# Raytracing mit POV-Ray

Ivo Vollrath
Universität Kaiserslautern
vollrath@informatik.uni-kl.de

14. Juni 1997

Computergrafik unter Linux — Ein Einstieg in die Erzeugung (fast) photorealistischer Bilder und Animationen mit Public Domain Softwarewerkzeugen.

# Gliederung

- Was ist Raytracing?
- POV-Ray — Stärken und Schwächen
- Die POV-Ray Szenenbeschreibungssprache
- Erstes Beispiel
- Komplexere Szenen und weitere Hilfsmittel
- Animationen
- Andere freie Software und Informationsquellen im Internet

# Was ist Raytracing?

Die Bezeichnung „ray tracing“ steht für ein Verfahren, bei dem die Punkte eines Bildes durch Verfolgung einzelner Lichtstrahlen berechnet werden.

Für jeden Bildpunkt wird ein oder mehrere simulierte Lichtstrahlen von der Kamera in die virtuelle Szene „geworfen“, gegebenenfalls gebrochen und weiterverfolgt.

Auf diese Weise können realistische Schattenwürfe, Reflexionen und Lichtbrechungen simuliert werden.

# POV-Ray — Stärken

Der „Persistance of Vision“ Raytracer ist ein *freeware* Programm, welches aufgrund seiner Fähigkeiten, ansprechende, effektvolle Bilder zu erzeugen, inzwischen einen weiten Bekanntheitsgrad erreicht hat.

POV-Ray bietet:

- Versionen für mehrere Plattformen
  (Unix, Linux, DOS, Windows, MacOS, …)
- eine relativ leicht zu lernende Beschreibungssprache
- relativ viele geometrische Primitive
- viele Hilfsprogramme anderer Anbieter, welche die Arbeit mit POV-Ray erleichtern. (Modellierungswerkzeuge, Textureditoren, etc.)

# POV-Ray — Schwächen

Einige Nachteile von POV-Ray sollen aber nicht verschwiegen werden:

- es gibt schnellere Raytracer
- POV-Ray bietet keine photorealistische, physikalisch exakte Simulation. Es wird dagegen mehr Wert auf „hübsche“ Bilder gelegt
- vieles ist zwar theoretisch machbar, kann aber aufgrund fehlender geeigneter Werkzeuge nur mit erheblichem Aufwand realisiert werden (komplexe Animationen und Bewegungsabläufe, Modellierung natürlicher Formen wie z.B. menschliche Körper.)

# Die POV-Ray Szenenbeschreibungssprache

POV-Ray selbst beinhaltet keinen Modeller. Oft werden Szenen von Hand in POV-Rays eigener Beschreibungssprache definiert.

Es gibt allerdings einige brauchbare public domain, freeware und shareware Modeller. Diese sind jedoch in der Regel Platformabhängig.

Ein Beispiel …

# Eine einfache kleine Welt

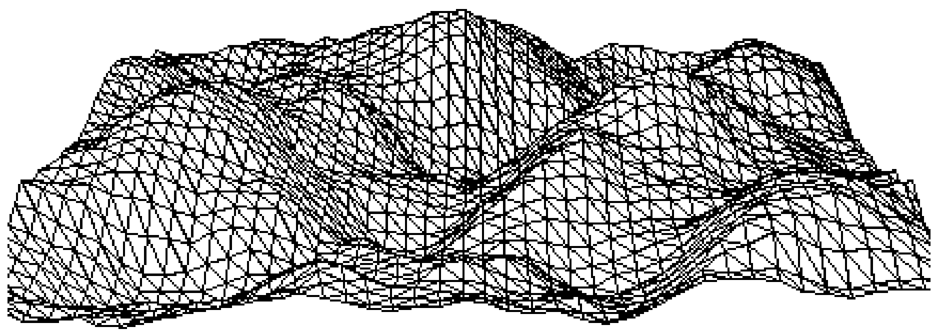
```
#include "colors.inc"  // Farbkonstanten

// Position und Blickrichtung der Kamera:
camera {
  location <4, 4, -4>
  look_at  <0, 1,  2> }

// eine gelbe Kugel und ein roter Klotz:
sphere { <1, 1, 2>, 2
  texture { pigment { color Yellow } } }
box { <-1, -1, -1>, < 1,  1,  1>
  pigment {color Red} }

// ohne Lichtquelle geht (fast) gar nichts:
light_source { <2, 4, -3> color White}
```

# Komplexe Szenenelemente

Viele gute und einführende Beispiele für einfache POV-Ray Szenen finden sich in der POV-Ray Dokumentation.

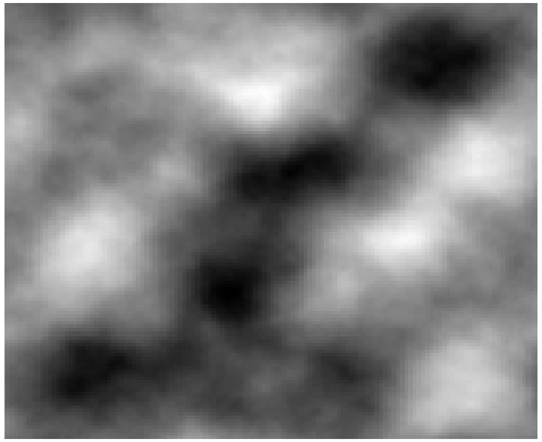
Im folgenden werden wir einige Werkzeuge kennenlernen, mit deren Hilfe man komplexere Szenen erzeugen kann.

Ein wichtiges Element der POV-Ray Beschreibungssprache ist das *height-field*.

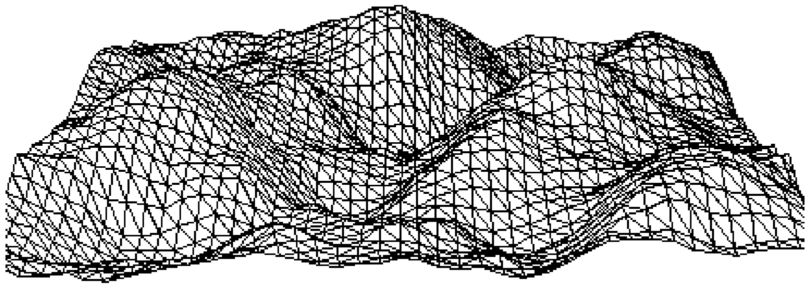


# Heightfields

Ein heightfield ist eine Oberfläche, deren Höhe durch die Farbwerte eines Graustufenbilds bestimmt wird. Heightfields sind Ideal um Gebirgszüge und Kraterlandschaften zu erstellen. Sie können aber auch hervorragend für andere Arten von Objekten eingesetzt werden.



$\longrightarrow$



# Erzeugung von Heightfields

Heightfields können auf verschiedene Arten erzeugt werden:

- Von POV-Ray selbst (siehe POV-Ray Dokumentation).
- Mit einem Bildbearbeitungsprogramm für Pixeldateien, z.B. *The GIMP*.
- Mit speziellen Werkzeugen, z.B. *hf-Lab*

# Beispielanwendung: Ein Schriftzug in Gold

Verfolgen wir nun eine Beispielanwendung für POV-Ray: Wir wollen einen Schriftzug erzeugen. Wir werden 3 grosse in Gold gehauene Buchstaben auf eine kleine Insel im Meer setzen.

Dazu benötigen wir

1. Buchstaben,
2. eine Insel,
3. das Meer und
4. die passende Athmosphäre.

# Buchstaben

POV-Ray erlaubt es, beliebige TrueType Schriften dreidimensional darzustellen. Diese Option unterliegt allerdings einigen Beschränkungen. so ist es z. B. nicht möglich, die Kanten solcher Schriften abzurunden.

Wir werden unseren Schriftzug als heightfield definieren. Dazu verwenden wir *The GIMP*.

Wir erzeugen unsere 3 Buchstaben und wenden den Weichzeichner darauf an.

# Eine Insel

Zur Erzeugung der Insel verwenden wir *hf-Lab*. (*hf-Lab* ist nicht Teil der POV-Ray Distribution.)

- Erzeugung eines zufälligen Gebirges:

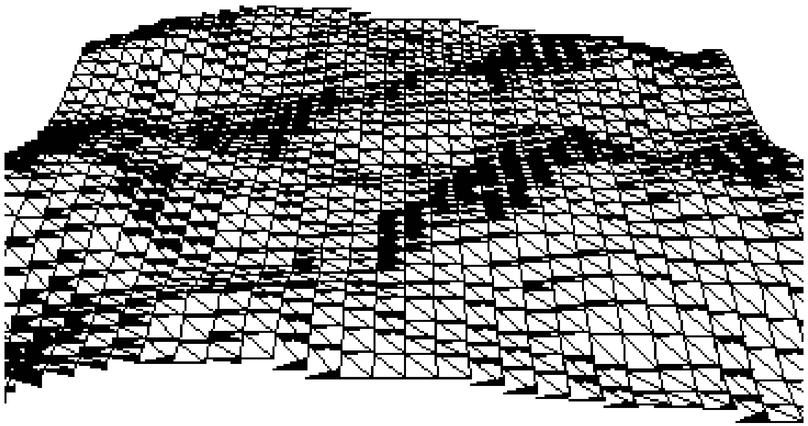
  ```
  HL> gforge 256
  ```

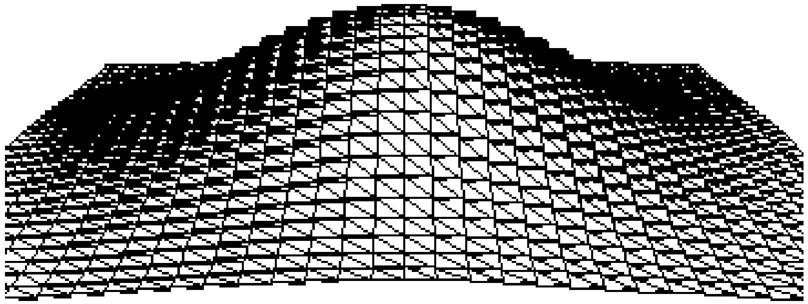
- Dieses Gebirge wird mit einem zweiten heightfield in Form einer Gaußglocke multipliziert.

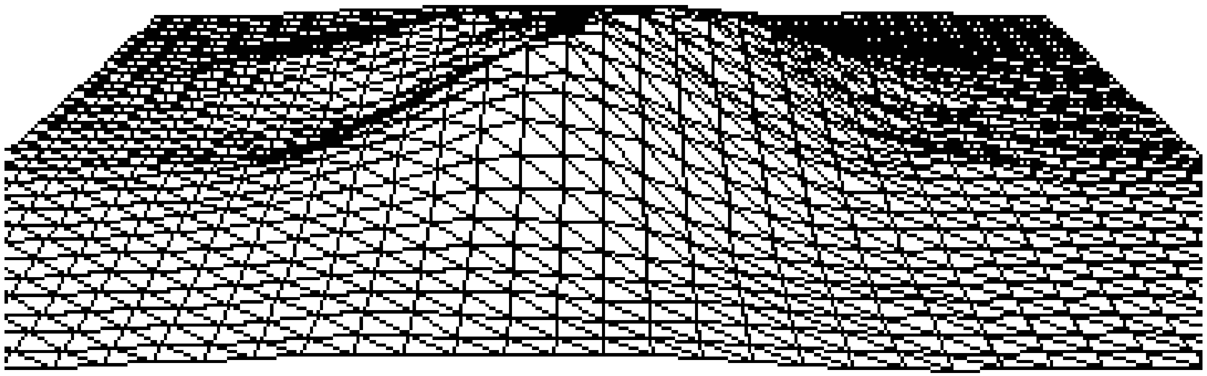
  ```
  HL> zero
  HL> gauss
  HL> mul
  ```

- Wir erhalten einen hübschen Hügel.

# Eine Insel (II)







# Buchstaben auf die Insel setzen

Die Buchstaben könnten jetzt mit *hf-Lab* auf den Hügel addiert werden. Dies hätte jedoch den Nachteil, daß wir den Buchstaben nur sehr schwer eine andere Textur geben können als dem Boden.

Wir werden daher zwei Heightfields erzeugen, die wir dann in POV-Ray ineinandersetzen.

Dennoch sollten wir den Buchstaben mit Hilfe von *hf-Lab* eine Wölbung geben, um sie an die Form der Insel anzupassen. Hierzu entfernen wir die feinen Unebenheiten der Insel mit einem Tiefpaß und addieren das Buchstabenheightfield. So werden die Buchstaben der Insel angepaßt, ohne deren rauhe Oberfläche zu übernehmen.

# erstes Ergebnis (ohne Meer)

Durch

- Positionierung der Heightfields in POV-Ray,
- geeignete Wahl der Texturen,
- selektive Beleuchtung durch einen Spot und
- Wahl eines geeigneten Blickwinkels

erhalten wir bereits ein erstes brauchbares Ergebnis (`cbr1.jpg`).

# Weitere Infos

*Diese Version des Vortrags ist leider noch unvollendet. Weitere Infos am LinuxTag 97. Ich plane zusätzlich, diese Folien auf meiner privaten Homepage anzubieten:*

http://wwwagr.informatik.uni-kl.de/ vollrath

POV-Ray: http://www.povray.org

The GIMP: http://www.xcf.berkeley.edu/ gimp/gimp.html

hf-Lab: (noch zu vervollständigen…)