Doku Gruppe 42 - IVC WiSe 15/16

Kamila Ignatowicz
Marco Pfomann
Felix Favre

26. Januar 2016

1 Umgebung

1.1 Himmel

Für den Sternenhimmel wurde eine sky_shpere mit bozo pigment genutzt

```
sky_sphere {
  pigment {
    bozo
    color_map {
      [0.0 White*3]
      [0.2 Black]
      [1.0 Black]
    }
    scale .006
  }
}
```

1.2 Boden

Der Boden ist eine einfache graue Plane auf der xz-Ebene

```
plane { y, 0 pigment { color red 0.1 green 0.1 blue 0.1} }
```

1.3 Media

Damit die Lichtstrahlen der Laser und Movindheads in der Luft zu sehen sind ist ein media nötig. Hier wurde bewusst kein atmospheric media genutzt, da povray nicht in

der Lage ist unendlich lange Lichtstralen in einem media zu samplen. Für unsere Zwecke Reicht eine Box mit 200x152x160 Metern. Der extinction Wert ist so gering gesetzt, weil wir mit Distanzen von über 100 Metern arbeiten.

```
box{<-50,-2,10>,<150,150,-150> pigment{rgbt 1}
  interior{
    media {
      scattering {
        4,
        1
        extinction 0.001
      }
      method 3
    }
  }
  hollow
}
```

1.4 Includes

Der Rest der Szene wurde in eigenen Dateien definiert und per #include Statement eingebunden

```
#include "helpers.inc"
#include "lightcolors.inc"
#include "switches.inc"

#include "camera.pov"
#include "lasers.pov"
#include "buehne.pov"
#include "movingheads_buehne.pov"
#include "movingheads_tower.pov"
#include "firework.pov"
#include "dj.pov"
#include "fans.pov"
```

2 Kamera

Wir haben mehrere camera-Objekte erstellt und diese dann in ein Array gesteckt und mit einer Random-Funktion ausgewaehlt. Damit wir nicht immer wieder die gleichen

Einstellungen haben, wechselt die Random-Funktion die Kamera schneller, als eine Einstellung lang ist. Somit haben wir den Effekt, dass sie Kameras immer wieder andere Positionen haben.

```
//camera 7: rear area of dancefloor floating forward
#declare camera7=camera {
  location <50, 5, -100>
  look_at <50, 5, 0>
  right x*image_width/image_height
}

#local CameraRand=seed (clock/5);
#local CameraArray=
array[9] {
  camera0,
  camera1,
  camera2,
  camera3,
  camera4,
  camera5,
  camera6,
  camera7,
  camera8,
};

camera{
  CameraArray[int(rand(CameraRand)*9)]
}
```

3 Buehne

Die Bühe wurde aus einer einfachen Kombination aus box, cylinder und torus Elementen zusammengestellt. Um Codeduplikationen zu vermeiden wurden einzelne Teile als object definiert und mit Rotation und Translation an die Richtige stelle gerückt. Die Farben der einzelnen Elemente wurde in der lightcolors.inc definiert. Ein Beispiel:

```
#declare towersegment=union{
  box {
    <0, 0, 0>
    <0.5, TowerHeight, 0.7>
  }
  #for (i, 1, 5, 2)
```

```
  cylinder {
    <0,0,0>,
    <0,0.01,0>,
    4
    rotate <0,0,90>
    translate <0,i*4.4,5>
  }
  #end
  torus {
    4.15, 0.35
    rotate <0,0,90>
    translate<0.25,4.4,5>
    pigment { color C_TowerBot }
    finish {ambient 1}
  }
  torus {
    4.15,.35
    rotate <0,0,90>
    translate<0.25,3*4.4,5>
    pigment { color C_TowerMid }
    finish {ambient 1}
  }
  torus {
    4.15,0.35
    rotate <0,0,90>
    translate<0.25,5*4.4,5>
    pigment { color C_TowerHi }
    finish {ambient 1}
  }

}

#declare tower=union {
  object { towersegment }
  object { towersegment rotate <0,270,0> translate <10,0,0> }
  object { towersegment rotate <0,180,0> translate <10,0,10> }
  object { towersegment rotate <0,90,0> translate <0,0,10> }
}
```

4 Laser und Movingheads

Die Laser und Movingheads werden von einer großen Anzahl Spotlights dargestellt. Für die Laser wurde ein Makro definiert und mithilfe von for-Schleifen vielfach eingesetzt. Als Parameter hatte dieses die Ursprungsposition des Lasers und die Farbe.

```
#for (i, 0, lasercount, 1)
  laser(<37,2,-0.05>, C_RGBLaser)
#end
```

Das Makro wiederum definiert das Spotlight

```
#macro laser (source, lasercolor)
light_source {
  <0,0,0>
  color lasercolor
  spotlight
  radius 0.10
  falloff 0.10
  tightness 50
  media_interaction on
  media_attenuation on
  point_at <0,0,-100>
  rotate <0,(i-(lasercount/2))/2,0>     //Spread
  rotate <0,0,laser_rotate(i)>  //Rotate
  rotate <laser_tilt(i),0,0> //Tilt
  translate source
}
#end
```

Die Rotation um x-Achse (Tilt) und z-Achse (Rotate) wurden mit 2 Funktionen berechnet. Diese haben verschiene Formeln von denen eine Anhand einer Zufallsvariable eine ausgewählt wird. Sowohl die Funktionen selbst als auch der Seed der Zufallsvariable sind von der Clock abhängig. Hiermit wird eine Flüssige Animation erzeugt sowie der Wechsel der Animationen alle paar Sekunden sichergestellt.

```
#local Rand_Lasermode = seed(clock+23);

#declare lasermode_rotate=int(rand(Rand_Lasermode)*3);
#declare lasermode_tilt=int(rand(Rand_Lasermode)*2);

//Only Nice Sine Wave lasers <3
#if (lasermode_tilt = 1)
```

```
  #declare lasercount = 50;
  #declare lasermode_rotate = 0;
#end

#local laser_rotate = function(i) {
  #switch (lasermode_rotate)
  #case(0)
    0
    #break
  #case(1)
    360*clock
    #break
  #case(2)
    sin(clock*pi)*45
    #break
  #else
    0
  #end
}

#local laser_tilt = function(i) {
  #switch (lasermode_tilt)
  #case(0)
    22.5+sin(clock*1.5*pi)*22.5
    #break
  #case(1)
    10+sin(i/lasercount*pi*2+(clock*2))*5
    #break
  #else
    0
  #end
}
```

5 Fans und DJ

Fuer die Fans und den DJ haben wir den Prototypen komplett selbst gebaut.

5.1 Koerper

- Die Cap ist eine union einer intersection von box und sphere mit einer platten sphere
- Der Kopf ist eine einfache Sphere, die auf der x-Achse etwas gestaucht wurde

- Der Hals ist ein einfacher Zylinder
- der Torso ist ein superellipsoid, den wir so geformt haben, dass er einem Oberkoerper nahe kommt
- Die Arme bestehen aus jeweils zwei Zylindern, die jeweils an Schultern und Ellbeugen durch eine sphere verbunden sind
- Die Haende sind einfach jeweils eine sphere
- Die Beine sind jeweils ein Zylinder, der zum Fuss hin schmaler wird
- Die Schuhe sind jeweils ein Prisma

```
// Cap
union {
  intersection {
    sphere { <0, 0, 0>, 0.090 scale <0.9, 1, 1.1> }
    box { <0.100, 0, 0.100>, <-0.100, -0.100, -0.100>
      inverse }
  }

  sphere { <0, 0, 0>, 0.090 scale
    <1, 0.05, 1> translate <0, 0, 0.060> }
  }
}

// torus
superellipsoid{
  <0.25, 1.00>
  scale <0.220, 0.350, 0.120>
}

// left leg
cone {
  <Legs, 0.850, 0>, 0.075, <Legs, 0.100, 0>, 0.040
}

prism {
  0, 0.100, 6,
  <0, 0>,
  <0, 0.300>,
  <0.050, 0.300>,
  <0.100, 0.100>,
  <0.100, 0>,
  <0, 0>
}
```

5.2 Arme

Um die Arme bei den springenden Fans und dem DJ immer gleich lang zu lassen, sind die Arme durch Variablen verbunden. Durch eine Normalisierungsfunktion bleiben die einzelnen Elemten der Arme immer gleich lang.

```
// Left Arm
#declare Intersection_Left_Shoulder = <-0.200, 1.450, 0>;
#declare Intersection_Left_Arm = vnormalize(<0, -3, 0>)
  *0.300+Intersection_Left_Shoulder;
#declare Left_Hand = vnormalize(<0, -2.5, 0>)
  *0.250+Intersection_Left_Arm;

//Right Arm
#declare Intersection_Right_Shoulder = <0.200, 1.450, 0>;
#declare Intersection_Right_Arm = vnormalize(<0, -3, 0>)
  *0.300+Intersection_Right_Shoulder;
#declare Right_Hand = vnormalize(<0, -2.5, 0>)
  *0.250+Intersection_Right_Arm;
```

6 DJ-Pult

Das DJ-Pult besteht ganz einfach aus mehreren Boxen, die aufeinander gestapelt sind. Die oberste, kleinste Box stellt das Mischpult des DJs dar. Das Mischpult ist ein template, das auf die box gelegt wurde.

```
#declare dj_pult=texture {
  uv_mapping
  pigment {
    image_map {
      jpeg "dj_pult_long.jpg"
      map_type 0
      interpolate 2
    }
  }
}
```

Um das Bild ordentlich auf auf die Box zu legen, mussten wir den Mischpult-Teil auf ein Bild legen, dass die ganze Box umspannt. Ansonsten hatten wir verzerrungen des Templates