eine Methode, bei der jede Schülerin und jeder Schüler reihum zu Wort kommt und einen Gedanken zu einem Thema äußert. In diesem Falle sollten sie einen positiven oder einen negativen Aspekt zu der durchgeführten Unterrichtsreihe nennen. Ebenfalls war es möglich, sowohl einen positiven als auch einen negativen Gedanken zu

artikulieren. Die Ergebnisse dieses *Blitzlichts* sind unter Punkt 4.6.2 dargestellt.

Während der letzten 30 Minuten nannte ich zunächst meine eigenen Eindrücke zu den vergangenen Wochen und der Arbeit mit den SuS. Danach wurde der zweite Erhebungsbogen (s. Anhang B) ausgeteilt und von allen SuS bearbeitet.

## **4.5 Beobachtungen und Besonderheiten**

In diesem Kapitel sollen nun systematisch die Erfahrungen und Beobachtungen, die ich in den einzelnen Doppelstunden machen konnte, beschrieben werden.

### **4.5.1 Erste Doppelstunde**

Die Einführung des UML Aktivitätsdiagramms stellte die SuS vor keine großen Herausforderungen. Bei der Sammlung der Elemente gab es eine rege Schülerbeteiligung. Auch bei den ersten eigenen Realwelt-Beispielen zeigten sie erfreulichen Ideenreichtum. Einige Gruppen produzierten dabei jedoch recht komplizierte Modelle, die semantisch und syntaktisch nicht immer fehlerfrei waren. Teilweise lag dies jedoch auch an dem Freeware-Programm DIA selbst, was in seiner Bedienung als unkomfortabel zu bezeichnen ist. Ich kann hierfür keine Empfehlung geben. Das Programm war jedoch für die Modellierung vorgeschrieben, da es auch im Abitur verwendet wird. Deshalb gab es hier keine Ausweichmöglichkeit.

Die Besprechung der Ergebnisse im Plenum funktionierte gut. Positive Aspekte wurden genannt und fast alle Fehler oder Unklarheiten von den Mitschülern herausgearbeitet.

### **4.5.2 Zweite Doppelstunde**

In der zweiten Doppelstunde beschäftigten sich die SuS zum ersten Mal selbstständig mit dem Spiel. Die freie Erprobungsphase, in der die Gruppen *Farmer's Valley* spielen durften, weckte sehr schnell den Wettbewerbseifer, bei dem es allen Gruppen darum ging, als erste den begehrten Mähdrescher kaufen zu können und der erfolgreichste Bauer zu sein. Dies stellte zugleich

den ersten echten Härtestest für das Server-Client-System dar, bei dem es durch die hohe Nutzeraktivität voll ausgelastet wurde. Es zeigte sich, dass sowohl der Server als auch die Clients sehr stabil liefen und keine Abstürze oder Fehler zu beklagen waren. Einziger Wermutstropfen war die Beschränkung der Leistungsfähigkeit, die durch die technische Infrastruktur des Klassenraums entstand (siehe Kapitel 4.3). Die starke Verlangsamung des Spiels, wenn alle Bauern gleichzeitig aktiv waren und sich mehrere Felder in der Regenerationsphase befanden, führte bei den SuS zu Ungeduld. Diese Verzögerungen lösten sich jedoch nach erfolgter Feld-Regeneration stets von selbst auf und zogen keine weiteren negativen Konsequenzen nach sich.

In der nachfolgenden Aufgabe, in welcher der act()-Zyklus modelliert werden sollte, zeigte sich die hohe intrinsische Motivation der SuS. Alle Gruppen arbeiteten akribisch an ihren Modellen und ignorierten zum Großteil das Klingeln zur 5-Minuten-Pause. Stattdessen wurde diese – bis auf wenige Ausnahmen – zum Weiterarbeiten genutzt, ohne dass dies von mir verlangt worden wäre oder ich eine knappe Zeitgrenze gesetzt hätte.

Die Gruppe, die das Modell-Skelett von mir als Hilfestellung gereicht bekam, weigerte sich zunächst dies anzunehmen. Mit der Begründung, dass sie das Ziel so wie die anderen ebenfalls eigenständig erreichen wollten, lehnten sie die Nutzung ab. Erst nachdem ich ihnen erklärt hatte, dass die Hilfe keine Schmälerung der Eigenleistung bedeuten würde, akzeptierten sie die veränderte Aufgabenstellung. Schließlich musste das Modell eigenständig in DIA nachgebaut werden, da es lediglich als Bilddatei vorlag. Außerdem musste ein für sie fremdes Modell verstanden werden, da es sonst nicht korrekt hätte ausgefüllt werden können. Es zeigte sich im Laufe der Stunde, dass dies tatsächlich eine Herausforderung darstellte, die die Gruppe beschäftigte und sie somit ihrem Leistungsniveau gerecht arbeiten konnte.

Ursprünglich war von mir geplant gewesen, den bisherigen Verlauf innerhalb einer Unterrichtsstunde zu absolvieren. Es stellte sich jedoch hier bereits heraus, dass ich für alle Aufgaben deutlich mehr Zeit hätte einplanen müssen als gedacht. Der Teil, der in der zweiten Unterrichtsstunde dieses Tages behandelt werden sollte, stellte genügend Stoff für die komplette nächste Doppelstunde dar.

### 4.5.3 Dritte Doppelstunde

Die Ideen, die von den SuS vorgebracht wurden um die Regenerationsgeschwindigkeit der Felder von einer Wetter-Variablen abhängig zu machen, waren vielfältig. Schnell kristallisierten sich zwei Möglichkeiten zum Festlegen des aktuellen Wetters heraus. Beim ersten Modell würde die entsprechende Variable nach einer bestimmten Anzahl von Durchläufen auf den jeweils anderen Wert gesetzt. Es würde somit stets eine gleich große Anzahl von Sonnen- und Regentagen geben, welche sich regelmäßig abwechselten. Das andere Modell sah vor, dass eine Zufallszahl ermittelt werden sollte, die entweder die Ausprägung  $0$  oder  $1$  annehmen konnte und zwischen Sonne und Regen entscheiden würde. Schnell stellte sich heraus, dass SuS einen hohen Realitätsanspruch haben, denn die meisten Gruppen entschieden sich für das zweite Modell, obwohl es in seiner Umsetzung ein wenig schwieriger war. Kurz nach Beginn der Gruppenarbeitsphase verschaffte sich erstaunlicherweise ein Schüler eigenständig die Aufmerksamkeit der Klasse und schrieb für alle den Java-Befehl zum Erstellen einer solchen Zufallszahl mit Erklärungen an die Tafel.

Es war zu beobachten, dass sich die Gruppen bei der Programmierung stark an dem orientierten, was an die Tafel geschrieben worden war. Es ist also wichtig zu beachten, dass dort von der Lehrkraft nur Stichpunkte und Möglichkeiten vorgegeben werden, jedoch keine zu konkreten Hilfestellungen. Dies würde die selbständige Konstruktion des Programmcodes behindern und die Lösungsvarianten einschränken.

Es war gut, dass zu diesem Zeitpunkt programmiert wurde, da die Motivation zum Modellieren zum Ende der letzten Aufgabe hin bereits leicht gesunken war. Ich stellte fest, dass die SuS nur schwer verstehen, warum – gerade bei solch vergleichsweise kleinen Aufgaben – vor dem Programmieren modelliert werden muss. Die Bedeutung dessen würde erst bei einem größeren Projekt deutlich. Stattdessen möchten sie lieber direkt mit der Programmierung beginnen.

Die Gruppen freuten sich über die sichtbaren Veränderungen, die sie im Spiel vollzogen hatten. Dies zeigte sich nicht nur in ihrem Verhalten untereinander, sondern auch darin, dass sie sich zwischendurch meldeten, um mir ihr Erreichtes vorzustellen. Mir fiel dabei erst spät auf, dass solche Veränderungen jeweils nur für einige andere Gruppen in derselben Welt sichtbar waren, nicht

jedoch für alle. Dies lag daran, dass die der Klasse *Feld* hinzugefügte Wettervariable verschieden genannt wurde – meist (boolean) *wetter* oder (boolean) *sonne*. Somit wurde das aktuelle Wetter eines Felds nur von den anderen Gruppen aufgefangen, die denselben Variablennamen gewählt hatten. Darauf hätte ich im Vorhinein achten und einen für alle verbindlichen Namen festlegen müssen.

Wieder reichte die veranschlagte Zeit bei den Gruppen nicht aus, um die Aufgabe vollständig zu lösen. Die meisten waren jedoch schon weit fortgeschritten.

#### 4.5.4 Vierte Doppelstunde

Die vierte Doppelstunde begann mit der Beendigung der Programmieraufgabe aus der vorherigen Stunde. Dabei stand etwas mehr Zeit zur Verfügung, als die schnellsten Gruppen zur Fertigstellung brauchten. Diese beschäftigten sich daraufhin freiwillig weiter mit *Farmer's Valley* und programmiertechnischen Veränderungen. So veränderte eine Gruppe beispielsweise das Aussehen der Statusleiste und eine andere wandelte die Klasse *Bauer* dahingehend ab, dass viel Geld zur Verfügung stand und schneller geerntet werden konnte. Dabei kam es jedoch zum Absturz des Servers. Die Ursache hierfür konnte nicht eindeutig geklärt werden, da das Szenario von der entsprechenden Gruppe sofort per eigener Sicherungskopie wieder auf den ursprünglichen Stand gebracht worden war. Hier zeigt sich eine Schwierigkeit bei der Arbeit mit *Farmer's Valley*: Es muss darauf geachtet werden, dass bestimmte Teile des Quellcodes unangetastet bleiben, da der gesamte Klassenraum vom zentralen Server abhängig ist. Durch den Absturz musste der Server neugestartet werden und alle Gruppen loggten sich neu ein. Dadurch gingen einige Minuten sowie das bisherige Arbeitsergebnis der verursachenden Gruppe verloren.

Das Vorstellen der Ergebnisse durch gruppenfremde Klassenmitglieder stellte sich als gut funktionierende Methode heraus. Das Verstehen von fremdem Quelltext ist schwierig und das Nachvollziehen von Erklärungen, die von der erstellenden Gruppe selbst präsentiert werden, ist oft kaum möglich und für SuS wenig motivierend. So jedoch beschäftigten sich alle mit dem Quellcode und waren – begünstigt auch dadurch, dass sie sich mit derselben Thematik befasst hatten – in der Lage, den Lösungsansatz und die Funktionsweise zu

erfassen und zu artikulieren. Es wurden zwei verschiedene Ergebnisse vorgestellt, die deutlich machten, dass man auf verschiedene Art und Weise zu funktionierenden Lösungen kommen konnte. Auch die jeweiligen Vor- und Nachteile der Ergebnisse wurden von den SuS kritisch beleuchtet.

Die zweite Stunde begann mit der Sammlung von Ideen zur Umsetzung der Methode, die per Mausklick das verfügbare Vermögen aufteilen und ausgeben sollte. Die Ideen der SuS waren dabei gut. Sie ermittelten alle nötigen Elemente. Es zeigte sich jedoch, was bereits durch den ersten Erhebungsbogen zu vermuten war: das Fachwissen zu wichtigen Java-Konstrukten war bei vielen SuS sehr eingeschränkt. So musste ich den Rest der Stunde und den Beginn der nächsten Stunde für die Vermittlung der theoretischen Grundlagen verwenden. Für die wenigen guten Programmierer unter den SuS war dies sicherlich ein wenig langwierig. Es galt jedoch, alle SuS auf die nächste selbständige Gruppenarbeitsphase vorzubereiten.

#### **4.5.5 Fünfte Doppelstunde**

Nach Abschluss der theoretischen Vorbesprechung gingen die Gruppen an ihre Arbeit. Die Aufgabe stellte für viele SuS eine große Herausforderung dar, da der Schwierigkeitsgrad etwas höher lag. Es mussten Veränderungen an verschiedenen Stellen im Spiel vorgenommen werden, und die eigene Programmier-Leistung war höher als in der letzten Aufgabe. Dies mündete in vermehrter Kollaboration zwischen den einzelnen Gruppen. Sie unterstützten sich gegenseitig und Gruppenmitglieder wechselten immer wieder die Arbeitsplätze um schwächeren SuS zu helfen, ohne dass ich dies als wünschenswert geäußert hatte. Für mich bedeutete dieser Arbeitsauftrag ebenfalls eine höhere beratende Tätigkeit, bei der ich jedoch lediglich Tipps zu geben versuchte und keine konkreten Lösungsvorschläge machte.

Eine Gruppe nutzte das neu erworbene Wissen, um ebenfalls per Mausklick das Bild der eigenen Spielfigur zu ändern.

#### **4.5.6 Sechste Doppelstunde**

Nachdem mit dem Ende der ersten 45 Minuten auch das Ende der programmierenden Tätigkeit erreicht war, arbeiteten einige der Gruppen, die die Aufgaben noch nicht vollständig bearbeitet hatten, zum wiederholten Male die

5-Minuten-Pause durch. Es bestand bei den meisten SuS eine hohe Motivation, das Arbeitsblatt weiter zu bearbeiten. Da jedoch die letzte Unterrichtsstunde der Reihe anstand, mussten auch diese Gruppen zu Beginn der zweiten Stunde ihre Arbeit einstellen.

Die letzte Stunde diente dem gegenseitigen Feedback. Die Ergebnisse des *Blitzlichts* und des zweiten Erhebungsbogens sowie der Vergleich zum ersten Erhebungsbogen werden im nächsten Kapitel ausführlich behandelt.

## 4.6 Evaluation der Unterrichtsreihe

In diesem Abschnitt nun sollen die Beobachtungen und Ergebnisse der Unterrichtsreihe umfassend bewertet werden. Hierzu werde ich zunächst eine persönliche Einschätzung der Studie und ihres Verlaufs vornehmen. Danach werden durch die Auswertung des *Blitzlichts* und des zweiten Erhebungsbogens sowohl das Feedback der SuS als auch deren tatsächlich beobachtbarer Lernfortschritt beleuchtet.

### 4.6.1 Persönliche Bewertung

Die Vorbereitung der Unterrichtsreihe stellte eine Herausforderung dar. Die Anpassung der verwendeten Software an die gegebene technische Ausstattung der Schule und das Einrichten des Servers und der Client-Computer stellten sich schnell als aufwändiges Unterfangen heraus. Allerdings ist diese Arbeit nur einmal im Voraus zu leisten. Während der Wochen des Unterrichts gab es keinerlei technische Ausfälle oder Schwierigkeiten zu beklagen. Das System lief stabil und zuverlässig, solange keine Anwendungsfehler von Seiten der Nutzer gemacht wurden.

Im Verlauf der sechs Wochen Unterrichtspraxis empfand ich die Motivation und den Eifer, mit denen die SuS agierten, als äußerst zufriedenstellend. Alle Aufgaben wurden stets gewissenhaft bearbeitet und Unterrichtsstörungen beschränkten sich auf ein Minimum. Dies lässt sich meiner Meinung nach maßgeblich dadurch begründen, dass ein Spiel mit Multiplayer-Aspekt im vernetzten Klassenraum als Unterrichtsbasis diente. Bereits während der Präsentation von *Farmer's Valley* und den Plänen, die ich für die folgenden sechs Wochen hatte, wurde mir von den SuS höchste Aufmerksamkeit entgegengebracht. Während der Phase, in der das Spiel zum ersten Mal selbst

ausprobiert werden durfte, bestätigte sich dieser Eindruck. Hier zeigte sich auch, dass ein verhältnismäßig simples 2D-Spiel wie *Farmer's Valley* auch von einem Oberstufenkurs der 13. Klasse nicht als langweilig empfunden wird. Die Möglichkeit, miteinander in den Wettbewerb zu treten, stellte sich als große Bereicherung heraus.

Die Konzentration auf das kollaborative Arbeiten in Gruppen bewerte ich als sehr positiv. Die SuS profitierten gegenseitig voneinander und halfen sich auch gruppenübergreifend. Besonders bei Aufgaben, die als anspruchsvoll angesehen wurden, fand ein reger Austauschprozess im gesamten Klassenraum statt. Dies lag sicherlich teilweise darin begründet, dass sich die SuS in einer gemeinsamen virtuellen Welt befanden. Auch Schüler, die eine Stunde verpassten, fanden durch die Eingliederung in bestehende Gruppen schnell Anschluss. Des Weiteren ermöglichte mir die Gruppenarbeit, den Teams individuelle Hilfestellungen und einen ihrer Leistung angepassten Grad an Unterstützung zukommen zu lassen. Dies bedeutete eine Binnendifferenzierung, wie sie mit anderen Unterrichtsformen nur schwer umzusetzen gewesen wäre. Insgesamt zeigten alle SuS einen hohen Grad an Kreativität und Engagement im Bearbeiten ihrer Aufgaben.

Der Vor- und Nachbereitungsaufwand der einzelnen Doppelstunden ist in einem schülerzentrierten und handlungsorientierten Unterricht größer als im klassischen Frontalunterricht. Dies bestätigte sich auch in dieser Reihe. Die von den SuS produzierten Ergebnisse fallen alle unterschiedlich aus und müssen von der Lehrkraft begutachtet werden, um differenziertes Feedback geben zu können und Stillstand zu vermeiden. Ein großer Vorteil dieses Konzepts zeigt sich aber dann im Unterricht selbst. Dieser ist als entspannt zu bezeichnen und eröffnet die Möglichkeit, auf alle Gruppen einzeln eingehen zu können und zum größten Teil lediglich im Hintergrund tätig sein zu müssen. So lässt sich auch die Art und Weise der Zusammenarbeit in und zwischen den Gruppen beobachten und bewerten.

Als schwierig empfand ich während meiner Arbeit das Zeitmanagement. Abschließend kann ich sagen, dass für alle Aufgaben, die von mir geplant worden waren, etwa das Doppelte der ursprünglich veranschlagten Zeit benötigt wurde. Gerade in einem praktisch angelegten Informatikunterricht passiert es leicht, den Aufwand der gestellten Arbeitsaufträge für die SuS zu unterschätzen. Begünstigt wird dies durch das Kommunizieren und Interagie-

ren der SuS untereinander, welches zwar zahlreiche Vorteile mit sich bringt, aber auch mehr Zeit in Anspruch nimmt.

#### 4.6.2 Blitzlicht und zweiter Erhebungsbogen

Die Methode des *Blitzlichts* wurde bereits in Kapitel 4.4.6 beschrieben. Die Ergebnisse spiegeln eine spontane Reflexion der SuS ohne konkrete Fragestellung wider. Hier folgt nun die Liste aller genannten Punkte. Mehrfachnennungen werden dabei nicht berücksichtigt. Rückschlüsse auf die Wertung einzelner Punkte sind also nicht möglich:

- |                               |  |
|-------------------------------|--|
| ⊕ Gruppenarbeit               | ⊖ Tempo zu langsam   |
| ⊕ Man sieht was man tut       | ⊖ Zu sehr ins Detail   |
| ⊕ Relation Theorie und Praxis | ⊖ Greenfoot nicht interessant                                  |
| ⊕ Mehr verstanden als sonst   | ⊖ Letzte Programmieraufgabe mit nur peripherem Bezug zum Spiel |
| ⊕ Java-Kenntnisse verbessert  |  |
| ⊕ Hat Spaß gemacht            |  |
| ⊕ Abwechslungsreich           |  |
| ⊕ Kein „typischer“ Unterricht |  |
| ⊕ Gut durchgeplant            |  |
| ⊕ Gute Berufseignung          |  |

Es lässt sich feststellen, dass viele der positiven Aspekte, die durch das kollaborative Arbeiten mit Hilfe einer vernetzten grafischen Simulation erreicht werden sollten, tatsächlich von den SuS als solche wahrgenommen und genannt wurden. Dabei stellte sich auch heraus, dass ein solches Konzept noch keinen Einzug in den alltäglichen Unterricht gehalten hat und infolgedessen von den SuS als etwas Besonderes angesehen wurde. Die Beschäftigung mit etwas Neuem wirkte in besonderem Maße motivierend.

Einige der leistungsstärkeren SuS hätten sich an einigen Stellen einen schnelleren Unterrichtsablauf gewünscht. Als Lehrkraft gilt es, den Schwierigkeitsgrad des Unterrichts an das durchschnittliche Leistungsniveau der Lerngruppe anzupassen. Es ist nur schwer vermeidbar, dass leistungsschwächere SuS an manchen Stellen überfordert und stärkere SuS zu wenig gefordert sind. Dieser

Effekt konnte durch die Konzentration auf Gruppenarbeit größtenteils vermieden werden. Die Kritik bezog sich hingegen auf die Phasen des Unterrichtsgesprächs im Plenum, in denen Grundlagenwissen aktiviert und verbessert wurde. Hier könnte man leistungsstärkere SuS ausnehmen und mit anderen Aufgaben beschäftigen. Dafür ist es allerdings wichtig, die Lerngruppe sehr genau zu kennen und das Niveau aller SuS genau einschätzen zu können. Nur ein Schüler empfand die Arbeit mit Greenfoot als grundsätzlich nicht interessant.

Im letzten Negativpunkt zeigt sich eine Schwierigkeit in der Programmierung mit *Farmer's Valley*. Durch die Komplexität des Quellcodes ist es nicht leicht, Veränderungen an bereits vorhandenen Klassen durchzuführen. Die Auswirkungen sind für die SuS oft unberechenbar und nur schwer nachvollziehbar. Zudem müssten hierfür große Teile des Quellcodes und die genaue Funktionsweise bis ins Detail erarbeitet werden, was den Zeitrahmen einer solchen Unterrichtsreihe sprengen und für einen großen Teil der Lerngruppe eine zu große Herausforderung darstellen würde. Deshalb habe ich eine Aufgabe konstruiert, in der wichtige Konzepte von Java, wie Arrays und Kontrollstrukturen, gelernt werden konnten, die aber größtenteils ohne Kenntnisse vom restlichen Quelltext zu bearbeiten war. Um den SuS einen besseren Überblick und größere Freiheit in der Programmierung zu ermöglichen, könnte man den Unterricht mit einem einfacheren (aber dennoch mehrspielerbasierten) Spiel gestalten, welches nur wenig Quelltext mit sich bringt.

Durch den zweiten Erhebungsbogen, der von den SuS am Ende der Unterrichtsreihe ausgefüllt wurde, bekamen sie die Möglichkeit, differenziertes Feedback zu den vergangenen sechs Wochen zu geben. Der erste Teil des Bogens bestand hauptsächlich aus Fragen, die durch Ankreuzen einer Skala von eins bis sechs beantwortet werden sollten, wobei eins für „sehr gut“ bzw. „sehr viel“ und sechs für „sehr schlecht“ bzw. „sehr wenig“ stand. Im zweiten Teil wurden sprachlich formulierte Antworten zu verschiedenen Fragestellungen verlangt, bevor im abschließenden Teil dieselben Fachfragen und Programmieraufgaben wie im ersten Erhebungsbogen zur Beantwortung standen. Alle ausgefüllten Erhebungsbögen sind in Anhang D zu finden.

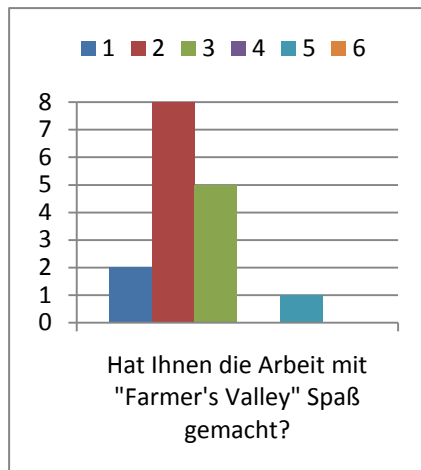


Abbildung 18: Bewertung der Reihe

Durch die Beantwortung der Fragen zeigte sich, dass die SuS die Unterrichtsreihe insgesamt als sehr positiv empfunden haben (s. Abb. 18). Bei der Frage nach den Elementen, die besonders gut gefielen, wurde am häufigsten die von der Lehrkraft unterstützte Teamarbeit im vernetzten Klassenraum genannt (insgesamt sieben). An zweiter Stelle stand die Art und Weise der Programmierung, bei der durch die Entwicklungsumgebung immer ein direktes Feedback gegeben wurde und Änderungen

sofort im Spiel sichtbar wurden (sechs). Dies machte das Arbeiten laut Aussagen der SuS interessanter und weniger eintönig als bereits erlebte Unterrichtssequenzen. Das Element des Spiels wurde ebenfalls mehrmals positiv erwähnt (fünf). Mit jeweils vier Kommentaren wurden die Beschäftigung mit Java und das praktisch angelegte und gut mit der Theorie verzahnte Unterrichtskonzept genannt.

Bei der konkreten Frage nach der Bewertung des Mehrspieler-Aspekts wird deutlich, dass dieser große Anerkennung erhielt. Insgesamt bewerteten ihn 13 SuS als positiv und empfanden den Unterricht dadurch als lebendig und interessant. Drei kritische Bemerkungen bezogen sich auf die streckenweise geringe Performanz des Servers und zwei SuS bemängelten die Fehleranfälligkeit des Systems bei falschem Nutzerverhalten. So führten die Veränderungen im Quelltext einer Gruppe an einer Stelle zum Absturz des Servers.

Die Gruppenarbeit wurde insbesondere aus den Gründen geschätzt, dass sie den SuS die Möglichkeit gab, voneinander zu lernen und sich gegenseitig zu unterstützen (zwölf). Drei Stimmen plädierten für mehr Einzelarbeit, um in größeren Anteilen eigenständig und ohne Hilfe programmieren zu können.

Der Schwierigkeitsgrad der Unterrichtsreihe sowie die Komplexität des vorhandenen Quellcodes wurden von den SuS als durchschnittlich angesehen. Die meisten gaben an, nur wenige Verständnisprobleme in Bezug auf die grundsätzliche Struktur und Arbeitsweise von *Farmer's Valley* gehabt zu haben. Dies liegt unter anderem daran, dass nur kleine Codesegmente für die Bearbeitung der Aufgabenstellungen von Bedeutung waren und kein globales

Verständnis des Spiels nötig war. Betrachtet man jedoch den Zeitaufwand zur Lösung der gestellten Aufgaben, wäre eine umfassendere Beschäftigung mit *Farmer's Valley* im Rahmen der sechs Doppelstunden nicht möglich gewesen. Dennoch empfanden fünf SuS die Einarbeitung in den Quelltext als herausfordernd und viermal wurde von Problemen im Verständnis der Syntax berichtet.

Zur Frage nach den Negativaspekten der Unterrichtsreihe gab es nur zwei Antworten. Eine Stimme empfand das Starten des Client-Server-Systems als umständlich und eine zweite Stimme bewertete den gesetzten Schwerpunkt der Programmierung negativ.

Sechs SuS hätten sich mehr Zeit und Möglichkeiten zu weiteren Veränderungen des Spiels gewünscht. Zwei Bemerkungen bezogen sich auf das Fehlen von Musterlösungen, welche es jedoch im Rahmen dieses Unterrichtskonzeptes nicht geben konnte und sollte. Stattdessen wurden den Gruppen stets individuelle Rückmeldungen gegeben, die auf gute sowie problematische Aspekte ihrer Lösungen hinwiesen.

Die meisten SuS waren der Meinung, durch den Unterricht ihre Modellierungs- und Programmierkenntnisse verbessert bis stark verbessert zu haben (s. Abb. 19 a und 19 b).

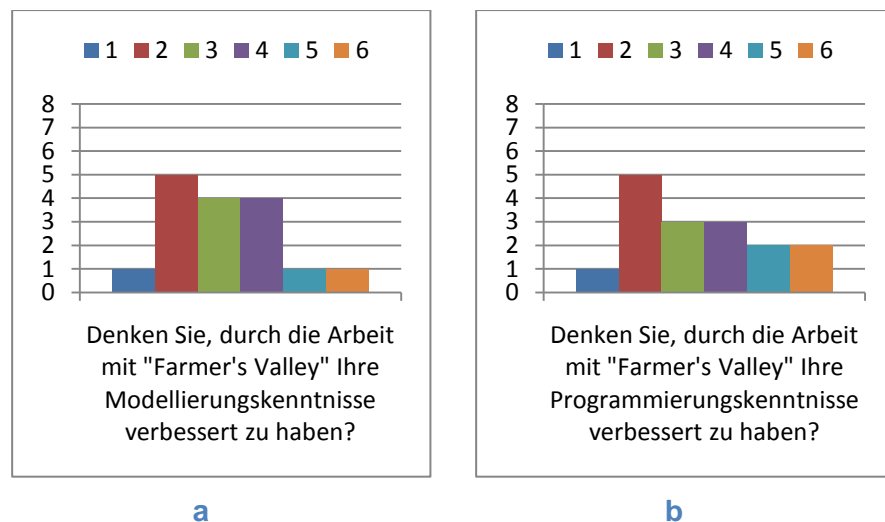


Abbildung 19: Bewertung des Lernzuwachses

Durch den letzten Aufgabenteil des zweiten Erhebungsbogens wurde konkret getestet, wieviel die SuS von den behandelten Inhalten in einem anderen Kontext und ohne die Hilfe einer Entwicklungsumgebung anwenden konnten.

Im Vergleich zu den Ergebnissen des ersten Erhebungsbogens zeigte sich hier eine deutliche Verbesserung der Antworten. Sieben SuS erreichten ein sehr gutes Ergebnis mit nur wenigen, kleinen Fehlern. Weitere sechs Lösungen waren durchschnittlich gut. Nur drei Teilnehmer gaben mangelhafte Antworten auf die Fragestellungen. Es gab jedoch niemanden mehr, der die Beantwortung der Aufgaben völlig frei ließ. Anzumerken ist, dass manche SuS beim Ausfüllen des Erhebungsbogens zeitlich in Bedrängnis gerieten, obwohl ihnen beinahe eine komplette Unterrichtsstunde zur Verfügung stand. Dies lag jedoch erneut daran, dass Programmieraufgaben teilweise nur langsam gelöst wurden. Es ist also davon auszugehen, dass das Ergebnis bei mehr zur Verfügung stehender Zeit noch positiver ausgefallen wäre.

## 5 Fazit

Die in dieser Arbeit beschriebene Studie hat Möglichkeiten für den Einsatz vernetzter grafischer Simulationen im schulischen Informatikunterricht getestet und bewertet. Dazu wurde zunächst die der Studie zugrundeliegende Software entworfen und vorgestellt. Die anschließend betrachteten theoretischen Grundlagen aus der Pädagogik dienten als Basis für den Entwurf einer Unterrichtsreihe, welche in einem gymnasialen Leistungskurs eines Abiturjahrgangs im Fach Informatik durchgeführt wurde. Die gesammelten Eindrücke und Ergebnisse, die sowohl durch den Unterrichtsverlauf als auch durch die Schülerarbeiten und die beiden Erhebungsbögen gewonnen werden konnten, dienten der abschließenden Evaluation.

Im Rahmen der Durchführung und Auswertung der beschriebenen Unterrichtsreihe hat sich gezeigt, dass die positiven Erwartungen, die sich aus der Betrachtung der theoretischen Grundlagen ergeben hatten, bestätigt werden konnten. Es wurde deutlich, dass eine vernetzte grafische Simulation, wie sie als Basis der hier vorgestellten Unterrichtsreihe diente, äußerst motivierenden Charakter hat und zudem die Möglichkeit bietet, Informatikunterricht anschaulich und verständlich zu gestalten. Die Vernetzung des Klassenraums und die daraus resultierenden gesteigerten Möglichkeiten der SuS, auf verschiedenen Ebenen kollaborativ miteinander zu arbeiten, führten zu einem positiven Lernerlebnis sowohl in fachlicher als auch in emotionaler Hinsicht. Diese Ergebnisse rechtfertigen den gesteigerten Aufwand in der Vorbereitung und Durchführung einer solchen Unterrichtsreihe.

Bei der Durchführung der Unterrichtsreihe wurde deutlich, dass große Teile des Programmcodes nicht thematisiert werden konnten. Dies lag zum einen an fehlenden Programmierkenntnissen der SuS, zum anderen an mangelnder Unterrichtszeit. Außerdem gestaltete sich die Fehlersuche schwierig, wenn an mehreren Stellen im Szenario Änderungen von SuS vorgenommen wurden. Deshalb wäre es interessant, eine ähnliche Unterrichtsreihe mit einem weniger komplexen mehrspielerbasierten Szenario durchzuführen. Die SuS hätten so noch größere Möglichkeiten zu eigenständigem Experimentieren, ohne dass sie schwer zu entdeckende Fehler verursachen oder die Stabilität des Server-Client-Systems beeinflussen würden. Die Aufgabenvielfalt könnte vergrößert werden, um so eine noch stärkere Differenzierung zwischen leis-

tungsstärkeren und leistungsschwächeren SuS zu erreichen. Bei größerem Zeiteinsatz bestünde zudem die Möglichkeit, auf die verwendeten Netzwerktechniken einzugehen und die Server-Client-Kommunikation zum Unterrichtsgegenstand zu machen.

# Anhang

## Anhang A: Erhebungsbogen 1

Liebe Schülerinnen, liebe Schüler,

im Folgenden finden Sie einige Fragen, die mir dabei helfen sollen, Erfolge und Misserfolge der sich anschließenden Unterrichtsreihe besser abschätzen zu können. In der letzten Unterrichtsstunde werden Sie noch einmal einen Erhebungsbogen bekommen auf dem Sie die Gelegenheit bekommen werden auch Ihre persönlichen Eindrücke und Meinungen zu schildern. Alle Angaben werden dabei stets anonym bleiben.

Einige Fragen sind mit einem einfachen Kreuz zu beantworten. Eine ① steht dabei für „sehr gut“ oder „sehr viel“, eine ⑥ dementsprechend für „sehr schlecht“ oder „sehr wenig“.

Vielen Dank!

Wie gut schätzen Sie Ihre Modellierungskenntnisse ein?	① ② ③ ④ ⑤ ⑥
Wieviel Spaß macht Ihnen das Modellieren?	① ② ③ ④ ⑤ ⑥
Wie gut schätzen Sie Ihre eigenen Kenntnisse in UML ein?	① ② ③ ④ ⑤ ⑥
Wie gut schätzen Sie Ihre Programmierkenntnisse in Java ein?	① ② ③ ④ ⑤ ⑥
Wieviel Spaß macht Ihnen das Programmieren in Java?	① ② ③ ④ ⑤ ⑥
Programmieren Sie auch privat?	① ② ③ ④ ⑤ ⑥
Falls ja, in welchen Programmiersprachen?	

Welche Modellierungstechniken kennen Sie (z.B. endliche Automaten etc.)?

Was ist UML, wofür steht die Abkürzung und wofür kann es genutzt werden?

Welche Kontrollstrukturen kennen Sie in Java (umgangssprachlicher Ausdruck und Java-Schreibweise)?

Hier sehen Sie das Gerüst für die Klasse „Schüler“ in der Programmiersprache Java. Bitte ergänzen Sie den fehlenden Programmcode so gut Sie können. Falls Sie einen Befehl nicht genau kennen nutzen Sie bitte Pseudo-Code.

Die Klasse soll folgende Daten beinhalten:

- Name
- Alter
- Geschlecht (gespeichert als Zahl: 1 steht für männlich, 2 für weiblich)
- Abiturfächer (gespeichert in einem Array der Größe 4)
- Gruppe (Variable zur Speicherung einer Zahl, die eine Gruppenzugehörigkeit angibt)

```
public class Schueler {
```

Was ist der Name für den folgenden Codeblock?

Antwort:

```
public Schueler(String dername, int dasalter, int
dasgeschlecht, String[] dieabiturfaecher) {

    name = dername;
    alter = dasalter;
    geschlecht = dasgeschlecht;
    abiturfaecher = dieabiturfaecher;
}
```

Mit der Methode „gebeAbiturfaecherAus()“ sollen nacheinander alle vier gewählten Abiturfächer ausgegeben werden können. Nutzen Sie hierfür bitte eine for-Schleife.

```
public void gebeAbiturfaecherAus() {

}

}
```

Die Methode „gruppeneinordnung()“ soll dazu dienen, der oben von Ihnen definierten Variable „gruppe“ eine Zahl zuzuordnen. Dabei soll „gruppe“ eine „1“ zugeordnet werden, falls das erste Abiturfach „Informatik“ ist. Eine „2“ soll zugeordnet werden, falls das erste Abiturfach „Mathematik“ ist. Ist das erste Abiturfach ein anderes Fach, so sollen alle vier gespeicherten Abiturfächer gelöscht werden.

```
public void gruppeneinordnung() {

}

}
```



## Anhang B: Erhebungsbogen 2

Liebe Schülerinnen, liebe Schüler,

im Folgenden finden Sie einige abschließende Fragen, die mir dabei helfen sollen, Erfolge und Misserfolge der hiermit abgeschlossenen Unterrichtsreihe abschätzen zu können. Sie haben hier nun auch die Gelegenheit Ihre persönlichen Eindrücke und Meinungen zu schildern. Alle Angaben werden dabei stets anonym bleiben.

Einige Fragen sind mit einem einfachen Kreuz zu beantworten. Eine ① steht dabei für „sehr gut“ oder „sehr viel“, eine ⑥ dementsprechend für „sehr schlecht“ oder „sehr wenig“.

Vielen Dank!

Hat Ihnen die Arbeit mit „Farmer’s Valley“ Spaß gemacht?	①   ②   ③   ④   ⑤   ⑥
Was fanden Sie am Spiel und/oder an der Unterrichtsreihe besonders gut?	
Sehen Sie Unterschiede zu Unterrichtsreihen aus der Vergangenheit? Wenn ja, welche?	
Wie schätzen Sie insgesamt den Schwierigkeitsgrad der zurückliegenden Unterrichtsreihe ein?	①   ②   ③   ④   ⑤   ⑥
Haben Sie das Gefühl, durch die Arbeit mit „Farmer’s Valley“ Ihre Modellierungskenntnisse verbessert zu haben?	①   ②   ③   ④   ⑤   ⑥

Haben Sie das Gefühl, durch die Arbeit mit „Farmer’s Valley“ Ihre Programmierkenntnisse verbessert zu haben?	① ② ③ ④ ⑤ ⑥
War „Farmer’s Valley“ ein für Sie komplexes Programm (komplexer Programmcode)?	① ② ③ ④ ⑤ ⑥
Haben Sie im Unterricht bereits mit komplexem Programmcode gearbeitet?	① ② ③ ④ ⑤ ⑥
Hatten Sie Probleme, die grundsätzliche Struktur und Arbeitsweise von „Farmer’s Valley zu verstehen?“	① ② ③ ④ ⑤ ⑥
Hatten Sie Probleme, den Quellcode von „Farmer’s Valley“ zu verstehen?	① ② ③ ④ ⑤ ⑥
Falls ja, was hat Ihnen Probleme bereitet?	
Was fanden Sie nicht so gut an der Arbeit mit „Farmer’s Valley“?	
Was hätten Sie sich zusätzlich gewünscht?	
Wie hat Ihnen der Multiplayer-Aspekt gefallen? Wie bewerten Sie diesen?	
Wie bewerten Sie die über große Strecken angewandte Sozialform der Partner-/Gruppenarbeit?	
Sonstige Kommentare:	

Hier sehen Sie das Gerüst für die Klasse „Schüler“ in der Programmiersprache Java. Bitte ergänzen Sie den fehlenden Programmcode so gut Sie können. Falls Sie einen Befehl nicht genau kennen nutzen Sie bitte Pseudo-Code.

Die Klasse soll folgende Daten beinhalten:

- Name
- Alter
- Geschlecht (gespeichert als Zahl: 1 steht für männlich, 2 für weiblich)
- Abiturfächer (gespeichert in einem Array der Größe 4)
- Gruppe (Variable zur Speicherung einer Zahl, die eine Gruppenzugehörigkeit angibt)

```
public class Schueler {
```

Was ist der Name für den folgenden Codeblock?

Antwort:

```
    public Schueler(String dername, int dasalter, int
dasgeschlecht, String[] dieabiturfaecher) {

        name = dername;
        alter = dasalter;
        geschlecht = dasgeschlecht;
        abiturfaecher = dieabiturfaecher;
    }
```

Mit der Methode „gebeAbiturfaecherAus()“ sollen nacheinander alle vier gewählten Abiturfächer ausgegeben werden können. Nutzen Sie hierfür bitte eine for-Schleife.

```
    public void gebeAbiturfaecherAus() {

    }
```

Die Methode „gruppeneinordnung()“ soll dazu dienen, der oben von Ihnen definierten Variable „gruppe“ eine Zahl zuzuordnen. Dabei soll „gruppe“ eine „1“ zugeordnet werden, falls das erste Abiturfach „Informatik“ ist. Eine „2“ soll zugeordnet werden, falls das erste Abiturfach „Mathematik“ ist. Ist das erste Abiturfach ein anderes Fach, so sollen alle vier gespeicherten Abiturfächer gelöscht werden.

```
public void gruppeneinordnung() {
```

```
}
```

Schreiben Sie hier bitte den Programmcode, um einen Schüler mit dem Namen „Peter Müller“ zu erstellen. Er ist 17 Jahre alt, männlich und hat Informatik, Mathematik, Englisch und Deutsch als Abiturfächer gewählt. Danach sollen die beiden gerade von Ihnen geschriebenen Methoden ausgeführt werden, bevor das Programm terminiert.

```
public static void main(String[] args) {
```

```
}
```

```
}
```

### **Anhang C: Ausgefüllter 1. Erhebungsbogen**

Die ausgefüllten Erhebungsbögen befinden sich auf der beigelegten CD im Ordner \Erhebungsbögen\Erhebungsbogen 1.

### **Anhang D: Ausgefüllter 2. Erhebungsbogen**

Die ausgefüllten Erhebungsbögen befinden sich auf der beigelegten CD im Ordner \Erhebungsbögen\Erhebungsbogen 2.

### **Anhang E: Überblick über die wichtigsten serverseitigen Methoden**

#### **public ActorWrapper login(Actor entity, String worldName)**

Mit dieser Methode wird ein Actor auf dem Server in einer anzugebenden Welt eingeloggt.

#### **public ActorWrapper login(Actor entity, String worldName, long id)**

Alle auf dem Server angelegten Objekte (durch login und create) erhalten automatisch eine ID. Diese kann mit dieser login-Methode manuell vergeben werden.

#### **public ActorWrapper login(Actor entity, String worldName, int x, int y)**

Beim Einloggen fügt die Methode automatisch Koordinaten des Objekts in der Welt hinzu. Diese können mit dieser Methode auch manuell gesetzt werden.

#### **public ActorWrapper login(Actor entity, String worldName, int x, int y, long id)**

Alles wird beim Einloggen manuell bestimmt.

#### **public ActorWrapper create(Actor entity, String worldName)**

Speichert ein Objekt aus einer Welt auf dem Server.

**public ActorWrapper update(Actor entity)**

Veränderungen eines Objekts in der Welt werden dem Server mit dieser Methode mitgeteilt. MOSES aktualisiert das richtige Objekt dann anhand seiner ID.

**public List <ActorWrapper> getWorldByActor(Actor entity, boolean self)**

Diese Methode liefert eine Liste aller Objekte, die sich in der Welt des eigenen Bauern (der entity) befinden. Zusätzlich kann mit true oder false der eigene Bauer mit in die Ergebnisliste aufgenommen werden oder nicht.

**public List <ActorWrapper> getWorldChangesByActor(Actor entity, boolean self)**

Ähnlich wie die vorherige Methode. Allerdings werden nicht alle Objekte vom Server geliefert, sondern nur die, die sich seit der letzten Abfrage verändert haben.

**public ActorWrapper get(long entityId)**

Diese Methode liefert eine Referenz auf das Objekt mit der als Parameter übergebenen ID zurück. Gibt es die ID nicht, so wird null zurückgegeben.

**public boolean deleteEntity(Actor entity)**

Mit dieser Methode löscht man das als Parameter übergebene Objekt auf dem Server.

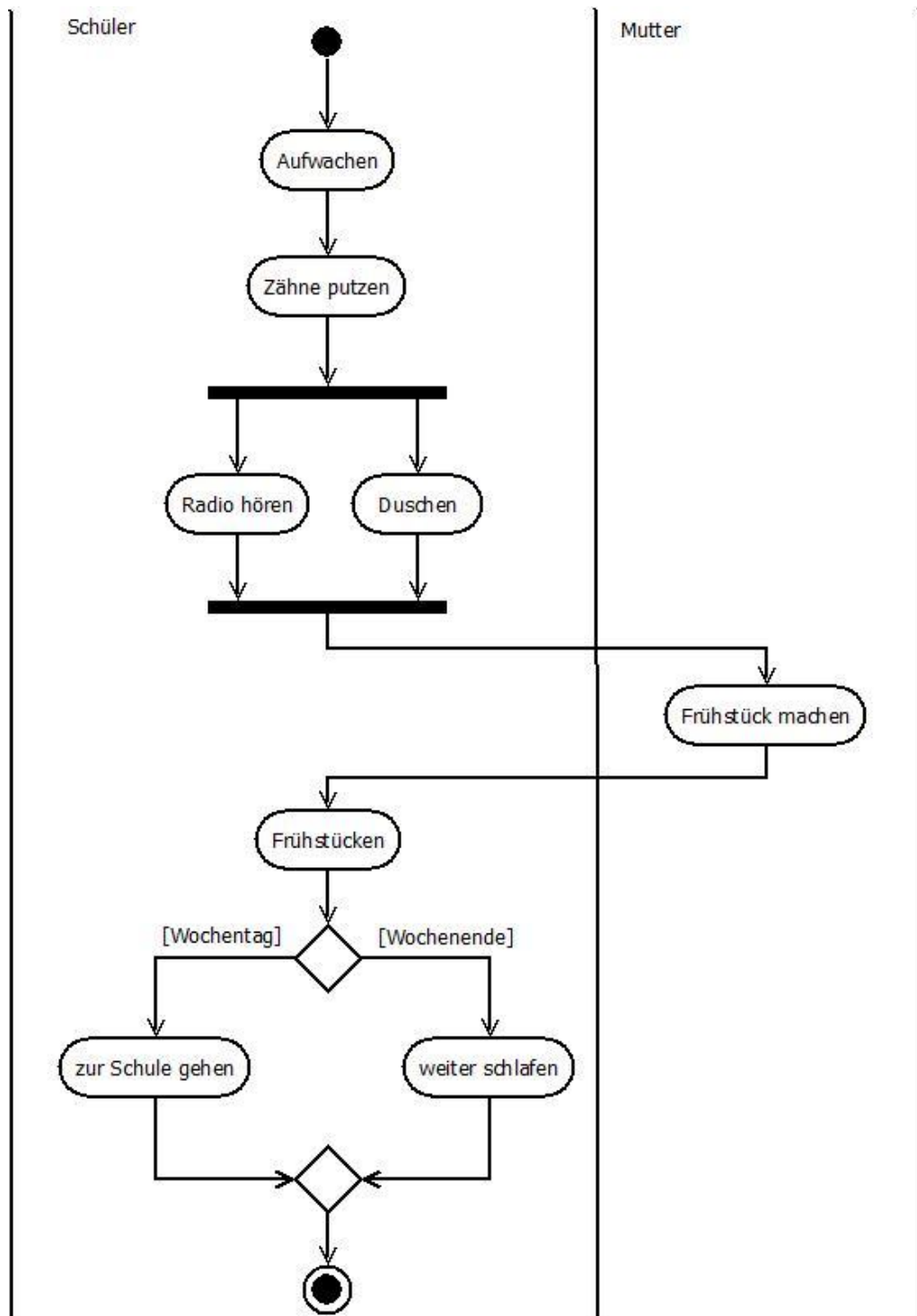
**public boolean deleteEntity(long entityId)**

Hier wird ein Objekt anhand seiner ID gelöscht.

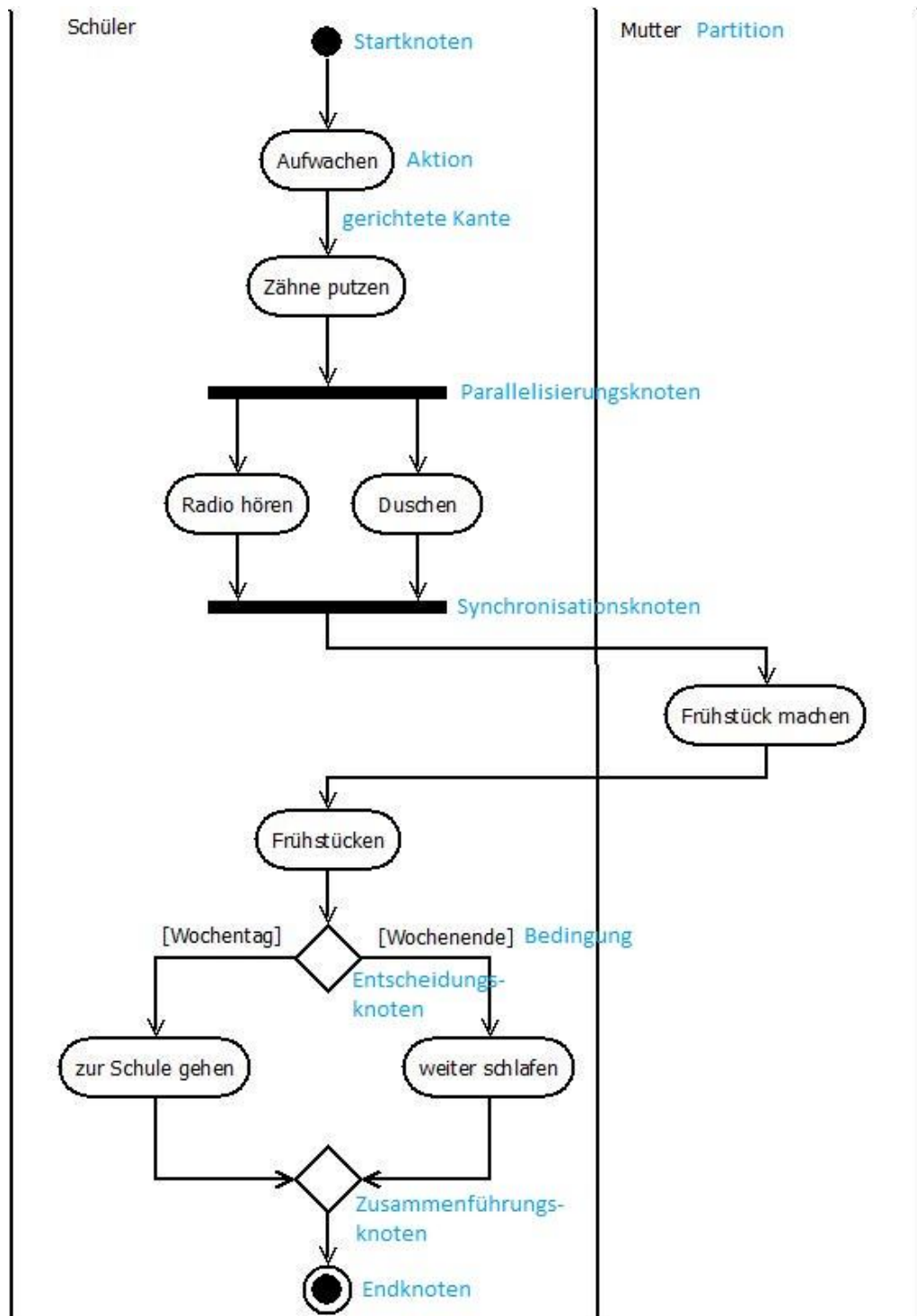
Bei allen Server-Methoden (außer den delete-Methoden) wird ein ActorWrapper-Objekt zurückgeliefert. Dies beinhaltet den Actor und zusätzlich seine Koordinaten in der Welt.

Um den Actor aus dem ActorWrapper zu extrahieren wird die Methode getActor() genutzt. Die x- und y-Koordinate werden mit getX() und getY() abgefragt.

## Anhang F: Einführungs-Aktivitätsdiagramm



## Anhang G: Einführungs-Aktivitätsdiagramm mit Beschriftung



## Anhang H: Schülerergebnisse der Erstellung von Aktivitätsdiagrammen

Die ersten selbst erstellten Aktivitätsdiagramme befinden sich auf der beigelegten CD im Ordner \Aktivitätsdiagramme\Realwelt-Beispiele.

## Anhang I: Die act()-Methode aus der Klasse „Feld“

```
/**
 * Falls ein Feld regenerieren muss, passiert dies im act-Zyklus.
 */

public void act()
{
    // Zuerst wird gecheckt, ob das Feld agieren soll.
    if(darfAgieren) {

        // Regenerieren braucht seine Zeit und so passiert 5 Einheiten lang
        erstmal gar nichts. Erreicht wird dies durch die Variable "zaehler", die
        jedes Feld hat.

        if(regeneriert && zaehler != 0)
            zaehler--;

        else if(regeneriert && zaehler == 0 && korn != 10) {

            // Danach wird dem Feld alle 6 act-Zyklen 1 Einheit Korn gutge-
            schrieben.
            korn++;
            zaehler = 5;

            // Auch hier wird der aktuelle Zustand des Feldes durch verschie-
            dene Bilder angezeigt.
            refreshImage();
        }

        // Wenn das Feld voll regeneriert ist, werden die entsprechenden Va-
        riablen wieder auf den Ursprungszustand zurückgesetzt.
        else if(regeneriert && zaehler == 0 && korn == 10) {

            regeneriert = false;
            bauerId = -1;
            zaehler = 5;
        }
    }
}
```

```
// Nun wird der aktuelle Zustand in MOSES geschrieben.
((FarmersValley)getWorld()).communicator.update(this);

// Das Feld braucht nun erstmal nicht mehr zu agieren.
darfAgieren = false;
}
}

/**
 * Diese Methode setzt jeweils das richtige Bild passend zum aktuellen
 * Zustand des Feldes.
 */

public void refreshImage() {

    // Falls ein Feld regeneriert, wird dies durch Bilder mit einem schwar-
    // zen Punkt gekennzeichnet.
    if (regeneriert) {

        // Verschiedene Bilder zeigen an, wie weit der
        // Regenerierungsprozess bereits fortgeschritten ist.
        if (korn >= 0 && korn <= 5)
            this.setImage("burlapreg.gif");
        else if (korn > 5 && korn <= 9)
            this.setImage("feldhalbreg.png");
        else if (korn == 10)
            this.setImage("feld.png");
    }

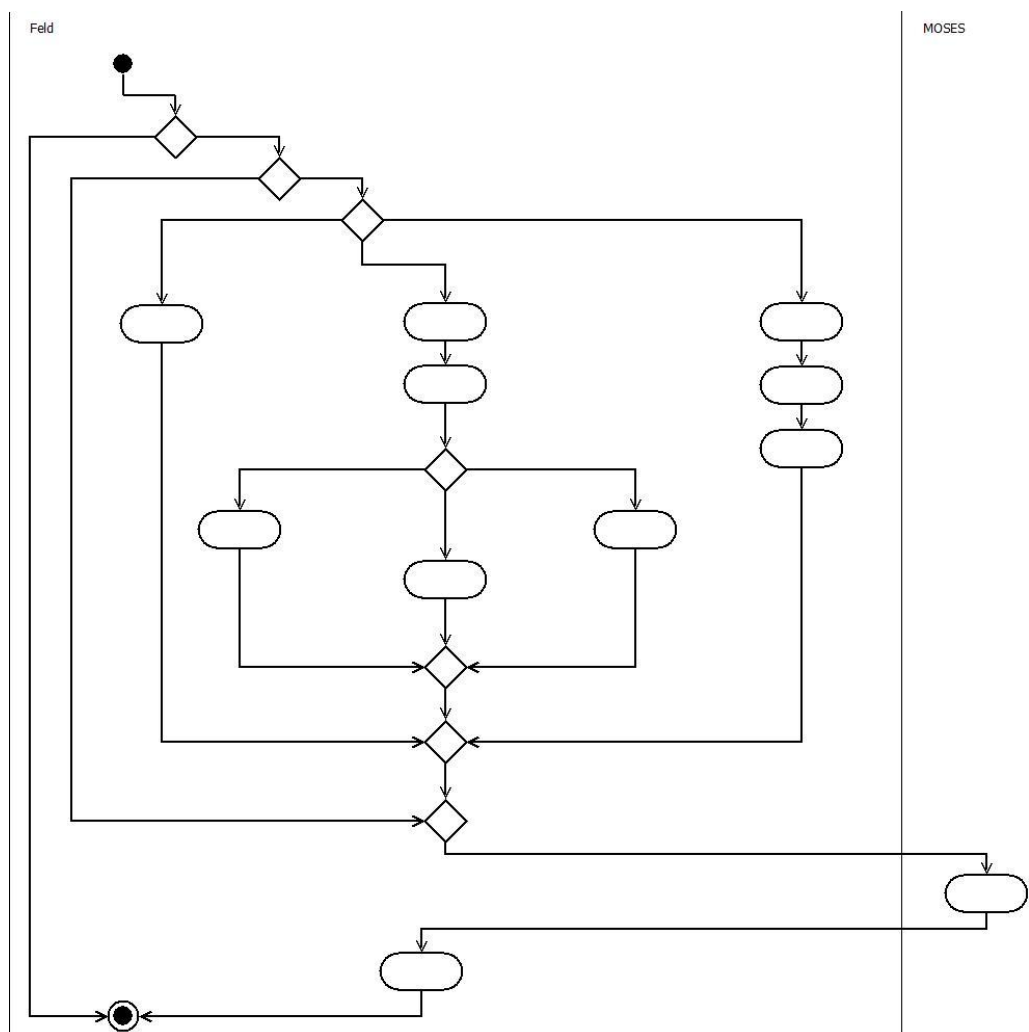
    // Selbiges wird angezeigt, wenn das Feld nicht regeneriert. Nur
    // fehlt dann der schwarze Punkt auf den Bildern.
    else {

        if (korn == 0)
            this.setImage("burlap.gif");
        else if (korn > 0 && korn <= 9)
            this.setImage("feldhalb.png");
        else if (korn == 10)
            this.setImage("feld.png");
    }
}
}
```

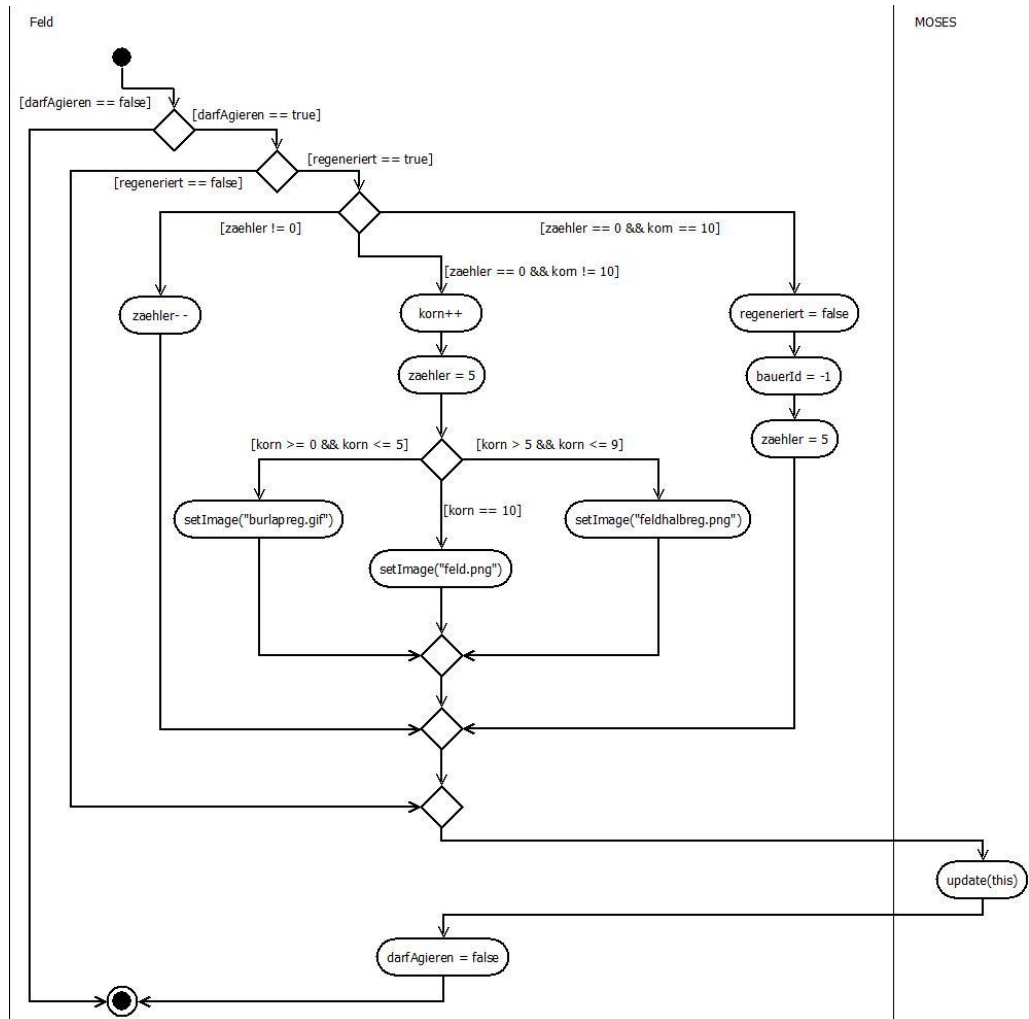
## Anhang J: Modelle der act()-Methode

Die erstellten Aktivitätsdiagramme zur act()-Methode befinden sich auf der beigelegten CD im Ordner \Aktivitätsdiagramme\act()-Methode.

## Anhang K: Aktivitätsdiagramm-Muster zur Schülerunterstützung



**Anhang L: Muster-Diagramm mit Beschriftung**



**Anhang M: Klasse „Feld“ mit wetterabhängigem Regenerierungsvorgang**

Die Schülerergebnisse der Programmieraufgabe, mit der die Veränderung des Regenerierungsvorgangs umgesetzt wurde, befinden sich auf der beigelegten CD im Ordner \Programmieraufgabe Wetter.

## Anhang N: Aufgabenblatt der zweiten Programmieraufgabe

Informatik

**LESSING**  
Gymnasium | Berufskolleg

Programmieren in Greenfoot (Farmers Valley)

Datum:

### Arbeitsauftrag:

Erstellen Sie eine Methode in der Klasse „Bauer“, die verschiedene Konsolenausgaben macht. Sie soll durch einen Klick auf ein Dollarzeichen aufgerufen werden. Das Bild dazu finden Sie im Klassenaustausch im Ordner „BILDER“. Es soll in die Statusleiste eingefügt werden (Koordinaten 10, 12).

Folgende Dinge sollen durch einen Klick ausgegeben werden:

1)

Eine Auflistung von Beträgen, die Anteile am momentanen Gesamtvermögen der Bauernfamilie (bestehend aus Bauer, Frau und zwei Kindern) darstellt. Die zehn zu berechnenden Beträge sind folgende:

- 1: Gesamtvermögen  
Tipp: Auf das eigene Farmhaus kann wie folgt zugegriffen werden:  
`((FarmersValley) getWorld()).myFarmhaus`
- 2: Anteil des Bauern: 50 % vom Gesamtvermögen
- 3: Anteil für Geräte und Instandhaltung des Hofes: 60% vom Anteil des Bauern
- 4: Anteil für die Versorgung der Tiere: 20% vom Anteil des Bauern
- 5: Geld zur freien Verfügung: Rest vom Anteil des Bauern
- 6: Anteil der Frau: 40% vom Gesamtvermögen
- 7: Anteil für die Versorgung der Familie und Hauspflege: 60% vom Anteil der Frau
- 8: Geld zur freien Verfügung: Rest vom Anteil der Frau
- 9: Taschengeld Kind 1
- 10: Taschengeld Kind 2  
Dabei bekommen beide Kinder die restlichen 10% vom Gesamtvermögen und das ältere Kind (Kind 1) 1,5 mal soviel Geld wie das jüngere.

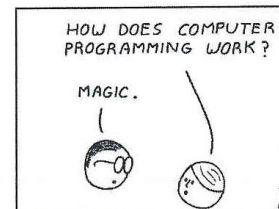
2)

Es soll ausgegeben werden, wieviel gespart werden würde, wenn jedes Familienmitglied 10% vom frei verfügbaren Geld zur Seite legen würde.

3)

Wenn dieser Betrag weniger als 5 beträgt, soll ausgegeben werden, dass es nur sehr wenig Geld. Ist es weniger als 10, dann dass es eine kleine Summe ist. Ist es weniger als 20, dann dass es eine nette Summe ist. In allen anderen Fällen soll ausgegeben werden, dass eine große Summe gespart werden kann.

Nutzen sie bei der Bearbeitung der Aufgaben Bedingungen, Arrays und for-Schleifen.



**Anhang O: Ergebnisse der zweiten Programmieraufgabe**

Die Schülerergebnisse der Bearbeitung des Arbeitsblattes aus Anhang N befindet sich auf der beigelegten CD im Ordner \Programmieraufgabe Bauer.

**Anhang P: Modellierung des Spiels in UPPAAL**

Das System aus endlichen Automaten, welches die Modellierung von *Farmer's Valley* darstellt, befindet sich auf der beigelegten CD im Ordner \UPPAAL.

**Anhang Q: Laufzeitumgebung**

Die der Unterrichtsreihe zugrundeliegende Software befindet sich auf der beigelegten CD im Ordner \Laufzeitumgebung.

## Literaturverzeichnis

Aebli, Hans. *Zwölf Grundformen des Lehrens. Eine Allgemeine Didaktik auf psychologischer Grundlage. Medien und Inhalte didaktischer Kommunikation, der Lernzyklus*. 13. Auflage. Stuttgart: Klett-Cotta, 2006.

Bengtsson, Johan, Kim Larsen, Fredrik Larsson, Paul Pettersson und Wang Yi. *UPPAAL – A Tool Suite for Automatic Verification of Real-Time Systems*. In: *Lecture Notes in Computer Science*. Volume 1066/1996: 232-243. Berlin: Springer, 1996.

Bien, Adam. *Real World Java EE Patterns. Rethinking Best Practices*. Adam Bien Press, 2009.

Bortz, Jürgen und Nicola Döring. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler*. 3. Auflage. Berlin und Heidelberg: Springer, 2002.

Bräu, Karin. *Die Betreuung der Schüler im individualisierenden Unterricht der Sekundarstufe. Strategien und Handlungsmuster der Lehrenden*. In: Rabenstein, Kerstin und Sabine Reh [Hrsg.]. *Kooperatives und selbstständiges Arbeiten von Schülern. Zur Qualitätsentwicklung von Unterricht*. Wiesbaden: Verlag für Sozialwissenschaften, 2007.

Brüning, Ludger und Tobias Saum. *Individuelle Förderung durch Kooperatives Lernen*. In: Kunze, Ingrid und Claudia Solzbacher [Hrsg.]. *Individuelle Förderung in der Sekundarstufe I und II*. 2. Auflage. Baltmannsweiler: Schneider, 2009.

Buchegger, Barbara, Julia Halwax, Lotte Krisper-Ullyett und Johann Ortner. *Collaborative Blended Learning. Eine Orientierung für Lehrende, ModeratorInnen und TutorInnen zum Thema: Wie kann ich das E-Medium für Lernprozesse in der Erwachsenenbildung nutzen?* 2. Auflage. Wien: Facultas, 2007.

Chang, Maiga, Rita Kuo, Kinshuk, Gwo-Dong Chen und Michitaka Hirose [Hrsg.]. *Learning by Playing. Game-based Education System Design and Development. 4<sup>th</sup> International Conference on E-Learning and Games, Edutainment 2009*. Bannf: Springer, 2009.

Cole, Helena und Mark D. Griffiths. *Social Interactions in Massively Multiplayer Online Role-Playing Gamers*. In: *CyberPsychology & Behavior*. August 2007, 10(4): 575-583. New Rochelle: Mary Ann Liebert Inc., 2007

Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3. Auflage. London: Pearson, 2003.

Gosling, James, Bill Joy, Guy Steele und Gilad Bracha. *The Java™ Language Specification*. 3. Auflage. Amsterdam: Addison-Wesley, 2005.

Green, Norm und Kathy Green. *Kooperatives Lernen im Klassenraum und im Kollegium*. 4. Auflage. Seelze-Velber: Kallmeyer und Klett, 2009.

Gutheil, Georg und Norbert Mügge. *Lernort Neue Medien*. Baltmannsweiler: Schneider, 2000.

Hamann, Kathrin. *Lerntypen, Lernstile, Lerntheorien. Eine didaktische Herausforderung für elektronisches Lernen*. Saarbrücken: Dr. Müller, 2007.

Henriksen, Poul, und Michael Kölling. *Greenfoot*. 24. Juli 2008. <http://www.greenfoot.org> (Zugriff am 14.12.2009).

Henriksen, Poul, und Michael Kölling. *Greenfoot: Combining object visualisation with interaction*. In: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications. Vancouver: Association for Computing Machinery, 2004.

Jank, Werner und Hilbert Meyer. *Didaktische Modelle*. 5. Auflage. Berlin: Cornelsen, 2002.

Konrad, Klaus und Silke Traub. *Kooperatives Lernen. Theorie und Praxis in Schule, Hochschule und Erwachsenenbildung*. 3. Auflage. Baltmannsweiler: Schneider, 2008.

Kron, Friedrich W. und Alivisos Sofos. *Mediendidaktik. Neue Medien in Lehr- und Lernprozessen*. München: Reinhardt, 2003.

Lamnek, Siegfried. *Qualitative Sozialforschung. Methoden und Techniken*. 3. Auflage. Weinheim: Beltz, 1995.

Mayer, Richard E. [Hrsg.]. *The Cambridge Handbook of Multimedia Learning*. New York: Cambridge University Press, 2005.

Rabenstein, Kerstin und Sabine Reh [Hrsg.]. *Kooperatives und selbstständiges Arbeiten von Schülern. Zur Qualitätsentwicklung von Unterricht*. Wiesbaden: Verlag für Sozialwissenschaften, 2007.

Reich, Kersten. *Konstruktivistische Didaktik. Lehr- und Studienbuch mit Methodenpool*. 4. Auflage. Weinheim und Basel: Beltz, 2008.

Sacher, Werner. *Schulische Medienarbeit im Computerzeitalter. Grundlagen, Konzepte und Perspektiven*. Bad Heilbrunn: Klinkhardt, 2000.

Storey, Margaret-Anne. *Theories, Methods and Tools in Program Comprehension: Past, Present and Future*. In: IEEE Computer Society. Proceedings of the 13<sup>th</sup> International Workshop on Program Comprehension. St. Louis: IEEE Computer Society Press, 2005.

Stadtfeld, Peter. *Allgemeine Didaktik und Neue Medien. Der Einfluss der Neuen Medien auf didaktische Theorie und Praxis*. Bad Heilbrunn: Klinkhardt, 2004

Teague, Donna und Paul Roe. *Learning to Program. From Pear-Shaped to Pairs*. In: CSEDU 2009. Proceedings of the First International Conference on Computer Supported Education. Volume 2. Lissabon: INSTICC, 2009.

Weidner, Margit. *Kooperatives Lernen im Unterricht*. 3. Auflage. Seelze: Kallmeyer und Klett, 2006.

*Ich versichere, dass ich die Schriftliche Hausarbeit selbstständig verfasst habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen und Formulierungen, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem einzelnen Fall unter genauer Angabe der Quelle als Entlehnung kenntlich gemacht.*

Ort, Datum

Martin Tilmans