Computergrafik SS 2014

Kapitel 1 + 2:
Einführung und GUI-Programmierung

Vorlesung vom 22.04.2014

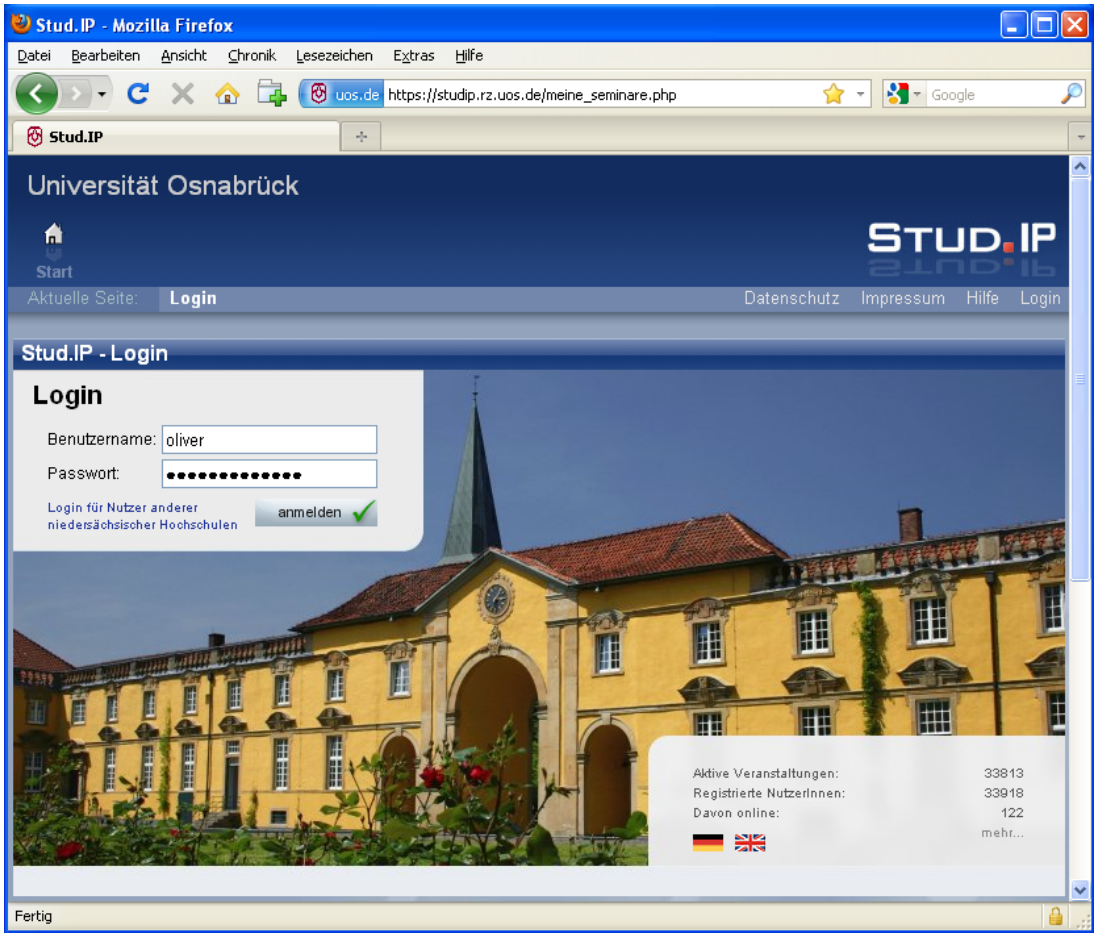
Oliver Vornberger

Institut für Informatik
Universität Osnabrück

Organisation

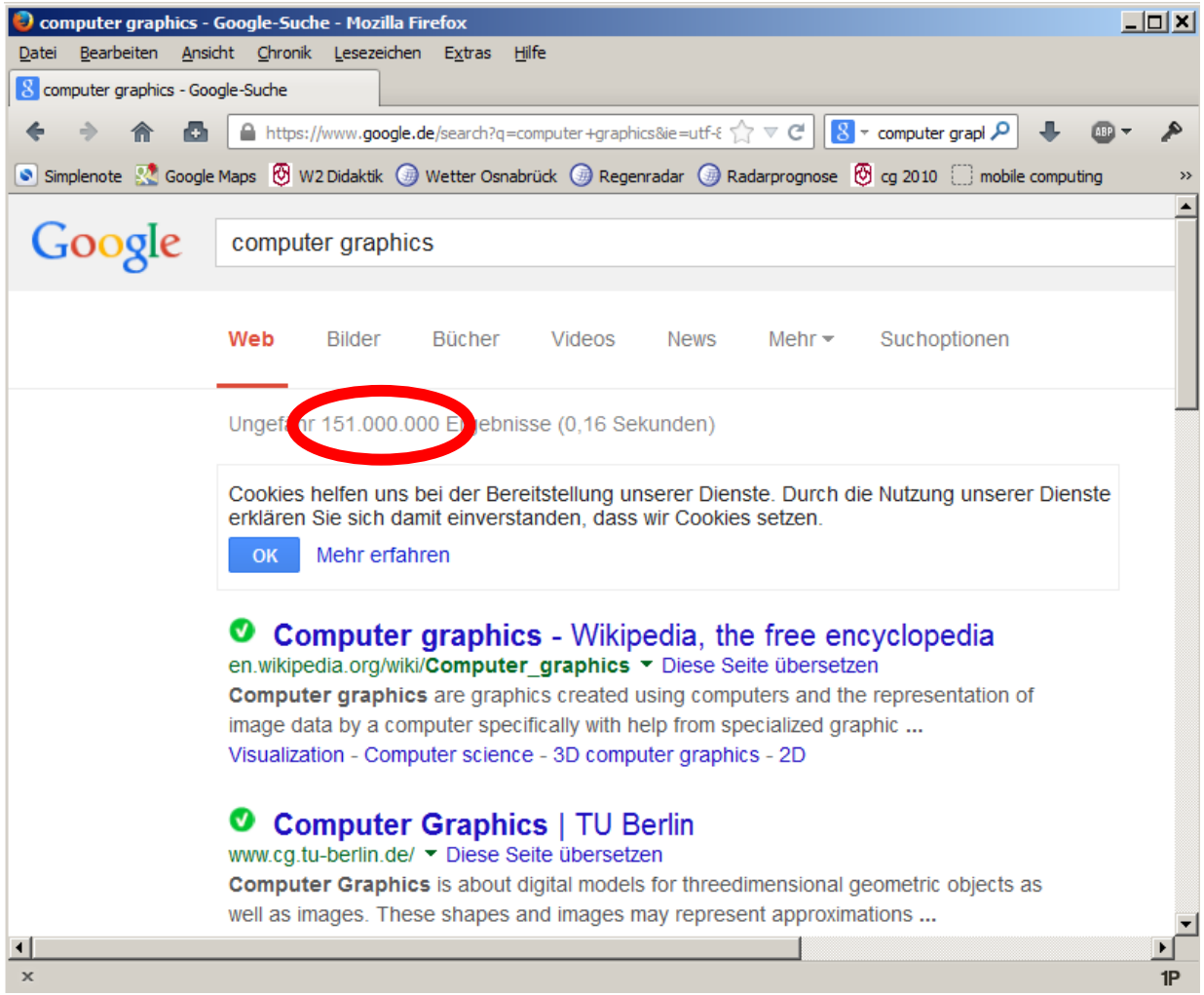
Vorlesung	montags	10:15 Uhr	31/E06
	dienstags	10:15 Uhr	31/E06
Übung	donnerstags	10:15 Uhr	69/125
	freitags	10:15 Uhr	31/E05
Übungsblatt	dienstags		
Testate	montags		
	dienstags		
	mittwochs		

stud.ip



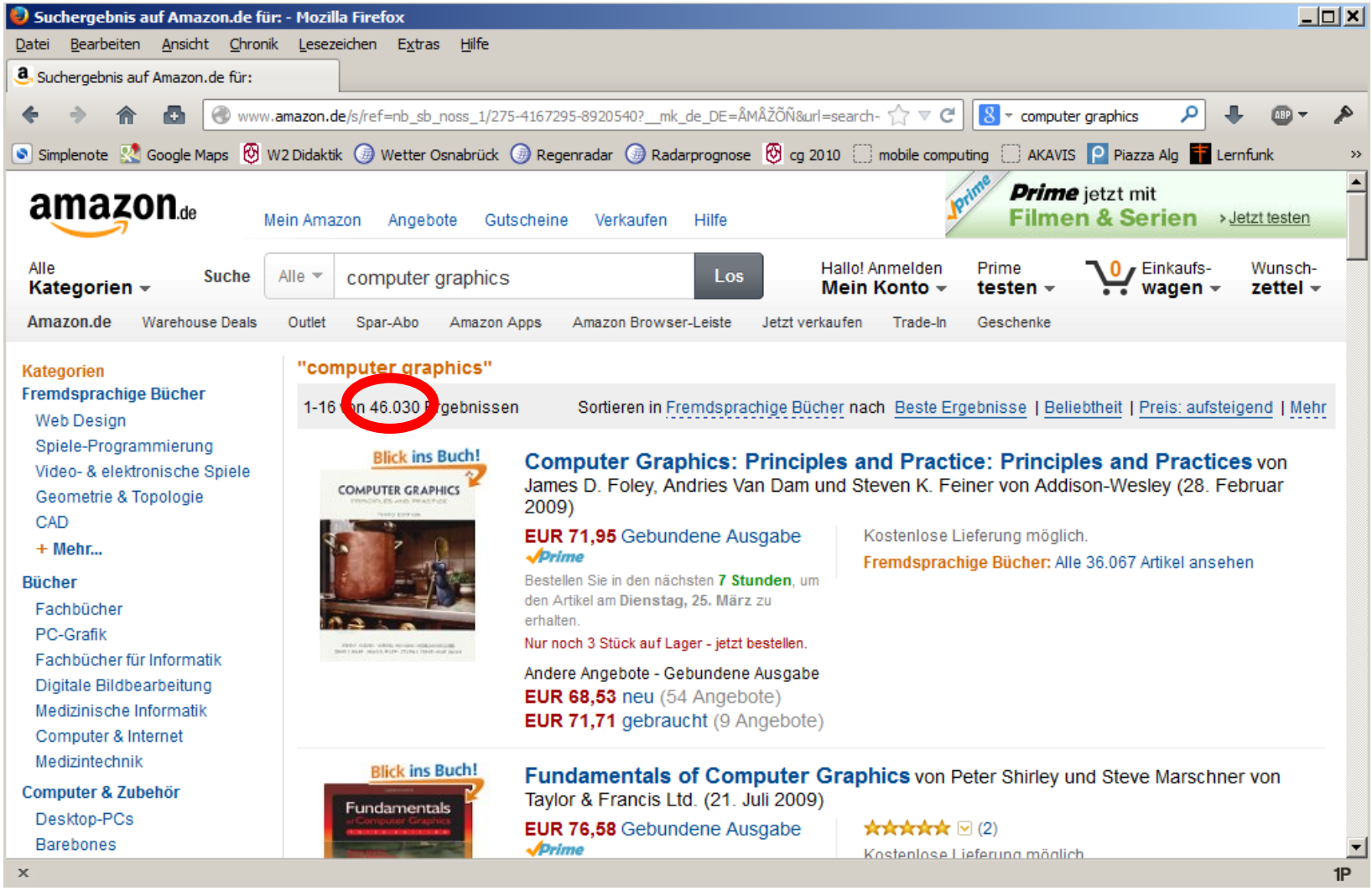
https://studip.rz.uos.de/

Google



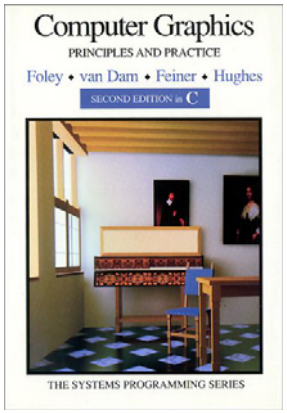
http://www.google.de/search?q=computer+graphics

Amazon

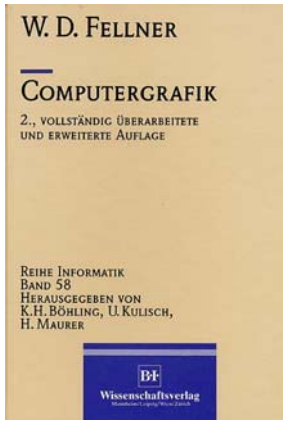


http://www.amazon.de

Literatur

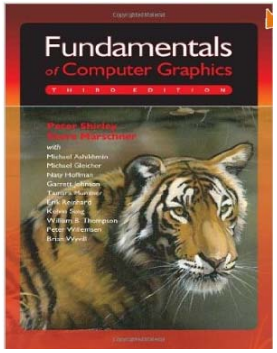


James Foley et al:
Computer Graphics Principles and Practice
2nd Edition
Addison Wesley 1995



Dieter Fellner:
Computergrafik
BI 1994

Klaus Zeppenfeld:
Lehrbuch der Grafik-programmierung
Spektrum 2004



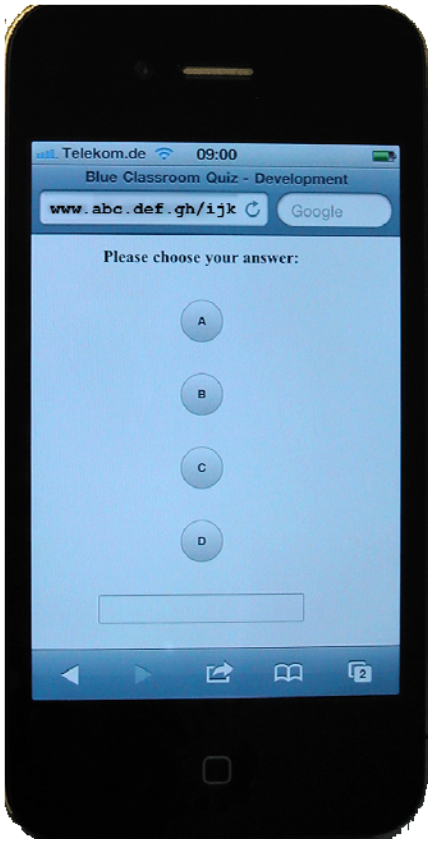
Peter Shirley:
Fundamentals of Computer Graphics
Tayler & Francis 2009

Begleitmaterial

- Skript in HTML
- Skript in PDF
- Folien in PDF
- Videomitschnitt im Matterhorn-Format
- Videopodcast im mp4-Format
- Audiomitschnitt im mp3-Format

http://www-lehre.inf.uos.de/~cg/2014

Classroomquiz



Motivation

- Bild sagt mehr als 1000 Worte
- Auge erfasst 40.000.000 Bit/sec
- Lesegeschwindigkeit
  = 10 Worte à 5 Zeichen/sec
  = 10 · 5 · 8 = 400 Bit/sec
- $\Rightarrow$ Faktor 100.000

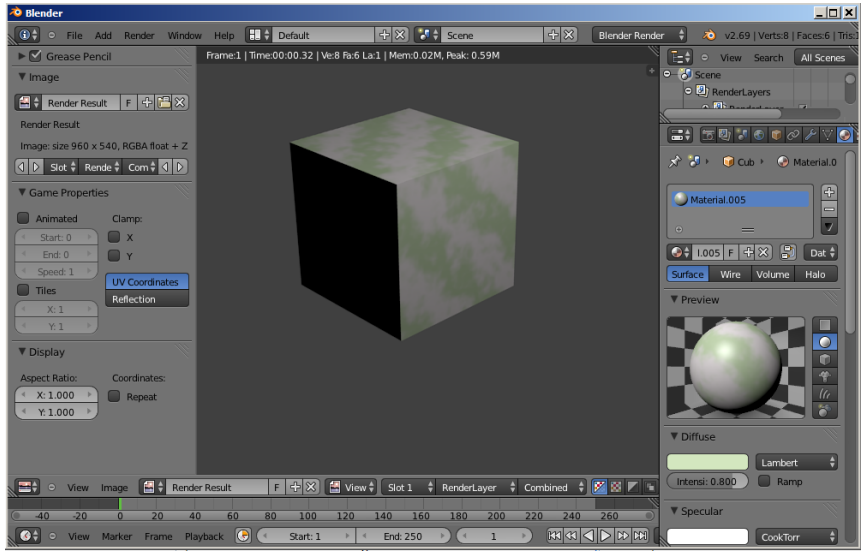
Grafische Datenverarbeitung

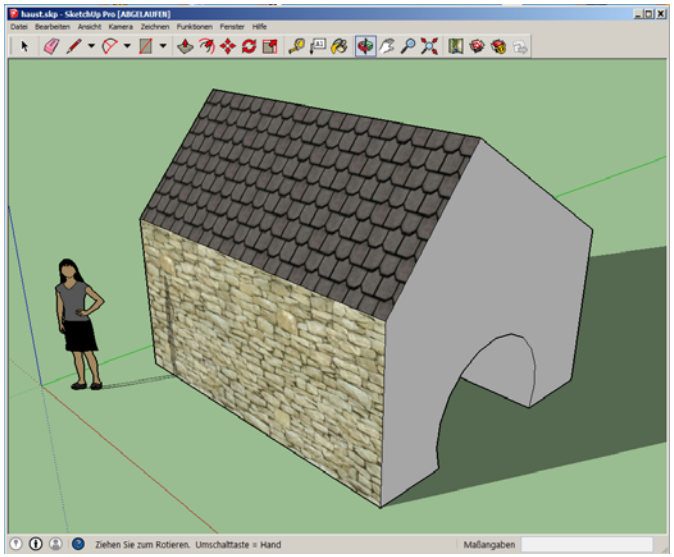
- Bildverarbeitung
  - Licht, Radar, Röntgen, Ultraschall, …
  - Vereinfachung, Verbesserung
- Mustererkennung
  - Analyse von Rasterdaten
  - Optical Character Recognition (OCR)
- Generative Computergrafik
  - Eingabe der Repräsentation
  - Ausgabe der Darstellung

Anwendungen

- Business-Grafik
- Grafische Benutzeroberflächen
- Kartografie
- CAD (Haus, Auto,...)
- Visualisierung (Molekül, Strömung, Scan)
- Simulation (Fahrzeug, Flugzeug,...)
- Virtual Reality (Computerspiele,...)

Modellieren, Projizieren, Rendern

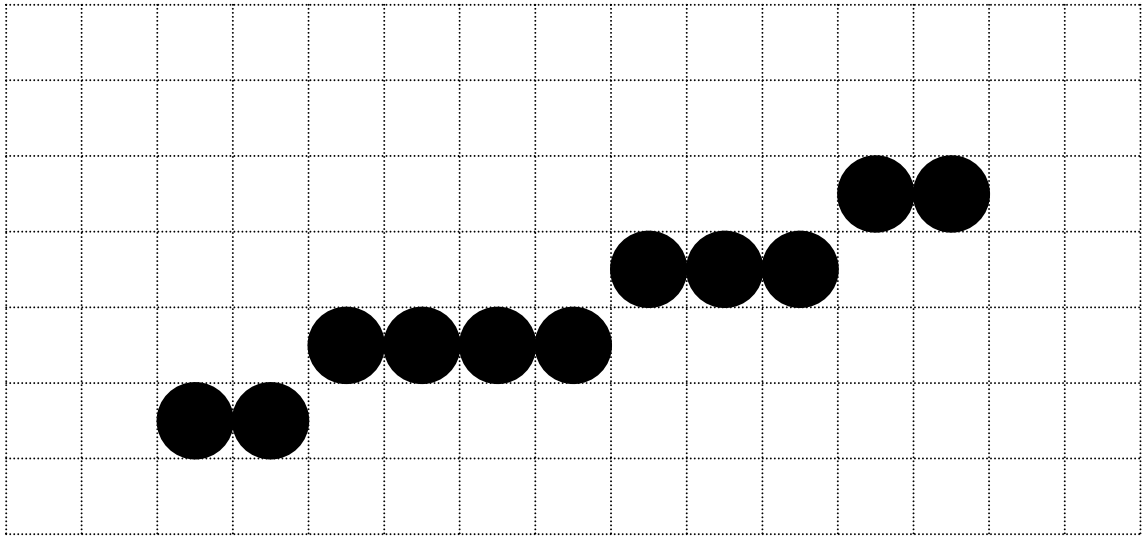




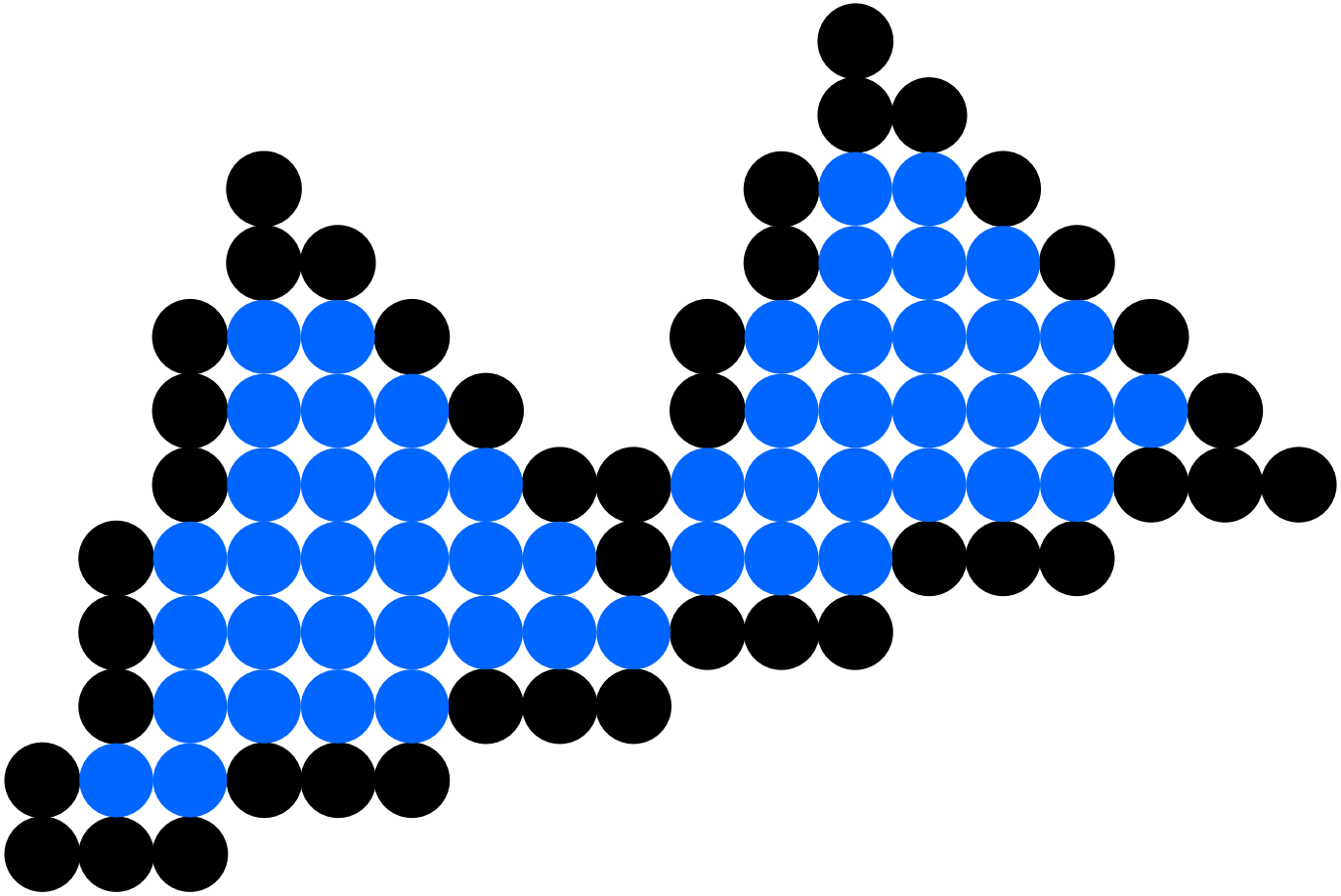
http://www.blender.org

http://www.sketchup.com/de

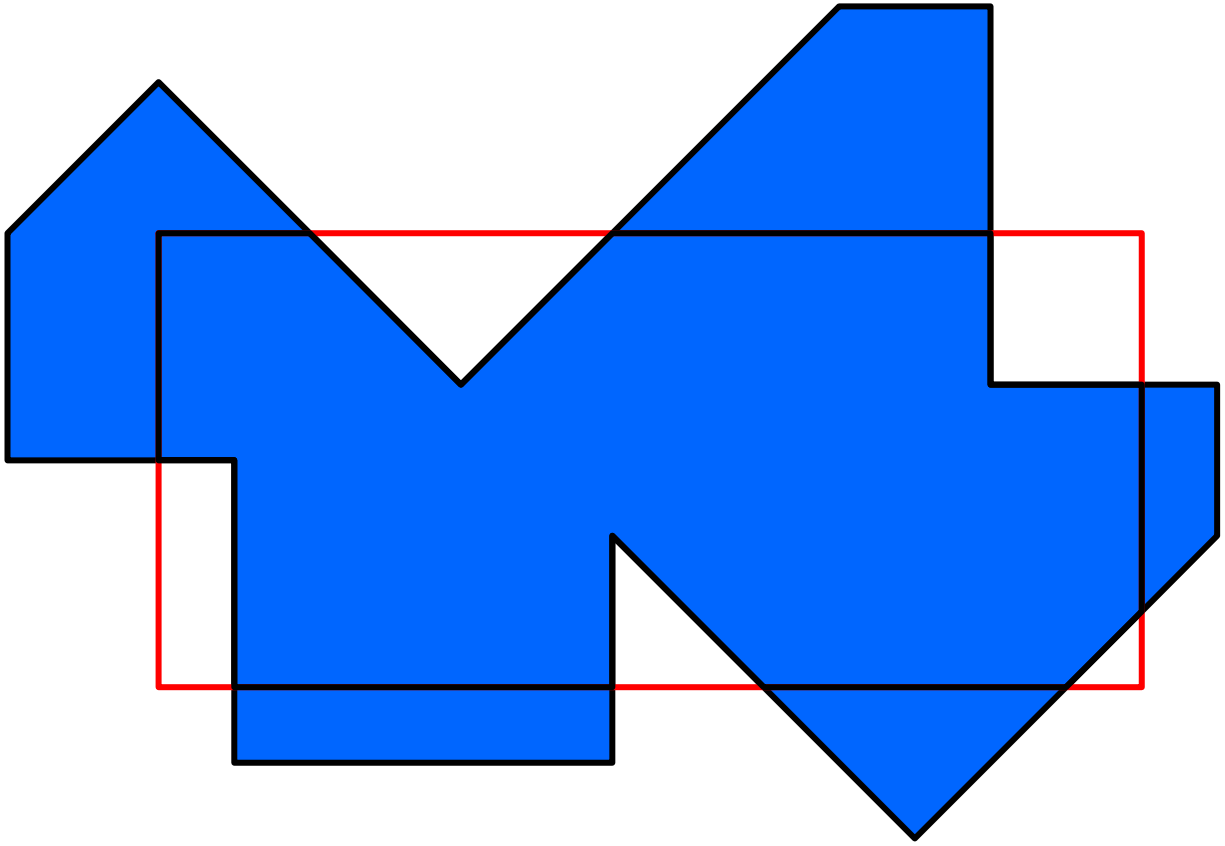
2D-Grundlagen



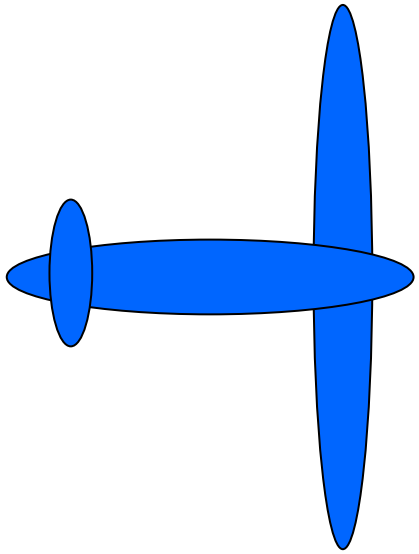
2D-Füllen



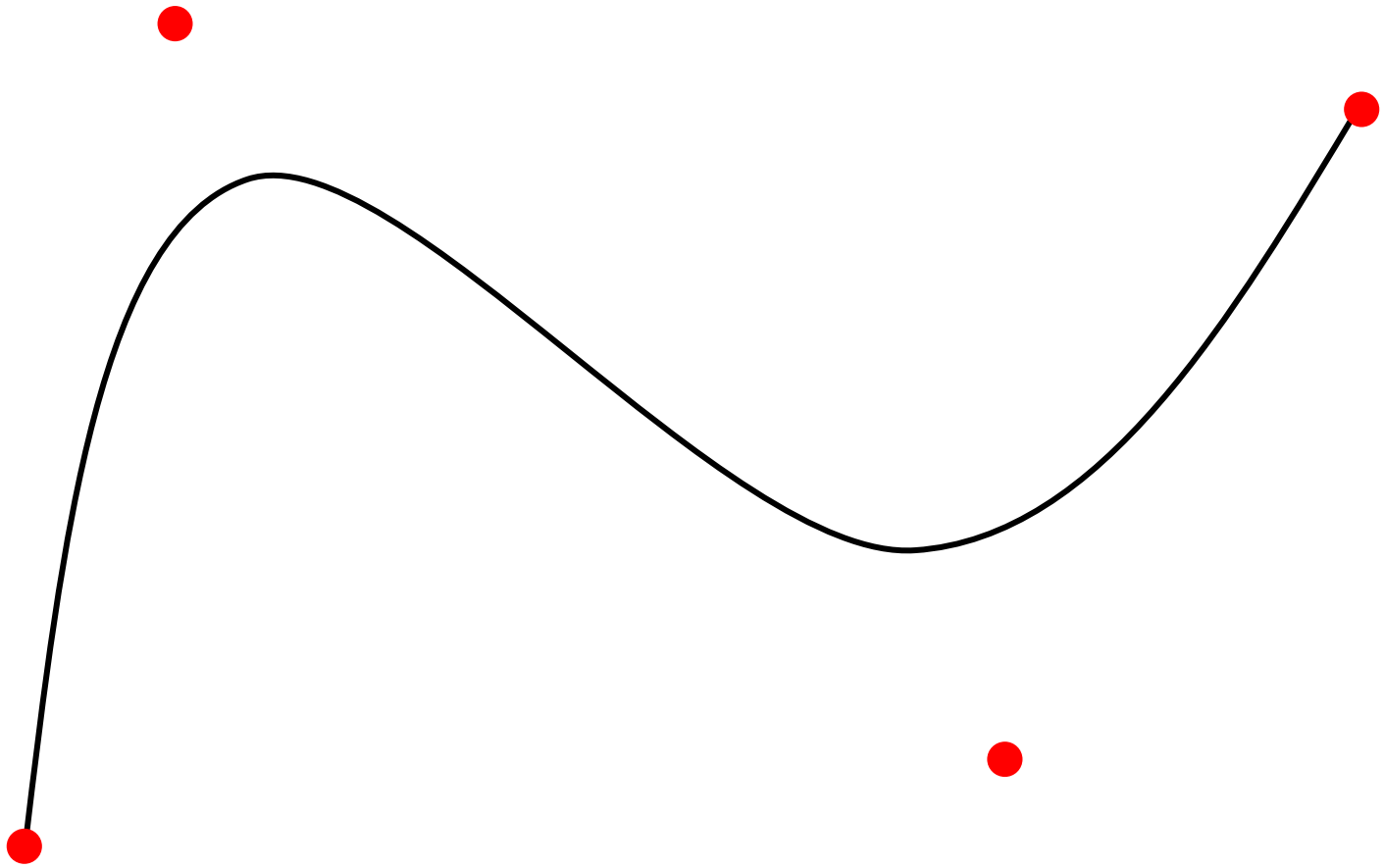
2D-Clipping



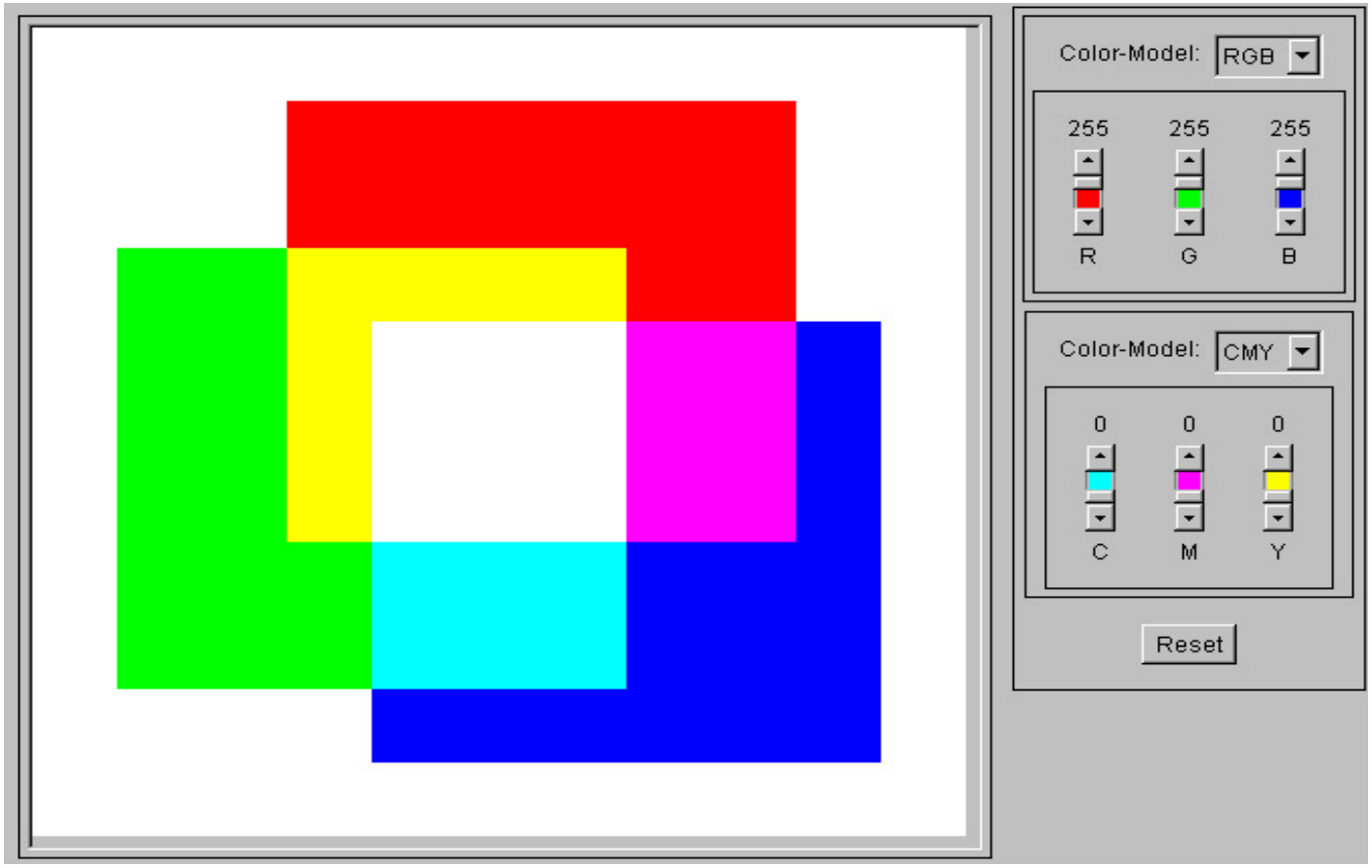
Transformation



Kurven



Farbe

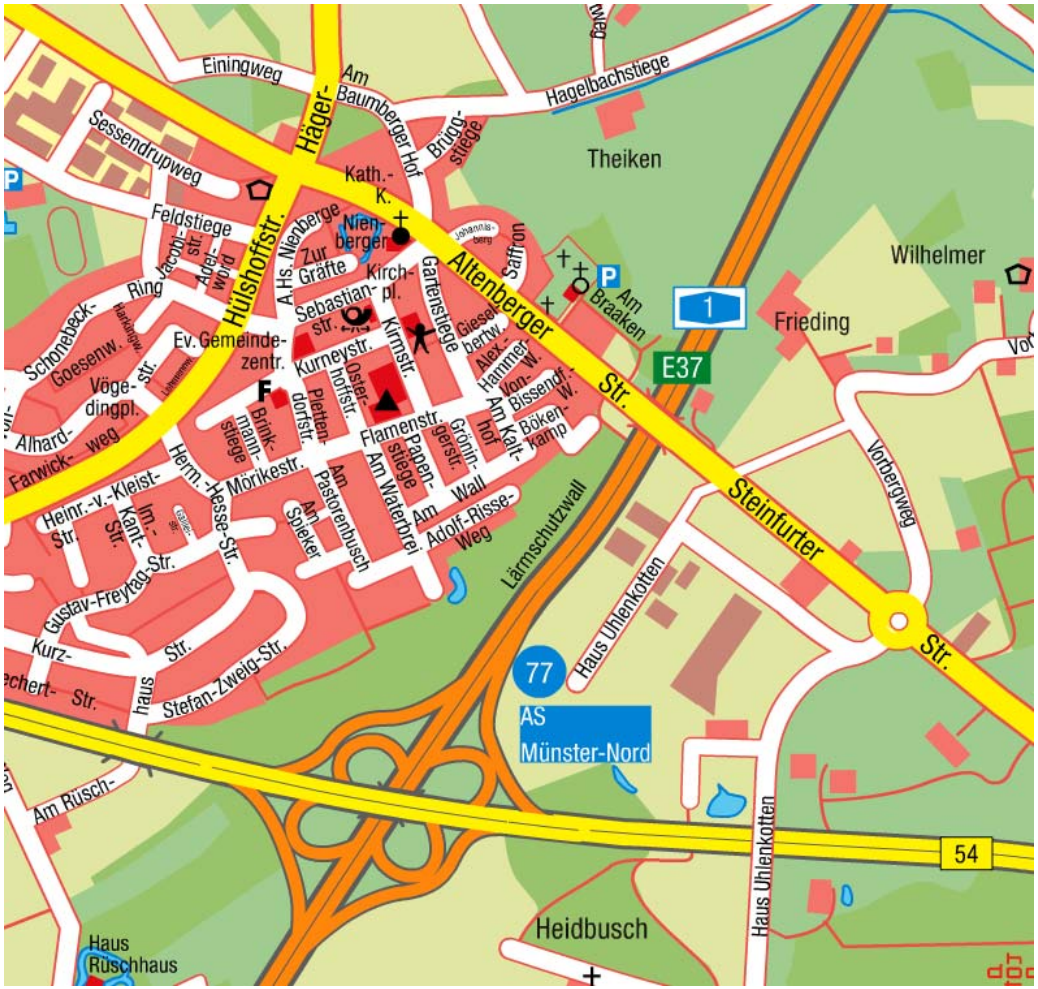


Pixeldateiformate

PNG mit 16 Millionen Farben

GIF mit 16 Farben

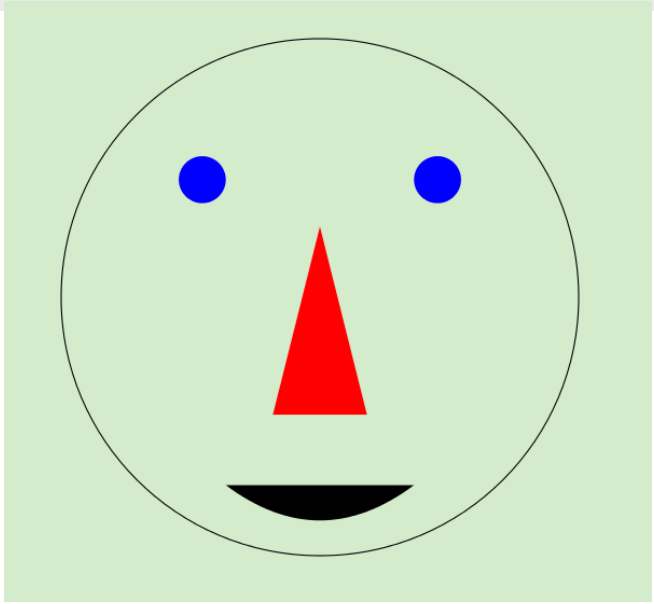
Macromedia Flash



http://www-lehre.inf.uos.de/~cg/2014/Flash/karte.html

SVG

```
<html>
  <body bgcolor="d4ebcc">
    <svg width="800" height="500">
      <circle  fill="none" stroke="black" cx="300" cy="250" r="220" />
      <circle  fill="blue" cx="200" cy="150" r="20" />
      <circle  fill="blue" cx="400" cy="150" r="20" />
      <polygon fill="red" points="300,190 340,350 260,350 300,190" />
      <path    d="M220,410 Q300,470 380,410" />
    </svg>
  </body>
</html>
```



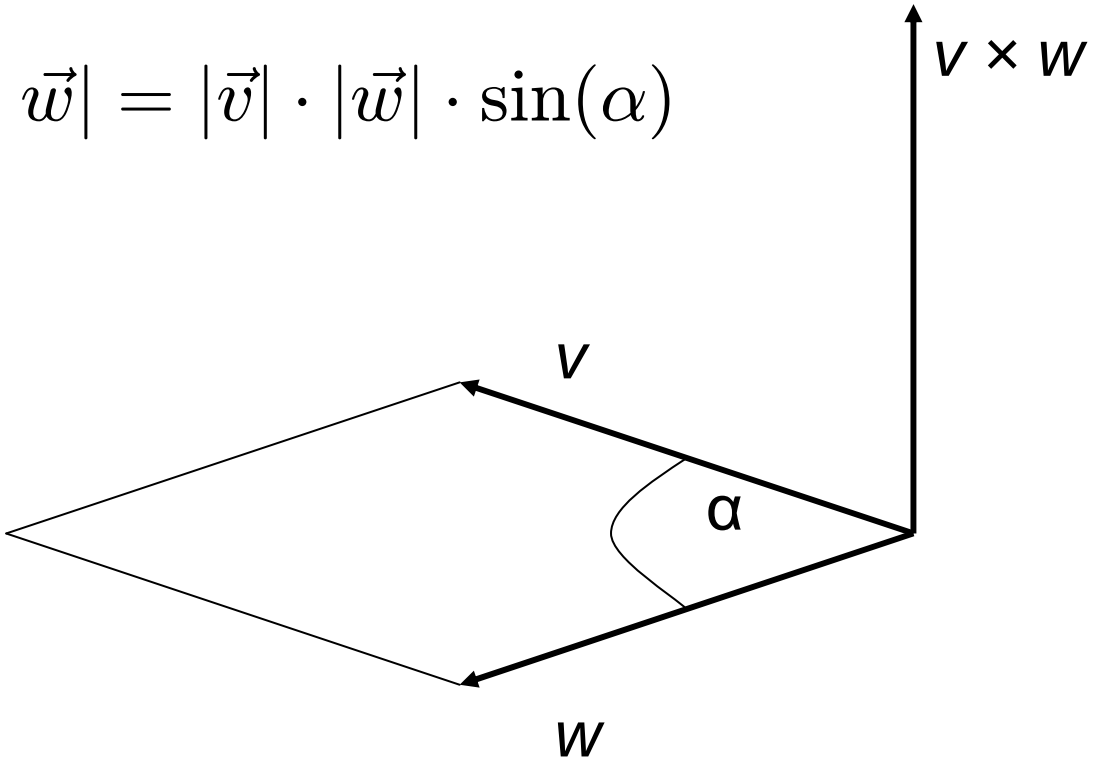
http://www-lehre.inf.uos.de/~cg/2014/SVG/gesicht.html

Fraktale

http://www-lehre.inf.uos.de/~cg/2014/skript/Applets/IFS/fraktal.html

3D-Grundlagen

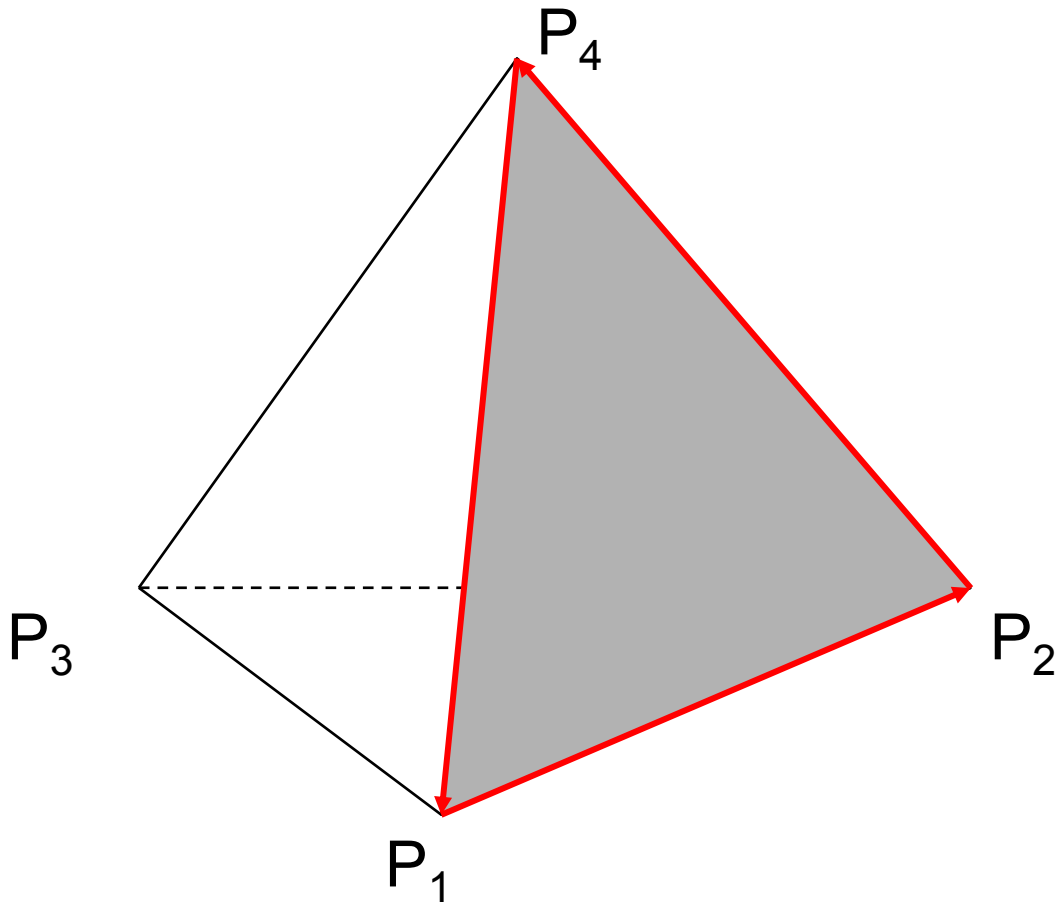
$$|\vec{v} \times \vec{w}| = |\vec{v}| \cdot |\vec{w}| \cdot \sin(\alpha)$$



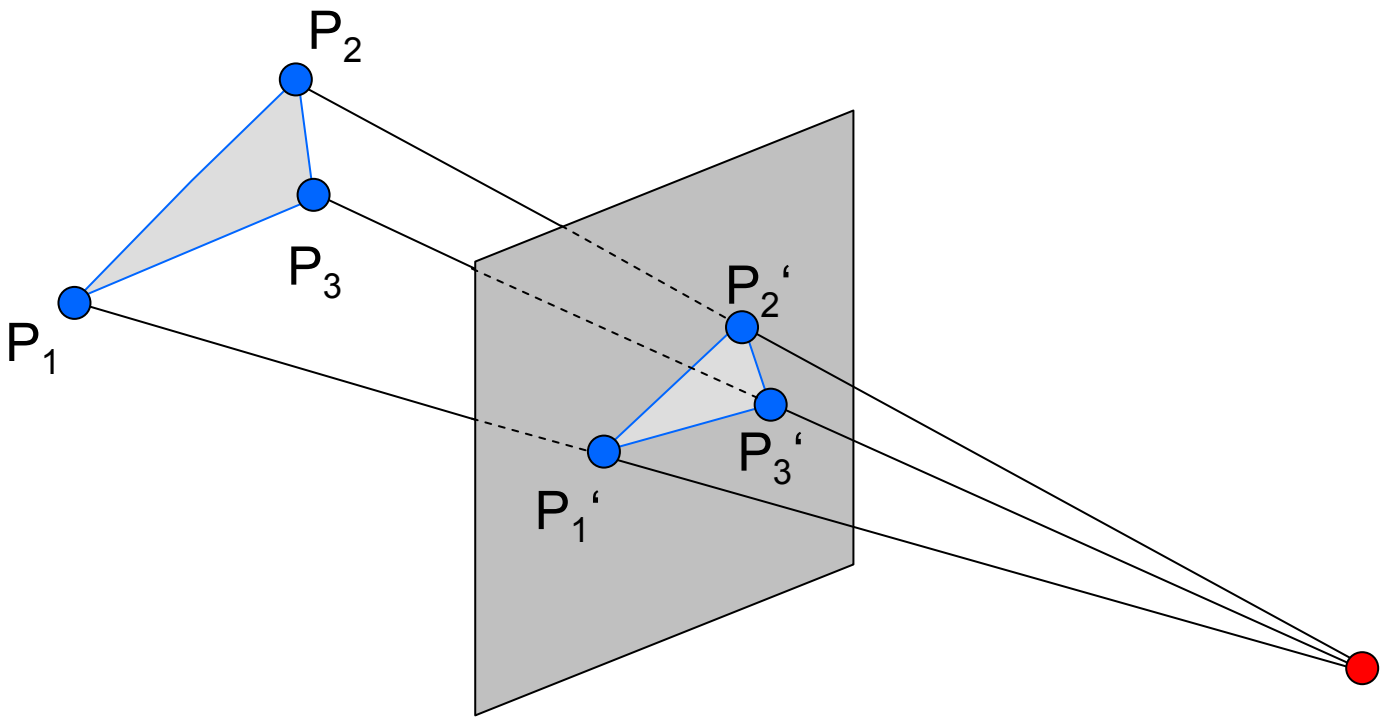
3D-Transformationen

$$R_y(\delta) = \begin{pmatrix} \cos(\delta) & 0 & \sin(\delta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\delta) & 0 & \cos(\delta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

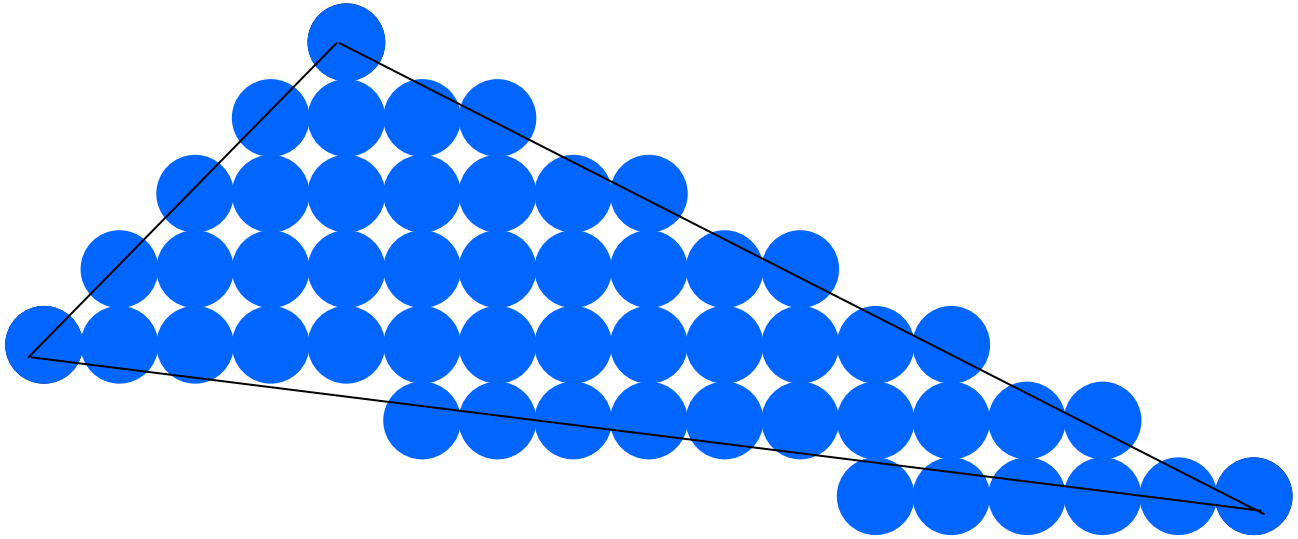
3D-Repräsentation



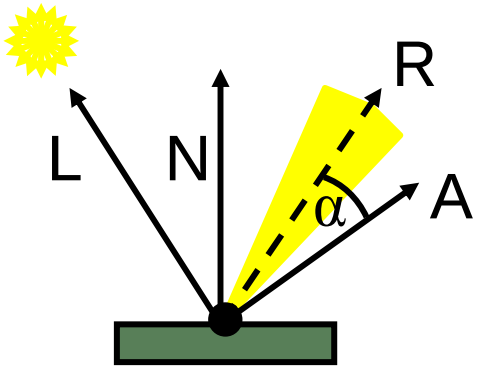
Projektion

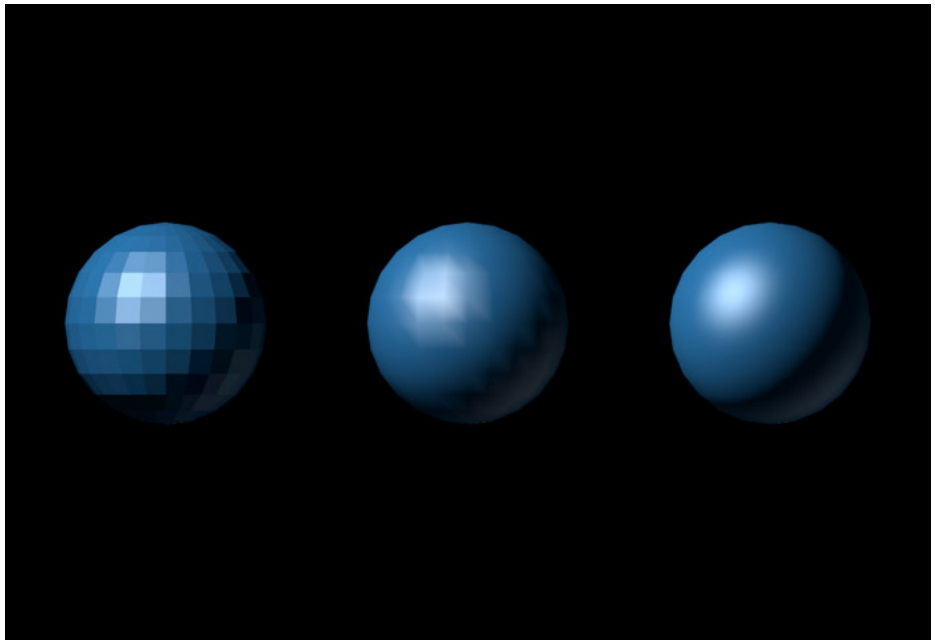


Rendern

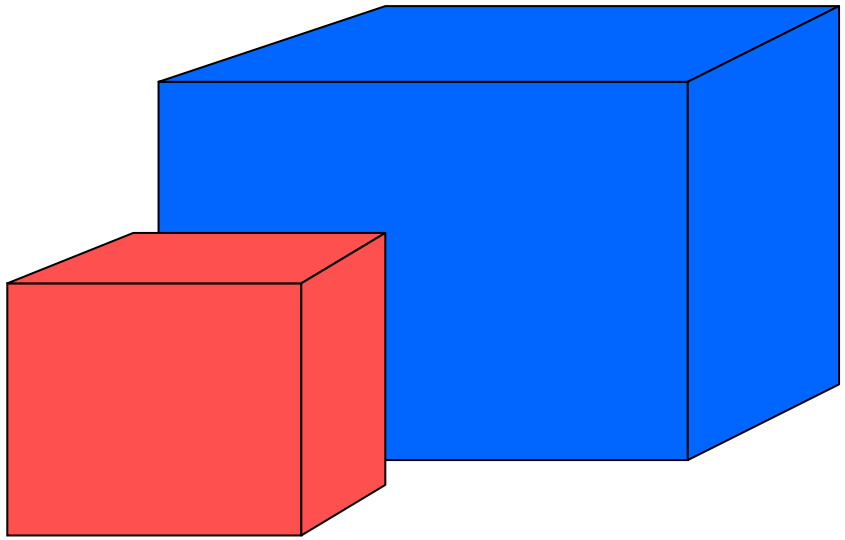


Beleuchtung

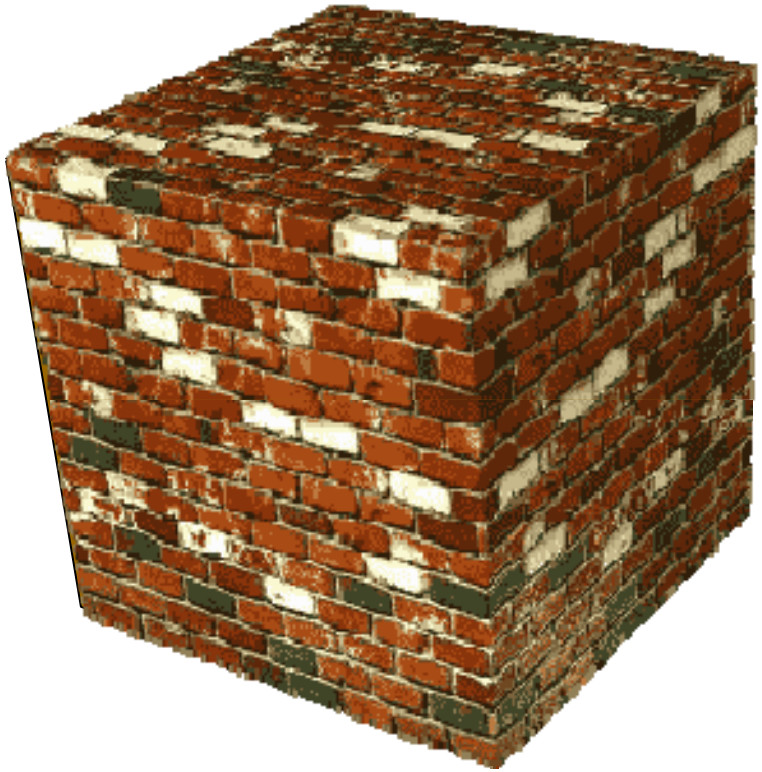




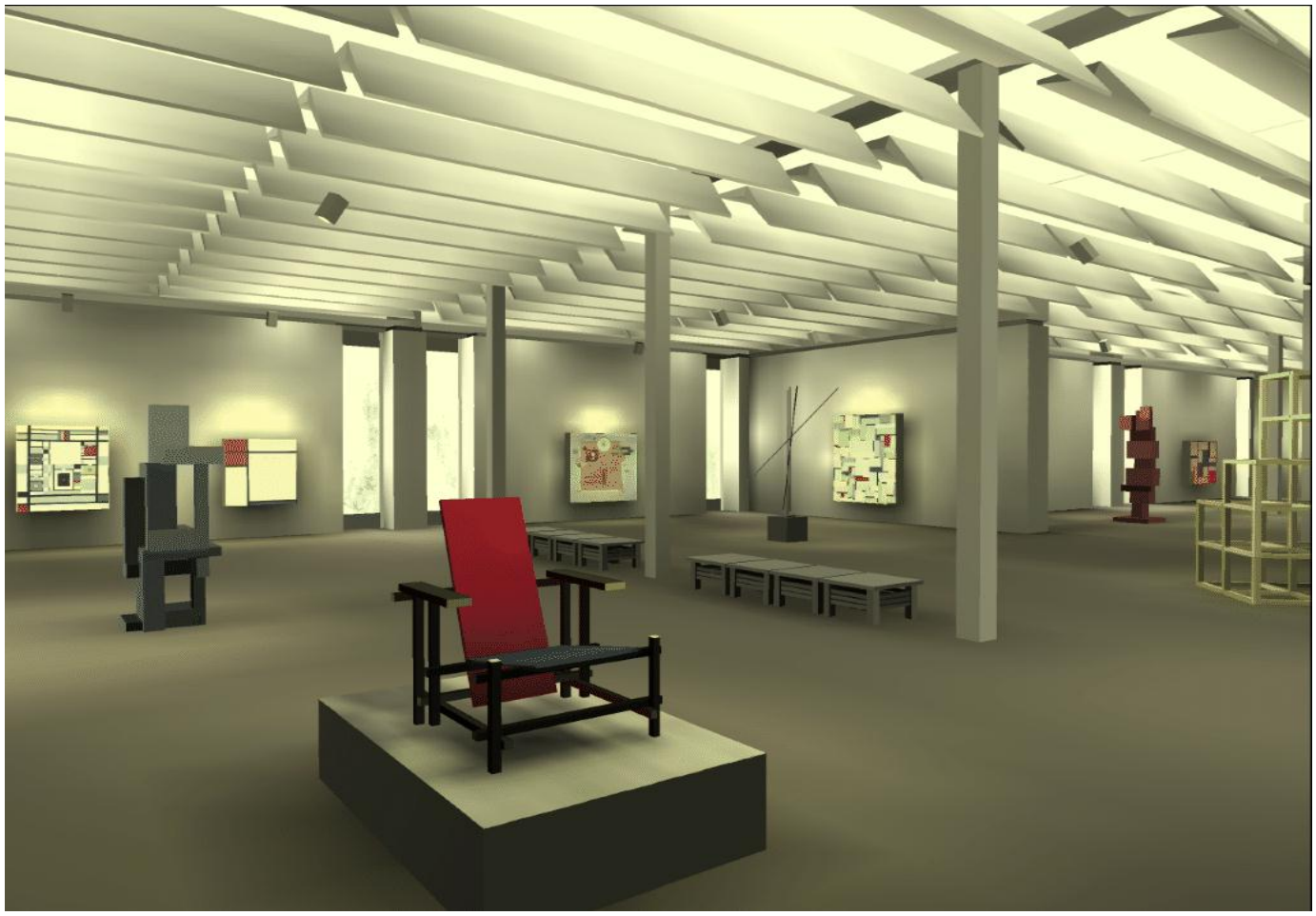
Culling



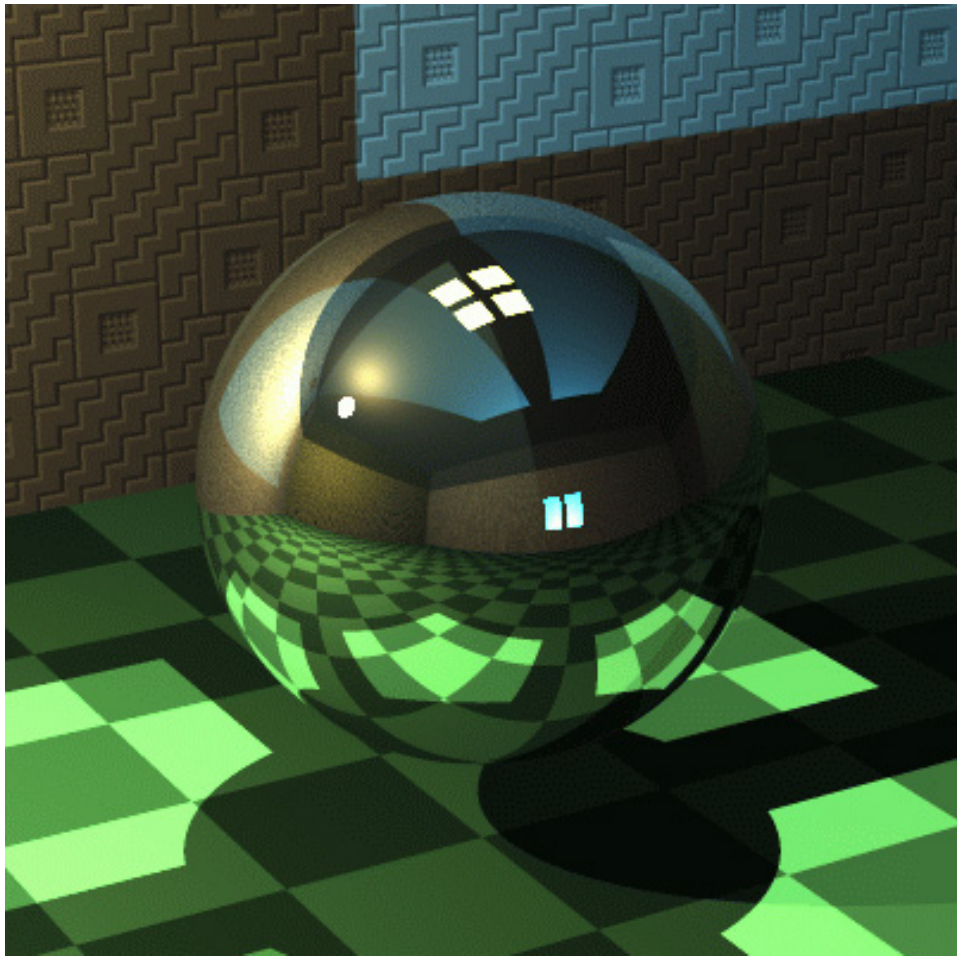
Texturing



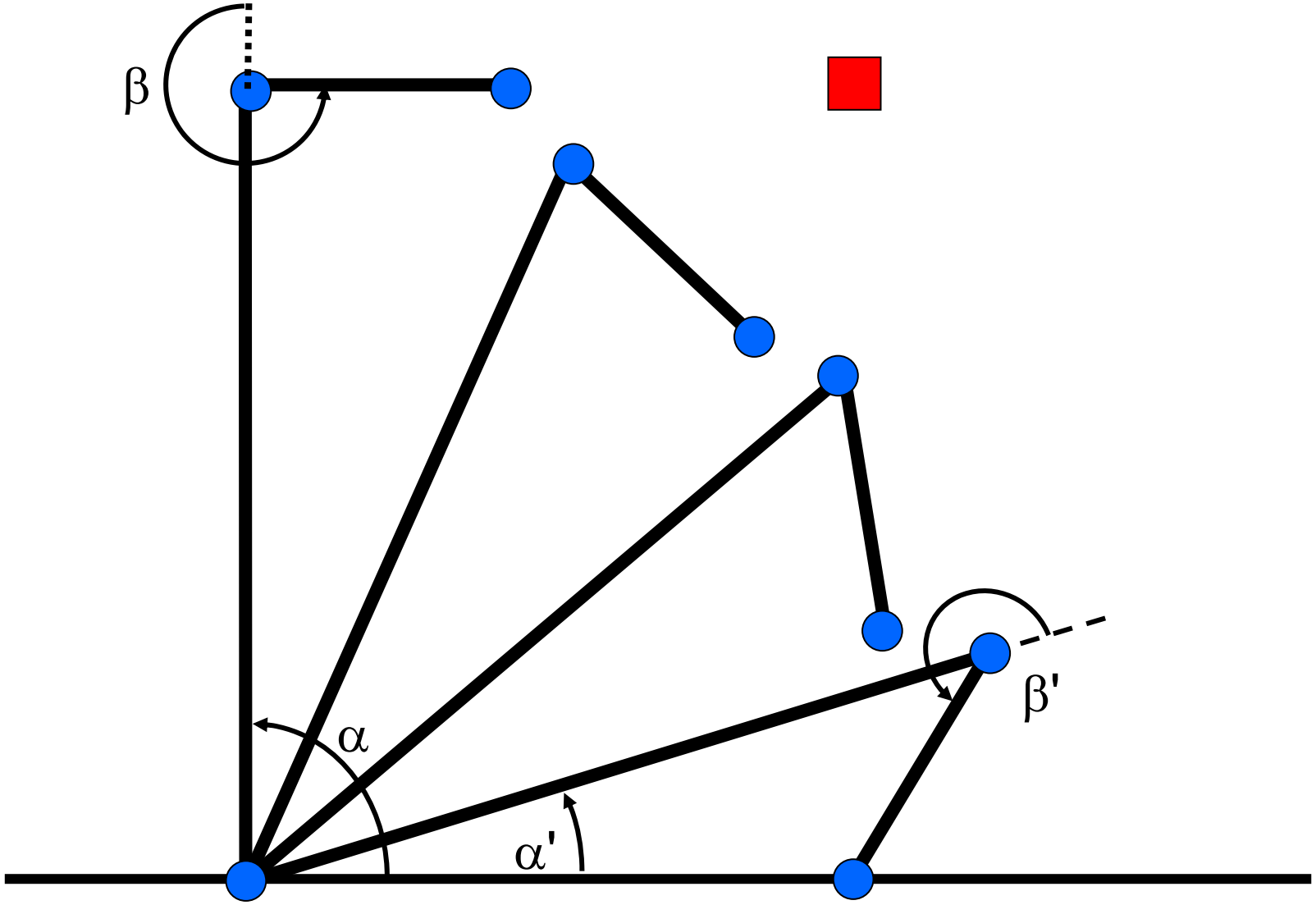
Radiosity



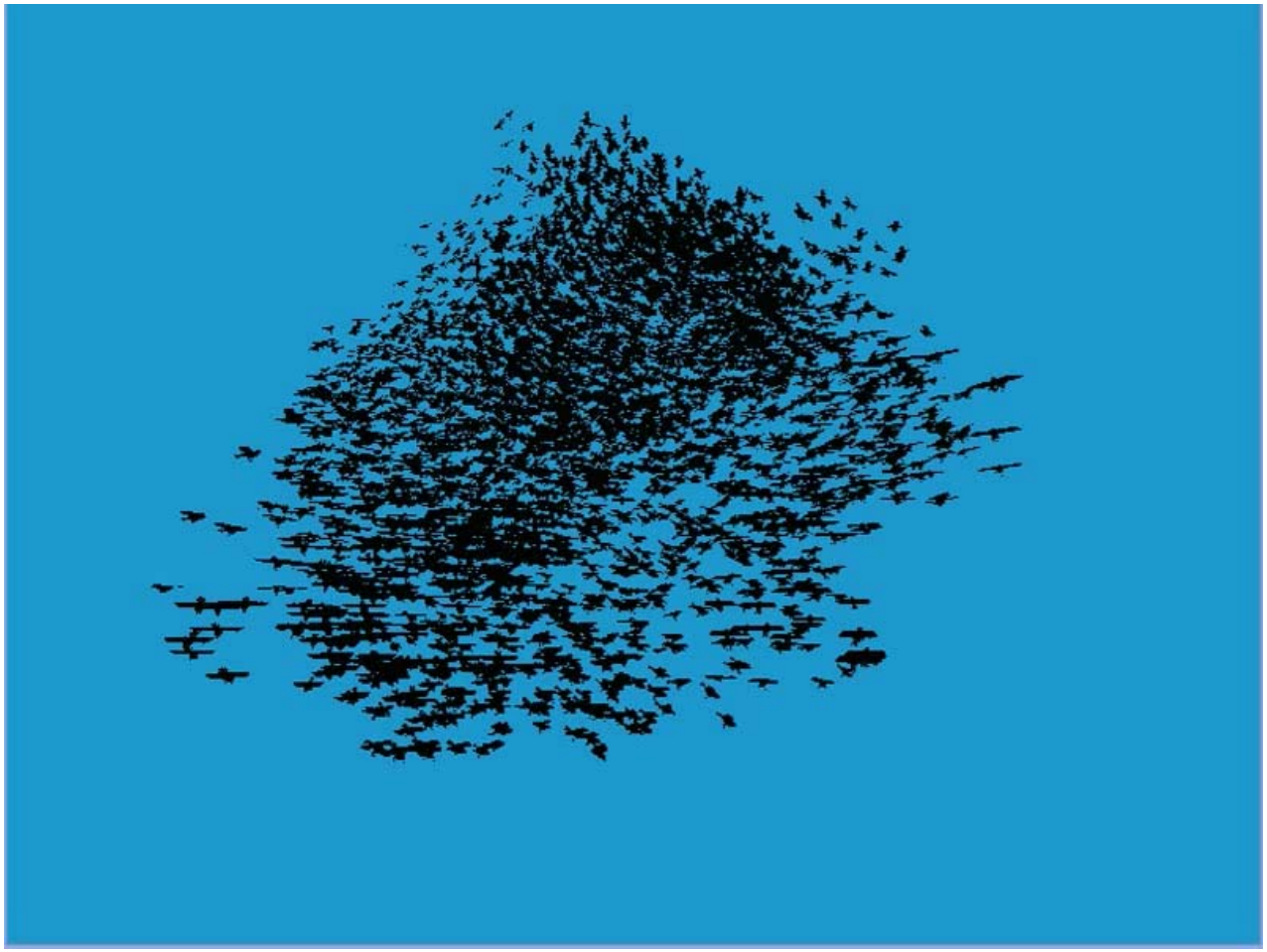
Ray Tracing



Animation / Inverse Kinematics



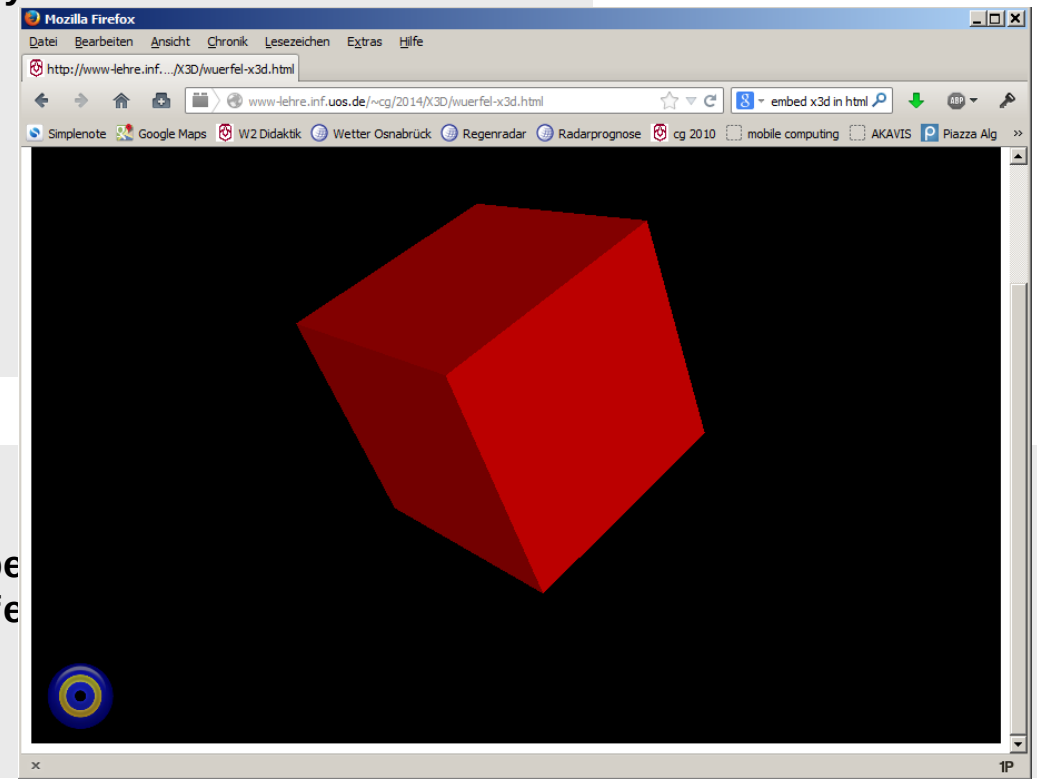
Partikelsysteme / Verhaltensanimation



Bachelorarbeit von Oliver Tschesche

X3D

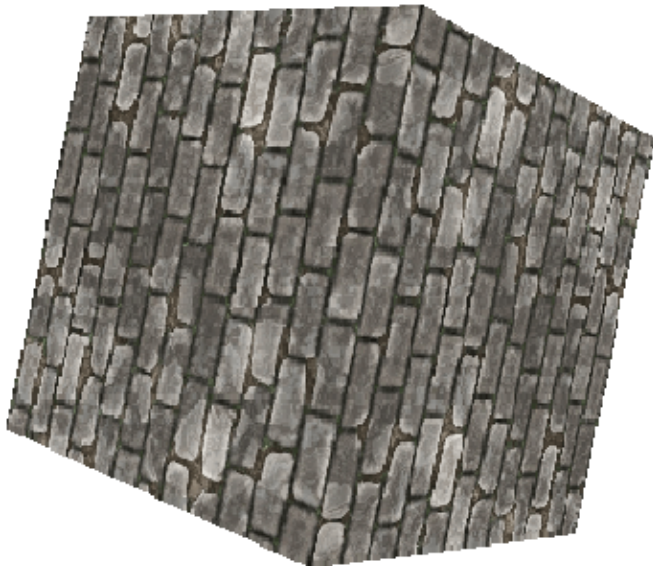
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
  "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D>
  <Scene>
    <Transform translation= "-0.03   0.00   -0.052"
               rotation=    " 0.82   -0.56   -0.039   2.10">
    <Shape>
      <Appearance>
        <Material ambientIntensity  ="0.2"
                  shininess
                  diffuseColor
       </Appearance>
       <Box size="1 1 1"/>
    </Shape>
    </Transform>
  </Scene>
</X3D>
```




```
<html>
  <body>
    <object data="wuerfel.x3d" type
    <param name="src" value="wuerfe
    </object>
  </body>
</html>
```

http://www-lehre.inf.uos.de/~cg/2014/X3D/wuerfel-x3d.html

OpenGL

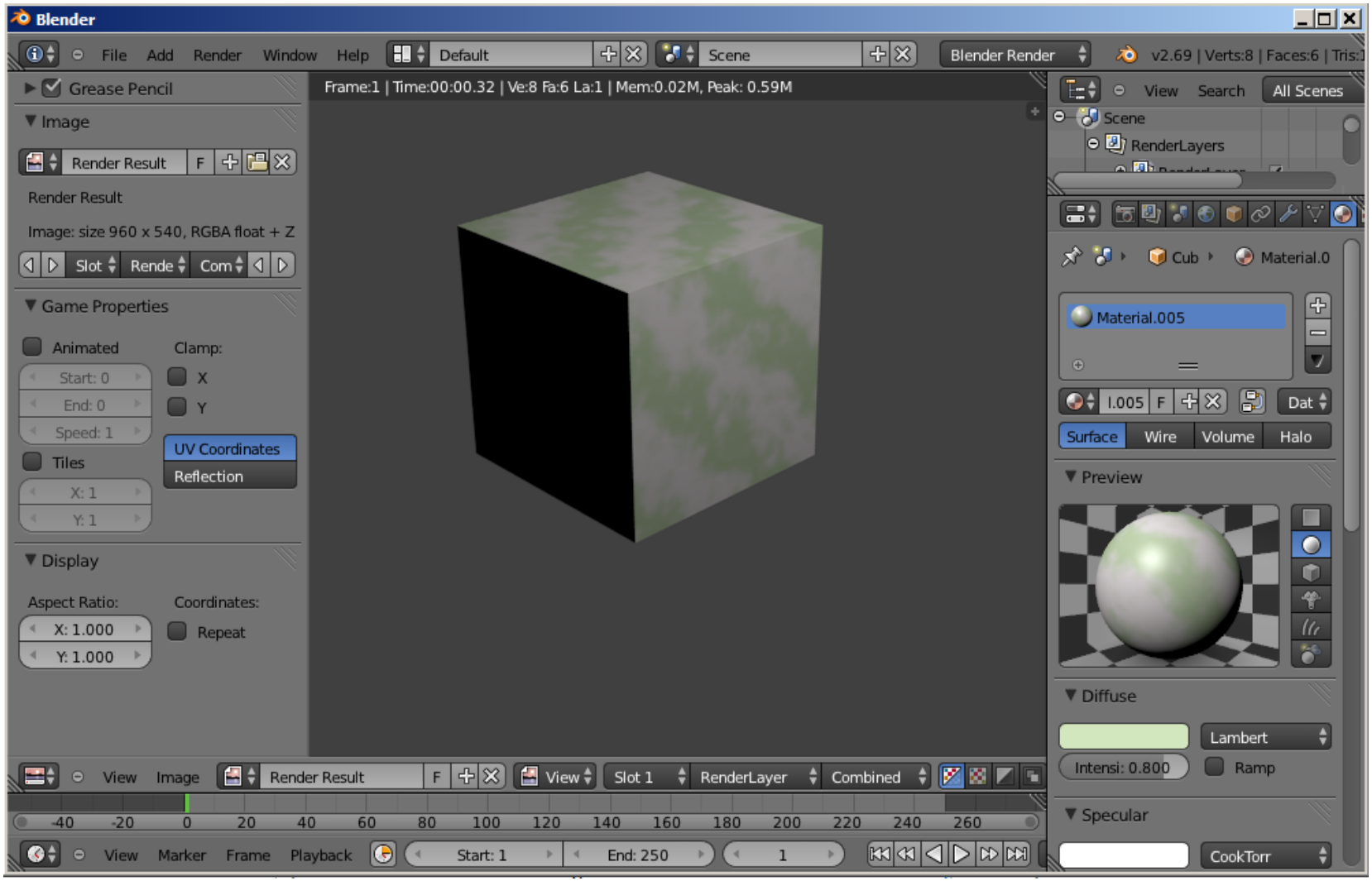


```
/* Auflösung */
    static int g_w = 1024;
    static int g_h = 768;

    /* für die Rotation */
    static float g_ry = (float)Math.PI * 0.25f;
    static float g_rx = (float)Math.PI * 0.1f;
    static boolean g_bAnimate = false;

    static long time0 = 0;
    static long time1 = 0;

    static FloatBuffer MATRIX_BUFFER
        = BufferUtils.createFloatBuffer(16);

    static void init()
    {
        glEnable(GL_CULL_FACE);
        glEnable(GL_DEPTH_TEST);
        glDepthFunc(GL_LEQUAL);
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        glClearColor(0.0f, 0.0f, 1.0f, 0.0f);

        createShaderProgram();
        createProjection((float)Math.PI * 0.5f, g_w / (float)g_h, 1e-2f, 1e3f);
        createView(0, 0, -5, g_rx, g_ry);
        createCube();
        createTexture();

        glUniform1i(glGetUniformLocation(g_shaderProgram, "g_texture"),
```

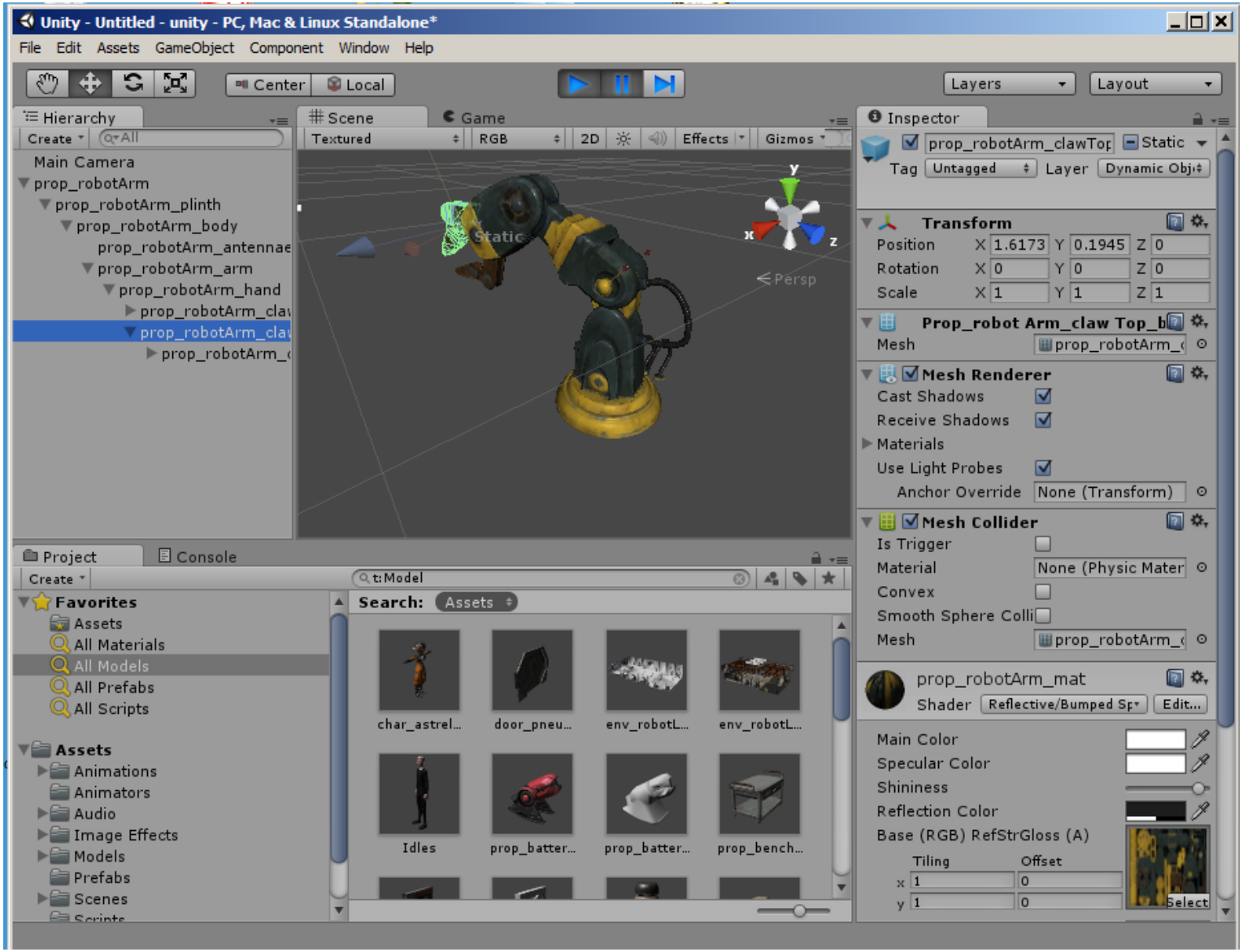
WebGL

Bachelorarbeit von Timo Bourdon:
http://www.informatik.uni-osnabrueck.de/prakt/pers/dipl/bourdon.php

Blender


Blender
File Add Render Window Help
Default
Scene
Blender Render
v2.69 | Verts:8 | Faces:6 | Tris:1
Grease Pencil
Image
Render Result
Render Result
Image: size 960 x 540, RGBA float + Z
Slot Rende Com
Game Properties
Animated
Clamp:
Start: 0
End: 0
Speed: 1
X
Y
Tiles
UV Coordinates
Reflection
X: 1
Y: 1
Display
Aspect Ratio:
Coordinates:
X: 1.000
Y: 1.000
Repeat
Frame:1 | Time:00:00.32 | Ve:8 Fa:6 La:1 | Mem:0.02M, Peak: 0.59M
View Search All Scenes
Scene
RenderLayers
Cub
Material.0
Material.005
Surface Wire Volume Halo
Preview
Diffuse
Lambert
Intensi: 0.800
Ramp
Specular
CookTorr
View Image
Render Result
View
Slot 1
RenderLayer
Combined
-40 -20 0 20 40 60 80 100 120 140 160 180 200 220 240 260
View Marker Frame Playback
Start: 1
End: 250
1

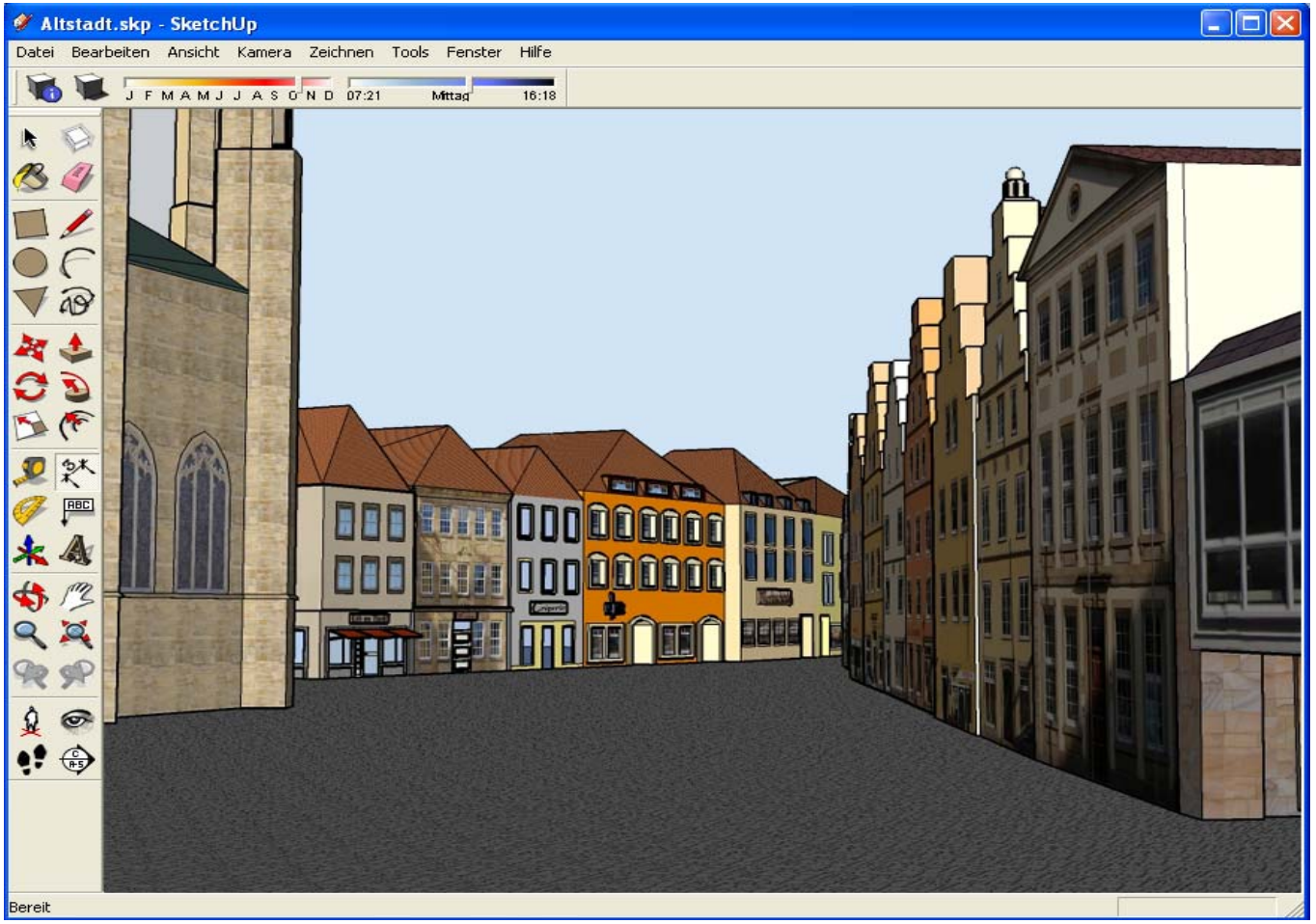
Unity 3D


Unity - Untitled - unity - PC, Mac & Linux Standalone*
File Edit Assets GameObject Component Window Help
Center Local
Layers Layout
Hierarchy
Create All
Main Camera
prop_robotArm
prop_robotArm_plinth
prop_robotArm_body
prop_robotArm_antennae
prop_robotArm_arm
prop_robotArm_hand
Scene Game
Textured RGB 2D Effects Gizmos
Static
Persp
Inspector
prop_robotArm_clawTop Static
Tag Untagged Layer Dynamic Obj
Transform
Position X 1.6173 Y 0.1945 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Prop_robot Arm_claw Top_b
Mesh prop_robotArm_
Mesh Renderer
Cast Shadows
Receive Shadows
Materials
Use Light Probes
Anchor Override None (Transform)
Mesh Collider
Is Trigger
Material None (Physic Mater
Convex
Smooth Sphere Colli
Mesh prop_robotArm_
Project Console
Create t:Model
Favorites
Assets
All Materials
All Models
All Prefabs
All Scripts
Assets
Animations
Animators
Audio
Image Effects
Models
Prefabs
Scenes
Search: Assets
char_astrel...
door_pneu...
env_robotL...
env_robotL...
Idles
prop_batter...
prop_batter...
prop_bench...
prop_robotArm_mat
Shader Reflective/Bumped Sp Edit...
Main Color
Specular Color
Shininess
Reflection Color
Base (RGB) RefStrGloss (A)
Tiling Offset
x 1 0
y 1 0
Select

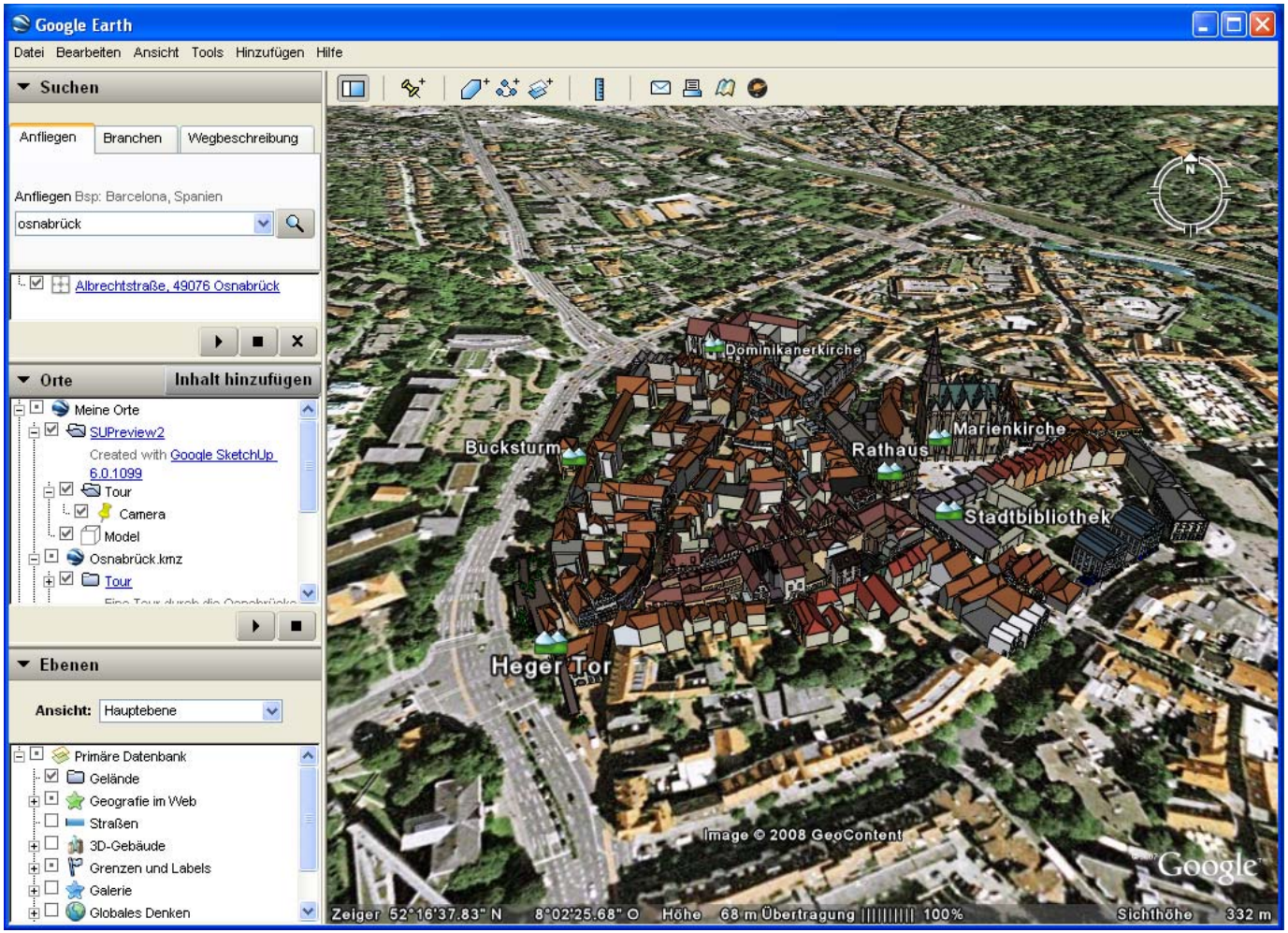
Unity3d Player

http://www.online3dgames.net/games/458/play-lose-the-heat-3-highway-hero

Google SketchUp



Google Earth

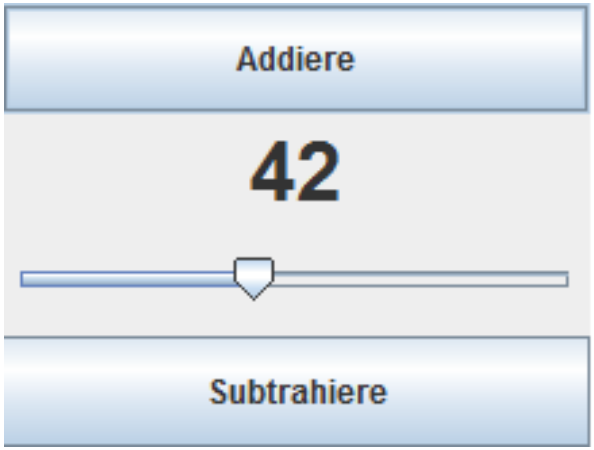


Computergrafik 2014

Oliver Vornberger

Kapitel 02:
Grafische Benutzeroberflächen

RaufRunterApplet



http://www-lehre.inf.uos.de/~cg/2014/skript/Applets/raufRunter/App.html

GUI-Programmierung

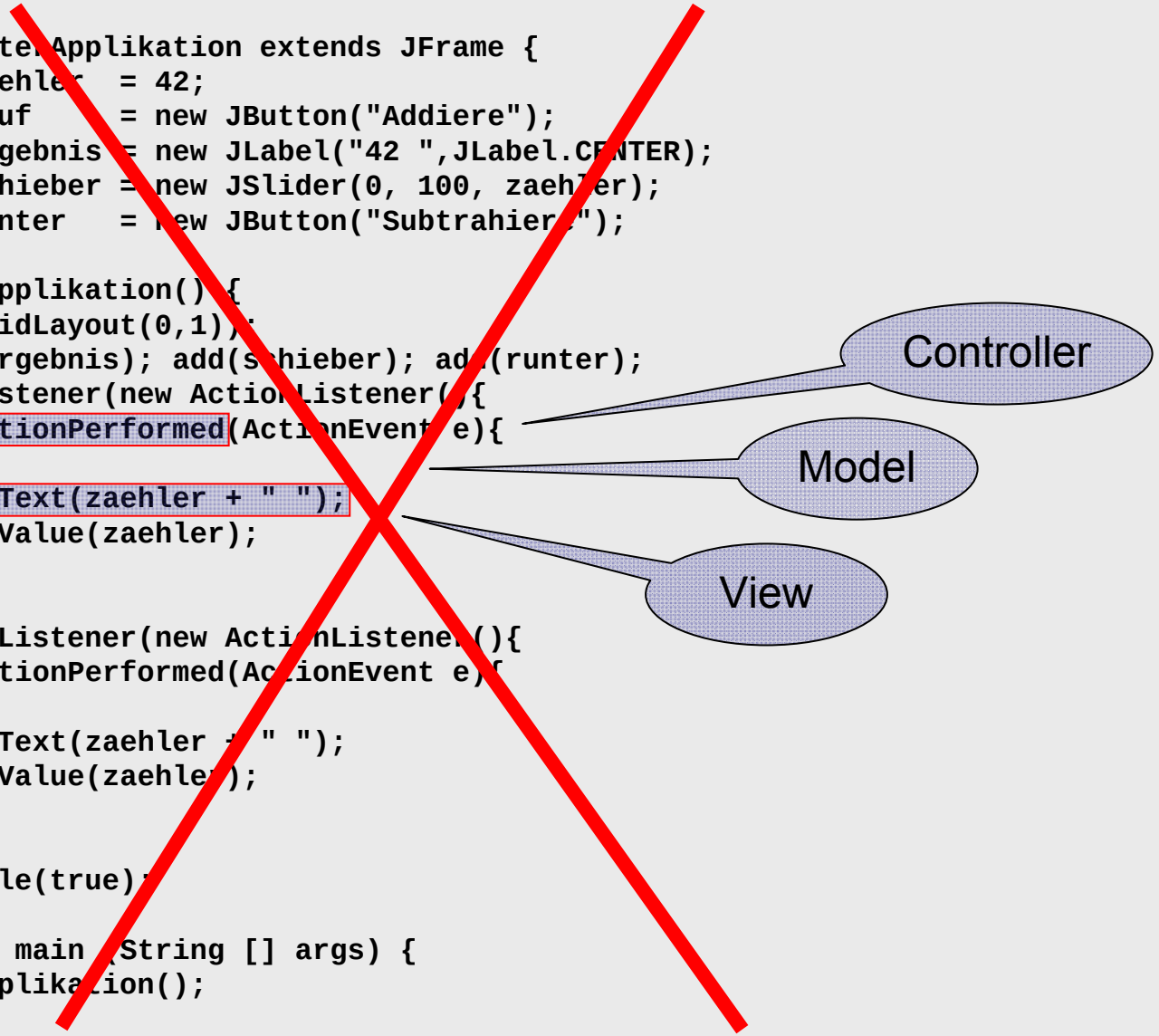
- Windowmanager
- AWT (Abstract Window Toolkit)
  - reicht Kommandos weiter an Betriebssystem
  - plattformabhängig
- Swing
  - pure Java
  - einheitliches Look & Feel

GUI-Komponenten

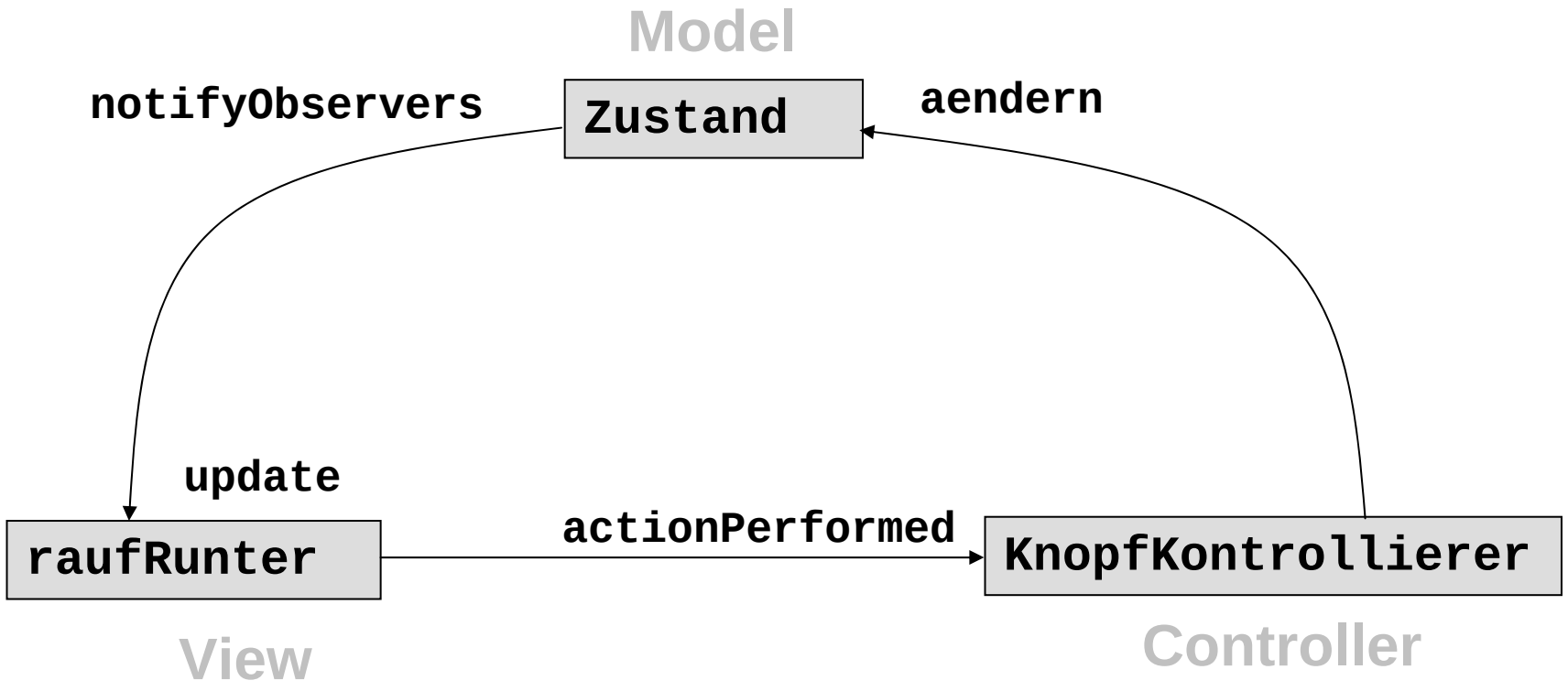
- JFrame
- JPanel
- GridLayout
- JButton
- JLabel
- JSlider
- ActionListener
- actionPerformed

RaufRunterApplikation.java

```
import java.awt.*; import java.awt.event.*; import javax.swing.*;

public class RaufRunterApplikation extends JFrame {
  private int     zaehler  = 42;
  private JButton rauf     = new JButton("Addiere");
  private JLabel  ergebnis = new JLabel("42 ",JLabel.CENTER);
  private JSlider schieber = new JSlider(0, 100, zaehler);
  private JButton runter   = new JButton("Subtrahiere");

  public RaufRunterApplikation() {
    setLayout(new GridLayout(0,1));
    add(rauf); add(ergebnis); add(schieber); add(runter);
    rauf.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e){
        zaehler++;
        ergebnis.setText(zaehler + " ");
        schieber.setValue(zaehler);
      }
    });
    runter.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e){
        zaehler--;
        ergebnis.setText(zaehler + " ");
        schieber.setValue(zaehler);
      }
    });
    pack(); setVisible(true);
  }
  public static void main (String [] args) {
    new RaufRunterApplikation();
  }
}
```



Model-View-Controller



Zustand.java

```
import java.util.Observer;
import java.util.Observable;

public class Zustand extends Observable{

  private int zaehler;

  public Zustand(int zaehler){
    this.zaehler=zaehler;
  }

  int get(){return zaehler;}

  void aendern(int delta){
    zaehler = zaehler + delta;
    setChanged();
    notifyObservers();
  }
}
```

KnopfKontrollierer.java

```
import java.awt.*;
import java.awt.event.*;

public class KnopfKontrollierer implements ActionListener {

  private Zustand z;
  private int delta;

  public KnopfKontrollierer(Zustand z, int delta) {
    this.z       = z;
    this.delta   = delta;
  }

  public void actionPerformed(ActionEvent e) {
    z.aendern(delta);
  }
}
```

RaufRunter.java, Teil 1

```
import java.util.*;
import java.awt.*;
import javax.swing.*;

public class RaufRunter extends JPanel implements Observer {

  private JButton rauf;
  private JButton runter;
  private Zustand z;
  private JLabel  ergebnis;
  private JSlider schieber;
  private Font    font;


 public void update(Observable z, Object dummy){
   ergebnis.setText(((Zustand)z).get() + " ");
   schieber.setValue(((Zustand)z).get());
}
```

RaufRunter.java, Teil 2

```
public RaufRunter() {
    setLayout(new GridLayout(0,1));
    rauf   = new JButton("Addiere");
    runter = new JButton("Subtrahiere");
    schieber = new JSlider(0,100,42);
    ergebnis = new JLabel("42",JLabel.CENTER);
    font = new Font("SansSerif",Font.BOLD,30);
    ergebnis.setFont(font);
    add(rauf);
    add(ergebnis);
    add(schieber);
    add(runter);
    z = new Zustand(42);
    z.addObserver(this);
    KnopfKontrollierer raufK;
    raufK = new KnopfKontrollierer(z,+1);
    rauf.addActionListener(raufK);
    KnopfKontrollierer runterK;
    runterK = new KnopfKontrollierer(z,-1);
    runter.addActionListener(runterK);
  }
}
```

RaufRunterApp.java

```
import java.awt.BorderLayout;
import javax.swing.JFrame;

public class RaufRunterApp {

  public static void main(String args[]) {
    JFrame rahmen = new JFrame("RaufRunter-Applikation");
    rahmen.add(new RaufRunter(),BorderLayout.CENTER);
    rahmen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    rahmen.pack();
    rahmen.setVisible(true);
  }

}
```

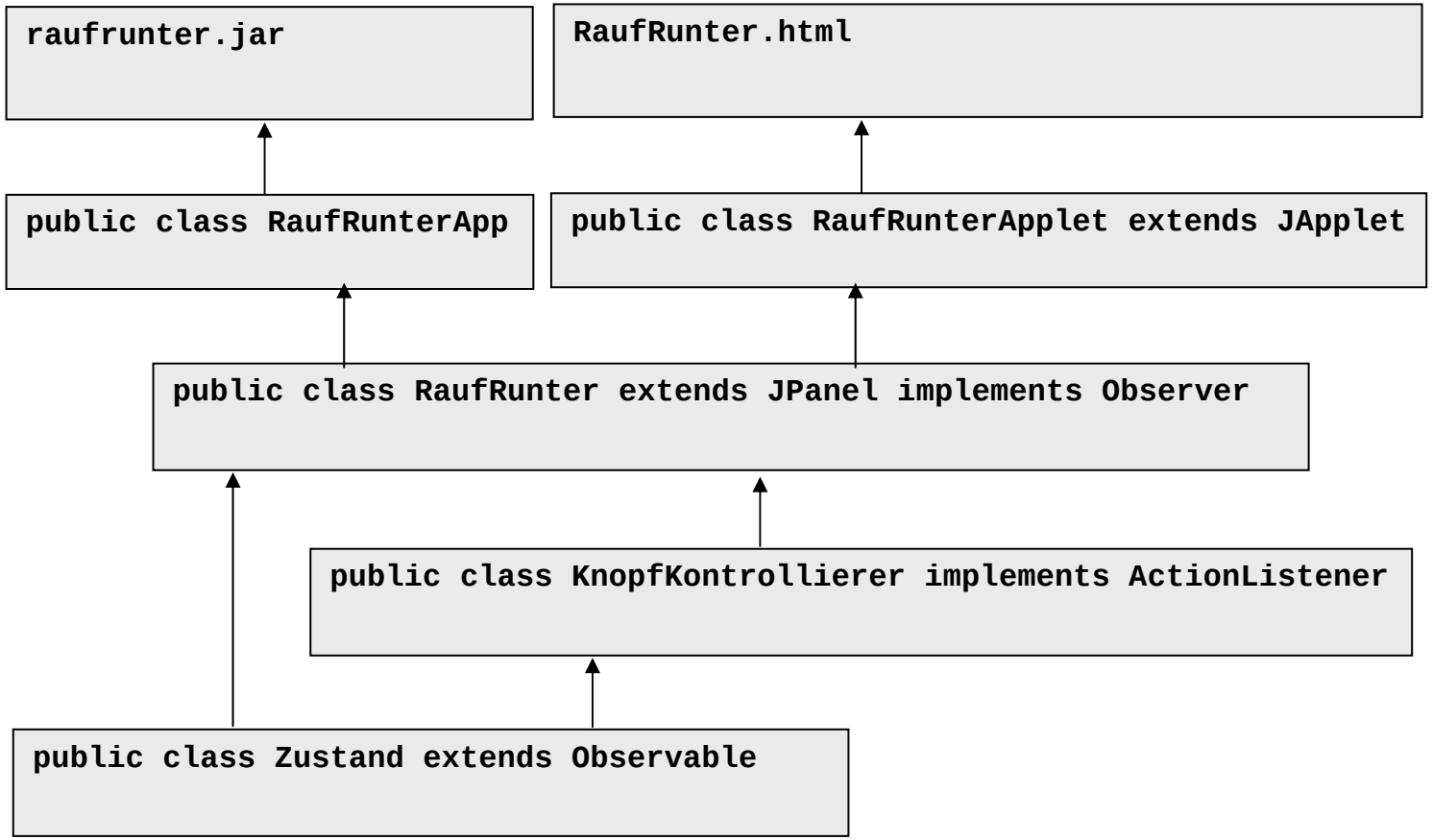
RaufRunterApplet.java

```
import java.awt.BorderLayout;
import javax.swing.JApplet;

public class RaufRunterApplet extends JApplet {

  public void init() {
    add(new RaufRunter(), BorderLayout.CENTER);
  }
}
```

RaufRunter.html

```
<HTML>
  <HEAD>
    <TITLE>RaufRunter-Applet</TITLE>
  </HEAD>
  <BODY>
    <CENTER>
    <H1>RaufRunter-Applet</H1>
    <APPLET
       width   = 200
       height  = 150
       code    = "RaufRunterApplet.class"
       archive = "raufRunter.jar">
    </APPLET>
    </CENTER>
  </BODY>
</HTML>
```

Dateisystem



http://www-lehre.inf.uos.de/~cg/2014/skript/Applets/raufRunter/App.html