Fachhochschule Wedel
Studiengang Medieninformatik

Eine Einführung in die prozedurale Landschaftsgenerierung

Seminararbeit

Tjark Smalla
Matrikel-Nummer 100554

Betreuer Prof. Bohn

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Allgemeine Symbole

Symbol	Bedeutung
a	der Skalar a
$\vec{x}$	der Vektor $\vec{x}$
$\mathbf{A}$	die Matrix $\mathbf{A}$

1 Einleitung

Dieses Beispieldokument ist von `http://www.bretschneidernet.de/tips/thesislatex.html` heruntergeladen worden.

Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst? Er presste sich ganz eng an die Wand hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang. Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden?

2 Noise

Synthetisch erzeugtes Rauschen *(engl. Noise)* erweist sich als hilfreiches Mittel zur Erzeugung von zufällig erscheinenden Strukturen. Als wohl bekannteste Implementierung ist hier die Implementierung von Ken Perlin[K85] zur Erzeugung einer Marmortextur auf einer Vase zu nennen[1].

Neben umfangreichen Anpassungsmöglichkeiten durch verschiedene Parameter ist die Performance dieses Verfahrens ein entscheidender Grund für die Nutzung. Noise verbraucht extrem wenig Speicher, ist relativ einfach zu berechnen und ist zu jeder Zeit an einer beliebigen Stelle auswertbar, was es auch für Echtzeitanwendungen geeignet macht.[HH]

Dieses Kapitel soll ein grundlegendes Verständnis über Noise-Funktionen bieten. Dazu werden zuerst grundlegende Komponenten, welche jeder Implementierung zugrunde liegen, erläutert. Anschließend werden *Value*-2.2, *Gradient-Noise*2.3 sowie *Fractal-Noise*2.4 erklärt, bevor es einen Ausblick auf den *Simplex-Noise*2.5 Algorithmus gibt.

Weitere ausführliche Beschreibung in [Bur08] (Gradient-Noise) und in [Gus05] (Simplex-Noise).

2.1 Grundlagen

2.1.1 Lattice-Function

Der erste Schritt zur Erzeugung von Noise ist in der Regel eine sogenannte *Noise-Lattice(Rausch-Gitter)-Funktion*[AC] der Form $l(\vec{k}) : Z^n \mapsto [-1 - 1]$. Diese dient zur Beschreibung eines

1 Auch als *Perlin-Noise* bezeichnete Implementierung von Gradient Noise in 3-D

Gitters, welches die Form unserer zukünftigen Noise-Funktion bestimmen wird. Die Funktion muss dabei unbedingt deterministisch sein[2].

Die Wahl der Lattice Funktion ist entscheident für das spätere Erscheinungsbild der Noise-Funktion. Eine gleichverteilte Folge von pseudo zufällige Zahlen wie sie etwa die meisten in Programmiersprachen implementierten Zufallsgeneratoren bieten erfüllt zwar die Anforderungen der Deterministik der Funktion, kann allerdings zu unerwünscht starken Differenzen zwischen zwei Benachbarten Gitterpunkten führen. Im folgenden gehen wir von einer Standartnormalverteilung aus um die Wahrscheinlichkeit für Werte nahe den Intervalgrenzen zu verringern.

2.1.2 Interpolation und Fade-Function

Um aufbauend auf der Lattice-Funktion2.1.1 eine Funktion $S(\vec{x}) : \mathbb{R}^n \mapsto \mathbb{R}, \vec{x} \in \mathbb{Z}^n$ zu definieren wird zwischen benachbarten Gitterpunkten lokal interpoliert. Dafür wird eine sogenannte Fade-Function[Per] der Form $f(t) : \mathbb{R} \mapsto \mathbb{R}$ mit $t \in [0, 1]$ definiert, welche den Übergang zwischen den Gitterpunkten steuert.

Um überhaupt eine stetige Noise-Funktion zu ermöglichen, muss

$$f(0) = 0 \wedge f(1) = 1 \tag{2.1}$$

gelten. Damit der Übergang zwischen den Gitterpunkten möglichst glatt und damit natürlich wirkt, sollte jedoch eine Stetigkeit von C^2 und damit die Eigenschaften

$$f'(0) = f'(1) = 0 = f''(0) = f''(1) \tag{2.2}$$

gelten.

Dafür wird im folgenden das Polynom $f(t) = 6t^5 - 15t^4 + 10t^3$ benutzt, welches auch in Perlins Referenzimplementierung Verwendung findet[Bur08] und alle Eigenschaften erfüllt.

Die mit $f(t)$ gebildete, interpolierende Funktion $S(\vec{x}) = noise(\vec{x})$ definiert nun - im einfachsten Fall - die Noise-Funktion.

2 Siehe2.4, sie muss also für jede Koordinate eines Gitterpunktes immer denselben Funktionswert liefern.

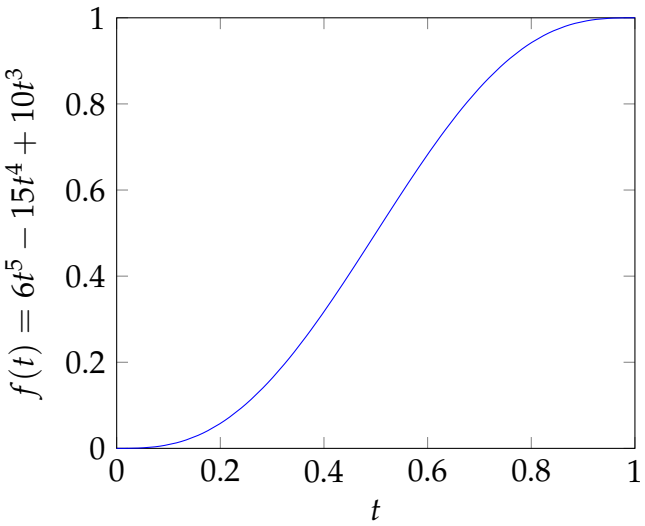


Abbildung 2.1.1: Fade-Function

2.2 Value-Noise

Value-Noise ist die wohl naivste Implementierung einer Noise-Funktion. Bei ihr werden die Gitterpunktwerte, welche durch die *Lattice-Funktion*2.1.1 erzeugt wurden, als Höhenwerte interpretiert.

Untenstehend ist eine Implementierung in C# für eine 2-Dimensionale Rauschfunktion zu sehen. Noise lässt sich problemlos in mehrere Dimensionen skalieren. Einzig die Interpolation der Werte muss hier angepasst werden. In der Implementierung ist zu sehen, wie die Gitterpunktwerte - ähnlich einer *billinearen Interpolation* - mit der *Fade-Function* interpoliert werden.

Value-Noise Implementierung C#

```
1  public float S1(float x, float y) {
2      int floorX = Mathf.FloorToInt(x),
3          ceilX = Mathf.CeilToInt(x),
4          floorY = Mathf.FloorToInt(y),
5          ceilY = Mathf.CeilToInt(y);
6      float tx = x - floorX,
7            ty = y - floorY;
8
9      return FadeFunction(1 - ty) *
10             (LatticeFunc(floorX, floorY) * FadeFunction(1 - tx) + FadeFunction(tx) * LatticeFunc(ceilX, floorY))
11             +
12             FadeFunction(ty) *
13             (LatticeFunc(floorX, ceilY)*FadeFunction(1 - tx) + FadeFunction(tx)*LatticeFunc(ceilX, ceilY));
14 }
```

2.3 Gradient-Noise

Der vorher erwähnte Value-Noise kann, je nach Parameterwahl, noch ein unruhiges Rauschen erzeugen. Um die Übergänge zwischen den Gitterpunktwerten noch sanfter und damit natürlicher aussehen zu lassen wurde der Gradient-Noise erfunden. Hier werden die Gitterpunktwerte nicht als Höhenwerte, Gradienten an den Nullstellen der Noise-Funktion gesehen. Es ergibt sich also:

$$noise(\vec{k}) = 0 \wedge noise'(\vec{k}) = S(\vec{k}), k \in \mathbb{Z}^n. \tag{2.3}$$

In der untenstehenden 2D C# Implementierung ist zu sehen, wie zuerst ein Höhenwert für den aktuellen Punkt $\vec{x}$ über das Skalarprodukt[3] zwischen dem Gradienten und der relativen Position $\begin{pmatrix} tx \\ ty \end{pmatrix}$ $tx, ty \in [0-1]$ und anschließend über die bekannte Interpolation berechnet wird.

Gradient-Noise Implementierung C#

```
1  public float S(float x, float y) {
2      int floorX = Mathf.FloorToInt(x),
3          ceilX = Mathf.CeilToInt(x),
4          floorY = Mathf.FloorToInt(y),
5          ceilY = Mathf.CeilToInt(y);
6      float tx = x - floorX,
7            ty = y - floorY,
8            //Calc slopes
9            n00 = Vector2.Dot(LatticeFunc(floorX, floorY), new Vector2(tx, ty)),
10           n10 = Vector2.Dot(LatticeFunc(ceilX, floorY), new Vector2(tx-1, ty)),
11           n01 = Vector2.Dot(LatticeFunc(floorX, ceilY), new Vector2(tx, ty-1)),
12           n11 = Vector2.Dot(LatticeFunc(ceilX, ceilY), new Vector2(tx - 1, ty - 1));
13
14     return FadeFunction(1 - ty) *
15            (n00 * FadeFunction(1 - tx) + n10 * FadeFunction(tx))
16            +
17            FadeFunction(ty) *
18            (n01 * FadeFunction(1 - tx) + n11 * FadeFunction(tx));
19 }
```

2.4 Fractal-Noise

Die bisher behandelten Noise-Funktionen erzeugen zwar natürlich erscheinende Zufallswerte, wenn man diese jedoch auf eine Heightmap überträgt wird deutlich, dass es ihr an Details fehlt um natürlich zu wirken.

3 engl. Dot-Product

Um den Detailgrad der Noise-Funktion beliebig zu erhöhen, wird die Noise-Funktion mit einer gestauchten, in der Amplitude verringerten, Version ihrer selbst addiert. Dieses Verfahren lässt sich beliebig oft anwenden, was einen beliebig hohen Detailgrad erlaubt.

In [Sau88] wird daher folgende Formel definiert:

$$\mathbb{H}(\vec{x}) = \sum_{k=k_0}^{k_1} \frac{1}{r^{kH}} S(r^k \vec{x}). \tag{2.4}$$

Wobei $H = 2 - D$ der Hurst-Exponent ist[JS06] und $D = -\frac{-log(k1)}{log(r)}$[fra] die fraktale Dimension.

2.5 Ausblick: Simplex-Noise

3 weiteres Kapitel

In diesem Kapitel wird einiges gemacht[1] Vor allem in Unterabschnitt 3.1.1 wird einiges gezeigt, was noch nie jemand gesehen hat. Es lohnt sich also, dranzubleiben.

3.1 eine Sektion

Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? TODO: das muss ich noch verfeinern, weil ich erst zur Hälfte verstanden habe Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst? Er presste sich ganz eng an die Wand hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang.

1 wobei einiges nicht vieles heißt, ich möchte hier also keine falschen Hoffnungen wecken.

Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden?

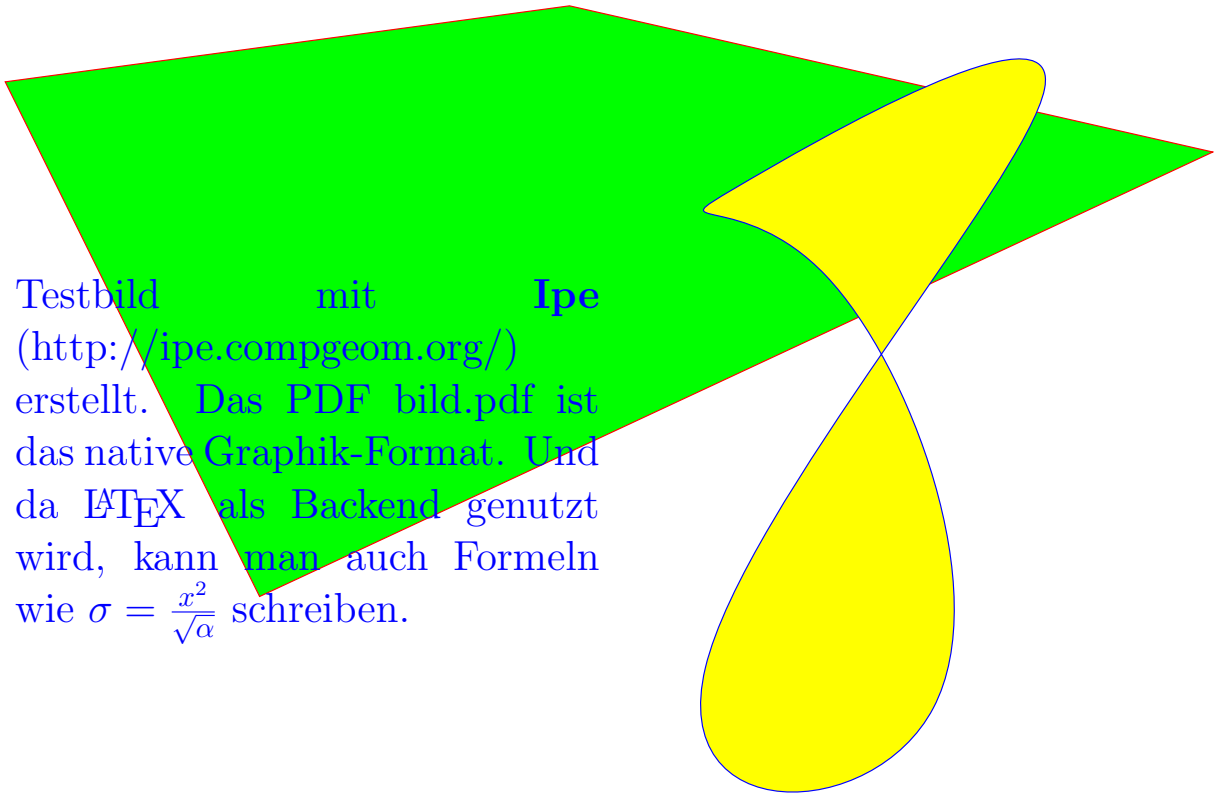


Abbildung 3.1.1: Test-Bild mit langer Bildunterschrift

Die Gleichung 3.1

$$a^2 + b^2 = c^2 \tag{3.1}$$

ist allseits bekannt und bedarf wohl keiner weiteren Erläuterung.

Auch nicht schlecht ist Abbildung 3.1.1. Aber überhaupt keinen Sinn macht Tabelle 3.1. Hieran sieht man den Vorteil des autoref-Befehls und das so Links erstellt werden.

3.1.1 jetzt geht es noch tiefer

Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten

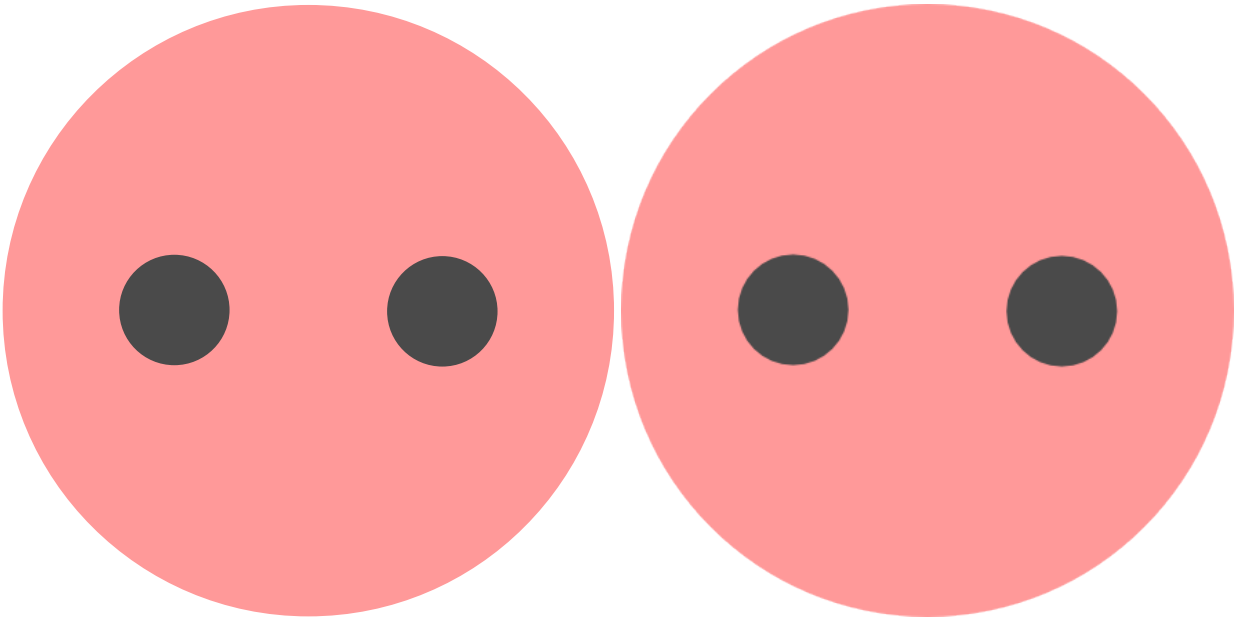
Formen	Städte
Quadrat	Bunkenstedt
Dreieck	Laggenbeck
Kreis	Peine
Raute	Wakaluba

Tabelle 3.1: eine sinnlose Tabelle

Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst? Er presste sich ganz eng an die Wand hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang. Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden?

Auch können Bilder in Bildern direkt angesprochen werden: Abbildung 3.1.2a und Abbildung 3.1.2b.

Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu



(a) Ein Bild im PDF mit einer Größe von nur 1,1 kB

(b) Das gleiche Bild als optimierte PNG-Datei mit einer Größe von 8,9 kB

Abbildung 3.1.2: Zwei Bilder werden mit dem LaTeX-Paket subcaption nebeneinander angezeigt

erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um.

- Erstens ist das soundso,
- dann darf man natürlich nicht vergessen und
- das ist auch noch wichtig.

Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst? Er presste sich ganz eng an die Wand

hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang. Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden?

Komplexe Tabellen sind nicht sehr einfach:

		dies			
		von dort	und dort	über hier	zu Los
das	hier	bla	bla	bla	bla
	dort	bla	bla	bla	bla
	da	bla	bla	bla	bla

Tabelle 3.2: eine kompliziertere Tabelle mit viel Beschreibungstext, der aber nicht im Tabellenverzeichnis auftauschen soll

Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst? Er presste sich ganz eng an die Wand hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang. Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam

bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden?

4 Zusammenfassung

Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst? Er presste sich ganz eng an die Wand hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang. Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden?

A Anhang

A.1 Quelltexte

cpu.c aus Linux 2.6.16

```
/* CPU control.
 * (C) 2001, 2002, 2003, 2004 Rusty Russell
 *
 * This code is licenced under the GPL.
 */
#include <linux/proc_fs.h>
#include <linux/smp.h>
#include <linux/init.h>
#include <linux/notifier.h>
#include <linux/sched.h>
#include <linux/unistd.h>
#include <linux/cpu.h>
#include <linux/module.h>
#include <linux/kthread.h>
#include <linux/stop_machine.h>
#include <asm/semaphore.h>

/* This protects CPUs going up and down... */
static DECLARE_MUTEX(cpucontrol);

static struct notifier_block *cpu_chain;

#ifdef CONFIG_HOTPLUG_CPU
static struct task_struct *lock_cpu_hotplug_owner;
static int lock_cpu_hotplug_depth;

static int __lock_cpu_hotplug(int interruptible)
{
  int ret = 0;

  if (lock_cpu_hotplug_owner != current) {
    if (interruptible)
      ret = down_interruptible(&cpucontrol);
    else
      down(&cpucontrol);
  }

  /*
   * Set only if we succeed in locking
   */
  if (!ret) {
    lock_cpu_hotplug_depth++;
    lock_cpu_hotplug_owner = current;
  }

  return ret;
}

void lock_cpu_hotplug(void)
{
  __lock_cpu_hotplug(0);
}
EXPORT_SYMBOL_GPL(lock_cpu_hotplug);

void unlock_cpu_hotplug(void)
{
  if (--lock_cpu_hotplug_depth == 0) {
    lock_cpu_hotplug_owner = NULL;
    up(&cpucontrol);
  }
}
EXPORT_SYMBOL_GPL(unlock_cpu_hotplug);

int lock_cpu_hotplug_interruptible(void)
{
  return __lock_cpu_hotplug(1);
}
EXPORT_SYMBOL_GPL(lock_cpu_hotplug_interruptible);
#endif  /* CONFIG_HOTPLUG_CPU */

/* Need to know about CPUs going up/down? */
int register_cpu_notifier(struct notifier_block *nb)
{
  int ret;

  if ((ret = lock_cpu_hotplug_interruptible()) != 0)
    return ret;
  ret = notifier_chain_register(&cpu_chain, nb);
  unlock_cpu_hotplug();
  return ret;
```

```
}
EXPORT_SYMBOL(register_cpu_notifier);

void unregister_cpu_notifier(struct notifier_block *nb)
{
  lock_cpu_hotplug();
  notifier_chain_unregister(&cpu_chain, nb);
  unlock_cpu_hotplug();
}
EXPORT_SYMBOL(unregister_cpu_notifier);

#ifdef CONFIG_HOTPLUG_CPU
static inline void check_for_tasks(int cpu)
{
  struct task_struct *p;

  write_lock_irq(&tasklist_lock);
  for_each_process(p) {
    if (task_cpu(p) == cpu &&
        (!cputime_eq(p->utime, cputime_zero) ||
         !cputime_eq(p->stime, cputime_zero)))
      printk(KERN_WARNING "Task %s (pid = %d) is on cpu %d\
 (state = %ld, flags = %lx)\n",
         p->comm, p->pid, cpu, p->state, p->flags);
  }
  write_unlock_irq(&tasklist_lock);
}

/* Take this CPU down. */
static int take_cpu_down(void *unused)
{
  int err;

  /* Ensure this CPU doesn't handle any more interrupts. */
  err = __cpu_disable();
  if (err < 0)
    return err;

  /* Force idle task to run as soon as we yield: it should
     immediately notice cpu is offline and die quickly. */
  sched_idle_next();
  return 0;
}

int cpu_down(unsigned int cpu)
{
  int err;
  struct task_struct *p;
  cpumask_t old_allowed, tmp;

  if ((err = lock_cpu_hotplug_interruptible()) != 0)
    return err;

  if (num_online_cpus() == 1) {
    err = -EBUSY;
    goto out;
  }

  if (!cpu_online(cpu)) {
    err = -EINVAL;
    goto out;
  }

  err = notifier_call_chain(&cpu_chain, CPU_DOWN_PREPARE,
          (void *)(long)cpu);
  if (err == NOTIFY_BAD) {
    printk("%s: attempt to take down CPU %u failed\n",
        __FUNCTION__, cpu);
    err = -EINVAL;
    goto out;
  }

  /* Ensure that we are not runnable on dying cpu */
  old_allowed = current->cpus_allowed;
  tmp = CPU_MASK_ALL;
  cpu_clear(cpu, tmp);
  set_cpus_allowed(current, tmp);

  p = __stop_machine_run(take_cpu_down, NULL, cpu);
  if (IS_ERR(p)) {
    /* CPU didn't die: tell everyone.  Can't complain. */
    if (notifier_call_chain(&cpu_chain, CPU_DOWN_FAILED,
        (void *)(long)cpu) == NOTIFY_BAD)
      BUG();

    err = PTR_ERR(p);
    goto out_allowed;
  }

  if (cpu_online(cpu))
    goto out_thread;

  /* Wait for it to sleep (leaving idle task). */
  while (!idle_cpu(cpu))
    yield();

  /* This actually kills the CPU. */
  __cpu_die(cpu);

  /* Move it here so it can run. */
  kthread_bind(p, get_cpu());
  put_cpu();

  /* CPU is completely dead: tell everyone.  Too late to complain. */
  if (notifier_call_chain(&cpu_chain, CPU_DEAD, (void *)(long)cpu)
      == NOTIFY_BAD)
```

```
187       BUG();
188
189     check_for_tasks(cpu);
190
191   out_thread:
192     err = kthread_stop(p);
193   out_allowed:
194     set_cpus_allowed(current, old_allowed);
195   out:
196     unlock_cpu_hotplug();
197     return err;
198   }
199   #endif /*CONFIG_HOTPLUG_CPU*/
200
201   int __devinit cpu_up(unsigned int cpu)
202   {
203     int ret;
204     void *hcpu = (void *)(long)cpu;
205
206     if ((ret = lock_cpu_hotplug_interruptible()) != 0)
207       return ret;
208
209     if (cpu_online(cpu) || !cpu_present(cpu)) {
210       ret = -EINVAL;
211       goto out;
212     }
213
214     ret = notifier_call_chain(&cpu_chain, CPU_UP_PREPARE, hcpu);
215     if (ret == NOTIFY_BAD) {
216       printk("%s: attempt to bring up CPU %u failed\n",
217           __FUNCTION__, cpu);
218       ret = -EINVAL;
219       goto out_notify;
220     }
221
222     /* Arch-specific enabling code. */
223     ret = __cpu_up(cpu);
224     if (ret != 0)
225       goto out_notify;
226     if (!cpu_online(cpu))
227       BUG();
228
229     /* Now call notifier in preparation. */
230     notifier_call_chain(&cpu_chain, CPU_ONLINE, hcpu);
231
232   out_notify:
233     if (ret != 0)
234       notifier_call_chain(&cpu_chain, CPU_UP_CANCELED, hcpu);
235   out:
236     unlock_cpu_hotplug();
237     return ret;
238   }
```

Literaturverzeichnis

[AC] A.J. Crilly, R.A. Earnshaw H.: *Fractals and Chaos*

[Bur08] Burger, Wilhelm: Gradientenbasierte Rauschfunktionen und Perlin Noise / School of Informatics, Communications and Media, Upper Austria University of Applied Sciences. Version: November 2008. http://staff.fh-hagenberg.at/burger/. Hagenberg, Austria, November 2008 (HGBTR08-02). – Forschungsbericht

[fra] *Fraktale Dimension*. https://www.wikiwand.com/de/Fraktale_Dimension#/.C3.84hnlichkeits-Dimension

[Gus05] Gustavson, Stefan: *Simplex noise demystified*. http://staffwww.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf. Version: 2005

[HH] H. Hauser, E. R.: *State of the Art in Procedural Noise Functions*. https://www-sop.inria.fr/reves/Basilic/2010/LLCDDELPZ10/LLCDDELPZ10STARPNF.pdf

[JS06] Jens Schneider, Rudiger W. Tobias Boldte B. Tobias Boldte: *Real-Time Editing, Synthesis, and Rendering of Infinite Landscapes on GPUs*. http://wwwcg.in.tum.de/fileadmin/user_upload/Lehrstuehle/Lehrstuhl_XV/Research/Publications/2006/vmv06.pdf. Version: 2006

[K85] In: K, PERLIN: *An image synthesizer*. vol. 19. 1985, S. 287– 296

[Per] In: Perlin, K.: *Improving noise*, S. 681–682

[Sau88] Saupe, D: Point evaluation of multi-variable random fractals. In: *Visualisierung in Mathematik und Naturwissenschaft, Bremer Computergraphik Tage*, 1988

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Ort, Datum Unterschrift