

Topthemen dieser Ausgabe

GIMP automatisieren

Wie einfach es ist, mit GIMP Bilder zu verbessern oder auch zu verschönern, wurde im Artikel „GIMP in 90 Minuten (kennenlernen)“ von Karsten Günther in freiesMagazin 01/2010 gezeigt. Verwendet man GIMP dann regelmäßig, kommt schnell der Wunsch auf, immer wiederkehrende Handgriffe, wie z. B. das Erstellen eines Rahmens oder das Einfügen eines Copyright-Vermerks, zu automatisieren. Auch hier hält GIMP das passende Werkzeug bereit: Skript-Fu. ([weiterlesen](#))

Seite 14



MegaGlest – Ein historisch nicht ganz korrektes Strategiespiel

Wer unter Linux spielt und Strategiespiele à la Age of Empires mag, der könnte das ein oder andere mal bereits über den Begriff Glest gestolpert sein. Glest ist ein 3-D-Echtzeitstrategiespiel, dessen Entwicklung bereits im Jahr 2001 begann. Im Moment ist es ruhig um das Open-Source-Projekt geworden. Fans rund um das Spiel fassten sich wegen der Stille um das Projekt ein Herz und gründeten, basierend auf Glest, einen separaten Entwicklungszweig: MegaGlest genannt. ([weiterlesen](#))

Seite 25



Wünsch Dir was – SUSE Studio als Tool zum eigenen Linux

Seit einiger Zeit, und ein wenig im Verborgenen, hat Novell mit SUSE Studio eine interessante Plattform bereit gestellt. Mit deren Hilfe können sich Linux-Freunde schnell und unkompliziert ihre eigene SUSE-Distribution zusammenklicken. Das funktioniert tatsächlich ebenso einfach wie es sich anhört und in dem Artikel wird gezeigt, wie es geht. ([weiterlesen](#))

Seite 29



Editorial

Von Viren und fremden Mächten unterwandert

Ein Albtraum? Oder nur die Realität?

Der CEO eines proprietären Betriebssystems ahnte es schon immer, fremde Mächte sind unterwegs und unterwandern alles ... So sehen wir das auch und stellen heute fest, dass wir in der vorliegenden Septemberausgabe von **freiesMagazin** den ersten Artikel zu Software veröffentlichen, die nur unter Windows läuft – Freie Software wohlgemerkt. Mit dem Artikel „Notepad++ und Notepad2 – Open-Source-Editoren für Windows“ auf **Seite 36** berichtet Jochen Schnelle über zwei freie Editoren, die – wie der Titel schon andeutet – unter Microsoft Windows laufen.

Man mag diesen Ansatz begrüßen oder verdammen, aber man sollte es zumindest nicht ignorieren, dass Freie Software auch auf proprietären Systemen läuft. Was halten Sie davon, liebe Leser? Fänden Sie es interessant und begrüßenswert, wenn wir in Zukunft weitere Artikel dieser Art in **freiesMagazin** veröffentlichen würden? Teilen Sie uns Ihre Meinung mit und nehmen Sie am besten gleich an der Umfrage dazu teil [1].

Ist die GPL viral?

Letzten Monat erhielten wir einen Leserbrief von Lutz Horn (im Übrigen einen von exakt zwei, was doch sehr wenig ist), in dem er einige Anmerkungen zur Rezension „Die Anarchie der Hacker“ aus **freiesMagazin** 08/2010 [2] hatte.

Zum einen wies er darauf hin, dass der Autor Dominik Wagenführ vergessen hatte, auf das Copyleft-Konzept [3] einzugehen. In der Tat hätte es nicht geschadet, dieses zu erwähnen, um die GNU General Public License [4] besser zu verstehen.

Die zweite Anmerkung betraf die schlechte Verwendung des Begriffes „viral“ für die GPL. Der Ausdruck stammt von Microsoft-Manager Craig Mundie aus dem Jahr 2001 [5], der den viralen Effekt der GPL als Bedrohung für das geistige Eigentum von Firmen angesehen hatte. Als Alternative schlug Lutz Horn die Beschreibung „*Copy-left ist freiheitsübertragend*“ vor.

Sie können uns gerne sagen, ob Sie die Sache mit der „viralen“ Bezeichnung auch so negativ sehen oder es nicht doch zur GPL passt. Schreiben Sie uns einfach an redaktion@freiesMagazin.de oder nutzen Sie die Kommentarfunktion am Ende der Seite.

Und nun wünschen wir Ihnen natürlich wieder viel Spaß mit der neuen Ausgabe.

Ihre **freiesMagazin**-Redaktion

LINKS

- [1] <http://www.freiesmagazin.de/20100905-freie-software-fuer-windows>
- [2] <http://www.freiesmagazin.de/freiesMagazin-2010-08>
- [3] <http://www.gnu.org/copyleft/copyleft.html> 
- [4] <http://www.gnu.org/licenses/gpl.html> 
- [5] <http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.msp> 

Inhalt

Linux allgemein

Der August im Kernelrückblick S. 3

Anleitungen

Symbian für Schlangenbeschwörer S. 5

MP3s von Amazon herunterladen S. 10

GIMP automatisieren S. 14

Software

MegaGlest – Ein Strategiespiel S. 25

SUSE Studio als Tool zum eigenen Linux S. 29

Notepad++ und Notepad2 S. 36

Community

Rezension: Python – Essential Reference S. 39

Magazin

Editorial S. 2

Veranstaltungen S. 41

Konventionen S. 41

Impressum S. 42

Das Editorial kommentieren 



Der August im Kernelrückblick von Mathias Menzer

Basis aller Distributionen ist der Linux-Kernel, der fortwährend weiterentwickelt wird. Welche Geräte in einem halben Jahr unterstützt werden und welche Funktionen neu hinzukommen, erfährt man, wenn man den aktuellen Entwickler-Kernel im Auge behält.

Linux 2.6.35

Der August war keine zwei Tage alt und hatte schon die Freigabe des Kernels 2.6.35 gesehen. Dessen Neuerungen spielen sich zu größten Teil hinter den Kulissen, kaum sichtbar für die meisten Anwender, ab.

Diese dürften noch am ehesten die nun nutzbaren Energiesparfunktionen einiger Radeon-Grafikkarten und AMD-Prozessoren bemerken. Dagegen eher im Hintergrund finden sich Verbesserungen der Netzwerkkommunikation. Hier stehen mit „Receive Packet Steering“ und „Receive Flow Steering“ zwei neue Techniken für die Optimierung der Arbeitsverteilung auf mehrere Prozessoren oder Kerne zur Verfügung. Receive Packet Steering verteilt die Bearbeitung von Netzwerkpaketen möglichst gleichmäßig über die verfügbaren Prozessoren/Kerne, Receive Flow Steering dagegen ordnet diese zu einem Strom von zusammengehörigen Netzwerkpakete („Flow“) der CPU zu, die auch die mit dieser Kommunikation betraute Anwendung abarbeitet (siehe „Der Mai im Kernelrückblick“, [freiesMagazin](#) 06/2010 [1]). Ebenfalls in den

Bereich Netzwerk fällt die Unterstützung für Multicast-Routing-Instanzen. Hier war man bislang auf Userspace-Daemons angewiesen, um mit Multicasts (Verbindungen von einem Sender an eine bestimmte Gruppe von Empfängern) umzugehen [2]. Auch verfügt der Linux-Kernel jetzt über native Unterstützung für das „Layer 2 Tunneling Protocol“ (L2TP) [3] und ermöglicht damit den Aufbau virtueller privater Netzwerke (VPN), also die Verbindung von Rechnern über das Internet hinweg zu einem virtuellen lokalen Netzwerk (VLAN).

Weniger Beachtung fand die Einführung von „delayed logging“ in das Dateisystem xfs. Diese Methode, um das Journaling-Verhalten von ext2/ext3 und ReiserFS im Kleinen nach, die nicht sofort bei der Änderung von Daten auf den Datenträger schreiben, sondern mit leichter Verzögerung. Dadurch wird das System weniger belastet, Schreibvorgänge laufen für den Anwender schneller ab. Stürzt jedoch das System ab, bevor die Änderungen tatsächlich physisch auf den Datenträger geschrieben wurden, so kann dies zu Datenverlust führen, wie es zum Beispiel bei der Einführung von ext4 passierte, das Änderungen um bis zu 60 Sekunden verzögert geschrieben hatte (siehe „Ein Tuz für den Kernel“, [freiesMagazin](#) 04/2009 [4]).

Für eine bessere Speicherverwaltung sorgt „memory compaction“, das versucht, belegte Pages

Kurz erläutert: „Speicherleck“

Wird ein Bereich des Arbeitsspeichers von einem Programm belegt, jedoch nicht verwendet und/oder auch nicht mehr freigegeben, so spricht man von einem Speicherleck oder „Memory Leak“. Grund dafür kann ein Fehler im Programm sein, das sich dann zwar einen Speicherbereich reserviert, jedoch den Zeiger, der die Adresse zu diesem Bereich enthält, verliert oder überschreibt. Wird dann durch einen veränderten Zeiger ein anderer Speicherbereich ausgelesen oder gar beschrieben, so können Probleme in ganz anderen Anwendungen auftreten, wodurch das Auffinden des ursprünglichen Problems sehr schwer ist.

zusammenzufassen. Dadurch stehen dann größere Blöcke freien Speichers zusammenhängend zur Verfügung, während der belegte Speicher ebenfalls an einem Stück ist. Die Zuweisung größerer Speicherbereich wird dadurch einfacher, da diese dann zusammenhängend zur Verfügung stehen.

Verschiedene Verbesserungen gab es an Entwicklerwerkzeugen, z. B. dem Leistungsmonitor „perf“, der nun einen Live-Monitor bietet und auch Gäste der Virtualisierungstechnik KVM (Kernel-based Virtual Machine) beobachten kann, sowie dem Kernel-Debugger „Kgdb“, der nun den von SGI [5] entwickelten Debugger „KDB“ als Aufsatz erhalten hat. Letzterer ermöglicht es, das Frontend auf dem zu untersuchenden System zu be-



treiben, was bei Kgdb bislang nicht möglich war. Eine weitergehende Auflistung der neuen Funktionen liefert hier wieder einmal die Seite Kernel Newbies [6].

Linux 2.6.36

Die Entwicklung des nächsten Kernels ist derweil bereits angelaufen. Zwei Wochen nach der Veröffentlichung von 2.6.35 schloss Torvalds das Merge Window mit der Freigabe des 2.6.36-rc1 [7] ab. Die übliche Mail zur Vorabversion gab es diesmal aufgrund von Problemen mit einem Speicherleck jedoch nicht, wie Torvalds bei der Veröffentlichung des -rc2 [8] mitteilte. Auch diesmal will Torvalds wieder standhaft bleiben und weiterhin nur Korrekturen während der weiteren Entwicklung in den Kernel aufnehmen, wobei er jedoch bei einigen Funktionen im Zusammenhang mit VFS (Virtual File System), einer Abstrahierungsschicht für Anwendungen zum Zugriff auf die Dateisysteme des Linux-Systems, ein Auge zugedrückt hat.

Als eine der hervorstechendsten Neuerungen sieht Torvalds „Fanotify“, das Dateisysteme auf Änderungen überwacht und gegebenenfalls Benachrichtigungen an Prozesse und Anwendungen versendet, die im Kontext des Anwenders laufen. Weiterhin kann „Fanotify“ auch das Öffnen bestimmter Dateien verhindern. Eine Anwendung hierfür sind Malware- und Viren-Scanner, die eine Echtzeitüberwachung des Dateisystems durchführen. Trotzdem verkündete Torvalds seine Absicht, in den nächsten Jahren keine weiteren Benachrichtigungsmechanismen mehr aufzu-

nehmen, es sei denn, sie würden mit einer besonders guten Begründung eingereicht werden.

Ebenso bemerkenswert ist „workqueue“, ein Hilfsmittel zur Prozessverwaltung, das komplett überarbeitet wurde und durch eine Minimierung der neu erzeugten Prozesse Redundanzen verhindert und damit die Leistung auf Mehrkern- und Mehrprozessor-Systemen verbessert. Daneben hat mit dem „Out-of-Memory Killer“ ein neuer Mechanismus zum Freigeben von Arbeitsspeicher Einzug gehalten, der das Verhalten des Systems verbessern soll, wenn im RAM kein freier Platz mehr zur Verfügung steht.

Nach langen Jahren konnte nun AppArmor den Weg in den offiziellen Kernel finden; seit 2.6.36-rc1 ist es Bestandteil des von James Morris gepflegten Security-Subsystems [9]. AppArmor basiert auf den Linux Security Modules, einem Framework, das die Werkzeuge zur Umsetzung von Mandatory Access Control (MAC) liefert, womit die Zugriffsberechtigungen in einem System basierend auf Regeln verwaltet werden können. Damit müssen zum Beispiel openSUSE oder Ubuntu künftig zumindest mit diesem Kernelmodul nicht mehr selbst den Kernel ihrer Distribution patchen.

LINKS

- [1] <http://www.freiesmagazin.de/freiesMagazin-2010-06>
- [2] <http://de.wikipedia.org/wiki/Multicast>
- [3] http://de.wikipedia.org/wiki/Layer_2_Tunneling_Protocol

- [4] <http://www.freiesmagazin.de/freiesMagazin-2009-04>
- [5] http://de.wikipedia.org/wiki/Silicon_Graphics
- [6] http://kernelnewbies.org/Linux_2_6_35 
- [7] <http://www.pro-linux.de/news/1/16041/linux-kernel-2636-tritt-in-die-testphase-ein.html>
- [8] <http://lkml.org/lkml/2010/8/22/105> 
- [9] <http://www.pro-linux.de/news/1/15985/apparmor-kommt-in-offiziellen-linux-kernel.html>

Autoreninformation



Mathias Menzer wirft gerne einen Blick auf die Kernel-Entwicklung, um mehr über die Funktion von Linux zu erfahren und seine Mitmenschen mit seltsamen Begriffen und unverständlichen Abkürzungen verwirren zu können.

Diesen Artikel kommentieren





Symbian für Schlangenbeschwörer von Christian Imhorst

Vergleicht man Symbian mit neueren Betriebssystemen für Smartphones, wirkt es doch recht altbacken. Böse Zungen behaupten sogar, dass es das Windows 98 der mobilen Betriebssysteme sei, weil die Oberfläche seit gefühlten 10 Jahren gleich aussieht. Die Unterstützung von Open-Source-Software ist bei Symbian allerdings vorbildlich modern. Zum einen ist das Betriebssystem selbst mittlerweile Open Source [1] und zum anderen werden etliche Werkzeuge an die Hand gegeben, um freie Software zu schreiben. Eines dieser Werkzeuge heißt Python, das Nokia 2005 auf die S60-Plattform portiert hat. Da die Einarbeitungszeit für Programmieranfänger kürzer ist als in C++, kann man schnell lauffähige Programme entwickeln, die durch Module flexibel erweiterbar sind. Es ist die perfekte Gelegenheit für Freunde von GNU/Linux, Solaris oder einem BSD-System, eigene Programme zu schreiben und ihr Handy für ein größeres Aufgabengebiet fit zu machen.

Installation

Nachdem man überprüft hat, ob das eigene Handy zur S60-Reihe gehört [2], benötigt man die Datei `PythonForS60_2.0.0.tar.gz` von der Webseite `garage.maemo.org` [3]. Von der Versionsnummer darf man sich nicht irritieren lassen; Python für S60 2.0 liegt Python 2.5.4 zugrunde. Nach dem Entpacken überträgt man

die Dateien `Python_2.0.0.sis`, `pips.sis` und `PythonScriptShell_2.0.0_3_0.sis` aus dem Ordner `PyS60Dependencies` entweder per Bluetooth, USB-Kabel oder WLAN [4] auf das Telefon. Anschließend installiert man sie in dieser Reihenfolge in den Telefonspeicher. Die Installation muss in den Telefonspeicher und darf nicht auf die Speicherkarte erfolgen, da Python-Programme ansonsten Probleme haben, den Python-Interpreter zu finden.

Nach der Installation

Jetzt gibt es verschiedene Möglichkeiten, Python-Programme auf dem Handy auszuführen. Die einfachste ist, das Programm gleich auf dem Telefon zu tippen. Dazu öffnet man die Python-Shell und wählt „Interactive console“. Je nach Tastatur kann das Tippen auf dem Handy sehr anstrengend sein; der schöne Nebeneffekt ist aber, dass man auch in der S-Bahn oder an der Supermarktkasse mit einem Python-Editor wie Ped [5] programmieren kann. Die Menschen um einen herum denken dann, dass man sehr beliebt sein muss, weil man eine SMS nach der anderen schreibt, und man wird ausnahmsweise mal nicht für einen vereinsamten Ober-Geek gehalten.

Eine Python-Shell fürs Handy

Im Lieferumfang von Python für S60 gibt es zwar schon eine Bluetooth-Konsole [6], die man auch sehr komfortabel mit den PUTools bedienen kann [7], aber was macht man, wenn kein Bluetooth, dafür aber WLAN vorhanden ist? Man rich-

tet sich einfach eine Python-Shell übers WLAN ein. Die Verbindung zur WLAN-Shell steht auch schneller als die über Bluetooth, was ein entscheidender Vorteil ist, selbst wenn der Rechner über eine Bluetooth-Schnittstelle verfügt.

Damit die Python-Shell über WLAN funktioniert, benötigt man das Programm `netcat`, das im Normalfall unter GNU/Linux, Unix oder MacOS X schon installiert ist. Mit dem Befehl `nc` startet man das Kommando, um Daten mit TCP im Netzwerk zu übertragen:

```
# stty raw -echo ; nc -l 1025 ; ↵
stty sane
```

Auf einigen GNU/Linux-Rechnern muss man noch den Schalter `-p` hinzufügen und `nc -l -p 1025` eingeben. Bevor man das Terminal via `netcat` in eine Python-Shell des Handys verwandeln kann, werden dessen Einstellungen mit dem Systembefehl `stty` neu gesetzt. Dabei schaltet die Option `raw` die Pufferung aus, damit eine Eingabe Zeichen für Zeichen an die Shell weitergereicht wird. Buchstaben werden dabei uninterpretiert weitergeleitet, was soviel heißt, dass es kein Newline-Zeichen oder ähnliches mehr gibt und `[Strg] + [D]` zum Beispiel keinen Datenstrom mehr beendet. Die Option `echo` verhindert, dass die eingegebene Taste nochmal im Terminal angezeigt wird. Anschließend startet `netcat`, um auf hereinkommende Verbindungsanfragen am Port 1025 zu lauschen. Im letzten Teil des Komman-



dos startet man noch einmal **stty**, diesmal aber mit der Option **sane**, um die Pufferung des Terminals wieder herzustellen. Damit kann man die Python-Shell erneut mit der Tastenkombination **Strg+D** beenden.

Das folgende Skript für Ubuntu 10.04 vereinfacht die Verbindungsanfrage:

```
#!/bin/bash
# Script: wifishell.sh
# Object: Starts TCP/IP Console
# for S60 mobiles afterwards
# start wifishell.py on your
# S60

IP=$(ifconfig | grep 'inet ' | \
grep -v '127\.0\.0\.1' | cut -d\
: -f2 | awk '{ print $1}')
```

```
PORT="1025"

echo "Your IP address is: $IP "
echo "Start now wifishell.py on\
your phone ..."
```

```
stty raw -echo ; nc -l 1025 ; \
stty sane
```

Listing 1: *symbian-wifishell.sh*

Im ersten Teil wird die IP-Adresse im eigenen Netzwerk ermittelt und in der Variable **IP** gespeichert, was aber nur für GNU/Linux-Rechner gilt. Verwendet man ein Betriebssystem wie MacOS X, Free-, Open- oder NetBSD, muss man stattdessen folgende Zeile in das Skript einfügen:

```
IP=$(ifconfig | grep -E 'inet\
.[0-9]' | grep -v \
'127\.0\.0\.1' | awk '{ print \
$2}')
```

Als Benutzer von OpenSolaris verwendet man folgende Zeile:

```
IP=$(ifconfig -a | grep 'inet ' \
| grep -v '127\.0\.0\.1' | awk \
'{ print $2}')
```

Wie bereits erwähnt, muss man bei manchen GNU/Linux-Betriebssystemen darauf achten, dass der Aufruf von **netcat** im zweiten Teil des Skripts mit den Schalter **-p** geschieht.

Das Gegenstück in Python, das Skript **wifishell.py**, muss auf das Handy kopiert werden. Wie alle anderen Python-Skripte auch, sollte es dort auf der Speicherkarte im Verzeichnis **E:\Python** liegen. Sobald **wifishell.sh** auf dem GNU/Linux-Rechner oder der Unix-Maschine nach Anfragen auf dem Port 1025 lauscht, startet man über die Python-Shell auf dem Telefon das Skript **wifishell.py**, um eine Verbindung zum Terminal auf dem Computer herzustellen. Läuft alles glatt, verwandelt sich das Terminal in eine Python-Shell auf dem Handy.

```
# Script: wifishell.py
# Object: Starts TCP/IP Console
# for S60 mobiles before start
# wifishell.sh on your Unix box
```

```
import btconsole, appuifw
import sys
try: # next lines are important\
to select an access point
    sys.modules['socket'] = \
__import__('btsocket')
except ImportError:
    pass
import socket

ip = appuifw.query(u"IP address\
:", "text")
port = 1025

sock=socket.socket(socket.\
AF_INET, socket.SOCK_STREAM)
sock.connect((str(ip), port))
btconsole.\
run_with_redirected_io(sock,\
btconsole.interact, None, None, \
locals())
sock.close()
```

Listing 2: *symbian-wifishell.py*

Im Terminal sieht das dann so aus:

```
$ stty raw -echo ; nc -l 1025 ; \
stty sane
Python 2.5.4 (r254:67916, Nov \
6 2009, 04:18:57) [C] on \
symbian_s60
>>> import appuifw
>>> appuifw.note(u'Hallo Welt\
!', 'info')
```



Der erste Programmiererergebnis mit Python. 🔍

Hallo Welt!

Als erstes Programm in Python für S60 schreibt man klassischerweise eines, das „Hallo Welt!“ auf dem Bildschirm ausgibt. Das wird in diesem Fall schon mit einer grafischen Oberfläche gemacht, was ganz leicht ist. Man lädt dazu einfach das Modul **appuifw**.

```
>>> import appuifw
>>> appuifw.note(u'Hallo Welt!
!', 'info')
```

Im ersten Teil des Befehls erzeugt die Funktion **note()** des Moduls **appuifw** ein Hinweifenster. Die Zeichenkette in der Klammer muss Unicode sein, daher das vorangestellte „u“. Die zweite Zeichenkette gibt an, dass es sich beim Fenster nur

um eine Information handelt. Eine Aktion des Benutzers ist nicht gefordert und das Fenster verschwindet nach einer Weile von allein.

Neben **appuifw** für das Erstellen von Benutzeroberflächen können noch weitere spezielle Module für PyS60 in Programme importiert werden. Das sind **e32** als Schnittstelle zum Betriebssystem, **sysinfo** liest Geräteinformationen aus, **audio** zeichnet Klänge auf und spielt sie ab, **e32calendar** ermöglicht den Zugriff auf den Kalender, **camera** auf die Kamera, **contacts** auf Kontakte, **e32dbm** auf die Datenbank, **location** auf die Standortinformationen, **messaging** auf SMS-Funktionen, **telephone** auf die des Te-

lefons und **graphics** zeichnet grafische Elemente [8].

Mit jedem einzelnen Modul kann man eine Menge Sachen anstellen, zum Beispiel das Handy etwas sagen zu lassen:

```
import audio
audio.say(u"Hello World!")
```

Listing 3: *symbian-hw.py*

Oder eine MP3-Datei abspielen:

```
import audio, e32
fn = u"e:\\Musik\\mymusic.mp3"
sound = audio.Sound.open(fn)
sound.play(times=1)
```

```
e32.ao_sleep(sound.duration()
/1000000 + 3)
sound.close()
```

Listing 4: *symbian-mp3.py*

Oder ein kleines Skript schreiben, das per SMS um Hilfe ruft:

```
import messaging, appuifw
contact_number = "+49123456789"
contact_name = "Linus"
message = (u"SOS, ruf mich an
.")
messaging.sms_send(
contact_number, message, name=
contact_name)
appuifw.note(u"Die Nachricht
wurde gesendet", "info")
```

Listing 5: *symbian-sos.py*

Dieses Skript könnte man weiter ausbauen: Statt um Hilfe zu rufen, könnte es eine automatisierte Antwort auf bestimmte SMS geben, oder man erweitert die Hilfe-SMS um eine Standortbestimmung mit dem Modul **positioning**, wenn das Handy GPS unterstützt, um seinen Standort mitzusenden.

Die Akku-Ladung wird normalerweise mit Strichen oder Balken angezeigt. Wer es genau wissen will, schreibt einfach ein kleines Skript mit **sysinfo**:

```
import appuifw, sysinfo
battery = sysinfo.battery()
```

```
appuifw.note(u'Battery: ' + str(
(battery) + ' %', 'info')
```

Listing 6: *sybian-bat.py*

Als Dateien speichert man die Skripte in das Verzeichnis **E:\Python** auf dem Handy. Die Python-Shell schaut bevorzugt in diesem Ordner nach. Ausgeführt werden sie dann in der Python-Shell auf dem Handy, indem man „Run script“ und anschließend das Skript auswählt.

SIS-Pakete selber bauen

Nun kann es ganz schön lästig sein, immer diesen Weg einzuschlagen, wenn man mal kurz den Ladezustand überprüfen will. Einfacher ist es doch, gleich das Skript ohne Umweg über die Python-Shell zu starten. Dabei hilft das Programm **ensymbble** [9]. Bei vielen Distributionen ist es in den Paketquellen enthalten, da es aber „nur“ ein Python-Skript ist, kann man die Datei auch einfach von der Webseite herunterladen. Mit **ensymbble** gibt es allerdings ein kleines Problem: Das Skript benötigt Python 2.5. Das wird besonders diejenigen nicht freuen, die Ubuntu 10.04 „Lucid Lynx“ benutzen, da dort die älteste Python-Version 2.6 ist. Wer kein Python 2.5 besitzt, sollte es kompilieren, und zwar mit Zlib-Unterstützung. Die freie Programmibliothek zum Komprimieren und Dekomprimieren von Daten ist bei Python 2.5 standardmäßig deaktiviert, wird aber von **ensymbble** benötigt [10].

Ist **ensymbble** heruntergeladen und steht Python 2.5 mit Zlib-Unterstützung bereit, müssen nur noch die Dateien **ensymbble.py**, ein Skript (zum



Das hochgeladene Skript findet man im Installationsordner wieder. 🔍

Beispiel das Skript für den Ladezustand des Akkus, das man **batinfo.py** nennen kann) und eine SVG-Datei als Icon im selben Ordner liegen. Für die SVG-Datei kann man sich für den Anfang gut bei den Icons des GNOME-Projekts unter **/usr/share/icons/gnome/scalable** bedienen, wenn man zu bequem ist, ein eigenes zu basteln. Dann kann die Verwandlung des Skripts in eine Installationsdatei für Symbian beginnen:

```
$ python2.5 ensymbble.py py2sis ~
--version=1.0.0 --icon=battery-
low.svg --caption="Battery Info"
" --vendor="Christian Imhorst" ~
batinfo.py batinfo.sis
```

Die Datei **batinfo.sis** lädt man anschließend auf das Handy und installiert es in den Telefonspeicher. Leider läuft das Programm nicht, wenn

man es auf die Speicherkarte installiert, und es funktioniert natürlich nur auf Telefonen, auf denen auch Python für S60 bereits installiert ist. Nach der Installation findet man das Programm mit eigenem Icon im Installationsordner.

Zum Schluss ...

... noch ein kleines Programm, um die Kalenderdatenbank des Telefons auszulesen und in einer Datei zu sichern. Die Termine kann man anschließend in den Kalender von Evolution oder Thunderbird importieren. Für solche kleinen Aufgaben ist Python für S60 einfach ideal:

```
# vcalendar export
import e32calendar, appuifw

cal = e32calendar.open()
if len(cal) == 0:
    appuifw.note( u"No calendar
entries.", "info" )
id_list=list()
for id in cal:
    id_list.append(id)

vCal = cal.export_vcalendars(
tuple(id_list))

# define the directory and file
name
vCalendar = u"e:\\vCal.vcf"

# create file
file = open(vCalendar, 'w')
```




```
# write vcal into file and
close the file
file.write(vCal)
file.close()

appuiw.note( u"Success!", "
info" )
```

Listing 7: [symbian-cal.py](#)

Zu Beginn des Skripts werden die beiden Module **e32calendar** für den Zugriff auf den Kalender und **appuiw** für Benutzeroberflächen geladen. Einmal teilt ein Hinweisenfenster im Skript mit, dass es keinen Kalendereintrag gefunden hat, und am Ende, dass es erfolgreich fertig geworden ist. Die Funktion **open()** aus dem Modul **e32calendar** öffnet den Kalender und übergibt ihn an **cal**. Die einzelnen Einträge werden nacheinander einer Liste übergeben, die dann komplett in die Datei **vCal.vcf** geschrieben wird.

Literatur

- Jürgen Scheible und Ville Tuulos, „Mobile Python: Rapid prototyping of applications on the mobile platform“, 2007
- Jürgen Scheible, „Python for S60 Tutorial“ [11]

LINKS

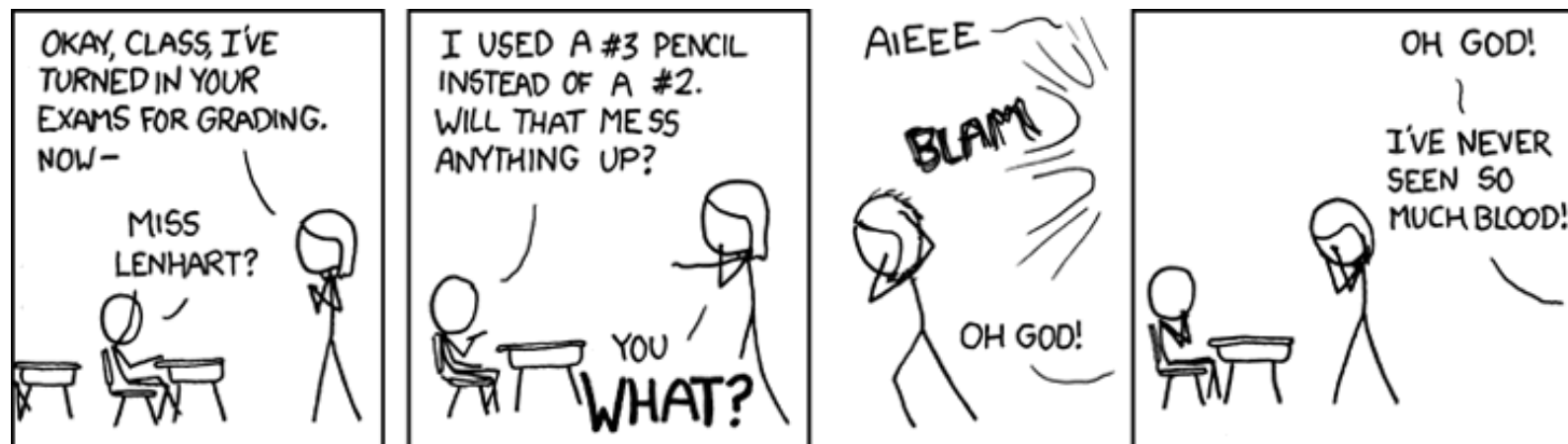
- [1] <http://www.heise.de/newsticker/meldung/Symbian-wird-komplett-Open-Source-921815.html>
- [2] <http://de.wikipedia.org/wiki/S60>
- [3] <https://garage.maemo.org/>
- [4] <http://www.datenteiler.de/ein-kleiner-webserver-mit-python/>
- [5] <http://code.google.com/p/ped-s60/>
- [6] <http://www.datenteiler.de/mobiles-python-ii/>
- [7] <http://www.datenteiler.de/mit-gnulinix-fuer-pys60-entwickeln/>
- [8] <http://pys60.garage.maemo.org/doc/s60/>

- [9] <http://code.google.com/p/ensymbler/>
- [10] <http://www.datenteiler.de/wie-man-python-2-5-mit-zlib-kompiliert/>
- [11] <http://www.mobilenin.com/>

Autoreninformation

Christian Imhorst hat wegen Appstore-Zwang, iTunes und der ständigen Gängelung durch Apple sein iPhone gegen ein Nokia E71 eingetauscht. Ein Android-Gerät kommt vielleicht später einmal, schließlich gibt es ja SL4A (Scripting Layer for Android).

[Diesen Artikel kommentieren](#)



„Scantron“ © by Randall Munroe (CC-BY-NC-2.5), <http://xkcd.com/499>

MP3s von Amazon herunterladen von Dominik Wagenführ

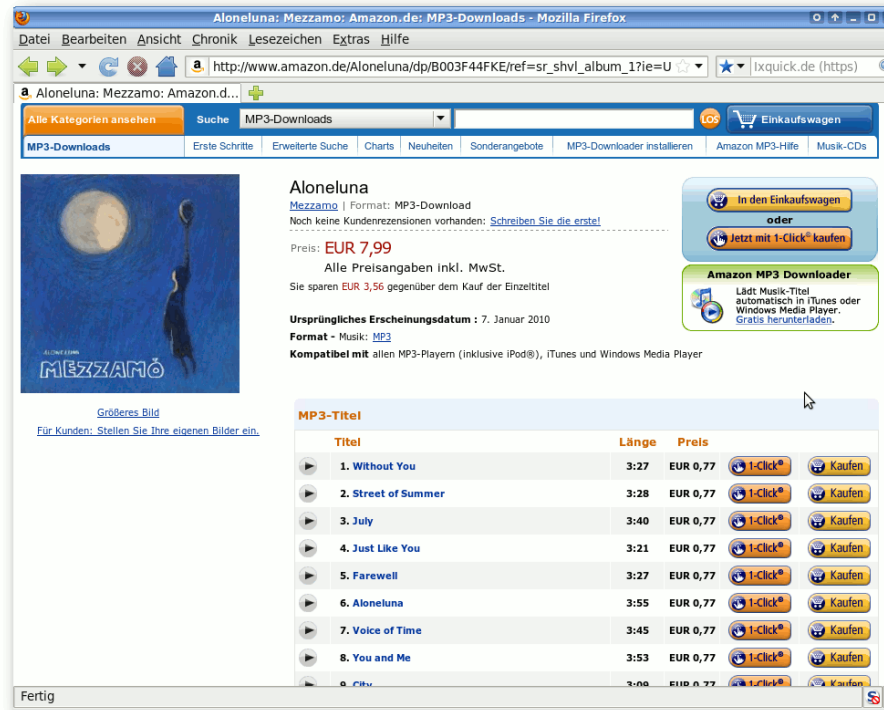
Musik wird heutzutage in vielen Fällen nur noch ohne physische Datenträger gehört. Sei es per MP3 oder OGG, in der Regel spielt die gekaufte Musik auf einem mobilen Musikabspielgerät oder direkt auf dem PC. Umso wichtiger ist es daher, dass man die Musik gleich im passenden Format kaufen kann und nicht erst die gekaufte CD in ein benutzbares Format wandeln muss. Einer der großen Verkäufer von Musik zum Herunterladen ist Amazon [1].

Bei Amazon gibt es fast alle neueren und auch die meisten älteren Musikstücke im MP3-Format. Dieses ist zwar nicht so verlustfrei wie OGG bei der Komprimierung, hat sich aber dennoch im letzten Jahrzehnt als mobiles Musikformat durchgesetzt.

Amazon bietet Einzeltitel direkt zum Kauf an, indem man sie einfach in den Einkaufswagen legt. Ganze MP3-Alben kann man auf diese Art aber nicht kaufen, das erfordert (normalerweise) den speziellen Amazon MP3-Downloader.

Installation

Auf der Amazon-Webseite gibt es bei jedem MP3-Album bzw. -Titel rechts oben unter dem Einkaufswagen auch den Link zum Amazon MP3-Downloader. Auch wenn in der Kurzbeschreibung nur von iTunes und dem Windows Media Player gesprochen wird, gibt es die Software auch für Linux.



Der Downloadknopf funktioniert auch mit Linux. 

Auf der Amazon-Downloadseite [2] findet man Pakete für Einige der am weitest verbreiteten Distributionen: Ubuntu 8.10, Debian 5, Fedora 9 und openSUSE 11.0. Dies sind zwar alle etwas ältere Versionen, aber selbst wenn man eine neuere Version dieser Distributionen einsetzt, sollte der Einsatz kein Problem sein. Eine Ausnahme ist das neueste Ubuntu 10.04 „Lucid Lynx“, da hier einige benötigte Bibliotheken nicht mehr vorhanden sind und erst manuell nachinstalliert werden müssen [3].

Nach dem Download des jeweiligen Pakets installiert man das DEB- bzw. RPM-Paket manuell wie es die eigene Distribution erfordert. Gegebenenfalls kommt es zu Fehlern, weil einige Abhängigkeiten nicht installiert sind. Das Amazon-Paket erfordert folgende zusätzliche Bibliotheken, die bei den meisten Distributionen nicht als Standard installiert sind:

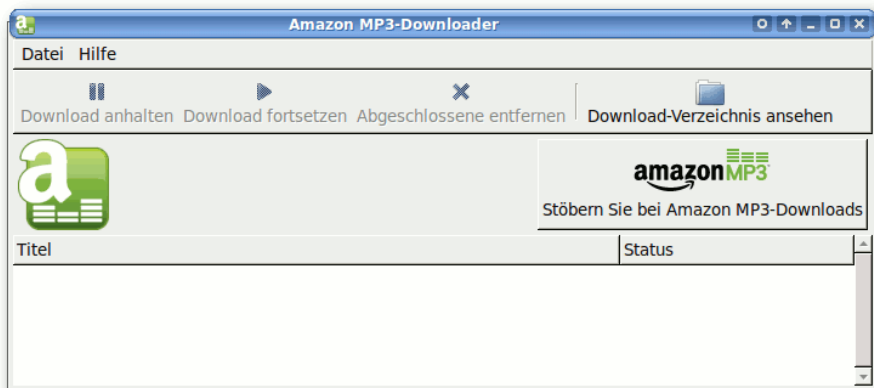
- libboost-date-time1.34.1
- libboost-filesystem1.34.1
- libboost-iostreams1.34.1
- libboost-regex1.34.1
- libboost-signals1.34.1
- libboost-thread1.34.1

Installation in 64-Bit-Systemen

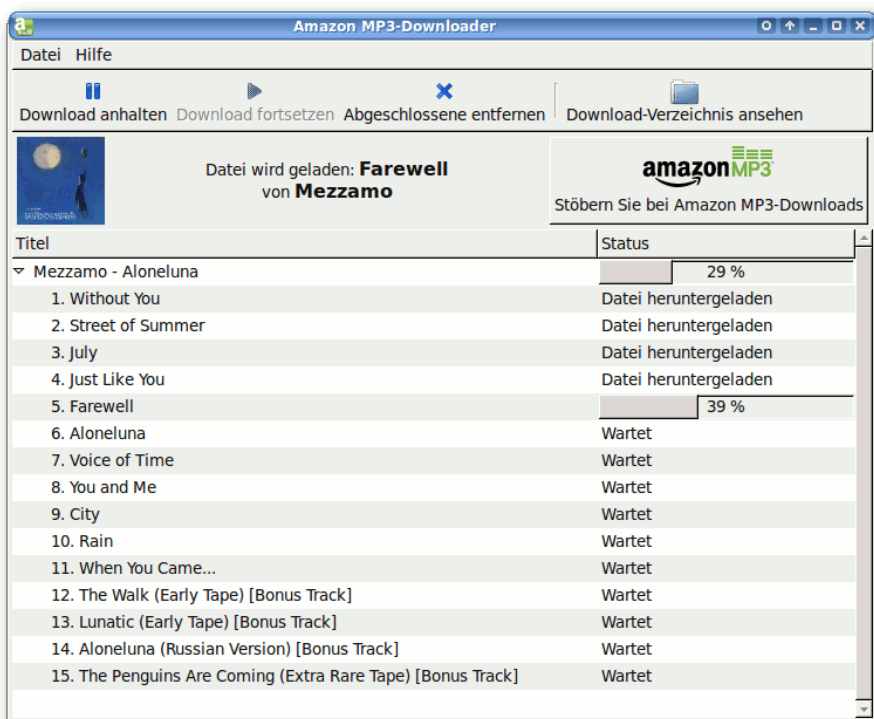
Leider gibt es von Amazon keine Version für 64-Bit-Systeme. Hier kann man aber etwas nachhelfen. Unter Ubuntu kann man beispielsweise mittels

```
# dpkg -i --force-architecture amazonmp3.deb
```

die Paketverwaltung dennoch anweisen, das Paket zu installieren.



Die Oberfläche wirkt sehr aufgeräumt. 🔍



Die Musikdateien werden automatisch heruntergeladen. 🔍

Mit Hilfe des Programms **getlibs** [4] muss man aber noch bestimmte Bibliotheken nachinstallieren lassen:

```
# getlibs ~
/usr/bin/amazonmp3
```

Benutzung

Durch den Aufruf von **amazonmp3** wird der Amazon-MP3-Downloader gestartet. Die Oberfläche des Programms ist sehr übersichtlich. Beim ersten Start landet man auf der Amazon-Webseite, denn nur dort kann man die Musik herunterladen. Bei den folgenden Starts des Programms klickt man auf den Amazon-MP3-Knopf, der mit „Stöbern Sie bei Amazon Mp3-Downloads“ beschriftet ist.

Im Amazon-Shop legt man sich das MP3-Album bzw. die einzelnen MP3-Titel in den Einkaufswagen und bezahlt dies ganz regulär. Nach der Bezahlung wird man aufgefordert, eine AMZ-Datei zu speichern bzw. diese gleich mit dem Amazon MP3-Downloader zu öffnen.

Hat man die Datei auf der Festplatte gespeichert, geht man im Amazon MP3-Downloader auf „Datei → AMZ-Datei öffnen“ und wählt die eben gespeicherte AMZ-Datei aus. Daraufhin wird auch gleich der Download der gekauften Dateien gestartet. Die Daten werden automatisch im Ordner **Amazon MP3** im Homeverzeichnis gespeichert. Das Verzeichnis kann man aber über „Datei → Einstellungen ...“ anpassen.

Nach dem Download kann man das Download-Verzeichnis öffnen (lassen) und die Musik normal abspielen.



Die AMZ-Datei sollte man sich auf der Festplatte sichern. 🔍

Hinweis: Wenn man die AMZ-Datei mit dem Amazon MP3-Download öffnet, wird diese von der Festplatte entfernt. Es empfiehlt sich also die Datei zu sichern, möchte man die Musik später ein zweites Mal herunterladen. Eine zweite Zuweisung der Datei seitens Amazon ist nicht mög-

lich. (Es ist aber nicht klar, wie lange die Datei gültig ist, sprich wie lange man nach dem Download der AMZ-Datei die Lieder herunterladen kann.)

Sollte es zu Problemen kommen, schadet ein Blick in die Amazon-FAQ [5] sicherlich nicht.

Alternativen

Clamz

Mittels Clamz [6] können die AMZ-Dateien ebenfalls geöffnet werden. Leider gibt es kein fertiges Paket auf der Webseite, sodass man sich die Software selbst kompilieren muss.

Dafür benötigt man neben einem C++-Compiler auch die Entwicklerpakete von **libcrypt**, **libcurl** und **libexpat**. Danach kann man das heruntergeladene TAR-Archiv entpacken und kompilieren:

```
$ tar xfvz clamz-0.4.tar.gz
$ cd clamz-0.4
$ ./configure
$ make
```

Dann kann man die Clamz per

```
# make install
```

bzw.

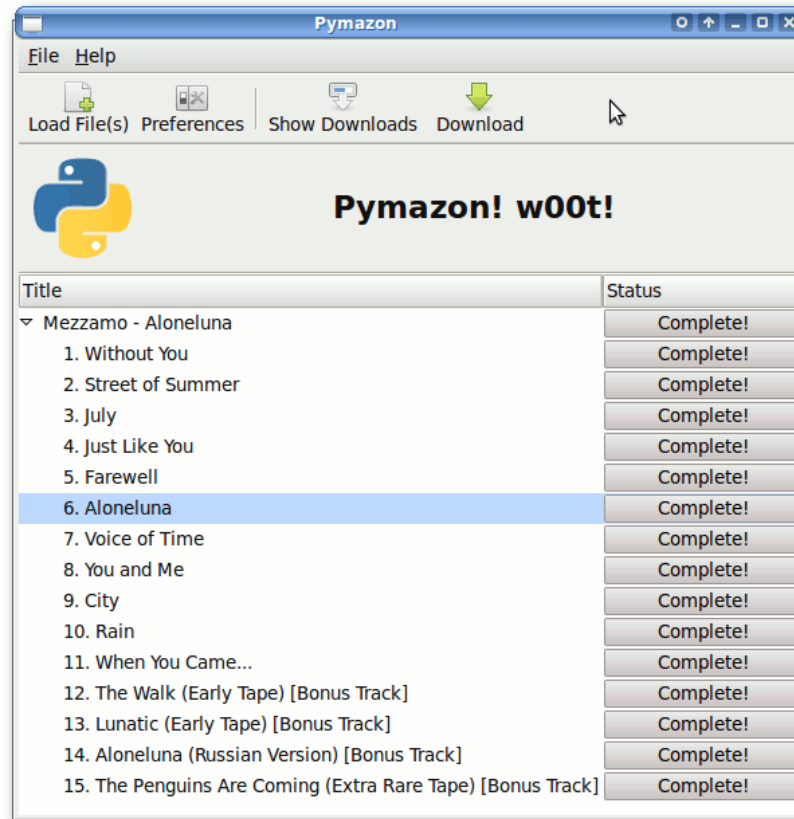
```
# checkinstall
```

installieren.

Clamz arbeitet auf der Kommandozeile und bekommt nur die AMZ-Datei als Argument übergeben:

```
$ clamz AmazonMP3-1281195544.amz
```

Per Standard werden die Lieder im aktuellen Verzeichnis gespeichert. In der Datei **.clamz/config** im Homeverzeichnis kann man dies aber ändern.



Auch Pymazon lädt die MP3-Titel problemlos herunter. 🔍

Pymazon

Pymazon [7] basiert auf Clamz (benötigt dies aber nicht) und ist, wie der Name andeutet, in

Python geschrieben. Das Programm bietet eine grafische Oberfläche ähnlich dem Amazon MP3-Downloader.

Für die Benutzung benötigt man die Programme PyCrypto und PyQt4 bzw. PyGtk. Nach dem Download des TAR-Archives kann man dieses entpacken und die Installation starten:

```
$ tar xzf Pymazon-0.9.tar.gz
$ cd Pymazon-0.9
# python setup.py install
```

Auch hier ist wieder die Benutzung von **checkinstall** empfehlenswert, damit der Paketmanager die Verwaltung der installierten Dateien übernehmen kann:

```
# checkinstall
python setup.py install
```

In der Datei **.pymazon/pymazonrc** im Homeverzeichnis kann man verschiedene Einstellungen vornehmen. Auf Systemen mit GNOME ist es wichtig, dass man

```
toolkit = gtk
```

einstellt. Dabei kann man auch gleich den Ort einstellen, an dem die Musiktitel gespeichert werden sollen. Dies kann man aber auch später in der Oberfläche unter „File → Preferences“ ändern.

Pymazon wird einfach über den Aufruf von **pymazon** gestartet und bedient sich recht intuitiv. Mittels „Load File(s)“ wählt man die AMZ-Datei aus und klickt danach auf „Download“.

Fazit

Mittels der verschiedenen Downloadclients wird das Herunterladen ganzer MP3-Alben von Amazon zum Kinderspiel. Einzig die Installation der Programme erfordert (auf neueren und/oder auf 64-Bit-Systemen) noch zu viel Handarbeit. Ein fertiges Paket, welches man über die Paketverwaltung installieren kann, wäre wünschenswert.

Am leichtesten war aber Pymazon zu bedienen und zu installieren. Da es sich um Freie Software handelt (Pymazon wird unter der GNU General Public License Version 3 vertrieben), ist diese dem proprietären Amazon Mp3-Downloader vorzuziehen, zumal die Anwendung auch auf neue-

ren und 64-Bit-Systemen ohne Probleme funktioniert.

Hinweis: Das für diesen Artikel gekaufte Album „Aloneluna“ von Mezzamo ist sehr empfehlenswert und kann auf der Homepage der Band [8] oder bei Jamendo [9] auch kostenlos heruntergeladen werden. Die Musik unterliegt dabei einer Creative-Commons-Lizenz [10].

LINKS

- [1] <http://www.amazon.de/>
- [2] http://www.amazon.de/gp/dmusic/help/amd.html/ref=dm_dp_amd
- [3] http://wiki.ubuntuusers.de/Amazon_MP3-Downloader
- [4] <http://ubuntuforums.org/showthread.php?t=474790> 
- [5] <http://code.google.com/p/clamz/> 
- [6] <http://code.google.com/p/pymazon/> 

- [7] <http://mezzamo.blogspot.com/p/downloads.html> 
- [8] <http://www.jamendo.com/de/album/58888>
- [9] <http://creativecommons.org/licenses/by-nc-nd/3.0/> 

Autoreninformation

Dominik Wagenführ liebt Musik – vor allem wenn sie frei ist. Wenn eine Band keinen Spendenknopf auf ihrer Webseite hat, ist der Kauf über Amazon eine gute Alternative, der Band etwas Unterstützung zukommen zu lassen.

Diesen Artikel kommentieren 



„Kindle“ © by Randall Munroe (CC-BY-NC-2.5), <http://xkcd.com/548>



GIMP automatisieren von Ralf Damaschke

Wie einfach es ist, mit GIMP Bilder zu verbessern oder auch zu verschönern, wurde im Artikel „GIMP in 90 Minuten (kennenlernen)“ von Karsten Günther in freiesMagazin 01/2010 [1] gezeigt. Verwendet man GIMP dann regelmäßig, kommt schnell der Wunsch auf, immer wiederkehrende Handgriffe, wie z. B. das Erstellen eines Rahmens oder das Einfügen eines Copyright-Vermerks, zu automatisieren. Auch hier hält GIMP das passende Werkzeug bereit: Skript-Fu [2].

Wer die Aktionen von Photoshop kennt, wird beim ersten Kontakt mit Skript-Fu mächtig enttäuscht sein. Man kann weder ein Makro aufzeichnen noch gibt es einen integrierten Editor, der einem bei den ersten Schritten ein wenig unter die Arme greift und z. B. Syntaxfehler gleich hervorhebt. Selbst das Anlegen eines neuen Skripts muss händisch im Skriptordner erledigt werden. Um die Leidenschaft des Nutzers wirklich zu testen, ist Skript-Fu zudem eine Scheme-basierte Sprache, aber dazu später mehr.

Es soll noch darauf hingewiesen werden, dass es für alle, die sich mit Skript-Fu nicht anfreunden können, als Alternative Python-Fu gibt. Im Rahmen dieses Artikels wird aber nicht weiter auf Python-Fu eingegangen; alle Interessierten seien für den Einstieg an gimp.org [3] verwiesen. Trotz aller Widrigkeiten lohnt es aber, sich mit

Skript-Fu zu befassen. Nach einer gewissen Eingewöhnungszeit lassen sich in kurzer Zeit Skripte erstellen, die den Funktionsumfang von GIMP auf die eigenen Bedürfnisse anpassen.

Skript-Fu

Die Syntax von Skript-Fu ist wie bereits erwähnt doch recht gewöhnungsbedürftig und kann in den GIMP-Tutorials [4] eingehend von Grund auf erlernt werden. In diesem Artikel wird nur das Nötigste besprochen, um einen schnellen Einstieg in das Beispielskript zu schaffen. Unter „Filter → Skript-Fu → Konsole“ in GIMP kann die Skript-Fu-Konsole aufgerufen werden, um die Beispiele

unten nachzuvollziehen. Die Eingabe muss dabei einzeilig sein; sie wird mit `Enter` abgeschlossen.

Syntax

Scheme [5] und damit auch Skript-Fu verwendet die Präfixnotation [6], d. h. der Operator wird den Operanden vorangestellt. Aus einer üblichen Addition

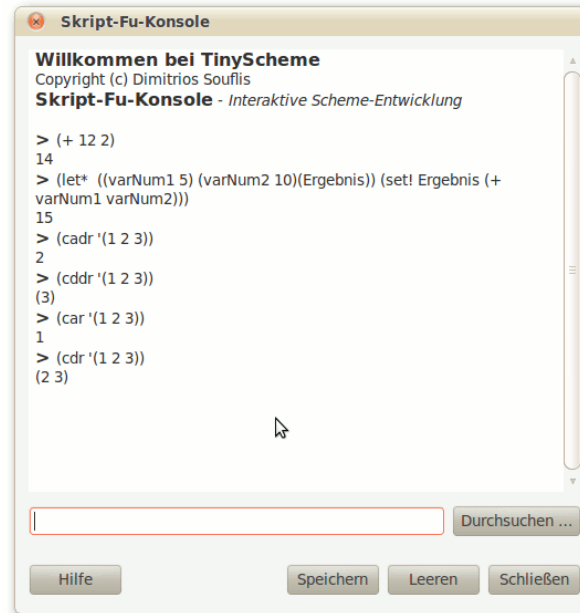
```
12 + 2
```

wird im Skript-Fu

```
(+ 12 2)
```

Wie im obigen Beispiel zu erkennen ist, wird die Operation von runden Klammern umschlossen. Dies gilt für alles in Skript-Fu: Ob If-Anweisung, Variablendeklaration oder Funktionen – alles wird von runden Klammern umschlossen. So ist es keine Seltenheit, dass am Ende von Anweisungen drei oder mehr schließende Klammern hintereinander stehen. Wer hier einen Editor ohne Klammernpaar-Hervorhebung verwendet, sieht bald den Wald vor lauter Klammern nicht mehr.

Unter GNOME beherrscht gedit die Syntaxhervorhebung für Scheme standardmäßig und unterstützt einen auch beim Finden der Klammernpaare. Dies muss allerdings erst unter „Bearbeiten → Einstellungen → Ansicht → Übereinstimmende Klammern hervorheben“ manuell aktiviert werden. Unter KDE ist Kate sehr zu empfehlen,



Skript-Fu-Konsole mit den Beispielen. 🔍



da dort die Klammerpaare für jede Ebene einzeln farbig hervorgehoben werden.

Kommentare werden wie in vielen Skriptsprachen mit einem Semikolon eingeleitet.

Variablen

Hauptsächlich werden Variablen in Skript-Fu mit **let*** lokal deklariert und können bei der Deklaration auch mit einem Defaultwert initialisiert werden. Das Zuweisen eines neuen Wertes im Skript erfolgt mit **set!**:

```
(let* ( Variablen ) Ausdrücke in denen
die Variablen gültig sind )
```

Das sieht dann im Skript so aus:

```
(let* ( (varNum1 5)
      (varNum2 10)
      (Ergebnis)
      )
      (set! Ergebnis (+ varNum1 varNum2))
      )
```

se zur Darstellung von Farben als RGB-Werte genutzt werden. Definiert wird eine solche Liste durch einen vorangestellten Apostroph. Die Werte in der Klammer müssen dabei nicht vom selben Typ sein.

```
'(0 0 0) ;definiert die Farbe Schwarz
'("Erster Eintrag" 2 3) ;ist auch eine
mögliche Liste
```

Möchte man die einzelnen Werte auslesen, so kann man erst einmal nur auf den Header (den ersten Wert) oder den Tail (alle restlichen Werte) zugreifen. Besteht dieser Rest aus mehreren Werten, so werden diese wieder in Kopf und Schwanz aufgeteilt. Möchte man also auf den letzten Wert einer Liste zugreifen, muss man sie solange in Kopf und Schwanz zerlegen bis der gewünschte Wert übrig bleibt. Auf den Kopf greift man mit **car** zu, den Rest bekommt man mit dem Befehl **cdr**

```
(car '(1 2 3))
(cdr '(1 2 3))
```

Die erste Zeile im Beispiel liefert die Zahl 1 zurück, während die zweite Zeile eine Liste mit den Werten 2 und 3 ergibt. Möchte man also auf den zweiten Wert der ursprünglichen Liste zugreifen sieht der Code wie folgt aus:

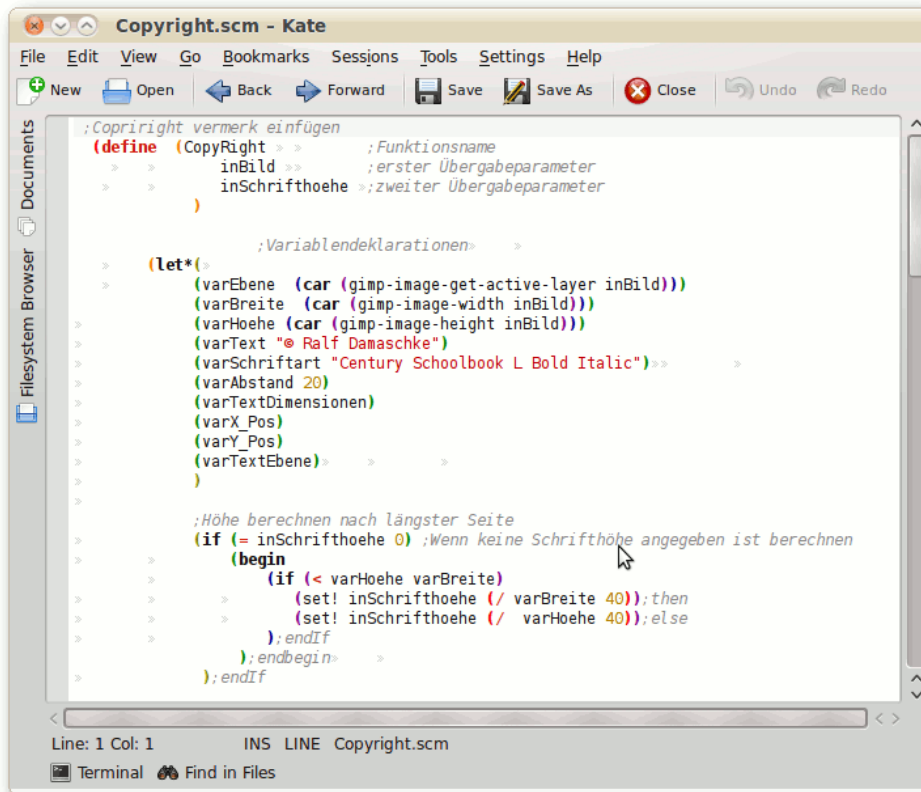
```
(car (cdr '(1 2 3)))
```

Den dritten Wert erreicht man mit:

```
(cdr (cdr '(1 2 3)))
```

Wahrscheinlich ist es dem einen oder anderem schon aufgefallen: Scheme ist eine typenfreie Sprache, bei der der Typ einer Variablen nicht extra definiert werden muss, sondern sich durch ihre Verwendung ergibt. Natürlich gibt es die üblichen Verdächtigen mit String, Integer usw., aber auch spezielle wie Image, Drawable oder Layer. Die drei zuletzt aufgezählten Typen übergeben jeweils eine Referenz auf ein Bild oder eine Ebene (Drawable ist die aktive Ebene).

Besonders zu erwähnen ist der Umgang mit Listen (Arrays), die beispielsweise



Klammernpaar-Hervorhebung bei Kate. 🔍



Um bei längeren Listen nicht völlig den Überblick zu verlieren kann man die Befehle zusammenfassen. So wird aus den beiden Beispielen von oben:

```
(cadr '(1 2 3))
(cddr '(1 2 3))
```

Wie man sieht wurde aus den zwei Befehlen einer, wobei ein **a** für ein **car** steht und ein **d** für ein **cdr**.

Funktionen

Funktionen werden wie folgt definiert.

```
(define
  (
    Funktionsname
    Parameter1
    Parameter2
  )
  Ausdruecke
)
```

Dabei gilt für die Übergabeparameter natürlich dasselbe wie für die Variablen auch: Ihr Typ muss nicht extra deklariert werden. Auch muss für eine Funktion kein Rückgabewert definiert werden, es wird einfach der Wert der letzten Anweisung zurückgegeben. Beim Aufruf der Funktion werden die Übergabewerte durch Leerzeichen getrennt:

```
(Funktionsname 1 "String")
```

Abfragen

Eine If-Abfrage wird in Scheme wie folgt geschrieben:

```
(if (bedingung)
    (then-block)
    (else-block)
)
```

Der else-Block ist dabei optional und kann, wenn er nicht benötigt wird, einfach weggelassen werden, was dann in einem Skript wie folgt aussehen kann:

```
(if (= (gimp-image-base-type bild) 2)
    (gimp-image-convert-rgb bild)
)
```

Der obige Codeblock wandelt das in der Variable **bild** gespeicherte Bild in RGB-Farben, wenn es zuvor indizierte Farben verwendet hat (GIMP-Base-Type = 2). Hat man mehr als nur eine Anweisung zu verarbeiten, so werden sie mit einem **begin** umschlossen:

```
(begin Anweisung1
      Anweisung2)
```

Skripte erstellen

Nachdem die Syntax nun bekannt ist, wird es Zeit, ein Beispielskript zu erstellen. Als Beispiele sollen drei kleinere Skripte zur automatischen Einhaltung von Beschränkungen beim Bildupload auf Online-Plattformen dienen. Zuerst wird das Größenlimit durch Skalieren des Bildes eingehalten, dann kann noch ein optionaler Rahmen zur Kontrasterhöhung gezogen werden und natürlich darf ein eingebetteter Copyright-Hinweis nicht fehlen.

Bevor man sich um den eigentlichen Code kümmert, braucht man einen Container, in den man die gewünschte Funktionen einfügen kann. Dazu erstellt man eine leere Textdatei im Ordner **/home/BENUTZERNAME/.gimp-2.6/scripts** mit der Endung **.scm**. Diese Endung ist wichtig, da GIMP nur Dateien mit dieser Endung nach Code durchsucht. Der Dateiname spielt dabei für GIMP keine Rolle, er dient lediglich als Unterscheidung zwischen den Dateien für das Dateisystem und natürlich zum einfacheren Wiederfinden der Skripte durch den Programmierer.

In der leeren Datei kann jetzt die neue Funktion erstellt werden:

```
;Skaliert das Bild auf max. Groesse
(define (BildSkalieren
  inBild
  inMax
  ;Variablendeklarationen
  (let* (
    (varBreite (car (gimp-~
  image-width inBild)))
    (varHoehe (car (gimp-~
  image-height inBild)))
  )
  ;hier wird das Bild ~
  verkleinert
  );endLet
);endDefine
```

Der Funktion **BildSkalieren** müssen nach dieser Definition zwei Parameter übergeben werden: der Zeiger auf das Bild und die maximal zulässige Größe. Danach werden die Variablen

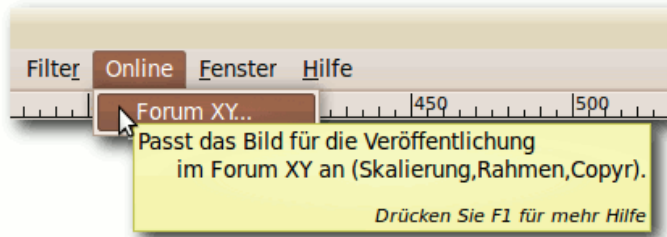


varBreite und **varHoehe** definiert und gleich aus den Dimensionen des übergebenen Bildes berechnet. Nach den Variablen steht ein Kommentar als Platzhalter für den eigentlichen Code.

Um die Funktion aus GIMP heraus verwenden zu können, muss sie noch mit **script-fu-register** in den GIMP-Menüs registriert werden:

```
(script-fu-register
  "BildSkalieren"           ;Aufzurufende Funktion
  "<Image>/Online/Forum XY..." ;Name und Position in GIMP
  "Passt das Bild fuer die Veroeffentlichung \
  im Forum XY an (Skalierung,Rahmen,Copyr).\"
  "Ralf Damaschke"
  "Copyright (C) 2010 Ralf Damaschke under \
  terms of GNU General Public License Version3\"
  "Jul 14, 2010"
  ""
  SF-IMAGE "Input image" 0
  SF-ADJUSTMENT "maximale Groesse" '(900 1.0 3000 1.0 0 0 0)
)
```

script-fu-register wird zuerst der Name der Funktion übergeben, die beim Aufrufen des Menüeintrags ausgeführt werden soll. Im zweiten Parameter wird die Position und der Name des Menüeintrages festgelegt. Man ist dabei frei, sich in die vorhanden Menüs einzufügen oder auch einen eigenen Eintrag zu schaffen. Einstiegspunkt ist dabei immer **<Image>**. Bis GIMP-Version 2.6 konnte auch noch **<Toolbox>** verwendet werden, diese Menüs wurden jedoch entfernt. Möchte man seinen Eintrag zwischen den GIMP-Menüs platzieren, so ist zu beachten, dass



Das neu erstellte Menü mit Tooltip. 🔍

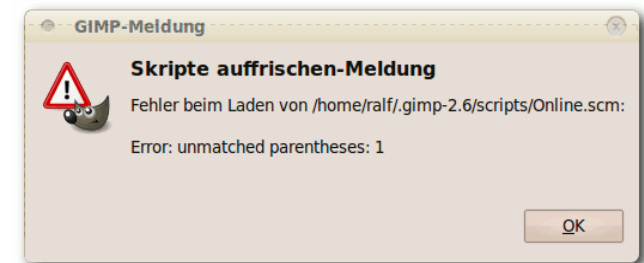
die englischen Menütexte verwendet werden. So würde ein **<Image>/Filters/Online/Forum XY...** einen neuen Ordner „Online“ unter „Filter“ mit dem Eintrag „Forum XY...“ schaffen. In diesem Beispiel wird jedoch ein eigenes Menü „Online“ erzeugt. Neu angelegte Menüs werden dabei alphabetisch einsortiert.

Der Parameter danach ist ein Hilfetext, der auch als Tooltip im Menü angezeigt wird, gefolgt von Autor, Copyright und Erstellungsdatum. Danach kann die Verwendung der Funktion auf bestimm-

te Bildtypen eingeschränkt werden (z. B. RGB). Ist dieser Parameter leer, kann das Skript auf alle Bildtypen angewendet werden.

Nach diesen festen Parametern kann man noch weitere nach Bedarf frei definieren. Fast immer werden mit den Parametern **SF-IMAGE** und **SF-DRAWABLE** Zeiger auf das aktuelle Bild und die aktive Ebene an die Funktion übergeben. Man kann aber auch interaktiv Eingaben vom Benutzer abfragen, in diesem Beispiel mit **SF-ADJUSTMENT** die maximal zulässige Größe des Bildes (in Pixel). Alle möglichen Parameter und ihre Verwendung sind im GIMP-Benutzerhandbuch im Abschnitt „3.4.7. Die Skriptparameter registrieren“ aufgeführt [7].

Nach dem Speichern der Datei müssen die Informationen in GIMP aktualisiert werden, entweder durch einen Neustart von GIMP oder durch „Filter → Skript-Fu → Skripte auffrischen“. Sollte in einer der Funktionen ein Syntaxfehler sein, deaktiviert GIMP das Skript – beim Starten ohne sichtbaren Fehler, beim Ausführen von „Skripte auffrischen“ mit einer Fehlermeldung. Leider sind diese Fehlermeldungen nicht sehr hilfreich. Vergisst



Fehlermeldung bei einer fehlenden Klammer. 🔍



man z. B. eine Klammer, erscheint lediglich die Fehlermeldung „*Error: unmatched parentheses: 1*“. Ein Hinweis auf die fehlerhafte Zeile fehlt, die Zahl hinter dem Fehler ist eine Fehlernummer.

Beispiele

Bilder skalieren

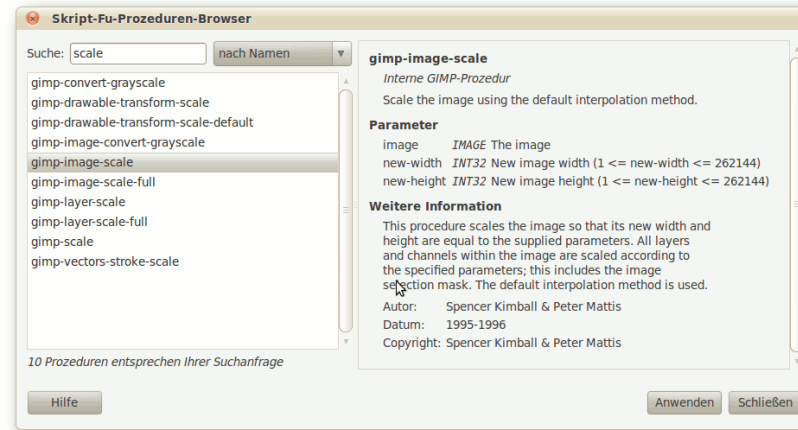
Nachdem jetzt das Gerüst steht, wird es Zeit, das Skript mit Leben zu füllen. Zuerst wird die richtige GIMP-Funktion für das Skalieren eines Bildes benötigt. Am einfachsten lässt sich diese mit dem Skript-Fu-Prozeduren-Browser finden. Er wird über den „*Durchsuchen...*“-Knopf in der Skript-Fu-Konsole aufgerufen und listet alle verfügbaren Befehle (auch die eigenen Skripte) mit Hilfetexten auf. Leider sind alle Prozedurnamen wie auch die Hilfetexte ausschließlich in Englisch. So sucht man am besten nach „*scale*“ und bekommt eine Auswahl von zehn Prozeduren angezeigt.

Ist die richtige Prozedur gefunden, kann man sie, mitsamt den benötigten Parametern, durch Drücken des „*Anwenden*“-Knopfes in die Eingabezeile der Skript-Fu-Konsole übertragen. Von hier wird sie via Copy & Paste in das eigene Skript kopiert.

In diesem Beispiel also die Prozedur **gimp-image-scale**:

```
(gimp-image-scale image new-width ↻
new-height)
```

Die Prozedur benötigt drei Parameter: das Bild auf welches sie angewendet werden soll, die



Der Skript-Fu-Prozedur-Browser. 🔍

neue Breite und die neue Höhe. Der erste Parameter stellt kein Problem dar, er ist der erste Übergabeparameter **inBild**, von den neuen Dimensionen des Bildes ist jedoch nur die längste Kante bekannt. Um das Bild proportional zu skalieren, muss die zweite Kante im Verhältnis dazu berechnet werden. Dazu werden noch die Variablen **varNeueBreite** und **varNeueHoehe** eingeführt, die wie folgt gebildet werden:

```
;Neue Breite und Hoehe nach laengster Kante berechnen
(if (< varHoehe varBreite)
  (begin ;Querformat-Block
    (set! varNeueBreite inMax)
    (set! varNeueHoehe (/ varHoehe(/ varBreite varNeueBreite)))
  );then
  (begin ;Hochformat-Block
    (set! varNeueHoehe inMax)
    (set! varNeueBreite (/ varBreite (/ varHoehe varNeueHoehe)))
  );else
);endIf
```

Je nachdem, ob das Bild ein Quer- oder Hochformat ist, wird die eine der beiden neuen Variablen mit dem Maximalwert belegt, während die andere Variable aus ihrer alten Länge, geteilt durch den Skalierungsfaktor, berechnet wird. Jetzt muss nur noch die Prozedur mit den gewünschten Werten gefüttert werden.

```
;Groesse aendern
(gimp-image-scale inBild
  varNeueBreite
  varNeueHoehe )
```

Daraus ergibt sich dann das fertige Skript [gimp-onlinegroesse.scm](#).

Copyrightvermerk

Wie schon am Anfang des letzten Kapitels erwähnt, soll bei Bedarf auch ein Copyrightvermerk in das Bild eingefügt werden. Bevor man sich jedoch auf den Code stürzt, sollte man überlegen, wie sich das Gewünschte unter GIMP realisieren lässt. Für einen Copyrightvermerk würde man



1. mit dem Textwerkzeug in einer neuen Textebene eine Textbox erstellen,
2. den Text eingeben und ggf. die Schriftart anpassen,
3. die Textbox an der richtigen Stelle platzieren und
4. die Deckkraft der Textebene anpassen, damit der Schriftzug nicht so heraussteicht.

All diese Schritte werden auch in Skript-Fu durchgeführt. Zusätzlich müssen noch die Kleinigkeiten, die im Normalfall nebenher laufen, bedacht und festgelegt werden, wie z. B. die Auswahl der Farbe. Arbeitet man in GIMP, so hat man meist schon die richtige Farbe in einem vorherigen Arbeitsschritt ausgewählt. Programmiert man den selben Arbeitsschritt in Skript-Fu, so weiß man nicht, was der Anwender zuvor gemacht hat und mit welchen Farben oder Schriftarten er gearbeitet hat.

Ein großer Vorteil von Skripten gegenüber dem normalen Arbeiten in GIMP ist die Möglichkeit, die Platzierung oder die Größe von Objekten zu berechnen. So kann z. B. der Copyrightvermerk immer mit einem definierten Abstand vom Bildrand platziert werden. Dies soll im folgenden Skript genutzt werden, um den Copyright-Schriftzug in die rechte untere Ecke, mit einem Abstand, der das Anderthalbfache der Schriftgröße ist, zu platzieren.

Zuerst muss natürlich eine Funktion deklariert und in den GIMP-Menüs registriert werden.

```

;Copyright einfuegen
(define (CopyRight      ;Funktionsname
      inBild            ;erster Uebergabeparameter
      inSchriftHoehe   ;zweiter Uebergabeparameter
      )

;Platzhalter fuer den Code

);enddefine

;#=====
;# Im GIMP Menue registrieren
(script-fu-register "CopyRight"
  "<Image>/Online/Copyright einfuegen..."
  "Fuegt ein Copyrightvermerk ein"
  "Ralf Damaschke"
  "Copyright (C) 2010 Ralf Damaschke under \
terms of GNU General Public License Version3"
  "Jul 14, 2010"
  ""
  SF-IMAGE "Input image" 0
  SF-ADJUSTMENT "Schrifthoehe" '(10 1.0 100 1.0 0 0 1)
)

```

Die Funktion **CopyRight** wird der Übersichtlichkeit halber mit nur zwei Parametern aufgerufen: zum einen das aktuelle Bild und zum anderen die Schriftgröße. Natürlich könnte man auch noch weitere Parameter übergeben, die sich vom Benutzer verändern lassen. Beispielsweise könnte man den Copyright-Text mit

```
SF-STRING "Copyrightvermerk" "(C) ↻
Ralf Damaschke"
```

oder auch eine ausgewählte Schriftart mit **SF-FONT** übergeben. Im Beispiel von **SF-STRING** oben ist der erste Parameter die Beschriftung vor dem Eingabefeld und der zweite Parameter der Standardwert, mit dem das Textfeld am Anfang vorbelegt wird.

Danach werden die Variablen deklariert und dort, wo es notwendig ist, gleich mit sinnvollen Werten belegt:



```
;Variablendeklarationen
(let*(
  (varEbene (car (gimp-image-get-
    active-layer inBild)))
  (varBreite (car (gimp-image-
    width inBild)))
  (varHoehe (car (gimp-image-
    height inBild)))
  (varText "(C) Ralf Damaschke")
  (varSchriftart "Century
    Schoolbook L Bold Italic")
  (varAbstand 20)
  (varTextDimensionen)
  (varX_Pos)
  (varY_Pos)
  (varTextEbene)
)
```

Sollte die übergebene Schriftgröße gleich Null sein, wird eine neue berechnet, die dann ein Viertel der längsten Kante betragen soll.

```
;Hoehe berech. nach laengster Seite
(if (= inSchrifthoehe 0) ;Wenn
keine Schrifthoehe angegeben ist...
  (begin
    (if (< varHoehe varBreite)
      (set! inSchrifthoehe (/
        varBreite 40));then
      (set! inSchrifthoehe (/
        varHoehe 40));else
    );endIf
  );endbegin
);endIf
```

Jetzt wird noch der richtige Abstand benötigt:

```
;Abstand = anderthalbfache Schriftgroesse
(set! varAbstand (* inSchrifthoehe 1.5))

;Schrifthoehe und Laenge in eine Liste
(set! varTextDimensionen (gimp-text-get-extents-fontname
  varText
  inSchrifthoehe
  PIXELS
  varSchriftart)
);endset

;Die Schriftpositionen berechnen
(set! varX_Pos (- varBreite (+ (car varTextDimensionen) varAbstand)))
(set! varY_Pos (- varHoehe (+ (caddr varTextDimensionen) varAbstand)))
```

Der Copyrightvermerk soll mit einem gleichmäßigen Abstand, das Anderthalbfache der Schrifthöhe, zur unteren rechten Ecke platziert werden. Um dies mit der Prozedur **gimp-text-fontname** zu bewerkstelligen, ist es nötig, die Dimensionen des Schriftzuges zu kennen, da sie die rechte obere Ecke des Schriftzuges platziert. Dies erledigt die Funktion **gimp-text-get-extents-fontname**; sie gibt eine Liste mit den Dimensionen des gewünschten Textes aus. Bei der Länge des Textes ist es einfach, es ist der erste Wert in der zurückgegebenen Liste, für die Höhe ist es etwas komplizierter. Nimmt man den zweiten Wert (die Höhe), stellt man fest, dass die Position zu hoch ist; man braucht den Wert von der maximalen Höhe bis zur Grundlinie der Schrift. In Englisch wird dies als „ascend“ [8] bezeichnet und ist der dritte Wert in der Liste.

Um zu verhindern, dass das Skript die gegenwärtigen Einstellungen in GIMP verändert, können diese mit **gimp-context-push** gesichert und **gimp-context-pop** wiederhergestellt werden. Zudem können mit **gimp-image-undo-group-start** und **gimp-image-undo-group-stop** alle Aktionen im Skript zu einer einzigen zusammengefasst werden. Verwendet man die Undo-Gruppierung nicht, so muss man, wenn einen das Ergebnis nicht zusagt, alle im Skript programmierten Aktionen einzeln rückgängig machen. Manchmal ist es auch nützlich, die Aktionen nacheinander rückgängig zu machen, so kann man zum Beispiel herausfinden, warum das eigene Skript nicht das gewünschte Ergebnis liefert.

```
;Einstellungen sichern
(gimp-context-push)
```



```
(gimp-image-undo-group-start inBild
)
```

Bevor die Textebene eingefügt wird, muss noch die Vordergrundfarbe auf die gewünschte eingestellt werden (hier schwarz). Danach wird der Variablen **varTextEbene** die neue Textebene zugewiesen, die an der errechneten Position den Schriftzug enthält.

```
;Vordergrundfarbe anpassen
(gimp-context-set-foreground '(0 0 0))
;Text-Layer einfüegen
(set! varTextEbene (car (gimp-text-
fontname
      inBild
      -1      ;neues TextLayer
      varX_Pos
      varY_Pos
      varText
      -1      ;Ramen=0
      TRUE    ;Antialias
      inSchriftHoehe
      PIXELS
      varSchriftart)))
```

Nach Ablauf des Skriptes sollen zwei Ebenen im Bild vorhanden sein: die gerade erstellte Textebene und die ursprüngliche Hintergrundebene. Daher wird der Textebene auch ein sinnvoller Name gegeben.

```
;Text-Layer umbenennen
(gimp-drawable-set-name
varTextEbene "Copyright")
```

Auch die Transparenz einer Ebene lässt sich anpassen und wird in diesem Beispiel auf 80 % eingestellt.

```
;Text-Layer Transparenz anpassen
(gimp-layer-set-opacity varTextEbene 80)
```

Abschließend wird nur noch die ursprünglich aktive Ebene angewählt, das Display aufgefrischt, die Undo-Gruppierung abgeschlossen und schlussendlich werden die ursprünglichen Einstellungen wiederhergestellt.

```
;die urspruengliche Ebene anwaehlen
(gimp-image-set-active-layer inBild varEbene)
(gimp-displays-flush)
(gimp-image-undo-group-end inBild)
(gimp-context-pop)
```

Daraus ergibt sich dann das fertige Skript gimp-copyright.scm.

Einfacher Rahmen

Um das eigene Bild gegen die Hintergrundfarbe des Forums abzuheben, ist ein kleiner Rahmen mit ca. 2 Pixel Breite sinnvoll. Um solch einen Rahmen in GIMP zu erstellen, geht man wie folgt vor:

1. Man vergrößert die Leinwand oder verkleinert die Bildebene,
2. fügt eine neue Ebene ein, mit der gewünschten Rahmenfarbe als Füllfarbe,
3. setzt die neue Ebene hinter die Bildebene und
4. vereint die Ebenen zu einem Bild.

Der erste Schritt kann auf zwei verschiedenen Arten verwirklicht werden: Entweder verkleinert man die Ebene mit **gimp-layer-scale** oder man vergrößert das Bild mit **gimp-image-resize**. Beides hat seine Vorzüge und auch seine Nachteile.

Verkleinert man die Ebene, so bleiben die Abmessungen des Bildes erhalten, allerdings muss man sich Gedanken über die Skalierung machen, da es ansonsten zu Verzerrungen des ursprünglichen Bildes kommt. Bei einem Rahmen von 2 Pixel kann dies sicher vernachlässigt werden. Möchte man aber einen breiteren Rahmen erstellen, ist es einfacher, die

Leinwand zu vergrößern, da man damit die Dimensionen des Bildes nicht verändert (allerdings ist das Seitenverhältnis des neuen Bildes anders als das ursprüngliche). Dafür ist es schwieriger, die geforderten Dimensionen einzuhalten, da das Gesamtbild ja größer wird, entweder skaliert man zweimal oder man rechnet den Rahmen gleich mit ein. Im folgenden Beispielskript wird die zuletzt beschriebene Variante verwendet; die Rahmenbreite wird beim Skalieren bereits berücksichtigt und später wird das Bild um den Rahmen vergrößert.

Umgesetzt wird dieser Rahmen in einer Funktion, die als Übergabeparameter außer dem Bild noch die Breite des Rahmens in Pixel und die Farbe entgegen nimmt. Dadurch kann man die Funk-



tion mehrmals aus einer anderen Funktion aufrufen, um so z. B. einen zweifarbigen Rahmen als Passepartout für den Ausdruck eines Bildes zu erzeugen, dazu aber später mehr.

Die Funktion **FarbRahmen** im Skript `gimp-farbrahmen.scn` erfüllt diese Aufgabe und ist ähnlich wie die beiden anderen aufgebaut. Daher wird lediglich das Erstellen von Ebenen in Skript-Fu kurz erklärt: Anders als in GIMP kann mit Skript-Fu nicht direkt eine neue Ebene erstellt werden. Sie muss mit **gimp-layer-new** erst definiert und einer Variablen zugewiesen werden. Danach kann man sie mit **gimp-image-add-layer** ins Bild einfügen.

Man könnte diese Funktion wieder in den GIMP-Menüs registrieren; für dieses Beispiel ist dies jedoch nicht vorgesehen und es reicht, sie in die Datei mit dem Skalierungsbeispiel einzufügen.

Die Beispiele verbinden

Die ursprüngliche Aufgabe war, eine Funktion zu erstellen, die das Bild nach den Vorgaben eines Forums oder einer Online-Plattform bearbeitet. Mit den beiden bis jetzt erstellten Dateien lassen sich nur die Bilder skalieren und nach einem weiteren Mausklick ein Copyrightvermerk einfügen. Um die Aufgabe zu erfüllen, wird noch eine weitere Funktion benötigt. Sie wird nach dem Forum oder Portal benannt und in der Datei mit der Funktion **FarbRahmen** und **BildSkalieren** gespeichert.

```
;Bild fuer das ForumXY anpassen
(define (Forum_XY
```

```
inBild)
);endDefine
```

Im Forum XY ist es nicht erwünscht, Bilder mit einer Kantenlänge größer als 1000 Pixel hochzuladen. Daher wird als Erstes die Funktion **BildSkalieren** auf das übergebene Bild angewendet. Dabei wird die maximale Kantenlänge aber auf 996 verringert, da später noch ein Rahmen mit 2 Pixel Breite hinzugefügt werden soll und es sonst zu groß wäre (siehe vorheriges Kapitel).

```
(BildSkalieren inBild 996)
```

Danach soll noch ein Copyrightstring eingefügt werden, damit man als Urheber diese Bilder identifiziert wird. Aber Halt! Die Funktion befindet sich ja in der andern Datei. Es ist möglich, dass es einen Scheme-Befehl gibt, um die Funktion **CopyRight** in dieser Datei bekannt zu machen. Dies ist aber gar nicht nötig, da Skript-Fu auf alle registrierten Funktionen zugreifen kann, also auch auf **CopyRight**. Sucht man diese Funktion im Skript-Fu-Procedure-Browser wird man feststellen, dass sie einen Parameter mehr hat als in der Datei definiert:

```
(CopyRight run-mode image value)
```

Der Parameter **run-mode** wird automatisch von GIMP mit den Konstanten RUN-INTERACTIVE oder RUN-NONINTERACTIVE belegt und darf beim Aufrufen der Funktion nicht verwendet zu werden. Dadurch sieht der Aufruf wieder wie in der Datei definiert aus:

```
(CopyRight inBild 0)
(gimp-image-flatten inBild)
```

Da die Funktion **CopyRight** die Ebenen am Ende nicht vereint, muss dies jetzt geschehen, bevor der 2 Pixel große Rahmen gezogen wird:

```
(FarbRahmen inBild 2 '(0 0 0))
```

Mit Undo-Gruppierung und Sicherung der Einstellungen sieht die Funktion wie folgt aus:

```
;Bild fuer das ForumXY anpassen
(define (Forum_XY
inBild)

;Einstellungen sichern
(gimp-context-push)
(gimp-image-undo-group-start ~
inBild)

;Bild bearbeiten
(BildSkalieren inBild 996)
(CopyRight inBild 0)
(gimp-image-flatten inBild)
(FarbRahmen inBild 2 '(0 0 0))

;Einstellungen wiederherstellen
(gimp-displays-flush)
(gimp-image-undo-group-end ~
inBild)
(gimp-context-pop)

);endDefine
```



Ein Beispielbild bearbeitet mit **Forum_XY**. 🔍

Natürlich muss jetzt noch die Registrierung in den Menüs angepasst werden, sodass anstelle der Funktion **BildSkalieren** jetzt die Funktion **Forum_XY** ausgeführt wird. Der Parameter **SF-ADJUSTMENT** fällt weg, da **Forum_XY** die maximale Größe fest einprogrammiert hat.

Bei der Vorstellung der Funktion **FarbRahmen** wurde erwähnt, dass man sie auch mehrmals

aufrufen kann, um z. B. ein Passepartout zu erzeugen. Dies soll für die Veröffentlichungen in der FotoCommunity verwendet werden und kann wie folgt aussehen:

```
;Bild fuer fc anpassen
(define (fotocommunity
  inBild)
```

```
;Einstellungen sichern
(gimp-context-push)
(gimp-image-undo-group-start ~
inBild)

;Bild bearbeiten
(BildSkalieren inBild 936)

;zweifarbiger Rahmen
(FarbRahmen inBild 2 '(0 0 0))
(FarbRahmen inBild 30 '(255 255~
255))

;Copyright in den weissen Rand ~
platzieren
(CopyRight inBild 10)
(gimp-image-flatten inBild)

;Einstellungen wiederherstellen
(gimp-displays-flush)
(gimp-image-undo-group-end ~
inBild)
(gimp-context-pop)

);endDefine
```

Auch hier wird dem Größenlimit zuerst Rechnung getragen, danach werden zwei Rahmen mit insgesamt 32 Pixel erzeugt, um dann einen Copyright-Schriftzug mit 10 Pixel Höhe und 15 Pixel Abstand zum Rand in den weißen Rahmen zu platzieren.

Fazit

Mit Skript-Fu lässt sich GIMP an die eigenen Bedürfnisse anpassen und immer wiederkehrende



LINKS



© Ralf Damaschke

Dasselbe Bild mit **FotoCommunity** bearbeitet. 🔍

Aufgaben können mit einem Mausklick erledigt werden. Dabei sind der Fantasie keine Grenzen gesetzt und die vorgestellten Skripte sind nur sehr einfache Beispiele, um die Arbeitsweise zu verdeutlichen. Komplexere Beispiel gibt es zuhauf im Internet [9] [10]. Besonders schön ist die Möglichkeit, Dimensionen im Bild zu berechnen und dadurch ein immer gleiches Aussehen zu erzeugen, egal wieviel Megapixel die zu bearbeitenden Bilder haben. So werden z.B. die Schatten an den Bildern im **freiesMagazin** über ein Skript erstellt,

so manches Parameters erst nach zeitraubender Recherche deutlich. Besonders schade ist, dass GIMP nur über Skripte zu steuern ist. Natürlich hat das auch Vorteile, aber es gibt viele Anwender, die keine Programmiersprache beherrschen. Ihnen würde ein Makrorekorder oder eine „grafische“ Programmierung den Einstieg erleichtern.

Alles im allem sind die Hürden aber zu meistern und hat man sich erst an die Eigenarten gewöhnt, macht es riesig Spaß, GIMP mit Skript-Fu zu automatisieren.

das ohne Zutun des Benutzers die Satzrichtlinie umsetzt.

Leider ist Scheme als Sprache nicht besonders benutzerfreundlich, und ich habe mich bis jetzt noch nicht mit ihr anfreunden können. Ein anderer Kritikpunkt ist die Hilfe: Zwar sind alle Funktionen in der Konsole beschrieben, aber eben nur in Englisch. Passende Anwendungsbeispiele muss man sich im Internet suchen, und so wird die Funktion

- [1] <http://www.freiesmagazin.de/freiesMagazin-2010-01>
- [2] <http://wiki.gimpforum.de/wiki/Skript-Fu>
- [3] <http://www.gimp.org/docs/python/index.html>
- [4] <http://docs.gimp.org/2.6/de/gimp-using-script-fu-tutorial.html> 🇬🇧
- [5] <http://de.wikipedia.org/wiki/Scheme>
- [6] http://de.wikipedia.org/wiki/Polnische_Notation
- [7] <http://docs.gimp.org/de/gimp-using-script-fu-tutorial-first-script.html>
- [8] http://en.wikipedia.org/wiki/Typeface#Font_metrics 🇬🇧
- [9] <http://www.td-e.com/soft-de/watermark-it.php>
- [10] <http://www.gimphelp.org/script24.shtml> 🇬🇧

Autoreninformation



Ralf Damaschke ist begeisterter Hobbyfotograf und verwendet seit Jahren diverse Bildbearbeitungsprogramme, um seine Bilder nachzuarbeiten. Als Setzer bei **freiesMagazin** muss er immer wieder Schlagschatten in definierte Größen zu Bildern hinzufügen; diese Aufgabe erledigt heute ein Skript-Fu.

Diesen Artikel kommentieren



MegaGlest – Ein historisch nicht ganz korrektes Strategiespiel von Michael Schwarz

Wer unter Linux spielt und Strategiespiele à la Age of Empires mag, der könnte das ein oder andere mal über den Begriff Glest [1] gestolpert sein. Glest ist ein 3-D-Echtzeitstrategiespiel, dessen Entwicklung bereits im Jahr 2001 begann.

Im Moment ist es ruhig um das Open-Source-Projekt [2] geworden. Man könnte fast meinen, zu ruhig. Fans rund um das Spiel fassten sich wegen der Stille um das Projekt ein Herz und gründeten, basierend auf Glest, einen separaten Entwicklungszweig [3]: *MegaGlest* [4] genannt.

Was *MegaGlest* genau ist, worin die Unterschiede zum bisherigen Glest bestehen und mit welcher Vielfalt die einzelnen Fraktionen aufeinander losgehen, darauf wird in den folgenden Zeilen etwas genauer eingegangen.

Installation

Zuallererst sei dem Projekt und dessen Entwicklern zugutegehalten, dass für *MegaGlest* eigens eine grafische Installation angefertigt wurde. Somit ist es selbst für frisch gebackene Ex-Windowsler oder generell nicht allzu versierte Linuxbenutzer ein leichtes Unterfangen, das Spiel in seiner Pracht auf die eigene Festplatte zu zaubern.

Nach dem abgeschlossenen Download des Installationsprogramms kann es allenfalls nötig sein, diesem das Recht zu gewähren, ausgeführt zu werden. Dazu ruft man eine Konsole

auf, wechselt in das Verzeichnis in dem sich das heruntergeladene Installationsprogramm befindet und führt den Befehl

```
$ chmod a+x Mega-Glest-Installer-*.*)_i386_linux
```

aus. Das sollte die nötigen Berechtigungen zum Ausführen der Datei festlegen. Wenn man sich sowieso gerade auf der Konsole befindet, kann man anschließend über diesen Befehl

```
$ ./Mega-Glest-Installer-*.*)_i386_linux
```

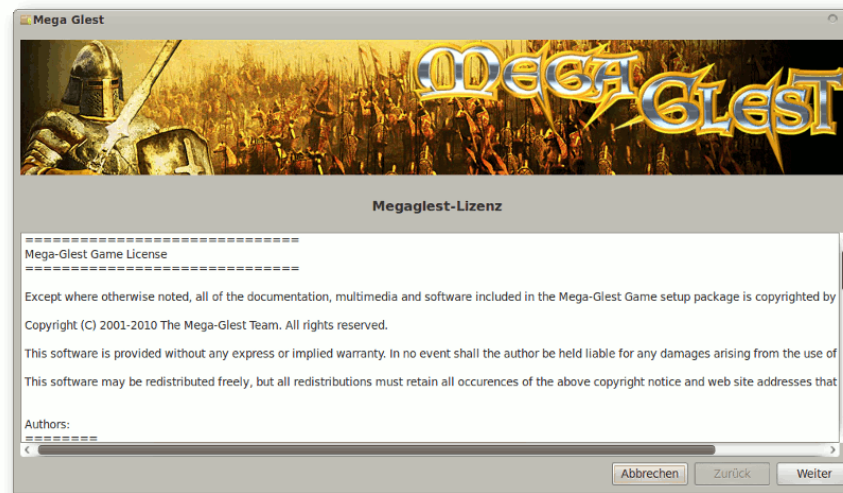
das Installationsprogramm starten. Konsolenmuffel können die Installation auch über einen Doppelklick der Datei aufrufen. Im Anschluss verhält sich die Installation sehr pflegeleicht und geleitet komfortabel durch den zugehörigen Prozess.

Alternativ zum Installationsprogramm können auch die ausführbaren Dateien sowie die Datenpakete des Spiels heruntergeladen und zusammen in den gleichen Ordner entpackt werden.

Die Grundlagen zuerst

Um nicht den ganzen Spaß am Selbstentdecken zu nehmen: An dieser Stelle erst mal nur die Grundlagen. Jede Fraktion des Spiels besitzt ihr Haupthaus. Das Haupthaus dient gleichzeitig als Lager für alle Ressourcen und Vorräte. Verliert eine Fraktion ihr Haupthaus, ist das Spiel für diese vorbei. Kein Lager, keine Ressourcen und keine Baumöglichkeit für ein neues Haupthaus. Bedeutet: Spiel vorüber, Schlacht verloren.

Um dem zu entgehen, stehen dem Spieler Arbeiter zur Verfügung die alle wichtigen Güter in Form von Gold, Stein und Holz heranschaffen und in Gebäude umsetzen können. Ob ein Gebäude gebaut werden kann, hängt davon ab, ob bereits alle Voraussetzungen dafür geschaffen wurden. Manchmal fehlt eine noch nicht erforschte Verbesserung, ein anderes Mal ein Gebäude und nicht selten beides. Wie so oft gilt es also, sich von ganz unten



Der grafische Installer von MegaGlest. 



Solch eine Idylle hält meist nicht lange an. 🔍

nach ganz oben zu hangeln, um am Ende stärkere und bessere Einheiten ausbilden zu können.

Damit rutscht man auch schon in den essenziellen Teil des Spiels, den Kampf zwischen zwei oder mehreren Gegnern. Eine schön aufgebaute

Basis gewinnt auf dem Schlachtfeld keinen Blumentopf ohne die passende Armee, die diese verteidigt oder Präventivschläge durchführt. *MegaGlest* bietet dafür Nah- und Fernkampf sowie einige Flugeinheiten. Je nach Kampfkraft und Nutzen kosten Einheiten einen Anteil an Roh-

stoffen, zumeist Gold und Holz. Außerdem benötigt jede Einheit im Spiel Nahrung, um zu bestehen.

Hier grenzt sich *MegaGlest* von anderen Titeln ab und geht konsequent seinen eigenen Weg. Während normalerweise eine Statusmeldung auf dem Bildschirm erscheint, die dem Spieler vermittelt, dass er wegen mangelnder Nahrung keine Einheiten mehr ausbilden kann, wird das in der Welt von *MegaGlest* anders gehandhabt. Dort kann auch bei erreichtem Limit fleißig weiter ausgebildet werden. Jedoch passiert es dann auch, dass man sich schnell fragt, wo die eben fertiggestellten Einheiten plötzlich hin sind. So viel sei gesagt: In *MegaGlest* ereilt einen der Hungertod relativ fix und unerwartet. Durch ein Zufallsprinzip werden bei Überschreiten der Nahrungsmittelgrenze Einheiten aus den Reihen des Spielers entfernt – so lange, bis die verbliebene Anzahl der Einheiten wieder mit Nahrung versorgt werden kann. Übereilte Panik gehört aber nicht hierher. Sollte das Limit nur kurzzeitig überschritten werden, nimmt einem das weder das Spiel noch die eigenen Einheiten übel. Einfach schnell genug eine Farm gebaut oder ein entsprechendes Nahrungstier gezüchtet, schon ist alles im Rahmen.

Unterschiede zum ursprünglichen Glest

In *Glest* zog man entweder mit den magisch begabten Menschen in die Schlacht oder schloss sich den nicht magischen, dafür aber hochtechnisierten Menschen an. Zusammengefasst lässt

sich das Spiel mit den Worten „Was wäre gewesen, hätte es damals im Mittelalter wirklich Hexen gegeben und hätten diese sich gegen die Normalen verbündet“ beschreiben. Zwei gegeneinander kämpfende Gruppen waren den scheinbar davon gelangweilten Fans von Glest wohl zu wenig. Sie schmiedeten und balancierten neue Gruppierungen aus. In *MegaGlest* stehen dem Spieler insgesamt sechs verschiedene Fraktionen zur Verfügung. Neben den immer noch vorhandenen Magiern und Technikern mischen sich dort auch die Perser, die Indianer, die Ägypter und die Normannen ins Geschehen mit ein. Diese vier Fraktionen sind nicht so strikt in ihren Eigenschaften getrennt wie die Techniker und Magier, können also durchaus tolle Gerätschaften und mächtige Zauber ihr Eigen nennen. Das Gleichgewicht zwischen allen sechs rivalisierenden Gruppen wurde hierbei sehr gut gehalten.

Eine weitere Verbesserung an *MegaGlest* ist das Bereitstellen eines Hauptservers für Spiele über das Internet. Während im normalen Glest noch, manchmal mühsam, die IP-Adresse des Servers angegeben werden musste, mit dem man spielen wollte, kann man bei *MegaGlest* auf einem dafür eingerichteten Spielserverserver danach suchen. Dort werden alle derzeit offenen Partien angezeigt und mittels Klick auf das bestimmtes Spiel nimmt man daran teil.

Alle Kreativen werden sich darüber freuen, dass der Karteneditor bei *MegaGlest* im Hauptteil enthalten ist, also nicht wie bisher separat heruntergeladen werden muss.

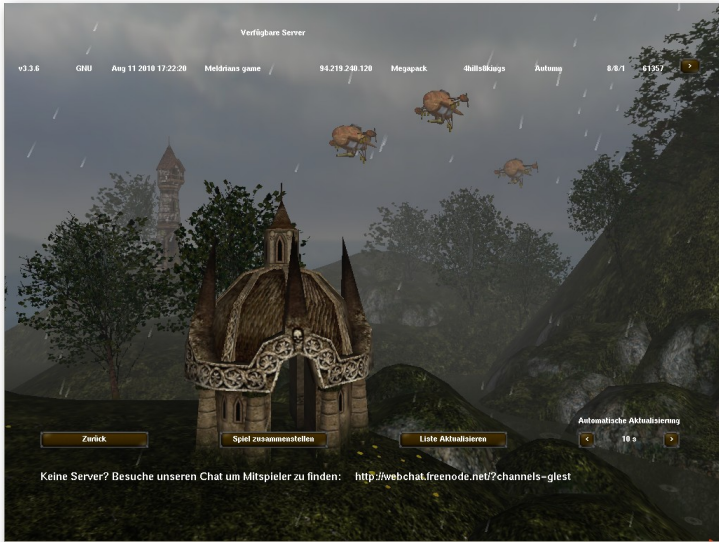


Die Basis der Ägypter wird angegriffen. 🔍

Taktisches Feingefühl gefragt

Wie beginnt man nun eigentlich am besten das Spiel, könnte sich der Leser in freudiger Erwartung fragen. An dieser Stelle eine kleine Warnung: Es macht viel mehr Spaß, wenn man sich die grundlegenden Spielzüge gegen einen

leichten Computergegner selbst aneignet. Wer also selbst auf Entdeckungsreise mit *MegaGlest* gehen möchte, überspringt diesen Abschnitt lieber. Wer gerne eine kleine Einstiegshilfe hätte oder sich nichts aus langem Herumprobieren macht, der darf gerne weiterlesen.



Alle laufenden Spiele im Blick. 

Einen „goldenen“ Einstieg, geltend für alle Teams, gibt es so nicht, dafür unterscheiden sich die einzelnen Fraktionen zu sehr voneinander. Wichtig ist aber die Beschaffung der Rohstoffe, speziell von Gold. Wenn eine Runde startet, stehen einem im Regelfall drei Arbeiter zur Verfügung. Diese drei Gesellen sollten sich direkt um den Goldabbau kümmern. Von den Anfangsressourcen bildet man schnell einen weiteren Arbeiter aus. Dieser sollte sich dann um die ersten Gebäude kümmern. Ich persönlich favorisiere den Bau einer Ausbildungsstätte für Krieger und im Anschluss daran eines Gebäudes zur Gewinnung von Nahrung. Mit diesen beiden Gebäuden kann bereits eine kleine Abwehrarmee aufgestellt werden. Zumindest im Kampf gegen den Computer braucht man diese auch. Dieser ist

selbst auf CPU-easy ein recht angriffs-lustiger Mitspieler, wenn auch kein sehr geschickter. Während die ersten Gebäude gebaut und einige Einheiten ausgebildet werden, sollte man noch einige Arbeiter ausbilden und zum Goldabbau schicken. Auch sollten drei bis vier Arbeiter zum Holz hacken befehligt werden und einer oder zwei am Steinbruch ihre Arbeit verrichten.

Im Anschluss spalten sich die Fraktionen in der weiteren Vorgehensweise auf. Jetzt heißt es selbst auszuprobieren und den geschicktesten Weg zwischen Gebäudebau und Forschung zu finden, dabei aber keinesfalls die Basisverteidigung aus den Augen zu verlieren.

Fazit

MegaGlest bereichert das leider eingeschlafene Projekt Glest durch Verbesserungen und neue Klassen. Die Weiterentwicklung des Spiels ist rege und in vollem Gange. Der Echtzeitstrategie-sektor im Bereich Linux und Open Source wird durch das etwas an *Warcraft III* erinnernde *MegaGlest* ausreichend ergänzt. Eines der vereinzelt wirklich störenden Mankos, die derzeit noch herrschen, ist die leicht ungenaue Anvisierung von feindlichen Einheiten auf dem Schlachtfeld. Nicht selten weist man seinen Trupp Bogenschützen versehentlich an, direkt auf den Feind zuzulaufen, anstatt ihn aus der Ferne zu beschießen. Doch dieser Kritikpunkt lässt sich mit etwas Eigendisziplin beheben. Das heißt, man gewöhnt sich daran, etwas vorsichtiger zu klicken.

Schnell ist eine der sechs Gruppen zum Liebling geworden, noch schneller mit dieser Gruppe der Computergegner niedergemacht und schon steht dem Onlineduell Mensch gegen Mensch nichts mehr im Wege. Wenn einem dann doch auf einer der vielen mitgelieferten Karten langweilig werden sollte, greift man zum Editor und gestaltet eine eigene. *MegaGlest* kann Hobbystrategen durchaus zufrieden stellen – und darauf kommt es beim Spielen an.

LINKS

- [1] <http://glest.org/en/index.php> 
- [2] <http://sourceforge.net/projects/glest/> 
- [3] http://glest.org/glest_board/index.php?board=20.0 
- [4] <http://sourceforge.net/projects/megaglest/> 

Autoreninformation

Michael Schwarz wurde es Mitte 2008 zu bunt auf dem DRM- und kopierschutzverseuchten Windows-spielmarkt. Nach seinem Umstieg auf Linux hat er auf der Suche nach Spielen für das freie Betriebssystem schon viele positive Überraschungen entdecken können, darunter auch MegaGlest.

[Diesen Artikel kommentieren](#) 



Wünsch Dir was – SUSE Studio als Tool zum eigenen Linux von Oliver Johandrees

Seit einiger Zeit, und ein wenig im Verborgenen, hat Novell mit **SUSE Studio** [1] eine interessante Plattform bereit gestellt. Mit deren Hilfe können sich Linux-Freunde schnell und unkompliziert ihre eigene SUSE-Distribution zusammenklicken. Das funktioniert tatsächlich ebenso einfach, wie es sich anhört, und im Folgenden wird gezeigt, wie das geht.

Wäre es nicht schön, wenn man sich sein SUSE so zusammenstellen könnte, wie man es selbst gerne hätte? Nur mit den Tools und Programmen, die nötig sind und ohne den Ballast einer großen Distribution? Wäre es nicht schön, wenn man das Ausgabeformat selbst bestimmen könnte? Während der eine sich das SUSE für seinen USB-Stick wünscht, möchte der andere vielleicht lieber eine Live-CD. Ein Dritter bevorzugt eventuell eine virtuelle Maschine für VirtualBox [2], VMware [3] oder für XEN [4].

Oder noch viel schöner: Wäre es nicht toll, wenn es bereits eine Sammlung von vorgefertigten Komplettlösungen gäbe, aus der man sich einfach nur zu bedienen braucht? Nun, all das leistet **SUSE Studio**.

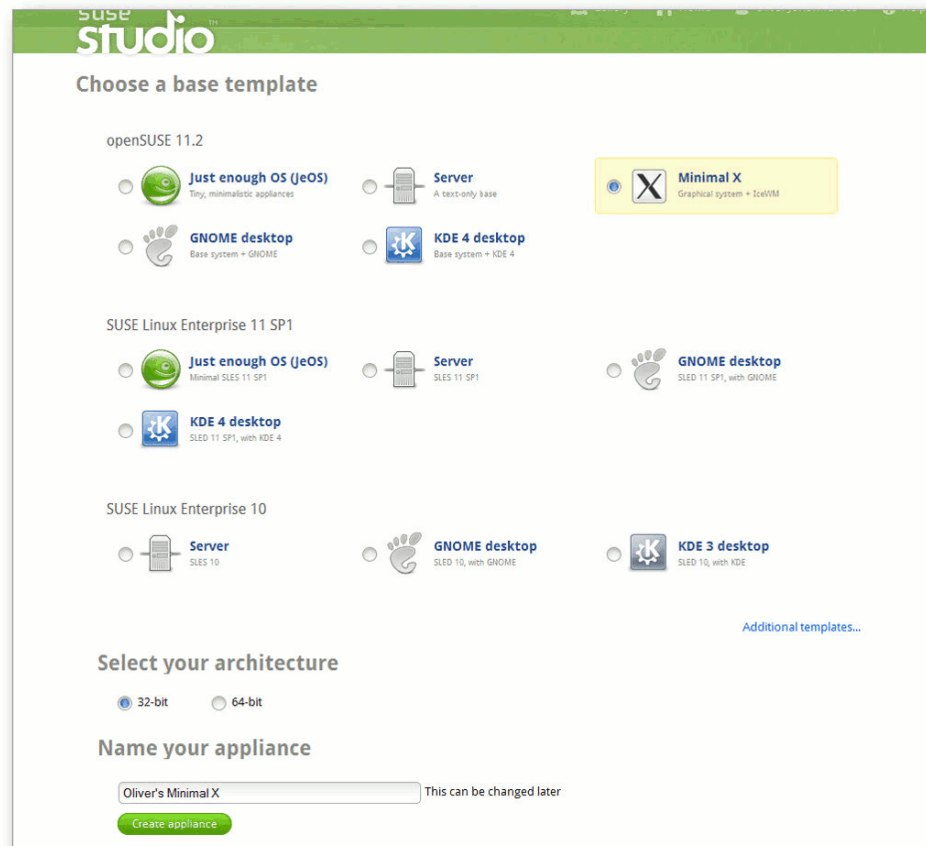
Es ist eine Plattform, die alles zur Verfügung stellt, um schnell und unkompliziert die eben genannten Anwendungen, sogenannte „Appliances“, zu generieren. Appliances sind komplett lauffähige Zusammenstellungen von Betriebssystem, Programmen und Konfigurationen, die

sofort einsatzfähig sind und alles enthalten, was zu ihrem Betrieb notwendig ist. Jeder kennt mittlerweile die virtuellen Maschinen, die ganz einfach per Mausklick gestartet werden können und,

dass man sie jemand anderem auf einem USB-Stick mitgeben könnte, der sie dann ohne Installationsaufwand bei sich zu Hause ausführen kann. Dabei muss er sich weder mit der Materie auskennen noch ein Linux installieren können und schon gar nicht Kenntnisse von tiefergehenden Konfigurationen haben.

Genau diesen Ansatz verfolgt **SUSE Studio**. Hier können Interessierte die besagten Appliances zusammenstellen und sie auf Wunsch auch anderen in der SUSE Gallery [5] zur Verfügung stellen. Sucht man beispielsweise ein SUSE Enterprise 11 mit einem Apache Webserver, so braucht man das nicht mehr selbst zusammenschrauben, sondern lädt sich aus der SUSE Gallery eine fertige Appliance, die bereits jemand anderes erstellt

und konfiguriert hat. Der Rest beschränkt sich auf das Starten und Nutzen des fertigen Gesamtpaketes.



Erste grundlegende Einstellungen. 🔍

in sich geschlossen, alles mitbringen, um zu funktionieren. Streng genommen ist eine virtuelle Maschine eine Appliance. Sie hat den Vorteil,



Willkommen im Club

Noch ist das alles nicht für die großen Massen freigegeben. Zwar steht *SUSE Studio* bereits seit längerem im Internet bereit, nutzen können es jedoch nur sogenannte eingeladene Gäste. Das Ganze ist aber müßig, denn einladen lassen kann sich jeder, sodass es sich eigentlich nur noch um einen simplen Registrierungsvorgang handelt, wie in jedem anderen Forum auch.

Interessanterweise nutzt Novell [6] neben eigenen Accounts auch Zugänge anderer Anbieter für die Autorisierung. So kann man sich nach erfolgter Einladung auch mit seinem Google-Mail-Account [7], seinem Yahoo-Account [8] oder seinem OpenID-Account anmelden und erhält anschließend Zugriff auf die Funktionen des Studios.

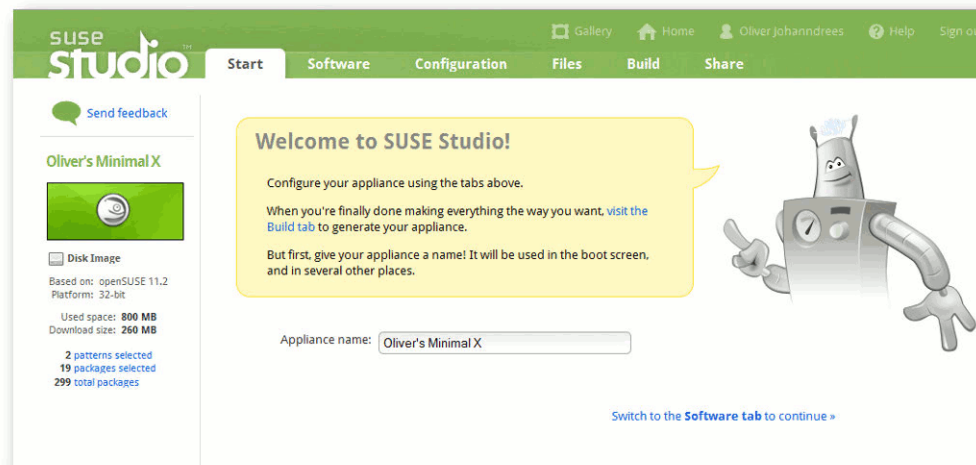
Ist man erst einmal eingeloggt, geht es auch direkt ans Eingemachte. Das Ganze beginnt mit der Auswahl der sogenannten Basisvorlage, die grundsätzlich den Funktionsumfang der geplanten Appliance definiert. Gemeint ist damit z. B. die Wahl der SUSE-Distribution. Möchte man einen Server aus der SUSE Enterprise 10 oder 11-Reihe? Oder vielleicht lieber ein openSUSE 11.2? Entscheidet man sich für ein 32-Bit-OS oder für die 64-Bit-Variante? Und, nicht zu vergessen, hier legt man bereits das „Look & Feel“ der maßgeschneiderten Distribution fest: GNOME [9], KDE [10] und andere.

Auf einer weiteren Seite können versierte Anwender Server-Vorlagen mit integrierter IngresSoft-

ware [11] wählen, einen LAMP-Server mit Apache, MySQL und PHP oder aktualisierte KDE- und GNOME-Versionen.

Bitte spezifizieren sie

Nach der grundlegenden Vorauswahl werden weitere Details festgelegt. So handelt man sich von nun an durch ein paar wenige Dateireiter, in denen durch kurzes, knappes Anklicken alle weiteren Feinheiten festgelegt werden – so zum Beispiel die Namensgebung der neuen Appliance.



Es geht an die Details. 🔍

Die erstellten Appliances werden unter ihrem Namen im Account des Anwenders abgelegt und für maximal sieben Tage gespeichert. Erst danach verfallen sie und werden gelöscht. Bis dahin kann man sie jederzeit modifizieren, testen und beliebig downloaden. Keine Angst: Ist die Appliance erst einmal verfallen, kann sie natürlich jederzeit neu zusammengestellt und erzeugt werden. Da

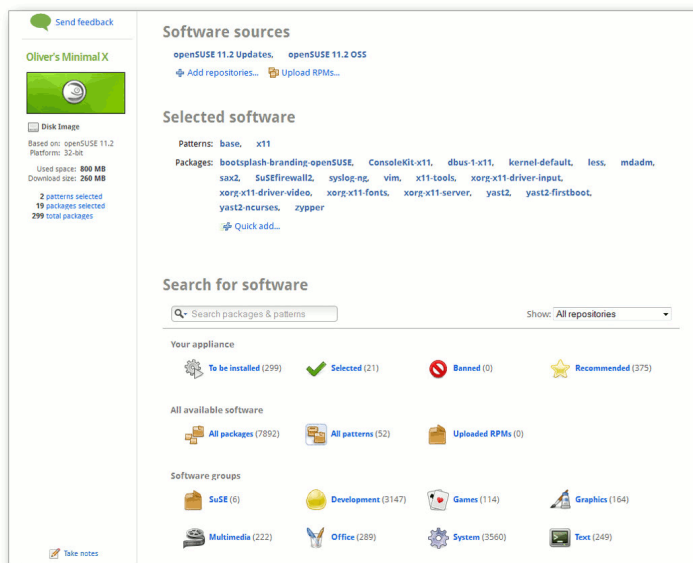
ganze Betriebssystem-Images reichlich Platz belegen, ist es verständlich, wenn Novell nicht alles für endlose Zeiten vorhält.

Auf der linken Seite wird während des Erstellens stets angegeben, wie groß die Appliance derzeit ist und welchen Umfang der komprimierte Download haben wird. Hinzu kommen Angaben zur Appliance selbst, wie der Name und die Anzahl der zusammengestellten Pakete. Wer also plant, sich eine Distribution für seinen USB-Stick

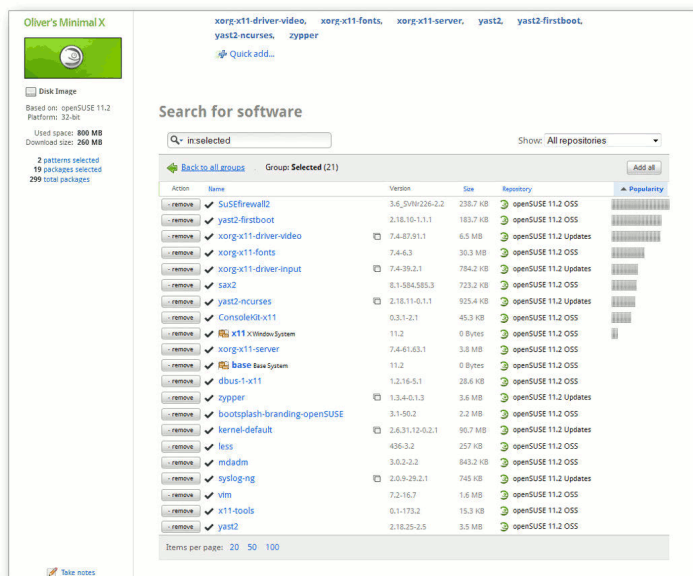
zusammenzustellen, der sieht hier jederzeit, wie groß die Appliance sein wird.

Ein ganz wichtiger Punkt, und wohl auch der Hauptreiter, ist das Thema „Software“. Hier wird, ebenfalls durch einen einfachen Mausklick, eine Auswahl von Softwarepaketen bestimmt, die in der

Appliance verfügbar sein werden. Unterschieden wird neben einzelnen Anwendungen auch zwischen ganzen Paketgruppen, so wie es der Linux-Anwender auch aus der RPM-Paketverwaltung in Yast kennt. Es gibt einen Punkt „Recommended“, also empfohlene Software, die man durchsehen sollte, um wirklich oft benutzte Anwendungen nicht zu übersehen.



Die Auswahl von Paketen und Paketgruppen. 🔍



Die Paketauswahl im Detail. 🔍

Im Detail werden in den Paketkategorien die einzelnen Softwarepakete aufgelistet und können durch einen einfachen Mausklick auf „add“ oder „remove“ hinzugefügt oder entfernt werden. In der linken Spalte der Auswahlseite, dort wo die Zusammenfassung der Appliance steht, werden zudem Hinweise und Warnungen ausgesprochen, wenn die Wahl des Benutzers Probleme verursacht.

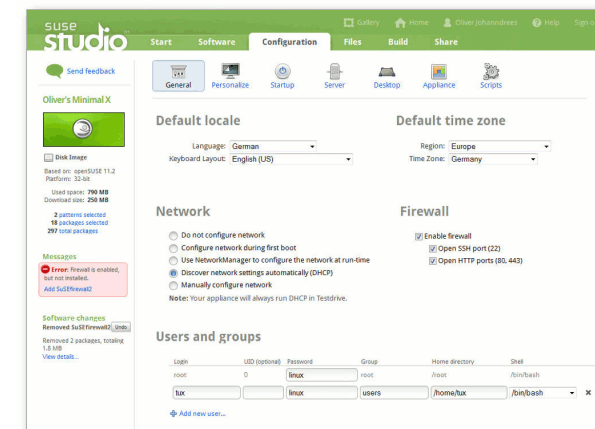
Ein Teil der Pakete lässt sich weder hinzufügen noch abwählen, da sie verständlicherweise zwingend für das Funktionieren der Appliance notwendig sind. Diesbezüglich hat jede Appliance bereits eine festgelegte Mindestgröße, die nicht unterschritten werden kann. Da die SUSE-Distributionen insgesamt recht mächtig sind, wird man hier keine Minimal-Rekorde aufstellen können. Wer auf der Jagd nach der kleinsten Distribution ist, ist hier nicht am richtigen Ort.

Finetuning

Um etwas Konfiguration kommt man auch in *SUSE Studio* nicht herum. Aber keine Sorge, es ist nicht mehr als die grobe Festlegung einfachster Dinge. Den Namen der Benutzer oder das Einschalten der Firewall sollte jeder Laie festlegen können.

An dieser Stelle legt man auch die Lokalisierungseinstellungen fest, also welche Sprache die Appliance haben wird und

welches Keyboardlayout gewünscht ist. Im Test waren diese Festlegungen weniger erfolgreich, denn obwohl mehrfach „German“ als Sprache gewählt wurde, enthielt die später erstellte Appliance englische Texte. Hier müsste Novell etwas nachlegen und diese Fehler beheben.



Grundlegende Einstellungen zu Benutzern und Netzwerk. 🔍

Es finden auch gewisse Plausibilitätstests statt. So wird etwa darauf hingewiesen, wenn man die Firewall als aktiviert anhakt, nicht aber die zugehörige Software installiert.

Am unteren Bildrand können weitere Nutzer für das System festgelegt und eingerichtet werden. Die Aktionen beschränken sich auch hier auf simple Mausklicks und ein paar wenige Angaben, reichen aber voll und ganz für die Wünsche des Anwenders.

Unter „Personalize“ lässt sich ein Bootscreen auswählen und es gibt sogar die Möglichkeit, hier



eine eigene Grafik einzusetzen. Das letzte Ur-laubsfoto als Bootscreen? – Kein Problem.

Unter „Startup“ lässt sich durch einfache Auswahl der Runlevel nach dem Start festlegen. Hier kann man sich schnell zwischen Konsole oder grafischem Login entscheiden. Interessanterweise bietet sich dort auch die Möglichkeit, eine eigene EULA (Nutzervereinbarung) zu integrieren, die mit der Appliance abgefragt wird.

Der Punkt „Server“ bietet derzeit lediglich die Auswahl, eine PostgreSQL- oder eine MySQL-Datenbank mit zu installieren und bereitzustellen.

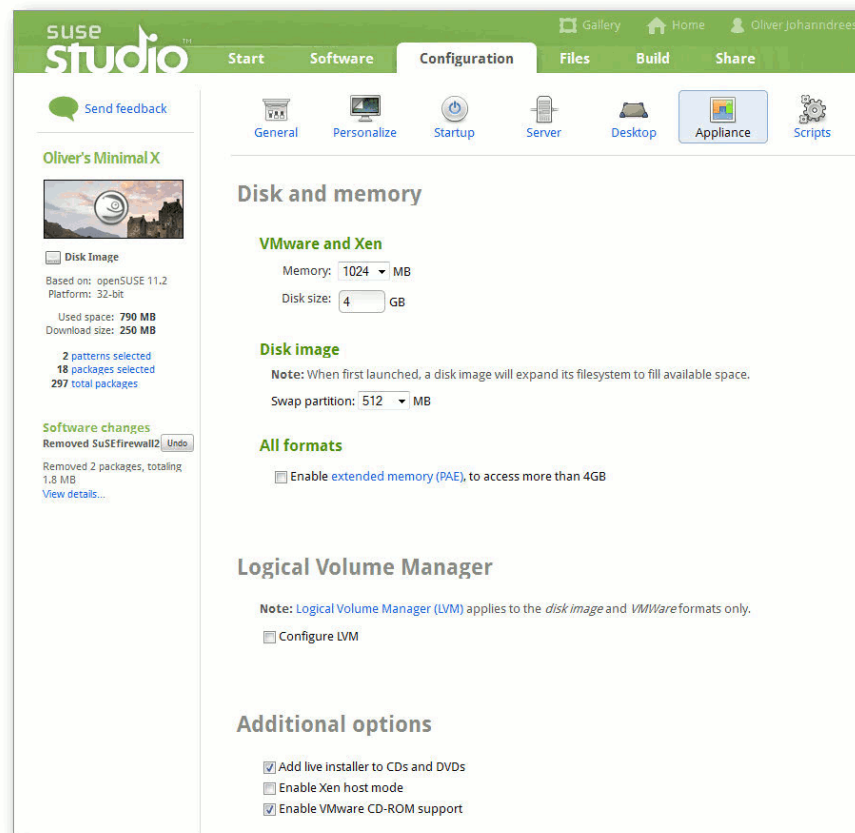
Der Konfigurationspunkt „Desktop“ fragt ab, ob und welcher Nutzer automatisch eingeloggt werden soll und bietet die Möglichkeit, hier Programme zu definieren, die bereits beim Autostart der Appliance ausgeführt werden sollen. Der klassische Autostart also.

Interessant ist der Punkt „Appliance“, definiert er doch grundlegende Parameter der Zusammenstellung: Wie groß wird ggf. der Hauptspeicher einer Appliance als virtuelle Maschine sein? Wie groß die verwendete virtuelle Festplatte?

Für Diskimages und USB-Varianten ist es interessant, die Größe der Auslagerungspartition (SWAP) zu bestimmen. Und am Ende kann hier entschieden werden, ob ein Live-Installer einer möglichen CD-Version hinzugefügt werden

soll oder CD-ROM-Unterstützung für VMWare Berücksichtigung findet.

Der Konfigurationspunkt „Scripts“ legt fest, ob und welches Skript am Ende des Zusammenstellungsprozesses oder vielleicht bei jedem Start



Speicherdetails. 🔍

der Appliance ausgeführt werden soll. Derzeit lässt sich wohl nur ein einziges Skript ausführen, aber man könnte das Problem ggf. dadurch umgehen, dass man innerhalb dieses Skriptes wei-

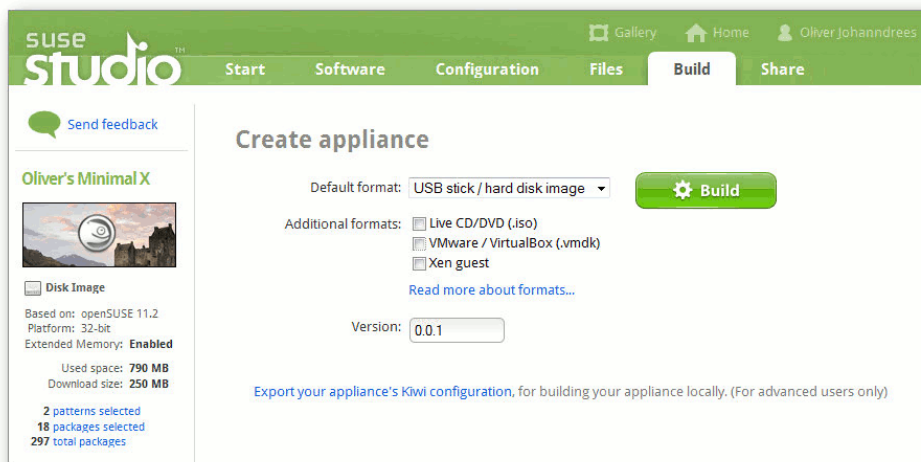
tere Skripte startet. In der Regel wird man diese Möglichkeiten kaum nutzen, aber es ist schön, dass es sie gibt.

Nachdem jetzt alles konfiguriert ist, können unter dem Hauptpunkt „Files“ einzelne Dateien oder auch ganze Archive festgelegt werden, die zur Appliance hinzugefügt werden sollen. Möchte also jemand seine persönliche Fotosammlung als zwingenden Bestandteil seiner Distribution mitgeben wollen, so kann er hier ganz individuell Dateien beilegen. Gibt man ein gepacktes Archiv an, so wird der Inhalt anschließend in ein festzulegendes Verzeichnis der Appliance extrahiert. Die gängigen Archiv-Formate, wie tar, tar.gz, tar.bz2 oder zip werden selbstverständlich unterstützt.

Machen sie es so!

Der Menüpunkt „Build“ macht seinem Namen alle Ehre: Hier erstellt man nach erfolgreicher Zusammenstellung die gewünschte Appliance. Aber zunächst will SUSE Studio wissen, um welche Variante es sich handeln soll.

Zur Wahl stehen hier die Live-CD/DVD, also ein ISO-Image, das auf eine silberne Scheibe gebrannt werden kann, USB-Stick- oder Harddisk-Image, VMWare- oder Virtualbox-VMDK oder ein XENGast.



Die Geburt der neuen Appliance. 🔍

Wer mehrere Versionen seiner Appliance in einem Zuge erstellen möchte, kann zusätzlich zur getroffenen Auswahl einige Häkchen setzen und wird anschließend passend bedient.

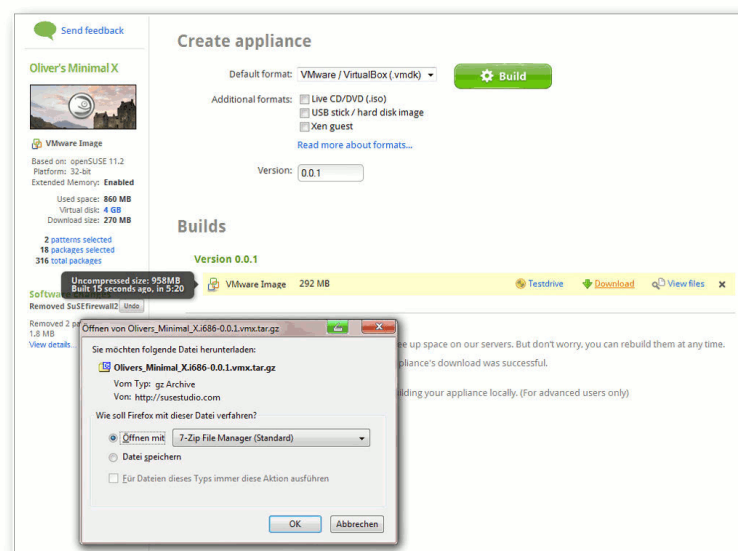
Die Version 0.0.1 des zur Verfügung stehenden Appliance-Builders zeigt, dass sich *SUSE Studio* noch ziemlich am Anfang befindet. Dennoch erstaunt die Ausgereiftheit des gesamten Konzeptes.

Mit dem abschließenden Klick auf den großen Button „Build“ geht es los. Die neue Appliance erblickt das Licht der Welt. Und das dauert gar nicht lange. In verschiedenen Schritten, die der Benutzer nur in Form eines Fortschrittbalkens wahrnimmt, werden nun alle notwendigen Softwarepakete in die Appliance kopiert, das Boot-Image erzeugt und die letzten automatischen Konfigurationen durchgeführt. Nach nicht einmal fünf

Minuten Kurzweil vermeldet *SUSE Studio* die erfolgreiche Erstellung der neuen Distribution. An dieser Stelle erfolgt auch der Hinweis, dass die Appliance nur für sieben Tage vorgehalten wird, aber natürlich jederzeit neu erstellt werden kann.

Es ist angerichtet

War die Zusammenstellung erfolgreich, kann mit der „Auslieferung“ begonnen werden. Zuvor erhält man einige Informationen über die Erstellungsdauer, die Größe der Appliance und deren Typ.

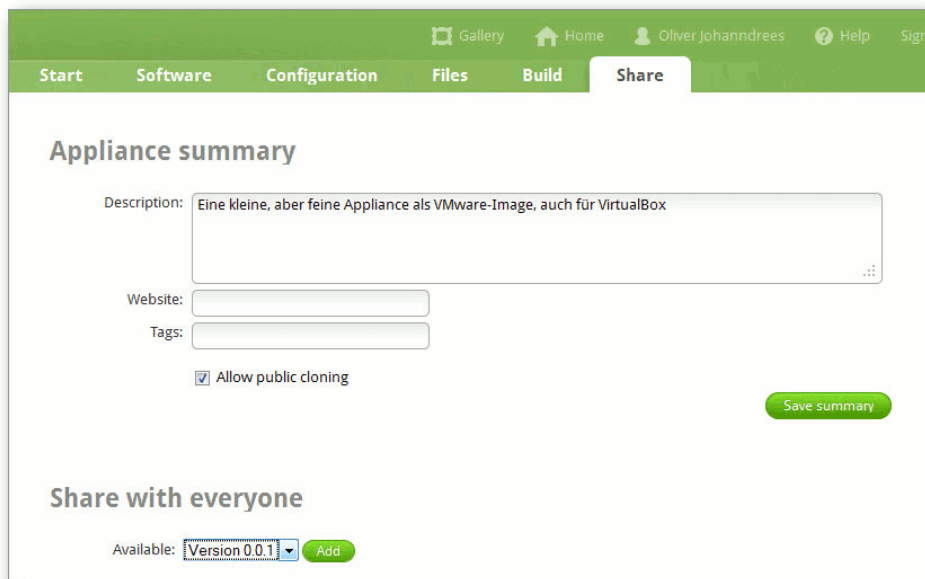


Der Moment der Auslieferung. 🔍

Sehr interessant ist, dass bereits hier die Möglichkeit besteht, das Werk online auf seine Funktion zu testen. Ein Klick auf den Link „Testdrive“ startet die Appliance schon im Browser und lässt einen ersten Blick auf das zu, was man gerade frisch erstellt hat. Dadurch erspart man sich ggf. lange Download-Zeiten, nur um anschließend festzustellen, dass man ein wichtiges Paket vergessen hat. Nachträgliche Änderungen an der Zusammenstellung sind rasch und per Mausklick durchgeführt und ein neuer Bauvorgang ist schnell angestoßen.

Mit einem simplen Download, der in der Regel länger dauert als das gesamte Prozedere, befördert man sich das frisch gepresste Werk auf den heimischen PC. Der Downloadumfang wird durch ein Gzip-komprimiertes Tar-Archiv soweit es geht in Grenzen gehalten.

Abschließend hat man die Möglichkeit, sein Werk mit anderen zu teilen und seine ganz persönliche Appliance der Community zur Verfügung zu stellen. Dies geschieht über den Menüpunkt „Share“ und bewirkt letztlich die Bereitstellung der fertigen Appliance in der „SUSE Gallery“, die es über eine Suchfunktion ermöglicht, gezielt Appliances mit vorgefertigten Anwendungen herauszusuchen. Eine kleine Beschreibung der gerade erfolgten Zusammenstellung wird hier abgefragt, um anderen zu sagen, was das Image enthält und wer es gebrauchen kann.



Stichwort bitte und ein paar letzte Worte. 🔍

Bereit zum Start

Wie kommt die Appliance nun zur Ausführung? Das ist natürlich abhängig vom erstellten Format. Das ISO-Image einer Live-CD/DVD wird ganz einfach mit einem entsprechenden Brennprogramm (z.B. K3b) auf einen Silberling gebrannt und steht beim nächsten Bootvorgang von der CD zur Verfügung, so wie man es von jeder anderen Live-CD kennt.

Harddisk- und USB-Stick-Images müssen per `diskdump (dd)` auf das Ziellaufwerk kopiert werden. Es ist zu beachten, dass das bestehende Dateisystem auf dem Ziel gänzlich durch das Dateisystem der Appliance ersetzt wird. Es ist also Vorsicht geboten!

gelegt, das alles auch wunderbar auf der GUI erledigt. Da es jedoch nur Dateien mit der Dateiendung IMG akzeptiert, muss man zuvor die RAW-Datei mit der richtigen Dateiendung versehen.

Virtuelle Maschinen für VMware kommen als zwei Dateien. Eine Datei mit der Endung VMX (Definition der virtuellen Maschine) und einer Datei mit der Endung VMDK. Die VMDK-Datei enthält im Prinzip die gesamte virtuelle Maschine.

Beide Dateien lassen sich problemlos im VMware Player importieren und starten. Wer lieber VirtualBox verwendet, der kann diese Dateien auch dort nutzen und starten.

Mittels

```
# dd if=<Dateiname
> of=/dev/sdb
```

wird das RAW-Image z. B. direkt auf den Stick kopiert, wobei darauf zu achten ist, dass man den Namen des USB-Gerätes (hinter `of=`) richtig angibt.

Hinweis: Es ist das Gerät anzugeben, nicht die Partition.

Wer das Image unter Windows schreiben möchte, dem sei das freie GPL-Werkzeug „Win32-DiskImager“ [12] ans Herz

Dazu ist ein kleiner Umweg nötig: Nach dem Start von VirtualBox wählt man zunächst den Menüpunkt „Datei → Manager für virtuelle Medien“ und wählt dort die VMDK-Datei als neue virtuelle Festplatte aus.

Anschließend erzeugt man eine neue virtuelle SUSE-Linux-Maschine, vergibt den gewünschten Hauptspeicher und weist ihr anschließend als Bootfestplatte die gerade festgelegte virtuelle Festplatte zu. Anschließend lässt sich die Appliance wunderbar starten und bootet wie jedes andere virtuelle Betriebssystem im separaten Fenster.

Safety first

Da eine Menge Leute munter ihre Appliances erstellen und mit der Community teilen, stellt sich verständlicherweise die Frage nach der Sicherheit. Kann man der erstellten Software vertrauen und welche Kontrollmechanismen sind vorhanden, um zu vermeiden, dass man sich Malware oder Viren einfängt?

Eine Nachfrage im Forum [13] von *SUSE Studio* brachte folgende Aussagen: „Für den Fall, dass wir auf schädliche Software stoßen, wird diese selbstverständlich sofort entfernt.“

Die Vertrauenswürdigkeit der erstellten Software hängt von der Vertrauenswürdigkeit der verwendeten Pakete ab. Da *SUSE Studio* hauptsächlich Pakete aus wohlbekanntem Repositorys zusammenstellt, ist das Ganze entscheidend von den Maßnahmen abhängig, die von den Verwaltern



dieser Repositorys getroffen werden, um Schadsoftware zu verhindern. Am effektivsten ist zudem die Beobachtung durch die Community. Da viele Leute ein Auge darauf haben, vertraue man zudem darauf, dass schwarze Schafe sofort entdeckt werden.

SUSE Studio weist darauf hin, dass man sich selbstverständlich jedes einzelne Paket ansehen kann, bevor es in die Appliance einfließt. Man kann vor dem Bau also alles selbst überprüfen, bevor man es aufnimmt.

Für Appliances, die auf auf SUSE Enterprise Linux basieren, gibt es eine sogenannte „Supportability-Analyse“, die maßgeblich die Integrität der Quelltexte und Pakete überprüft.

Fazit

SUSE Studio ist eine wunderbar einfache Lösung, um sich seine eigene SUSE-Distribution mit wenigen Mausklicks zusammenzustellen und herunterzuladen.

Die Menüs und Auswahlpunkte sind übersichtlich angeordnet und zum größten Teil selbsterklärend. Es gibt keine quälenden Fragen zur Partitionierung oder zu den Details softwaretechnischer Zusammenstellungen. Auch der Laie kann schnell zu brauchbaren Ergebnissen kommen.

Besonders gelungen ist die Ergänzung durch SUSE Gallery, die auch diese Arbeit einspart, in dem man ganz einfach auf die Früchte anderer zurückgreift und bereits bestehende Appliances nutzt. Wer z. B. einen fertigen Webserver sucht

oder einen Android-Entwickler-Desktop braucht, der wird hier garantiert fündig werden.

Novell hat gerade frisch angekündigt, SUSE Gallery nun der Allgemeinheit zu öffnen, der Startschuss ist also gefallen.

Wo Licht ist, ist jedoch auch Schatten. Die Tests von *SUSE Studio* waren nicht ganz fehlerfrei. So funktioniert die Sprachlokalisation anscheinend nicht, da es nur möglich war, englischsprachige Appliances zu erstellen – trotz Auswahl des deutschen Layouts.

Des Weiteren war es durchaus möglich, eine KDE-Version zu erhalten, die es aus unerfindlichen Gründen nicht schaffte, sich eine DHCP-Adresse für die Netzwerkkarte zu holen und die auch nur über einen eingeschränkten Befehlsatz verfügte. Der Befehl **ping** z. B. war nicht verfügbar und ebenso nicht die Manpages, obwohl sie angeblich Bestandteil der Appliance waren. Im *SUSE Studio*-Forum finden sich weitere Probleme von Anwendern. Allerdings sollte man bedenken, dass es davon immer welche geben wird und sich das gesamte Konzept noch in einer frühen Phase befindet.

Dennoch ist es nie einfacher gewesen, zu einem USB-Stick mit lauffähigem openSUSE zu kommen. Mittlerweile wurde auch die Möglichkeit hinzugefügt, openSUSE 11.3-Appliances zu generieren und damit absolut auf der Höhe der Zeit zu sein.

Insgesamt ist das Team *SUSE Studio*/SUSE Gallery eine tolle Einrichtung, die Spaß macht

und Anreiz gibt, sich mit den verschiedenen Varianten zu befassen.

LINKS

- [1] <http://susestudio.com/>
- [2] <http://www.virtualbox.org/>
- [3] <http://www.vmware.com/>
- [4] <http://www.xen.org/>
- [5] <http://www.susegallery.com/>
- [6] <http://www.novell.com/>
- [7] <http://www.googlemail.com/>
- [8] <https://login.yahoo.com/>
- [9] <http://www.gnome.org/>
- [10] <http://www.kde.org/>
- [11] <http://www.ingres.com/>
- [12] <https://launchpad.net/win32-image-writer/>
- [13] <http://suse-studio-users.1598176.n2.nabble.com/Getting-Malware-and-Viruses-by-using-appliances-td5371406.html#a5372682>

Autoreninformation

Oliver Johandrees ist langjähriger Freund und Anwender von Linux und Gründer der [Linux User Group Erwitte](#). Seine Hauptdistribution ist seit jeher openSUSE, er nutzt aber auch Ubuntu für Laptop und Netbook.

[Diesen Artikel kommentieren](#)

Notepad++ und Notepad2 – Open-Source-Editoren für Windows von Jochen Schnelle

Unter Windows gibt es mindestens zwei Missstände: Den mitgelieferten Editor Notepad und zu wenig freie Open-Source-Software (fragt man überzeugte Linux-Nutzer, so gibt es noch mehr Missstände – aber das ist ein anderes Thema ...). *Notepad++* [1] und *Notepad2* [2] schaffen zumindest bei diesen beiden Punkten Abhilfe.

Editoren unter Windows

Microsofts Notepad ist schon lange Bestandteil der Standardinstallation von Windows. Leider ist der Funktionsumfang von Notepad bestenfalls als spartanisch zu bezeichnen, zumindest dann, wenn man die Standardeditoren aus der Linux-Welt wie gedit (GNOME) oder Kate (KDE) kennt. Selbst der einfache Xfce-Editor Mousepad wirkt gegen Notepad gut ausgestattet.

Der Funktionsumfang von Notepad ist minimal und beschränkt sich auf Suchen und Ersetzen sowie automatischen Zeilenumbruch. Es fehlt z. B. der Umgang mit verschiedenen Zeichencodierungen und Zeilenenden. Somit ist es z. B. schon schwierig, eine unter UNIX erzeugte Textdatei mit Notepad korrekt zu öffnen. Ganz zu schweigen von Dingen wie Syntax-Hervorhebung, Umwandeln von Tabulatoren in Leerzeichen, Anzeigen aller (auch nicht-druckbaren) Zeichen usw.

Es gibt unter Windows durchaus sehr leistungsfähige Editoren. So gilt z. B. Ultraedit [3] als hervorragender Editor, selbst im Linux-Lager. Nur ist

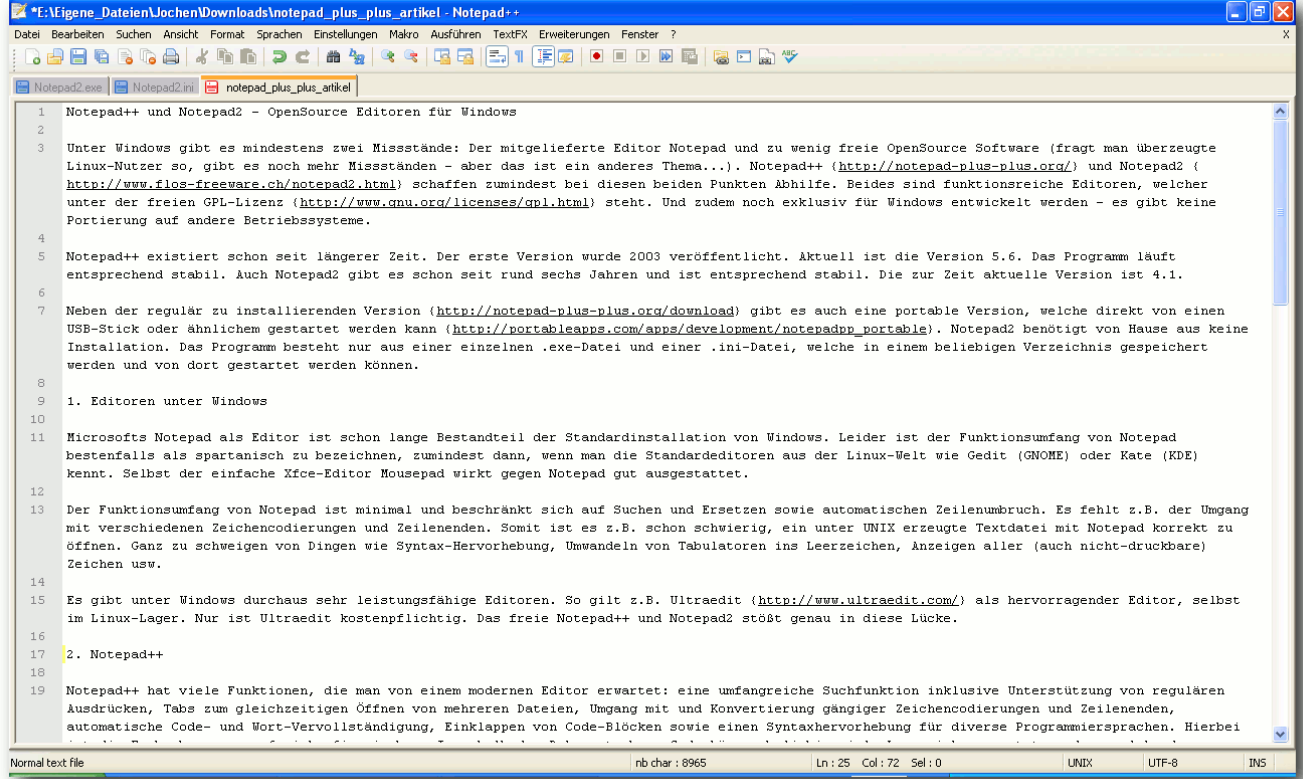
Ultraedit kostenpflichtig. Die freien Editoren *Notepad++* und *Notepad2* stoßen genau in diese Lücke.


Beides sind funktionsreiche Editoren, welche unter der freien GNU General Public License (GPL) [4] stehen – und zudem noch exklusiv für Windows entwickelt werden; es gibt keine Portierung auf andere Betriebssysteme.

Notepad++

Notepad++ existiert schon seit 2003. Aktuell ist die Version 5.6. Das Programm läuft entsprechend stabil.

Neben der regulär zu installierenden Version [5] gibt es auch eine portable Version von *Notepad++*, welche direkt von einem USB-Stick oder ähnlichem gestartet werden kann.



Benutzeroberfläche von Notepad++ 

Diese Version steht auf portableapps.com [6] zum Herunterladen bereit.

Notepad++ hat viele Funktionen, die man von einem modernen Editor erwarten kann und auch findet: eine umfangreiche Suchfunktion inklusive Unterstützung von regulären Ausdrücken, Tabs zum gleichzeitigen Öffnen von mehreren Dateien, Umgang mit und Konvertierung gängiger Zeichenkodierungen und Zeilenenden, automatische Code- und Wortvervollständigung, Einklappen von Code-Blöcken sowie eine Syntaxhervorhebung für diverse Programmiersprachen. Hierbei ist die Farbgebung sogar frei konfigurierbar. Innerhalb des Dokuments bzw. Codes können beliebig viele Lesezeichen gesetzt werden, welche dann schnell angesprungen werden können, was die Navigation innerhalb großer Dokumente erleichtert.

Weiterhin kann *Notepad++* Makros aufzeichnen und ausführen. Dies ist sehr pragmatisch gelöst: Man startet den Makroeditor einfach über die entsprechende Schaltfläche, führt alle Aktionen innerhalb von *Notepad++* aus und beendet die Aufzeichnung. Danach kann man das Makro immer wieder laufen lassen, sodass die aufgezeichneten Funktionen wiederholt werden. *Notepad++* ist in eine Reihe von Sprachen übersetzt, darunter auch in Deutsch.

TextFX

TextFX ist eine Ergänzung von *Notepad++*, welche in der Standardinstallation bereits enthalten ist. Dabei geht es nicht darum, dass der Text mit Effekten ausgestattet wird, sondern dass man

Text schnell und effektiv bearbeiten kann. Die Möglichkeiten von TextFX sind sehr umfangreich, sodass hier nur einige genannt werden können. Dazu gehören unter anderem das Escapen und un-Escapen von Zeichen, das Konvertieren von Zahlen in andere Zahlensysteme und Text nach Hexadezimal, Suchen von zusammengehörenden Klammern, HTML Tidy [7] und vieles mehr. Alle Aktionen sind über einen eigenen Menüpunkt im TextFX-Menü erreichbar.

Erweiterungen

Notepad++ besitzt eine Schnittstelle für Erweiterungen. Installierte Erweiterungen sind über den Menüpunkt „Erweiterungen“ in der Menüleiste erreichbar. In der Grundinstallation sind bereits einige Erweiterungen enthalten. So gibt es z. B. die Möglichkeit, zwei Dateien miteinander am Bildschirm zu vergleichen, wobei dann die Ansicht von *Notepad++* zweigeteilt wird. Eine andere Erweiterung bietet umfangreiche Möglichkeiten zum En- und Dekodieren von Text (z. B. nach Base64).

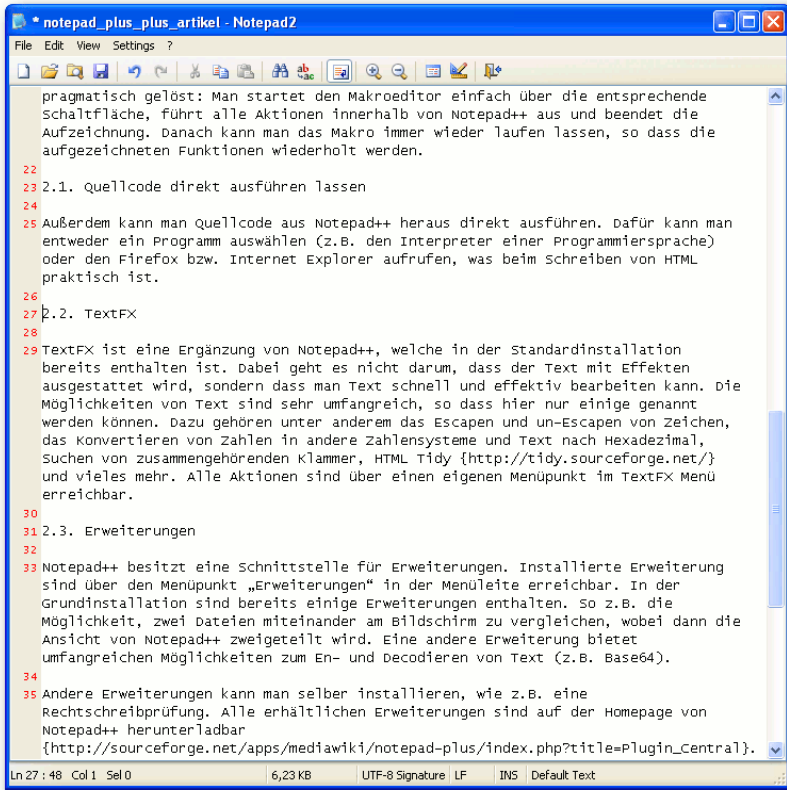
Andere Erweiterungen kann man selbst installieren, wie z. B. eine Rechtschreibprüfung. Alle erhältlichen Erweiterungen sind auf der Homepage von *Notepad++* [8] herunterladbar. Die Installation erfolgt nach erfolgreichem Download von Hand: zuerst ist das zip-Archiv der Erweiterung zu entpacken, dann ist die darin enthaltene DLL-Datei ins Verzeichnis **plugin** innerhalb des *Notepad++*-Verzeichnisses zu kopieren. Enthält das Plugin eine eigene Konfigurationsdatei, so ist diese ins Verzeichnis **plugin\config** zu kopie-

ren. Beim nächsten Start von *Notepad++* ist die neue Erweiterung verfügbar.

Notepad2

Notepad2 gibt es schon seit rund sechs Jahren und ist entsprechend stabil. Die zurzeit aktuelle Version ist 4.1. *Notepad2* benötigt von Haus aus keine Installation. Das Programm besteht nur aus einer einzelnen .exe- und einer .ini-Datei, welche in einem beliebigen Verzeichnis gespeichert werden und von dort gestartet werden können. Leider ist die Oberfläche von *Notepad2* komplett in Englisch und es gibt auch keine deutsche Übersetzung.


Notepad2 bringt alle grundlegenden Funktionen mit, die man von einem Editor erwartet. Dazu gehört der Umgang mit verschiedenen Zeichen-Codierungen und Zeilenenden, wobei *Notepad2* sehr viele Windows-, Iso- und Mac-Codierungen kennt. Das Konvertieren von Codierungen ist ebenfalls möglich. Weiterhin kennt der Editor viele Optionen zum Einstellen der Darstellung, Syntax-Hervorhebung für verschiedene Programmiersprachen und Suchfunktionen, auch mit regulären Ausdrücken. Die Farbgebung für die Syntax-Hervorhebung ist auch bei *Notepad2* frei konfigurierbar. Weiterhin bietet *Notepad2* die Möglichkeit, den Anwender über Änderungen außerhalb vom Editor am aktuellen geöffneten Dokument zu informieren. Dies ist z. B. nützlich, wenn man eine textbasierte Log-Datei geöffnet hat, in die währenddessen ein anderes Programm schreibt.



```

pragmatisch gelöst: Man startet den Makroeditor einfach über die entsprechende
Schaltfläche, führt alle Aktionen innerhalb von Notepad++ aus und beendet die
Aufzeichnung. Danach kann man das Makro immer wieder laufen lassen, so dass die
aufgezeichneten Funktionen wiederholt werden.
22
23 2.1. Quellcode direkt ausführen lassen
24
25 Außerdem kann man Quellcode aus Notepad++ heraus direkt ausführen. Dafür kann man
entweder ein Programm auswählen (z.B. den Interpreter einer Programmiersprache)
oder den Firefox bzw. Internet Explorer aufrufen, was beim Schreiben von HTML
praktisch ist.
26
27 2.2. TextFX
28
29 TextFX ist eine Ergänzung von Notepad++, welche in der Standardinstallation
bereits enthalten ist. Dabei geht es nicht darum, dass der Text mit Effekten
ausgestattet wird, sondern dass man Text schnell und effektiv bearbeiten kann. Die
Möglichkeiten von Text sind sehr umfangreich, so dass hier nur einige genannt
werden können. Dazu gehören unter anderem das Escapen und un-Escapen von Zeichen,
das Konvertieren von Zahlen in andere Zahlensysteme und Text nach Hexadezimal,
Suchen von zusammengehörenden Klammern, HTML Tidy {http://tidy.sourceforge.net/}
und vieles mehr. Alle Aktionen sind über einen eigenen Menüpunkt im TextFX Menü
erreichbar.
30
31 2.3. Erweiterungen
32
33 Notepad++ besitzt eine Schnittstelle für Erweiterungen. Installierte Erweiterung
sind über den Menüpunkt „Erweiterungen“ in der Menüleiste erreichbar. In der
Grundinstallation sind bereits einige Erweiterungen enthalten. So z.B. die
Möglichkeit, zwei Dateien miteinander am Bildschirm zu vergleichen, wobei dann die
Ansicht von Notepad++ zweigeteilt wird. Eine andere Erweiterung bietet
umfangreiche Möglichkeiten zum En- und Decodieren von Text (z.B. Base64).
34
35 Andere Erweiterungen kann man selber installieren, wie z.B. eine
Rechtschreibprüfung. Alle erhältlichen Erweiterungen sind auf der Homepage von
Notepad++ herunterladbar
{http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=Plugin_Central}.
Ln 27 : 48 Col 1 Sel 0 6,23 KB UTF-8 Signature LF INS Default Text

```

Benutzeroberfläche von Notepad2. 

Konfiguration

Über die Menüs von *Notepad2* lassen sich zwar alle Änderungen an Voreinstellung und Aussehen vollziehen, wer diese aber dauerhaft speichern will, sollte dies in der Datei **notepad2.ini** machen. Dies ist, im besten Unix-Stil, eine einfache Textdatei, in der alle Optionen und Einstellmöglichkeiten zeilenweise aufgelistet sind. Die Datei kann direkt mit einem Editor – wie *Notepad2* – bearbeitet werden.

Programme starten

Notepad2 bietet die Möglichkeit, Quelltext direkt aus dem Editor heraus auszuführen. Dazu wählt man im dem Menü „*File* → *Launch* → *Execute Document*“ aus. Kann *Notepad2* anhand der Dateierdung das Dokument identifizieren, so wird direkt der passende Compiler bzw. Interpreter aufgerufen. Ansonsten fragt der Editor, mit welchem Programm der Quelltext ausgeführt werden soll.

Zusammenfassung

Notepad++ und *Notepad2* sind zwei gelungene Editoren für Windows, welche unter einer freien Open-Source-Lizenz steht.

Notepad++ deckt alle Funktionen, angefangen von der Basis über fortgeschrittene Funktionen bis hin zu speziellen Erweiterungen ab. Daher

eignet sich *Notepad++* nicht nur zum Editieren, sondern auch für kleine oder mittelgroße Programmierprojekte.

Notepad2 ist im direkten Vergleich zu *Notepad++* weniger umfangreich ausgestattet, dafür aber auch schlanker. *Notepad2* bietet aber trotzdem alle Funktionen, die man zum schnellen Editieren von Dateien und Schreiben von kurzen Skripten und Programmen benötigt.

Welcher Editor der richtige ist, muss letztendlich jeder Nutzer nach persönlichen Vorlieben und Ansprüchen entscheiden. Leistungsfähiger und universeller als das mit Microsoft Windows mitgelieferte Notepad sind beide Programme allemal.

LINKS

- [1] <http://notepad-plus-plus.org/> 
- [2] <http://www.flos-freeware.ch/notepad2.html> 
- [3] <http://www.ultraedit.com/> 
- [4] <http://www.gnu.org/licenses/gpl.html> 
- [5] <http://notepad-plus-plus.org/download> 
- [6] http://portableapps.com/apps/development/notepadpp_portable 
- [7] <http://tidy.sourceforge.net/> 
- [8] http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=Plugin_Central 

Autoreninformation

Jochen Schnelle, üblicherweise Ubuntu-Nutzer, muss beruflich Windows nutzen. Als Editor zum schnellen Editieren von Textdatei und Schreiben von kleinen (Python-)Skripten kommt dabei Notepad++ zum Einsatz.

Diesen Artikel kommentieren 

Rezension: Python – Essential Reference (Fourth Edition) von Jochen Schnelle

Das englischsprachige Buch „Python – Essential Reference“ bietet auf etwas mehr als 700 Seiten einen fundierten Einblick in die Programmiersprache Python. Das Buch ist inzwischen in der vierten, aktualisierten und erweiterten Version erschienen. Erklärtes Ziel des Buchs ist es, dem Leser sowohl die Grundideen und grundlegenden Eigenschaften von Python näher zu bringen, als auch einen Überblick über eine ganze Reihe von Modulen zu geben, welche standardmäßig in Python enthalten sind.

Die Zielgruppe des Buches ist relativ weit gefasst. Sowohl ambitionierte Einsteiger mit grundlegenden Python-Kenntnissen als Quereinsteiger mit Kenntnissen in anderen Programmiersprachen, aber auch fortgeschrittene Python-Nutzer werden angesprochen.

Das Buch beginnt mit einem rund zwanzigseitigen Python-Tutorial, in dem alle charakteristischen Datentypen, Funktionen, Module etc. von Python kurz dargestellt werden. In den folgenden Kapiteln werden diese dann nochmals ausführlich vorgestellt.

Bekanntlich ist Python zwar durch und durch objektorientiert aufgebaut, erlaubt aber verschiedene Programmierstile. Daher sind Funktionen und funktionaler Programmierung sowie objektorientierter Programmierung und Klassen jeweils ein eigenes Kapitel gewidmet.

Der erste Teil wird mit Kapiteln zur Modulerstellung, den Grundlagen der Ein- und Ausgabe sowie dem Testen, dem Debugging und der Optimierung abgeschlossen.

Im zweiten Teil des Buches, der etwa zwei Drittel des Gesamtumfangs ausmacht, werden verschiedenste Module vorgestellt, welche standardmäßig in Python enthalten sind. Das Spektrum ist dabei sehr weit gefasst und deckt viele Bereiche der Programmierung ab. Recht ausführlich werden systemnahe Module vorgestellt; weiterhin findet man Informationen unter anderem zu Threads, dem Umgang mit Dateien, der Python-Datenbank-API, Internet-Applikationen (HTTP, FTP, SMTP usw.), Serveranwendungen (CGI, WSGI), mathematischen Modulen und vielem mehr.

Der dritte und schließlich letzte Teil, der vom Umfang her auch der kleinste ist, wendet sich dann auch an den fortgeschrittenen Anwender. Hier geht es darum, wie man Python unter Verwendung der Programmiersprache C erweitern kann sowie darum, wie man den Python-Interpreter einbetten kann.

Die einzelnen Kapitel des Buches sind sinnvoll nach Themen zusammengefasst, bauen aber nicht aufeinander auf. Wen z. B. das Thema „Netzwerkprogrammierung und Sockets“ nicht interessiert, der kann diese Kapitel einfach überspringen und im nächsten weiterlesen. Alle Kapitel sind in sich abgeschlossen, so dass

Buchinformationen

| | |
|---------------|---|
| Titel | Python – Essential Reference (Fourth Edition) |
| Autor | David M. Beazley |
| Verlag | Addison Wesley, 2009 |
| Umfang | 716 Seiten |
| ISBN | 978-0672329784 |
| Preis | 29,95 Euro |

man diese grundsätzlich auch in einer beliebigen Reihenfolge lesen kann. Dadurch eignet sich das Buch sehr gut als Nachschlagewerk.

Das Buch bietet eine enorme Materialfülle. Beim Lesen hat man nie das Gefühl, dass etwas fehlt und Fragen offen bleiben. Dies liegt sicherlich auch daran, dass sehr viele Beispiele enthalten sind. Diese sind zwar zumeist recht kurz, wirken aber nie theoretisch oder konstruiert, sondern zeigen immer Praxisrelevanz. Die Beispiele bringen das zuvor vorgestellte Thema bzw. die Funktionen eines Moduls immer verständlich auf den Punkt.

Auch wenn das Buch nicht den gesamten Umfang der Python-Module abdeckt – was allein aufgrund ihrer Zahl nicht möglich ist – hat man beim Lesen nie den Eindruck, dass entscheidende Informationen fehlen. Das Englisch des Buches ist sachlich und stets gut und flüssig zu lesen. Sofern man eine Grundidee von Python hat und sich dazu bereit fühlt, ein englisches Buch zu lesen, sollte es keine Verständnisprobleme geben.



Weiterhin ist positiv anzumerken, dass das Buch Python 2.6 und Python 3.0 (welche zum Zeitpunkt der Überarbeitung der hier vorgestellten „4th Edition“ aktuell waren) absolut gleichwertig behandelt. Das heißt, dass alle Module, Funktionen etc. für beide Python-Versionen erklärt werden, sofern es Unterschiede gibt. Alle im Buch verwendeten Beispiele sind gleichermaßen unter Python 2.6 und 3.0 lauffähig. Im Anhang des Buches gibt es zusätzlich einen knapp zwanzigseitigen Teil, welcher die essentiellen Neuerungen von Python 3 darstellt. Hier findet man auch Hinweise bzw. Tipps vom Autor, wer wann Python 3 (oder Python 2.x) verwenden sollte.

Beim Lesen des Buchs fällt auf, dass der Autor über eine weitreichende Programmiererfahrung (mit Python) verfügt. An vielen Stellen findet

man Hinweise, wie man bestimmte Probleme am effizientesten löst, warum ein bestimmter Code schneller ist als andere Lösungen oder warum das gerade vorgestellte Modul für eine bestimmte Anwendung besser geeignet ist als ein anderes. Hier können sicherlich auch fortgeschrittene Python-Programmierer noch den einen oder anderen Tipp mitnehmen.

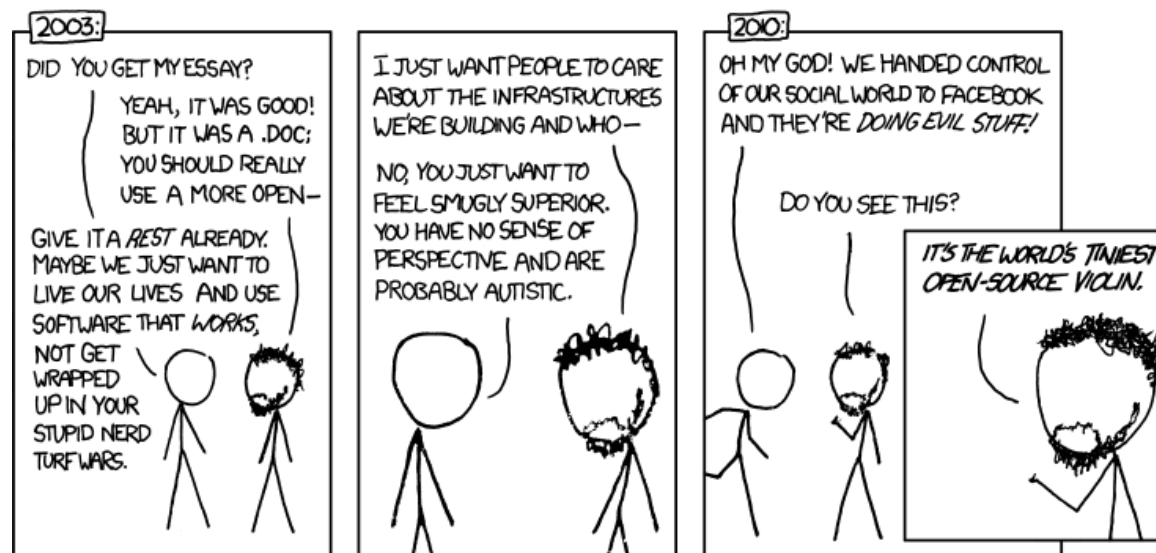
Zusammenfassend kann man sagen, dass das Buch eine Empfehlung für jeden ist, der sich für Python ernsthaft interessiert bzw. in Python programmiert – egal, ob Python 2.6 oder Python 3. Das Buch bildet ein essentielles Nachschlagewerk zu vielen Python-Modulen. Diese sind zwar auch in der offiziellen Python-Dokumentation beschrieben, jedoch ist das Buch an vielen Stellen ausführlicher, bietet kompakte, verständliche und

praxisnahe Beispiele und gibt obendrein noch Tipps für performanteren Python-Code.

Autoreninformation

Jochen Schnelle verwendet Python also Hauptprogrammiersprache für diverse Werkzeuge für den eigenen Gebrauch sowie für verschiedene Intranet-Applikationen, die sein Arbeitsleben (und das der Kollegen) leichter machen.

Diesen Artikel kommentieren



„Infrastructures“ © by Randall Munroe (CC-BY-NC-2.5), <http://xkcd.com/743>

Veranstungskalender

Messen

| Veranstaltung | Ort | Datum | Eintritt | Link |
|-------------------------------------|------------|----------------|----------|---|
| FrOSCamp | Zürich | 17.-18.09.2010 | frei | http://frosccamp.org |
| FUDCon | Zürich | 17.-19.09.2010 | frei | https://fedoraproject.org/wiki/FUDCon:Zurich_2010 |
| Software Freedom Day | Weltweit | 18.09.2010 | frei | http://softwarefreedomday.org |
| Kieler Linxstage | Kiel | 01.-02.10.2010 | frei | http://www.kieler-linxstage.de |
| Ubucon | Leipzig | 15.-17.10.2010 | 10 EUR | http://www.ubucon.de |
| TechTalk: Mobile Anwendungen mit Qt | München | 19.10.2010 | frei | http://www.opensourceschool.de |
| Brandenburger Linux-Infotag | Potsdam | 06.11.2010 | frei | http://blit.org/2010 |
| OpenRheinRuhr | Oberhausen | 13.-14.11.2010 | 3 EUR | http://www.openrheinruhr.de |
| Open Source Expo | Karlsruhe | 15.-16.11.2010 | 10 EUR | http://openexpo.de |
| 12. LinuxDay | Dornbirn | 17.11.2010 | frei | http://linuxday.at |

(Alle Angaben ohne Gewähr!)

Sie kennen eine Linux-Messe, welche noch nicht auf der Liste zu finden ist? Dann schreiben Sie eine E-Mail mit den Informationen zu Datum und Ort an redaktion@freiesMagazin.de.

Konventionen

An einigen Stellen benutzen wir Sonderzeichen mit einer bestimmten Bedeutung. Diese sind hier zusammengefasst:

- \$:** Shell-Prompt
- #:** Prompt einer Root-Shell – Ubuntu-Nutzer können hier auch einfach in einer normalen Shell ein **sudo** vor die Befehle setzen.
- ↵:** Kennzeichnet einen aus satztechnischen Gründen eingefügten Zeilenumbruch, der nicht eingegeben werden soll.
- ~:** Abkürzung für das eigene Benutzerverzeichnis **/home/BENUTZERNAME**
- 🇬🇧:** Kennzeichnet einen Link, der auf eine englischsprachige Seite führt.
- 🔍:** Öffnet eine höher aufgelöste Version der Abbildung in einem Browserfenster.

Impressum

freiesMagazin erscheint als PDF und HTML einmal monatlich.
Redaktionsschluss für die August-Ausgabe: 19. September 2010

Kontakt

E-Mail redaktion@freiesMagazin.de
Postanschrift **freiesMagazin**
c/o Dominik Wagenführ
Beethovenstr. 9/1
71277 Rutesheim
Webpräsenz <http://www.freiesmagazin.de/>

Autoren dieser Ausgabe

Ralf Damaschke **S.14**
Christian Imhorst **S.5**
Oliver Johandrees **S.29**
Mathias Menzer **S.3**
Jochen Schnelle **S.36, S.39**
Michael Schwarz **S.25**
Dominik Wagenführ **S.10**

Logo-Design

Arne Weinberg (GNU FDL)

ISSN 1867-7991

Erscheinungsdatum: 5. September 2010

Redaktion

Dominik Honnef Thorsten Schmidt
Dominik Wagenführ (Verantwortlicher Redakteur)

Satz und Layout

Ralf Damaschke Yannic Haupenthal
Nico Maikowski Matthias Sitte
Günther Wutz

Korrektur

Daniel Braun Frank Brungräber
Stefan Fangmeier Mathias Menzer
Karsten Schuldt Franz Seidl
Stephan Walter

Veranstaltungen

Ronny Fischer

Dieses Magazin wurde mit \LaTeX erstellt. Mit vollem Namen gekennzeichnete Beiträge geben nicht notwendigerweise die Meinung der Redaktion wieder. Wenn Sie freiesMagazin ausdrucken möchten, dann denken Sie bitte an die Umwelt und drucken Sie nur im Notfall. Die Bäume werden es Ihnen danken. ;-)

Soweit nicht anders angegeben, stehen alle Artikel und Beiträge in freiesMagazin unter der GNU-Lizenz für freie Dokumentation (FDL). Das Copyright liegt beim jeweiligen Autor. freiesMagazin unterliegt als Gesamtwerk ebenso der GNU-Lizenz für freie Dokumentation (FDL) mit Ausnahme von Beiträgen, die unter einer anderen Lizenz hierin veröffentlicht werden. Das Copyright liegt bei Dominik Wagenführ. Es wird die Erlaubnis gewährt, das Werk/die Werke (ohne unveränderliche Abschnitte, ohne vordere und ohne hintere Umschlagtexte) unter den Bestimmungen der GNU Free Documentation License, Version 1.2 oder jeder späteren Version, veröffentlicht von der Free Software Foundation, zu kopieren, zu verteilen und/oder zu modifizieren. Die xkcd-Comics stehen separat unter der Creative-Commons-Lizenz CC-BY-NC 2.5. Das Copyright liegt bei Randall Munroe.