TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik
Institut für Software- und Multimediatechnik
AG Didaktik der Informatik / Lehrerbildung

Masterarbeit

zum Thema

Werkzeuge zur Spieleentwicklung und deren Eignung für den Unterricht

vorgelegt von: Marcello Montemaggi
(Geboren am 21.02.1984 in München)
Matrikelnummer.: 3520265
Abgabetermin: 04.02.2014

Aufgabenstellung

Thema:

Informatikunterricht ist stets geprägt durch handlungsorientierten Unterricht, welcher meistens durch die Arbeit am Computer umgesetzt wird. Die Qualität dieses Unterrichts und die Motivation der Schüler gehen dabei immer auch damit einher, inwieweit die im Unterricht verwendeten Programme geeignet sind, um damit die in den Lernzielen festgelegten informatischen Sachverhalte zu vermitteln.

Somit ergibt sich ein Zusammenhang zwischen der Motivation der Schüler und den verwendeten IT-Werkzeugen. Sinnvoll wäre die Verwendung von Programmen, die sich für die Vermittlung der gewünschten Lerninhalte eignen und dabei gleichzeitig Interesse bei den Schülern wecken bzw. bereits existierende Interessen der Schüler aufgreifen, ohne die aus didaktischer und methodischer Sicht vermutete Eignung des Werkzeugs zu verlieren.

Bei Software, die dieses Kriterium erfüllen könnte, handelt es sich um Werkzeuge, die für die Spieleentwicklung konzipiert wurden. Ziel dieser Arbeit soll sein, einige solcher Programme, welche speziell für die Erstellung von einfachen 2D-Videospielen bzw. Animationen konzipiert sind, hinsichtlich ihrer Nützlichkeit für den schulischen (Informatik-)Unterricht zu untersuchen.

Die wissenschaftliche Analyse soll dabei unterscheiden bezüglich des Nutzens dieser Programme, einerseits als Werkzeuge für Lehrer und Tutoren, um damit "serious games" oder "learning content" zu erstellen und andererseits als Hilfsmittel für die Schüler, um bei der Verwendung dieser Art von Software für das Erstellen von Spielen, Kompetenzen zu entwickeln - in und außerhalb des Informatikunterrichts.

Betreuer: Dr. rer. nat. Holger Rohland

Betreuender Hochschullehrer: Prof. Dr. paed. habil. Steffen Friedrich

Institut: Software- und Multimediatechnik

Beginn am: 15.10.2013

Einzureichen am: 04.02.2014

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Im Rahmen des Unterrichts an Schulen kann häufig die Motivation der Schüler ein Problem darstellen. Sinnvoll erscheint deshalb stets ein handlungsorientierter Unterricht im Sinne des Konstruktivismus, welcher die Schüler durch selbstständiges Arbeiten und Lösen von Problemen zu einem Kompetenzerwerb führt.

Während Lehrer in unterschiedlichen Schulfächern häufig noch auf die Unterrichtsmethode des Lehrervortrags setzen um theoretische Inhalte zu vermitteln, stellt der Informatikunterricht eine Ausnahme dar, da hier eine rein theoretische Vermittlung ohne die Verwendung eines Computers undenkbar wäre. Dies ist auch deshalb notwendig, weil ein Verstehen von Informatiksystemen meist nur in der direkten Auseinandersetzung mit diesen möglich ist. Das heißt, dass im Informatikunterricht neben einem geringen Anteil reinen angeleiteten Lernens hauptsächlich selbstständig gearbeitet wird und auch gearbeitet werden sollte. Das hat zur Folge, dass die im Unterricht verwendeten Werkzeuge eine zentrale Rolle für den erwünschten Kompetenzerwerb spielen.

Es findet also häufig ein solcher ganzheitlicher Unterricht statt, dessen Qualität dabei immer auch davon abhängt, wie geeignet die jeweils verwendete Software ist, um damit bestimmte informatische Sachverhalte zu vermitteln. Das hängt wiederum davon ab, wie die Programme aufgebaut sind, welche Funktionen sie bieten und welche Verbindung zwischen ihnen und dem Lerninhalt besteht.

Somit ergibt sich also auch ein didaktischer und lerntheoretischer Zusammenhang zwischen der Motivation der Schüler und den verwendeten IT-Werkzeugen. Unterschiedliche Software kann sich unterschiedlich gut für die Vermittlung eignen [vgl. Mod2011, S.36], da auch die Arbeit damit verschieden stark motivierend ist. Schüler hinterfragen beispielsweise den Nutzen von Programmen, mit denen sie arbeiten sollen oder kennen Softwarealternativen, mit denen sie bevorzugt arbeiten würden.

Sinnvoll wäre dabei die Verwendung von Programmen, die sich für die Vermittlung der gewünschten Lerninhalte eignen und dabei gleichzeitig Interesse bei den Schülern wecken bzw. bereits existierende Interessen der Schüler aufgreifen, ohne auf ihre aus didaktischer und methodischer Sicht vermutete Eignung verzichten zu müssen. Positiv wäre es außerdem, wenn eine Software, die zu diesem Zweck in der Schule zum Einsatz kommt, sich in ihrer Komplexität so einstellen lässt, dass sie sich für den Einsatz in mehreren Jahrgangsstufen eignet. Somit können damit in den jüngeren Schulklassen mit weniger Wissen einfachere Probleme gelöst werden, während in den höheren Klassen die Nutzung der grundlegenden Programmfunktionen teilweise bekannt ist und sofort damit begonnen werden kann,

tionen teilweise bekannt ist und sofort damit begonnen werden kann, komplexere Aufgaben zu bearbeiten.
So kommen auch andere Autoren zu dem Ergebnis, dass Werkzeuge auf die Anforderungen der Lernenden zugeschnitten sein müssen:

> "Werkzeuge sind dazu nötig, Inhalte des Informatikunterrichts zu realisieren. Daher sind die Werkzeuge am angemessensten, die genau so viel können, wie für die Umsetzung der Inhalte nötig ist. Jedes mehr an Möglichkeiten ist für die Schüler eher verwirrend oder (insbesondere bei Entwicklungsumgebungen für Programmiersprachen) ablenkend." [Bur2000, S.45]

Hier sei als Beispiel dafür, wo eine solche Einstellbarkeit der Komplexität gewährleistet wird, *Lazarus for Education* genannt, eine IDE für die Programmiersprache *Pascal* die den Compiler *Free Pascal* nutzt und eine individuelle einstellbare Komponentenauswahl ermöglicht [oV1].
Zwar soll es auch das Ziel des Informatikunterrichts sein, solche Kompetenzen herauszubilden, die die Schüler darauf vorbereiten, sich selbst in unterschiedliche Programme einzuarbeiten zu können und deren allgemeine Gemeinsamkeiten in ihrer Gestaltung und in ihren Funktionsprinzipien zu verstehen. Jedoch muss dafür letztlich immer auf vereinzelte Software zurückgegriffen werden. Erst durch eine Vermittlung durch den Lehrer können Schüler dazu befähigt werden, diese allgemeinen Zusammenhänge zu erkennen, zumal auch der Lerneffekt zu beachten ist, der entsteht, wenn die Schüler bei der Wiederverwendung einer bereits bekannten Software ihre bisherigen Erfahrungen im Umgang mit dieser reflektieren können, um damit zu neuen Erkenntnissen zu gelangen.
Bei Software, die diese Kriterien erfüllen könnten, handelt es sich um Werkzeuge, die für die Spieleentwicklung konzipiert sind. Sie besitzen die Möglichkeit sich damit Kenntnisse und Fertigkeiten aus dem Bereich der Informatik anzueignen, erlauben dabei dieses Wissen praktisch und für die Schüler erkennbar sinnvoll einzusetzen und sind andererseits stark verbunden mit dem weit verbreiteten Interesse an Video- bzw. Computerspielen. Ein sehr häufig verwendetes, bekanntes und speziell in der informatischen Bildung verwendetes Beispiel ist dabei die Software *Scratch*.

1.1 Zielsetzung

Ziel dieser Arbeit soll sein, einen Teil der im Internet angebotenen, freien Software, welche für die Erstellung von einfachen 2D-Videospielen konzipiert sind, hinsichtlich ihrer Nützlichkeit im Unterricht an allgemeinbildenden Schulen - dabei besonders für den Informatikunterricht - zu untersuchen. Dabei soll auch ein Vergleich mit dem Werkzeug *Scratch* stattfinden,

dessen Intention es ist, die Unterschiede dieses speziell für die schulische Bildung entwickelten Programms zu der untersuchten Software darzustellen.
Die Schaffung eines oder mehrerer Unterrichtskonzepte, die sich damit befassen, wie speziell die Erstellung von digitalen Lernspielen durch die Schüler realisiert werden kann, soll in dieser Arbeit nicht erfolgen. Hierzu sei auf bereits existierende Literatur hingewiesen, welche sich mit solchen Konzepten und der korrekten Erstellung von *serious games* durch Lehrer und/ oder Schüler befassen. So haben beispielsweise Mäkilä et al. ("Three Approaces Towards Teaching Game Production"), Lemke et al. ("Animationen mit Scratch gestalten") und weitere Autoren bereits Konzepte für konkrete Möglichkeiten der Umsetzung erarbeitet.
Eine Übersicht der wissenschaftlich untersuchten Werkzeuge findet sich unter Punkt 2.5.3, eine Liste aller angesprochenen Programme im Anhang.

1.2 Leitfragen

- Welche Werkzeuge existieren und welche kommen für den Unterricht in Frage?
- Wie funktionieren die untersuchten Werkzeuge, wie sind sie aufgebaut, welche Funktionen bieten sie und wie benutzerfreundlich ist deren Handhabung?
- Eignen sich die Funktionen der Werkzeuge, um damit Sachverhalte und Probleme aus der informatischen Bildung darzustellen, zu erklären und zu lösen?
- Wie unterscheiden sich die Werkzeuge hinsichtlich ihrer Funktionen, Bedienungsfreundlichkeit und Eignung für den Informatikunterricht?
- Ist mit den Werkzeugen die Erstellung digitaler Lernspiele / *serious games* möglich?
- Wie sehen Lernspiele aus, die mit diesen Werkzeugen erstellt wurden und wie unterscheiden sie sich von anderen Lernspielen?
- Wie effizient sind die Werkzeuge bei der Erstellung von Lernspielen?
- Lässt sich mit den Werkzeugen *Learning Content* erstellen, wie effizient lässt sich dieser *Content* erstellen und wie einfach ist der Umgang mit diesen zu erlernen?
- Lassen sich die Werkzeuge als informatische Werkzeuge von Schülern verwenden und welcher Erkenntnisgewinn ist dabei möglich?
- Welche anderen Werkzeuge und Hilfsmittel existieren, die den Umgang mit den Werkzeugen erklären und/ oder andere Einflüsse auf die Verwendbarkeit der Werkzeuge haben?
- Wie müssten Werkzeuge gestaltet sein, damit sie sich im Bereich der informatischen Bildung für die Umsetzung der Lehrplaninhalte eignen?

1.3 Begriffsklärung

Bevor weiter auf die Analyse dieser Programme zur Erstellung von 2D Videospielen eingegangen werden kann, soll an dieser Stelle eine kleine Begriffsunterscheidung stattfinden. So werden die Bezeichnungen "framework" und "game engine" in dieser Arbeit, wie auch sonst sehr häufig, synonym verwendet, obwohl ein gewisser Unterschied besteht. Das begründet sich dadurch, dass in der Literatur nur wenige Definitionen existieren, die diese Begriffe klar trennen.

Frameworks sind dabei Sammlungen vorgefertigter Programme, die dem Programmierer bei der Erstellung eines Programms das eigenhändige Programmieren bestimmter Grundfunktionalitäten für dieses ersparen.

So weist Wurm auf den folgenden Sachverhalt hin:

> "Dabei sei nicht vergessen, dass eine Programmiersprache allein heutzutage nur einen Teil Ihrer Arbeitsmittel darstellt. Viele Sprachen wie alle .NET-Sprachen (wie VB.NET, C# etc.) oder Java besitzen ein entsprechendes Framework, also eine Sammlung von Grundfunktionalitäten, die Sie benutzen können. Dieses Framework bietet Hunderte und Tausende von Programmbausteinen, die Sie bei der Entwicklung Ihrer eigenen Programme benutzen können." [Wu2010, S.97]

Das bedeutet beispielsweise, dass ein Programmierer bei der Erstellung eines Computerspiels nie damit beginnen wird, ein Programm zu schreiben, das es ihm ermöglicht, Musikdateien des *WAVE* Dateiformats zu verwenden. Stattdessen wird er sich eine entsprechende vorgefertigte *Library* (Bibliothek) suchen, welche er in sein Projekt einbinden muss.

Ein *Framework* könnte also auch Bibliotheken zur Verfügung stellen, die es erlauben, Aufgaben bezüglich der Grafik zu lösen. *Frameworks* haben dabei aber an sich nichts mit Videospielen zu tun. Anders als bei *Game Engines* wird beispielsweise nicht bestimmt, ob ein damit erstelltes Videospiel in 2D oder 3D ist. Zusammengefasst lässt sich also sagen, dass "Game Engines" nichts anderes als "Frameworks" sind, die speziell auf die Erstellung von Spielen einer bestimmten Form, teilweise auch auf ein bestimmtes Genre (Rollenspiele, Shooter, etc.) zugeschnitten sind. Weiterhin kann angenommen werden, dass *Frameworks* und *Game Engines* eher in der Nähe von Programmierumgebungen angesiedelt sind, während die in dieser Arbeit betrachteten Programme meist keine echten Programmiersprachen verwenden und deshalb eher als "game editor", "game creator" oder "game maker" bezeichnet werden.

Es werden demnach in dieser Arbeit "Game Engine" wie auch "Framework" synonym verwendet, wenngleich wie gesagt Unterschiede bestehen.

Von diesen zwei "Software-Arten" wiederum unterscheidet sich der Begriff "IDE". Dieser ist deshalb für die ausstehende Untersuchung der Spielerstellungssoftwares wichtig, weil er den Unterschied zwischen den bisher im schulischen Unterricht verwendeten Programmen für die Programmierung und den in dieser Arbeit untersuchten Werkzeugen aufzeigt.
IDE's (*Integrated Development Environments*) sind Entwicklungsumgebungen, die all die Anwendungsprogramme beinhalten, welche für das Programmieren benötigt werden (Quelltexteditor, Debugger, Compiler, Linker, etc.). Das bedeutet, dass erst in einer IDE der Code geschrieben wird, der beispielsweise *Frameworks* einbindet und verwendet. Somit ist die Erstellung eines "aufwendigeren" Spiels bei Verwendung einer IDE ein langer Weg. *Frameworks* und *Game Engines* hingegen übernehmen eine Vielzahl von Aufgaben und erleichtern dadurch den Erstellungsprozess. Ein treffendes Zitat, das diesen Umstand verdeutlicht, findet sich bei Wang:

> "...stellen Sie sich doch einfach vor, Sie entwickelten ein Spiel und seien gezwungen, all diese technischen Ausdrücke und wie Sie diese selbst programmieren, zu lernen. Das ist der Grund, weshalb viele Programmierer neue Spiele mit Hilfe von Game Engines entwickeln. Ohne Game Engine ist das Entwickeln von Spielen so kompliziert wie das Entwickeln eines Textverarbeitungsprogramms, nur um einen Brief schreiben zu können" [Wa2000, S.382]

Der Unterschied zwischen der Programmierung im Schulunterricht in IDE's sowie der Nutzung der untersuchten *Game Engines* ergibt sich nun also aus der Bildungsabsicht, die man mit diesen Programmen verfolgt. Wurden vor etlichen Jahren Computerspiele sicherlich noch in einer solchen IDE entwickelt, wäre dies heute viel zu aufwendig. Durch den Fortschritt aktueller Spiele in Bezug auf die Kriterien Grafik, Sound, Steuerung, etc., wäre es nun viel zu umständlich, alle für die Umsetzung dieser Elemente nötigen Funktionen stets neu zu programmieren.
Dasselbe gilt auch für die Erstellung von Spielen im Informatikunterricht. Zwar ist es möglich, einfache Spiele in solch einer IDE zu erstellen, doch wirken diese – verglichen mit dem, was mit den *Game Engines* möglich ist – für die Schüler meist veraltet. Außerdem handelt es sich bei den in einer IDE erstellten Spielen meist um sehr kleine Projekte, die nach deren Fertigstellung keine Gründe mehr bieten, sich weiter mit ihrer Programmierung zu beschäftigen.
Dies ist auch der Unterschied der Nutzung eines *Frameworks* im Gegensatz zur Verwendung einer Programmierumgebung (IDE) im schulischen Unterricht. Der Schwerpunkt bei letzteren liegt auf dem reinen Erlernen des Programmierens; mit den Spieleerstellungssoftwares lassen sich hingegen noch andere informatische Inhalte umsetzen. Der Lernerfolg, der dabei zu ver-

zeichnen ist, hängt wie in jedem Unterricht weiterhin von seiner Gestaltung, dem Inhalt, der Schülermotivation und weiteren Einflussfaktoren ab.
Durch den Einsatz von *Game Engines* soll hingegen vor allem der wichtige Einflussfaktor Motivation verstärkt werden. Das Programmieren eines Computerspiels in einer IDE ist in vielen Fällen zu aufwendig bzw. es dauert weitaus länger, um damit ein Spiel umzusetzen bzw. überhaupt erst einen Stand zu erreichen, an dem Teile des Endergebnisses zu erahnen sind, welche dann die Motivation verstärken können.
Eine Übersicht über die Aufgaben die *Game Engines* dem Programmierer abnehmen fasst Freiknecht wie folgt zusammen:

> "Eine Engine übernimmt unter anderem folgende Aufgaben:
> - Bietet Stammfunktionen, die in Spielen häufig benötigt werden, wie z.B. für Kollisionsabfragen, [...] und Rendering von 3D-Objekten, das Laden von Sounds etc.
> - Übernimmt die Ressourcenverwaltung, sodass wir uns nicht um die Verwaltung der Buffer im Grafikspeicher kümmern müssen oder in Speicherlecks laufen.
> - Partikeleffekte werden speicherschonend angeboten, um Nebel, Regen, Schnee oder Feuer darzustellen [...]
> - Netzwerkfunktionen für Multiplayerspiele und Internetupdates
> - Abbilden von Graphical-User-Interface-(GUI)-Elementen, wie etwa von Texten, Knöpfen, Schiebereglern, Eingabefeldern oder Bildern
> - Verwalten von und Kommunizieren mit Eingabegeräten, wie Maus, Tastatur oder Gamepads
> - Einfache Funktionen für Dateizugriffe für z.B. Speicherstände
> - Schnittstellen für Erweiterungen in Form von DLLs oder Bibliotheken
> - Multiplattformfähigkeit" [Frei2012, S.1f]

Wie aus diesen Punkten zu schlusszufolgern ist, wäre die Erstellung eines Videospiels ohne ein *Framework* also weitaus aufwendiger. Auch für die Betrachtung der untersuchten Programme hinsichtlich ihrer Eignung für den Informatikunterricht können aus dieser Zusammenfassung Schlüsse gezogen werden. So würde allein die Kollisionsberechnung von 3D-Objekten bereits universitäres Niveau darstellen, ebenso wie das bloße Ändern einer Textur zur Laufzeit fortgeschrittene Kenntnisse in Grafikprogrammierung und Softwarearchitektur (Entwurfsmuster, etc.) voraussetzt. Es ist daher für den Unterricht ausreichend, wenn für die Programmierung von Spielen *Game Engines* verwendet werden, die sich gezielt für die Vermittlung bestimmter Lehrplaninhalte eignen, beispielsweise durch das Anbieten von Sprachelementen zum Erlernen von Inhalten des Bereichs Algorithmen und Datenstrukturen.

Es bleibt jedoch auch zu klären, für die Umsetzung welcher Computerspiele (und auch generell für welche Lehrplaninhalte) beispielsweise nicht schon gewöhnliche Programmierumgebungen ausreichend sind, da diese unter Umständen für die Erstellung sehr einfacher Spiele (Tic-Tac-Toe, etc.) genügen, bei denen es beispielsweise gar nicht nötig ist, während des Spiels eine Textur zu wechseln.
Es scheint allerdings so, als würde gerade die Algorithmierung und Programmierung, welche teilweise über die Arbeit mit Programmierumgebungen umgesetzt wird, besonders häufig ein Problem für Schüler darstellen. Hierzu stellt Modrow fest:

> "Sehen wir uns die schon mehrfach zitierte 80-Prozent-Ausstiegsquote im Informatikunterricht an, so stellen wir fest, dass sie nur dann auftritt, wenn die Schülerinnen und Schüler "Programme schreiben" sollen." [Mod2011, S.36]

Jedoch nicht nur das Themengebiet, wie hier die Algorithmierung / Programmierung, bestimmt, ob der Einsatz von *Frameworks* oder Programmierumgebungen vorzuziehen ist, auch die Lerninhalte selbst tun dies. Ein Beispiel wäre die Vermittlung von rekursiven Algorithmen in einer Programmierumgebung wie *MSW Logo* sehr viel anschaulicher, als dies in einer Spielerstellungssoftware der Fall sein wird.

1.4 Hypothesen:

- Die untersuchten Werkzeuge eignen sich für bestimmte Sachverhalte als Entwicklungsumgebung für digitale Lernspiele / "*serious games*" besser als bestimmte andere Software (Programmierumgebung, etc.).
- Wenn die genannten Werkzeuge den Schülern als Entwicklungsumgebung für kreative Gestaltung von Videospielen angeboten werden, dann lassen sich damit einige informatische Sachverhalte so vermitteln, dass sie von den Schülern durch selbstständige Arbeit und mit erhöhter Motivation erarbeitet werden. Außerdem werden die Schüler dazu angeregt, sich selbstständig und auf freiwilliger Basis weiter mit informatischen Sachverhalten zu befassen.
- Die genannten Werkzeuge eignen sich zur Vermittlung von informatischen Inhalten zur Grafik, zu Algorithmierung (Grundstrukturen, Struktogramme, Programmierparadigma, Programmierkonzepte, etc.), zu Automaten, zur Datensicherheit und zur Darstellung von Zahlen.
- Die genannten Werkzeuge eignen sich für wenige Sachverhalte als Entwicklungsumgebung für Learning Content besser als allgemein bekannte und häufig verwendete Software.

4. Zusammenfassung

Die erfolgte Untersuchung hat gezeigt, dass Programme zur Erstellung von Computerspielen eine Vielzahl von Möglichkeiten für den Unterricht eröffnen. Der mögliche Einsatz hängt dabei hauptsächlich vom gewählten *Framework* ab, welches den Einarbeitungsaufwand, das Anforderungsniveau der Schüler (Voraussetzungen) und die Art der Inhalte bestimmt, die damit vermittelt werden können. Aufgrund der zu Beginn der Arbeit durch eine Umfrage festgestellten Altersstruktur lässt sich vermuten, dass dabei besonders der Einsatz in der Sekundarstufe 1 und 2 erfolgsversprechend ist. Durch das hohe Anforderungsniveau des *LÖVE Frameworks* bietet sich dieses unter Umständen gar erst für die Nutzung im Tertiärbereich an.

Bezüglich der konkreten Umsetzung in Form eines Planungsentwurfes für einen regelmäßigen Einsatz im Unterricht oder für zeitweise Nutzung, beispielsweise während einer Projektwoche, wurden in dieser Arbeit keine Aussagen getroffen. Für den Informatikunterricht kann hierzu jedoch hinzugefügt werden, dass der Einsatz bzw. das Testen mehrerer *Frameworks* ratsam ist. Einerseits deshalb, da gerade hinsichtlich der Zufriedenheit im Sinne der *Usability* Schüler selbst entscheiden müssen, ob die Arbeit mit den *Frameworks* zufriedenstellend ist, und andererseits aber auch, da durch die Realisierung eines Projekts in mehreren Spieleerstellungssoftwares und ein damit verbundenes Erkennen von Gemeinsamkeiten und Unterschieden dieser Arbeit durch die Schüler, womöglich auch zusätzliche informatische Kompetenzen erworben werden können.

Eine Analyse ob sich die die *Frameworks* für die Erstellung von *Learning Content* eignen, wie es Inhalt einer Hypothese war, wurde in dieser Arbeit nicht untersucht. Es lässt sich aufgrund der Ergebnisse jedoch vermuten, dass dies nur in Einzelfällen, beispielsweise für die Erstellung von Animationen, ratsam ist. So wäre es möglich, durch Kombination eines Bildschirmaufnahmeprogramms mit den *Frameworks* teilweise möglich, Animationen ähnlich wie durch die Software *Adobe Flash* herzustellen, ohne jedoch ein solches kommerzielles Programm kaufen zu müssen. Um jedoch genauere Aussagen treffen zu können, wäre es notwendig gesonderte Untersuchung anzustellen.
Weiterhin zeigte sich, dass alle bis auf die eben besprochene Hypothese bestätigt werden konnten. Eine abschließende Klärung des vermuteten Motivationscharakters der Programme wäre jedoch erst auf Basis von empirischen Studien möglich, welche Schüler, die über längere Zeit mit den Programmen im Schulunterricht gearbeitet haben, hinsichtlich der Akzeptanz der *Frameworks* befragt und diese Ergebnisse mit den Ansichten der Lehrer bezüglich der damit erreichten Lernerfolge vergleicht.

In Betrachtung aller *Frameworks* hinsichtlich der Umsetzung des Informatiklehrplans kann außerdem festgestellt werden, dass es theoretisch noch weitere Möglichkeiten der Verwendung im Unterricht gäbe, welche jedoch durch den Funktionsumfang dieser Programme nicht abgedeckt werden. Besonders bezüglich der Informatikthemengebiete der Datenbanken und Rechnernetze ergäben sich hier noch viele Erweiterungsmöglichkeiten. So ist es beispielsweise in keinem *Framework* (bis vermutlich auf das *LÖVE Framework*) möglich, eine eigene Datenbank zu erstellen, auf die im Spiel selbst zugegriffen werden kann. In diesem Sinne sind die untersuchten Programme (bis auf *Scratch*) weit davon entfernt, wie eine speziell auf den Informatikunterricht zugeschnittene Spieleerstellungssoftware aussehen könnte.

Zusammenfassend kann also festgestellt werden, dass der Lernerfolg beim Einsatz einer Spielerstellungssoftware von dessen Wahl und den genannten, damit einhergehenden Faktoren abhängig ist. Dennoch sei erwähnt, dass dies auch bedeutet, dass der Lernerfolg somit letztlich vollständig durch den Lehrer bestimmt wird, welcher die Rahmenbedingungen, die Auswahl des Programms und die didaktische und methodische Umsetzung vorgibt und letztlich auch geeignete Ideen für mögliche Spiele bzw. Projekte haben muss, welche den Schülern Spaß machen und dennoch didaktisch sinnvoll und mit einem Lerngewinn verbunden sind.

5. Glossar

IDE	Abkürzung für "Integrierte Entwicklungsumgebung". Ist eine Sammlung von Anwendungsprogrammen, mit denen die Aufgaben der Softwareentwicklung bearbeitet werden können. Besteht in der Regel aus einem Texteditor, Compiler bzw. Interpreter, Linker, Debugger und einer Quelltextformatierungsfunktion (Quelle: wikipedia).
A*Pathfinding	Der A*-Algorithmus gehört zur Klasse der informierten Suchalgorithmen. Er dient in der Informatik der Berechnung eines kürzesten Pfades zwischen zwei Knoten in einem Graphen mit positiven Kantengewichten. Der Algorithmus gilt als Verallgemeinerung und Erweiterung des Dijkstra-Algorithmus, in vielen Fällen kann aber umgekehrt A* auch auf Dijkstra reduziert werden. Der Algorithmus ist optimal. Das heißt, dass immer die optimale Lösung gefunden wird, falls eine existiert. Quelltextformatierungsfunktion (Quelle: wikipedia).
Beta Status	Der Beta Status einer Software bedeutet, dass diese in einer Beta-Version, also einer unfertigen Version vorliegt. Häufig sind Beta-Versionen die ersten Versionen eines Programms, die vom Hersteller zu Testzwecken veröffentlicht werden (Quelle: wikipedia).
Effektivität	Effektivität ist ein Maß für Wirksamkeit, das das Verhältnis von erreichtem Ziel zu definiertem Ziel beschreibt. Es gibt Aufschluss darüber, wie nahe ein erzieltes Ergebnis dem angestrebten Ergebnis gekommen ist. Effektiv arbeiten bedeutet, so zu arbeiten, dass ein angestrebtes Ergebnis erreicht wird (Quelle: wikipedia).
Effizienz	Das Kriterium Effizienz betrifft den Aufwand, der zur Erreichung des Ziels nötig ist. Effizienz ist somit ein Maß für die Wirtschaftlichkeit bzw. Kosten-Nutzen-Relation (Quelle: wikipedia).
Interpreter	Ein Interpreter (im Sinne der Softwaretechnik) ist ein Computerprogramm, das einen Programm-Quellcode im Gegensatz zu Assemblern oder Compilern nicht in eine auf dem System direkt ausführbare Datei übersetzt, sondern den Quellcode einliest, analysiert und ausführt. Die Analyse des Quellcodes erfolgt also zur Laufzeit des Programmes(Quelle: wikipedia).
Klon	In der Informationstechnik ist ein Klon [...] ein Software System, welches entwickelt wurde um ein anderes System genau nachzubilden. Bei Computerspielen, welche versuchen ein bekanntes andere Computerspiel so nah wie nur möglich nachzubilden, wird dabei ebenso von einem Klon gesprochen (Quelle: wikipedia).
Leveln	Als Stufe oder oft auch englisch Level bezeichnet man in vielen Rollenspielen einen Zahlenwert, der zusammenfassend die Stärke und den Fortschritt der Entwicklung eines Spielercharakters angibt. Als Leveln wird dabei allgemein das Aufsteigen des Charakters (also das Fortschreiten der Entwicklung der Spielfigur) bezeichnet (Quelle: wikipedia).
open source	Open Source und quelloffen nennt man Werke, deren Lizenzbestimmungen besagen, dass man mit deren Empfang auch den dazugehörigen Quelltext

	empfängt. Der Copyrightbesitzer behält dabei lediglich das Recht, die Lizenzbestimmungen zu verändern (Quelle: wikipedia).
Programmierparadigma	Ein Programmierparadigma ist ein fundamentaler Programmierstil. Der Programmierung liegen je nach Design der einzelnen Programmiersprache verschiedene Prinzipien zugrunde. Programmierparadigmen unterscheiden sich durch ihre Konzepte für die Repräsentation von statischen (wie beispielsweise Objekte, Methoden, Variablen, Konstanten) und dynamischen (wie beispielsweise Zuweisungen, Kontrollfluss, Datenfluss) Programmelementen. Grundlegend für den Entwurf von Programmiersprachen sind die Paradigmen der imperativen und der deklarativen Programmierung (Quelle: wikipedia).
Rollenspiel	Die Bezeichnung "Rollenspiel" als Konzept eines interaktiven Spiels, ist ein Überbegriff für unterschiedliche Arten von Video- bzw. Computerspielen des Genre der Abenteuerspiele. Als Basis diesen unterschiedliche Regelwerke, wie die der Pen-&-Paper-Rollenspiele oder der japanische Rollenspiele. In den Anfängen der Computer-RPGs wurden solche als ein Spiel definiert, bei dem man einen Charakter (oder mehrere) steuerte und Fähigkeiten durch die Steigerung von Spielwerten vergrößerte, außerdem möglichst unbegrenzte Handlungsfreiheit hatte. Ein Spiel, das den Schwerpunkt auf feste Handlungsstränge legt, gilt nach dieser klassischen Definition als Adventure. Viele der aktuellen Rollenspieltitel distanzieren sich jedoch mit ihren ausgedehnten, festen Storylines von diesen Grundlagen und würden nach klassischer Definition eher als Adventures mit Rollenspielelementen oder Rollenspiel-Adventure-Hybriden bezeichnet werden (Quelle: wikipedia).
Rückruffunktion	Eine Rückruffunktion (*callback function*) bezeichnet in der Informatik eine Funktion, die einer anderen Funktion als Parameter übergeben und von dieser unter gewissen Bedingungen aufgerufen wird. *Event handler* sind Beispiele einer Art von Rückruffunktion, welche bei einem Ereignis Aktionen ausführen (Quelle: wikipedia).
Tetris	Tetris ist ein Puzzle-artiges Computerspiel des russischen Programmierers Alexei Paschitnow. Tetris gilt inzwischen als Computerspiel-Klassiker, hat sich inzwischen 100 millionenfach verkauft, wurde vielfach ausgezeichnet und ist wie kaum ein anderes Spiel für nahezu jedes elektronische Rechensystem erschienen (Quelle: wikipedia).
TicTacToe	Tic Tac Toe ist ein klassisches, einfaches Zweipersonen-Strategiespiel, dessen Geschichte sich bis ins 12. Jahrhundert v. Chr. zurückverfolgen lässt. Auf einem quadratischen, 3×3 Felder großen Spielfeld setzen die beiden Spieler abwechselnd ihr Zeichen (ein Spieler Kreuze, der andere Kreise) in ein freies Feld. Der Spieler, der als erster drei Zeichen in eine Zeile, Spalte oder Diagonale setzen kann, gewinnt (Quelle: wikipedia).

6. Literatur- und Quellenverzeichnis

[001en1] http://www.Engine001.com/Games/Wolf%20Hunt_28.htm

[Bigl2009] Bigl Benjamin (2009), Game over? - Was vom Spielen übrig bleibt - Empiri sche Studie zu assoziativen Transfereffekten zwischen Spiel- und Alltagswelt, Martin Meidenbauer Verlagsbuchhandlung, München. ISBN 978-3-89975-913-6

[Bur2000] Burger Helmut, Galata Rudolf, Hechenleitner Andrea, Kreisel Dr. Klaus, Leppmeier Max, Liessel Werner, Wiedemann Albert (2000), Informatik in der Schule. Zentralstelle für Unterricht, Augsburg, http://www.fachberater-informatik.de/uploads/media/ZUM_Informatik_02.pdf (Stand: 07/11/13)

[cnet] http://download.cnet.com/RPG-Maker-XP/3000-7536_4-10437117.html (Stand: 07/11/13)

[E001] http://www.Engine001.com/education.htm (Stand: 07/11/13)

[Eg2011] Egenfeldt-Nielsen Simon, Meyer Bente, Sørensen Birgitte Holm (2011), Seri ous games in education - A global Perspective. Aarhus University Press, Aar hus, ISBN 978-87-7124-259-1

[euroG] http://www.euroGamer.net/articles/super-columbine-massacre-rpg-part-1-article (Stand: 07/11/13)

[fsm] "First Seed Material", http://www.tekepon.net/fsm (Stand: 07/11/13)

[Frei2012] Freiknecht Jonas (2012), Spiele entwickeln mit Gamestudio - Virtuelle 3D-Welten mit Gamestudio A8 und Lite-C. Carl Hanser Verlag, München, ISBN 978-3-446-43119-5

[GaSa] http://gamesalad.com/education/learn-more (Stand: 07/11/13)

[Gedit1] http://Game-editor.com/forum/viewtopic.php?f=1&t=11454 (Stand: 07/11/13)

[Gedit2] http://Game-editor.com/docs/how_Game_editor_works.htm (Stand: 07/11/13)

[Gedit3] http://Game-editor.com/forum/viewtopic.php?f=6&t=6028 (Stand: 07/11/13)

[Gedit4] http://Game-editor.com/docs/gettingstarted_maineditormenu.htm (Stand: 07/11/13)

[Gedit5] http://Game-editor.com/docs/gettingstarted_theactorcontrolpanel.htm (Stand: 07/11/13)

[Gedit6] http://Game-editor.com/docs/scripting.htm (Stand: 07/11/13)

[Gedit7] http://Game-editor.com/forum/viewtopic.php?f=4&t=9219 (Stand: 07/11/13)

[Gedit8] http://Game-editor.com/forum/viewtopic.php?f=6&t=11178&p=77841 (Stand: 07/11/13)

[Gedti9] http://Game-editor.com/docs/script_reference.htm (Stand: 07/11/13)

[Itune] https://itunes.apple.com/app/prince-of-dragon/id396451476?mt=8 (Stand: 07/11/13)

[Kirk2001] Kirk Susanne (2001), Aus der virtuellen Welt in die surplus reality. In: From me, Johannes; Meder, Norbert (Hrsg.), 2001, Bildung und Computerspiele - Zum kreativen Umgang mit elektronischen Bildschirmspielen, Leske + Budrich, Opladen, ISBN 3-8100-2841-X

[Klim2008] Klimmt Christoph (2008), Unterhaltungserleben beim Computerspielen - Theorie, Experimente, (pädagogische) Anwendungsperspektiven. In: Mitgutsch, Konstantin; Rosenstingl, Herbert (Hrsg.), 2008, Faszination Computerspielen - Theorie - Kultur - Erleben, Wilhelm Braumüller Universitäts-Verlagsbuchhandlung GmbH, Wien, ISBN 978-3-7003-1674-9

[Love1] https://www.love2d.org/wiki/love (Stand: 07/11/13)

[Love2] https://www.love2d.org/wiki/love.graphics.draw (Stand: 07/11/13)

[Love3] https://love2d.org/forums/viewtopic.php?f=4&p=63001 (Stand: 07/11/13)

[Love4] https://love2d.org/forums/viewtopic.php?f=5&t=11187 (Stand: 07/11/13)

[Love5] https://love2d.org/forums/viewtopic.php?f=5&t=7720 (Stand: 07/11/13)

[LPG2011] Sächsischer Lehrplan Informatik an Gymnasien, 2011

[Mal2008] Maloney John, Peppler Kylie, Kafai B. Yasmin, Resnick Mitchel, Rusk Natalie (2008), Programming by Choice - Urban Youth Learning Programming with Scratch. In SIGCSE: Proceedings of the 39th SIGCSE technical symposium on Computer science education, ACM, New York, pp.367-371, ISBN 978-1-59593-799-5

[Marr2010] Marr Ann Christine, Kaiser Ronal (2010), Serious Games für die Informations- und Wissensvermittlung - Bibliothken auf neuen Wegen. In: Fuhlrott Rolf, Krauß-Leichert Ute, Schütte Christoph-Hubert (Hrsg.), 2010, B.I.T.online-Innovativ Band 28, Dinges & Frick GmbH Verlag, Wiesbaden, ISBN 978-3-934997-31-8

[MIT1] o.V., Programming with Scratch, Lifelong Kindergarten Group MIT Media Lab, http://scratched.media.mit.edu/sites/default/files/Programming-with-Scratch.pdf (Stand: 07/11/13)

[mission] http://www.immersiveeducation.eu/index.php/missionmakerm (Stand: 07/11/13)

[Mod2011] Modrow Eckart, Mönig Jens, Strecker Kerstin (2011), Wozu JAVA? - Plädoyer für grafisches Programmieren. In LOG IN 30 2011, Heft Nr. 168, Berlin, pp.35-41, ISSN 0720-8642

[Nie2008] Niegemann Helmut M., Domagk Steffi, Hessel Silvia, Hein Alexandra, Hupfer Matthias, Zobel Anett (2008), Kompendium multimediales Lernen. Springer Verlag, Berlin / Heidelberg, ISBN 978-3-540-37225-7

[Oust1998] Ousterhout John K. (1998), Scripting: Higher Level Programming for the 21st Century. In: IEEE Computer Society Press Volume 31, Issue 3, Los Alamitos, pp.23-30

[oV1] http://www2.sn.schule.de/~matdb/matdb2/index.php?action=articleview&a=384 (Stand: 07/11/13)

[Pren2001] Prensky Marc (2001), Digital Game-Based Learning - New roles for trainers and teachers - How to combine computer games and learning, Real-life case studies from organizations utilizing game-based techniques. Paragon House, St.Paul, ISBN 1-55778-863-4

[RPGM1] http://tryace.rpgmakerweb.com/download-lite (Stand: 07/11/13)

[RPGM2] http://rpgmaker.net/events/rm_ace_lite_cook_off/ (Stand: 07/11/13)

[RPGM3] http://www.rpgmakerweb.com/download/additional/run-time-packages (Stand: 07/11/13)

[RPGM4] http://forums.rpgmakerweb.com/index.php?/topic/84-lunas-tiles/ (Stand: 07/11/13)

[RPGM5] http://rpgmaker.net/Games/4844/downloads/4680/ (Stand: 07/11/13)

[RPGM6] http://rpgmaker.net/events/rm_ace_lite_cook_off/ (Stand: 07/11/13)

[RPGM7] http://www.rpgmakervxace.net/topic/14510-pac-man-remake/) (Stand: 07/11/13)

[RPGM8] http://www.rpg-studio.de/scientia/RGSS (Stand: 07/11/13)

[Rup2006] Ruppik, Sindy (2006), Game-Based-Learning für Kinder. Konzeption Entwick lung und Praxis eines kollaborativen Lernspiels. VDM Verlag Dr. Müller e.K. und Lizenzgeber, Saarbrücken, ISBN 978-3-86550-493-7

[Schra2008] Schrammel Sabrina (2008), Play-based learning - Die Aktivität des Computer spielens als Lernanlass. In: Mitgutsch, Konstantin; Rosenstingl, Herbert (Hrsg), 2008, Faszination Computerspielen - Theorie - Kultur - Erleben, Wilhelm Braumüller Universitäts-Verlagsbuchhandlung GmbH, Wien, ISBN 978-3-7003-1674-9

[Scra1] http://scratch.mit.edu/ (Stand: 07/11/13)

[Scra2] http://wiki.scratch.mit.edu/wiki/Build_Your_Own_Blocks_(Scratch_Modification)

[Scra3] http://scratch.mit.edu/search/projects/?q=metal+slug&date=anytime (Stand: 07/11/13)

[Scra4] http://info.scratch.mit.edu/sites/infoscratch.media.mit.edu/files/file/ScratchProgrammingConcepts-v14.pdf (Stand: 07/11/13)

[Scra5] http://scratch.mit.edu/explore/projects/Games/?date=this_month&order_by=view_count (Stand: 07/11/13)

[Scra6] http://scratch.mit.edu/projects/13701368/ (Stand: 07/11/13)

[Scra7] http://wiki.scratch.mit.edu/wiki/Define_%28%29_%28block%29 (Stand: 07/11/13)

[Seda] Seda, Roman (2008), Interactive Storytelling im Computerspiel - Adventure Games im Spiegel polymedialer Einflüsse. Werner Hülsbusch Verlag, Boizen burg ISBN 978-3-940317-33-9

[Stab] http://stabyourself.net/ (Stand: 07/11/13)

[Stencyl] http://www.stencyl.com (Stand: 07/11/13)

[Timm2000] Timmermann Bettina (2000), Programmierung in der Lehrerausbildung, In: Schubert S., Schwill A. (Hrsg.), 2000, Tagungsband zum Workshop Lehrerbildung Informatik - Konzepte und Erfahrungen, Workshop im Rahmen der GI-Jahrestagung, Berlin

[Vest1975] Vester Frederic (1975), Denken, Lernen, Vergessen - Was geht in unserem Kopf vor, wie lernt das Gehirn, und wann lässt es uns im Stich?. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, ISBN 3-421-02672-6

[Wa2000] Wang Wallace (2000), Programmieren für Dummies - Gegen den täglichen Frust beim Programmieren. MIPT-Verlag, Bonn, ISBN 3-8266-2869-1

[Wu2010] Wurm Bernhard (2010), Programmieren lernen! - Schritt für Schritt zum ersten Programm. Galileo Press, Bonn, ISBN 3-8362-1462-8

[Ytube] http://www.youtube.com/watch?v=SaoHMjS04vU (Stand: 07/11/13)

7. Anlagen: