IMST – Innovationen machen Schulen Top

Informatik kreativ unterrichten

SCRATCH GEGEN GAME MAKER

ID 413

Martin Schenk

BRG-Viktring

Viktring, Juli 2011

INHALTSVERZEICHNIS

ABSTRACT

Es gibt zahlreiche virtuelle Umgebungen, die es Lernenden einfacher machen in die Welt des Programmierens einzusteigen. Mit Scratch und Game Maker bieten sich zwei bekannte und hochentwickelte grafische Benutzeroberflächen zum Erstellen von Programmen an. Mit ganz unterschiedlichen Ansätzen bei der Erstellung spielerischer Anwendungen sollen ProgrammiereinsteigerInnen den Umgang mit Programmiersprachenkonzepten und das Erstellen von Programmen erlernen. Einerseits soll geklärt werden, ob diese virtuellen Lern- bzw. Programmierumgebungen die SchülerInnen zum Anwenden algorithmischer Problemlösungsstrategien motivieren können. Andererseits sollen die Stärken und Schwächen beider Systeme herausgearbeitet werden.

Die Projektdurchführung war so aufgebaut, dass sich je eine Informatikgruppe über mehrere Wochen mit einem der beiden Programme beschäftigte. Dabei mussten sie alleine und auch in Zweier- oder Dreiergruppen Aufgaben bearbeiten und auch eigene Programmideen entwickeln und verwirklichen. An einem eigenen Nachmittagstermin stellten sich die beiden Gruppen die selbst programmierten Anwendungen vor, wobei sie auch auf den Entwicklungsprozess mit dem jeweiligen Tool – Scratch oder Game Maker – eingingen. In späteren Einheiten erfolgte abschließend eine Reflexion der gemachten Erfahrungen sowohl von Seiten der SchülerInnen als auch von mir als Lehrer.

Schulstufe:	9
Fach:	Informatik
Kontaktperson:	Martin Schenk
Kontaktadresse:	BRG-Viktring Stift-Viktring-Straße 25 9073 Viktring

1 EINLEITUNG

1.1 Motivation

Im aktuellen Informatik-Lehrplan der 5. Klassen der AHS findet man folgenden Satz:

„Einblicke in wesentliche Begriffe und Methoden der Informatik, ihre typischen Denk- und Arbeitsweisen, ihre historische Entwicklung sowie ihre technischen und theoretischen Grundlagen gewinnen und Grundprinzipien von Automaten, Algorithmen und Programmen kennen lernen." (BMUKK, 2004, S. 2)

Diese Forderung bietet einen relativ weiten (theoretischen) Spielraum, dem aber eine konkrete Unterrichtsgestaltung folgen muss. Schließt aber die Passage „kennen lernen der Grundprinzipien von Programmen" die Programmentwicklung mit ein? Aus meiner Sicht kann diese Frage mit einem klarem „Ja" beantwortet werden. Allerdings ist heute die Frage nach einer geeigneten Entwicklungsumgebung, bedingt durch die Qual der Wahl unter vielen Mikrowelten und gängiger Produktionssprachen, gar nicht so einfach zu beantworten.

Im Rahmen meiner Ausbildung zum Informatiklehrer an der Universität Klagenfurt habe ich mich auch mit zwei Mikrowelten beschäftigt, die aus meiner Sicht taugliche Werkzeuge für einen niederschwelligen Einstieg ins Programmieren sind. Es handelt sich um die zwei Programme „Scratch" und „Game Maker". Nicht nur das derzeit weltweit boomende Werkzeug „Scratch", sondern auch das Programm „Game Maker" bietet viele kreative Möglichkeiten. Einerseits können kreative Spielideen umgesetzt werden und andererseits können „Drehbücher" zur Abfolge von interaktiven Bildsequenzen verwirklicht werden.

Es reizt mich nun, einerseits ausgehend von identen Aufgabenstellungen beide Systeme im unterrichtlichen Einsatz zu vergleichen, andererseits die Kreativität von SchülerInnen anzusprechen, in dem sie zunächst eigene Skripts und Drehbücher entwickeln und diese umzusetzen versuchen. Dabei sollen die Stärken und Schwächen beider Entwicklungsumgebungen herausgearbeitet werden.

1.2 Rahmenbedingungen

Das Projekt wurde im Zuge des Informatikunterrichts im BRG-Viktring durchgeführt. Beteiligt waren zwei SchülerInnengruppen, eine aus einer Klasse des bildnerischen Zweigs (5B) und die andere aus einer Klasse des musikalischen Zweigs (5D). Die SchülerInnen dieser Schwerpunktklassen haben ein erhöhtes Stundenausmaß (5 – 7 Wochenstunden) bei den jeweiligen Schwerpunkten. Ich persönlich arbeite sehr gerne mit diesen Schwerpunktklassen, vor allem die SchülerInnen des bildnerischen Zweigs scheinen mir etwas mutiger beim Einsetzen ihres kreativen Potentials als andere. Aber durch die schwerpunktmäßige Ausrichtung kommt es auch immer wieder zum Ausfall von Informatikunterrichtsstunden, z.B. durch Chorwochen, Chorproben, BE-Wochen, Be-Projekten, Vorbereitungen für unser Schulfest etc.

Die Aufteilung der SchülerInnengruppen war wie folgt:

Gruppe 5B (Schwerpunktklasse „Bildnerische Erziehung"): 15 SchülerInnen, 10 Mädchen, 5 Burschen. Informatikunterricht: Donnerstags 1. und 2. Stunde.

Gruppe 5D (Schwerpunktklasse „Musik"): 14 SchülerInnen, 12 Mädchen, 2 Burschen. Informatikunterricht: Mittwochs 1. und 2. Stunde.

Der Unterricht fand wie gewohnt in den Informatikräumen der Schule statt und im Normalfall stand jeder Schülerin bzw. jedem Schüler ein eigener Computer zur Verfügung. In Einzelfällen kam es jedoch vor, dass sich zwei SchülerInnen einen Rechner teilen mussten.

1.3 Ziele

1.3.1 Ziele auf SchülerInnenebene

1.3.1.1 Teilziel 1: Spielerisches Kennenlernen von Programmstrukturen

Die SchülerInnen sollen nach dem Projekt die wichtigsten Strukturen eines Programms (z.B. Schleifen, Wenn-Dann-Abfragen, Variablen, …) erkennen bzw. selbst anwenden können. Das möchte ich einerseits durch das spielerische Kennenlernen eines einfachen Werkzeugs zur Programmierung von Minianwendungen erreichen. Einfache aber ganz konkrete Aufgabenstellungen sollen dabei den Nutzen bzw. die Notwendigkeit der verschiedenen Programmstrukturen für die Lernenden erkennbar machen.

Andererseits soll durch die Einbindung von Bild- und Tonbearbeitung in den Arbeitsablauf die Phantasie angesprochen werden, um die gestalterischen Fähigkeiten der SchülerInnen zu aktivieren. Ich erhoffe mir dadurch kreative Ideen, welche zusätzlich motivieren sollen, sich mit den in der jeweiligen Miniwelt zur Verfügung stehenden Programmelementen auseinanderzusetzen.

1.3.1.2 Teilziel 2: Motivation zum Anwenden algorithmischer Problemlösungsstrategien

Im Vergleich zum „klassischen“ Unterricht einer Programmiersprache wie Java oder C, wo man sich erst mühsam mit einer Benutzeroberfläche und der Syntax vertraut machen muss, bietet eine Miniwelt wie Scratch oder Game Maker eine relativ kurze Einarbeitungszeit und schnelle Erfolgserlebnisse. Dadurch sollen die SchülerInnen motiviert werden, sich selbstständig und bestenfalls sogar über den Informatikunterricht hinaus mit algorithmischen Problemlösungsstrategien zu beschäftigen. Durch das Nachdenken über algorithmische Strukturen in der Erfahrungswelt der SchülerInnen, z.B. im Mathematikunterricht, soll auch das Interesse an höheren Programmiersprachen geweckt werden. Was nicht zuletzt auch bei der Entscheidung für einen späteren Besuch des Wahlpflichtfachs Informatik ausschlaggebend sein kann.

1.3.1.3 Teilziel 3: Softwareentwicklung als kreativen schöpferischen Prozess erkennen

Für mich ist besonders wichtig, dass das Entwickeln eines Programms von den SchülerInnen nicht als langweiliges und mühseliges Lernen einer Programmiersprachensyntax und dem Auswendiglernen von Programmstrukturen in Erinnerung bleibt. Die SchülerInnen sollen vielmehr erkennen, dass Softwareentwicklung in erster Linie einen kreativen schöpferischen Prozess darstellt.

1.3.2 Ziele auf LehrerInnenebene

1.3.2.1 Teilziel 1

Durch den Vergleich von Scratch und Game Maker sollen die Stärken und Schwächen der beiden Systeme herausgearbeitet werden. Den InformatiklehrerInnen soll damit eine Entscheidungshilfe für die Auswahl des richtigen Werkzeugs geboten werden.

1.3.2.2 Teilziel 2

Sowohl die verwendeten Aufgabenstellungen als auch die fertig erstellten Programme der SchülerInnen können als Anregung für den eigenen Unterricht oder nachfolgende Projekte dienen.

1.4 Vorgangsweise

1.4.1 Gruppenteilung

Durch die Beteiligung von SchülerInnen aus zwei Klassen bot sich eine Aufteilung in zwei Vergleichsgruppen an. Die Gruppe „Scratch“ wird von den SchülerInnen der Klasse 5D gebildet und die Gruppe „Game Maker“ wird von den SchülerInnen der Klasse 5B gebildet.

1.4.2 Phase I

In Phase I wird die Lern- bzw. Programmierumgebung vorgestellt, die Möglichkeiten werden aufgezeigt und einfache vorgegebene Spiel- bzw. Anwendungskonzepte umgesetzt.

1.4.3 Phase II

In dieser Phase sollen die SchülerInnen selbstständig ein Programmkonzept entwickeln und bis zu einem fertigen Programm ausarbeiten.

1.4.4 Phase III

In Phase III sollen die SchülerInnen der beiden Klassen zusammentreffen, dafür ist ein eigener Unterrichtstermin am Nachmittag geplant. Die SchülerInnen werden zu MentorInnen des jeweiligen Systems und helfen sich beim Einarbeiten in das jeweils andere System. Und zwar mit den ähnlichen Aufgabenstellungen wie in Phase I. Dabei wird interessant zu beobachten sein, ob das Vorwissen aus Phase I und II die Einarbeitung in das jeweils andere System abkürzt oder gar behindert.

Der geplante Zweck dieser dritten Phase besteht aus zwei Teilen, die SchülerInnen sollen

1. zeigen was sie auf dem gelernten System beherrschen und zumindest Teile davon auch weitergeben.
2. sich mit dem anderen System vertraut machen - zumindest soweit, dass sie zu einer kritischen Gegenüberstellung fähig sind.

1.5 Zeitplan

September 2010

- Konzeptuelle Planung und grobe Fixierung im Jahresplan

Oktober / November 2010

- Einarbeitung in die Systeme (2 Gruppen) seitens der SchülerInnen, die weitestgehend selbstständig erfolgen soll, mit mir als Lerngestalter und Third Level Support
- Realisierung von konkreten, vorgegebenen Aufgabenstellungen

Jänner 2011

- Kurzwiederholung, Schreiben von Drehbüchern durch die SchülerInnen und Realisierung eines „Spieles“ im Sinne von „serious games“, also bevorzugt Spiele mit didaktischem Wert

Feber 2011

- Zusammentreffen der beiden Gruppen, gegenseitiges Vorstellen der Programmkonzepte und damit realisierter Ergebnisse

Mai 2011

- Nachhaltigkeitsüberprüfung, selbstständige Aufgabenbewältigung
  Was ist „hängengeblieben“?
- Evaluation des Projekts – Schülerfragebogen

2 PROJEKTINHALT

2.1 Ablauf des Projekts

Alle Phasen der Planung wurden durchgeführt, allerdings konnte der Zeitplan nicht ganz eingehalten werden, weshalb sich die Realisierung der eigenen Spielkonzepte und Phase III, also das Zusammentreffen der Gruppen etwa um vier Wochen nach hinten verschoben hat. Außerdem beschränkte sich Phase III auf das gegenseitige Vorstellen der entwickelten Programme. Eine zusätzliche Einarbeitung in das jeweils andere System wurde unter anderem aus Zeitgründen nicht mehr durchgeführt. Es gab aber wohl eine von mir durchgeführte Präsentation der wichtigsten Unterschiede von Scratch und Game Maker sowie eine Diskussion der Vor- und Nachteile der jeweiligen Konzepte.

2.2 Phase I

2.2.1 Gruppe Scratch

Mit den SchülerInnen der Gruppe Scratch wurde nach dem Testen von bereits fertigen Anwendungen gemeinsam das erste Programm erarbeitet. Ich habe die nötigen Schritte vorgezeigt und die SchülerInnen konnten sich durch einfaches „Nachklicken" mit der Benutzeroberfläche von Scratch vertraut machen. Ein einfaches Spiel, bei welchem man eine Kugel, welche vom Rand abprallt, mit einem Schläger treffen muss. Sinn dieses ersten Programms war hauptsächlich, die Möglichkeiten aufzuzeigen und Lust auf das selbstständige Tun zu machen. Da alle SchülerInnen Gefallen an diesem Spiel fanden, wurde es in der nächsten Einheit noch etwas erweitert und ausgebaut. Dabei gab ich nur Anregungen zum Verändern bereits vorhandener Programmelemente und ließ die SchülerInnen weitgehend selbstständig arbeiten. Bei der Gestaltung der Sprites hatten alle freie Hand und konnten ihrer Kreativität freien Lauf lassen, was auch ausgiebig genutzt wurde.

Abb. 1: „Paddle Ball" – ein erstes Programm, hier bereits in erweiterter Form

In weiterer Folge konnten die SchülerInnen ein Konzept, wie Programmschleifen oder Verzweigungen, von einem vorgegebenen Beispielprogramm erlernen und dann selbst nachbauen bzw. ergänzen. Die jeweiligen Beispielprogramme wurden über das schuleigene Moodle zur Verfügung gestellt und auch die von den SchülerInnen erstellten Programme mussten wieder in eine Datenbank im selben Moodlekurs hochgeladen werden. Nach dem Fertigstellen des jeweiligen eigenen Programms konnten so die Programme der MitschülerInnen getestet werden.

Phase I zog sich insgesamt über 4 Einheiten (4 Doppelstunden), diese Einheiten fanden aber nicht hintereindander statt, sondern wurden durch andere Themengebiete der Informatik unterbrochen.

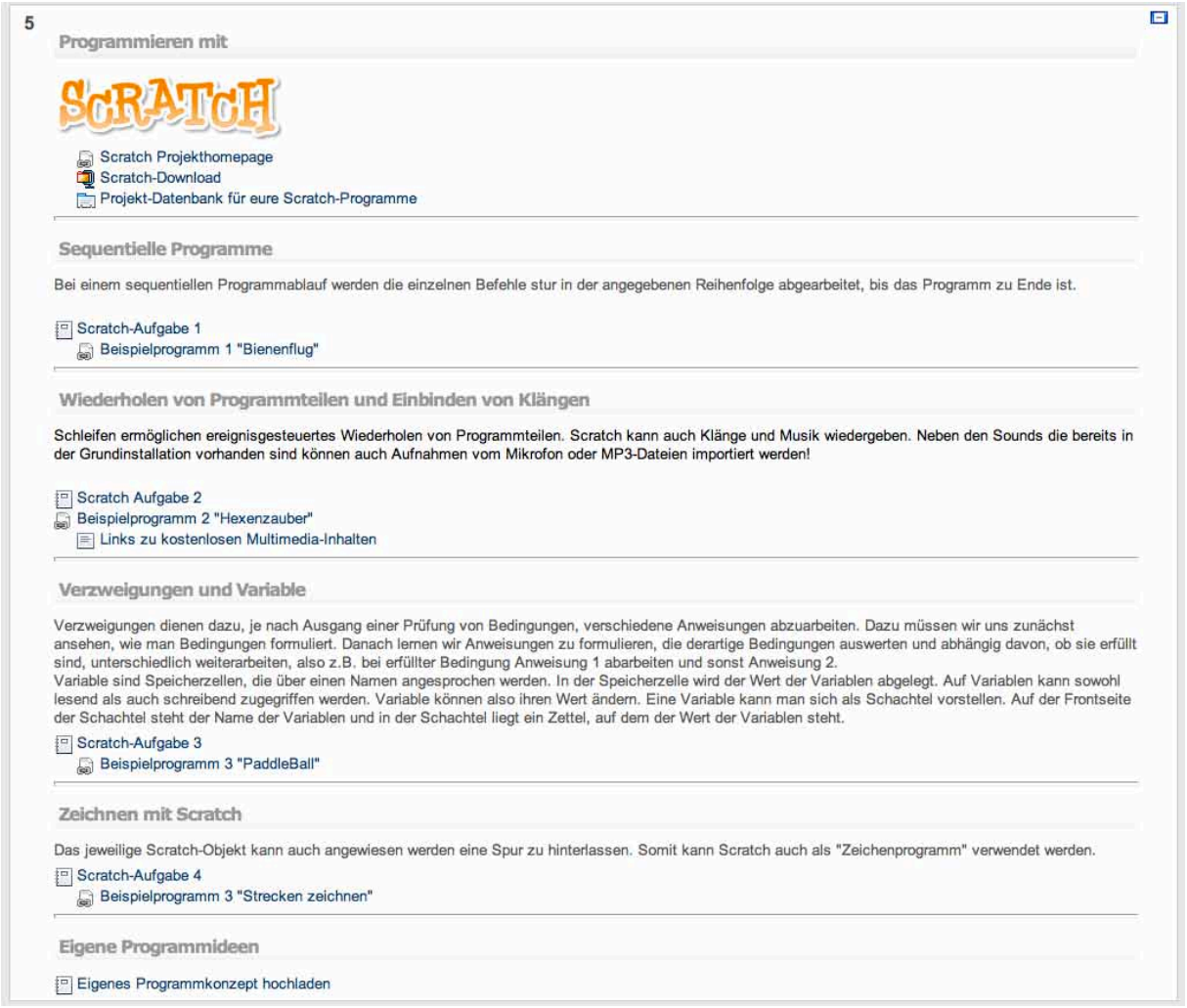


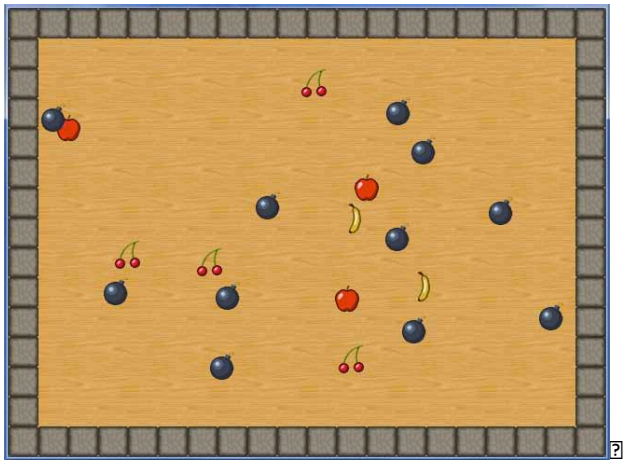
Abb. 2: Aufbau des Scratch-Themenblocks in Moodle

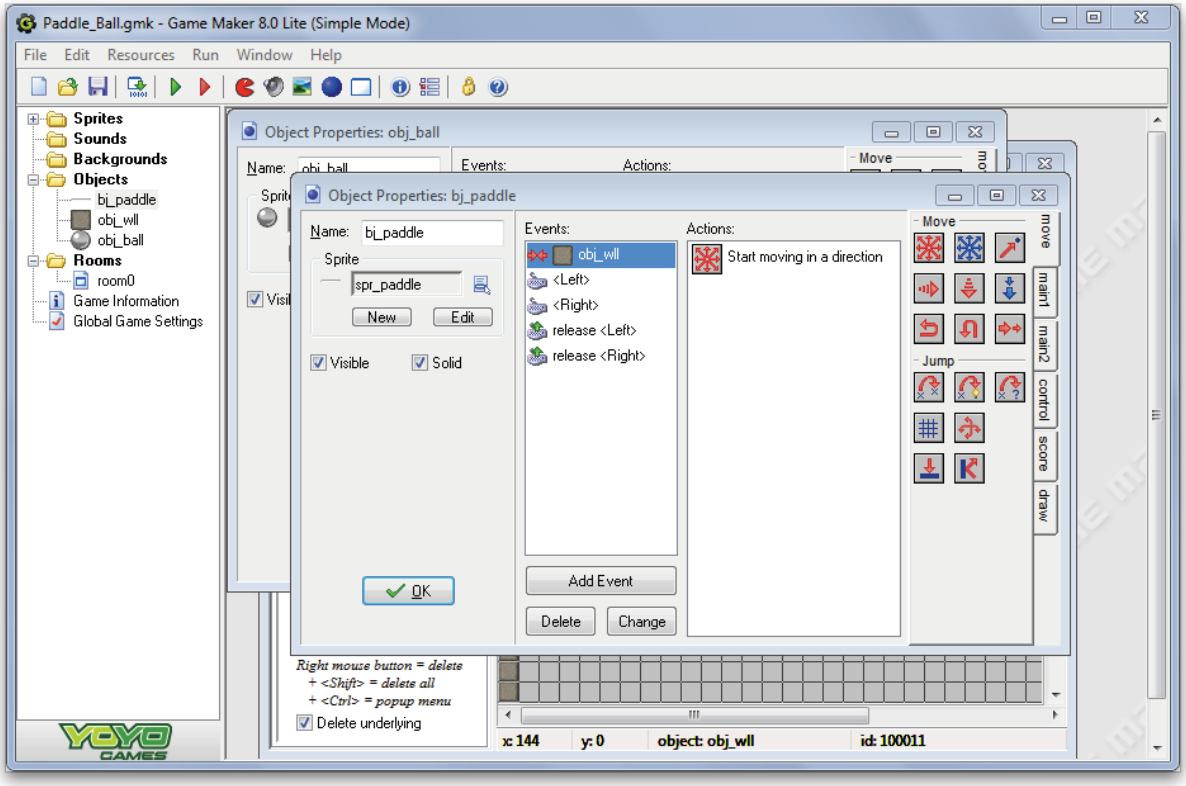
2.2.2 Gruppe Game Maker

Auch die SchülerInnen der Gruppe Game Maker konnten zu Beginn ausgiebig bereits fertige Programme testen und sich so einen Eindruck über die Möglichkeiten verschaffen. Da die mit Game Maker erstellten Endprodukte oftmals sehr professionell aussehen, waren die SchülerInnen von den im Internet gefundenen Anwendungen sehr angetan.

Als nächsten Schritt programmierten wir unsere erste Game Maker-Anwendung. Unter meiner Anweisung erstellten die SchülerInnen ein einfaches Spiel, bei welchem man mit der Maus sich über den Bildschirm bewegende Elemente erwischen muss. Auch hier war es meine Absicht zu motivieren und einen Überblick über die Benutzeroberfläche zu schaffen. Leider waren die nötigen Schritte doch so komplex, dass eine zweistündige Einheit nicht ausgereicht hat, um diese erste Anwendung fertigzustellen. Deshalb beschäftigten wir uns auch die nächste zweistündige Einheit mit dieser ersten Anwendung. Nach dem Fertigstellen konnten die SchülerInnen das Spiel durch eigene Elemente (Kopien von bereits vorhandenen Objekten) ergänzen und die Sprites[1] und Hintergründe nach eigenem Geschmack anpassen bzw. verändern.

[1] Mit Sprite bezeichnet man in diesem Zusammenhang die grafische Oberfläche eines Objekts. Ein Objekt kann auch mehrere Sprites besitzen, welche zur Laufzeit des Programms gewechselt werden können.




Paddle_Ball.gmk - Game Maker 8.0 Lite (Simple Mode)
File Edit Resources Run Window Help
Sprites
Sounds
Backgrounds
Objects
bj_paddle
obj_wll
obj_ball
Rooms
room0
Game Information
Global Game Settings
Object Properties: obj_ball
Object Properties: bj_paddle
Name: bj_paddle
Sprite
spr_paddle
New
Edit
Visible
Solid
Events:
Actions:
obj_wll
<Left>
<Right>
release <Left>
release <Right>
Start moving in a direction
Move
Jump
move
main1
main2
control
score
draw
Add Event
Delete
Change
OK
Right mouse button = delete
+ <Shift> = delete all
+ <Ctrl> = popup menu
Delete underlying
x: 144
y: 0
object: obj_wll
id: 100011

2.3 Phase II

Die Aufgabenstellung in Phase II war bei beiden Gruppen die selbe und unterschied sich einzig in der Durchführung. Die erste Aufgabe der SchülerInnen war es, ein eigenes Programmkonzept zu entwickeln und schriftlich festzulegen. Das Konzept sollte folgende Punkte beinhalten:

- Eine vollständige Beschreibung der Funktionalität des Spiels/der Anwendung/der Filmsequenz
- Eine Beschreibung der vorkommenden Figuren/Sprites/Hintergründe
- Mögliche Erweiterungen dieser Anwendung

Die zweite Aufgabe war das Realisieren der im Konzept beschriebenen Anwendung. Beim Programmieren sollten sie sich möglichst an das Konzept halten und nur in Ausnahmefällen von diesem abweichen.

Phase II zog sich bei beiden Gruppen über drei Einheiten (insgesamt sechs Stunden), welche hintereinander stattfanden.

Den Auftrag, dass bevorzugt Konzepte von Anwendungen mit didaktischem Wert (serious games) entwickelt werden sollen, hab ich wohl etwas zu wenig kommuniziert. Jedenfalls fallen die meisten Anwendungen nicht in diese Kategorie. Da für mich dieser Aspekt nicht im Vordergrund stand, hab ich es dabei belassen und die meisten Konzepte, so wie sie waren, akzeptiert.

2.3.1 Die entwickelten Scratch-Anwendungen

Die Konzepte und realisierten Anwendungen wurden in Zweier- bzw. Dreiergruppen komplett eigenständig erarbeitet. Meine Hilfe wurde nur selten beim Zusammenstellen des Programms benötigt, meistens musste ich kleinere Fehler ausbessern.

2.3.1.1 Anwendung 1: „Baby füttern“

Abb. 5: „ Baby füttern“

Ein Baby erscheint am Bildschirm, welches in einer Sprechblase meldet, dass es Hunger hat. Pfeile zeigen auf drei Objekte, welche beim Anklicken dem Baby gefüttert werden. Es gibt ein Fläschchen, ein Happy-Meal und Dynamit. Beim Klicken auf Fläschchen oder Happy-Meal kommt eine zufriedene Rückmeldung des Babys in Form einer Sprechblase. Bei Klick auf das Dynamit, fliegt das Baby rotierend aus dem Bild und das Spiel ist zu Ende.

2.3.1.2 Anwendung 2: „Dart“

Abb. 6: „Dart“

Ziel ist es, einen sich auf- und abbewegenden Dartpfeil im richtigen Moment loszuschießen, um die Zielscheibe zu treffen. Trifft man die Scheibe, bekommt man Punkte und es ändert sich der Hintergrund und die Geschwindigkeit der Auf- und Abbewegung. Trifft man oft genug, endet das Spiel mit einem Gewinner-Bildschirm; schießt man oft daneben, endet das Spiel mit einem Verlierer-Bildschirm.

2.3.1.3 Anwendung 3: „Der Burger“

Abb. 7: „Der Burger“

Ein Burger hat sein Gürkchen verloren, dieses wird in wechselnden Hintergrundbildern gesucht. Im letzten Bild muss der Benutzer das Gürkchen mit der Maus zum Burger steuern. Das Spiel endet mit dem Aufessen des Burgers.

2.3.1.4 Anwendung 4: „ Bauernhof“

Abb. 8: „ Bauernhof“

Ein Schwein soll gefüttert werden, was durch Klicken auf eines von fünf Elementen geschieht. Je nachdem was gefüttert wird, ändert sich die Sprechblasen-Rückmeldung des Schweins, das Aussehen des Schweins und der Hintergrund. Es gibt verschiedene Möglichkeiten das Spiel durch Füttern von unverdaulichen Objekten zu beenden.

2.3.1.5 Anwendung 5: „ Sun-Pong“

Abb. 9: „ Sun-Pong“

Diese Anwendung ist ein Nachbau des Spieleklassikers „ Pong“und kann von zwei Spielern gegeneinander gespielt werden. Jeder Spieler steuert seinen Schläger so, dass er den hin- und herfliegenden Ball erwischt. Fliegt der Ball daneben, bekommt der gegnerische Spieler einen Punkt. Als zusätzliche Schwierigkeit gibt es eine herabschwebende Sonne, von der der Ball abprallt. Das Spiel endet, wenn einer der Spieler zehn Punkte erreicht hat.

2.3.2 Die entwickelten Game-Maker-Anwendungen

Die SchülerInnen der Game-Maker-Gruppe ließen sich nur sehr schwer zum Entwickeln eines Konzepts für eine eigene Programmidee motivieren. Mein Eindruck war, dass viele zwar Ideen hatten, aber sich eine Realisierung mit Game Maker nicht zutrauten.

Auch hier wurden die Konzepte und realisierten Anwendungen in Zweier- bzw. Dreiergruppen erarbeitet. Meine Hilfe wurde jedoch sehr viel häufiger benötigt. Einige SchülerInnen orientierten sich am Beispiel eines interessierten Mitschülers und versuchten sein Konzept „nachzubasteln“.

2.3.2.1 Anwendung 6: „Farben-Labyrinth“

Abb. 10: „ Farben-Labyrinth“

Diese Anwendung wurde von einem Schüler in Einzelarbeit erstellt. Der Benutzer muss mit den Cursor-Tasten eine Figur durch ein Labyrinth steuern und Diamanten sammeln. Dabei muss er sich vor bewegenden gegnerischen Figuren in Acht nehmen. Wenn er alle Diamanten aufgesammelt hat, dann muss er zum letzten Symbol (im ersten Level ein Buch) und kommt ins nächste Level. Wird er zuoft von gegnerischen Figuren berührt, endet das Spiel.

Es gibt mehrere Levels (Green, Blue, Red, etc.) und das Spiel ist noch ausbaufähig. Alle Grafiken wurden vom Schüler selbst erstellt bzw. angepasst, einen großen Teil der Arbeit erledigte der Schüler zuhause, nach eigenen Angaben aus „persönlichem Interesse am Spiele programmieren“.

2.3.2.2 Anwendung 7: „Tennis“

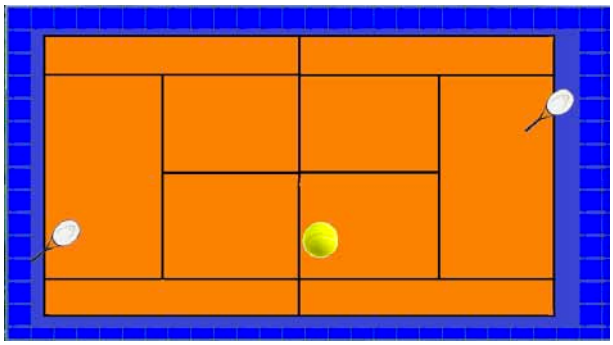


Abb. 11: „Tennis“

Auch bei dieser Anwendung handelt es sich um einen Pong-Nachbau. Leider sind die beiden Schüler nicht ganz fertig geworden, so lassen sich zwar die Schläger bewegen und auch der Ball „ fliegt“ über den Bildschirm, aber das Abprallen des Balls von den Schlägern ist noch nicht implementiert und auch die Punktezählung fehlt noch völlig.

2.3.2.3 Anwendung 8: „Wer bekommt Justin Bieber“

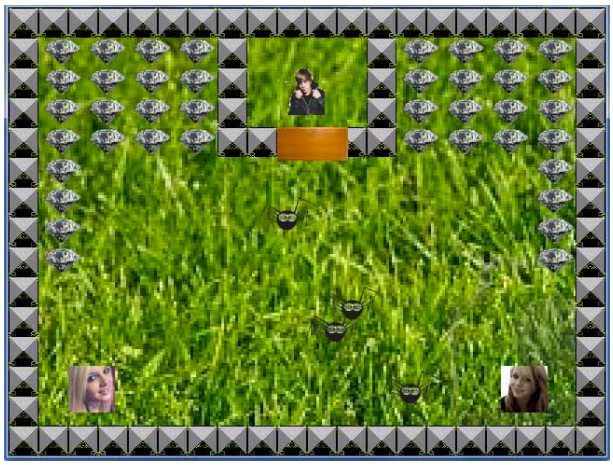


Abb. 12: „Wer bekommt Justin Bieber“

Diese Anwendung wurde von zwei Schülerinnen erstellt. Zwei Figuren, mit den Fotos der beiden Schülerinnen als Sprites, spielen gegeneinander. Es geht darum Spinnen zu fressen und Diamanten zu sammeln bis das Tor geöffnet wird, welche Figur als erstes Justin Bieber erreicht hat gewonnen.

2.3.2.4 Anwendung 9: „Mr und Mrs Frick“

Abb. 13: „Mr und Mrs Frick“

Diese Anwendung wurde von einem Schüler und einer Schülerin realisiert und wird zu zweit gespielt. Bei diesem Spiel geht es um einen Ehekrach, wobei sich Mann und Frau mit Gegenständen (Fernbedienung und Bratpfanne) bewerfen. Als Ziel hatten sich die Beiden ausgedacht, dass derjenige mit mehr Treffern gewinnt. Leider fehlt auch hier die engültige Implementierung des Punktestands.

Die Schülerin und der Schüler, die diese Anwendung erstellt haben, sind ein Paar. Durch - nicht ganz ernst gemeinte - Aussagen ihrer MitschülerInnen („ihr seid ja wie ein altes Ehepaar“) kamen sie zur Idee dieser Anwendung. Ich hatte den Eindruck, dass die Beiden damit die typischen Rollenklischees – Mann vor dem Fernseher, Frau in der Küche – eher ins Lächerliche ziehen wollten.

2.3.2.5 Anwendung 10, 11 und 12: „Labyrinth-Spiele“

Abb. 14 – 16: „Labyrinth-Spiele“

Die Funktionalität dieser drei Anwendungen ist der von Anwendung 6 „Farben-Labyrinth“ nachempfunden. Auch wenn sich die SchülerInnen, welche diese Spiele erstellt haben, kein gänzlich eigenes Konzept überlegt haben, so haben sie doch zumindest versucht, die Steuerung selbstständig zu implementieren. Es wurden auch alle Sprites, Hintergründe, Geräusche und Aufbau der Labyrinthe dem eigenen Geschmack angepasst.

2.4 Phase III

Anders als geplant beschränkte sich das Zusammentreffen der beiden Gruppen auf das gegenseitige Vorstellen der entwickelten Anwendungen. Ein Einarbeiten in das jeweils andere System hätte weitaus mehr Zeit in Anspruch genommen als zur Verfügung stand. Ganz wollte ich aber nicht auf einen Vergleich der Programmkonzepte verzichten, darum versuchte ich die wichtigsten Unterschiede von Scratch und Game Maker aufzuzeigen und eine Diskussion der Vor- und Nachteile der jeweiligen Konzepte in Gang zu setzten.

2.5 Nachbearbeitung

Einige Wochen nach dem Zusammentreffen der beiden Gruppen baute ich eine – nicht angekündigte - Nachhaltigkeitsüberprüfung in den Informatikunterricht ein. Die SchülerInnen der Gruppe Scratch mussten ein von mir erstelltes Programm nur anhand der fertigen Anwendung nachbauen. Die SchülerInnen der Gruppe Game Maker bekamen ein Programm vorgegeben und mussten es nach entsprechenden Anweisungen umbauen.

Diese Aufgabenstellungen erfüllten die SchülerInnen beider Gruppen sehr gut und ich gehe davon aus, dass doch einige von den behandelten Programmstrukturen als dauerhaftes Wissen bei den SchülerInnen verfügbar ist.

Im Zuge dieser Einheit teilte ich auch einen kurzen Fragebogen zur Meinung der SchülerInnen zum Thema Softwareentwicklung aus. (Siehe Anhang)

2.6 Projektergebnisse

Es lässt sich sagen, dass der Einstieg mit Scratch deutlich einfacher gelingt. Innerhalb von Minuten hat man eine sich bewegende Figur auf dem Bildschirm, was für die meisten SchülerInnen gut als Motivation taugt. Die ersten Schritte sind mit dem Game Maker etwas aufwändiger und man sieht auch nicht so schnell ein Ergebnis, das wirkt auf die SchülerInnen, die noch kein fertiges Konzept im Kopf haben, eher demotivierend.

Programmstrukturen wie Schleifen, Verzweigungen oder Variablen sind in Game Maker hinter der grafischen Benutzeroberfläche versteckt. Bei Scratch sind diese Programmstrukturen unmittelbar zu sehen, das macht es natürlich einfacher diese Programmstrukturen im Unterricht zu thematisieren und darauf einzugehen. Überhaupt sind kleine Programme meiner Meinung nach in Scratch einfa-

cher zu realisieren, dafür sind kompliziertere Programmideen oft nur schwierig oder gar nicht realisierbar. So ist es beispielsweise mit Scratch nicht möglich, Instanzen eines Objekts während der Laufzeit des Programms zu erzeugen und das schränkt schon die einfachsten Ideen von ProgrammieranfängerInnen erheblich ein.

Durch den relativ komplexen Aufbau der Benutzeroberfläche des Game Maker ist es nach meiner Meinung ratsam, dass die SchülerInnen die ersten Anwendungen mit Hilfe einer ausführlichen Schritt-für-Schritt-Anleitung erstellen. Das kann sowohl schriftlich aber auch durch Vorzeigen erfolgen. Mit Scratch ist es durch die sichtbaren Programmskripte für die SchülerInnen etwas einfacher ein vorgegebenes Programm nachzubauen und nach den eigenen Wünschen umzugestalten.

Ein weiterer Vorteil von Scratch ist der, dass es auch für nicht so versierte SchülerInnen ein leichtes ist, sich eine Geschichte auszudenken und diese dann, sozusagen als Filmsequenz, mit einer Programmabfolge nachzubauen. Das wäre natürlich auch mit dem Game Maker möglich, aber da kommt man nicht so leicht auf diese Idee.

Bei der abschließenden Diskussion über die Vor- und Nachteile der beiden Programmkonzepte gab es sehr unterschiedliche Meinungen der SchülerInnen. Als erste Reaktion konnte ich Ablehnung gegenüber dem jeweils anderen Konzept erkennen. Später favorisierten jedoch auch viele aus der Game-Maker-Gruppe das Scratch-Konzept. Von der Scratch-Gruppe konnte sich allerdings niemand mit dem Konzept des Game Maker anfreunden, auch nicht als ich versucht habe explizit die Vorteile dieser Software herauszustreichen. Das kann aber meiner Meinung nach auch daran liegen, weil niemand aus dieser Gruppe ernsthaft mit diesem Programm hat arbeiten müssen und es auf den ersten Blick mit den vielen Elementen und Unterelementen unübersichtlicher als Scratch ist. Der Schüler, der sich auch zuhause mit dem Game Maker beschäftigt hat, erkannte jedoch sofort die Nachteile von Scratch und sagte, dass Scratch keine ernsthafte Alternative sei, um Spiele zu programmieren.

3 EVALUATION

3.1 Ziele auf SchülerInnenebene

3.1.1 Teilziel 1: Spielerisches Kennenlernen von Programmstrukturen

Ein formuliertes Teilziel war, dass die SchülerInnen nach dem Projekt die wichtigsten Strukturen eines Programms, also Schleifen, Wenn-Dann-Abfragen, Variablen, etc. erkennen bzw. selbst anwenden können sollten. Dieses Ziel habe ich formuliert, bevor ich mich tiefer in das Programm Game Maker eingearbeitet habe. Hätte ich gewusst, dass diese Strukturen beim Game Maker sehr versteckt sind, dann hätte ich dieses Ziel nicht verfolgt, bzw. ein alternatives Ziel erarbeitet.

Ich denke, dass die SchülerInnen der Scratch-Gruppe diese Strukturen mit Hilfe von Scratch ausreichend kennengelernt haben. Die SchülerInnen der Game-Maker-Gruppe haben diese Strukturen jedoch so gut wie gar nicht im Zuge des Projekts kennengelernt. Andererseits konnte ich mit dem Game Maker sehr gut das Konzept der Objektorientierung vermitteln. Denn jede erzeugte Spielfigur ist eine Instanz eines Objekts mit Eigenschaften und gewissen Aktionsmethoden. Alle SchülerInnen der Gruppe verstanden auf Anhieb und ganz intuitiv, was es mit einem Objekt auf sich hat. Gleich bei der ersten Anwendung erzeugten wir zur Laufzeit des Programms mehrere Objektinstanzen. Das ist ja mit Scratch nicht möglich. Der Game Maker bietet zusätzlich zur grafischen Benutzeroberfläche eine Programmiersprache an – die GML (Game Maker Language), damit wäre es natürlich auch möglich Programmstrukturen sichtbar zu machen. Aus zeitlichen Gründen bin ich im Unterricht aber nicht näher auf diese Funktionalität eingegangen.

3.1.2 Teilziel 2: Motivation zum Anwenden algorithmischer Problemlösungsstrategien

Anhand der erzielten Ergebnisse der SchülerInnen erkenne ich durchaus eine gewisse Bereitschaft sich informatischen Problemen mit algorithmischen Strategien zu nähern. Zugegeben, diese Bereitschaft ist hier auch durch den Spieltrieb der SchülerInnen begründet, aber ich meine, sie ist jedenfalls da und wird bei Bedarf instinktiv eingesetzt. Ob ein Unterrichtskonzept mit Scratch oder Game Maker mehr zur Anwendung algorithmischer Problemlösungsstrategien motiviert als ein Unterricht mit Hilfe einer „klassischen“ Programmiersprachenumgebung, müsste noch weiter erforscht werden.

3.1.3 Teilziel 3: Softwareentwicklung als kreativen schöpferischen Prozess erkennen

Zur Überprüfung dieses Ziels diente mir die Auswertung des Fragebogens, welchen ich in der letzten Scratch- bzw. Game-Maker-Einheit ausgeteilt habe. Es freut mich, dass über 70% der SchülerInnen beider Gruppen die Softwareentwicklung als kreativen Prozess ansehen. Leider empfinden aber auch 60% Softwareentwicklung als langweilig und über 75% sagen, dass das nichts für sie wäre. Weiters findet sie weniger als die Hälfte interessant und über 83% würden der Aussage „Softwareentwicklung ist ein langwieriger Vorgang“ zustimmen.

Auffällig sind auch die unterschiedlichen Antworten der Gruppen zur Aussage „Softwareentwicklung ist nur für professionelle Programmierer“. Mehr als die Hälfte der Scratch-Gruppe stimmt dem zu, dagegen lehnt das fast 70% der Game-Maker-Gruppe ab.

Leider waren an dem Tag, an dem ich die Fragebogen ausgegeben habe, nicht alle SchülerInnen anwesend. Insgesamt haben vier SchülerInnen keinen Fragebogen ausgefüllt, davon drei Burschen. Dadurch wurde der ohnehin schon geringe Anteil männlicher Teilnehmer (7 Burschen, 22 Mädchen) beim Fragebogen noch weiter dezimiert. Durch diesen Umstand eignen sich die gewonnenen Ergebnisse nicht für eine seriöse Gegenüberstellung der Geschlechter. (Siehe auch Kapitel 4)

5D (Scratch-Gruppe)

Softwareentwicklung ist ...		zustimmend	ablehnend
	ein kreativer Prozess	66,7%	33,3%
	eine spannende Aufgabe	16,7%	83,3%
	langweilig	58,3%	41,7%
	nichts für mich	91,7%	8,3%
	ein langwieriger Vorgang	83,3%	16,7%
	interessant	41,7%	58,3%
	nur für professionelle Programmierer	58,3%	41,7%

5B (Game-Maker-Gruppe)

Softwareentwicklung ist ...		zustimmend	ablehnend
	ein kreativer Prozess	76,9%	23,1%
	eine spannende Aufgabe	30,8%	69,2%
	langweilig	61,5%	38,5%
	nichts für mich	61,5%	38,5%
	ein langwieriger Vorgang	84,6%	15,4%
	interessant	46,2%	53,8%
	nur für professionelle Programmierer	30,8%	69,2%

Gesamt

Softwareentwicklung ist ...		zustimmend	ablehnend
	ein kreativer Prozess	72,0%	28,0%
	eine spannende Aufgabe	24,0%	76,0%
	langweilig	60,0%	40,0%
	nichts für mich	76,0%	24,0%
	ein langwieriger Vorgang	84,0%	16,0%
	interessant	44,0%	56,0%
	nur für professionelle Programmierer	44,0%	56,0%

3.2 Ziele auf LehrerInnenebene

3.2.1 Teilziel 1

Beide Programme haben Vor- und Nachteile. Die wichtigsten davon sind nach meiner Einschätzung:

	Stärken	Schwächen
Scratch	• Einfacher, logischer Programmaufbau • Schnell sichtbare Ergebnisse • Programmstrukturen sind direkt „sichtbar“ • Steile Lernkurve • Komplette Menüführung in deutscher Sprache	• Limitierter Funktionsumfang (man stößt schnell an Grenzen) • Keine Objekte, von denen Instanzen während der Laufzeit erzeugt werden können • Anwendungen sind nur innerhalb von Scratch lauffähig • Die Auflösung des „Fensters“ auf dem die Anwendung läuft, ist in der Größe fixiert • Ein „scrolling“ des Hintergrunds ist nicht möglich
Game Maker	• Entsprechend gespeicherte Anwendungen sind unabhängig vom Game Maker lauffähig • Integrierte Skriptsprache, um Programme zu erweitern • Es können beliebig viele Instanzen eines Objekts erzeugt werden • Die Größe des Fensters, in dem die Anwendung läuft, kann frei bestimmt werden. • Der Hintergrund der Anwendung kann größer sein als die Größe des Fensters. („scrolling“ ist möglich)	• Durch den größeren Funktionsumfang ist es für Anfänger schwierig sich zurechtzufinden. • Man muss zuerst viele Einstellungen machen und Objekte anlegen bis sich das erste Mal etwas am Bildschirm bewegt • Alle Menüs und auch die Hilfefunktionen sind in englischer Sprache

Will man Programmstrukturen, wie Schleifen, Verzweigungen, Variablen oder ähnliches, möglichst einfach für die SchülerInnen ersichtlich machen, ist meiner Meinung nach Scratch bedeutend besser geeignet. Auch die relativ kurze Einarbeitungszeit spricht eher für einen Einsatz von Scratch im Informatikunterricht.

Demgegenüber hat man mit dem Game Maker eine richtig gute grafische Oberfläche zum einfachen Erstellen von Spielen oder spielerischen Anwendungen. Außerdem gibt es mit der eingebauten Skriptsprache GML vielfältige Möglichkeiten die Anwendungen zu erweitern und auch Programmieren zu lernen.

3.2.2 Teilziel 2

Alle SchülerInnen haben sich damit einverstanden erklärt, dass ihre erstellten Programme von interessierten LehrerInnen im Informatikunterricht verwendet werden.

Und auch ich gebe die verwendeten Aufgabenstellungen und Materialien gerne weiter. Unter folgendem Link findet man die verwendeten Materialien und die erstellten Scratch und Game-Maker-Anwendungen:

http://www.brg-viktring.at/fachgruppen/imst.php

4 GENDERASPEKT

Sowohl in der Scratch-Gruppe als auch in der Game-Maker-Gruppe waren die Burschen deutlich in der Minderheit (2 von 14 bzw. 5 von 15). Trotzdem konnte ich einige wenige geschlechterspezifische Unterschiede beobachten. So waren einige der erstellten Anwendungen geradezu klischeehaft einem Geschlecht zuzuordnen. Allerdings ließ sich das einzig und allein an den verwendeten Farben und Sprites festmachen, bei der Komplexität der Anwendung und beim Aufbau des Programms konnte ich keine Unterschiede ausmachen. Weiters gab es einen Burschen, der sich angeregt durch den Informatikunterricht auch zuhause mit der Programmierung von spielerischen Anwendungen beschäftigte. Erfahrungsgemäß sind es eher (nach meiner bisherigen Beobachtung sogar „immer“) Burschen, die sich auch über den Unterricht hinaus mit informatischen Inhalten beschäftigen. Natürlich könnte das auch an meiner Unterrichtsgestaltung liegen; ich versuche bei den Inhalten und Methoden möglichst beide Geschlechter anzusprechen, aber eventuell wird das von Schülerinnen oder Schülern doch anders gesehen.

Neben diesen Unterschieden gab es aber auch viele Aspekte, die beim Erstellen einer Anwendung oder beim Ergebnis keine geschlechterspezifische Unterscheidung zulassen. So habe ich beispielsweise nicht beobachtet, dass eine Geschlechtergruppe mit größerer Motivation an informatische Problemstellungen herangeht, bessere Lösungen findet oder beim Erlernen von Programmstrukturen schneller ist.

Da die Anwendungen meist zu zweit erstellt wurden, gab es natürlich auch Teams, die aus einem Mädchen und einem Burschen bestanden. In der Scratch-Gruppe gab es kein rein männliches Programmierteam (zwei gemischte Teams, sonst weibliche Zweierteams oder einzelne Programmiererinnen). In der Game-Maker-Gruppe gab es ein gemischtes Team, ein männliches Dreierteam, einen einzelnen Programmierer und sonst nur weibliche Zweierteams.

Durch den geringen Anteil an männlichen Teilnehmern ist die Auswertung des Fragebogens nach meiner Ansicht nicht geeignet eventuelle geschlechterspezifischen Sichtweisen herauszuarbeiten.

5 LITERATUR

BMUKK (2004). Lehrplan Informatik. Online unter
http://www.bmukk.gv.at/medienpool/11866/lp_neu_ahs_14.pdf [25.02.2011]