Inhaltsverzeichnis

1 POV-Ray

Installation und Unterlagen

- POV-Ray für die KST
- Beispiele von Knaus

1.1 Bauteile

- **Kugel**. Eine Kugel kann mit `sphere{ <x,y,z> r pigment{color Red}}` erstellt werden x, y und z sind die Koordinaten des Mittelpunkts, r der Radius.
- **Zylinder**. Ein Zylinder kann mit `cylinder{ <x1,y1,z1>, <x2,y2,z2> r pigment{color Red}}` erstellt werden. Die Koordinaten sind dabei die Koordinaten des Fuss- und Kopfpunktes des Zylinder und r sein Radius.
- **Quader**. Ein Quader kann mit `box{<x1,y1,z1>, <x2,y2,z2>, pigment{color Red}}` erstellt werden. Die beiden Koordinaten bezeichnen dabei die Ecke vorne unten links und hinten oben rechts.
- **Polygon**. Ein Polygon in der xy-Ebene kann als `polygon{n <x1,y1> , <x2,y2>,...,<xn,yn>}` erstellt werden. n ist dabei die Anzal der Eckpunkte des Polygons gefolgt von den Punktepaaren $(x_1, y_1), \ldots, (x_n, y_n)$.
- **Lichtquelle**. POV-Ray unterstützt diverse Lichtquellen. Es kann mehr als eine geben. Eine Standard-Lichtquelle kann mit `light_source{ <x,y,z> color White}` erreicht werden. (x, y, z) ist dabei die Position der Lichtquelle gefolgt von der Definition der Farbe.
- **Kamera**. Jede Szene muss von einer Kamera aufgenommen werden. Typischerweise folgt daraus eine Zentralprojektion. Die Kamera kann wie folgt definiert werden

```
camera{
        location <xo,yo,zo>
        look_at  <xp,yp,zp>
        right -x*image_width/image_height
        sky <0,0,1>
        angle  30
}
```

  Dabei kann der Ort der Kamera (x_o, y_o, z_o) wie auch wo sie «hinschaut» (x_p, y_p, z_p) definiert werden. Zusätztlich dazu kann der Winkel (`angle`) definiert werden (im obigen Beispiel bei 30°). Die restlichen Parameter stellen sicher, dass POV-Ray mit einem rechtshändigen Koordinatensystem arbeitet.
- **Himmel** resp. **Hintergrund**. Um nicht alles vor einem schwarzen Hintergrund zu sehen, muss ein «Himmel» definiert werden. Die einfachste Variante dazu ist `sky_sphere{pigment{color White}}`.

1.2 Bewegungen

- **Rotieren**. Um ein Objekt um eine Achse zu rotieren kann `rotate <x,y,z>` verwendet werden. Die Werte x, y und z ergeben jeweils die Rotation in Grad um die jeweilige Achse.
- **Verschiebungen**. Verschiebungen können mit `translate<x,y,z>` realisiert werden. Dabei wir das Objekt vom x, resp. y resp. z Einheiten in die entsprechende Richtung verschoben.

1.3 Animation

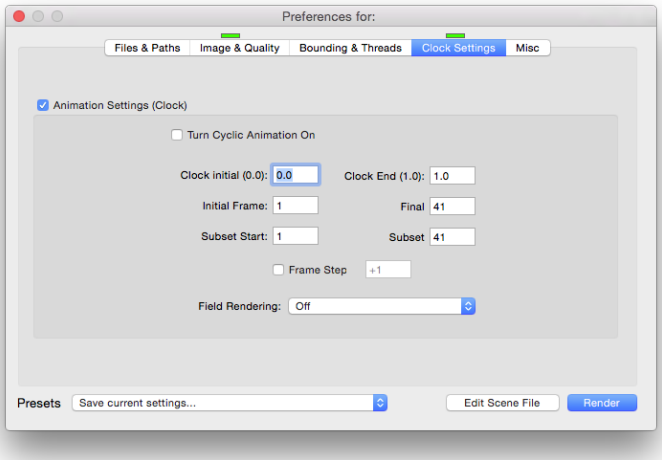
Um Objekte zu animieren kann die `clock` Variable verwendet werden. Die Clock Variable nimmt Werte zwischen `InitialClock` und `FinalClock` an. Der Abstand ist dabei durch die Anzahl Frames definiert.

1.3.1 Windows

In den vorgebenen INI-Files läuft die Clock-Variable von 0 nach 1 in insgesamt 41 Frames, das heisst, die Clock-Variable nimmt die Werte $0, 0.025, \ldots, 0.975, 1$ in den Frames $1, 2 \ldots, 40, 41$ an.

1.3.2 Apple

Steuer der Animation erfolgt über den Reiter Clock. Dabei ist zu beachten, dass die Clock Variable ebenfalls von 0 bis 1 in 41 Frames läuft. Das heisst, die Clock-Variable nimmt die Werte $0, 0.025, \ldots, 0.975, 1$ in den Frames $1, 2 \ldots, 40, 41$ an. Zu diesem Zweck sind folgende Einstellung vorzunehmen:



1.4 Beispiel

Im folgenden Beispiel wandert die rote Kugel von $(1, 2, 2)$ zu $(1, 2, 6)$ und die blaue Kugel verändert ihren Radius von $r = 1$ zu $r = 3$.

example1.pov

```
sphere{ <1,2,2+clock*4> 2 pigment{color Red}}
sphere{ <4,-2,2> (1+2*clock) pigment{color Blue}}
```

Möchte man ein Objekt rotieren könnte dies entweder mit `rotate` geschehen oder mit der entsprechenden Wahl von Koordinaten:

example2.pov

```
// Die Koordinaten eines Kreises mit Radius 3 sind
// 3*cos(winkel in bogenmass) und 3*sin(winkel in bogenmass)
cylinder{
 <3*cos(clock*2*pi),3*sin(clock*2*pi),0>, <3*cos(clock*2*pi),3*sin(clock*2*pi),3> 1
 pigment{color Blue}
}
//Fixe Koordinaten um nachher zu rotieren mit rotate
cylinder{
 <-1.5,-1.5,0> <-1.5,-1.5,3> 1
 rotate<0,0,-360*clock>
 pigment{color Yellow}
}
```

Genau so gut könnte die Kamera animiert werden. Im nachfolgenden Beispiel bewegt sich die Kamera in einem Kreis mit Radius 20 um den Mittelpunkt $(4, 4)$ und steigt in der z-Richtung von 10 auf 30 auf.

example3.pov

```
#include "colors.inc"
camera{
location <20*sin(clock*4*pi)+4,20*cos(clock*4*pi)+4,10+20*clock>
look_at <1,1,0>
right -x*image_width/image_height
sky <0,0,1>
angle 30
}
light_source{ <10,10,10> color White}
sky_sphere{ pigment{color White}}


cylinder{ <0,0,0>, <4,0,0>, 0.05 pigment{color Red}}
cylinder{ <0,0,0>, <0,4,0>, 0.05 pigment{color Blue}
cylinder{ <0,0,0>, <0,0,4>, 0.05 pigment{color Green}}

sphere{ <0,3,0> 1 pigment{color Brown}}

box{<0,0,0> <1,1,1> pigment{color Yellow}}
```

1.5 Weiteres

- Kombination von Objekten (z.B. Kanten eines Tetraeders). Mehre Objekte können zu einem kombiniert werden indem `union{objekt 1 objekt 2 ...}` verwendet wird.
- Texturen. Um Objekten eine interessantere Oberfläche zu verschaffen kann auf bestehende «Texturen» («textures») zurückgeriffen werden.

  example4.pov

  ```
  #include "colors.inc"
  #include "textures.inc"

  box{<0,0,0> <1,1,1> texture{Blood_Marble}}
  ```

1.6 Internetressourcen

Eine sehr hilfreiche Webseite ist die Sammlung von Friedrich Lohmühller: http://www.f-lohmueller.de/

2 Aufgaben

1. Schau dir das `example0.pov` an. Verändere bestehende Werte (Koordinaten der Objekte, der Kamera, des Lichts, die Farben etc.) und rendere das Bild erneut.

   Eine Übersicht über die Farben gibt es z.B. hier.

2. Füge der Szene eine Kugel oder ein Zylinder hinzu.

3. Erstelle eine Animation deiner Wahl (z.B. basierend auf `example1.pov` oder `example2.pov`). Videos der Animationen sind entweder im Beispiel-Ordner oder auf Youtube zu finden.