

R.I.S. Journal

Remix Improve Share - das freie, creative-commons lizenzierte Journal zu den Themen Open-Source, Spiele Programmierung, Open Education. **Preis:** Zahlen Sie so viel Sie wollen (siehe Editorial).



Inhalt

Impressum: ⇒impressum, S. 2

Editorial: Horst Jens (Herausgeber) über den Zweck dieser Zeitschrift, Creative-Commons Lizenzen und Free Software culture: ⇒editorial, S. 3

Kalender: Termine für 2014: ⇒kalender, S. 60

Sponsorenhinweise: siehe Impressum und Kalender

Report

Linuxtage 2013 in Chemnitz, Vortrag über Linux in Schulen, OSDomotics Smarthome: ⇒chemnitz, S. 6

Lifestyle

Freiwillig obdachlos: ⇒nomad, S. 9

Billionär werden: ⇒trillions, S. 12

Beruf

Berufsbild: Spiele **Tester:** ⇒spieletester, S. 14

Firmengründer Stefan Reinalter

erzählt über seine Spielegrafik-Engine: ⇒molecularmatters, S. 17

Lebensweg: Wie ein Österreicher seinen Weg in die games industry fand: ⇒austrianguy, S. 19

Berufsbild: **Counterstrike-Profispieler:** ⇒counterstrike, S. 22

Warnung: Meide die Games Industry ! (Rant auf reddit.com): ⇒redditrant, S. 25

Schule

Meinung: Wie smart ist die Samsungs Smartschool ? ⇒smartschool, S. 28

Kanada: c3inspire bringt Studierende von Technik und Wirtschaft zusammen: ⇒c3inspire, S. 29

England: Umstellung einer Mädchenschule auf Linux: ⇒westcliff, S. 32

USA / Vietnam: Neil Fraser macht in Vietnam Urlaub, erforscht den Computer Science Unterricht vor Ort und programmiert ein grafisches Lern-tool: ⇒vietnam, S. 36

OLPC Austria: Christoph Derndorfer macht sich Gedanken über das **One Laptop per Child** Projekt: ⇒olpcnewyear, S. 38

Mädchen programmieren

Pyladies Vienna Gründerin Floor Drees im Interview ⇒floor, S. 40

Railsgirls haben mein Leben verändert: Blogposting von Laura Gaetano über Railsgirls, Pyladies und wie man Schriftstellerin wird: ⇒laura, S. 43

Praxis

Tutorial: Freies Software-Repository für **fdroid** für Android installieren: ⇒fdroid, S. 45

Hardware: Tutorial: **Zahnbürsten-Roboter** selber machen: ⇒bristlebot, S. 46

Game

Open Source game: Chemiebaukasten: Welten zerstören mit **The Powder Toy**: ⇒powdertoy, S. 48

Tutorial: Wasser bewegen mit The Powder Toy: ⇒powdertoytutorial, S. 53

Tutorial: Ballspiel grafisch programmieren mit **Scratch2**: ⇒scratch, S. 54

Impressum

Informationspflicht lt. §5 E-Commerce Gesetz, §14 Unternehmensgesetzbuch bzw. §63 Gewerbeordnung und Offenlegungspflicht lt. §25 Mediengesetz, Auszug aus dem Firmen A-Z der Wirtschaftskammer Österreich

Firmenname: **Horst JENS**

Unternehmensbezeichnung:

spielend-programmieren

Adresse: Brigittenauerlande 6, 1020 Wien, Austria

Telefon: +43 / (0)660 52 65 377

Fax: +43 (0)1 / 33 285 30 14

Web: **spielend-programmieren.at**

E-Mail:

horst.jens@spielend-programmieren.at

UID-Nummer: keine

Behörde gem. ECG (E-Commerce Gesetz): Magistratisches Bezirksamt des II. Bezirkes

Anwendbare

Rechtsvorschriften:

www.ris2.bka.gv.at

Rechtsform: **Einzelunternehmer**

Bankverbindung:

BIC OPSKATWW

IBAN AT696000000075159553

Offenlegung gemäß §25 Mediengesetz: Firmensitz: **Wien**. Unternehmensgegenstand:

EDV-Training, Programmierung

Mitglied der Wirtschaftskammer Wien, Fachgruppe: **IT-Dienstleistung**.

Blattlinie



Das RIS-Journal ist eine hoffentlich bald regelmäßig im Internet erscheinende, creativ-commons lizenzierte (cc-by-sa) Zeitschrift zum selber ausdrucken. Daseinszweck ist die Bewerbung von Dienstleistungen (Programmierungskursen) der Firma **spielend-programmieren** (Horst JENS) sowie der Propagierung und Bekanntmachungen von frei lizenzierten (cc-by-sa) Inhalten die sich mit den Themen der freien Software Kultur befassen, insbesondere in Schulen.

Amazon-Partnerprogramm

Horst JENS ist Teilnehmer des Partnerprogramms von Amazon EU, das zur Bereitstellung eines Mediums für Websites konzipiert wurde, mittels dessen durch die Platzierung von Werbeanzeigen und Links zu Amazon.de Werbekostenerstattung verdient werden kann.

Sponsoren

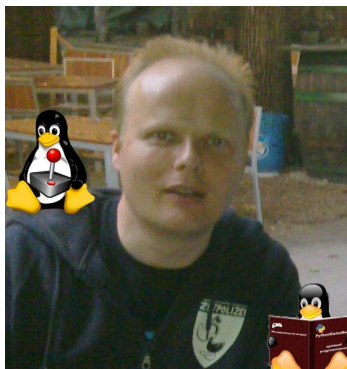
Das RIS-Journal dankt folgenden Sponsoren dieser Ausgabe:



Wukonig.com, die Internetagentur aus Österreich

Editorial

von HORST JENS



Horst JENS [1]

Wenn Sie dieses Editorial zu Ende lesen verrate ich [1] Ihnen wo sich die Werbung im *R.I.S.-Journal* versteckt, warum Sie zu viel dafür gezahlt haben, wofür der Name *R.I.S.* steht und was es mit den *Creative-Commons Lizenzen* auf sich hat. Und wie Sie mich dazu bringen an in Ihrer Schule einen *Game-Programming Workshop* zu veranstalten.

Wie alles begann...

Genau vor einem Jahr, zum Jahreswechsel 2012 auf 2013, hatte ich die Idee um mein Geschäft (Programmierkurse für Jugendliche)¹ anzukurbeln anstatt der üblichen Plakate, Flugzettel und Inserate gleich eine eigene Zeitschrift herauszubringen um sie an Schulen als Werbemittel zu verteilen. Genialerweise ist es teurer ein einziges Inserat in einer uninteressanten Zeitung zu bezahlen als eine interessante Zeitung ohne uninteressante Inserate zu erstellen. Weiter unten erkläre ich wie Sie das *R.I.S. Journal* für Ihre eigenen Zwecke nutzen können.

Diverse Projekte verzögerten mein Vorhaben bis es endlich zum Jahreswechsel 2013 / 2014 so weit war: Ich hatte genug Zeit um die

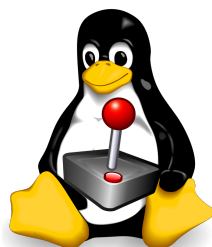
bisher gesammelten Artikel noch einmal durchzuarbeiten, die Layoutsprache *TEX* zu lernen und sehr viele E-Mails zu schreiben um weltweit nach Übersetzungs- und Abdruck-Genehmigungen zu fragen.

Lausch- und Lesefutter

Die meisten der Artikel dieser Ausgabe des *RIS-Journals* habe ich vom Englischen ins Deutsche übersetzt. Viel weniger Artikel als ursprünglich geplant sind von mir selbst... freuen Sie sich auf die nächste Ausgabe des *RIS-Journals* dort werden mehr Artikel von mir drin sein. Werfen Sie bis dahin einen Blick auf die *Tutorials meiner Homepage* [2] oder in mein *Python Game Book* [3] bzw. hören Sie sich meinen wöchentlichen *Podcast* [4] an, dann wissen Sie recht gut welche Themen ich in den nächsten Ausgaben dieses Journals behandeln werde und über welche Themen ich an Ihrer Schule einen Workshop anbieten kann:

Meine Werbung...

Warum ich an Ihrer Schule zeigen will was ich kann: Einerseits weil es (selten, aber doch) vorkommt dass Schulen mich direkt für einen Workshop bezahlen; vor allem aber weil ich dadurch Kunden für meine Programmierkurse [1] gewinnen kann. Bei *spielend-programmieren* gibt es für Jugendliche jede Altersstufe Programmierkurse, sowohl während der Schulzeit als auch während der Ferien. Näheres erfahren Sie auf *meiner Homepage* [1]!



Logo der Firma *spielend-programmieren*: Tux mit Joystick

...an Ihrer Schule

Womit wir beim Sinn dieses Journals angelangt sind: Ich kann gerne in Ihrer Schule / für Ihre Organisation einen Workshop (bzw. eine Probestunde) zum Thema *Open Source / Programmieren / Game Programming* abhalten, kombinierbar mit beinahe jedem Unterrichtsfach² - kontaktieren Sie mich einfach per Email, Kontaktmöglichkeiten und Einzelheiten finden Sie auf meiner meiner Homepage *spielend-programmieren.at*.

Frei lizenziert

Alle (übersetzten) Texte im *RIS Journal* sind frei lizenziert und auch fast alle Grafiken³. Da (zumindest die Texte) alle unter eine **Creative Commons cc-by-sa Lizenz** fallen können Sie diese Texte nicht nur im Unterricht beliebig (legal!) verteilen und verwenden sondern daraus auch ihre eigene Zeitschrift machen...mit Ihrer Werbung und mir Ihren eigenen, zusätzlichen Inhalten. Unter Beachtung folgender Bedingung:

- Namensnennung (recognition): Sie müssen auf die Originalquelle verlinken / hinweisen.
- Weitergabe unter gleichen Bedingungen (share-alike): Die daraus entstehenden Artikeln oder Werke müssen ebenfalls cc-by-sa lizenziert sein und frei kopiert werden können.

Was Sie *nicht* können ist eine **cc-by-sa** Lizenz rückgängig zu machen: meine Artikel bleiben frei lizenziert. Sie können trotzdem für Ihre Zeitschrift Geld verlangen, Sie können aber nicht verhindern dass die freien Artikeln kopiert werden. Sie dürfen keinen Kopierschutz oder Copyright-Vermerk zu Inhalten des *RIS Journals* hinzufügen, davor schützt die **Creative-Commons Lizenz**. Womit wir beim Preis angelangt sind:

¹Open Source Game Programming - Kurse für Kinder, Jugendliche und Schulen: Firma *spielend-programmieren* [1]

²Sogar Turnstunden, und sogar ohne Computer. Am Liebsten sind mir Schulen mit gut bestücktem EDV-Saal

³bitte beachten Sie im Einzelfall die Bildunterschriften

Preis

Das RIS Journal wird immer kostenlos erhältlich sein (z.B. indem Sie die Online-Version herunterladen, ausdrucken und verteilen). Wo bei es vollkommen in Ordnung ist das R.I.S.-Journal zu verkaufen: kostenlos ist ja nicht das selbe wie wertlos, und die **cc-by-sa Lizenz** erlaubt auch den Verkauf ausdrücklich. Sie können dieses R.I.S.-Journal nehmen und jemand anderem verkaufen, ich freue mich sogar darüber! Bitte schreiben Sie mir welchen Preis Sie erzielt haben.

R.I.S. Journal ?

Im Name Journal kommt steckt das Wort 'jour' (französisch für Tag), gemeint ist eine tägliche bzw. tagesaktuelle Zeitschrift. Sie halten Ausgabe 001 in der Hand und ich habe über ein Jahr dafür gebraucht... realistischweise habe ich vor etwa vier Ausgaben pro Jahr herausbringen, später vielleicht mehr. R.I.S. steht für **Remix - Improve - Share** und fasst recht gut das Gedankengebäude der freien Software / Open Source Welt zusammen, nämlich die vier Grundfreiheiten der **gpl-Lizenz**: Die Freiheiten ein Werk (z.B. Software)

- zu nutzen (**use**, remix)
- zu verbessern (**modify**, improve)
- zu verbreiten (**share**)
- zu studieren (**study**) (davon zu lernen, Zugriff auf die Quelldateien /Baupläne zu haben)

Diese vier Grundfreiheiten bilden das Fundament der Welt der freien Software (Free/libre Open Source Software) und führen zu einer Kultur (**Hackerethik**) des miteinander Teilens (von Wissen), des weltweiten Zusammenarbeitens⁴. Diese Kultur ist der Grund warum heute technisches Spitzenprodukte frei erhältlich sind wie etwa das Betriebssystem **Gnu/Linux**, die Grafiksoftware **Gimp**

oder die 3D-Software **Blender**. Diese Kultur ist der Grund warum tausende Freiwillige die größte jemals verfügbare, creativ-commons lizenzierte Enzyklopädie erschaffen haben (**Wikipedia**) an der jeder mitarbeiten kann. Diese Kultur ist der Grund warum sich weltweit Leute zusammentun um ein freies Hochglanzmagazine wie *Gimpmagazine.org* zu erschaffen oder gemeinsam an einem freien Film wie *Sintel* [5] zu arbeiten.



Szene aus dem Blender-Film *Sintel*.
Bildrechte: [5]

Diese Kultur bleibt nicht auf immaterielle Güter wie Software beschränkt sondern betrifft z.B. dank austauschbarer Pläne für 3D Drucker mehr und mehr Bereiche außerhalb der klassischen Software-Industrie. Werfen Sie einmal einen Blick auf den Blog von **Open Source Ecology**: erdverbundener geht es nicht mehr.

Mitmachen

Diese freien Inhalte und die damit verbundene Kultur bekannt zu machen und zu fördern, speziell in Schulen, ist mein Ziel für das R.I.S. Journal. Ich wünsche viel Spaß beim Lesen. Bitte schreiben Sie mir was Ihnen gefallen hat, und vor allem: Wenn Ihnen ein Artikel nicht gefallen hat, schreiben Sie mir einen besseren Artikel für die nächste Ausgabe! Wie Sie sehen können brauche ich auch dringend Grafiker, Übersetzer, Redakteure, Korrekturleser und Leute die sich mit **LaTeX** auskennen.

Spenden

Auch wenn Sie das RIS Journal kostenlos downloaden, ausdrucken und verbreiten dürfen bedeutet dies keinesfalls dass ich mich gegen Spenden wehre. Wenn Sie mir absolut keinen Auftrag für einen Programmierkurs geben wollen sondern mir lieber einfach so Geld zukommen lassen wollen, ganz ohne Gegenleistung:

Banküberweisung: Bitte schreiben Sie in den Buchungstext 'Spende für RIS-Journal 001': Kontoinhaber: Horst JENS

IBAN: AT69 6000 0000 7515 9553

BIC: OPSKATWW

Selbstverständlich freue ich mich auch über **Bitcoins**:

1CZ4 dCgF 8sMX Zmbd wjiR 9nPM

52b4 QSxa nb Sowie über **Flattr** -

Spenden: <http://goo.gl/5nJz6y>



Bitcoin-Adresse (links) und Flattr Adresse (rechts) von Horst JENS

Gliederung

Bis ich eine bessere Layout Idee habe ist jeder Artikel folgendermaßen aufgebaut:

- Anrisstext, **fett gedruckt** wo kurz erklärt wird worum es in dem Artikel geht damit Sie entscheiden können ob Sie ihn lesen wollen oder nicht.
- Der Artikeltext selbst, inklusive Bildern, →**fett geschriebenen Fachbegriffen mit Pfeil** und *kursiven Links zu Web-Adressen (URL)* [1], mit Zahlen in eckigen Klammern dahinter (siehe Quellen) falls die URL nicht selbst-erklärend ist. In der Online-Version ist die URL direkt anklickbar.
- Fachbegriffe (sofern vorhanden) einzeln erklärt. Falls

⁴die Open Source Welt kann mehr Programmierer auf eine Aufgabe konzentrieren als jede einzelne Firma

die Website zum Fachbegriff nicht dem gleichlautendem Wikipedia-Artikel entspricht wird sie extra angegeben.

- **Download & Feedback:** Link zum Direkt-Download des Artikels, falls vorhanden Link zur Audio-Datei (**Podcast**). Falls vorhanden E-Mail Adresse der Autoren.
- **Lizenzangabe** diese Lizenzangabe bezieht sich immer auf den (in Deutsche übersetzten) Text. Bilder und Originaltexte können abweichende Lizensierungen haben !
- **Quellen:** Alle Links und Quellenangaben dieses Artikels

Fachbegriffe:

cc-by-sa creative-commons, recognition, share-alike Lizenz: Namensnennung, Weitergabe unter gleichen Bedingungen. Siehe auch weiter unten

gpl General Public License, ermöglicht die 4 Grundfreiheiten *use, study, share, improve* im Bereich Software. Sie wurde von Richard Stallman erfunden und wird weltweit von der Free Software Foundation (Europa: fsfe.org) gefördert und durchgesetzt. Für Dokumentationen, Text, Musik und künstlerische Werke hat sich die Creative Commons Lizenz cc-by-sa bewährt.

Hackerethik Laut Steven Levy's Buch *Hackers. 25th Anniversary Edition: Heroes of the Computer Revolution*



Bildrechte: Amazon Partner Programm [8]

geht es bei der Hackerethik um folgende Werte:

- **Free access:** Zugang zu Computern soll frei sein
- **Sharing:** Information soll frei sein
- **Decentralization:** Misstrauen Autorität (Behörden) - fördere Dezentralisierung
- **Openness:** Beurteile Hacker nach deren Taten, nicht nach deren Zeugnissen
- **World Improvement:** Du kannst Kunst und Schönheit mit dem Computer erschaffen
- **World Improvement:** Computer können Dein Leben verbessern

Gnu/Linux auch einfach nur Linux genannt, zum großen Leidwesen von Richard Stallman. Ein freies Betriebssystem für Computer. Steckt u.a. in Ihrem Android-Smartphone. Das Maskottchen von Linux, den Pinguin Tux, haben Sie sicher schon gesehen.

Podcast Eine Radiosendung (cast) welche als mp3 Datei via Internet gesendet wird und per Computer oder mp3-Player (iPod) angehört wird.

Wikipedia Lesen Sie einmal die Lizenzinformation eines beliebigen Wikipedia-Artikels !

Open Source Ecology [6]: Eine Gruppe von open-source begeisterten Landfreake die an einem *Global Village Construction Set* werkeln: vom



Ziegelschneider über den Traktor bis zum Fertighaus werden alle Pläne und Erfahrungen über das Internet verteilt, frei lizenziert und weltweit nachgebaut und verbessert.

Flattr eine Micro-payment Dienst der es erlaubt sehr kleine Beträge zu spenden (z.B. einen Zwanzigstel von einem Cent). Auf <http://flattr.com> erfahren Sie mehr darüber.

Bitcoins Bitcoins sind elektronisches Geld welches sich teils ohne Überweisungsgebühr übertragen lässt, in beliebig kleinen Beträgen. Näheres erfahren Sie jede Woche im *Bitcoin-Update* [7] **Podcast**.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/editorial
Feedback ✉ horst.jens@spielend-programmieren.at

Lizenz, Quellen



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] Spielend-Programmieren.at
- [2] <http://goo.gl/0X37fd>
- [3] ThePythonGameBook.com
- [4] Biertaucher.at
- [5] Sintel.org
- [6] OpenSourceEcology.org
- [7] BitcoinUpdate.com
- [8] goo.gl/EAk5Q

Chemnitzer Linuxtage 2013

von HORST JENS

Horst JENS [1] berichtet von den Chemnitzer Linuxtagen 2013 [2], dem erfreulichen Zusammentreffen mit leibhaftigen Open-Source Programmierern, einem Vortrag von Kurt Gramlich [3] zum Thema Linux in Schulen [4] und dem OSDomotics [5] Messestand zum Thema Smart Home

Ein Reisebericht

Der jährliche stattfindende *Chemnitzer Linuxtag* [2] vereinigt alles was in der deutschsprachigen Open Source Welt Rang und Namen hat. Ein Wochenende lang stellen über 400 Aussteller Ihre Open Source Projekte bzw. Firmen vor, es gibt Vorträge und Workshops, Dr.Tux hilft bei Computerproblemen, Kinder basteln mit dem Raspberry Pi und löten Roboter; für den reibungslosen Ablauf sorgt ein Heer engagierter Freiwilliger HelferInnen.



Horst JENS und Florian Schweikert.
Bildrechte: [1], cc-by-sa

Für mich war es *nach 2009* [6] der zweite Besuch auf einem Chemnitzer Linuxtag, dementsprechend erwartungsvoll fuhr ich hin. Meine Reisegruppe bestand aus acht Leuten vom Verein *OSDomotics* [5]. Nach einer langen Zugfahrt (Wien - Nürnberg - Chemnitz) und Ankunft in einem Wolkenkratzer-Hotel (W-Lan im Zimmer nur gegen Aufpreis) machten wir uns auf den Weg zum Gelände der technischen Universität wo der Chemnitzer Linuxtag

(Hashtag #CLT2013) jährlich statt findet.

Der Freitag Abend ist für das Aufbauen und Verkabeln des Messestandes vorgesehen - für das Publikum öffnet sich die Messe erst am Samstag Früh. Die erste schöne Überraschung erlebten wir gleich nach der Ankunft: Es gab erfreulich wenig zu tun, die Organisatoren hatten ganze Arbeit geleistet: Das Messestandplakat war schon ausgedruckt und aufgeklebt, der Tisch war schon hergerichtet (inkl. Stromkabeln und Sesseln), Ausstellerausweise mit Essensgutscheinen und Workshop / Vortragsverzeichnis lagen schon bereit, und das W-Lan funktionierte einwandfrei.

Bei andere Messen (nicht nur Linuxtage, sondern auch kommerziellen Messen) ist eine so gute Organisation keineswegs selbstverständlich.

Bier und Schokolade



Geany Entwickler bekommt Schokolade.
Bildrechte: [1], cc-by-sa

Nachdem Aufbau unserer Messeattraktion (ein Puppenhaus mit 6LoWPAN Funksensoren und Lichtern) schlenderte ich mit *Florian Schweikert* [7] (bekannt vom *Biertaucher-Podcast* [8]) durch die Messehalle um die anderen Standplakate anzuschauen. Zu unserer großen Freude entdeckten wir vom freien Programmiereditor **Geany** und vom Python Framework **Django** eigene Messestände. Spontan beschlossen wir am Samstag mit Bier und Schokolade bei den Projektbetreibern vorbei zu schauen.

Am Samstag (nach einem Spaziergang durch das eher menschenleere Chemnitz) kam unsere Geschenkkaktion sehr gut an. Geany wird

von einem über die ganze Welt verstreuten Team von Open Source Programmierern (unentgeltlich) entwickelt, und die beiden anwesenden Entwickler aus Deutschland waren sichtlich erfreut über ein Stück handfeste Dankbarkeit. Die Geany Homepage hat mittlerweile zusätzlich zum *PayPal Link* auch einen **Flattr** Spendenbutton.



Geany der Flaschengeist. Bildrechte: [1], cc-by-sa

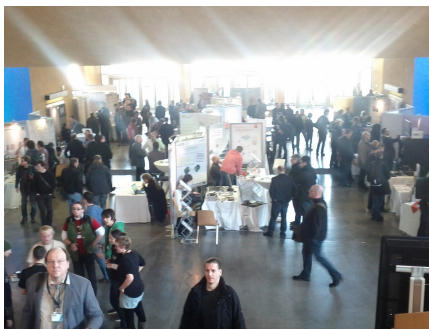
Ebenso freute sich der anwesende Django Entwickler, der stellvertretend für die in der *Django-Foundation* organisierten Programmierer das Bier annahm und versprach es so gerecht wie möglich zu verteilen. Uns machte das Schenken ebenfalls große Freude, vielleicht entsteht eine kleine nette Tradition daraus.

Da der OSDomotics Messestand mit acht Leuten sehr gut bemannt war hatte ich ein wenig Zeit mir am Samstag die anderen Messestände anzusehen und sogar zwei Vorträge zu besuchen. Weil ich nach Möglichkeit mit jedem Messestandpersonal (eigentlich gab es keine uninteressanten Messestände) kurz plaudern wollte verging der Samstag wie im Flug.

Am Abend eröffnete eine Trommlertruppe das große Festbuffet, spät am Abend fuhr die OSDomotics Gruppe gut gelaunt, aber müde zurück ins Hotel.



Unser Messestand. Rechts im Bild mit Fotoapparat: Harald Pichler, Verein OSDomotics. Bildrechte: [1], cc-by-sa



Besucher am Chemnitzer Linuxtage 2013. Bildrechte: [1], cc-by-sa

Top-down vs. Bottom-Up: Revolutionen (im Bildungsbereich) wirken nachhaltig von unten nach oben, nicht umgekehrt

Kulturelle Unterschiede: Es gibt große kulturelle Unterschiede zwischen Skandinavien und Deutschland. In Skandinavien gibt es z.B. keine Schul-Internetfilter, in Deutschland werden sie von Behörden und Eltern verlangt.

Learning by Doing: Die beste Art **SysAdmins** für ein SkoleLinuxSystem auszubilden ist es, engagierten Schülern so viel Verantwortung für die Schul-EDV wie möglich zu geben ('macht Euch das alles selbst aus'). Diese Schüler werden teils später erfolgreiche Admins.

Nehmen was da ist: Die Schule sollte laut Kurt Gramlich nicht versuchen mit Gewalt Technologien zu unterrichten welche an der Lebenswirklichkeit der Schüler vorbeigehen sondern die Technologie benutzen welche die Schüler sowieso besitzen (Smartphones, Tablets) und versuchen damit etwas Vernünftiges zu machen.

Open Source braucht Zeit: Linux / Open Source Neulingen sollte man als gutmeinender Open Source Evangelist nicht das Kostenargument (Linux kost' nix!) als *den* Vorteil freier Software 'verkaufen'. Auch freie Software kostet: Zeit, Nerven und Support. Speziell für Schulen welche Raubkopien verwenden oder Ihre Software geschenkt / vom Staat bezahlt bekommen ist das Kostenargument zweitrangig.

Der Geist von Open Source, - das Erkennen und Erleben was freie Software bedeutet- dauert laut Kurt Gramlich 2 bis 3 Jahre, erst dann versteht ein neuer User dass er Teil einer Gemeinschaft ist, etwas geschenkt bekommt (Support, Bier, Schokolade...) und sich kreativ und hilfreich in diese Gemeinschaft einbringen kann. In einem Beratungsgespräch oder mit einer Gratis-CD ist diese Erfahrung nicht zu vermitteln.

Kleinunternehmen

Am Sonntag war deutlich weniger Publikumsandrang als am Samstag, der OsDomotics Messestand war trotzdem immer gut besucht. Erst um 16:00 gelang es mir ein Foto vom Messestand ohne Besucher davor zu machen. Nur zwei Vorträge konnte ich vollständig besuchen: Einen Vortrag von zwei selbständigen Computer / Open Source Beratern welche dazu aufriefen kleine Unternehmen zu betreuen anstatt sich auf große Unternehmen zu konzentrieren. Die Zahlen für Deutschland, Österreich und Schweiz ähneln sich:

Circa 80 Prozent aller Unternehmen sind sogenannte *Kleinunternehmen* mit weniger als 10 Mitarbeitern. (Die Definition *Kleinunternehmen* ist von Land zu Land unterschiedlich). Und während diese Masse kleiner Unternehmen nur einen geringen Anteil am Gesamtumsatz hat -siehe **Paratoprinzip**-, so stellen sie doch eine große Masse an potentiellen Kunden. Meist mit knappem Budget und eigenwilligen Prioritäten (*Das Ding soll funktionieren sonst schreib' ich die Rechnungen mit der Schreibmaschine*).

Wie die beiden Vortragenden in einer Art Doppelconference darlegen haben kleine Unternehmen einen Vorteil den es bei großen Unternehmen nicht gibt: Aus erfolgreicher Arbeit bei kleinen Unternehmen entstehen oft persönliche Freundschaften.

Linux in Schulen

Der andere Vortrag den ich besucht habe war von **Kurt Gramlich** über den Einsatz (und das Scheitern) von **SkoleLinux** im deutschen Bundesland Rheinland-Pfalz. Kurt Gramlich ist ein sehr engagierter, weißhaariger Volkshochschul-Lehrer aus Nordreihn-Westfalen der sich über Jahre hinweg mit Linux in Schulen befasst hat. Als aus politischen Gründen das Bundesland Rheinland-Pfalz eine Microsoft-Alternative suchte und 'von oben' das Skolelinux-Projekt förderte war er stark darin involviert. Nachdem -aus politischen Gründen- das Bundesland Rheinland-Pfalz an Skolelinux nicht mehr interessiert war sprangen viele vorher beteiligte Schulen aus dem Projekt wieder ab.

Es gibt immer noch Schulen in Rheinland-Pfalz welche SkoleLinux verwenden, allerdings nicht mehr so viele wie zu Zeiten der offiziellen Projektförderung. Für mich waren vor allem folgende Erkenntnisse aus dem Vortrag interessant:



Kurt Gramlich. Bildrechte: [3]

Go East!

Neben der wirklich hervorragenden Organisation seitens der Chemnitzer Studierenden -es gab z.B. ein ständig gut gefülltes Buffet im Backstage Bereich für Standbetreuer- war ich angenehm überrascht von den Besuchern am **OSDomotics** Messestand. Viele Besucher hatten selber kleine Firmen oder waren Handwerker / Wohnungs- bzw. Haus-(um)bauer und hatten sofort Ideen was man mit einer Funkvernetzung von Sensoren und Aktoren machen könnte. Darunter waren Ideen (Gärtnereien, Alarmanlagen, Passivhäuser) auf welche wir OSDomotics Leute von selbst nicht gekommen wären.

Außerdem erfuhr ich dass viele Firmen in Sachsen (**Silicon Saxony**) Mitarbeiter suchen - wegen dem Bevölkerungsrückgang und dem Abwandern vieler junger Studienabgänger nach Westdeutschland suchen die Firmen teils von Prag bis Portugal nach qualifizierten IT-Mitarbeitern und Facharbeitern.



Trust is ...? Bildrechte: Chemnitzer Linuxtage 2014

Ich freue mich schon auf den nächsten Chemnitzer Linuxtag (15. und 16. März 2014, siehe RIS Journal 001, kalender, Seite page 60) und kann eine Reise dorthin jedem an Linux Interessiertem empfehlen.

Fachbegriffe:

OSDomotics: ist ein Verein der freie Software und freie Hardware im Bereich Haustechnik fördert. Der Verein entwickelt Sensoren (z.B. Temperaturfühler) und Steuerungen (z.B. eine Funksteckdose) welche drahtlos via **6LoWPAN**-Protokoll miteinander kommunizieren. Damit ist es z.B. möglich per Android Smartphone vom Urlaub aus die Heizung im

Haus einzuschalten oder die Feuchtigkeit zu überwachen.



opensource**domotics**
Bildrechte: osdomotics.com

Hashtag: Durch Voranstellen einer Raute (#, englisch: Hash) vor einen Begriff (englisch: Tag) kann dieser Begriff in Social Media Anwendungen leichter gefunden werden.

6LoWPAN besonders energiespezifischer Datenfunk-Standard für 'Das Internet der Dinge'

Geany sehr beliebter freier (free/libre open Source) Texteditor für verschiedenste Programmiersprachen. Die ~~TeX~~Files für diese Zeitschrift werden z.B. mit Geany bearbeitet. <http://geany.org>

Django: Ein Web-Entwicklungs-Framework. Ermöglicht die Programmierung komplexer Web-Projekte mit Datenbank-Anbindung (z.B. Webshops, Wikis etc) mit Hilfe der eleganten Programmiersprache Python. <https://www.djangoproject.com>

Flattr: *flattr.com* ist ein skandinavischer Spenden / Micropaying Dienst der vor allem in Europa bei der Podcast-Szene populär ist.

Paretoprinzip: oder die 80-20 Regel besagt dass oft 20 Prozent der Kunden/Pflanzen/Tätigkeiten 80 Prozent der Eträge bringen.

Skolelinux skolelinux.org ist ein aus Skandinavien kommendes Linux-Projekt welches auf den Einsatz in Schulen spezialisiert ist. Weitere Schul-Linuxe sind z.B. *Edubuntu* [9], *Desktop4Education* [10], *Linux Advanced* [11] (beide aus Österreich) bzw. das Schweizer Projekt *Lernstick* [12], welches gemeinsam mit Linux Advanced entwickelt wird.

System-Administratoren, Sysadmins sind die technischen Betreiber eines großen, vernetzten Com-

putersysteme.

Silicon Saxony: eine Ansammlung von IT-Firmen aus den Bereichen Mikroelektronik und Photovoltaik, hauptsächlich im Raum Dresden, mit Ihren Zulieferern und dazugehörigen Firmen.

Podcast



Einen ausführlichen Audio-Bericht von Horst JENS und Florian Schweikert über den Chemnitzer Linuxtage 2013 hören Sie im **Bier-taucherpodcast 096:**

goo.gl/rmu79.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/chemnitz
Feedback ✉ horst.jens@spielend-programmieren.at

Lizenz, Quellen:



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] spielend-programmieren.at
- [2] chemnitzer.linux-tage.de/2013/
- [3] <http://goo.gl/Fa6mu2>
- [4] <http://goo.gl/EoIDpy>
- [5] osdomotics.com
- [6] goo.gl/fUs552
- [7] goo.gl/THuHS9
- [8] biertaucher.at
- [9] edubuntu.org
- [10] Desktop4Education
- [11] linuxadvanced.at
- [12] imediass.ch/lernstick

Technik-Nomade

von HORST JENS UND JONAS SMEDEGAARD

Ein Lifestyle Bericht von Horst JENS [5] über Jonas SMEDEGAARD [1], einen Sysadmin aus Dänemark der jahrelang freiwillig obdachlos lebte, aber niemals arbeitslos war und der dank Internet konsequent ortsunabhängig arbeitet.



Jonas Smedegaard [1], cc-by-sa

'Herumzigeunern', 'obdachlos', 'Herbergssuche', 'nicht-sesshaft': Diese Worte haben im deutschen Sprachraum keinen guten Beigeschmack. Keinen festen Wohnsitz zu haben wird eher mit Armut als mit Freiheit assoziiert, eher mit Begriffen wie 'Versager' als mit 'Lifestyle'. Zumindest in meinem Freundes- und Bekanntenkreis erregt der Satz 'Ich weiß nicht wo ich morgen schlafen werde' Mitleid anstatt Neid.

Aus eigener Anschauung habe ich fast nur positive Erfahrungen mit kurzfristiger, freiwilliger Wohnungs-Ungewissheit gemacht: Eine Woche in England ausschließlich per Couchsurfing unterwegs sein, als Jugendlicher mehrmals einen Monat lang per Interrail mit der Bahn herumfahren und in Fähren, Bahnhöfen, Felshöhlen und Gepäcknetzen schlafen, in Griechenland mit Schlafsack und Isomatte unterwegs sein: Aber immer mit der beruhigenden Gewissheit dass 'zu Hause' ein warmes Zimmer auf mich wartet, maximal ein, zwei Tagesreisen entfernt, komplett

mit technischer und sozialer Infrastruktur.

Umso mehr fasziniert es mich wenn ich Leute treffe die Obdachlosigkeit bewusst leben, nicht aus Not heraus oder als Urlaubssabenteurer, sondern sozusagen als Lebensabschnitt-Lifestyle:

Jonas, einem System-Admin aus Dänemark, habe ich in Deutschland (bei einem OSDomotics-Workshop⁵ im November 2013 kennen gelernt. Jonas sieht eine Spur unangepasst aus (so als ob er nicht oft Krawatten trägt) und hat eine sehr freundliche, extrovertierte Art. Speziell wenn es darum geht über die Vorteile von Linux oder freier Software zu reden wird er sehr enthusiastisch.

Jonas

Jonas wurde 1972 in Dänemark geboren, er beendete die (in Dänemark zehn-jährige) Grundschule aber flog zwei mal durch in der (drei-jährigen) Oberstufe. 1992, mit 20 Jahren, 'stopped trying to get an education' gab er die Schule endgültig auf und ging seinen Interessen nach: Computern und Sozialarbeit.

Mit 19 Jahren (noch als Student) hatte er seinen ersten IT-Job: in einem 'Pre Press office' wurden Printings von Werbeagenturen druckfertig gemacht. Er war für die Vernetzung der Macintosh Computer zuständig. Im Gegensatz zu Windows PC's waren Macs zu dieser Zeit leicht vernetzbar, per Telefonkabel.

Mit einem Freund [2] welcher hobbymäßig Cartoons zeichnete erstellte Jonas erste Webseiten mit anklickbarer Grafik (image map) für den Netscape Navigator⁶. Jonas zog von zu Hause aus und wohnte in Kopenhagen und Aarhus. Er startete seine eigene IT-Firma Macværk 1996 (später unbenannt in IT-guide dr: Jones) als Einzelunternehmer.

Sein wichtigster Kunde war (und ist immer noch) die Kaospilot University [3] in Aarhus für die er als Systemadministrator arbeitete. Die Universität schickte ihre Studenten und

Angestellten manchmal für ein Jahr zum Arbeiten ins Ausland, darunter auch Freunde von Jonas. Er versprach seinen Freunden, sie im Ausland zu besuchen.

Auf einer Reise in die Schweiz (Jonas begann sich mit Unix/Linux zu beschäftigen) wählte er sich vom Zug aus per Handy-Modem in einen Computer in Dänemark ein um seine Emails zu checken. Das war möglich weil er mit Linux im Terminal (Textmodus) arbeitete (ohne GUI) und daher nur wenig Bandbreite brauchte. Dabei kam ihm zum ersten Mal die Erkenntnis das er von jedem Ort der Welt aus arbeiten konnte.

1998 besuchte er einen Freund der gerade ein Auslandsjahr in San Franzisko (USA) machte. Vor der Abfahrt stellte Jonas sicher keine offenen Rechnungen zu haben, kündigte seine Wohnung, aber nicht seinen Job. Auch von der USA aus arbeitete er weiter für seine Kunden in Dänemark. Der US-Aufenthalt war auf 3 Monate begrenzt, danach hatte er keine genauen Pläne... vielleicht weiter nach Kanada reisen.

Distance working war damals gerade erst ein Schlagwort, alle Welt war der Meinung man müsste dafür erst mal schnelle ISDN-Verbindungen ausbauen. Jonas arbeitete per Dial-Up Modem und SSH Verbindung von San Franzisko aus um die Computer in Dänemark zu warten, sein Geld wurde weiterhin auf ein dänisches Konto gezahlt und er war weiterhin in Dänemark sozialversichert. Jonas musste nie um ein US-Arbeitsvisum ansuchen. Nur er selbst befand sich in den USA, seinen Wertschöpfung erfolgte in Europa.

In dieser Zeit baute er einmal ordentlich Mist: er verwechselte bei einem Linux-Kommando ein < mit einem > während er die Datei /etc/passwd bearbeitete. Dadurch wurde die Datei geschrieben anstatt gelesen und alle Studenten der Kaospilot University hatten keinen Zugang mehr zu ihren Emails. Während Jonas in San Francisco händeringend nach einer Lösung suchte (er bekam

⁵siehe Artikel chemnitz, RIS Journal 001, Seite 6)

⁶aus diesem Browser ging später der populäre Mozilla Firefox hervor

schließlich durch ein Webmin-Web-Interface mit schlecht konfigurierter Security Zugang zu einem Backup-File) schrieb ihm ein (!) Student eine Email mit der Frage warum er seine E-mails nicht mehr checken kann. Der Rest der Studenten merkte nichts bis der Fehler behoben war oder beschwerte sich zumindest nicht. (Dabei spielt auch der Zeitunterschied zwischen Europa und USA eine Rolle). Seitdem schätzt Jonas die *Pipe* (!) für Linux Kommandos.

In Dänemark hatte sich Jonas für die Zeit seines Auslandsaufenthaltes einen Büro-Raum in einem Internetcafe gemietet wo er seine Sachen einquartierte. Als er von den USA zurück kam plante er in diesem Büroraum (illegalerweise) auf dem Boden zu schlafen. Diese Pläne scheiterten daran dass das Internetcafe mittlerweile Pleite gegangen war, Jonas kam gerade noch rechtzeitig um seine Sachen abzuholen. Mangels Platz musste er einige alte, aber funktionierende Computer wegschmeissen, was ihm sehr leid tat. Schließlich erlaubte ihm sein Arbeitgeber, die Kaospilot University, einen Lagerraum zu benutzen. In diesem Lagerraum schlief Jonas auch. Einer seiner Freunde (Morten P.) hatte Mitleid und in der Universität einen festen Büroraum als Arbeitsplatz. Er ließ Jonas Nachts in diesem Büroraum schlafen - unter dem Schreibtisch im Schlafsack. Sein Freund weckte ihn jeden Morgen um seinen Arbeitsplatz zu reklamieren.

Jonas hatte es nach eigenen Angaben zu dieser Zeit einfach satt ständig nach neuen Wohnungen (bzw. Lagerräumen) zu suchen die er doch nur für ein paar Monate behalten würde. Er beschloss daher bewusst auf Wohnungssuche zu verzichten und am Arbeitsplatz zu schlafen.

Mit der Zeit wurde dies zu seinem Lifestyle. Er schlief oft bei verschiedenen Freunden. Da er nicht um 'Schlaferlaubnis für ein paar Tage' betteln wollte und seinen Freunden nicht lästig werden wollte unterließ er jede Vorausplanung seiner Schlafplätze sondern 'nahm jeden Tag so wie er kam'. Er wusste sprichwörtlich am Morgen nicht wo er am Abend

schlafen würde...aber es machte ihm nichts aus. Wenn er einmal gar keinen Schlafplatz fand nahm er sich ein Hotelzimmer, eine Jugendherberge oder (weil billiger als ein Hotel) eine Schlafwagen-Bahnkarte für die Fahrt zwischen Aarhus und Kopenhagen. Der Zug fuhr nachts über ein Fährschiff, weshalb die Fahrt recht lange dauerte.



Smoke (Spielfilm). Bildrechte: Amazon Partner Programm [4]

Duschen konnte er bei Freunden. Ein Problem war der Mangel an Kleidung: Er hatte drei verschiedene Sets von Kleidung, damit er immer ein Set waschen konnte. Sein Gepäck bestand aus einem Beutel mit den Kleidungs-Sets, Handtuch, Zahnbürste und Laptop.

In Waschsalons zog er sich zurück wenn er ein paar Momente 'alleine' sein wollte... Waschsalons sind besser geheizt als Bahnhöfe oder andere öffentliche Plätze, sind durchgehend geöffnet und weniger überfüllt. Mit Kopfhörern konnte er sich dort ruhig in eine Ecke setzen um zu computern, Musik zu hören oder Wäsche zu waschen.

Im Winter fand er ein Gentlemen's Agreement mit einem Kaffeehausbesitzer in Aarhus. Dieser Mann war laut Jonas 'very laid back, like the owner in the film *Smoke* [4]'. Unter anderem gab es in dem Kaffeehaus keine fixen Preise, sondern der Besitzer sagte jedem Kunden aus sei-

ner Laune heraus wie viel er zu zahlen hat.

Die Kunden liebten den schrulligen Besitzer. Während Jonas mit dem Besitzer tratschte (und von Gästen Whiskey spendiert bekam) fragte er wohin die Kellertüre führte. Ein Lagerraum den er für seine Sachen mieten konnte? Ohne nachzudenken oder gar über einen Preis zu verhandeln gab ihm der Besitzer den Schlüssel...zum ganzen Kaffeehaus, nicht nur zum Lagerraum. Jonas musste für den Lagerraum (der zwar nicht geheizt war, aber ein Bett hatte) niemals etwas zahlen, nur einmal bat ihn der Cafehausbesitzer um einen Gefallen: Eine Rechtsanwaltskanzlei hatte sich das Kaffeehaus für die Firmen-Weihnachtsfeier gemietet und der Besitzer wollte dass Jonas in die Party hineinplatze mit der (wahren) Behauptung er wohne hier. Jonas tat ihm den Gefallen und ging einigen Jung-Anwälten auf die Nerven weil er sich großzügig vom Buffet bediente und mit den Sekretärinnen flirtete, aber schließlich trank er friedlich mit dem Chef der Anwaltskanzlei welcher ihn lustig fand.

Das Kaffeehaus wechselte den Besitzer und Jonas gab den Schlüssel zum Lagerraum zurück.

Jonas lebte weiter sein Nomadenleben in Dänemark. Eine Wohnung vermisste er zu diesem Zeitpunkt nicht. Was ihm 'etwas' abging war 'die Bequemlichkeit mehr Kleider zu besitzen als man ständig herumtragen kann'. Im Herbst 1999 rief ihn eine japanische Bekannte an die er in Kalifornien kennen gelernt hatte. Ob er nach New York kommen wollte, das liege genau zwischen Kalifornien und Dänemark? Zuerst wollte er nicht, doch nach einer Woche nachdenken packte er seinen Laptop und flog nach New York. Dort wohnte er in einer Jugendherberge und dann bei Freunden von Freunden von Leuten die er zufällig kennen gelernt hatte. Zu dieser Zeit war **Couchsurfing** ein Begriff, noch keine Web Service.

Zu einer erhofften Romanze mit der japanischen Bekannten kam es nicht da sie nicht nur von ih-

rem Boy-Friend und einer Gruppe zusätzlicher Freunde, sondern auch von ihrer kleinen Schwester 'bewacht' wurde. Jonas fuhr zurück zu seiner nicht-existenten Wohnung nach Dänemark, sein mittlerweile trans-kontinentales Nomadenleben fortführend. In Dänemark lebte er zeitweise in der Wohnung seines Bruders, oder bei Freunden. Einmal betätigte er sich auch als 'house-sitter' und passte auf das Haus eines Bekannten auf.

Er schlief unter anderem bei Kunden und einmal in einem Dachboden der Kaospilot University (wo er beim Aufstehen merkte das im Raum darunter eine Vorlesung stattfand).

Im Jahr 2000 (er wohnte gerade bei seinem Bruder) merkte er dass sein Bruder gerne mehr Privatsphäre hätte. Jonas wurde bewusst dass er nicht ewig bei Freunden und Verwandten leben konnte. Im Frühling 2000 lud Jonas die japanische Immer-noch-nicht-Freundin für einen Monat nach Dänemark ein. Als sie zusagte ging ihm auf dass er keine geeignete Wohnung sondern nur ein WG-Zimmer hatte und er seine Bekannte schlecht in Lagerräumen unterbringen konnte. Somit miete er kurzfristig einen zweiten Raum in der Wohngemeinschaft, sich vom obdachlosen Wohnraum-Schnorrer zum multiplen Zimmerbesitzer entwickelnd.

Zu seiner großen Freude kam die Japanerin ohne Boyfriend nach Dänemark, zu seinem großen Leidwesen brachte sie ihre kleine Schwester mit und er hatte kein einzige Mal die Gelegenheit mit der älteren Schwester alleine zu sein. Um das Maß voll zu machen wollten die beiden Mädchen ständig mit ihm Sightseeing oder Aktivitäten machen - stets alle zusammen. Die Situation wurde so unerträglich für ihn dass er sich schließlich äußert erratisch benahm, sich in Arbeit verkroch, von den japanischen Schwestern nichts wissen wollte und beide schlussendlich aus der Wohngemeinschaft (und damit aus Dänemark) hinauswarf.

Jahre später hat er sich für sein Verhalten entschuldigt und (der

älteren Schwester) seine damaligen Gefühle gestanden. Sie dachte die ganze Zeit, er wäre homosexuell und deshalb nicht an ihr interessiert.

Jonas miete sich in diversen Wohngemeinschaften ein und schließlich 2001 in einem **Hackerspace** in welchem Nachts elektronische Musik gemacht wurde und wo regelmäßig Konzerte stattfanden. In diesem Hackerspace (einer ehemaligen Fabrik) gab es einen Kontrollraum welcher gegen Funkenschlag geschützt war...ein *faradayscher Käfig*. Jonas deponierte seine Sachen (und oft auch sich selbst) in diesem Kontrollraum. Laut seinen Angaben machte es ihm nichts aus in einem Käfig zu schlafen, aber die betrunkenen, lauten Konzertbesucher gingen ihm manchmal auf die Nerven.

Im September 2001 lernte Jonas seine jetzige dänische Freundin -Siri- kennen. Nach einer kurzen Beziehung mit ihr war er wieder Single. 2003 verließ Jonas Kopenhagen und zog in eine Ein-Zimmer Wohnung in einem *Ecovillage* (Ökodorf). Sesshaft geworden merkte er dass ihm Siri (sie hatte inzwischen einen anderen Freund) fehlte und er wollte dass sie ihren damaligen Freund verlässt und zu ihm zog. Siri hatte zu diesem Zeitpunkt schon zwei Kinder und wollte zwar zu ihm ziehen, aber (aus für Jonas nicht ganz nachvollziehbaren Gründen) nicht in eine Ein-Zimmer-Wohnung. 2004 zogen Siri und Jonas endgültig zusammen - in eine größere Wohnung. Heute, 10 Jahre später, ist Jonas glücklich mit Siri zusammen und bewohnt mit ihr ein Haus auf einer kleinen dänischen Insel. Jonas hat nie aufgehört als selbstständiger IT-Administrator zu arbeiten und hat gute und langjährige Kunden.



Jonas und Siri in ihrem Haus. Bildrechte: [1]

Obwohl mittlerweile ziemlich sesshaft und *gut im Leben angekommen* hat Jonas sein Nomadentum nicht gänzlich abgelegt.

Immer wieder zieht es ihn in die Ferne auf ausgedehnte internationale Vortragsreisen. Dort schläft er bei Fremden welche er nur von Online-Chats kennt, fliegt ohne Retourticket in fremde Länder für die er kein Visum hat um sich dort von Zufallsbekanntschaften am Flughafen zu Hackertreffen in noch fremdere Länder einladen zu lassen um schließlich bei Verwandten von Bekannten von Nachbarn von Leuten die er nicht kennt aufzuwachen und zu merken dass er weder Sprache noch primitivste örtliche Hygiene- oder Essgewohnheiten versteht. Jonas hält Vorträge über Linux und Web-Technologie und lernt gerne neue Leute und fremde Kulturen kennen.

Und selbstverständlich betreut er seine dänischen Kunden weiter, egal wo er sich gerade befindet.

Jonas betreibt einen kleinen Blog auf <http://dr.jones.dk/>

Fachbegriffe

Couchsurfing: Eine besonders bei jungen Reiselustigen Leuten beliebte Website, auf der man unkompliziert kostenlose Übernachtungsmöglichkeiten anbieten oder anfragen kann. Siehe <https://www.couchsurfing.org/>.

GUI: Graphical User Interface, (grafische Benutzerschnittstelle):

wenn man einen Computer durch anklicken von Symbolen bedienen kann (z.B. Windows). Im Gegensatz dazu muss man im Textmodus Befehle eintippen, um z.B. alle Dateien in einem Verzeichnis zu sehen oder um z.B. ein Programm zu starten.

Pipe: (engl. f. Rohrleitung)
Ein Umleitungsfunktion, die es ermöglicht dass der Output eines Befehls als Input eines anderen Befehls verwendet wird. Analog zur Anweisung: 'frag den Installateur welche Ersatzteile ihm fehlen und geh dann ins Lager und hole diese Ersatzteile'. Eine Pipe erzeugt man auf deutschen Tastaturen durch gedrückt halten der rechten Alt (Altgr) Taste und drücken der Taste links vom kleinen Ypsilon... deren dritte Belegung ist ein senkrechter Strich, die Pipe. Mehr zum Thema erfahren Sie auf der Wi-

kiseite von ubuntuusers.de.

Hackerspace ein Raum wo gemeinschaftlich an Computer- und Hardwareprojekten geforscht und gearbeitet wird. In Wien z.B. das metalab.at

Podcast



Horst erzählt über seine Begegnung mit Jonas im **Biertauerpodcast 134**:
goo.gl/WcUuX4

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene For-

mate:spielend-programmieren.at/risjournal/001/nomad
Feedback ✉ dr@jones.dk

Lizenz, Quellen:

Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen

4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] dr.jones.dk
- [2] binx.com
- [3] kaospilot.dk
- [4] goo.gl/9ENgMR
- [5] [spielend-programmieren](http://spielend-programmieren.at)

Wie man Billionär wird

von DAVID CAIN

In diesem Blogposting vom Jänner 2011 (!) erklärt *David Cain* [1] den Unterschied zwischen viel Geld, wirklich viel Geld und viel zu viel Geld. Das Originalposting erschien auf dem *Raptitude.com Blog* [2]. Originaltitel: *How to make trillions of dollars*.



David Cain. Bildrechte: [1]

Übersetzung von Gerald PINK und Horst JENS mit freundlicher Genehmigung des Autors. Anmerkung: Die Namen für große Zahlen sind im Amerikanischen etwas anders als im Deutschen (ein beliebter Übersetzungsfehler). Zur Übersicht:

Eine Million (engl.: million)
(eine 1 und 6 Nullen):
1.000.000
Eine Milliarde (engl.: billion)
sind 1.000 Millionen:
1.000.000.000
Eine Billion (engl.: trillion)
sind 1.000 Milliarden:
1.000.000.000.000
(Beginn der Übersetzung)

Zu Beginn möchte ich klarstellen, dass ich das nicht empfehle. Ich erkläre diese Strategie nur um zu informieren, damit du verstehst in welcher Umgebung du arbeitest, und damit du für dich persönlich gute Entscheidungen treffen kannst, darüber, wie du dein Geld verdienst und wie du mit Geld umgehst.

Ich ermuntere Dich dazu ein Millionär zu werden, wenn dich das interessiert. Falls du Milliardär werden möchtest finde ich das zwar verdächtig, aber ich möchte dich nicht vorverurteilen. Wenn du Billionär werden willst: das ist nur etwas für die wirklich Bösen. Dann können wir leider keine Freunde sein.

Richtig großes Geld steckt nicht

im Produzieren von Gütern sondern im Schaffen von Kunden. Ein einzelner, lebenslang treuer Kunde der sein Leben so verbringt, wie du es willst, ist Millionen wert. Millionen solcher Kunden zu schaffen, das ist der Weg zum Billionär.

Du kannst Millionen machen mit dem Verkauf eines großartigen Produkts, welches die Leute brauchen; aber du machst Milliarden und Billionen damit, ganze Nationen und Völker dazu zu erziehen, auf jedes WehWeh, auf jeden Wunsch, auf jede flüchtige Begierde oder Angst mit KAUFEN zu reagieren.

Du brauchst etwas Geld um die Billionärs-Strategie in Schwung zu bekommen. Das geht nicht mit indischen 5 Dollar/Stunde Assistenten. Aber, wie gesagt, ich möchte sowieso nicht dass du Billionär wirst, weil das nicht so gut für den Rest von uns ist.

Um in Schwung zu kommen braucht es einen Kulturwandel. Kleinunternehmer kriegen das nie hin, aber wenn du etwas 'altes Geld' (Vermögen) geerbt hast (irgendwer hat das immer) dann bist du in der Position um Politiker, Zeitungen und

Trendsetter zu beeinflussen, und dann kannst du ziemlich schnell definieren, was in Zukunft als 'normal' gilt.

Nach dem Ende des 2. Weltkriegs entwickelten ein paar privilegierte Amerikaner eine brillante Formel um eine unvorstellbar große Ökonomie zu erschaffen:

American retail analyst Victor Lebow [3]:

[Unsere Wirtschaft] verlangt dass wir ein Dasein als Konsument führen, dass wir aus dem Kauf von und dem Gebrauch von Gütern ein Ritual machen, dass wir unsere spirituelle und persönliche Erfüllung im Konsum suchen. Um sozialen Status zu messen, Prestige, soziale Akzeptanz ... das findet man in unserem Konsumverhalten [...] Wir müssen Dinge konsumieren, verbrennen, verbrauchen, ersetzen, wegschmeißen in einem immer schnelleren Tempo. Wir müssen die Menschen essen, trinken, anziehen, Auto fahren lassen mit immer komplizierteren - und deshalb immer teurerem - Konsumgütern.

(Danke an Leserin Anna für den Hinweis)

Das ist High-level-Marketing, und es hat den Großteil der entwickelten Welt um uns herum geformt.

Durch Benutzung von vor allem Massenmedien (Fernsehen) haben es High-Level Marketing Männer geschafft eine Nation zu erschaffen deren Menschen typischerweise:

- fast die ganze Zeit arbeiten
- sich jede Nacht zu Hause stundenlang Werbung reinziehen
- müde sind, ungesund leben [4] und mit ihrem Leben unzufrieden sind
- auf Langeweile, Unzufriedenheit oder Ängstlichkeit nur auf

genau eine Weise reagieren: Dinge KAUFEN

- Geld zur Verfügung haben aber keine erfüllendere Arbeit finden aus Angst davor die Sozialversicherung zu verlieren
- Gesundheitsprobleme für sich selber erzeugen, welche mit Tabletten behandelt werden.
- weitaus mehr Dinge besitzen [5] als sie benutzten, und trotzdem glauben dass sie hat nicht genug (Dinge) haben
- sehr leicht abgelenkt sind von ihrer ungesunden Lebenssituation und -Kultur durch ständige Nachrichten (breaking news) sowie TV-Tratsch über Prominente
- ständig selbst überzeugt sind dass genau jetzt nicht der richtige Zeitpunkt ist um den Lebensstil zu ändern
- gerne Sachen kaufen die ein Jahr später kaputt gehen und von dem niemand weiß wie man sie repariert
- gelernt haben durch die Medienkultur der Schuldzuweisung, dass der Schlüssel zur privaten und öffentlichen Problemlösung darin liegt herauszufinden welche Leute man hassen muss

Im Rest der entwickelten Welt ist es nicht viel anders. Billionäre und ihre Angestellten arbeiten rund um die Uhr daran diese Gewohnheiten per TV in die Wohnzimmer der Menschheit zu pumpen. Große Wirtschaftsmacht, große Bevölkerung, globaler Einfluss.

Gesunde Menschen - Menschen die wissen wie man mit Enttäuschungen umgeht, Menschen die nicht an 'Wunderlösungen' (Magic Bullets) glauben, Menschen welche nicht ungehemmt TV schauen, Menschen welche Erfüllung finden durch die Dinge im Leben die kein Geld kosten - solche Menschen sind schlechte (poor) Konsumenten. Die

High-Level-Marketing Männer haben eine Kultur produziert die genau das Gegenteil propagiert.

Heutzutage wirst du gedrängt, aus allen Richtungen, ungesund und unerfüllt zu werden oder zu bleiben, denn dann willst du mehr KAUFEN. Ich möchte dir nicht Angst machen, aber dies ist der primäre Zweck des matt schimmernden Kastens in deinem Wohnzimmer - dich auffordern zu schlechter (gerade noch nicht Konsum hemmender) Gesundheit, genereller Wehleidigkeit und einem unbewussten Reflex: Geld auszugeben.

Dieser Text ist nicht ein Aufruf dazu den nächsten **Wal-Mart** nieder zu brennen. Der Anti-Kapitalismus, das ist doch etwas für kleine Kinder.

Ich rate dir auch nicht dazu den Fernseher aus dem Fenster zu schmeissen. Obwohl das ein Akt mit einem hohen **Return of Investment** wäre, falls dich der Gedanke interessiert.

Ich rate nicht mal dazu einen Hass auf 'den Mann' (der diese Strategie erfunden hat) zu schüren. Es wäre eine Untertreibung zu sagen dass wäre so wie den falschen Baum anbellern. Da draußen ist nichts - keine Ideologie, Methodologie, Verhalten, Person oder Idee - welche es wert wäre zu hassen. 'Der Mann' macht was er macht weil er es nicht besser weiß - er ist selbst abhängig und weiß wenig vom Leben.

Deine Lebensfähigkeiten (Skills) können besser sein, so dass sie dich davor schützen zerstörerischen und selbst-beschädigenden Drängen nachzugeben. Solchen Drängen wie - unter anderem- z.B. dem Drang eine Kultur zu schaffen der schlechten Gesundheit, eine Kultur der Angst, eine Kultur der Sucht nur damit du exklusive Sensationen erlebst wie das Tragen von 10.000 Dollar Unterwäsche. Oder an der Börse mit Firmen zocken so als wären sie Spielkarten.

Ein wirklich armer Mann ist derjenige der es nicht zusammenbringt ein komfortables Leben zu führen ohne dafür Millionen Menschen zu verarmen um leichter an deren Geld zu kommen. Er kann da nicht raus - er

ist geldsüchtig.

Also hasse nicht 'den Mann', das macht dich nur schwach und lenkt dich ab davon was du erreichen kannst wenn du nicht seinen Ratschlägen folgst. Hass hat einen wirklich lausigen Return Of Investment, zumindest was die Lebensqualität betrifft. Wenn du überkochst vor Wut und Dampf ablassen musst, betreibe Sport. Richte deinen Zorn weg von anderen Menschen und Tieren. Wenn du genug Dampf abgelassen hast, richte deine Energie auf die Verbesserung deiner Lebensgewohnheiten und -Fähigkeiten.

Ich rede gar nicht über spezielle Fähigkeiten wie z.B. Reifen wechseln, Hemden bügeln oder Waschräume öffnen ohne die Türklinke zu berühren, obwohl sowas nützlich ist.

Ich rede über die Fundamente: Ein selbstbestimmter Mensch zu sein.

Kreativität. Neugier. Widerstand gegen Ablenkung. Geduld mit Mitmenschen.

Und um das alles zu ermöglichen: Selbstsicherheit - unbeugsamer Willen die Verantwortung für das eigene Leben zu übernehmen. Egal wer aller daran mitgewirkt hat dass dein Leben so ist wie es ist.

Kultiviere diese Fähigkeiten in dir selbst und in Anderen. Wenn diese Lebenseinstellung normal wird dann werden die Lifestyle-Tipps von Plakatwerbung, Sonderangeboten und CNN -die bewährte Billionärs-Strategie- nicht mehr wirken. Bei niemandem mehr.

Sogar von einem ohnmächtigen Startpunkt aus - ein mieses Zimmer in einer vergessenen Ecke der Gesellschaft welches geradezu erschaffen wurde dazu dich krank und impotent zu machen - Diese Lebensfähigkeiten werden mehr für dich tun als jede 'Anti'-Haltung die du dir vorstel-

len kannst. Das System zu hassen ist Amerikas Lieblingsbeschäftigung in der Vergangenheit gewesen. Es fühlt sich gut an, einmal begonnen ist es schwer damit aufzuhören, und es führt dich exakt nirgendwo hin. So etwa wie **Doritos** essen.

Es geht nicht um wir gegen die, es geht um wir für uns. (englisches Wortspiel: us for U.S.)

Obwohl 'der Mann' in vielerlei Hinsicht unreif ist, so hat er doch eines: sehr viel Geld. Seine Macht wirst du überall spüren.

Er hat deinen Lifestyle designt [6], beginnend mit den Jahren wo du einer Arbeit nachgehst die nichts mit deinen Leidenschaften zu tun hat; von der Autowerbung die du anstarrst während du in einer Restauranttoilette bist, über deinen DVD-Player der genau 30 Tage länger lebt als die Gewährleistungspflicht vorschreibt, bis zum ausgetretenen Pfad zwischen deiner Mikrowelle und deiner Fernseh-Couch.

Die Billionärs-Strategien die du erlebst sind so schrecklich weil sie schon als so normal aufgefasst werden. Du wirst merken wie sie sich in dein Leben hineinwinden. Nicht nur explizit - durch Zeitschriften und Werbung - sondern auch implizit, in der Art wie deine Mitmenschen, deine Lieben, von dir erwarten zu leben.

Du wirst ungewöhnlich werden müssen. Ich hoffe du traust dich.

(Ende der Übersetzung)

Fachbegriffe:

Wal-Mart Ein amerikanische Supermarktkette, gehört den erzkonservativen Koch-Brüdern (u.a. Sponsoren der Tea-Party)

Doritos amerikanische Kartoffelchips (Markenbezeichnung).

Return of Investment abgekürzt ROI: Grob gesagt der Gewinn dividiert durch das eingesetzte Kapital: Wie schnell man eine Investition wieder 'hineinverdient' hat.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/trillions
Feedback @DavidDCain (Twitter)

Lizenz, Quellen:

Übersetzung (deutsch):



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Original (englisch):



Der Originaltext ist lizenziert unter einer Creative Commons-Lizenz Namensnennung - NoDerivatives 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-nd/4.0/>

Quellen:

- [1] David Cain
bzw: raptitude.com/contact
- [2] goo.gl/g7bSCF
- [3] en.wikipedia.org/wiki/Victor_Lebow
- [4] <http://goo.gl/Q2F9cM>
- [5] <http://goo.gl/S5xbPH>
- [6] <http://goo.gl/AL1Rcy>

Beruf: Spiele-Tester

von CHRISTIAN HAUMER UND HORST JENS

Horst JENS [1] interviewte Dipl. Ing. Christian Haumer [2], der in Wien als Tester für Computerspiele arbeitet. Christian spricht über seinen Werdegang, seinen Beruf und

die dafür notwendigen Qualifikationen. Außerdem gibt er Tipps für Bewerbungsschreiben und redet über den Frauenmangel in der Branche



Christian Haumer [2]

Interview

H: Servus Christian.

C: Grüß Dich!

H: Du hast einen super Job in Wien, du wirst bezahlt fürs Computerspielen. Wie hast Du das geschafft?

C: Ich hab einfach angefangen mich zu informieren. Ich hab einmal geschaut was es in Österreich überhaupt für Firmen gibt. Manche Firmen habe ich schon vom Namen her gekannt, wie z.B. *Sproing* [3], die anderen habe ich gegoogelt. Und dann habe ich mich einfach blind beworben.

H: Aus der Diktion deiner Vokale schließe ich dass bist nicht aus Wien bist?

C: Nein, ich bin aus Niederösterreich.

H: Wie alt warst du, als du dich damals zu bewerben begonnen hattest?

C: Ich war schon ein wenig älter, knapp über 30, hatte gerade mein *Medientechnik Studium* an der FH St. Pölten beendet und war auf der Suche nach einem Job.

H: In welchem Bereich arbeitet man da dann üblicherweise?

C: Es war eine sehr breite Ausbildung: Audiotechnik, Videotechnik, Computeranimation, Fotografie und vieles mehr. Arbeiten kann man dann praktisch bei jeder Firma, die mit so was zu tun hat.

H: Wurden dort auch Spiele-relevante Themen unterrichtet?

C: Computer Spiele haben mich immer schon interessiert, waren in der Ausbildung aber leider nicht dabei.

H: Wie hast Du deinen derzeitigen Job bekommen?

C: Ich hatte halt leider keine direkte Games-Ausbildung. Ich kein Ar-

tist, kein Programmierer, kein Projektmanager, kein Level-Designer oder dergleichen. Ich sah aber dann im *Making Games Magazin* eine Praktikums-Ausschreibung von *Sproing* für einen **Q.A.** Job und hab mich einfach mal beworben. Nach einiger Zeit kam dann eine E-mail, ob ich noch verfügbar wäre und ob ich mal für ein Bewerbungsgespräch vorbeikommen könnte. Das Gespräch war erfolgreich und so bin ich dann dort in der Qualitätssicherung gelandet. Was ich daraus gelernt habe: Bewerbungen sind extrem wichtig! Auch wenn man nicht gleich eine Antwort oder erst einmal eine Absage erhält: Die Firmen behalten einem im Auge. Wenn später dann doch ein Job frei wird, wird man eventuell kontaktiert. Wenn die Bewerbung gut und aussagekräftig war, versteht sich.

H: War es ein unbezahltes Praktikum?

C: Nein, es war bezahlt, sogar nicht einmal schlecht. Man hört ja immer Geschichten, dass man als Praktikant nichts gezahlt kriegt. Zumindest meiner Erfahrung nach werden Praktika in unserer Branche aber ganz gut bezahlt.

H: Hat es eine Rolle gespielt dass Du schon dein Studium absolviert hattest?

C: Nur indirekt. **Q.A.** ist ein typischer Quereinsteiger-Job, für den man eigentlich keine spezielle Ausbildung braucht. Es ist aber nicht so leicht wie viele Leute glauben: Man ist mitten drin im Projekt, man begleitet das ganze Projekt, überprüft den Stand des Spiels, pflegt die *Bug*-Datenbank, weist die **Bugs** (Programmierfehler) den Programmierern zu und vieles mehr.

H: Das machst auch Du?

C: Ja, wenn der Programmierer den Bug gefixt (ausgebessert) hat, bekomme ich automatisch eine Meldung. Dann muss ich auf die neue Version des Spiels warten, wo der Fix schon drin ist, und dann kann ich testen, ob der Bug wirklich gefixt ist. Es ist eine recht kleinteilige Arbeit: Bugs reporten, schauen ob der Bug schon in der Datenbank drin ist, schauen ob vielleicht schon von jemand anderem

gefixt wurde, doppelt vorhandene Einträge schließen und so weiter.

H: Das Nachtesten ist notwendig?

C: Ja, das ist notwendig. Der Grund warum es **Q.A.** als Job gibt ist dass (Computer) Spieleprojekte extrem groß und kompliziert sind, und viele Leute daran beteiligt sind. Wenn der Programmierer eine Kleinigkeit ändert, und danach alles testen würde, was an der Änderung des Programmcodes noch dranhängt, dann kann das gleich ein paar Stunden dauern. Natürlich würde das möglich sein, aber effizienter ist es, wenn das Testen jemand er anderes macht: Die **QA**. Ideal ist sowieso eine Mischung aus beidem: Der Programmierer 'checkt seine Änderungen ein', schaut sich dann die unmittelbaren Auswirkungen an (z.B. startet das Spiel noch?), und wir, die **Q.A.**, testen dann im großem Rahmen alles ab, während der Programmierer schon an den nächsten Sachen arbeitet.

Man arbeitet aber nicht nur mit dem Programmierer zusammen sondern auch mit allen anderen Projektmitgliedern, den Designern, den Grafikern, den Projektmanagern, den Musikern und so weiter. Fehler können leider ja quasi überall passieren.

H: Wie sehr kommt es auf die 'Social Skills' an? Ich nehme an du musst jemandem auch diplomatisch sagen können das etwas nicht so toll ist?

C: Das ist wahrscheinlich der Punkt wo sich die **Q.A.** Spreu vom Weizen trennt. Wir sind diejenigen, die dem Entwickler sagen müssen *Du hast einen Fehler gemacht*. Das muss halt in einer Weise passieren, so dass er nicht beleidigt ist und dann nachher immer noch mit mir redet.

Social Skills sind speziell in einer kleinen Firma extrem wichtig. Bei *Sproing* haben wir Teams von 5 bis 20 Leuten, da hat man mit jedem Mitglied direkten Kontakt. Bei richtig großen Spielen anderer Firmen, wie z.B. *Call of Duty* oder *Gran Turismo* sind wahrscheinlich hunderte Menschen involviert. Dort gibt es dutzende Tester die dann ganz streng Prozessen und Workflows folgen müssen, und kaum direkten Kontakt

mit den Programmierern haben. Bei einem Autorennspiel kann es gleich mal sein, dass man dann zwei Jahre lang nur das Menü testet, wo man das Auto customizen kann. Da sieht man praktisch nichts vom fertigen Spiel. Ehrlich gesagt, das wäre nichts für mich. Ich persönlich arbeite sehr gerne in einem eher kleinen Team, da sind die Aufgaben viel abwechslungsreicher und interessanter.

H: Wie ist das wenn man 8 Stunden am Tag Spiele testet? Gehst du dann nach Hause und spielst noch Computer, oder fühlt sich das dann nach Arbeit an?

C: (Lacht) Ich probier's! Früher habe ich mehr gespielt, aber jetzt werde ich selektiver. Ich möchte unkomplizierte Spiele zum Entspannen, zum Beispiel Nintendo Spiele. Früher habe ich auch epische Rollenspiele mit dicken Handbüchern gespielt, in die man sich lange einarbeiten muss. Das mache ich jetzt nicht mehr. Wenn sich ein Spiel zu sehr nach Arbeit anfühlt, dann mache ich einen Bogen darum herum.

H: Was wäre rückwirkend ein guter Schwerpunkt in der Schule gewesen für einen Beruf als Spieltester?

C: Hmm. Alles, was mit Social Skills, Projekt-Management und Selbst-Management zu tun hat. Ganz wichtig ist auf jeden Fall eine gewisse Ausdauer und Frust-Resistenz. Es kann z.B. sein dass man zum zehnten Mal über einen speziellen Bug drüber stolpert, der immer wieder auftaucht und schon bereits schon mehrmals gefixt wurde. Man sagt dann dem Programmierer: 'Es tut mir Leid, aber das Problem tritt immer noch auf'. Der Programmierer fixt den Bug dann erneut und man testet wieder. Der Bug scheint weg zu sein, alles ist super. Stunden später, man testet gerade an einer ganz anderen Ecke des Spiels, taucht der Bug dann plötzlich wieder auf. Der Programmierer ist verärgert und meint 'Das kann nicht sein, das kann nicht sein! Ich habe das schon gefixt!' Dann muss man ihm eben ganz genau zeigen, wie man es geschafft hat, dass der Bug wieder auftritt, und er ihn noch einmal fixen kann. Es ist meis-

tens so, dass aus Gründen, die man vorher nie erraten würde, ein Bugfix ewig braucht.

H: Schreibst Du den Bug einfach in die Datenbank oder versuchst Du auch ihn zu reproduzieren? Inwieweit bist du da involviert?

C: Natürlich muss ich den Bug immer reproduzieren, bevor ich ihn reporten kann. Aber damit komme ich zum nächsten Punkt: Man braucht ganz gute schriftliche und kommunikative Skills. Und Englisch ist extrem wichtig!

H: Obwohl Eure Firma in Österreich ist läuft die Bug-Datenbank auf Englisch?

C: Genau. Bei uns ist die Firmensprache Englisch, weil einfach viele Englisch-Sprechende bei uns sind. Wir haben Leute aus Neuseeland, wir haben Leute aus Island, aus Portugal, wir haben Amerikaner, Engländer etc. Du musst Dich in der Branche unbedingt auf Englisch verständigen können, es muss zum Glück jedoch nicht perfekt sein. Wir haben aber leider schon einmal einen Bewerber abweisen müssen, weil sich beim Bewerbungsgespräch herausgestellt hat, dass sein Englisch nicht gut genug war. Obwohl er ansonsten einen richtig positiven Eindruck gemacht hat. Das war auch für uns bitter.

H: Ist (englische) Schrift oder eher Sprache wichtig?

C: Beides. Ich kann zum Beispiel auch nicht perfekt Englisch, aber ich kann mich ausreichend verständigen. Wichtig ist dass die Leute verstehen was ich schreibe und sage. In kurzen Worten eine genaue Anleitung verfassen, wie man den Bug reproduzieren kann, ist essentiell. Wenn der Programmierer erst einen langen Text lesen muss um zu verstehen worum es überhaupt geht, kostet das zu viel Zeit. Der Entwickler, der den Bugreport zugewiesen bekommt, soll sofort verstehen wo das Problem liegt.

H: Du bist schon länger in der Firma, verstehst Du dich noch immer mit allen Entwicklern gut? Oder bist du verschrien als der Mann, der die schlechten Nachrichten bringt? Gehen die Leute nicht mehr mit dir Bier trinken?

C: (Lacht) Nein, ich verstehe mich mit allen gut. Jeder, der professionell in der Branche arbeitet, weiß dass Fehler passieren. Es ist eigentlich nur die Ausnahme dass sich jemand persönlich angegriffen fühlt.

H: Was sagst du, wie hoch ist der Frauenanteil in deiner Firma / der Branche?

C: Bei *Sproing* arbeiten fast 90 Leute und ich glaube knapp 15 davon sind Frauen. Das ist für die Branche schon sehr gut, aber insgesamt halt leider doch noch viel zu wenig.

H: Du sprichst jetzt als Mitarbeiter und nicht als Firmenleitung, aber hättet ihr gerne mehr Frauen / Bewerberinnen?

C: Ja, auf jeden Fall! *Sproing* hat zum Beispiel sogar eine Geschäftsführerin. In der Entwicklung arbeiten eine Programmiererin, ich glaube zwei Grafikerinnen, drei Projekt Managerinnen, eine QA Testerin und der Rest der Damen ist dann (eher klassisch) in der Verwaltung tätig. Wir würden mehr Frauen sicher begrüßen. Und soweit ich weiß gibt es bei *Sproing*, und auch in der Branche bei uns in Österreich keine Vorurteile gegenüber Frauen. Es gibt halt einfach überhaupt keinen Grund, warum ein Mädchen nicht Programmieren studieren, und danach als Programmiererin arbeiten sollte.

H: Hast Du was gehört über schlechtes Betriebsklima, *Wää, lauter Nerds..*

C: Nein, im Gegenteil! Das sind doch alles eher Klischees.

H: Du selbst hast nie programmieren gelernt?

C: Die Grundlagen hab ich einmal gelernt, aber es ist nie über *Hello World* hinausgekommen.

H: Hast Du das Gefühl, die Entwickler freuen sich wenn Du als Spieltester etwas vom Programmieren verstehst, oder ist ihnen das eher lästig?

C: Sagen wir so: Wenn man siebengescheit sein will und ihnen erklären will, wie sie zu arbeiten haben und was da abgeht, dann werden sie das eher nicht so cool finden. Aber es hilft natürlich, wenn du die Grundlagen vom Programmieren verstehst. Du weißt dann eher warum welche

Sachen passieren, obwohl du es ganz genau nie verstehen wirst. Du solltest auf jeden Fall einen Hang zu Computer und Technik haben, schließlich hast Du den ganzen Tag mit sehr technischen Dingen zu tun. Es hilft auch sehr, wenn man kleine Computerprobleme selber lösen kann und nicht immer gleich zur IT rennen muss.

H: Möchtest Du den Lesern abschließend noch etwas mit auf den Weg geben?

C: Wenn man vorhat, in der recht kleinen österreichischen Spielebranche einen Job zu finden, dann muss man auf jeden Fall sehr geduldig und ausdauernd sein. Nie das Ziel aus den Augen verlieren, immer weiter an sich selbst arbeiten und sich nicht davon entmutigen lassen, wenn man nicht gleich was findet. Es kann oft viele Jahre dauern bis man sein Ziel erreicht.

H: Dein privater Blog?

C: Mein privater Blog ist www.humepage.at [2] und als **GameDev** Blog habe ich noch projectdev.humepage.at [6].

H: Danke für das Interview !

C: Sehr gerne!

Fachbegriffe:

Q.A: steht für Quality Assurance, Qualitätssicherung.

Bug: englisch für Wanze, Käfer: steht für einen Programmierfehler, da einer der ersten dokumentierten Computerfehler durch ein totes Insekt in einem Rechner ausgelöst wurde. Den (langwierigen) Vorgang der Fehlerbeseitigung nennt man debuggen.

GameDev: Abkürzung für Game Developer, Spiele-Entwickler bzw. Programmierer.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/spieletester
Feedback ✉ ch.haumer@gmail.com

Podcast



Das Interview wurde am 13. Dezember 2013 im Museumsquartier Wien im Rahmen einer *Subotron* [5] Veranstaltung aufgezeichnet und ist im Originalton im Biertaucherpodcast 135: goo.gl/HJN1Ec zu hören.

Lizenz, Quellen:



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] spielend-programmieren.at
- [2] www.humepage.at
- [3] sproing.at
- [4] projectdev.humepage.at
- [5] subotron.com

Molecular Matters

von STEFAN REINALTER UND
CHRISTIAN HAUMER

Christian Haumer war für seinen Blog *project:DEV* [1] beim österreichischen Game-Dev Veteranen Stefan Reinalter zu Besuch: dabei wurde über Stefans Werdegang und die Gründung seiner Firma *Molecular Matters* [2] geplaudert.



Molecular Matters Firmenlogo. Bildrechte: Molecular Matters

Dieses Interview erschien am 22.

Mai 2013 im *GameDev Blog* von Christian Haumer [3]

Interview:

C: Hallo! Kannst Du dich und Molecular Matters bitte kurz vorstellen?

S: Mein Name ist Stefan Reinalter und ich bin jetzt schon seit etwa zehn Jahren in der Industrie tätig. Vor zwei Jahren habe ich mich selbstständig gemacht und dann das Unternehmen *Molecular Matters* gegründet, mit dem ich Middleware für die Computerspiel-Industrie herstellen und verkaufen möchte. Daneben lehre ich an der FH Technikum Wien die Gegenstände *Objektorientierte Programmierung in C++*, *Game Development*, *Game Engine Design* und *fortgeschrittene Konsolen-Programmierung*.

C: Was ist deine Vorgeschichte, was

hast Du bisher gemacht?

S: Ich habe vor etwa zehn Jahren bei Rabcat als Programmierer angefangen, als sie noch ein internes Programmierer-Team hatten und an einem eigenen Projekt gearbeitet haben. Nachdem dieses Projekt leider nicht fertiggestellt werden konnte, bin ich ein paar Monate später zu *Sproing* gewechselt, wo ich dann über sechs Jahre lang gearbeitet habe. Ja, und vor kurzem habe ich mich dazu entschlossen, mich selbstständig zu machen um mich voll und ganz dem widmen zu können, von dem ich glaube, dass ich es gut kann – also die Entwicklung einer neuen, eigenen Engine von Grund auf.

C: Wann war das genau?

S: Molecular Matters gibt es jetzt seit September 2012.

C: Wie hat eigentlich alles begonnen,

wie entstand die Idee, ein eigenes Unternehmen und eine eigene Game Engine zu entwickeln?

S: Das Entwickeln von Technologie war schon immer das, was mir am meisten Spaß gemacht hat. Ich wollte mich schon immer voll und ganz auf das Entwickeln einer Technologie konzentrieren, ohne nebenbei an ein Projekt oder an eine tatsächliche Entwicklung gebunden zu sein, die in 9-12 Monaten fertig gestellt sein muss. Mich hat einfach interessiert, was man technisch auf die Beine stellen kann, wenn man die Zeit dafür hat, und an keine externen Deadlines gebunden ist. Daraus ist über die Zeit diese Idee entstanden, die mich letztendlich zur Gründung meines eigenen Unternehmens bewegt hat.

C: Wie lang dauerte es ungefähr vom Aufkommen der Idee bis zum tatsächlichen Schritt der Selbstständigkeit?

S: Eigentlich gar nicht so lange. Ich habe eh schon immer daheim viel herumgebastelt und Ideen ausprobiert. Aber so drei bis vier Monate vor der tatsächlichen Gründung war mir diesmal klar, dass ich den Schritt wagen und endlich meine eigene Engine realisieren werde.

C: Wie verlief der Schritt zum Unternehmen?

S: Ehrlich gesagt, am Anfang wusste ich überhaupt nicht, was mich erwartet. Ich habe versucht mich so gut es geht auf diversen Webseiten darüber zu informieren, und habe dann über Freunde und Bekannte erfahren, dass es beim AMS das *Unternehmensgründungsprogramm (UGP)* [4] gibt.

Nachdem ich auch arbeitslos gemeldet war, habe ich mich relativ bald für das UGP beworben und bin auch aufgenommen worden. Dann bekam ich von einer Consulting Firma einen Betreuer zugeteilt, der mir durch die ganze Bürokratie geholfen hat. Er sagte mir, welche Schritte ich in welcher Reihenfolge erledigen soll, wies mich auf die *Jungunternehmerförderung* [5] hin und half mir bei so gut wie allen Fragen zur Unternehmensgründung weiter. Das ganze hat etwa zwei bis drei Mo-

nate in Anspruch genommen, weil man auch seine Geschäftsidee ausgearbeitet haben und einen Businessplan und einen Finanzplan vorlegen muss. Die tatsächliche Gründung ist dann im September 2012 erfolgt.

C: Wann wurde der Name für das Unternehmen und die Engine gefunden – und wie aufwändig war die Namenssuche?

S: Das muss irgendwann im Sommer 2012 gewesen sein, kurz vor der Gründung. Mir war schnell klar, was der USP (unique selling point) der Engine sein wird, nämlich dass man einzelne Bauteile kaufen kann, ohne gleich die ganze Engine erwerben zu müssen. Dieser Aspekt des modularen Aufbaus hat dann recht schnell zur Analogie der Atome und Moleküle geführt, und der Name *Molecule Engine* war bald klar. Den Namen für das Unternehmen zu finden hat dann schon länger gedauert. Gemeinsam mit meiner Lebensgefährtin wurde sehr lange gebrainstormt, was mehrere A4 Seiten mit Ideen und Namens-Kandidaten produziert hat. Wir haben darauf geachtet, dass sich der Name gut in ein Logo umsetzen lässt, und dass er Bezug zum Namen der Engine hat. Und daraus ist dann letztendlich *Molecular Matters* entstanden.

C: Bietest Du mit Molecular Matters neben der Engine auch noch andere Produkte oder Services an?

S: Das Hauptaugenmerk liegt auf der Engine, früher oder später soll das Unternehmen natürlich wachsen. Intern entwickle ich mit einigen Partnern auch ein kleines Spiel, damit die Engine auch gleich einen eigenen *Showcase* (Vorführspiel) bekommt der zeigt, was man mit ihr alles machen kann. Ich biete aber auch Consulting und Contract Work an, was ich auch noch länger beibehalten will. Im letzten Jahr habe ich zum Beispiel bei *Mi'pu'mi Games* an der Xbox 360- und Playstation 3-Portierung der *Hitman HD-Trilogy* mitgearbeitet.

Meine Stärken liegen da vor allem in den Bereichen Debugging und Bug-Fixing, Low-Level-Programmierung, sowie Optimierun-

gen jeglicher Art. Bei Hitman habe ich mich vor allem um schwierige Crash-Bugs und Konsolen-spezifische Optimierungen gekümmert. An einem anderen Projekt, zu dem ich im Moment leider noch nichts sagen darf, war ich in den letzten Monaten auch beteiligt. Dabei würde ich gerne etwas darüber erzählen, weil ich auf das Ergebnis wirklich sehr stolz bin. Aber das muss halt leider noch warten.

C: Kommen wir zur *Molecule Engine*. Was kannst Du uns genaueres dazu erzählen?

S: Die Entwicklung der Engine ist schon sehr weit fortgeschritten, aber dauert leider noch etwas an. Im Moment gibt es zwei fertige Module, die man sich von der Website runterladen kann. Das eine ist das Basis-Modul, das alles abdeckt, was man sich an Basisfunktionalitäten einer Engine erwartet, also Filesystem, Debugging, Memory Management, Task-Scheduler, Profiling, Threading Utilities, Konfigurationsfiles und so weiter. Das zweite Modul ist dazu da, um mit diversen Input-Devices zu kommunizieren. Es supportet Joysticks, Keyboard, Maus, Konsolen-Controller, etc. Diese beiden Module kann man sich wie gesagt jederzeit kostenlos von meiner Website herunterladen und nach Herzenslust ausprobieren und evaluieren.

Richtig ins Detail möchte ich jedoch erst in ein paar Monaten gehen, dann wenn der große Grafikteil fertig ist. Weil heutige Spiele einfach extrem grafiklastig sind, ist dieser Teil eindeutig der aufwändigste Part der Engine. Aber wenn er fertig ist wird es dann auch endlich ausführliches Material zum Herzeigen geben. Und dann kann ich hoffentlich auch schon meinen Anteil an dem besagten, aber noch geheimen Projekt präsentieren.

C: Wann wird das voraussichtlich sein?

S: Mein Plan ist es, das Grafik-Modul im Oktober/November fertig gestellt zu haben. Dann möchte ich schon verschiedene Tech-Demos und vielleicht einen ersten Showcase vorzeigen können und endlich mit dem

eigentlichen Marketing für die Molecule Engine durchstarten.

C: Das klingt ja nach einem perfekten Zeitpunkt für ein weiteres Interview, das sich dann voll und ganz auf die Engine konzentriert, oder?

S: Auf jeden Fall, es würde mich sehr freuen!

C: Wunderbar, dann beenden wir dieses Interview an dieser Stelle und sehen uns später wieder!

S: Abgemacht!

C: Gibt es von deiner Seite noch etwas zu erwähnen?

S: Ja! Zur Entwicklung der Molecule Engine gibt es seit zwei Jahren den Blog *Molecular Musings* [6], der tiefe und detaillierte Einblicke in den Prozess und die Engine selbst gibt. Besonders stolz bin ich darauf, dass der Blog inzwischen von richtigen Größen der Entwickler-Szene gelesen wird. Er wird von Leuten von *Blizzard*, *Crytek*, *Valve*, *Ubisoft* und vielen mehr besucht, die sich oft direkt in den Kommentaren zu Wort

melden. Inzwischen habe ich über den Blog schon einige sehr interessante Kontakte gemacht. Das gibt dem ganzen dann natürlich einen zusätzlichen Motivations-Schub :)

C: Das kann ich mir vorstellen. Hast du noch abschließende Worte?

S: Hmm. Ich glaube für den Moment ist alles Wichtige gesagt.

C: Na dann vielen Dank für das nette Interview und bis bald!

S: Bitte, gern geschehen, hat mich sehr gefreut!

Wer mehr über Stefan Reinalter erfahren will, dem sei neben seinen zahlreichen Artikeln auf *#AltDevBlog* [7] besonders der Artikel *How the austrian guy ended up working in the games industry* [8]⁷ empfohlen.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene For-

mate:spielend-programmieren.at/risjournal/001/molecularmatters
Feedback ✉ ch.haumer@gmail.com

Lizenz, Quellen:

Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] projectdev.humepage.at
- [2] molecular-matters.com
- [3] projectdev.humepage.at/1419
- [4] www.ams.at/sfa/14081_10435
- [5] <http://goo.gl/zGU1xy>
- [6] molecularmusings.wordpress.com
- [7] goo.gl/Zrxf3O
- [8] <http://goo.gl/Sv060y>

Vom Computerspieler zum Programmierer

VON STEFAN REINALTER

Stefan Reinalter, Gründer der Firma *Molecular-matters* [1] beschreibt in diesem Blogposting (Originaltitel: *How the austrian guy ended up working in the games industry* [2]) seine Entwicklung vom jungen Vidospieler zum erwachsenen Programmierer in der Game-Industrie, inklusive einer kleinen Zeitreise zurück in die 80er Jahre mit ihren ersten Videospiele und Homecomputern. Übersetzung und Republikation unter Creative-Commons Lizenz von Horst JENS [3] mit freundlicher Genehmigung des Autors.



Bildrechte: cc-by-sa by David Liuzzo[3],[4],
public domain: [5],[6],[7],[8]



Firmenlogo von Stefan's Firma

Beginn der Übersetzung

Ich dachte es wäre nett hier auf meinem Blog (*#altdevblogaday* [4]) ein Vorstellungs-Posting zu haben; und was wäre dazu besser geeignet als meine Geschichte: von ei-

nem Bub aus Österreich (wir sind das Land ohne Kängurus) der in der (Computer)Spiele-Industrie gelandet ist....

Frühe Sucht

Es begann vor über zwanzig Jahren, als mir mein Vater eine →**ColecoVision Spielkonsole** zeigte - ich war damals circa fünf Jahre alt. Ich habe jetzt noch schönste Erinnerungen daran, damit *Mouse Trap*, *Donkey Kong* und *Pitfall* auf unserem alten Fernsehgerät zu spielen. Weil die Spiele damals so teuer waren kauften meine Eltern nur ein- oder zweimal im Jahr neue Spiele. Wir waren stolze Besitzer einer Sammlung von acht (8!) Spielen. Trotzdem, wir Kinder spielten damit so oft wir nur konnten.

⁷Siehe auch RIS-Journal 001, *austrianguy*, Seite 19



ColecoVision-Spielekonsole. Bildrechte: public domain

Commodore 64

Ein paar Jahre später investierte mein Vater sein Geld in unseren ersten Computer - einen →**Commodore 64**. Ah, der C64 !

Nicht nur gab es für den C64 eine riesige Auswahl an Spielen; es war sogar möglich Spiele zu kopieren (Raubkopien waren ein großes Problem). Es gab damals nicht viele offizielle Computerspiele-Verkäufer in Österreich; wir organisierten uns die Spiele durch Kopieren bei Bekannten. Es stellte sich heraus dass mein Vater einige Leute kannte, und wir spielten unzählig viele verschiedene Spiele. Obwohl das jetzt mehr als zwei Jahrzehnte her ist bekomme ich immer noch eine wohlige Gänsehaut wenn ich an die Spiele denke und speziell an die Melodien vom SID Soundchip des C64... Jeder von uns nahm Ladezeiten von über zehn Minuten und mehr in Kauf (mit dem Datasette - Kassettenrekorder) einfach weil die Spiele so großartig waren. Der C64 war eine tolle Maschine, ich liebte ihn !

Mit ihm konnte man nicht nur eine ungeheure Menge Computerspiele spielen, es war auch möglich Computerprogramme in der Programmiersprache →**BASIC** zu schreiben und zu starten. Bevor ich überhaupt wusste was Programmieren oder gar BASIC war, zeigte mir mein Vater →**LOGO** (mit →**Turtle-Grafik!**), wahrscheinlich mein erster Kontakt mit einer Programmiersprache. Nicht lange danach bekam ich heraus wie man damit einfache Schneeflockenmuster zeichnet mittels den Turtle-Befehlen *turn-left*, *move-forward*, *turn-right*. Kurz danach beschäftigte ich mich mit **BASIC**.

Obwohl ich damals noch nicht in der Lage war höhere Konzepte wie

z.B. →**GOSUB** zu verstehen kann ich mich daran erinnern dass es mir Spaß machte einfache **BASIC** Programme zu schreiben. Immer wenn meine Eltern nicht da waren schlich ich in ihr Schlafzimmer um den Computer zu benutzen. Immer öfter versuchte ich einfache Konzepte nach zu basteln die ich in anderen Spielen gesehen hatte, um herauszubekommen wie man das macht. Natürlich hatte ich noch keine Ahnung von →**Sprites** oder der C64 Architektur, aber es faszinierte mich den Computer dazu zu bringen etwas zu tun.



Commodore 64 Heimcomputer. Bildrechte: Bill Betram [5], cc-by-sa

erste Programmier Schritte

Wenige Jahre später gab mein Vater einen Haufen Geld aus für unseren ersten PC - einen 386er mit 33 MHz. Wir spielten zeitlose Klassiker darauf wie →**Wing Commander**, →**Syndicate**, →**Ufo (X-Com)**, →**Doom** - und wir luden Gäste ins Haus ein, um die ganze Nacht Doom Deathmatches zu zocken. Ich kann mich gut daran erinnern wie sehr ich die Leute bewunderte die in der Spiele-Industrie arbeiteten. Ich wusste es ist mein Traum eines Tages ein Spiele-Entwickler zu sein und an solchen Spielen zu arbeiten. Um diese Zeit herum begann mein älterer Bruder mit →**Turbo Pascal** unter →**MS-Dos** zu programmieren. Ich versuchte so viel zu lernen wie ich konnte und nervte ihn andauernd mit Fragen.

PC Demo Szene

Und dann kam der Tag an dem ich Bekanntschaft mit der *PC Demo Szene* machte.

Dies muss der Punkt gewesen sein in meinem Leben an dem ich anfing mich brennend für Low-Level

Programming zu interessieren, für Hardware-Details, für Arbeiten mit den grundlegendsten Hardwarekomponenten um aus allen Teilen das Letzte herauszuholen. Ich konnte einfach nicht glauben dass so etwas Fantastisches wie *Second Reality* oder *Crystal Dream 2* auf der dem armen 386 meines Vaters möglich war. Diese Demos gehören zu den besten Demos welche jemals gemacht wurden, und ich werde sie nie vergessen. Ich war entflammt. Ich wollte jedes noch so winzige Detail wissen über Texture Mapping, über 3D Algorithmen und mode 13h programming. Aber ich wusste nicht wo ich anfangen sollte. Zum Glück war mein Bruder schon in der Oberstufe und half mir dabei über Vektoren und Matrizen zu lernen sowie generell programmieren zu lernen.



Second Reality. Bildrechte: public domain

Ausbildung

Ich ging bald selbst in eine HTL und lernte langsam →**C/C++** in meiner Freizeit. Überraschenderweise lernten wir in der Schule fast nie programmieren, nur ein wenig Pascal, was für eine technische Ausbildung eigentlich ungewöhnlich ist. Zu dieser Zeit beschäftigte ich mich immer noch mit Texture-Mapping und Ähnlichem, aber ich kam nie recht voran mit meinem eigenem Spiel. Ich war immer viel zu beeindruckt von den neusten technischen Möglichkeiten und Fachartikeln die ich im Internet las. All das Fachchinesisch über 3D Grafik machte es für mich noch viel mehr interessant. Ich war richtig süchtig nach Technologie und Low-Level Details.

Nach dem HTL Abschluss wusste ich dass ich nicht zurück wollte in einen technischen Beruf, denn dies hatte durch die Schulzeit seinen Reiz

für mich verloren. Nicht richtig wissend was und wo ich studieren sollte stolperte ich in ein Computer Science Master Programm an der TU Wien. Ich übersiedelte nach Wien und studierte Computer Graphics und Digital Image Processing.



Bildrechte: TU-Wien [6]

Erster Lohn

Zurückblickend weiß ich das manches von dem was mir an der Universität gelehrt wurde nicht ganz meinen Erwartungen entsprach (höflich ausgedrückt) aber manches davon bahnte mir meinen Weg in die Spiele-Industrie. Ich war im 3. Semester von meinem Bachelorprogramm und wir Studenten mussten in Gruppen ein eigenes 3D Spiel mit Open-GL programmieren. Ich wusste dies war meine Chance um meine Fähigkeiten zu erweitern und state-of-the-art 3D Programmierung zu lernen. Und natürlich war ich ganz aus dem Häuschen bei der Idee dass wir unser eigenes Spiel schreiben konnten und dafür noch Noten bekamen, also steckte ich sehr sehr viel Zeit hinein. Unser Spiel gewann den *Best Game of 2004* [6] Preis der TU-Wien !

Später im Sommer dieses Jahres kontaktierte ich einen Studienkollegen der in einem österreichischem Entwicklungs-Studio arbeitete (die Szene in Österreich ist winzig klein). Ich fragte ihn ob sein Chef Praktikumsplätze anbietet. Es stellte sich heraus dass mein Kollege nicht mehr für die Firma arbeitete und nichts über deren Praktika wusste. Aber er bot an ein Job-Interview für mich zu arrangieren bei seinem derzeitigem Arbeitgeber, wenn ich interessiert wäre. Und ob ich interessiert

war! Ich ergriff die Chance, zeigte mein Portfolio her, zeigte das Spiel welches wir an der Uni gemacht hatten und ich begann am nächsten Montag zu arbeiten.

Nach all den Jahren war ich in der Spiele-Industrie gelandet.

fast forward

In den letzten 7 Jahren lernte ich extrem viel über Programmieren in meiner Zeit in der Branche. Ich arbeitete mit sehr talentierten, leidenschaftlichen und kreativen Leuten. Und auch wenn ich ein paar Crunchtimes erlebte mit über 100 Arbeitsstunden pro Woche, kann ich es von ganzem Herzen empfehlen in der Spiele-Industrie mit Gleichgesinnten zu arbeiten. In die Spiele-Industrie zu gehen war die beste Entscheidung in meinem Leben, es war immer mein Traum.

Ende der Übersetzung

Fachbegriffe:

Colecovision: Eine der ersten Spielekonsolen, produziert von 1982 bis 1984. Sie zeichnete sich dadurch aus dass damals populäre Spielautomaten-Klassiker wie *Donkey Kong* oder *Zaxxon* recht gut auf die Colecovision portiert wurden, da ähnliche Hardware wie in den 'großen' Spielautomaten verbaut wurde.

Commodore 64: ein sehr populärer Heimcomputer, mit einer unüberblickbarern Anzahl an (raubkopierten) Spielen. Wurde produziert von 1982 bis 1993. Zum Zeitpunkt seines Erscheinens war er mit 64 KiloByte (Kb) Ram sehr leistungsstark und hatte für damalige Verhältnisse erstaunliche Grafik- (16 Farben!) und Sound-Fähigkeiten. Mitgeliefert wurde ein BASIC Interpreter von Microsoft.

BASIC steht für 'Beginner's All-purpose Symbolic Instruction Code, eine einfach zu erlernende Programmiersprache war in auf jedem Heimcomputer der 80er Jahre installiert

(allerdings in zueinander nicht kompatiblen Dialekten).

Logo eine in den 60er Jahren von Seymour Papert (der unter anderem Tablet-Computer voraussah) entwickelte Programmiersprache, gedacht um Kindern Programmieren beizubringen. War ihre Zeit weit voraus.

Turtle-Grafik simuliert eine Schildkröte, an der ein Stift befestigt ist und die über ein Papier wandert. Man kann der Schildkröte Anweisungen geben (z.B. gehe nach vorn, dreh dich nach links) wodurch Muster auf dem Papier entstehen. Turtle Grafik gibt es heute für verschiedenste Programmiersprachen.

GOSUB 'geh runter' ein Sprungbefehl der Programmiersprache BASIC. Der Computer wurde damit angewiesen ein Unterprogramm auszuführen.

Sprites: (engl. Geist) sind bewegliche (2D)-Computergrafiken, z.B. die Spielfiguren beim Spiel 'Pac Man'. Wenn man eine Briefmarke auf einem Bild herumschiebt entspricht dies der Bewegung eines Sprites in einem Computerspiel.

Wing Commander Space-Opera Raumschiffspiel für den, 1990 erschienen.

Syndicate 1993 für MS-Dos-PC'erschiedenes taktisches Echtzeit-Computerspiel. Der Spieler steuert in Iso-Grafik eine Gruppe von Agenten seines Konzerns in einem finsternen Cyberpunk Szenario welche Missionen erledigen müssen.

Ufo (X-Com) eine Serie von runden-basierten, taktische Computerspielen (ab 1994). Der Spieler steuert eine kleine Anzahl soldaten um die Erde gegen Außerirdische zu verteidigen und muss sich auch um Forschung, Basenbau und Personalmanagement kümmern. Mehrere Fortsetzungen und Spin-offs, unter anderem das Opens-Source Spiel *ufoai.sf.net*

Doom (1993) einer der ersten netzwerkfähigen 3D Egoshoooter von ID Software. Machte den Begriff 'Killerspiel' populär.

MS-Dos Kommandozeilen-Betriebssystem (ab 1982) von Microsoft für 386er Computer.

Turbo Pascal war eine Variante der Programmiersprache Pascal komplett mit Entwicklungsumgebung (Compiler, Editor) und kam von 1983 bis 1993 in immer neuen Versionen heraus. Da deutlich schneller und leistungsfähiger als **Basic** war Turbo Pascal u.a. vorzüglich geeignet um Programme für das Betriebssystem *MS-Dos* zu schreiben und wurde oft in Schulen als Programmiersprache für den EDV-Unterricht gelehrt. Mit dem Aufkommen des Betriebssystems Windows verlor Turbo Pascal an Bedeutung. Open Source Projekte wie Free Pascal werden auch heute noch weiterentwickelt und sind mit

Turbo-Pascal kompatibel.

C,C++ Moderne Programmiersprachen.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/editorial
Feedback ✉ stefan.reinalter@molecular-matters.com

Lizenz, Quellen



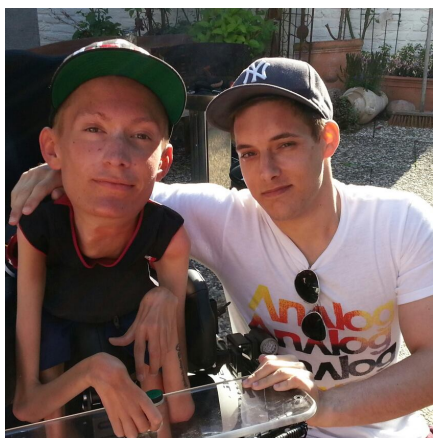
Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] molecular-matters.com
- [2] goo.gl/ohJNMH
- [3] spielend-programmieren.at
- [4] altdevblogadey.com
- [5] goo.gl/pyDNWP
- [6] goo.gl/jUmhba

Beruf: Counterstrike Spieler

von JULIAN WALTER UND HORST JENS



Julian Walter (im Bild rechts). Bildrechte: [1]

Julian Walter [1] erzählt über seine Zeit als professioneller Counter-Strike-E-Sportler und die damit verbundenen Verdienstmöglichkeiten

Interview:

H: Servus Julian

J: Hallo

H: Julian du hast mir erzählt du warst **Counterstrike**-Legionär

J: Das ist soweit korrekt. In meiner

Glanzzeit, zwischen 16 und 21 habe ich sehr viel gespielt. Ich war sehr aktiv und habe praktisch alle meine Freizeit in dieses Spiel gesteckt um so gut zu werden wie es möglich ist und hatte viel Spaß dabei.

H: Kannst du erzählen, hattest du vorher schon irgendeinen Sport oder irgendetwas professionell betrieben? E-Sport? Oder hast du aus dem Nichts heraus angefangen?

J: Nee, Nein, so kann man das definitiv nicht sagen ich habe angefangen sehr früh mit der ersten Counterstrike Version die damals zu kaufen war.

H: Nicht mit dem **Mod** (zu *Half-Life*) sondern mit dem gekauften Spiel?

J: Genau mit dem gekauften (Spiel), da war ich 11 (Jahre alt).

H: Mit 11 hast du schon herum geballert?

J: Genau. Man merkt halt mit der Zeit - durch das public (auf public servern) spielen, dass man besser ist als andere Leute und hat dann angefangen sich Leute zu suchen mit denen man längere Zeit zusammenspielt um dann auf Wettbewerbsebene zu bleiben. So ist das im Prinzip entstanden.

H: Du hast gemerkt dass du aufgestiegen bist in den Rankings?

J: Ja genau. Man wurde immer besser und immer besser und dann kam der Ehrgeiz hinzu. So jung wie man ist macht das dann auch Spaß.

H: Wie viel Zeit hast du ins Counterstrike investiert? Hast du tägliche Trainingszeiten gehabt?

J: Definitiv. Zu meinen Top-Zeiten, wo ich monatlich bezahlt wurde fürs Spielen,

H: Das war dann später..

J: Ja, das war erst später, da war es täglich jeden Abend, 5 x die Woche, (je) 4 bis 5 Stunden. Da hat man dann wirklich schon sein Leben (da)mit verbracht, kann man so sagen.

H: Und du hast nebenher noch Beruf oder Ausbildung gemacht?

J: Genau. Erst Schule, dann Zivildienst, Ausbildung.

H: Hat deine Ausbildung darunter gelitten? Hast du gemerkt du hast weniger Zeit für Freundin...

J: Ja. Definitiv. Ich hatte auch Beziehungen nebenher. Das war natürlich besonders stressig, weil die Freundin hat das dann halt nicht nachvollziehen können dass ich da so viel Zeit hineinstecke. Und das alles unter einen Hut zu kriegen ist sehr schwierig. Meistens - leider Gottes muss ich das sagen - haben die persönlichen Beziehungen darunter gelitten.

H: Du hast lieber gespielt als dich .. *lacht*

J: Ja, das war einfach so.

H: Bist du rekrutiert worden von **Clans** oder hast du selber einen gegründet? Wie hat sich deine Profes-

sionalisierung entwickelt?

J: Ich bin durch die Amateur-Series -so hieß das damals- aufgestiegen aus eigenen Kräften mit meinen 4 Freunden quasi in die Pro(fessional)-Series. Nach einer Saison, wo wir uns eigentlich ganz gut gezeigt hatten wurde ich sofort rekrutiert und ...

H: Nur du oder auch deine Freunde?

J: Damals nur ich.

H: Du wurdest aus deinem Clan herausgerissen sozusagen?

J: Genau.

H: Wie hat der Clan dann reagiert? Haben sich alle gefreut *Jawoll, du wirst jetzt Profi?*

J: Die waren geteilter Meinung. Die einen sagten: *O.K., Du bist ein arschloch, weil Du hast den Clan verlassen*

H: Du lässt Dich kaufen.. [lacht]

J: Genau, *Du verlässt Deine Freunde...* Andererseits muss man dazu sagen: Will man Erfolg haben muss man letzten Endes egoistisch sein, das hat mir nicht geschadet.

H: Wie läuft das dann, du bist ja nicht oft physikalisch mit dem neuen Clan zusammen. Oder schon? Geht man da gemeinsam auf ein Trainingscamp -wie beim Rudersport?

J: Kommt ganz drauf an wie mächtig, wie groß der (Clan) ist, wir groß diese Organisation ist. Wenn es eine große Firma ist dann hat man da Räume gemietet und hat sich da regelmäßig getroffen.

H: Das heißt ihr wart auch physikalisch zusammen?

J: Ja. Ich habe auf jeden Falle alle (vom neuen Clan) kennengelernt.

H: Meinst du das bringt beim E-Sport etwas, wenn man wirklich physikalisch im selben Raum ist? Im Gegensatz zu nur online zusammenzuarbeiten...

J: Für mich: Ja. Es gibt allerdings auch Leute, die nennt man gerne *die Onliner*, das sind halt die Leute die halt lieber zu Hause spielen in ihrem Kämmerchen.

H: Die gehen nicht gerne raus...

J: Genau. Aber es gibt auch Leute die sind deutlich besser wenn man (physikalisch) zusammen sitzt. Weil auch die Stimmung ganz anders ist.

H: Macht man dann nachher noch gemeinsam viel, wie beim Sport, dass

man sagt wir gehen nachher noch auf ein Bier nach dem Training? Oder ist das dann eher selten?

J: *Nach dem Trainings Bier* das kann man so nicht sagen. Natürlich hat man viel Spaß miteinander. Auf Events, wenn man auf großen Turnieren ist, da geht man dann auch miteinander feiern. Ansonsten...Tja.

H: Ansonsten spielt es sich so ab dass ihr zusammen sitzt, ihr trainiert, spielt euer Spiel und seid nachher müde?

J: Ja. So sieht es aus.

H: Spielt man dann zum Ausgleich ein anderes Spiel, z.B. ein Grafikadventure, irgendwas ruhiges zum runter kommen? Oder hat man nebenbei noch einen Zweit-Egoshoooter?

J: (Zweit-)Egoshoooter wäre fatal, das sollte man nicht tun, weil Counterstrike sehr speziell ist, was vor allem das Spielgefühl und so angeht.

H: Da lernt man dann den *falschen* Egoshoooter? *Du sollst keinen Egoshoooter haben neben deinem?*

J: Sollte man besser nicht tun. Zum Ausgleich ganz gerne DOTA damals - ein Warcraft3 Mod. Kennst du *Hon, LoL, League of Legends?*

H: Ähhhhh...

J: Das sind alles Ableger von *Dota*.

H: Also ein ganz ein anderes Genre zum Ausgleich.

J: Ja, aber auch nicht zu viel.

H: Kannst du zum Finanziellen etwas sagen, also in der Zeit wo du Profi warst, ist das jetzt lange her?

J: Das war ein Jahr her, 2011 war meine letzte Top Season.

H: Was kann man als Professioneller oder als Semi-Professioneller verdienen?

J: Kommt drauf an. In Deutschland ist es nicht so hoch.

H: Du warst damals in Deutschland?

J: Klar. In Deutschland sind die Top-Gehälter so bei 1.000,- bis 1.500,- Euro (pro Monat). Es gibt aber nur eine Handvoll (Spieler) maximal (die so viel verdienen). Wenn man ins Ausland guckt, gerade im asiatischen Bereich, da verdienen sie deutlich mehr. Bis 5.000,- , 7.000,- Euro....

H: Hast du realistisch Chancen als Europäer nach Asien rekrutiert zu werden wenn du so gut bist?

J: Das gibt es, ja. Aber selten.

H: Ich nehme an da leidet das Sozialleben dann noch mehr...

J: Genau, diesen Schritt hätte ich nicht gewagt.

H: Du sagst in Deutschland kann eine Handvoll Leute vom Counterstrike spielen leben, ca. 1.000, Euro, toller Verdienst

J: Es ist ein guter Nebenverdienst.

H: Man sollte daneben noch einen Job haben?

J: Kommt auf die Zeit an. Mit 16, 17 Jahren ist das lustig. Vor allem Geld mit counterstrike zu verdienen macht Spaß, hebt nochmal den Ehrgeiz an... Später dann, wenn man nicht *nur ein Studium* macht, sollte man damit aufhören. Weil das raubt Kraft und das raubt Zeit.

H: Der durchschnittliche Counterstrike-Verdiener ist Student oder Schüler? Oder sind das dann so 40-jährige Arbeitslose?

J: Nein, um Gottes Willen. Eigentlich gehört man mit 25 (Jahren) schon zu den Rentnern in diesem Sport, in der Regel sind die nicht älter als 25 und die meisten sind Studenten.

H: Die haben einfach Zeit...

J: Genau.

H: Kannst du sagen was verdient die große Masse der Counterstrike-Verdiener? Als nicht die Handvoll Top-Leute?

J: Mittlerweile hat sich das schon fast erübrigt mit dem Bezahlen, da verdienen -sag ich mal- noch ca. 40 Leute in Deutschland, und auch nicht mehr als 200,- , 300,- Euro..

H: Die bessern sich ihr Taschengeld auf sozusagen

J: Ja genau. Und sie machen nur es für den Ehrgeiz, für den Spaß.

H: Also wenn man es auf die Stunden umrechnet, die man rein steckt, ist es eigentlich blöd...

J: Rechnet sich auf jeden Fall nicht.

H: Hast du da Einblick, wie sehr vererbt sich das Hobby Counterstrike? Der Generationen-Durchsatz muss ja recht hoch sein wenn die Leute nicht lange im Sport bleiben weil sie zu alt werden.. Wird das immer wieder weitergegeben an die nächste Generation? Vereinsleben, E-Sport... Könnte es sein das Counterstrike ein-

mal so ein Großvater-Spiel wird und die Jungen auf ganz etwas Anderes abfahren?

J: Das Ur-Counterstrike 1.6, das war eigentlich der Beste E-Sport Titel meiner Meinung nach, das ist dieses Jahr komplett gecancelt worden, es gibt keine Turniere mehr, der Betrieb wurde fast eingestellt.

H: Von den E-sport-Betreibern aus? Oder weil du das Game nicht mehr kriegst?

J: Natürlich kriegst du das Spiel noch. Es wird eigentlich auch noch gespielt, aber nicht mehr auf Wettbewerbsebene, weil es zu alt ist. Das ist jetzt einfach 11 Jahre alt... Jetzt gibt es den Nachfolger, *Global Offensive*, wird jetzt wieder mehr gespielt.. ich denke das andere Genres den Überflieger machen, so wie League of Legends und DOTA 2.

H: Das sind nicht Egoshoooter?

J: Das sind keine Egoshoooter.

H: Der E-sport verlagert sich... Kannst Du sagen welcher Ego-Shooter ist der mit dem (noch) meisten Esport-Leben?

J: Nur Counterstrike.

H: Die anderen Ego-Shooter ... können da nie richtig dagegen anstinken?

J: Battlefield z.B. ist nie wirklich auf Wettbewerbsebene gespielt worden. Damals halt natürlich Quake und Unreal Tournament, aber die sind mittlerweile auch quasi ausgestorben.

H: Magst du irgendeine (Lebens)Erfahrung mitgeben für jugendliche Hörer von unserem Podcast die sich denken *Ja, ich werde mein Geld mit Videospielen verdienen...*?

J: [lacht] Das ist ein schöner Traum, macht extrem viel Spaß, man lernt viele nette Leute kennen. Wenn man gut ist reist man auch tatsächlich viel um die Welt. Allerdings muss man darauf Acht geben dass man nicht seine persönlichen Beziehungen aufs Spiel setzt. Das passiert einfach schnell, wenn man den Überblick verliert. Das ist schon fast wie eine

Sucht. Man muss aufpassen dabei. Und man muss Prioritäten setzen. Definitiv NICHT Counterstrike über die Schule stellen oder über das Studium oder die Ausbildung. Mann muss einfach Prioritäten setzen und dann macht es Spaß, ja.

H: O.K. Kannst du ein schönes Erlebnis erzählen?

J: Mein schönstes Counterstrike Erlebnis war: In Paris wurde ich zu einem Stage-Match, ein Spiel auf der Bühne, eingeladen gegen ein sehr gutes Team. Wir waren die deutschen Underdogs. Wir haben uns sehr gut vorbereitet und haben sie wirklich, wirklich geklatscht ⁸. Und dieses Gefühl, vor 3.000 Leuten zu stehen die applaudieren weil du gewonnen hast...das ist unvergleichlich.

H: Und das Publikum ist mitgegangen, oder kriegst du davon eh nichts mit weil du Kopfhörer auf hast..

J: Man sieht es (das Publikum) nicht (während dem Spiel), nein. Aber (nachher) super Gefühl, unvergleichlich.

H: O.k. Seltsamstes Erlebnis?

J: Leute kennen zu lernen von denen man noch nie Bilder gesehen hat und plötzlich etwas erschrocken sein weil...das Äußere ein wenig ... ungepflegt ist. Drücken wir es mal so aus.

H: [lacht] Du hättest nichts dagegen gehabt die (Leute) weiterhin nur online zu kennen....

J: Ja.

H: Danke Julian !

Fachbegriffe:

Counterstrike: Multiplayer, Team-Ego-Shooter, (*Killerspiel*) bei dem 2 Teams aus bis zu je 16 Spielern gegeneinander kämpfen und versuchen eine Aufgabe für ihr Team zu lösen. Counter-Strike entstand 1999 als Mod für das Spiel Half-Life und ist inzwischen ein eigenständig verkaufte Spiel.

E-Sport: Elektronischer Sport, organisiertes Computerspielen mit Wettkämpfen, Meisterschaften Verbänden etc. Besonders populär in Südkorea.

Download:

Dieses Interview wurde am 15. Oktober 2012 von *Horst JENS* [2] aufgenommen und ist im *Biertaucherpodcast 074* [3] zu hören.

Podcast



Dieses Interview wurde am 15. Oktober 2012 von *Horst JENS* [2] aufgenommen und ist im *Biertaucherpodcast 074*: goo.gl/50iZE zu hören.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/counterstrike
Feedback ✉ horst.jens@spielend-programmieren.at

Lizenz, Quellen:



Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] goo.gl/PqQ42q
- [2] spielend-programmieren.at

⁸abgewatscht, besiegt

Die Spieleindustrie ist ein einziger Betrug, arbeite dort nicht!

VON GAMESINDUSTRYVET



anonymer Autor

In diesem rant auf der Diskussionsseite [reddit.com/r/gamedev](https://www.reddit.com/r/gamedev) schrieb sich Nutzer „gamesindustryvet“ (Profil mittlerweile gelöscht) 2012 seinen Frust von der Seele und warnt vor den schlechten Arbeitsbedingungen in der (U.S. amerikanischen) „games industry“.

Originaltitel: *The game industry is a scam and this is why you shouldn't work for it* [1]

Kommentiert und übersetzt von Horst JENS[2], Abdruck mit freundlicher Genehmigung von [reddit.com](https://www.reddit.com)

(Beginn der Übersetzung)

Posting auf r/gamedev

Hallo Leute, ich bin seit 10 Jahren ein Veteran der (Computer)Spieleindustrie. Ich kann es selbst nicht glauben dass ich schon so lange drin bin. Durch anstrengende Misserfolge, gefolgt von noch mehr Misserfolgen (meist das Resultat haarsträubender Management Entscheidungen) hat sich jeder Hauch von Interesse zerstört den ich für diese Branche je hatte.

Ich suche nach Arbeit außerhalb der Branche Dank der in der Computerspieleindustrie gemachten traumatisierenden Erfahrungen. Ein paar Beispiele:

1) Ich arbeitete nonstop 52 Stunden. Ich hatte fast 2 Stunden Schlaf in dieser **Crunch-time** aber blieb die ganze Zeit über im Büro um das Projekt am Laufen zu halten. Weil ich Angststörungen (*anxiety disorder*) habe nahm ich meine Medikamente nicht (während der Crunch-time) und hatte einen totalen Zusammenbruch (*mental breakdown*) am Ende.

2) Ich wurde zwei mal in eine andere Stadt versetzt von einer großen Firma und 3 mal gekündigt (von der selben Firma), jedes mal nicht einmal ein Jahr nachdem ich angefangen hatte. Beim letzten Mal 2 Wochen nachdem ich angefangen hatte.

3) Ich verlor mein Haus, musste Privatkonkurs anmelden (had to file bankruptcy) und humpelte durchs Leben nach meinem ersten Umzug. Ich verlor (die Beziehung zu) vielen Freunden und Verwandten während dieser Zeit wegen Stress.

4) Vor Kurzem arbeitete ich für ein kleines Entwicklungsstudio und musste mich operieren lassen was 2 Wochen Arbeitsausfall nach sich zog. Am Tag nach der Operation wurde ich gekündigt, obwohl mir vorher gesagt wurde es wäre kein Problem wenn ich die Operation machen lasse. Ich hätte wegen der Operation nie meinen Job riskiert... aber warte, das ist die *Games Industry*, da ist jeder Job ständig gefährdet, außer wenn du ein Programmierer (Coder) bist.

Ich wechselte die letzten 5 Jahre mindestens einmal pro Jahr die Firma. Ich war schon lange nicht mehr bei einem Produkt-Launch (Veröffentlichung eines Spiels) dabei wegen der schlechten Entscheidungen die *senior manager* machten. Viele der Computerspiele (titles) werden gecancelt unabhängig von der Qualität, weil jemand oben in der Hierarchie mit irgendeiner Entscheidung nicht einverstanden ist.

Ich brachte enorme Opfer für diese Industrie. Ich bin fast 40 und habe keine Freundin (*no relationship to speak of*). Ich lebe in einer fremden, überbezahlten Stadt woch ich -wieder einmal- arbeitslos bin, und meine Lebenslauf schaut aus wie Mist (*looks like hell*). Ich ging auf eine bekann-

te technische Universität und machte einen begehrten Abschluss (nicht als Programmierer) aber gab das alles auf um meinem Traum zu folgen.

Dieser Traum wurde total und vollständig vernichtet von den höheren Managern. Jenen die lachen, winken und hier auf **Reddit** darüber posten wie großartig es ist in der *Games Industry* zu sein. Die sind an ihre Position gelangt weil sie die rücksichtslosesten Leute im Geschäft sind und Karrieren, Familien und Nachbarschaften zerstören. Alles damit sie Stars in der *Games Industry* werden obwohl sie außerhalb der Branche niemand kennt. Ich habe Freunde und Kollegen die in dieser Branche waren und physikalisch krank geworden sind, emotionale Schäden haben, und absolut niemand kümmert sich darum wie Mitarbeiter in der *Games Industry* behandelt werden, weil, Hey, da gibt's immer 5 Neue hinter Dir die mehr als willig sind alles zu opfern für den Erfolg der Bosse.

Ich hab lange darüber nachgedacht dies alles aufzuschreiben. Ich weiß das viele von Euch (Lesern) in die *Games Industry* rein wollen. Aber ich warne Euch: es gibt einen Grund warum dies ein Erfolgsabhängiges Geschäft ist (hit-driven business), und das alte Herren Netzwerk ist fest eingessesen in dieser Branche. Ich lernte dass hart arbeiten und gute Qualität zu produzieren den Bossen absolut nichts bedeutet. Die wollen dass du das tust. Es wird vorausgesetzt, und wenn du fertig bist mit produzieren schmeissen sie dich raus.

Dieses Geschäft ist ruchlos, halsabschneiderisch und voll mit nachäffenden Firmen (copycat companies) welche versuchen erfolgreichen Titeln die Kunden zu stehlen und dazu zu bringen den eigenen, unoriginellen Mist zu kaufen. Und hey, mit genug Werbung werden sie auch Erfolg haben.

Die ganze Industrie hat fast gar nichts zu tun mit der Qualität eines Spiel. Ja, manchmal kommen gute Titeln heraus und deren Entwicklungsstudios sind trotz allem erfolgreich.

Aber selbst solche Studios (ich denke an das *LA Noire team*) leiden.

Die *Games Industry* Angestellten verlassen in Scharen diese Branche... nur sagt dir das niemand. Nachdem *EA-Spouse* unterging war *EA's (Electronic Arts)* Antwort jeden Mitarbeiter unterhalb der Management Ebene nur noch stundenweise zu bezahlen, aber Vollzeitbeschäftigung verlangte mindestens 45 Stunden pro Woche. Wenn du das nicht geschafft hast bekamst du keine Vergünstigungen. Niemand erwähnte das jemals. Zum Glück arbeitete nie wirklich jemand 45 Stunden pro Woche in dieser Branche. Vielleicht tun es diese Executives die man niemals persönlich sieht aber die ihre *Hinter mir die Sinflut* Strategie verfolgen um einen *Hit* zu schaffen.

Die meisten Angestellten halten es keine 5 Jahre aus. Entweder arbeitest du an einem Titel den du nicht leiden kannst, oder du wirst gefeuert nach dem du dir deine Seele zerstört hast (*busting your balls*) um einen Titel zu prozieren damit die Spielebranche weiterhin an der harten Arbeit der kleinen Leute profitiert.

Mein Posting wirkt bitter, weil es bitter ist. Meine Karriere ist komplett zerstört, und als Resultat ist mein Ziel Nummer Eins diese Branche zu verlassen. Ich möchte Euch da draußen warnen vor den Risiken wenn man in der *Games Industry* arbeitet.

Ja, es gibt einige wenige (SEHR WENIGE) Entwickler-Studios da draußen die sich um ihre Angestellten ein wenig kümmern aber selbst die großen Namen in diesem business quetschen ihre Leute aus wie Zitronen (*run their people into the ground*). Und wenn die Leute ganz ausgequetscht sind, dann werden sie ausgespuckt um neuen Trotteln Platz zu machen.

Deine beste Chance, wenn du wirklich mit Computerspielen arbeiten willst, ist es dein eigenes Produkt zu machen. Denk dran, wenn es ein Erfolg wird dann bereite dich darauf vor dass oben erwähnte Spielefiremen tun werden was sie können um deine Idee zu stehlen, und du bist am Ende der Dumme.

Wenn Computerspiele wirklich das sind was du beruflich machen willst, dann mach es nebenher, zu deinen Bedingungen. Heutzutage ist es einfacher als je zuvor ein eigenes Spiel zu publizieren. Du machst vielleicht nicht das große Geld, du arbeitest vielleicht nicht für ein großes Studio oder einen großen Publisher (glaub mir, das hört sich weit besser an als es ist), aber wenigstens hast du eine professionelle Karriere und wirst (hoffentlich) mit Respekt behandelt und nicht wie eine Ameise unter einer Lupe.

EDIT

Ich werde nicht auf jedes einzelne posting einzeln antworten aber ich versuche ein paar Fragen zu beantworten. Ich merke dass machen Leute speziell abgestumpft scheinen und ich glaube das sind die Leute die gerade in den Markt eintreten, voll mit Träumen und Hoffnungen wie ich sie damals hatte. Ich glaube ich muss auf die selbe Weise antworten wie schon früher. Ich bin nicht in die Branche reingegangen weil ich mittelmäßig oder faul sein wollte. Ich hatte Leidenschaft, und wer das nicht glaubt ist ein Idiot.

Ich entschuldige mich dafür absichtlich unklar gewesen zu sein, aber ich muss irgendwie meine Gehaltschecks verdienen.

1) Mein technischer Uniabschluss war kein Online-Studium.. er war von einer beglaubigten, sehr begehrten Universität. Ich bekam eine akademisches Stipendium, studierte trotz einer Behinderung und arbeitete halbtags um mein Studium zu beenden. Niemand half mir dabei, ich musste alles alleine machen. Ich wuchs sehr arm auf und dass, denke ich, war der einzige Weg um aus diesem Zustand rauszukommen.

2) Meine Situation mag einzigartig für mich sein, aber ich kenne viele Andere deren Erfahrung einzigartig für sie waren, uns die endeten alle mit Burnout. Sie arbeiteten alle sehr hart für einen Traum, während es die Führungskräfte ganz oben unmöglich machten dass dieser

Traum in Erfüllung geht.

3) Der Grund warum ich in der 52 Stunden crunch time keine Medikamente nahm war dass ich keine Ahnung hatte dass ich so lange in der Arbeit festsitzen würde. Der Grund war dass ein Manager einen Outsourcing Entwickler anheuerte der unglaublich Mist baute (*screwed us all*). Ich hatte keinen Zugriff auf Medikamente. Und lass mich eins klarstellen, ich habe Angststörungen... das macht mich NICHT schwach oder weniger fähig als andere Leute. Ich bin ziemlich sicher ich arbeitete härter als die meisten Leute die ich kenne trotz meiner Behinderung und der Angststörungen (welche über die Jahre schlimmer wurden durch die Schwankungen des Computergame-Marktes).

4) Für ca. 5 Jahre bis zur großen Rezession hatte ich sogar einen milden Erfolg und war konstant bei einer anständigen Firma beschäftigt. Aber das ging zu Ende und ich schaffte es danach für eine sehr lange Zeit nicht durchgehend angestellt zu sein...was es noch schwieriger macht in dieser Industrie wieder Fuß zu fassen. Hier gilt dass du ein Versager bist wenn du nicht an erfolgreichen Titeln beteiligt warst.

5) Ich habe einige meiner Erfahrungen gar nicht aufgeschrieben, hauptsächlich weil sie sehr schmerzhaft für mich waren. Aber ich gewann einige Freunde in dieser Industrie und das Schlimmste ist das durch die ständigen Kündigungen dieses Freunde über die ganze Welt verstreut werden. Das erlaubt ein brauchbarers virtual network, aber es ersetzt nicht ein Netzwerk von echten Freunden die dich wirklich unterstützen. Wenn du so ein Freundesnetzwerk finden kannst, trotz all der Kündigungen, dann empfehle ich dir dringend dass Du diese Freundschaften ausbaust und pflegst.

6) Ich bin nicht hierher (**Reddit**) gegangen um ein allwissender Veteran zu sein oder so etwas. Ehrlich, ich kam her um meine Geschichte zu erzählen, zu erzählen dass ich von vielen anderen ähnlich Schauerhaftes hören musste, und das die

Erfolgreichen in dieser Branche die Techniker und die rücksichtslosesten Haie im Geschäft sind. Jede Industrie braucht mehr als nur Programmierer und höhere Manager. Es gibt Designer, Manager, Verwaltungsleute etc. etc.... nur weil jemand kein Programmierer ist macht ihn das nicht wertlos.

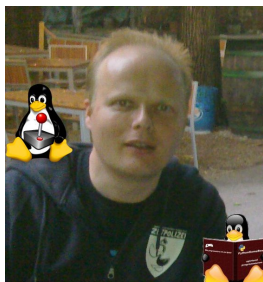
7) zur 45 Stunden Woche: Das passierte kurz nachdem ich zu Electronic Arts ging. Ich machte diese Erfahrung nicht selbst, sondern einer meiner Mitarbeiter. Ich denke es hörte recht schnell auf sobald die Leute merkten wie illegal es war. Ich denke es dauerte vielleicht ein paar Monate, aber der Punkt ist dass keine dieser Firmen es notwendig finden die Arbeitszeitregeln einzuhalten (none of those companies bother to follow). Ich selbst arbeitete niemals weniger als 45 Stunden pro Woche und ich arbeite sehr sehr hart.

Wie auch immer, lest aus meinem Posting heraus was ihr wollt. Manche Leute haben Glück in diesem hit-driven business, sehr sehr viele haben kein Glück. Ich wünschte ich hätte solch tolle Erfahrungen gehabt wie sie hier auf **Reddit** gepostet werden aber das ist mir nie passiert.

(Ende der Übersetzung)

Das Posting löste eine Reihe von Antworten aus, die man auf **Reddit.com** [1] direkt unter dem Original-Posting nachlesen und bewerten kann. Neben Ablehnung gab es auch Zustimmung.

Anmerkungen von Horst JENS:



Horst JENS. Bildrechte: [2] cc-by-sa

Als Anbieter von Programmierkursen für Jugendliche[2] und

Übersetzer des Postings möchte ich einige Anmerkungen beisteuern:

- Die Situation des Autors ist wegen der -für europäische Verhältnisse schlechten- U.S. Sozialversicherung nicht Eins zu Eins mit der Situation in Österreich vergleichbar.
- Nicht nur wer in der Spielebranche arbeiten will, einfach jeder sollte heutzutage programmieren (coden) können! Mit moderner Lernsoftware wie z.B. *Scratch* [3] ist das in kürzester Zeit möglich, z.B. innerhalb der (kostenlosen) Probestunde bei *spielend-programmieren* [2]!
- Ein guter Einstieg in die Spiele-Industrie ist es sicher, ein eigenes (Spiele)Projekt herzeigen zu können bzw. an einem großen (freien) Projekt mitgewirkt zu haben. Zum Beispiel eines der vielen Open Source Projekte auf Github.com oder ähnlichen Plattformen. Wer noch nicht so gut programmieren kann findet genügend Aufgaben um sich bei bestehenden Projekten einzubringen, z.B. als Übersetzer, Grafiker, Level-Designer, Bug-Reporter, Tester etc.
- Eine andere gute Idee für jeden Interessierten ist es an einem **GameJam** mitzumachen. Wer sich beim *spielend-programmieren Newsletter* [3] anmeldet wird über entsprechende Veranstaltungen im Raum Wien per Email informiert.
- Ich kann den im Posting erwähnten Tipp (...mach es nebenher; zu deinen Bedingungen...) nur bekräftigen. Ähnlich wie das Spielen eines Musikinstruments hat (Spiele)-Programmierung viel mit Kunst zu tun. Es gibt Künstler, die mit ihrer Musik genug Geld verdienen, aber Geld verdienen sollte nicht der einzige Beweggrund

sein ein Musikinstrument zu lernen. Spiele programmieren zu können ist eine Fähigkeit die unter Umständen zu einem Neben- oder gar Hauptberuf werden kann, so wie jedes Hobby. Es ist vor allem eine Fähigkeit mit der man sich selbst und anderen viel Freude machen kann.

Fachbegriffe

Reddit.com ist eine größtenteils englischsprachige, sehr beliebte Website welche zu zahlreichen Themen Diskussionsseiten (Webforen) anbietet, sogenannte Sub-Reddits. Im Gegensatz zu simplen Social-Media Seiten wie Facebook oder Google+ sind auf Reddit *Threads* möglich bei Kommentaren, außerdem können einzelne Beiträge und Kommentare sowohl hinauf- als auch hinunterge"voted"werden. Aufgrund des in beide Richtungen möglichen Voting-Mechanismus *schwimmen* beliebte Themen innerhalb einer Gruppe schnell nach oben und bekommen mehr Aufmerksamkeit. Die Registrierung auf Reddit ist kostenlos, und jeder User kann sowohl offene als auch private Subreddits anlegen. Reddit basiert auf *freier Software* [5].

Crunch-time (englisch, to crunch: zerbeißen, zermahlen) Zeit kurz vor der Veröffentlichung eines Spieles die von großem Streß gekennzeichnet ist und von den Programmierern Überstunden erfordert, da der Veröffentlichungstermin feststeht und letzte Fehler beseitigt werden müssen.

Thread wörtlich übersetzt Gesprächsleitfaden ist ein Feature von guten Email-Programmen, Usenet-Readern oder Webforen. Es bedeutet dass die einzelnen Nachrichten (bzw. Kommentare oder Emails) nicht streng chronologisch geordnet sind sondern dass eine Antwort B auf eine Nachricht A direkt (und meist nach rechts eingerückt) unter der Nachricht A steht. Siehe auch dieses Schaubild:

```

blaue Papageien
Re: blaue Papageien
  gelbe Elefanten (was: blaue Papageien)
Re: gelbe Elefanten
Re: blaue Papageien

```

Thread-Darstellung

Game Jam ist ein organisiertes Treffen bei dem Grafiker, Musiker und Programmierer zusammenkommen um gemeinsam in einer vorgegebenen Zeit (meist ein Wochenende) und zu einem vorgegebenem Thema Computerspiel(e) zu entwickeln. In Österreich wird jährlich im Rahmen des *Global Game Jam* [6] der *Austria Game Jam* [7] organisiert, zu dem

sich jede(r) Interessierte(r) anmelden kann. Andere GameJams werden rein über das Internet organisiert.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
 spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/redditrant
Feedback ✉ horst.jens@spielend-programmieren.at

Lizenz, Quellen:



Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] goo.gl/ivfVkJ
- [2] spielend-programmieren.at
- [3] scratch.mit.edu
- [4] goo.gl/wHP0ID
- [5] github.com/reddit
- [6] globalgamejam.org
- [7] austriagamejam

Samsung macht die Schule smart?

von MARTIN MAYR

Martin Mayr macht sich in seinem *Blogposting* [1] vom April 2013 ein wenig lustig über die Samsung Smartschool Initiative im Bildungsbereich. Originaltitel: *The future of education, powered by Samsung*. Martin lebt in Wien und beschäftigt sich mit Ernährungssouveränität, Urban Gardening sowie Montessori-Pädagogik. Er betreibt einen Blog und ist manchmal zu Gast im *Biertaucher-Podcast* [2].



Martin Mayr. Bildrechte: Horst JENS[3]
cc-by-sa

(Beginn des Original-Textes)

Eine Antwort auf einen *Futurezone Bericht über Smartschools* [4].

Wie soll das Klassenzimmer der Zukunft aussehen, wie soll es funktionieren und wie stellen sich Schüler den Unterricht vor? - Das frage ich mich auch schon lange. Das Spektrum der Lösungen soll nun von zwei relativ neuen Akteuren [5] gefunden werden: *Futurezone* und Samsung starten den Ideenwettbewerb *Samsung Smart School*, der vom Bundesministerium für Unterricht, Kunst und Kultur unterstützt wird.

Der Technologiekonzern lanciert so wie es aussieht eine internationale Kampagne (siehe *Leaflet* [6]) um seine Produkte in die Klassenzimmer der Welt zu bekommen, und in Österreich springen Medien wie Ministerien, im Angesicht der allgemeinen Ratlosigkeit bei der Bewältigung der vielfältigen Probleme im Bildungssystem, mit größter Freude auf.

Wie soll die Schule der Zukunft nun also aussehen?

„Österreichweit sind alle Volksschulen und alle Mittel- und Oberstufen eingeladen, sich Gedanken über die Schule der Zukunft, über den Unterricht der Zukunft, über mobile Lösungen und

den interaktiven Unterricht und selbstbestimmtes Lernen außerhalb des Unterrichts zu machen.“

Selbstbestimmtes Lernen soll aber außerhalb des Unterrichts stattfinden? Genau dort hat selbstbestimmtes Lernen schon immer stattgefunden und wird durch neue Technologien nicht zwangsläufig verbessert.

Als Erzfeind wurde vollkommen richtig der Frontalunterricht identifiziert, ob neue Lernkonzepte ihren Weg in den Schulalltag finden werden *bleibt aber offen* [4] Samsung schreitet nun gottlob als weiser Ritter heran, um uns von der kinder- verblödenden Kreidetafel zu erlösen. Dies erfolgt umsatzsteigernd durch Tablets, Pads, Slates oder PCs, in Kooperation mit Bundesministerium für Unterricht, Kunst und Kultur sowie futurezone.at.

Eine kleine Auswahl was uns in diesem Sinne erwartet: (Quellen: *Futurezone* [4], *Futurezone* [7])

- „...es [ist] nun möglich, dem Lehrer ... auf privatem Wege Fragen zu stellen“
- „...eine integrierte Lernplattform, mit der Lehrer und Schüler in einer interaktiven Umgebung arbeiten können.“

- "das Wort ... von einem Native Speaker vorgelesen ..."

Bleibt die Frage: Welche heute im Klassenzimmer abwesende Technologie hindert unsere Gesellschaft daran diese Erfahrungen den Schülern zu bieten?

Aber welche Umgebung soll nun laut Samsung den Schulalltag zum Besseren wenden? Die Futurezone Bildergalerie[4] gibt Aufschluss darüber: Das Ersetzen von grünen Tafeln, Kreiden und Schulbüchern durch **Smartboards** und **Tablets** soll die Lernerfolge unserer SchülerInnen verbessern (oder doch ein neues Geschäftsfeld erschliessen?).

Als 'integrierte Lernplattform' kommt mir das, mit einigem guten Willen, bekannt vor. Im *Montessori Kinderhaus* [8] wird so gelernt, jedes Kind mit einem Material, welches bis ins kleinste Detail wohlüberlegt konstruiert vom absorbierenden Geist verinnerlicht wird. Die perfekte vorbereitete Umgebung. Wenn nicht der Tablet die Repräsentanz aller Lehrmaterialien wäre, als denkbar ungeeignetste Möglichkeit Inhalte an Kindergartenkinder zu Vermitteln.

Leider ist so eine Umgebung für Kinder im Volksschulalter alles andere als geeignet. Montessori hat aus gutem Grund für jede Altersgruppe ein neues, aber auf ihren Erkenntnissen beruhendes Konzept erarbeitet. Und dass jedes Kind alleine vorm

Lehrmaterial sitzt gehört für dieses Alter jedenfalls nicht dazu. Ganz im Gegenteil, die Kinder erarbeiten ihr Wissen selbstständig, in Kleinstgruppen und nutzen dafür vielfältige Materialien und Quellen.

Samsungs Vision von der Schule der Zukunft aber bleibt eine Erweiterung des Bildungssystems wie wir es in den letzten Jahrzehnten beobachtet haben. Die Schule der Zukunft erfordert nämlich keine neuen Technologien sondern einen grundlegend neuen Zugang zu der Art und Weise wie Bildung vermittelt wird. Und warum dieser gerade von einem Technologiekonzern in die Welt gebracht werden soll muss zumindest ausgiebig hinterfragt werden.

Eine andere Antwort [9] liefert Stephen Hughes, der als Kinder-Neuropsychologe vielleicht ein klein wenig qualifizierter ist, sich zu diesem Thema zu äußern.

Fachbegriffe:

Smartboards oder **digitale Whiteboards**: Elektronische Schultafel, meist eine Kombination aus vertikalem Bildschirm bzw. Projektorfläche für Beamer und geeigneten Eingabemedium in Form eines Stiftes sowie spezieller Software. Statt den Computer per Maus oder Touchpad zu bedienen kann an der vertikalen 'Tafel' am Computer wie mit Kreide 'gemalt'

werden.

Tablet: Jeder flache, nur aus berührungsempfindlichem (Touch)-Bildschirm bestehender Computer, z.B. Apple's iPad

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001

Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/smartschool

Feedback ✉ martindermayr@gmail.com

Lizenz, Quellen:



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen

4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] <http://goo.gl/1zMxod>
- [2] <http://biertaucher.at>
- [3] spielend-programmieren.at
- [4] <http://goo.gl/Q42rbW>
- [5] <http://goo.gl/WrHB4i>
- [6] <http://goo.gl/Ev9j0u>
- [7] <http://goo.gl/c6v2BA>
- [8] montessorie-kindergarten.at
- [9] <https://vimeo.com/40791188>

C3 inspire

von ASSEM SHARMA



C3 Inspire [2]

Folgendes Interview habe ich (*Aseem Sharma* [1]) mit Alim Maherali, dem Gründer von *C3 Inspire* [2], geführt und habe dadurch einiges gelernt darüber wie man eine Organisation aufbaut

und erfolgreich führt basierend auf den *Open-Source* Prinzipien Zusammenarbeit, Vernetzung und Teilen (*collaboration, connections, and sharing*)

Dieser Artikel von erschien am 17. Dezember 2013 in *opensource.com* unter einer cc-by-sa Lizenz und wurde daher ohne eine Antwort auf Rückfrage abzuwarten von Horst JENS ins Deutsche übersetzt und republiziert.



Aseem Sharama. Bildrechte: [1]

(Beginn der Übersetzung)

Open Source als Grundlage von Startup Firmen

Tatkräftige und motivierte Leute zusammenzubringen um Startups zu gründen welche eine positive Auswirkung auf die Gesellschaft haben ist ein großartiges, visionäres Unterfangen. Eines davon ist *C3 Inspire*, dessen Ziel es ist Unternehmer von morgen schon heute zusammenzubringen und ihnen eine Plattform und Umgebung zu schaffen auf der sie ihre besten Fähigkeiten einbringen können für gemeinsame, größere Aufgaben.

Im Interview erzählt Alim über eine neue Startup-Businesskultur mit einem vorherrschenden Bewusstsein für den Sinn von Zusammenarbeit und dem Einsatz von free/libre Open Source Software.

Interview:

Aseem: Welche Notwendigkeit in der Welt und im Ausbildungsbereich führte Dich dazu *C3 Inspire* zu gründen? Welche Prinzipien leiteten Dich, welche Mission verfolgst Du mit *C3 Inspire*?

Alim: Ich gründete *C3 Inspire* wegen der Lücken im Ausbildungswesen die ich selbst erfahren hatte als Undergraduate (Bachelor) Student in Waterloo Ontario (Kanada). Ich hatte die einmalige Gelegenheit der erste Absolvent eines 2-Studien-Programms zu sein mit Abschlüssen von der Universität Waterloo und der Wilfrid Laurier Universität. Als ein Student der sowohl eine business school als auch eine technische Universität gleichzeitig besucht merkte ich Herausforderungen und Fähigkeitslücken denen sich alle Studenten beider Universitäten gegenübersehen:



Ontario, Kanada. Bildrechte: Tubs [3], cc-by-sa

Technik-Studenten von der Uni Waterloo fanden es z.B. schwer ihre unternehmerischen Ideen zu verwirklichen wegen dem fehlendem Business-Bereich, z.B. in Finanz und Marketing. Daran mangelte es den Studenten an der Wilfrid Laurier Uni nicht, die konnten unglaublich gut Business-Pläne zusammenstellen aber sie hatten Schwierigkeiten damit eine Webseite oder ein eCommerz zu erschaffen weil ihnen dazu die technischen Fähigkeiten fehlten.

C3 Inspire wurde gegründet um Studenten mit unterschiedlichen Fähigkeiten und von unterschiedlichen Studienrichtungen zusammen zu bringen damit sie sich gegenseitig dabei helfen können ihre Ideen zu verwirklichen.

Ein anderer Grund *C3 Inspire* zu gründen war der dass sich in jeder der beiden Universitäten die besten und scharfsinnigsten Studenten nicht miteinander trafen ...erst sehr spät in ihrem Studium. Es gab keine Infrastruktur für die besten Studenten einer Uni um zusammen zu kommen. Konsequenterweise bedauerten viele Studenten rückblickend dass sie sich gegenseitig gerne viel früher besser kennengelernt hätten um gemeinsam Projekte zu verwirklichen.

Schlussendlich: Wirklich jeder dem ich in der Schule über den Weg gelaufen bin

wollte etwas Sinnvolles (mit seinem Leben) machen ...etwas das die Welt zu einem besseren Platz macht. Doch nur einigen wenige taten das wirklich.

Viele Studenten wollten *später* im Leben etwas Sinnvolles tun nachdem sie sich ein paar Jahre lang finanziell abgesichert hatten. Leider schaut

die Wirklichkeit so aus dass wenn die Jahre vergehen, es immer schwieriger wird die Welt zu verändern. Bei *C3 Inspire* wollen wir *Soziales Unternehmertum* fördern und anderen Studierenden zeigen dass es möglich ist die Welt zu verbessern und gleichzeitig eine bezahlte Karriere zu machen.

Ich war als *undergraduate student* mit all diesen Herausforderungen konfrontiert, und nachdem ich meinen Abschluss gemacht hatte hörte ich Jahr für Jahr wie sich die Studenten über die gleichen Schwierigkeiten -wie ich sie hatte- beklagten. Nachdem ich die selben Klagen von so vielen verschiedenen Studenten gehört hatte dachte ich mir, dagegen muss man etwas unternehmen, und damit war die Idee zu *C3 Inspire* geboren. Heute ist jeder *C3 Event* so strukturiert um genau diese Probleme anzugehen!

Aseem: Inwieweit verwenden die Startups mit denen du arbeitest freie Software und Open Source Technologien bzw. Open Source Methoden?

Alim: Open Source Lösungen liegen jedem Startup zugrunde mit dem ich heute zu tun habe. In Fakt, viele Startups verwenden Open Source Technologien für fast alles. Es ist nicht ungewöhnlich für neue Firmen dass sie *Wordpress* für ihre Website benutzen, *Joomla* oder *Drupal* für ihr **CMS**, *SugarCRM* für ihr **CRM**, und *TRAC* für ihr **Issue-Tracking System**.

Unternehmertum bekommt mehr und mehr *kollaborativ*. Die Tage der geheimniskrämerischen Startups welche niemandem erzählen was sie vorhaben sind vorbei. (Junge) Unternehmer verstehen dass die Vorteile von Zusammenarbeit und Ideen-Austausch bei weitem das Risiko aufwiegen, jemand könne ihre Ideen stehlen. Das ganze Unternehmer-Ökosystem gleicht sich mehr und mehr dem Open-Source Ökosystem an, wo die Menschen wissen dass Zusammenarbeit, etwas beitragen (to contribute), allen hilft. Also trägt jeder etwas bei. Es gibt da eine gemeinsame Strömung in fast allen unternehmerischen Aspekten, von Gemeinschaftsbüros (shared spaces) zur

Standardisierung von Risikokapital-Deals.

Aseem: Was ist deine Beziehung zur open source Technologie? Benutzt Du freie Software innerhalb Deiner Organisation? Auch draußen, im Feld?

Alim: Ich sehe freie Software als einen *Befähiger* für Menschen. Freie, Open Source Software erlaubt Menschen finanzielle Hürden zu überwinden und gibt ihnen die Chance zum Testen, zum Entdecken und zum Erfinden die sie sonst nicht hätten. Nachdem das gesagt ist; ich habe enormen Respekt vor kommerzieller Software. Es gibt definitiv Situationen wo eine kommerzielle Softwarelösung mehr Sinn macht als eine Open Source Lösung, aber beide Arten von Lösungen haben ihren Platz und dienen Menschen und Organisationen. C3 Inspire benutzt eine breite Auswahl an Open Source Lösungen von Wordpress über **GitHub** zu SugarCRM.

Aseem: Welche Rolle spielt Deiner Meinung nach Open Source in der Zukunft der Bildung?

Alim: Open Source Technologien werden weiterhin ein kraftvoller Motor für den Bildungssektor sein durch die verstärkte Anwendung von **eLearning** Techniken, speziell in Entwicklungsländern. Software wie *Moodle* ist mittlerweile sehr robust und dramatisch positive Resultate sind möglich wenn man Lösungen implementiert in Orten wo bisher Kosten eine Hürde darstellten. Noch wichtiger: Open Source Software ist der Bereich wo Erfinder neue Ideen erträumen und austesten. Indirekterweise ist dadurch der Einfluss von Open Source Lösungen auf Leute im Bildungsbereich außerhalb der traditionellen Bildungsinsti-

tutionen ganz schön gewaltig. Meiner Meinung nach werden Open Source Lösungen Bildung und Gesellschaft sehr verändern in den kommenden Jahren.

Aseem: Inwieweit unterscheidet sich C3 Inspire gegenüber anderen Organisationen und Programmen?

Alim: C3 Inspire unterscheidet sich in 3 wesentlichen Punkten von anderen Organisationen.

Erstens, C3 Inspire bringt Studenten von verschiedensten Universitäten zusammen in Programmen mit unterschiedlichen Fähigkeiten.

Zweitens, C3 Inspire bemüht sich gleichermaßen um die Gründung von sozialen Initiativen wie um die Gründung von (business) Startup's.

Drittens bringt C3 Inspire Unternehmer, OpenSource Aktivisten und erfahrene Profis zusammen um den Studenten zu ermöglichen ihre eigenen Ideen zu realisieren. Viele andere Organisationen bringen nur Unternehmer zusammen die schon ihre eigenen Ideen haben und deshalb nicht sehr auf Zusammenarbeit aus sind.

Ich glaube es ist die Kombination dieser 3 Elemente die den Studenten wirklich hilft ihr Potential zu entfesseln.

Aseem: Wie genau arbeitest Du mit Startups?

Alim: Genau wie in der Open Source Community ist Zusammenarbeit auch die Grundlage von unseren Werten bei C3 Inspire. Wir machen was wir können um andern Startups die mit uns zu tun haben zu helfen und mit ihnen zusammen zu arbeiten: Wir haben ein Forum wo Leute sich treffen um neue Organisationen gemeinsam zu gründen; wir stellen Startups in unserem Blog vor, wir lassen Startups auf unseren Veranstaltungen Vorträge halten. Wir mögen

Startups und helfen ihnen gerne.

Fachbegriffe

Open Source Quelloffene (freie) Software. Mit Open Source Prinzipien sind im Artikel der Austausch von Wissen (Sourcecode), Offenheit und Zusammenarbeit gemeint.

CMS: Content Management System, (Inhalts-Management) z.B. ein Wiki (wie Wikipedia) oder ein Blog.

CRM: Customer Relationship Management System, Kundenbeziehungsverwaltung. Im Prinzip eine Kontakt- bzw. Kundendatenbank.

Issue Tracking: auch Trouble-Ticket genannt. Eine Datenbank um Programmierfehler oder Anforderungen an Software zu verwalten.

e-learning elektronisches Lernen: Lernen bei dem moderne Medien (Internet) eingesetzt wird.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/c3inspire
Feedback ✉ horst.jens@spielend-programmieren.at

Lizenz, Quellen:

Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] aseemsharma.info/about-me
- [2] c3inspire.com
- [3] <http://goo.gl/rylcEz>

Ein Jahr danach: Westcliff High School for Girls migriert zu Linux

von STU JARVIS UND MALCOLM MOORE

Im Sommer 2012 begann die *Westcliff High School for Girls Academy (WHSG)* [1] im Südosten Englands ihre Schülercomputer auf Linux (*KDE Desktop*) umzustellen. Der Netzwerk-Manager der Schule, Malcom Moore, kontaktierte *dot.kde.org* damals. Ein Jahr danach führte *Stuart Jarvis* [2] folgendes Interview mit Schuladministrator Malcolm Moore um herauszufinden wie es ihm und den Studentinnen geht in einer Welt ohne Windows.



Stu Jarvis [2]

Dieser Artikel von *Stu Jarvis* [3] erschien am 4. Juli 2013 auf *dot.kde.org* [4] und wurde am 31. Juli 2013 auf *opensource.com* [5] unter einer creative-commons Lizenz publiziert. Übersetzung von *Horst JENS* [6], mit freundlicher Genehmigung des Autors.

(Beignn der Übersetzung)

Interview

Stu: Hallo Malcom, danke für deine Bereitschaft zu diesem Interview. Kannst du uns ein wenig über die Schule und deinen Job erzählen?

Malcolm: Die *Westcliff High School for Girls Academy* ist eine *selective Grammar School*⁹ mit einer 6. Schulform mit ungefähr 340 Schülerinnen. Gegründet 1920 als eine gemischte Schule (Burschen und

Mädchen) in der *Victoria Avenue / Southend*, zog sie auf 1931 auf den heutigen Platz und unterrichtet derzeit 1.095 Mädchen.

Der IT Support besteht aus drei Angestellten: ich selbst, Paul Anonelli und Jenny Lidbury. Meine Rolle ist die des Netzwerk-Managers. Der IT Support ist zuständig für die Wartung und Anschaffung von allem was in der Schule mit Computern zu tun hat. Dazu zählen 200 Lehrer-Computer, über 400 Schüler-Computer, 33 IMacs, über 100 Laptops und ein paar Android Tablets. Wir warten auch alle Multimedia-Geräte wie Beamer, interaktive Whiteboards, Fernsehgeräte etc.

Stu: Wessen Idee war es die Computer auf Linux umzustellen? Was waren die Gründe dafür?

Malcolm: Wir haben schon länger Linux benutzt als das Betriebssystem für unseren Email Server, für unser *VLE (Virtual Learning Environment)* und für unsere Homepage und ich hatte schon in meinem vorigen Job in der Finanzindustrie Linux verwendet.

Es war meine Idee die Schüler Computer auf Linux umzustellen denn es wurde mehr und mehr klar dass die Größe, Komplexität und Kosten der IT exponentiell wuchsen. Etwas musste getan werden und mein Stolz verbietet mir bei der Qualität der Maschinen die wir warten zu sparen.

Wir begannen zu testen mit einer kleinen Gruppe von 60 Computer, bekamen Rückmeldungen von den Schülerinnen, machten notwendige Adjustierungen, testeten erneut und so weiter. Nachdem wir diesen Test-Zyklus mehrmals wiederholt hatten mit Red Hat / Fedora und mit Suse/OpenSuse Konfigurationen und damit zufrieden waren reichte ich meinen Vorschlag beim Schulträger (Senior Leadership Team) ein.

Die Motivation dafür war ursprünglich sowohl Kosten als auch Philosophie: Selbst in einer ausgezeichneten Schule ist das Budget immer limitiert (Politiker kapierten einfach nicht die Volksweisheit: Wenn du glaubst Bildung ist teuer, probier

Ignoranz). Die Kosten Windows zu benutzen sind hoch aber nicht immer klar ersichtliche. Windows bringt einen Haufen Gepäck mit sich was die Kosten gegenüber einem Linux System deutlich in die Höhe treibt. Die philosophische Motivation war wahrscheinlich die Philosophie des Pragmatismus.

Wir wollten das bestmögliche IT System und die bestmögliche Bildung mit dem vorhandenen Budget anbieten.

Anstatt Geld aus zu geben um im Endeffekt bei den Schülern Werbung für Microsoft Windows und Microsoft Office zu machen geben wir das Geld lieber für altmodische Dinge wie Lehrkräfte und echte Bildung aus.

Glücklicherweise verwarf um diese Zeit die Regierung von Großbritannien den EDV-Lehrplan (welcher hauptsächlich daraus bestand den Schülern beizubringen wie man Microsoft Office verwendet) und sagte den Schulen sie sollen einen mehr Computerstudien-basierten Lehrplan (go for a more computer studied-based syllabus) verwenden. Wir rannten sozusagen offene Türen ein.

Stu: Gab es Widerstände gegen die Linux-Migration und wie bist du damit umgegangen?

Malcolm: Überraschenderweise sehr wenig. Das Schulträger-Komitee „verhörte“ mich zwei langen Sitzungen, das war ein Spaß!

Sobald du einen Schritt zurückgehst von der falschen Vorstellung Computer = Windows und beginnst ernsthaft nachzudenken überwiegen die Vorteile glasklar die Nachteile. Die Welt ändert sich sehr schnell. Da gibt es eine Studie aus dem Jahr 2000 die besagt dass 97% der Computer weltweit Windows installiert haben. Aber heutzutage, mit Tablets, Smartphones etc. ist der Windows-Anteil auf nur noch 20% geschrumpft. In der Welt von *Big Iron* herrscht Linux unangefochten. Wir

⁹entspricht ungefähr einem Gymnasium

als Schule sind spezialisiert auf Wissenschaft und Technik und möchten dass unsere Schülerinnen später Großes bewirken, wie das nächste Google gründen oder das Universum von CERN zum Einsturz zu bringen. In diesen Umgebungen werden die Schülerinnen Wissen über Linux brauchen.

Stu: Was für eine Wahl hast du bezüglich Software getroffen und warum? Musste neue Hardware gekauft werden?

Malcolm: Wir überlegten und dass die Schülerinnen das Interface mögen müssen also war *pretty is a feature* eine Bedingung für die Schülercomputer. Für uns von der IT Abteilung ist Stabilität das Wichtigste für die Server. Obwohl ich weiß dass viele Leute da draußen sind die unterschiedliche Linux-Distributionen mögen, kenne ich selbst mich eigentlich nur mit den **RPM** basierten aus. Hätten wir die Ressourcen gehabt hätten wir uns andere Distributionen genauer angeschaut, aber wir probierten nur *Red HAT / Fedora* und *Suse / openSuse*. Am Schluss hat Suse/openSuse gewonnen wegen deren KDE Software Support. Zuerst wollten wir nicht den Schülerinnen zu viele Änderungen zumuten und KDE's **Plasma** kann so hergerichtet werden dass er sehr ähnlich aussieht wie (von Windows?) gewohnt. Danach, während der Test-Phase, forderten wir die Schülerinnen auf sowohl KDE Plasma als auch **GNOME** auszuprobieren. Plasma gewann mit Abstand bei der Beliebtheit. Die endgültige Entscheidung für Software war openSUSE 12.2 und Plasma Desktop 4.10.

Was die **Workstations** (Schüler-PC's?) angeht war keine neue Hardware notwendig. Einer der Hauptgründe warum wir Linux wollten war weil Linux gut auf älteren PC's läuft.

Der übliche Zirkus, 400 Schüler-PC's alle drei oder vier Jahre zu ersetzen verursacht horrenden Kosten

. - Viele Schulen können sich das in Zeiten der Krise nicht erlauben. Mit der Performanz die wir jetzt haben

will ich unsere Maschinen so lange laufen lassen bis sie auseinanderfallen! Ich rate aber jedem sicherzustellen dass sie ein gutes Netzwerk haben bevor auf Linux umgestellt wird.

Stu: Wie war die Umstellung? Gab es technische Probleme und wie wurden sie gelöst?

Malcolm: Die Umstellung auf Linux erfolgte während der Sommerferien 2012. Zu diesem Zeitpunkt rannten wir in keine signifikanten technischen Probleme, aber das soll nicht heißen das wir nicht später welche bekamen!

Stu: Gab es im Vergleich zum alten System Software die ihr vermisst habt?

Malcolm: Zur Zeit arbeiten unsere Schülerinnen mit Linux und die Lehrkräfte mit Windows7. Gäbe es bei uns einen SIMS (Schools Information Management System) Client für Linux dann wäre die Umstellung der ganzen Schule einfach zu bewältigen. So wie es jetzt ist hätten wir damit über das Ziel hinausgeschossen. Im Educational Software Bereich fehlt uns bei Linux nichts, aber wir haben ein paar Windows Programme die wir unter **WINE** laufen lassen damit die Schülerinnen ihre laufenden Projekte in Ruhe abschließen können und dann langsam auf andere Software umsteigen können.

Erwähnenswert ist der Einsatz von **Raspberry Pi** und ähnlichen Geräten in Schulen. Das Raspberry Pi team sagt dass ein Vorteil beim Einsatz ihres Computers ist dass die Studenten damit experimentieren können ohne den PC der Schule (bzw. den der Eltern) kaputt zu machen. Dank Linux können die Studenten herumexperimentieren.

Unser *ICT department* bietet bereits Programmierunterricht für Schüler ab 11 Jahren an, und mit unserem System ist das Schlimmste was passieren kann dass sich die Kinder den eigenen Account kaputt machen. Selbst wenn sie den komplett zerstören kann er innerhalb von Minuten wiederhergestellt werden und die nächste Person die an dem selben Computer arbeitet wird davon nicht tangiert.

Stu: Hast Du KDE oder openSuse wegen Unterstützung kontaktiert? Wie war die Reaktion?

Malcolm: Ich habe häufig die **Foren** sowohl von KDE als auch von openSuse benutzt sowie die **Bugzilla** Seiten; beide waren sehr hilfreich. Die openSUSE Foren können manchmal ein bisschen ruppig sein wenn Leute dort finden dass eine Frage schlecht formuliert ist und nicht ganz durchdacht. Dies ist zum Glück bei den KDE Foren nicht so, dort ist jeder sehr hilfreich und höflich. Bei openSUSE muss man dazu sagen dass manche der Fragen die ich dort gepostet hatte nicht klar formuliert waren. Leider gibt es wie gesagt nur drei Leute in unserer Abteilung und **RTFM** ist keine realistische Option für uns. Ein Tag hat einfach nicht genug Stunden. Wenn ich etwas posten kann -selbst wenn es für manche dumm klingt- dann ist das eine großartige Hilfe. Wenn wir alles über Linux erst lernen müssten dann würde es dieses Migrationsprojekt niemals geben, wir würden immer noch **RTFM**!

Obwohl ich manchmal als Idiot beschimpft wurde bekam ich gute, brauchbare Antworten auf alle meine Fragen, sodass ich die Foren jedermann empfehlen kann. Manchmal ist es notwendig nicht allzu dünnhäutig zu sein.

Stu: Wie denken Schülerinnen, Eltern und Lehrer über den Wechsel zu Linux?

Malcolm: Jüngere Schülerinnen akzeptieren es als normal. Ältere Schülerinnen sind etwas weniger flexibel. Es gibt ein paar wenige die sich an Microsoft Word klammern. Lehrer sind genau so -überraschenderweise hat es nichts mit dem Alter zu tun. Manche finden es in Ordnung und manche hassen es. Dies gesagt habend, eine gleich große Anzahl Lehrer hasste Windows 7 und kein einziger Lehrer mochte Windows 8. Ich denke das grundlegende Problem ist das Windows XP ein Opfer des eigenen Erfolgs wurde. Es funktioniert ausreichend gut vom Benutzerstandpunkt aus betrachtet und es ist schon seit ewigen Zeiten da. Außer-

dem mögen die Leute generell keine Veränderungen, seltsamerweise auch manche Schülerinnen nicht.

Sobald wir uns entschieden hatten die Umstellung auf Linux zu beginnen sendeten wir einen Rundbrief an alle Eltern. Wir haben vielleicht weniger als ein halbes Dutzend Schreiben erhalten von Eltern die nicht einverstanden waren und meinten dass Office Skills nützliche Skills wären. Ich akzeptiere deren Ansichten, aber ich argumentiere dass eine heute 11 Jahre alte Schülerin die im September 2014 bei uns anfängt wird den Arbeitsmarkt nicht vor 2024 betreten wird. Wie wird Office 2024 ausschauen ? *Your guess is as good as mine.* Gute Grundlagenkenntnisse und ein logischer und analytischer Umgang mit Computern werden lebenslang nützliche Fähigkeiten sein.

Stu: Im Rückblick ein Jahr danach, was hat funktioniert und was hat nicht geklappt ? Was würdest du anders machen oder anderen Schulen raten anders zu machen ?

Malcolm: Es wäre schön wenn ich sagen könnte alles hat perfekt geklappt, aber so war es nicht. Das erste Halbjahr nach der Umstellung war fürchterlich. Das Hauptproblem war (mangelnde) Geschwindigkeit des Systems und das es zu lange dauerte sich in KDE Plasma einzuloggen. Unsere Tests rannen mit nur 60 Maschinen denn es war schwierig so viele Schülerinnen während der Mittagspause aufzutreiben um das System so schlimm wie möglich zu testen. Außerdem hatten wir während dieser Test-Phase noch 400 Schülerinnen PC'S mit Windows XP zu betreuen.

Zusammenfassung: Linux rennt sehr gut auch auf alten Kisten, aber wenn man **LDAP authentication** und **NFS home directories** haben will - so wie in jeder besseren Schule oder in einer mittleren Firma - dann braucht man ein gigabit Netzwerk. Mit nur 100Mb funktioniert es gerade noch, aber es ist eine unbefriedigende Erfahrung wie wir zu unserem Leidwesen feststellen mussten. Schließlich mussten wir acht **Netzwerk-Switches** ersetzen um das

gesamte Netzwerk auf Gigabit Bandbreite zu verbessern. Die Ersetzung der Switches war zwar sowieso geplant, aber ich hätte das gerne ohne Zeitdruck später gemacht.

Außerdem kooperieren manche Dinge im KDE nicht von selbst mit NFS home directories - ich weiß dass dieses Problem bei späteren Versionen zu einem gewissen Grad gelöst wurde. Wir haben mehrere Skripts gemacht die auf unseren Servern rennen und die gewisse KDE Optionen erzwingen um das Netzwerk zu entlasten. Nachdem wir das geschafft hatten während der Semesterferien beruhigte sich die Situation dankenswerterweise.

Stu: Was könnte KDE tun um so eine Umstellung einfacher zu machen ?

Malcolm: Dokumentation ! Man kann KDE über das **GUI** unglaublich genau konfigurieren, aber Admins müssen *Defaults* (Vorgabewerte) für alle User setzen. Die *openSUSE Defaults* sind o.k. für Einzelplatzrechner, aber sie müssen angepasst werden für einen Schulbetrieb. Wir machten dies indem wir uns einen Rechner nahmen, Änderungen machten und dann durch alle *dot Dateien* durchgingen um zu sehen welcher davon betroffen war. Außerdem löcherten wir Ben Cooksley im KDE Forum mit Fragen. Es war harte Arbeit ! Das Problem hierbei - und ich glaube es gibt keine einfache Lösung dafür - ist dass mit Linux, und teils auch mit Windows - die Technologie sich in manchen Fällen so schnell weiterentwickelt dass die Dokumentation veraltet ist noch bevor sie überhaupt ausgedruckt wurde.

Stu: Welche Programme (KDE und andere) waren beeindruckend? Wo fehlt noch was ?

Malcolm: Ausgenommen ein halbes Dutzend Schülerinnen die **GNOOME** benutzten: Jeder ist begeistert von der Möglichkeit Desktop und Programme selbst zu konfigurieren. Die meisten Admins sperren dieses Feature bei Windows weil sich User - speziell Studenten - damit recht schnell das System kaputtkonfigurieren können. Wir sind der Ansicht dass

wir zurück wollen zum Begriff des persönlichen Computers (PC). Wenn Schülerinnen ihren PC konfigurieren können so viel sie wollen gibt ihnen dies ein Gefühl von Eigentum über ihren Desktop. Wir haben Restriktionen gemacht damit man eine Maschine nicht unbenutzbar machen kann. Die Schülerinnen bekommen generell (wenn sie sich verkonfiguriert haben) einen Desktop Reset bevor es ein *Gespräch* gibt.

So viel Freiheit zuzulassen ist ein ungewohnte Idee für Schulen

. Am Anfang wurden manche Desktops wahrlich kaputtkonfiguriert. Nachdem dies den Reiz des Neuen verloren hat schauen die Desktops vernünftiger aus, und wir mussten schon monatelang keinen Reset mehr machen. In dieser Hinsicht ist es ein großer Erfolg dass die Schülerinnen nun Verantwortung für ihre eigene Arbeitsumgebung übernehmen und dass sie solche Aufgaben erledigen anstatt gesagt zu bekommen: "Da hast du Windows und Office. Benutze es!"

Stu: Andere Kommentare, Beobachtungen oder Erfahrungen?

Malcolm: Hatte ich schlaflose Nächte ? Ja. Wurde ich fast wahnsinnig? Ja. Würde ich es nochmal machen? Sofort!

Stu: Vielen Dank und auch in Zukunft viel Erfolg!

Zusammenfassung

Das Beispiel der Westcliff High School for Girls Academy gibt uns viel zum Nachdenken. Linux, mit KDE Software, kann klar in solch einer Umgebung arbeiten, aber es gibt noch Schwierigkeiten beim Einsatz und bei der Gewöhnung an den neuen System. Die Erfahrungen von Malcolm unterstreichen die Bedeutung der KDE Foren im Bereich willkommen heißen und Unterstützen neuer Mitglieder um freie Software zu größerem Publikum zu verhelfen.

(Ende der Übersetzung)

Fachbegriffe

Windows XP existiert seit 2001 am Markt.

KDE K Desktop Enviroment, eine Sammlung verschiedenster, gut zusammenpassender Linux Software inklusive eines Desktops.

C.E.R.N Europäisches Kernforschungszentrum für physikalische Grundlagenforschung in der Nähe von Genf. Bekannt durch den riesigen Teilchenbeschleuniger.

RPM stand für *RedHat Package Manager* und ist heute ein über mehrere Linuxdistributionen verbreiteter Standard zur Bereitstellung von Linux-Software. Libre-Office oder andere Software für Linux wird z.B. direkt auf der entsprechenden Homepage als rpm-Paket angeboten, was Download und Installation für rpm-baiserte Linuxdistributionen (Redhat, Suse, Fedora..) erleichtert.

Plasma: Eigentlich *KDE Plasma Workspaces* ist eine Weiterentwicklung von KDE.

Gnome Softwaresammlung für Linux inklusive Desktop. Steht in Konkurrenz zu KDE und anderen, leichteren Desktops. Bei jeder Linux-distribution ist es mögliche sich entweder beim Start für einen installierten Desktop (z.B. Gnome oder KDE) zu entscheiden bzw. die Programme gemischt einzusetzen, z.B. einen KDE-Testeditor in einem Gnome-Desktop zu verwenden.

Wine steht für *Wine Is No Emulator* und bietet eine Laufzeitumgebung unter Linux an mit der es möglich ist viele Windows-Programme direkt aus Linux heraus zu starten, wo sie dann in einem Fenster laufen.

Raspberry Pi sehr billiger Computer (ca. 30 Euro) welcher extra für den Schulunterricht entwickelt wurde und unter Linux läuft. Wegen dem geringen Preis, kleinen Abmessungen (ungefähr so groß wie eine Zigarettenschachtel) und den niedrigem Stromverbrauch sehr beliebt bei Bastlern und Do-it-Yourself Pro-

jekten. Wird unter anderem bei *Programmierungskursen der Firma spielend programmieren* eingesetzt.

(Web)Forum Diskussionsforum im Internet, die das öffentliche posten von Nachrichten erlauben sowie das ebenfalls öffentliche Antworten auf ein Posting.

Bugzilla Eine Webbasierte Fehlerverwaltungs-Software (*Issue Tracker*) bei der man Programmfehler (engl.: bugs) eintragen und verwalten kann, z.B. um herauszufinden ob jemand schon eine Lösung für ein bestimmtes Problem gefunden hat oder daran arbeitet.

RTFM speziell in Webforen gebräuchliche Abkürzung. Wörtlich: *Read the fucking Manual* (schau im Handbuch nach).

LDAP steht für *Lightweight Directory Access Protocol* Gemeint ist ein über das Web erreichbarer Verzeichnisdienst der es z.B. ermöglicht Email-Adressbücher, Telefonverzeichnisse und ähnliches zentral zu verwalten. Ist in größeren Firmen und Institutionen üblich, damit nicht jeder Mitarbeiter selbst eine Datenbank aller internen Telefonnummern pflegen muss. Die im Artikel erwähnte *authentication* bezieht sich darauf dass jede Schülerin per Webbrowser auf diese Verzeichnisse zugreifen kann. Dazu muss sie sich üblicherweise mit Username und password anmelden, was vom LDAP-Server ermöglicht werden muss.

NFS home directories steht für *Network File System Heimatverzeichnisse* (manche Begriffe sollte man erst gar nicht übersetzen). Gemeint ist dass jede Schülerin ein *home-directory* hat auf der sie ihre eigenen Dateien abspeichern kann. Dieses Verzeichnis ist für sie immer an der gleichen Stelle erreichbar, egal von welchem der 400 Schulcomputer sie sich anmeldet (bzw. auch wenn sie sich von zu Hause aus anmeldet). Das Verzeichnis liegt in Wirklichkeit auf einem Schul-Server, zu welchem sich die Schülerin beim An-

melden am Computer verbindet. Der Service ist ähnlich, aber nicht identisch mit *Cloud-Computing* wie z.B. *Google Drive* oder *Dropbox*.

Netzwerk-Switch: Eine Art Verteilerkasten für Netzwerk-Kabel. Ähnlich einer Mehrfachsteckdose, nur eben nicht für Strom sondern für Lan-Kabel.

GUI bedeutet "Graphical User Interface", grafische Benutzerschnittstelle und umfasst alles was man mit Maus (bzw Touch) bedienen kann, anklicken und verschieben von Icons und Fenstern etc.

Spenden

Stu möchte selbst keine Spenden bekommen aber empfiehlt folgende Webseiten: jointhegame.kde.org und kde.org/community/donations

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001

Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/westcliff

Feedback ✉ stuart.jarvis@gmail.com

Lizenz, Quellen

Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] www.whsg.info
- [2] dot.kde.org/users/stuart-jarvis
- [3] www.asinen.org/about
- [4] goo.gl/4paq3n
- [5] goo.gl/Je8Hx1
- [6] [spielend-programmieren](http://spielend-programmieren.at)

Computer-Unterricht in Vietnam

VON NEIL FRASER

Neil arbeitet in den USA bei Google. In diesem *Blogposting* vom 16. März 2013 [1] berichtet er über seine Reise nach Vietnam und vergleicht den Computer-Science Unterricht in Vietnam mit jenem der USA. Anlässlich seines Besuches programmierte Neil die Lernsoftware *Blocky Maze* [3].



Neil Fraser. Bildrechte: Neil Fraser

Übersetzung aus dem Amerikanischen von Horst JENS [10] mit freundlicher Genehmigung des Autors. Die Bildrechte der Fotos waren nicht mit vernünftigen Aufwand abzuklären, aber alle Fotos wurden von Neil Fraser freundlichst zur Verfügung gestellt.

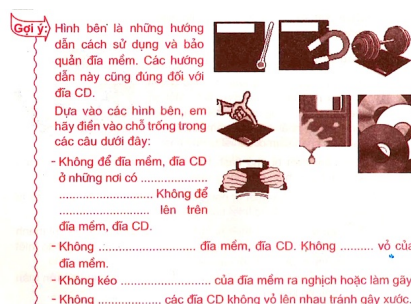
(Beginn der Übersetzung)

Während meinem letzten Besuch in Vietnam hatte ich Gelegenheit einige Schulen zu besuchen und zu sehen wie dort Computer Science (CS) vor Ort unterrichtet wird. Offiziell dürfen Ausländer nur mit Genehmigung des Unterrichtsministeriums Schulen betreten. Doch es ist wesentlich einfacher (und die Ergebnisse sind repräsentativer) wenn man einfach

unangemeldet auftaucht. Eine Visitenkarte von Google zu haben hilft.

Computer Science in Schulen in Vietnam und in den USA

Computer Science Unterricht startet in der 2. Klasse (second grade)¹⁰. Die Kinder beginnen mit den Grundlagen – inklusive dem korrekten Umgang mit Disketten.



Unterrichtsmaterial über Disketten. Bildrechte: Volksrepublik Vietnam

In der 3. Klasse lernen die Schüler Microsoft Windows zu benutzen. Vietnam ist zu 100 Prozent eine Windows XP Monokultur. Wahrscheinlich haben alle Windows-Lizenzen im Land die selbe Seriennummer. Verständlich, wenn man bedenkt dass eine Windows Lizenz soviel kostet wie ein Monatsgehalt.

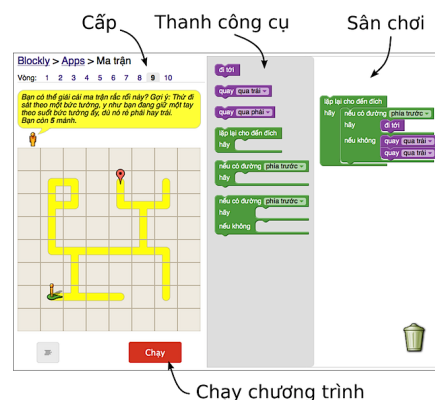
Tippen (10-Finger-System) wird gelehrt mittels Microsoft Word. Ganz Vietnam verwendet englischsprachige Software in Schulen, was das Tippen lernen für die Kinder in diesem Alter erschwert.

Ab der 4. Klasse starten die Kinder mit der Programmiersprache Logo. Zuerst mit Sequenzen von Befehlen, danach mit Schleifen (Loops). In der 5. Klasse schreiben die Kinder Prozeduren (Unterprogramme) welche Schleifen enthalten welche Unterprogramme aufrufen welche wiederum Schleifen enthalten.

Ein Vergleich mit den USA ist hier angebracht: Einige Besuche in San Francisco's 'magnet school for science and technology (Galileo Academy)' zeigten mir Schüler der 11. und 12. Klasse¹¹ die sich schwer taten damit den *image tag* in Html zu verstehen.

Computer Science Hausaufgaben waren von der Schuldirektion verboten worden.

Zu sagen dass ich beeindruckt war vom Computer Science Unterricht und Lehrplan in Vietnam's Grundschulen ist eine Untertreibung. Ich fragte was ich tun könne um zu helfen. Unerwarteterweise war die Antwort: *Software*! Education-Software existiert in vietnamesischer Sprache überhaupt nicht, und es gibt auch kein Geld um sie zu kaufen, selbst wenn sie existieren würde. Ich verbrachte den Rest meines Urlaubs damit, Software zu schreiben. Das Resultat ist *Blocky Maze* [2], welches in verschiedenen Sprachen läuft. Hier ein Link zur Online-Demo-Version von *Blocky Maze* [3]. *Blocky Maze* ist ein Selbst-Lern-Programm mit einer Reihe von Tutorials um Programmieren zu lernen. Verwendet werden Schleifen (loops) und Bedingungen (conditionals) sowie im Level 10 auch Listen, Variablen und Prozeduren. Ich musste alles auf eine CD brennen denn die Schule konnte sich keinen zuverlässigen Internetzugang leisten.



Blocky Maze, ein mehrsprachiges Programmierlernspiel. Bildrechte: Volksrepublik Vietnam

Die Übergabe der Software zwei Wochen später war ein surreales Erlebnis: Im Büro des Schuldirektors zu sitzen ist als Erwachsener genauso furchterregend wie als Schulkind. Von der Wand sah der allgegenwärtige *Ho Chi Minh* auf mich herab. Vor der Schule begannen

¹⁰Im Amerikanischen werden die Klassen fortlaufend gezählt. First Grade entspricht der ersten Klasse Volksschule, Fifth Grade entspricht der 1. Klasse Hauptschule / Gymnasium

¹¹entspricht 16-jährigen, Anm. d. Ü.

1.500 Kinder ihre Morgengymnastik, zum Teil bestehend aus dem *Chicken dance* [4].

Der Computer Science Lehrer war hoch erfreut und versprach mir Blocky schon am nächsten Tag im Unterricht einzusetzen. Wegen fehlendem Budget konnte sich die Schule keinen zweiten Computer Science Lehrer leisten, und nur die Hälfte der Kinder bekam Computer Science Unterricht. Ich fragte nach wieviel ein Computer Science Lehrer im Monat verdient: 100 Dollar. Daraufhin ging zum nächsten Bankomat und kaufte der Schule einen zweiten Computer Science Lehrer für ein Jahr.

Es stellte sich heraus dass man ziemlich populär wird wenn man Software und einen Lehrer spendet:



happy Neil, happy students. Bildrechte: Volksrepublik Vietnam

Wenn 5-grader in Vietnam mindestens genauso gut "performen" wie 11-grader in den USA, wie gut sind dann die 11-grader in Vietnam? Ich ging in eine CS-Klasse in einer High School, auch diesmal ohne Vorankündigung. Die Klasse arbeitete an einem Test:

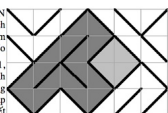
Bể cá

Problem in summary:

You can put a stick of $\sqrt{2}$ length in diagonal to make a maze. The maze contains closed area or open area. Given the maze, you are required to count the number of closed area and to find the area of the largest one.

For the maze of the picture below, we have 2 closed area and the largest one has area of 8.

Trên một tấm kính hình chữ nhật kích thước $M \times N$ được chia thành $M \times N$ ô vuông kích thước 1×1 , người ta gắn những miếng kính nhôm vuông góc với nó tại vị trí một trong hai đường chéo của mỗi ô, chiều dài mỗi miếng là $\sqrt{2}$ và chiều cao là 1, bề dày của các miếng kính là không đáng kể (xem hình dưới). Các vị trí tiếp xúc được gắn đủ để nước không thấm qua. Bằng cách đó, có thể có những vùng độc lập với nhau chứa được những lượng nước nhất định. Kết quả là ta thu được một bể cá kỳ quái, có thể dùng để thả cá vào mỗi vùng đó.



Yêu cầu: Cho trước một trạng thái của bể cá, hãy xác định bể có bao nhiêu vùng và lượng nước chứa được trong vùng lớn nhất (vùng chứa nhiều nước nhất). Hình bên cho thấy bể cá có 2 vùng, vùng lớn nhất chứa lượng nước là 8.

Computer Science Test. Bildrechte: Volksrepublik Vietnam

Gegeben war eine Data File der ein Labyrinth mit diagonalen Wänden beschreibt. Aufgabe war

es die Nummer der umschlossenen Flächen zu zählen und den Flächeninhalt der größten umschlossenen Fläche auszurechnen.

Data File:

```
1000000010001000100000001
0100000100010100010000010
0010001000100010001000100
0001010001000001000101000
1000100010001000100010000
0100000100010001010001000
0010001000100010001000100
0001010001000100000100010
0000100010001000000010001
00010001010001000000101000
0010001000100010001000100
0100010000010001010000010
1000100010001000100010001
0100000101000001000100010
0010001000100010001000100
0001010000010100010001000
0000100000001000100010000
```

Nach meiner Rückkehr in die USA fragte ich einen senior engineer wie er die Prüfungsfrage einschätzen würde bezüglich eines Job-Interviews bei Google. Ohne dass er über die Herkunft der Frage Bescheid wusste antwortete mir der Senior Engineer die Frage entspräche dem oberen Drittel. Die Klasse in Vietnam hatte 45 Minuten Zeit eine Lösung in **Pascal** zu implementieren. Fast alle Schüler wurden fertig, und nur einige wenige brauchten zusätzliche fünf Minuten. Die Hälfte dieser Schüler würden den Google Job Interview Prozess überstehen.

Ich war in die High School in Vietnam hineingegangen mit der Bereitschaft zu helfen so gut ich konnte. Doch anstatt dass die Schule von mir und meiner Erfahrung profitierte, lernte ich etwas von den Schülern. Diese Schüler zeigten mir wie man Computer Science Unterricht machen sollte: *Früh mit Programmierunterricht beginnen* [5], verpflichtend für alle Schüler. Den Schülern die sich dafür interessieren unbegrenzt Raum zum wachsen geben.

Allerdings gibt es einen Hacken im Vietnamesischen System. Die Einführung des Computerunterrichts

in das Vietnamesische Schulsystem ist relativ neu. Es scheint so als ob es in allen Altersstufen gleichzeitig begann vor ein paar Jahren. Als ich eine Universität besuchte war ich nicht beeindruckt davon was die Studenten dort taten. Doch dies wird sich schnell ändern sobald die derzeitigen Schüler auf die Universitäten kommen mit ihrer bereits in der Schule erworbenen Programmiererfahrung.

Vergleich mit USA

Der Zustand der Amerikanischen Computer Science Education ist erschreckend im Vergleich dazu.

Schulträger kämpfen darum *Computer Science* [6] aus den Schulen herauszuhalten, denn jede Minute die mit CS verbracht wird fehlt bei den Kernfächern Englisch und Mathematik. Die Testergebnisse der Schüler in diesen Kernfächern beeinflussen das jährliche Schulbudget, deshalb wird Computerunterricht als Bedrohung wahrgenommen.

Lehrer weigern sich oft Computer Science vernünftig zu unterrichten, nicht zuletzt deshalb weil sie nichts davon verstehen. *Stattdessen unterrichten Sie Textverarbeitung und HTML und nennen dies dann Computer Science* [7].

Eltern opponieren gegen Computer Science Klassen weil deren Noten sich nicht direkt auf die akademischen Chancen ihrer Kinder auswirken. Dazu kommt

„Dass Eltern den Unterschied zwischen Kindern die Videospiele spielen [8] und Kindern die Videospiele programmieren nicht erfassen.“

Schüler verlassen absichtlich die Computer Science Klassen da als **Nerd** zu gelten unglaublich unpopulär ist.

Das Resultat in den USA ist ein Widerstand gegen CS auf jeder Ebene. Diese Situation sinnvoll zu

ändern erscheint unmöglich. Ich arbeite für die Bildungsabteilung von Google und die Geschichten mit denen unsere externen Pädagogen zurückkommen sind schockierend und nicht veröffentlichbar. Wir *spenden Unsummen* [9] und enorme Ressourcen mit minimalem Ergebnis.

Im Kontrast dazu ist es in Vietnam genau umgekehrt. Schulen, Lehrer, Eltern und Schüler sind lernwillig in einer Weise die ich in den USA nie erlebt habe. Es brauchte weniger als 10 Minuten um Blocky's Maze der Vietnamesischen CS Lehrerin zu zeigen. Ihre Kinder spielten es in einer Schulstunde durch, die meisten davon beendeten die ersten 9 Level. Und wollten mehr.

(Ende der Übersetzung)

In Neils Blog gibt es weitere, auf diesen Artikel beziehende Postings, z.B. über *Neils Hochzeit* [11] oder die *Popularität von Blockly* [12].

Fachbegriffe:

Disketten: Magnetbeschichtete, scheibenförmige Datenträger, bis

zum Aufkommen von USB-Sticks weit verbreitet.

Logo: wegweisende Programmiersprache, siehe auch RIS-Journal 001, austrianguy, Seite 19

Ho Chi Min: Vietnamesischer Guerillaführer und Präsident, 1890-1969.

Pascal: beinahe ausgestorbene Programmiersprache, siehe auch RIS-Journal 001, Artikel austrianguy Seite 19

Nerd engl. für Computerfreak, Fachidiot, Streber, Außenseiter.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/vietnam

Feedback ✉ root@neil.fraser.name

Lizenz, Quellen:



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1]: neil.fraser.name/news/2013/03/16/
- [2]: code.google.com/p/blockly
- [3]: <http://goo.gl/gz5j50>
- [4]: youtu.be/4xmV5uHWNag
- [5]: neil.fraser.name/news/2012/07/01/
- [6]: goo.gl/W1ixgX
- [7]: blog.carolynworks.com/?p=572
- [8]: youtu.be/Lql-otlQfNo
- [9]: neil.fraser.name/news/2011/07/16/
- [10]: spielend-programmieren.at
- [11]: neil.fraser.name/news/2013/04/15/
- [12]: neil.fraser.name/news/2013/12/31/

Gedanken über das One Laptop per Child Projekt [1]

von CHRISTOPH DERNDORFER



Christoph Derndorfer. Bildrechte: [2]

Kategorie der Billig-Netbooks und günstigen Tablet-Computer und Entwicklung von *open source* Lernsoftware [5]. Das *One Laptop per Child Projekt* wurde vom charismatischen (und umstrittenen) MIT Professor *Nicholas Negroponte* [6] gegründet und hatte den bisher größten Erfolg in Peru und Uruguay. In Österreich gab es zwischen 2008 und 2012 eine *Probeklassen in der Steiermark die den OLPC XO Laptop* [7] regulär im Unterricht einsetzte.

Übersetzung von *Horst JENS* [8].
Originaltitel: *Happy new year from OLPC News* [9]

(Beginn der Übersetzung)

Während ich hier sitze und durch die Postings von 2013 durchblättere anstatt mich für die Silvesterparty vorzubereiten bekomme ich einen

Christoph Derndorfer [2] ist Mitgründer von *OLPC Austria* [3], und forscht, reist, programmiert und bloggt [4] auf englisch und spanisch über das *One Laptop per Child (OLPC) Projekt* [1] und über Möglichkeiten die Bildungssituation in Entwicklungsländern zu verbessern.

Hintergrund:

Ziel des *One Laptop per Child Projekts* war es einen billigen 100 US\$ Computer für jedes Kind in Entwicklungsländern zu entwickeln. Obwohl dieses Ziel nie ganz erreicht wurde führten die damit einhergehenden technologischen Durchbrüche zur

Gedanken nicht aus meinem Kopf: 2013 war ein seltsames Jahr für One Laptop per Child.

Ich meine nicht dass es ein ereignisloses Jahr war. Aber die geringen Anzahl an Postings -nur zwei Dutzend im letzten Jahr- zeigt doch deutlich dass nicht allzuviel passiert ist was mich zum schreiben inspiriert hätte. Die Wahrheit ist, dass vieles von dem was passiert ist mich eher deprimiert hat.

Auch wenn das viele Leute innerhalb der globalen OLPC Community nicht wahrhaben wollen: Es wurden sehr offensichtlich dass OLPC heute eine andere Organisation ist als sie 2012 und davor war. Natürlich bedeutete 'anders' nicht automatische 'schlechter'. Ich persönlich kann mich aber weder sehr für das neue XO Tablet (auf das sich die OLPC Association in Miami 2013 großteils konzentriert hat) begeistern noch finde ich es besonders wertvoll.

Die große Herausforderung, die mich antreibt, bleibt es herauszufinden wie man Informations- und Kommunikationstechnologien in das Bildungssystem von Entwicklungsländern integrieren kann. Wir haben viel darüber gelernt was funktioniert und was nicht funktioniert seit OLPC 2005 gegründet wurde. Und ich traue mich zu behaupten, dass die Welt dank der Bemühungen der Organisation, ihrer Angestellten und weltweiten Unterstützer ein besserer Ort geworden ist.

Allerdings habe ich das Gefühl dass 2013 gezeigt hat dass wir an einem toten Punkt angelangt sind.

OLPC Association als Organisation kümmert sich nicht länger um die richtigen Fragen, produziert keine relevanten Antworten und hat an personellen Kapazitäten verloren, welche sie die letzten Jahre über innehatte.

Als olpc Community und größeres Ökosystem haben wir noch nicht herausgefunden wie wir mit diesen Änderungen umgehen sollen.

Klar, immer noch arbeiten Leute an interessanten technischen Lösungen von denen einige sicherlich später sehr wertvoll werden. Aber davon abgesehen ist nicht klar wer die schweren Herausforderungen lösen wird (oder zumindest versuchen wird sie zu lösen) welche ich die sechs Kriterien einer erfolgreichen Integration von Informationstechnologie in den Bildungsbereich von Entwicklungsländern nennen möchte:

- Infrastruktur
- Wartung
- Unterrichtsmaterialien (Content)
- Einbindung der Communities
- Lehrerausbildung
- Evaluierung (Erfolgskontrolle)

Wenn wir erfolgreich sein wollen, müssen wir uns diesen Herausforderungen stellen und nicht darauf hoffen das jemand anderer dies für uns tun wird. Denn das ist der Geist der OLPC von Anfang an angetrieben hat.

Die Frage ist -egal ob in Österreich oder in Zambia- nicht länger *ob* wir Informationstechnologie im Unterricht verwenden sollen oder nicht. Die Frage ist welche Technologie wir verwenden sollen und -wichtiger noch- *wie* wir sie verwenden wollen.

Wenn wir bei diesen Fragen mitreden wollen anstatt zuzuschauen wie sich die Klassenzimmer der Welt mit ungeeigneten Geräten und veralteten pädagogischen Konzepten füllen - dann sollten wir uns besser mehr anstrengen und auf die wichtigen Fragen konzentrieren.

Anderenfalls finden wir uns in ein paar Jahren abseits der Entwicklung stehend und darüber jammern und mit Vorwürfen und Fehlersuche beschäftigt. Ich weiß nicht wie es Euch damit geht aber ich habe andere Pläne für 2018.

Mit diesen Gedanken im Kopf wünsche ich Euch allen ein glückliches neues Jahr und freue mich auf 2014!

(Ende der Übersetzung)

Spenden:

OLPC (Austria) freut sich über jegliche Unterstützung und Spenden. Kontakt: office@olpc.at.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001

Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/olpcnewyear

Feedback ✉ christoph@olpcnews.com

Lizenz, Quellen:

Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] one.laptop.org
- [2] christoph-d.blogspot.co.at
- [3] olpc.at
- [4] olpcnews.com
- [5] <http://goo.gl/1SE4Xm>
- [6] Nikolas Negroponte
- [7] fuzo-archiv.at/artikel/1503081v2
- [8] spielend-programmieren.at
- [9] goo.gl/uvojYk

Pyladies Vienna

von FLOOR DREES UND HORST JENS



Floor Drees. Bildrechte: Floor [1]

Floor Drees [1], Gründerin der Lerngruppe Pyladies Vienna [2], beantwortete in diesem Email-Interview Fragen zur über die Programmiersprache Python [3] und spricht über Pyladies, den Unterschied zwischen Holland und Österreich sowie über die Situation von Frauen in der IT-Branche. Übersetzung von Horst JENS [4]



Logos von python und pyladies. Bildrechte: [3], [2]

(Beginn der Übersetzung)

Horst: Hallo Floor, bitte erzähl kurz über dich. Wo bist du her, warum bist du in Wien, wovon lebst du?

Floor: Hallo. Naja, ich studierte Grafik Design in Rotterdam und arbeitete dort einige Jahre als **Community Managerin** bevor ich nach Wien zog um in einem **Startup** zu arbeiten. Ich sagte mir ich bleibe für vielleicht ein Jahr in Wien, aber es wurden zweieinhalb Jahre daraus. Vor zwei Wochen übersiedelte ich nach Berlin. Seit ich nach Wien zog hat sich viel verändert.

Ich begann mich fürs Programmieren zu interessieren, ich begann CSS, Ruby und Python zu lernen und ich bekam einen Job als **'Developer evangelist'** bei CheckiO [5], einer Spiele-Plattform für Programmierer. Ich liebe es Kommunikationsfähigkeiten mit technischen Skills zu kombinieren, und Dank der Arbeit bei CheckiO (**gamified learning**) konnte ich meine Coding-Skills ordentlich verbessern!

Horst: Was ist für dich der auffälligste Unterschied bei Land und Leuten zwischen Holland (bzw. anderen Ländern) und Österreich, deiner Erfahrung nach?

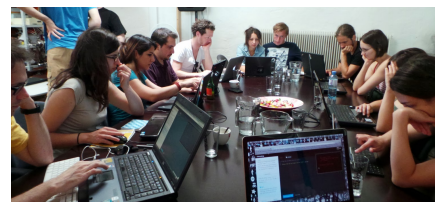
Floor: Was ich als größten Unterschied erlebt habe generell ist die 'Schau ma mal..' Mentalität hier. Ich kann ziemlich enthusiastisch (excited) werden für ein Projekt und mag sofort loslegen. Die meisten Österreicher sind mehr... konservativ, schätze ich. Vielleicht sind Holländer einfach nur direkter. Ich hatte das Gefühl sobald ich jemanden gefunden hatte der gern mit mir arbeitet waren sie (die Österreicher) voll dabei. Es hat nur ein bisserl länger gebraucht.

Horst: Warum sollten junge Leute (speziell Mädchen) programmieren lernen?

Floor: Ich denke die Fähigkeit zu programmieren ist DER 'Skill' der Zukunft. So wie die Baby-Boomer lernen mussten mit all den Computer um sie herum umzugehen.

Wer das nicht macht würde sich selbst vom Arbeitsmarkt ausschließen. Um von meiner eigenen Erfahrung zu sprechen, ich wurde wesentlich besser in meinem Job als Community-Manager sobald ich lernte die Wünsche der Community den

Entwicklern (devs) zu übersetzen. Ich denke es ist eine wichtige Fähigkeit wenigstens die Grundlagen von Markup und Styling (**html, css**) zu verstehen, zu wissen was *Backend* passiert und vielleicht leicht verständlichen Programmcode wie **Ruby** oder **Python** lesen zu können.



Pyladies-Treffen. Bildrechte: [1]

Und falls diese Jugendlichen zufällig Mädchen sind und sich für solche Sachen interessieren dann hoffe ich sehr dass sie ihre Studien in diesem Gebiet weiterverfolgen, denn wir brauchen dringend mehr Diversity um bessere Software zu erschaffen!

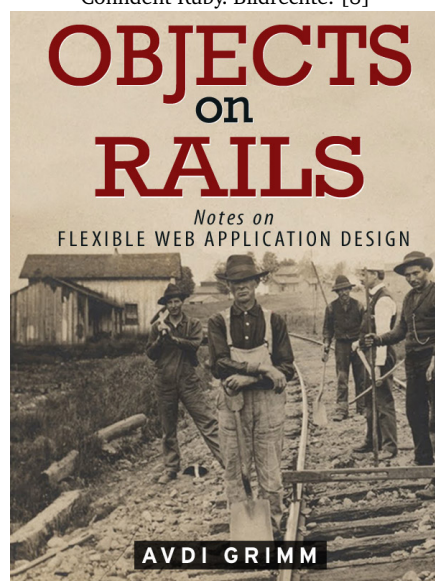
Horst: Was war deine Motivation programmieren zu lernen? Hattest du ein (weibliches) Rollenmodell?

Floor: Das klingt vielleicht lustig, aber ich wollte eigentlich **Bugs fixen**. Und ich wollte lernen wie man mit den Entwicklern bei dem Startup halbwegs professionell kommuniziert. Ich fing an die Rails Girls Tutorials durch zu arbeiten mit einem Arbeitskollegen. Er ist extrem gut mit CSS vertraut und ein sehr geduldiger Typ. Später verbrachte ein anderer Freund und Ex-Arbeitskollegen seine Wochenenden damit mir und einem Freund Ruby beizubringen, *'the test driven way'*. Diese Unterstützung bedeutete enorm viel für mich. Was ich als Community-Managerin über die Jahre gelernt habe ist dass ein großer Teil der Arbeit darin besteht gut reden zu können (*Talking the Talk...* sich selbst und die eigenen Fähigkeiten verbal präsentieren zu können). Ich begann all die Konferenz-Talks über Programmierung anzuschauen, und ich lauschte dem *Ruby Rogues Podcast* [6] inbrünstig. Am Anfang musste ich alle 10 Minuten auf die Pause Taste drücken um die Fachbegriffe nachzuschlagen. Aber nach einer Weile begann ich Dinge im Kopf miteinander zu verbinden und erkannte Begriffe

und ich konnte eine ganze Episode ohne Pause anhören. All die Jungs und Mädels bei Ruby Rogues wurden für mich zu Vorbildern. Ich mochte speziell Bücher von Avdi Grimm, (z.B. *Objects on Rails* [7] oder *Confident Ruby* [8]) . Seine Bücher habe ich von vorne bis hinten gelesen. Und auch *Katrina Owen* [9]).



Confident Ruby. Bildrechte: [8]



Objects on Rails. Bildrechte: [7]

Horst: Deine erste Programmiersprache war?

Floor: Zuerst probierte ich mit PHP herum als ich anfang mit WordPress zu arbeiten. Ich war damals 18 Jahre alt oder so. Aber das war mehr ein löschen, reloaden und schauen-was-passiert Ansatz !

Horst: Warum Python?

Floor: Ich arbeite für mehrere Startups die Python verwendeten, und ich hatte mit Django zu tun, sodass es logisch war dass Python als nächstes auf meine Liste kam. Und, nicht zu vergessen, Python Erfinder Guido van Rossum ist Holländer !

Horst: Was ist PyLadies Vienna?

Floor: Immer wenn ich etwas neues lerne brauch ich ...sozialen Druck. Alleine lernen kann sehr öde sein und ich wollte andere AnfängerInnen finden die den Samstag mit mir gemeinsam verbringen wollten um zu lernen. Deshalb organisierte ich einen ersten Workshop, suchte mir ein Tutorial aus, besorgte Kekse und legte los.

Dass ich eine PyLadies Gruppe (chapter) gründete war eher Zufall. Ich las über diese internationale Bewegung und dachte mir dass diese Marke zu übernehmen (*franchising this brand*) würde mir dabei helfen TeilnehmerInnen und Unterstützung zu finden. Was funktionierte. Ich bin sehr zuversichtlich dass *PyLadies Vienna* weiterhin wachsen wird, mit Laura als neuer Führerin und dass die geplante Zusammenarbeit mit Laber's Lab (die haben die Arduino bootcamps in Wien organisiert) funktionieren wird.

Horst: Deine Meinung zu anderen Programmiersprachen?

Floor: Ich beschäftige mich in letzter Zeit mit **JavaScript**, und ich verfluche mich dass ich nicht schon viel früher damit begonnen habe. Es macht Spaß - manchmal. Für ein Nebenprojekt musste ich etwas Java lernen und mochte das auch. Sicher, Java ist umständlich und man muss sehr viele Programmzeilen schreiben damit man ein Resultat sieht... aber Java hat seine Verdienste. Ich mag dieses *language bashing* gar nicht dass man manchmal auf Konferenzen sieht. Es ist so unnötig.

Horst: Deine Erfahrung mit Frauen in der Technik?

Floor: Generell gibt's da zu wenig Frauen - haha ! Aber immer wenn ich Entwicklerinnen (female developers) kennen lerne (*they are usually super excited about coding*) sind die begeis-

tert vom Programmieren und dieser Enthusiasmus ist ansteckend.

Horst: Warum glaubst du gibt es hauptsächlich männliche Programmierer in Österreich / in anderen Ländern?

Floor: Ich persönlich denke -und vielleicht liege ich falsch- dass Informatik als Fach in den Schulen nicht sehr attraktiv für Mädchen ist. Ich war wirklich schlecht in Mathematik und habe die Aufgaben nie interessant gefunden. Warum soll ich die Wahrscheinlichkeit, eine rote Kugel aus einem Sack zu ziehen, ausrechnen? Ich durstete nach Beispielen aus dem echten Leben. Unsere Informatikstunden ertranken in Mathematik-Rätseln, ich hätte wirklich gerne gelernt wie man Homepages selber macht (HTML, CSS, vielleicht JavaScript). Etwas machen wo man -nach Knopfdruck- die Früchte der eigenen Arbeit bewundern kann, dass hätte mich total fasziniert.

Mir fällt auf dass auf den Rails Girls Workshops wo ich unterrichtete (und die ich organisiere) die meisten Teilnehmerinnen ebenfalls dieses sofortige Feedback suchen.

Horst: Dein schlimmstes Vorurteils-Erlebnis (in Bezug auf Frauen in Technik)? Welche typische Aussage nervt dich speziell?

Floor: Wenn sie (Frauen) sich (für Technik) interessieren würden wären sie hier. Wir sind offen für alle. Das ist soooo vergiftet. Viele Frauen die mit Programmieren anfangen sind schon etwas älter weil sie in der Schule nie dafür interessiert wurden. Manchmal sind sie oder fühlen sie sich wie Anfängerinnen. Wenn sie dann Bilder sehen von Meetings mit gegen 100% Männeranteil dann wirkt das nicht sehr einladend. Du fürchtest dich dass deine Fähigkeiten in Frage gestellt werden. Was manchmal auch passiert. Bestes Beispiel: Oft kommen neue Leute zum Vienna.rb (Ruby) Meeting die ich mitorganisiere. Bis die Leute sehen dass ich den Eröffnungsvortrag halte glauben sie ich bin die Servierkraft vom Catering. Das ist nicht cool, aber ich kann meistens drüber lachen.

Horst: Dein bestes Technik/Gender

Erlebnis?

Floor: Ich finde die Ruby/Rails Gemeinschaft als Ganzes sehr einladend. Ich teile mit anderen meine Code Schnipsel, oder erzähle welches Buch ich gerade lese, und ich bekomme super nettes Feedback via Twitter. Vielleicht spielen Rails Girls und Railsbridge eine große Rolle dabei ein so sicheres Umfeld zu schaffen.

Als ich neulich fragte wer mit mir zusammen an einem Javascript Projekt zu arbeiten will bekam ich auch extra nette Antworten.

Horst: Welchen Rat hast du für junge (Schul)Mädchen die sich für eine Programmier-Karriere interessieren?

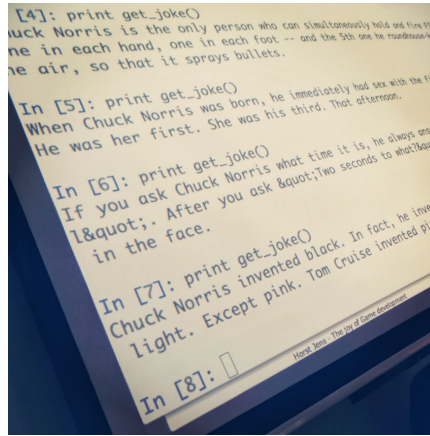
Floor: Du fragst eine Kunststudentin die Social-Media-Managerin wurde um dann eine Kehrtwendung zu machen und Webseiten zu erstellen! Mein Rat:

Folge deinem Herz, da draußen gibt es Ressourcen die dir helfen werden wenn du deine Meinung änderst und etwas anderes lernen willst. Und ich hoffe du bewahrst dir Offenheit und eine gesunde Person Neugier (aufs Programmieren natürlich!).

Horst: Letzte Lebensweisheiten für unsere jungen LeserInnen?

Floor: Was sie dir in der Schule beibringen ist wie man sich Wissen aneignet, nicht endgültiges Wissen. Da draußen gibt es so viel, benutze deine Fähigkeiten um etwas zu lernen was DU nützlich und interessant findest. Oft wirst du Leute finden die dich dabei anleiten, vielleicht sogar deine Lehrer!

(Ende der Übersetzung)



Output eines Pyladies Treffens. Bildrechte: [1]

Fachbegriffe

Pyladies Vienna [1] steht für Python Ladies, eine Selbstlerngruppe für die Programmiersprache *Python* [3], speziell für Frauen.

Python [3] einfach zu erlernende, Allzweck-Programmiersprache. Wird in den Programmierkursen der Firma spielend-programmieren [4] eingesetzt.

Community Manager betreut die Gemeinschaft (Community) im Internet. Kümmt sich z.B. für eine Spielefirma um die Anliegen der im Internet organisierten Fans eines Computerspiels.

Startup junges Unternehmen.

Developer evangelist versucht Programmierer für eine Firma oder Technologie zu begeistern.

Gamification, gamified learning versucht die Belohnungsmechanismen und die Motivation von (Computer)spielen auf andere Bereiche zu übertragen, in diesem Fall Lernen. Ein Beispiel dafür sind die Punkte und Badges (Abzeichen) die man bei der Website KhanAcademy.org erwerben kann.

ben kann.

html steht für Hyper Text Markup Language. Seitenauszeichnungssprache. Damit werden Webseiten beschrieben, z.B. Überschriften, Tabellen, Hyperlinks etc.

Css steht für Cascading Style Sheet. Ermöglicht es, Webseiten (html) optisch zu gestalten (Farben, Layout)

Bugs fixen: Programmfehler (so genannte Bugs) reparieren

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001

Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/floor

Feedback ✉ floordrees@gmail.com

Lizenz, Quellen



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] FloorDrees
bzw. plus.google.com/+FloorDrees
- [2] twitter.com/pyladies_vie
bzw. meetup.com/pyladies-vienna
- [3] python.org
- [4] spielend-programmieren.at
- [5] www.CheckiO.org
- [6] rubyrogues.com
- [7] objectsonrails.com
- [8] confidentruby.com
- [9] kytrinyx.com

Railsgirls haben mein Leben verändert

von LAURA GAETANO

Blogposting [3] schrieb sie im Oktober 2013. Übersetzung von Horst JENS [4].

(Beginn der Übersetzung)



Rails-Treffen The Hague. Bildrechte: [1]



Logos von railsgirls. Bildrechte: [5]

Laura bloggt über ihre lebensveränderndes *erstes Mal* bei einem Rails Girls Treffen wo sie die Programmiersprache *Ruby* kennen lernte und seither selbst unterrichtet. Außerdem schreibt sie über Sexismus in der Tech-Szene, Javascript lernen, Tricks zum regelmäßigen Schreiben und ihr Leben.



Laura Gaetano. Bildrechte: [1]

Laura Gaetano, im Internet bekannt als *AliceTragedy* [1] arbeitet in Wien als Junior Developer und Social Media Expertin bei *Bitstem* [2]. Seit Ende 2013 organisiert sie die *Pyladies Vienna Meetings*. Folgendes

Es war einmal, in einer weit, weit entfernten Galaxie...

Ich habe das Schreiben dieses Blog-Postings lange vor mir her geschoben. Das Leben ist ein so viel netterer Platz als das Internet in letzter Zeit, also habe ich lieber das Leben genossen als es still vom Rande aus zu beschreiben wie ich das sonst mache.

Für die (paar) die es interessiert, hier ist eine kleine Zusammenfassung der letzten Monate:

RailsGirls:

Ich besuchte den RailsGirls Event im Jänner, und er hat mein Leben verändert. Ich sah dort eine Menge Dinge die Frau verbessern könnte um das Erlebnis als Besucherin angenehmer zu machen. Deshalb besuchte ich im August zum ersten Mal überhaupt einen **RailsGirls** Event als *Trainerin*, in *Bratislava* [5]. Dieses Wochenende war ich auch seit sechs Monaten in meinem neuen Job. Wirklich? Ein halbes Jahr kämpfte ich mich schon täglich mit **Ruby on Rails** ab? Wer hätte vor neun Monaten gedacht dass ich einmal von Stadt zu Stadt reisen würde um Vorträge über die **Kommandozeile** zu halten?

Aber es gibt für Alles ein erstes Mal und ich lernte einige wirklich großartige und interessante Leute kennen, ich lernte besser in der Öffentlichkeit Vorträge zu halten, und ich bekam gratis Aufkleber. Vor allem: Ich freue mich immer so zu sehen wenn wenigstens eine meiner *students* sich für das Programmieren begeistert (*get excited about making stuff*) und sich ernsthaft überlegt in dieser Richtung zu arbeiten.

(Nebenbemerkung: In letzter Zeit gab es einiges Aufsehen über das Thema **Sexismus** in der Technologie-Branche. Ich denke man kann nicht genug darauf hinweisen dass dies nichts speziell mit der Ruby Community zu tun hat, oder mit Programmieren im Allgemeinen, oder mit der **IT-Industrie**. Sexismus, und jede andere Form von Diskriminierung, ist unglücklicherweise ein weltweites, branchenübergreifendes Problem. Es ist traurig, aber ich denke es hilft nicht dass wir Frauen uns als Opfer betrachten. Vielleicht ist der Effekt von Organisationen wie RailsGirls die Frau welche Teil der IT-Industrie werden will öffentlich zu machen und das Problem damit größer zu machen als es eigentlich ist; Ich weiß es nicht. Für mich war **RailsGirls** der erste Schritt in Richtung Programmieren und ich möchte diese positive Erfahrung an andere zurück geben.)

Meetups:

Momentan ich ich regelmäßige Besucherin der *Vienna.rb* [6] **Meetups** und der *WordPress Vienna* [7] Meetups in Wien. Die Meetups fühlen sich fast so nett an wie ein Familientreffen, deshalb schlage ich vor ihr besucht uns wenn ihr Interesse daran habt mit netten Leuten zusammen zu sein und Programmieren zu lernen. *Pyladies Vienna* [8], welches ich zusammen mit **Floor DREES**¹² organisiere, findet auch bald statt.

750 Wörter:

Vor Jahren, ich ging noch ins Gymnasium, fand ich ein Buch über 'kreativ

¹²Siehe Artikel *floor* in dieser Ausgabe

Schreiben' welches empfahl täglich zu schreiben. Nicht notwendigerweise gut konstruierte kleine Essays oder Geschichten; aber trotzdem drei volle Seiten täglich (die 'morning pages') ohne zu viel über den Inhalt nachzudenken. Der Grund dafür ist dass dies dabei hilft die *Schreibblockade* zu mindern.

Einige Jahre später nahm ich an *Nanowrimo* [9] teil und bald darauf meldete ich mich bei **750 Words** an, inspiriert durch meine Schreibübungen. 750 Words erlaubt dem User täglich Seiten zu schreiben, führt Statistiken über Schreibgeschwindigkeiten und Stimmungen, und man kann Abzeichen (*badges*) gewinnen.

Als ich in Bratislava war beschloss ich am monatlichen 750words Wettbewerb teilzunehmen: Ich habe seitdem nicht mehr aufgehört zu schreiben und finde jeden Tag 15 bis 20 Minuten Zeit dafür. Ironischerweise blogge ich seitdem deutlich weniger, aber man kann nicht alles gleichzeitig machen.

Und jetzt ?

Ich lese einen Haufen Sachbücher (Gay Talese und Norman Mailer sind gerade meine Favoriten), ich verbessere meine **Typografie** Kenntnisse und versuche langsam **Javascript** in meinen Kopf hinein zu bekommen weil ich Javascript noch überhaupt nicht kann. Außerdem verfolge ich ein paar Nebenprojekte. Und ich habe erstmals in meinem Leben **GTA** gespielt (ich weiß, unakzeptabel, aber wozu gibt es einen Freund mit Playstation?). Und obwohl ich nur sporadisch blogge bleibt Blogs schreiben meine erste Liebe zu der ich hoffentlich bald zurück komme.

(Ende der Übersetzung)

Anmerkungen

Alterslimit:

Laut Laura gibt es weder für Rails

Girls noch für Pyladies Treffen ein Alterslimit, die ältesten Teilnehmerinnen bisher waren über 50. Mit minderjährigen Teilnehmerinnen hat sie noch keine Erfahrung und ersucht bei der Anmeldung extra auf das Alter hinzuweisen, um die Organisation zu erleichtern.

Vorträge:

Wer auf einem (Pyladies) Treffen einen sogenannten Lightning-Talk halten will (max. 5 Minuten, sehr empfehlenswert) möge Laura im Vorfeld kontaktieren.

Fachbegriffe

RailsGirls Internationaler Verein, organisiert Treffen für Frauen welche die Programmiersprache **Ruby on Rails** lernen möchten.

Ruby on Rails ist ein *open source web application framework*, geschrieben in der Programmiersprache **Ruby**.

Ruby ist eine *general-purpose programming language* ähnlich wie **Python**

Kommandozeile oder **command line**: Text-Eingabemethode im Unterschied zu grafischen User Interfaces (*GUI*), welche mit der Maus bedient werden. Die Kommandozeile wird vor allem von Programmierern bevorzugt da man damit im Unterschied zu einem GUI deutlich schneller arbeiten kann (wenn man sich auskennt).

Sexismus Diskriminierung aufgrund des Geschlechts. In der IT-Branche ist damit meist die Diskriminierung von Frauen gemeint.

IT-Industrie Internet-Technologie

Pyladies Vienna [8] steht für Python Ladies, eine Selbstlerngruppe für die Programmiersprache **Python** speziell für Frauen.

Python leicht zu erlernende general purpose Programmiersprache

Wordpress populäre Open-Source Software um Webseiten, speziell **Blogs**, zu erstellen

Blog eine öffentlich sichtbares *Tagebuch*, *Logbuch* im Internet.

Typografie Druckkunst, gestalterische Merkmale eines Schriftsatzes.

Javascript Programmiersprache (nicht zu verwechseln mit Java) die innerhalb eines Browserfensters läuft.

Grand Theft Auto, **GTA**: Sehr populäre Computerspiel-Serie

Download, Feedback:

R.I.S.-Journal, Ausgabe 001: spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/laura
Feedback ✉ girl@alicetragedy.org

Lizenz, Quellen:

Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] www.alicetragedy.org/blog/
- [2] bitstem.com
- [3] goo.gl/WG7AvI
- [4] spielend-programmieren.at
- [5] railsgirls.com/bratislava
- [6] vienna.rb
- [7] wpvienna.com
- [8] meetup.com/pyladies-vienna
- [9] nanowrimo.org/
- [10] 750words.com/

Fdroid Installationsanleitung

von HORST JENS

In diesem Tutorial wird erklärt wie Sie sich auf einem Android Smartphone oder Tablet das Fdroid Repository für freie (free/libre Open Source) Software installieren, um eine sinnvolle Ergänzung zum Google Play Store zu haben.



Fdroid

Das freie Softwarerepository Fdroid für Android installieren

Wirklich *freie* (free/libre Open Source) Software [1] für Android [2]-Geräte zu finden ist gar nicht so leicht: Der *Google Play Store* [3] lässt sich zwar nach *kostenlos* durchsuchen, aber *kostenlos* (free as beer) ist eben nicht das selbe wie wirklich frei (quelloffen, free as freedom) im Sinne der *GPL Lizenz* [4]. Mit relativ wenig Aufwand ist es allerdings möglich, auf Android-Geräten (Devices wie Smartphones, Tablets, ...) ein alternatives Softwareverzeichnis, ein sogenanntes *Repository* [5] zu installieren, welches nur wirklich freie

Software anbietet. Keine *Crippleware* [6], keine nervenden Werbebanner, kein geheimer Programmcode. Dieses Tutorial erklärt wie es geht:

Anleitung:

Die folgende Schritte möglichst direkt mit dem Android-Gerät ausführen:

Gerätezugriff gewähren: Die **FDroid-App** verlangt beim Installieren weitgehende Rechte auf dem Android-Gerät, um weitere Apps installieren und löschen zu dürfen:

- Im Systemmenü des Android-Geräts unter *Einstellungen - Sicherheit* die Option *Unbekannte Herkunft - Installation von Apps aus anderen Quellen als dem PlayStore* zulassen anklicken (das Häkchen muss gesetzt sein).
- Die Website <http://f-droid.org/> ansurfen und die Seite durchlesen. (Die Seite akzeptiert Spenden per Flattr, PayPal und Bitcoin.) Im Fdroid FAQ (im Wiki) wird u.a. erklärt, warum FDroid NICHT im Google Play Store drin ist (Konkurrenzverbot).
- Direkt auf der FDroid-Website kann man mit dem Menüpunkt *Browse* schauen was für freie Apps es gibt, um zu entscheiden ob man FDroid überhaupt braucht.
- Direkt auf der *FDroid-Website* [7] ist ein Download-Link mit dem man die Datei *FDroid.apk* [8] downloaden kann. Oder man scannt mit dem Android-Device diesen QR-Code:



- Die heruntergeladene Datei *FDroid.apk* öffnen. Dazu oben in der Statusleiste auf das Downloadsymbol (Pfeil nach unten) mit dem Finger wischen und auf die Meldung *Download abgeschlossen* drauf klicken. Nach einigem Nachfragen wird die FDroid-App auf dem Android-Device installiert.
- Die FDroid-App im App-Menü des Android-Geräts suchen und starten.
- Wenn die FDroid-App läuft, den Menü-Button drücken und *Update* wählen. Nach einiger Zeit erscheint eine Liste verfügbarer freier Apps, die man mittels *Install*, *Update* und *Remove* managen kann.
- Um zu schauen ob es neuere Versionen von freien Apps gibt muss man FDroid starten und im Menü auf *update* klicken. Die fdroid-Apps werden NICHT über den Google Play Store aktualisiert.
- Es empfiehlt sich manchmal, nicht die allerneueste Version einer App zu installieren sondern die vorletzte Version (die 2. von oben in der Versionliste). Dadurch verzichtet man zwar auf die allerneuesten Features, erspart sich dafür die allerneuesten Fehler.
- FDroid selbst lässt sich mittels FDroid updaten :-)

Fachbegriffe:

freie Software [1]: ist nicht zu verwechseln mit kostenloser Software (Freeware). Freie Software gewährt die **4 Grundfreiheiten** gemäß der **GPL-Lizenz** und ermöglicht es, den Quellcode (SourceCode) der Software anzuschauen, zu benutzen, zu verändern und zu kopieren und weiterzugeben.

Gnu Public License (GPL) [4]



garantiert dem Nutzer der Software die 4 Grundfreiheiten **use, study, share, improve**. Näheres dazu können Sie im Leitartikel dieser Ausgabe nachlesen.

Android [2]: Ein freies, Linux-basiertes Betriebssystem, vor allem für Smartphones und Tablets. Android selbst ist GPL-lizenzierte, freie Software, aber viele der Apps die auf Android laufen (z.B. Skype, aber auch der **Google Play Store** sind unfreie Software..

Google Play Store ist ein von Google vorinstallierter Softwareshop für Android-Geräte, ähnlich Apple's AppStore. Derzeit (Dez 2013) lässt sich Google's Play Store zwar nach kostenlosen Apps filtern, nicht aber nach freien Lizenzen.

Software-Repository [5]: Ein Verzeichnis installierbarer Software

(oder Apps) welches eine Versionsverwaltung beinhaltet und das installieren, löschen, updaten und verwalten von Software oder Apps erlaubt. Im Linux-Bereich hat fast jede Distribution ein eigenes Repository (z.B. Debian, Ubuntu, Suse...). Man kann sich auch selbst Repositories erstellen.

Crippleware, Krüppelware: ein verächtlicher Begriff für kostenlose Versionen von Software, deren Leistungsumfang stark eingeschränkt ist und erst nach einer entsprechenden Zahlung sinnvoll verwendbar ist. Viele Apps im Google Play Store sind in der "kostenlos"-Variante als Crippleware zu bezeichnen, da ihnen entweder wichtige Features fehlen oder weil sie den Nutzer ständig durch Werbebanner nerven, bis er ein kostenpflichtiges Upgrade zur Vollversion durchführt.

Danksagung:

Die Android-Screenshots wurden von Phillip Seibt erstellt.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
spielend-programmieren.at/de:ris:001

Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/fdroid

Feedback ✉ horst.jens@spielend-programmieren.at

Lizenz, Quellen:



Dieses Material steht unter der Creative Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] [wikipedia/Free-Software](http://wikipedia.org/wiki/Free-Software)
- [2] android.com
- [3] play.google.com
- [4] <http://goo.gl/4UTW5B>
- [5] de.wikipedia.org/wiki/Repository
- [6] de.wikipedia.org/wiki/Crippleware
- [7] f-droid.org
- [8] f-droid.org/FDroid.apk
- [9] [spielend-programmieren](http://spielend-programmieren.at)

Zahnbürstenroboter

von WINDELL H. OSKAY, PH.D.

Windell "Evil Mad Chief Scientist" Oskay [1] zeigt in diesem Tutorial wie man Roboter aus Zahnbürsten baut, so genannte *Bristlebots*

Der englischsprachige Original Artikel ist copyright-geschützt von *Windell Oskay* [1] und erschien erstmals am 19.12.2007 auf seiner *Homepage* [2]. Übersetzung von *Horst JENS* [3] mit freundlicher Genehmigung des Autors.



Windell Oskay, Ph.D. Bildrechte: [1]

(Beginn der Übersetzung)

Der Zahnbürstenroboter (BristleBot) ist ein einfacher, kleiner, emsiger Roboter. Zutaten: Eine Zahnbürste, eine Batterie, und ein Vibrationsmotor. Das Resultat kann man *auf Youtube bewundern* [4].



Bristlebot. Bildrechte: [1]

Der BristleBot ist unser Versuch einen *Vibrobot* [5] zu bauen: primitive Roboter die durch einen einzigen

Vibrations(Exzentrik)-Motor bewegt werden. Sehenswerte Varianten davon sind z.B. der mint-tin (siehe Make Magazin) und der *Kid's draw bot* [6]: ein Vibrobot mit Beinen aus Bleistiften.



Zahnbürste mit schrägen Borsten. Bildrechte: [1]

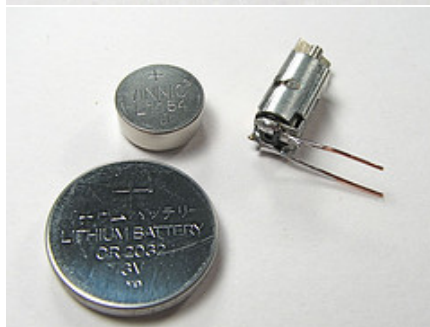
Zu Anfang steht naturgemäß die Zahnbürste. Wir brauchen eine Zahnbürste deren Borsten alle mehr oder weniger schief geneigt sind (nicht senkrecht). Möglicherweise funktioniert auch ein Bristlebot mit senkrechten Borsten die man von Hand biegt, aber ich habe es nicht versucht. Wenn die Borsten unterschiedlich lang sind (wie auf den Fotos) brauchen wir vielleicht eine Schere um alle Borsten gleich lang zu stutzen.



Bürstenkopf abschneiden. Bildrechte: [1]

Schneide den Griff der Zahnbürste ab, sodaß nur eine kleine Roboterplattform übrigbleibt.

Nächster Punkt: Wir brauchen einen Vibrationsmotor oder einen anderen winzigen Motor mit einer Unwucht an der Drehachse. Wenn Du einen genügend kleinen Motor findest kannst du das Gewicht der Unwucht notfalls selbst dran montieren. Man findet solche Motoren z.B. für Pager in allen möglichen Größen. Ich habe meinen für ein paar Dollar auf Ebay bekommen, schau mal dort nach.



Vibrationsmotor. Bildrechte: [1]

Der Motor den ich bekommen habe lief zufrieden mit fast jeder herkömmlichen Spannung von 1 bis 9 Volt. Als Stromquelle funktionieren Alkaline oder Lithium-Batterien (Uhr-Batterien) mit 1,5 Volt oder

3 Volt. Um den Motor an die Batterie anzuschließen habe ich einen Lötkolben benutzt und etwas Kupferdraht an die Motordrähte gelötet.

Außerdem brauchen wir Doppelklebeband. Gib einen kleinen Streifen davon auf die Oberseite der Zahnbürste (die Roboterplattform) auf welcher der Motor aufliegt.



Doppelklebeband. Bildrechte: [1]

Klebe den Motor auf das Klebeband. Das Klebeband sorgt für Abstand damit das Unwucht-Gewicht nicht auf die Zahnbürste hämmert. Außerdem sorgt das Klebeband für eine starke, flexible Verbindung welche die starken Vibrationen des Roboters aushält. Eine erste Verbindung zur Batterie ist möglich indem man die Batterie einfach draufstellt - das ist allerdings nicht sehr stabil und sie wird bald runterfallen.

Eine bessere Methode ist es einen der Drähte runterzubiegen zum Klebeband, sodaß man die Batterie draufkleben kann und trotzdem eine elektrisch leitende Verbindung schafft. Mit dem anderen Draht die andere Seite der Batterie berühren und es kann los gehen.



Batterien. Bildrechte: [1]

Der vollendete BristleBot, aktionsbereit. Wenn Du einen Bristlebot laufen lässt merkst Du dass er eine Tendenz hat nach rechts oder links zu steuern. Das wird beeinflusst durch Borstenlänge (z.B. eine einzelne lange Borste), die Position von Motor und Batterie und auf welcher Seite man die Batterie anschliesst. Drehe die Batterie einmal um wenn der Bristlebot nicht vernünftig läuft.

Nur zur Erinnerung: Dies ist ei-

ner von vielen verschiedenen Arten von Vibrobots - da draußen gibt es viele andere Varianten wenn du nur suchst. Wir haben viele andere Varianten gesehen oder davon gehört, und wir wissen dass Bürsten mit schiefen Borsten schon zuvor als Antrieb verwendet wurden. Diese spezielle Variante ist vielleicht einzigartig, auf jeden Fall spannend. Nur wenige Roboter die man selbst so einfach bauen kann sind so belohnend. Mit den richtigen Teilen kannst du einen in ein paar Minuten bauen. Mach eine Horde davon und lass sie um die Wette rennen !



Bristle Bot. Bildrechte: [1]

(Ende der Übersetzung)

Fachbegriffe:

Vibrobot von Vibrations-Roboter.

Bristlebot von engl. bristle = Bürste.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
spielend-programmieren.at/de:ris:001

Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/bristlebot

Feedback ✉ drwho@evilmadscientist.com

Lizenz der Übersetzung, Quellen



Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] evilmadscientist.com
- [2] evilmadscientist.com/go/bristlebot
- [3] spielend-programmieren.at
- [4] youtu.be/rUSTXUis_ys
- [5] makezine.com/projects/make-10/vibrobots
- [6] finkbuilt.com/blog/kids-art-bot

Spielen mit Schießpulver:

The Powder Toy

von HORST JENS

Dieser Artikel beschreibt das Physik/Chemie-Simulationsspiel *The Powder Toy* [1] und zeigt einen kleinen Teil der Möglichkeiten die man damit anstellen kann. *The Powder Toy* ist freie (free/libre Open Source) Software, die von jedem Interessierten benutzt, kopiert, verändert und weiterentwickelt werden kann, gemäß den

vier Grundfreiheiten welche die **GPL-Lizenz** [2] gewährt.



Horst JENS. Bildrechte: [3], cc-by-sa

Hinweis: Wenn Sie Internetverbindung haben, schauen Sie sich die folgenden Bilder online an in Farbe

oder drucken Sie diesen Artikel auf einem Farbdruker aus.

erst die Arbeit...

Auf der Homepage von *The Powder Toy* [1] kann man sich die aktuelle Version oder die noch in Entwicklung befindliche, allerneuste Beta-Version für verschiedene Betriebssysteme downloaden. Zum nur damit spielen ist nichts weiter notwendig (Download starten und Spiel installieren). Weil ich mir den **Sourcecode** genauer anschauen will habe ich mir das Programm selbst **kompiliert**. Dazu muss ich mir die tagesaktuelle Version des Programms von Github her-

unterladen. Wie das geht beschreibt das englische Tutorial *Compile for Linux* [4] im The Powder Toy Wiki ganz gut. Für Windows und Mac User gibt es im Wiki eigene Tutorials. Hier sind die Schritte die ich gemacht habe (Ubuntu Linux, 64-bit, 2 Prozessoren):

- Ein Terminal öffnen (Alt+STR+T) und folgendes eintippen

```
horst@bigschleppi:~$ sudo apt-get install git build-essential
libstdc++2-dev libbz2-dev zlib1g-dev liblua5.1.0-dev git scons
libfftw3-dev
```

- In jenes Verzeichnis wechseln unter welchem The Powder Toy installiert werden soll. Nachdem ich mir ein *games* Verzeichnis erstellt hatte (mkdir games) war das bei mir das Verzeichnis home/horst/games deshalb musste ich dorthin wechseln mit: cd games

- Jetzt die neue Version von Github klonen (folgenden Befehl in eine einzige Zeile schreiben):

```
horst@bigschleppi:~$ git clone git://github.com/simtr/
The-Powder-Toy.git
```

- das Kommando ls (oder ein Blick auf meinen Datei-Manager) verrät mir dass ich ein neues Verzeichnis auf meiner Festplatte habe, in welches ich gleich hineinwechsle: cd The-Powder-Game

- Jetzt kann der Kompilierungsvorgang starten! Ich tippe:

```
horst@bigschleppi:~$ scons --lin --release --64bit
--sse3 -j2
```

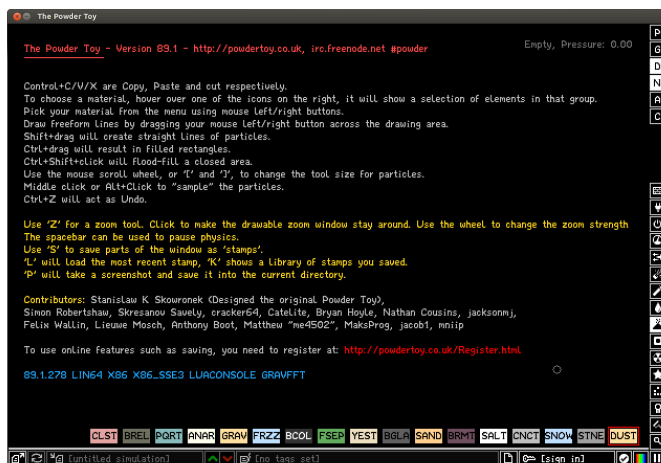
Was die Parameter bedeuten wird im Wiki erklärt. Wer kein 64bit System hat lässt das

--64bit weg, wer einen alten Computer hat lässt das --sse3 weg und wer 4 Prozessoren hat schreibt -j4 anstatt so wie ich -j2 oder lässt diesen Parameter ganz weg. Nach ein paar Minuten ist das Programm fertig kompiliert und befindet sich im Unterverzeichnis 'build'

- In dieses Unterverzeichnis hineinwechseln, entweder per Dateimanager oder per cd build und ls.
- Je nachdem welche Parameter man gewählt hat heisst das Programm jetzt unterschiedlich. Meines heisst powder64 und ich kann es per Doppelklick starten oder per . powder64.

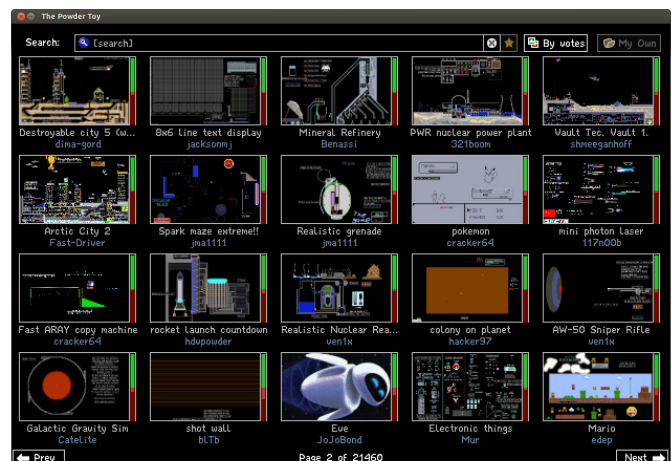
...dann das Vergnügen

The Powder Toy begrüßt den Spieler äußerst unspektakulär:



Das wichtigste Icon ist jenes mit dem man die Fenstergröße einstellen kann bzw. den Fullscreen-Modus aktiviert. Es ist das 3. Icon von rechts auf der Iconleiste am unteren Bildschirmrand, ein weisser Kreis mit einem Haken drin. Dieses Icon anklicken und entweder auf 'Large Screen' oder 'Full Screen' klicken, zumindest wenn man einen großen Monitor besitzt.

Jetzt kann es losgehen, zum Aufwärmen zerstöre ich einen Unterwasserbunker mittels Sprengstoff: Mit der 'Open' Schaltfläche ganz links unten öffne ich (sofern Internetverbindung vorhanden ist) eine hübsche Vorlage:



In diesem Fall den 'Underwater bunker' aber eigentlich ist es nicht so wichtig welche Szene ich öffne.



Man beachte die hohe *Framerate* links oben von fast 60 FPS und das sich das Spiel im Pause-Modus befindet. Mit dem Pause-Icon (ganz rechts unten im Eck) kann ich jederzeit zwischen Pause und Echtzeit umschalten.

Die Spielszene empfiehlt (Schrift oben im Bild) mittels eines Funkens (Sprk) das 'Gift-Leck' (Poison-Leak) anzuzünden, aber ich habe andere Pläne:

Ich klicke das Menü für Explosionsstoffe rechts an und wähle bei der Auswahl unten Thermit (im Spiel als 'THRM' bezeichnet), welches laut Spielbeschreibung sehr heiss verbrennt und laut Wikipedia sehr gut geeignet ist um Wolkenkratzer und Stahlkonstruktionen abzureißen. (Ich hätte auch mit dem Luben-Icon (2. von unten, Iconleiste rechts) nach allen Verfügbaren Stoffen suchen können). Noch im Pause-Modus male ich wie mit einem Malprogramm die beiden Bunkerräume unter dem 2. Baum von links voll mit Thermit.



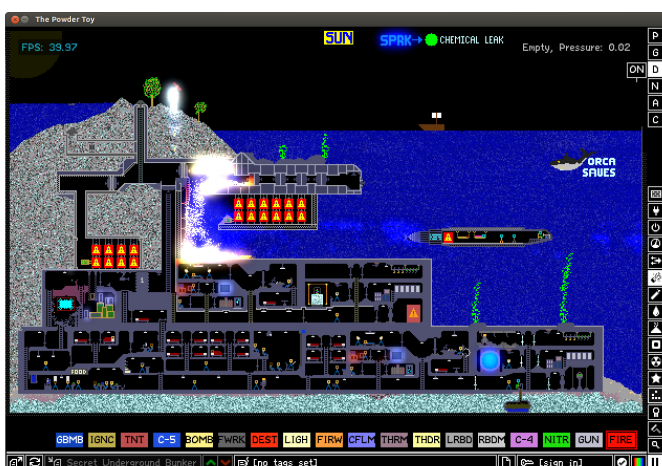
Nach Umschalten auf den Echtzeit - Modus (Icon ganz rechts unten) merke ich dass mein Thermit nicht nur nach unten rieselt sondern durch eine Taucherluke auch gleich ins Wasser. Normalerweise verhindert die Luftblase über der Taucherluke das Eindringen von Wasser in den Bunker, aber da ich die Luft mit Thermit ersetzt habe und das mein herunter-rieselnder Sprengstoff Wasser verdrängt (Die Physik in The Powder Game ist etwas vereinfacht) schwappt jetzt übelriechendes Meer-

wasser zu in meinen Bunker. Womöglich rosten dadurch die Giftmüllfässer !

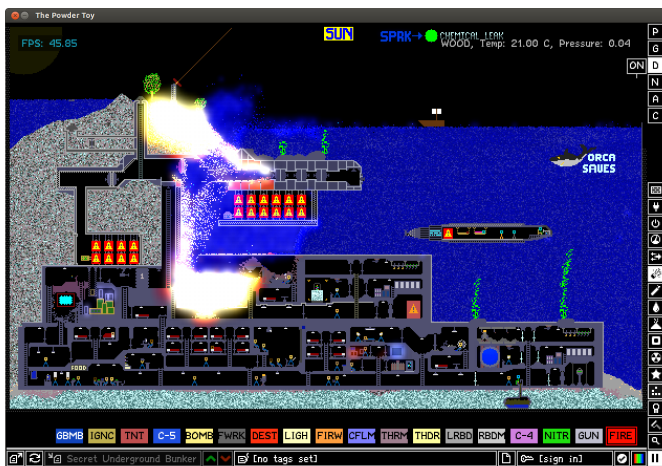
Schnell klicke ich unten rechts auf 'Fire' und klicke damit (im Echtzeitmodus) auf das Thermit, um das Wasser per Hitze zu vertreiben.



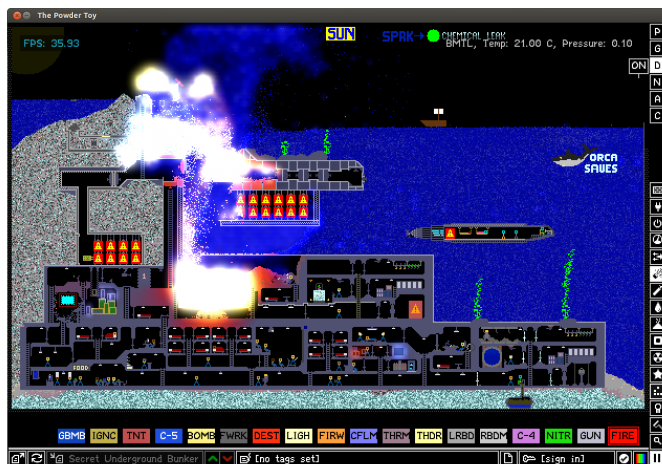
Zu dumm aber auch, ich habe das herunterrieselnde Thermit angezündet, nicht das neben den Giftfässern:



Oha, das Feuer frisst sich nach oben in den Berg und nach unten in den Bunker!. Wenigstens lässt es die Giftfässer in Ruhe



Aber ein Teil des Meerwassers wird in die Luft geschleudert und die Stahlwände im Bunker fangen an rot zu glühen.



Autsch, das waren die Giftfässer. Na, wenigstens sind sie nicht gerostet.



Im Bunker wirds ganz schön heiss...



Und das (dekorative, nicht-brennbare) Segelboot verliert durch die Verdampfung das Wasser unter dem Bug. Das U-Boot zeigt sich hingegen unbeeindruckt:



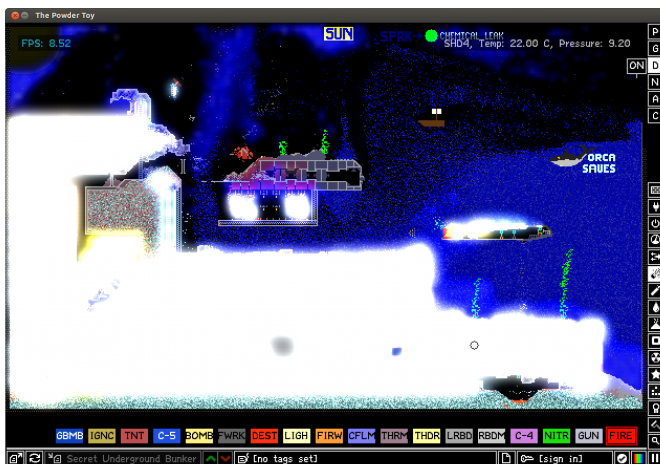
Spät aber doch vertreibt die Hitze das verbliebene Wasser aus dem (ehemaligem) Giftfass-Raum wo es der thermische Aufwind nach oben treibt:



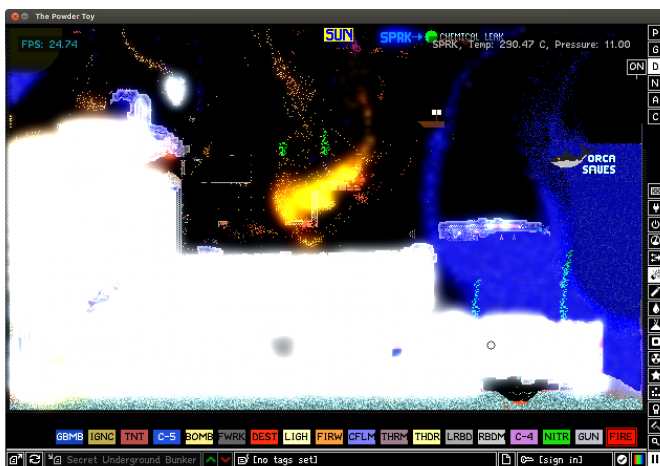
Jetzt zerlegt es auch noch die letzten im Berg verbuddelten Giftfässer



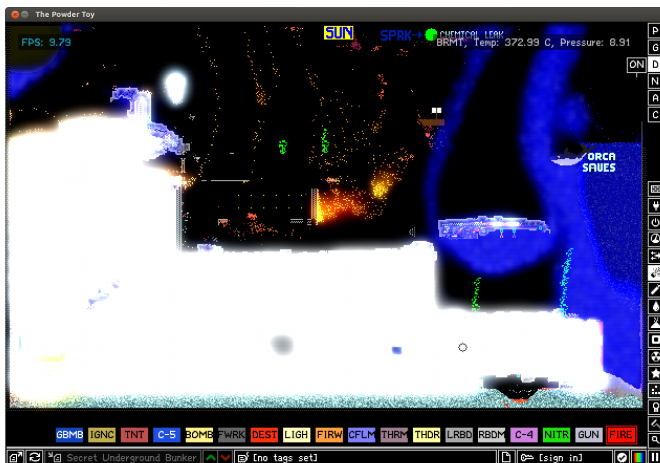
Und auch das U-Boot kommt nicht ungeschoren davon:



Weder vom Bunker noch vom Meer bleibt viel übrig:



Wie lange das Plasma im Bunker wohl brennt ?



Auch im Pause Modus kann man mittels Mauszeiger jederzeit Pixelgenau erfahren um welches Material es sich handelt und welchem Druck und welcher Temperatur er gerade ausgesetzt ist.

Möglichkeiten

Das Wiki der The Powder Game Homepage bietet ein Forum und ein FAQ sowie einige Tutorials. Man lernt wie

man mittels Lua eigene Materialien mit eigenen Eigenschaften programmiert, man kann mittels The Powder Toy elektrische Schaltkreise und Logik-Schaltungen simulieren (umständlich) oder einfach schöne Pixelkunstwerke pinseln (und sie dann anzünden). Das Spiel bietet neben Elektrizität (leitende Materialien, Isolierungen etc.) auch Gravitation und Temperatur-Veränderer (im Echtzeitmodus) mit denen sich etwa Ventilatoren, Staubsauger oder schwarze Löcher simulieren lassen. Die Meisten Materialien interagieren mit anderen Materialien: Manche lassen nur Gas durch, manche nur Flüssigkeit, manche reagieren unterschiedlich auf Hitze oder Elektrizität. Besonders gefallen mir die Schaltflächen Game of Life mit denen man das scheinbar 'lebende' Organismen erschaffen kann (siehe *Conways Game of Life*) und die Specials, z.B. Materialien die Partikel verdoppeln (damit ist ein endloser Wasserfall möglich) oder die kleinen, per Tastatur steuerbaren 'Stickmen' und die sie jagenden 'Fighter'.

Fachbegriffe:

GPL Lizenz [2]: garantiert die vier Grundfreiheiten 'to use, to study, to modify and to share'.

Sourcecode oder Quellcode sind die Programmzeilen aus denen ein Computerprogramm besteht. The Powder Toy ist in C++ geschrieben, für kleinere Veränderungen (z.B. ein neues Material hinzufügen) reichen aber Kenntnisse der relative leicht zu lernenden Programmiersprache *Lua*.

komilieren, to compile bedeutet ein Programm zu übersetzten, gemeint ist aus den Programmzeilen ein ausführbares Programm (binary) machen, mit Hilfe eines Compilers.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:

spielend-programmieren.at/de:ris:001

Download Ordner, verschiedene Formate:

spielend-programmieren.at/risjournal/001/powdertoy

Feedback ✉ horst.jens@spielend-programmieren.at

Lizenz, Quellen:



Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] powdertoy.co.uk/
- [2] gnu.org/philosophy/free-sw.de.html
- [3] spielend-programmieren.at
- [4] goo.gl/LK4z01

The Powder Toy Tutorial

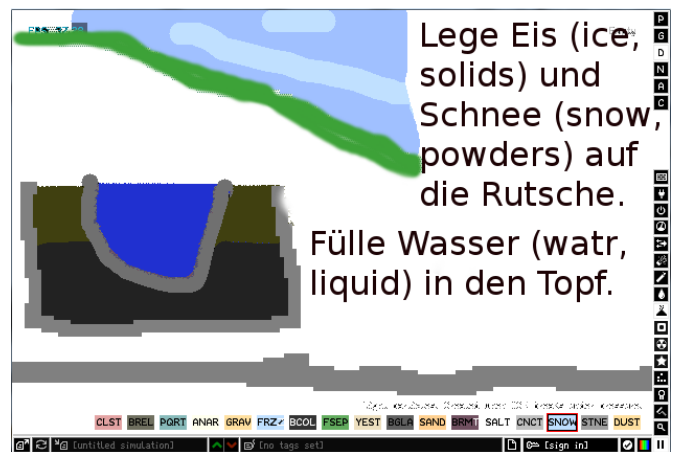
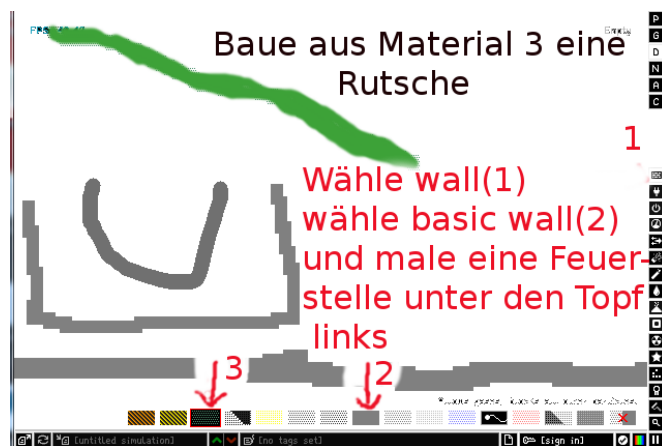
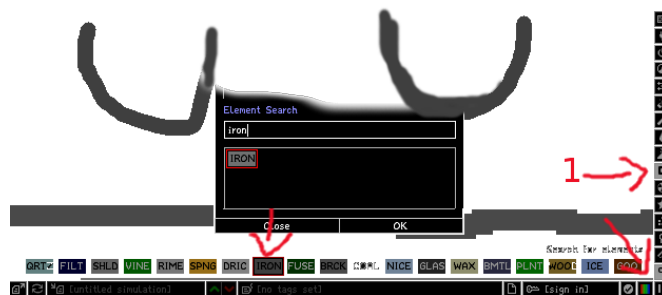
von SEBASTIAN KRÄHNSMAIER

Sebastian Krähsmaier (12 J) lernt gerade die Programmiersprache Python bei *spielend-programmieren* [1] und erforscht mittels *The Powder Toy* [2] physikalische Zusammenhänge. In diesem Tutorial versucht er die Aufgabe zu lösen, Wasser von einem Topf links in einen Topf rechts zu befördern.

FPS: 30.07

Empty

wähle aus solid(1) das material iron und zeichne damit 2 Töpfe



Lege Eis (ice, solids) und Schnee (snow, powders) auf die Rutsche.

Fülle Wasser (watr, liquid) in den Topf.



Zünde das Öl an mit Feuer (fire, explosives)

Starte die Simulation mit der Playtaste rechts unten

Erklärung: Die Flammen sollten das Öl anzünden und anschließend die Kohlen unter dem Metalltopf links. Da Metall wärmeleitend ist steigt aus dem Topf Wasserdampf auf. Die Rutsche über dem Topf ist dampfdurchlässig, der Dampf schmilzt den Schnee und das Eis. Da die Rutsche Flüssigkeiten abweist und nach rechts geneigt ist rinnt das Schmelzwasser in den Topf rechts. Hat man alles richtig gemacht dann verdampft das Wasser im linken Topf komplett und es sammelt sich Wasser im rechten Topf.

Lizenz, Quellen:



Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Quellen:

- [1] spielend-programmieren.at
- [2] powdertoy.co.uk/

Scratch 2 Tutorial

von HORST JENS

Horst JENS zeigt in diesem Tutorial (welches üblicherweise während einer Probestunde bei spielend-programmieren [1] verwendet wird) die Neuheiten bei der grafischen Programmieroberfläche Scratch 2.0 [2]. Konstruiert wird ein kleines Spiel bei welchem man Abwurfwinkel und Abwurfgeschwindigkeit einstellen muss um einen Ball auf (mehrere) Fledermäuse zu schießen.

Scratch

Die erste Version der Programmieroberfläche Scratch [1] wurde von der Abteilung Lifelong Kindergarten group [3] des MIT Media Labs (Massachusetts Institute of Technology [4]), 2003 entwickelt. Scratch ist bis zur Version 1.4 freie Software (GPL lizenziert). Der Sourcecode der neueren Version (Scratch 2.0) ist laut Scratch Homepage noch nicht verfügbar, und wird vermutlich ebenfalls frei lizenziert werden.

Scratch 2.0 läuft rein im Webbrowser, außer einer Internet Verbindung und einem modernen Webbrowser (mit installiertem Adobe Flash) ist keine Installation erforderlich. Auf der Scratch-Website wird auch eine noch experimentelle Offline-Version [5] zum Download angeboten

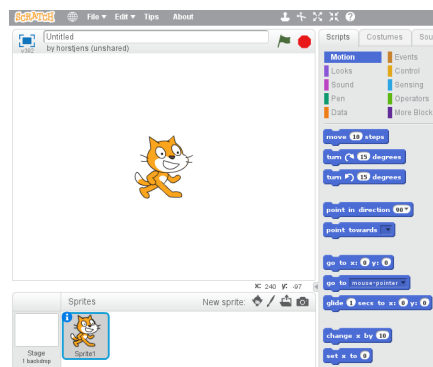
In Scratch selbst können bewegbare, animierbare Grafiken - sogenannte **Sprites** - mit Skripten programmiert werden ohne einen einzigen Befehl eintippen zu müssen. Stattdessen werden Logik-Blöcke mit der Maus aneinandergefügt, ähnlich wie Puzzle-Steine. Dadurch eignet sich Scratch sehr gut für Programmieranfänger und Schulen, um z.B. kleine Spiele, Filme oder Animationen zu erstellen.

Scratch wird mit einigen Grafiken und Soundeffekten ausgeliefert, welche üblicherweise alle **Creative Commons (cc-by-sa)** lizenziert sind,

ebenso wie das Scratch Logo. Die mit Scratch erstellten Spiele lassen sich per Knopfdruck weltweit veröffentlichen (*Share*) wodurch sie ebenfalls **cc-by-sa** lizenziert sind. Durch die *look inside* Funktion kann man jedes veröffentlichte Werk auf der Scratch Website klonen (*Fork*) und umbauen, ähnlich wie bei erwachsenen **Code Repositories**.

Account anlegen

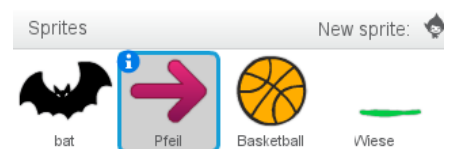
Der erste Schritt um mit Scratch zu arbeiten besteht darin sich auf der Scratch Homepage [1] (kostenlos) zu registrieren und einen Scratch Account anzulegen. Sodann kann mittels *Create Link* der Scratch Programmiereditor aufgerufen werden. Man sieht das Scratch Maskottchen, eine orange Katze. Es empfiehlt sich zu diesem Zeitpunkt das Scratch Projekt zu benennen (anstatt *Untitled* im Feld oben einen eigenen Namen hineintippen, z.B. *Wurfdemo*) sowie die Spracheinstellung auf Englisch umzustellen (kleines Weltkugel Icon direkt rechts vom Scratch Schriftzug oben links). Der Grund dafür ist dass in fast allen Programmiersprachen englische Befehlsnamen verwendet werden und man sich (und seinen Schülern) nichts gutes damit tut *Wenn* *sonst* anstatt das international gebräuchliche *If - Else* lernen zu lassen. Die Spracheinstellungen lassen sich jederzeit ändern.



Vier Sprites

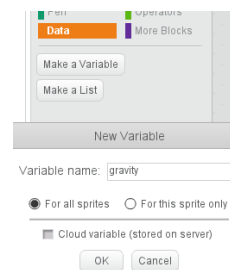
Die orange Katze kann man gleich löschen und erstellt stattdessen vier eigene Sprites: Eine Fledermaus

(bat), einen Pfeil (nach rechts schauend) ein Wurfgeschoss (Basketball) sowie einen dicken grünen Strich (Wiese). Die ersten drei Sprites habe ich mit Hilfe des kleinen *New Sprite* Icons aus dem Scratch Katalog übernommen, die Wiese hab ich mit Hilfe des Scratch Editors selbst gezeichnet (Vektormodus). Im Prinzip kann man alle Sprites selber zeichnen oder eigene Grafiken uploaden: Zu Beachten ist dabei dass man die Rechte an den Grafiken haben sollte (sonst gibt es legale Probleme sobald man das Scratch Spiel veröffentlicht) und dass das Wurfgeschoss absichtlich rund gewählt wurde: Bei einem runden Flugobjekt (Ball, Kanonenkugel) fällt es weniger auf wenn sich das Flugobjekt nicht in Flugrichtung dreht.



Noch sind alle 4 Sprites starr und unbewegt, ihnen fehlen Anweisungen (Script). Zuerst kümmern wir uns um ein paar globale Variablen:

Globale Variablen



Eine Variable kann man sich wie eine Sparsbüchse vorstellen: Ein Behälter mit einem Namen (z.B. Klassenkasse) in dem ein sich über die Zeit ändernder Betrag drin ist. Globale Variablen bedeuten bei Scratch dass alle Sprites diese Variablen sehen (und verändern) dürfen. Für dieses Beispiel werden folgende globale Variablen benötigt: Gravitation (gravity), Winkel, Kraft, Punkte und Level. Die Namen der Variablen sind beliebig wählbar. Allerdings sollte das Zielpublikum bedenken: Wenn man mit seinem Scratch-Projekt weltweit

herzeigen (share) will sind englische Namen (oder noch besser: selbst-erklärende Grafiken) besser geeignet als deutsche Namen. Man erzeugt eine Variable durch Klick auf den orangenen **data** Knopf in der Bildschirmmitte und danach durch Klick auf die weiße 'Make a Variable' Schaltfläche.



Bei Erzeugung einer globalen Variable muss der schwarze Optionskreis *for all Sprites* gesetzt sein. Hat man die vier Variablen erzeugt so sieht man sie angezeigt als orangefarbene, verschiebbare Beschriftung im Spielfeld links. Ob eine Variable überhaupt angezeigt wird kann mit einem Häkchen in der Bildschirmmitte beim Variablennamen festlegen. Die ersten drei Variablen sollen per Maus einstellbare Werte haben (Sliders): Dazu auf mit der rechten (!) Maustaste auf die Variablen-Beschriftungen klicken und erst *Slider* und dann *Set Slider min and max* anklicken (wie gewohnt mit der linken Maustaste). Bei Gravity habe ich -20.00 bis 0.00 gewählt (die 2 Nullen hinter dem **Dezimalpunkt** sind wichtig!), für Winkel Werte zwischen 0.00 und 90.00 und für Kraft Werte zwischen 0.00 und 100.00 - Punkte und Level bekommen keine Slider sondern dienen rein zur Anzeige der Variablenwerte.

Wiese

Nun zu den Scripts für die einzelnen Sprites: Am unkompliziertesten ist das Wiesen-Sprite: Es tut gar nichts und läuft einfach nur herum. Sein Daseinszweck besteht -neben hübsch ausschauen- darin den Basketball bei Berührung erkennen zu lassen dass sein Flug vorbei ist. Mit dem Vergrößer / Verkleiner Icon (oben rechts)



kann die Größe eines Sprites verändert werden. Die Wiese sollte den ganzen unteren Bildschirmrand ausfüllen. Wie jedes Sprite kann sie per *Drag and Drop* (linke Maustaste gedrückt halten) verschoben werden.

Pfeil

Relativ einfach zu programmieren ist das Skript für den (nach rechts schauenden) Pfeil. Er symbolisiert Abwurfwinkel und Abwurfgeschwindigkeit. Diese beiden *globale Variablen* lassen sich sowohl per Maus als auch per Tastatur (Pfeiltasten) verändern, wenn man folgendes Skript richtig nachbaut: Das Pfeilsprite anklicken und danach folgende Befehle zusammenklicken (siehe auch vergrößerte Abbildung **Pfeilcode** am Ende dieses Artikels):



pfeilcode

Erklärung: Falls Sie dieses Tutorial auf einem Schwarzweissdrucker ausgedruckt haben überlegen Sie sich einen Farbausdruck oder werfen Sie

einen Blick auf die (bunte) online Version dieses Artikels. Die braunen, halbrunden 'Buckel' finden Sie unter dem Menüpunkt *Events*. Die orangenen *set* (setze) und *change* (verändere) Befehle finden Sie unter *data*. Die gelbe *forever* Schleife findet man unter *Control*, den blauen *point in direction* Befehl unter *Motion* und den violetten *set size* Befehl unter *Looks*. Die grünen Minus und Plus Operatoren findet man unter -erraten- *Operators*. Auch wenn man nicht versteht was die Befehle im Einzelnen tun kann man sie (sogar ohne Lesekenntnisse) nachbasteln.

Hier die Erklärung: Die grüne Fahne (rechts oben im Spielfeld) startet das Spiel. Danach werden mit den 'set' Befehlen die Startwerte für die globale Variablen gesetzt - der letzte Spieler hat sie möglicherweise hoffnungslos verstellt. Die wie ein Schraubstock aussehende *forever* Schleife wiederholt (loop) endlos die von ihr eingezwickten Befehle: Das Pfeil-Sprite dreht sich in Richtung des angegebenen Winkels (Die Berechnung $90 - \text{Winkel}$) ist erforderlich da Scratch2 den Winkel 0 nach oben zeigen lässt und nicht nach rechts. Per Klick auf das blaue **i** Symbol beim Pfeilsprite kann folgende Detail-Ansicht geöffnet werden:



Dort den *Rotation style* auf 360 Grad (linkes Icon) einstellen.

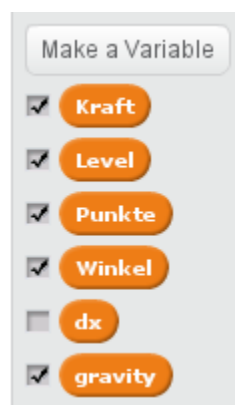
Der *set size to (100 + Kraft)%* Befehl sorgt dafür dass der Pfeil schrumpft und wächst. Das 100+ ist notwendig damit der Pfeil auch bei einem Kraftwert von 0 sichtbar bleibt.

Die vier *when key pressed* Befehle erlauben die Veränderung von Winkel und Kraft mit den Pfeiltasten.

Fledermaus



Ein ganz anderes Biest ist das Skript der Fledermaus, hier ist mehr zu tun. Das Fledermaus Sprite besteht aus 2 (oder mehr) Bildern um im Spiel animiert flattern zu können. Auf das Fledermaus Sprite klicken, oben mitte auf *Costumes* klicken und per Katalog-Icon (links neben dem Pinsel) ein zweites Fledermaus-Kostüm hinzufügen. Praktischerweise sind im Scratch-Katalog gleich zwei passende Fledermaus-Kostüme enthalten. Wer künstlerisch begabt ist kann noch mehr Fledermaus-Kostüme hinzufügen, dadurch schaut die Animation im fertigen Spiel flüssiger aus. Ein beliebter Trick ist es ein Kostüm öfter zu duplizieren (per Rechtsklick) und dann in jedem Kostüm z.B. die Augenfarbe zu ändern - dadurch entsteht später ein *glühender Augen* Effekt der gut aussieht und nicht allzu viel Arbeit erfordert.



only einstellen und den Namen *dx* vergeben. Diese Variable muss **nicht** ständig sichtbar sein, deshalb das Häkchen wegnehmen:

Wie man sieht bleiben die **globalen Variablen** sichtbar wenn man das Fledermaus Sprite anklickt. Die

private Variable *dx* wird unsichtbar wenn man zwischendurch ein anderes Sprite (z.B. die Wiese) anklickt. Das bedeutet das jedes Sprite unabhängig voneinander eine private Variable namens *dx* haben kann, ohne dass diese Variablen sich gegenseitig in die Quere kommen. *Dx* steht übrigens für delta-x und meint die Geschwindigkeit auf der X-Achse (rechts-links Bewegung). Man kann Variablen jeden beliebigen anderen Namen vergeben.

Das Batcode-Skript ist etwas aufwendiger zu programmieren (siehe auch vergrößerte Abbildung **Batcode** am Ende dieses Artikels):

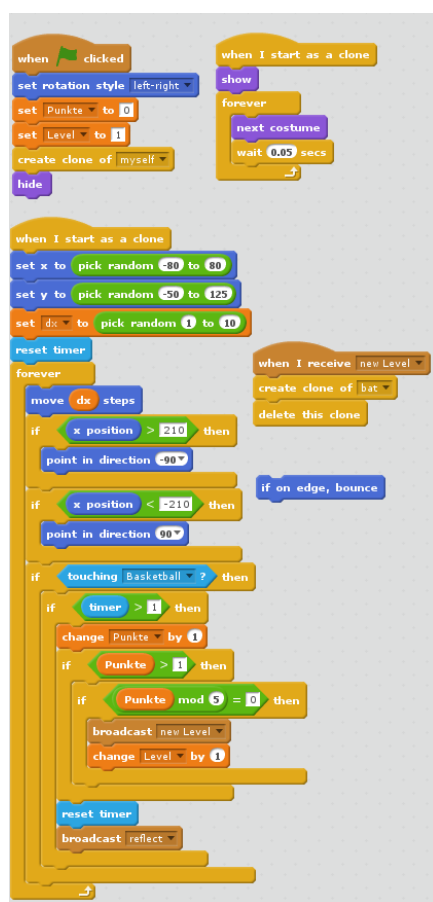


Abbildung Batcode

Den blauen *if on edge, bounce* Befehl habe ich nur als Erklärung dazugetan, er kann auch weggelassen werden. Da er nicht an einen 'Buckel' angedockt ist wird er sowie so nicht ausgeführt. Zur Erklärung: Der Code unter der grünen Fahne setzt zu Spielstart die globalen Variablen *Level* und *Punkte* auf den

Wert Null macht das Fledermaus-Sprite unsichtbar. Kurz zuvor wird aber noch ein sichtbarer *Klon* des Fledermaus-Sprites erzeugt.

Der eigentliche Programmcode beginnt mit *When i start as a clone*. Was macht der frisch erzeugte Fledermaus-Klon? Er bekommt von den grünen *pick random number* Befehlen eine zufällige Position (X und Y Koordinate, blau) zugewiesen und eine zufällige Geschwindigkeit (private Variable *dx*, orange). Danach wird per *reset timer* (hellblau) eine interne Stoppuhr auf Null gesetzt und eine sehr große *Forever*-Schleife abgearbeitet: Zuerst bewegt sich die Fledermaus mit der Geschwindigkeit *dx* in Flugrichtung (*move* Befehl, blau). Der blaue *set rotation style to left-right* Befehl bei Spielstart hat dafür gesorgt dass die Fledermaus nicht nach einer Richtungsänderung mit dem Kopf nach unten fliegt. Die zwei gelben *if* Befehle testen ob sich die blaue *x-Position* der Fledermaus in *verbotenen Bereichen* (größer als 210 oder kleiner als -210) befindet und dreht ggf. die Fledermaus in eine andere Richtung. Diesen Teil hätte eigentlich der blaue *if on edge, bounce* Befehl erledigen sollen, allerdings führte dies auf meinem Computer dazu dass die Fledermaus immer wieder in der Wand hilflos flatternd stecken blieb. Die X und Y Position eines Punktes auf dem Spielfeld erfährt man indem man den Mauszeiger dorthin bewegt und die Koordinaten rechts unten unter dem Spielfeld abliest.

Danach kommt ein vierfach verschachtelter *if* Befehl bei dem sich alles um eine **collision detection** mit dem Basketball dreht:

Da auf Berührung zwischen Fledermaus und Basketball mehrmals pro Sekunde getestet wird (pro **Frame**) kann der Basketball gar nicht so schnell von der Fledermaus abprallen (dazu später mehr) und würde pro Treffer mehrere **collision detections** auslösen. Deshalb wird mit der Abfrage der hellblauen *Timer* Variable geschaut ob seit der letzten Berührung mehr als eine Sekunde vergangen ist und erst dann weiter-

gemacht:

In diesem Fall wird die globale Variable *Punkte* um eins erhöht (change) und falls die Punktezahl glatt durch 5 teilbar ist ($\text{Punkte} \bmod 5 = 0$) dann kommt der Spieler in den nächsthöheren Level. Dabei wird aber nicht einfach nur die globale Variable *Level* erhöht sondern ein sogenannter **Event** generiert und per *broadcast* Kommando verkündet. Anschließend wird die Stoppuhr per *reset timer* (hellblau) wieder auf zurückgesetzt und noch ein *reflect* Event per *broadcast* versendet, damit der Basketball weiß dass er von der Fledermaus abprallen soll.

Parallel dazu wird ein zweiter *when i start as a clone* Befehl ausgeführt: Der sorgt dafür dass die Fledermaus sichtbar ist (show, violett) und wechselt alle 0.05 Sekunden (**Dezimalpunkt!**) das Fledermaus-Kostüm - dadurch entsteht der Flatter-Effekt oder die Animation.

Am seltsamsten ist vermutlich der braune *When i receive next Level* Befehl: Hier wird erst ein neuer Fledermaus-Klon generiert und danach der aktuelle Klon gelöscht mit *delete this clone*. Warum? Ich empfehle einfach den *delete this clone* Befehl einmal wegzulassen und das fertige Spiel ein paar Runden zu spielen: Exponentielles Fledermauswachstum droht! Meine Erklärung: Jede Fledermaus, auch die unsichtbare 'Ursprungs' Fledermaus, reagiert auf den *NextLevel-Event*. Wenn eine unsichtbare und eine sichtbare Fledermaus jeweils einen sichtbaren Klon erzeugen gibt es bei Level 2 ($2 \times 2 = 4 - 1 = 3$) schon drei Fledermäuse. Bei Level 3 ($4 \times 2 = 8 - 1 = 7$) sind es dann schon sieben sichtbare Fledermäuse, bei Level 4 schon ($8 \times 2 = 16 - 1 = 15$) fünfzehn usw.

Basketball

Nun zum schwierigsten Sprite-Skript, dem für den Basketball. Zunächst braucht der Basketball gleich drei unsichtbare *private Variablen*, nämlich *dx*, *dy* und *fliegt*. Letztere ist ein soge-

nanntes *Flag* oder auch *boolsche Variable*: Sie soll nur zwei verschiedene Werte annehmen können, nämlich: *Der Basketball fliegt* oder eben *Der Basketball fliegt nicht*. In Scratch wird das durch die beiden Werte Eins (fliegt) und Null (fliegt nicht) realisiert.



Delta-X und Delta-Y hingegen sind *Vektoren*¹³ und geben die Fluggeschwindigkeit in der X-Achse (links-rechts) bzw. in der Y-Achse (oben-unten) an. Beide zusammen-

genommen beschreiben ob der Ball *schräg fliegt*.

Würde man noch eine Variable für Delta-Z sowie eine für die Z-Position erzeugen dann ließe sich mit Scratch ein perspektivisches 3D Spiel realisieren - Ein Beispiel für den **B.E.** Unterricht finden Sie auf *verlinkt auf meiner Homepage* [6]

Der Code für den Ball benutzt die dunkel-violette *More Blocks* Funktionalität von Scratch2 um ein Subprogramm zu realisieren (siehe auch vergrößerte Abbildung **Ballcode** am Ende dieses Artikels):

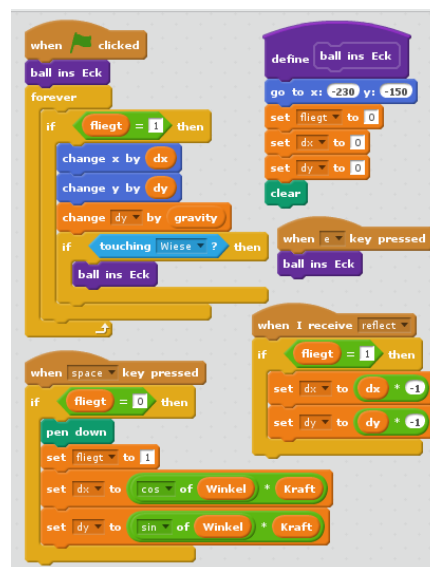


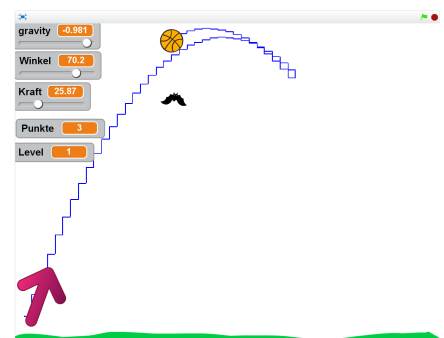
Abbildung Ballcode

Die *Ball ins Eck* Subroutine wird von drei verschiedenen Stellen aufgerufen und stellt den Ball wieder in die Ecke links unten, knapp über die Wiese.

Delta-X und *Delta-Y* werden über die Cosinus- bzw. Sinus-Funktion von Scratch berechnet: dorthin fliegt der Ball. Die Schwerkraft (gravity) zieht die ganze Zeit über den Ball nach unten und verändert sein Delta-Y. Wenn der Basketball die Wiese berührt oder die Taste *e* gedrückt wurde hört er auf zu fliegen und wird in die Ecke teleportiert. Außerdem lauscht der Basketball auf den *reflect* Event welcher von einem beliebigen Fledermaus-Klon *gebroadcastet* werden kann: In diesem Fall drehen sich Delta-X und Delta-Y um, der Ball wird reflektiert.

Geschickte Spieler können den Ball zwischen mehreren Fledermäusen reflektieren lassen.

Der grüne *Pendown* Befehl sorgt dafür dass der Ball eine (treppenförmige) Flugbahn zeichnet, der grüne *clear* Befehl löscht diese Spur wieder weg. Beide Befehle befinden sich im grünen *Pen* Menü.



Fledermaus wurde im Flug getroffen reflektiert den Basketball

Fachbegriffe

Sprite (englisch: Geist) bewegbare Computergrafik, z.B. der Mauszeiger.

Code Repository eine Website auf der (Open Source) Code-Projekte gesammelt und verwaltet werden um weltweites gemeinsames Arbeiten zu ermöglichen. Beispiel. *github.com*

Collision Detection: Kollisionserkennung, üblicherweise zwischen zwei Sprites oder zwischen einem Sprite und einer Hintergrundfarbe.

¹³Der Mathematiklehrer Ihres Vertrauens äußert sich sicher gerne ausführlichst zu diesem Thema

Frame, Framerate, Frames per Second (fps) gibt an wie viele Einzelbilder (Frames) der Computer pro Sekunde (FPS) anzeigt um die Illusion einer Bewegung zu erzeugen. Ein idealer Wert sind 60 *frames per second*, der aber speziell von schwächeren Computern und komplexen Spielen mit vielen Sprites nicht erreicht wird. Unter 10 FPS wird ein Spiel unangenehm ruckelig.

Event englisch: Ereignis. Bei Scratch berichtet ein Skript per *broadcast* (=herumschreiben, funken) allen anderen Sprites von einem Ereignis. Diese können per *when i receive*

broadcast darauf reagieren.

Lizenz, Quellen:



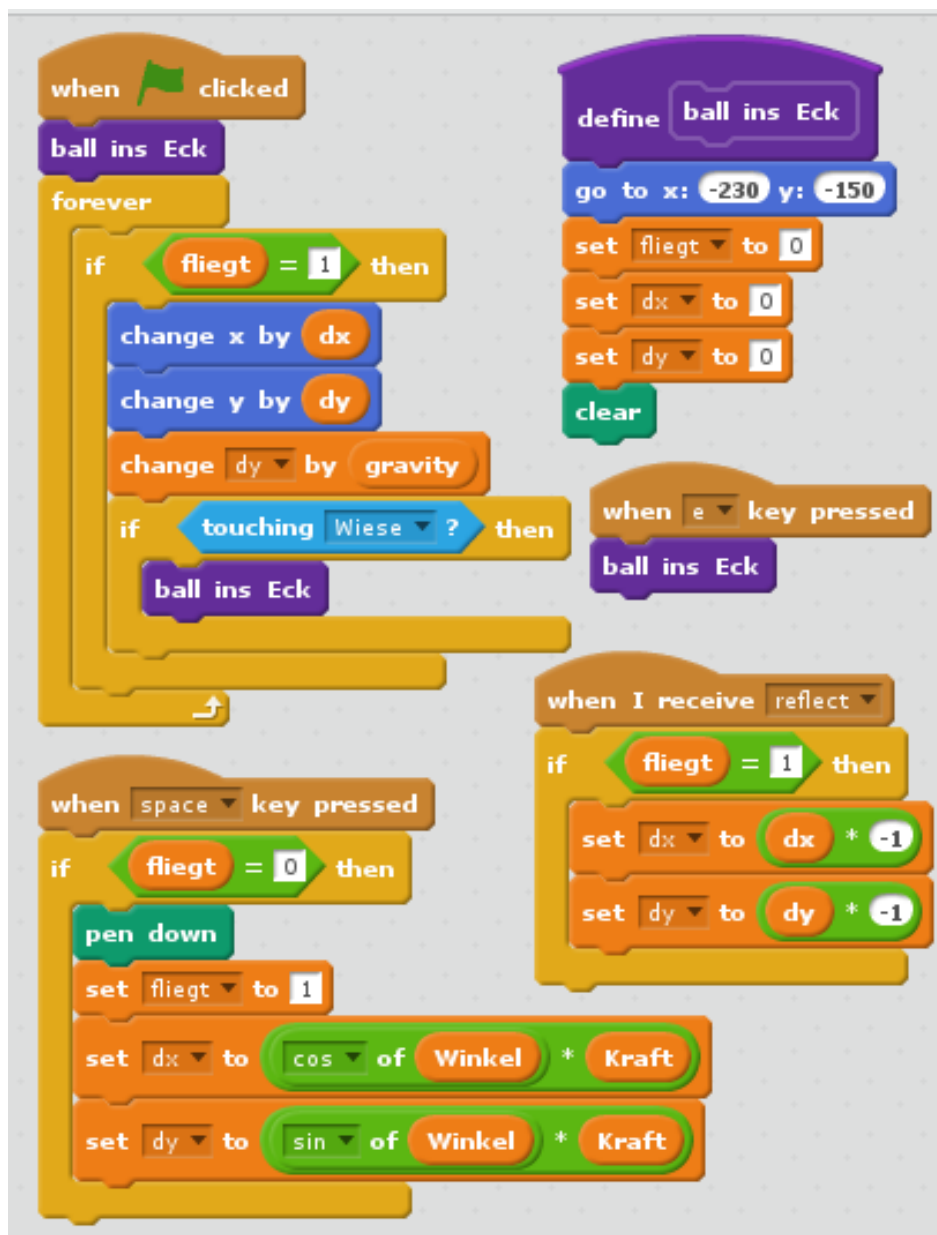
Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/deed.de>.

Download, Feedback:

R.I.S.-Journal, Ausgabe 001:
spielend-programmieren.at/de:ris:001
Download Ordner, verschiedene Formate: spielend-programmieren.at/risjournal/001/scratch
Feedback ✉ horst.jens@spielend-programmieren.at

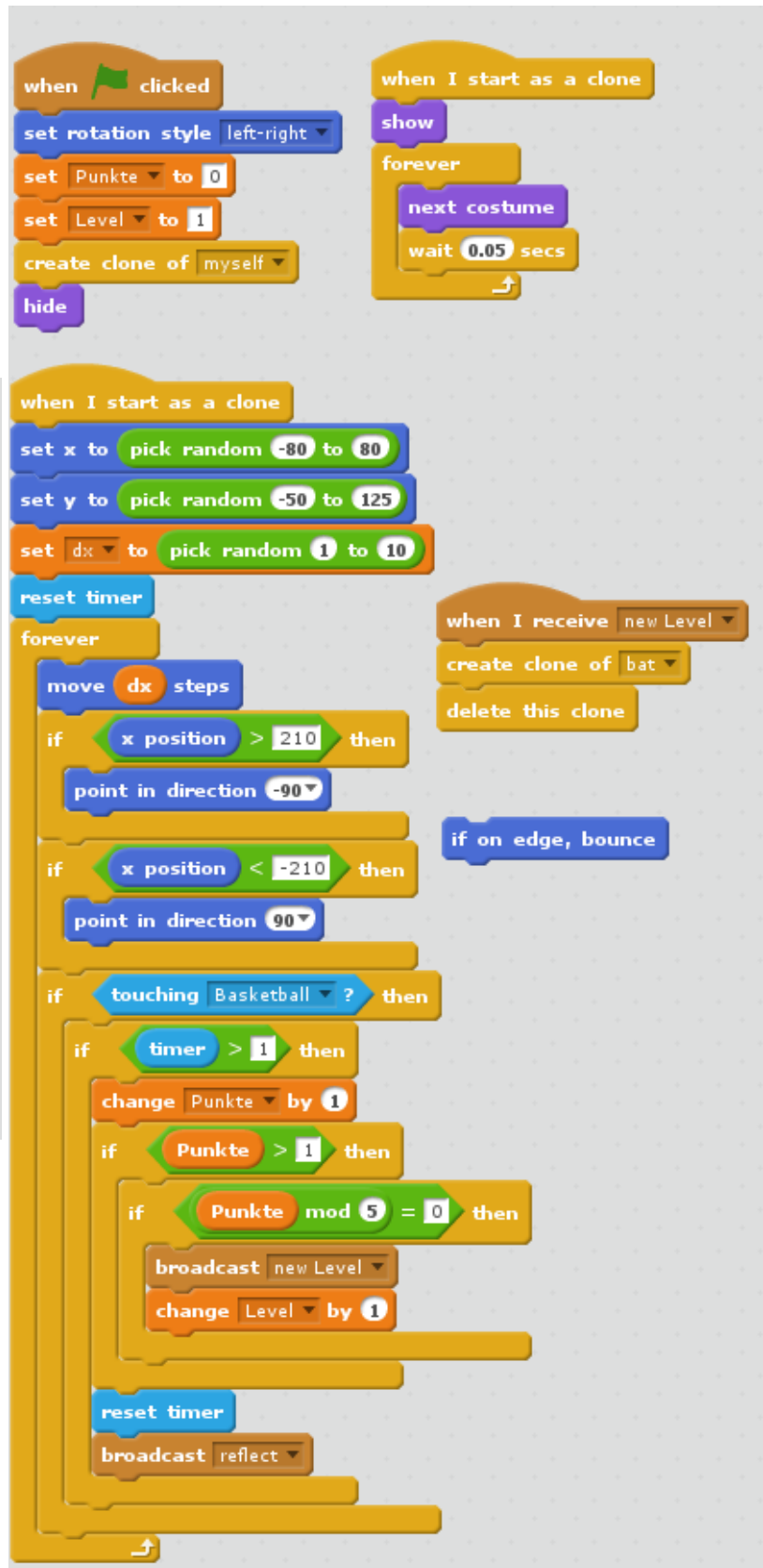
Quellen:

- [1] spielend-programmieren.at
- [2] Scratch.mit.edu
- [3] llk.media.mit.edu
- [4] www.media.mit.edu
- [5] scratch.mit.edu/scratch2download
- [6] goo.gl/ClnYs0





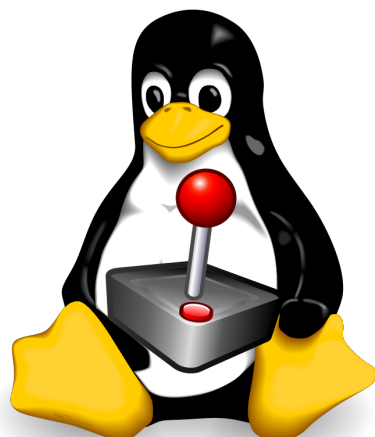
Abbildungen: Pfeilcode (links), Batcode (rechts)



Kalender

von HORST JENS

Die Firma spielend-programmieren (Horst JENS) bietet Prorammiereurse für Jugendliche sowohl während der Schulzeit als auch während der Ferien an und freut sich 2014 auf folgende Termine:



spielend-programmieren.at

Februar 2014

2014-02-01 bis 2014-02-02

Fosdem.org/2014 Brüssel, Belgien **2014-02-02** (So) Bel-me

2014-02-03 bis 2014-02-08

Semesterferien Intensiv Kurse: spielend-programmieren.at Mo-Fr, 9:00 bis 12:00 Wien

2014-02-06 (Do) subotron.com Vortrag: Indies go Playstation. Wien, MQ, quartier21, 19:00

2014-02-07 (Fr) subotron.com Vortrag: Emerging Swiss Game Design – Pitch Session, Wien, MQ, quartier21, 19:00

2014-02-20 (Do) subotron.com Vortrag: Projektanalyse österreichischer Games: 'Son of Nor', Wien, MQ, quartier21, 19:00

2014-02-21 (Fr) fsfe.org FSFE Treffen im Metalab Wien

2014-02-28 (fr) subotron.com Vortrag: Game Development: lokale, soziale & kulturelle Einflüsse, Wien, MQ, quartier21, 19:00

März 2014

2014-03-06 (Do) subotron.com Vortrag: Rovio Stars and free-to-play publishing, Wien, MQ, quartier21, 19:00

2014-03-07 (Fr) www.aec.at/u19 Einreichfrist für U19 Programmierwettbewerb

2013-03-14 (Fr) subotron.com Vortrag: Geschlechterrollen und Sexismus in Computerspielen, Wien, MQ, quartier21, 19:00

2014-03-15 bis 2014-03-16 chemnitzer.linux-tage.de/ TU Chemnitz, Deutschland

2014-03-20 (Do) subotron.com Roundtable: Ausbildungsmöglichkeiten für die Gamesindustrie 2014, Wien, MQ, quartier21, 19:00

2014-03-21 (Fr) fsfe.org FSFE Treffen im Metalab Wien, 19:00

2014-03-26 (Mi) documentfreedom.org weltweit

2014-03-28 (Fr) subotron.com Vortrag: Das Spiel mit dem 'Virtual Other in Mind', Wien, MQ, quartier21, 19:00

April 2014

2014-04-03 (Do) subotron.com Ausflug: Bildungs-Tour in Wiener Game-Developer-Studios. Treffpunkt: 19:00h: Fernkorngasse 10, 1100 Wien, Anmeldung erwünscht

2014-04-04 bis 2014-04-05

LinuxTage.at Graz, FH Joanneum

2014-04-12 bis 2014-04-22

Osterferien Kurse spielend-programmieren.at Mo-Fr, 9:00 bis 12:00 Wien

2014-04-18 (Fr) fsfe.org FSFE Treffen im Metalab Wien, 19:00

Mai 2014

2014-05-16 (Fr) fsfe.org FSFE Treffen im Metalab Wien, 19:00

2014-05-20 bis 2014-05-21 open-commons.linz.at Linz

Juni 2014

2014-06-20 (Fr) fsfe.org FSFE Treffen im Metalab Wien, 19:00

2014-06-28 bis 2014-08-30

Sommerferien Intensiv Kurse: spielend-programmieren.at Mo-Fr, 9:00 bis 12:00 Wien

Juli 2014

2014-07-07 bis 2014-07-09

d4e.at/schulungen Open Source for Education, BRG Weiz (Steiermark)

2014-07-21 bis 2014-07-27

ep2014.eurpython.eu Europython Konferenz, Berlin

August 2014

2014-08-28 bis 2014-08-30

d4e.at/schulungen Weizer Knoppix-tage, BRG Weiz (Steiermark)

Sponsoren

Das RIS-Journal dankt folgenden Sponsoren dieser Ausgabe:



Wukonig.com, die Internetagentur aus Österreich