TECHNISCHE UNIVERSITÄT DRESDEN

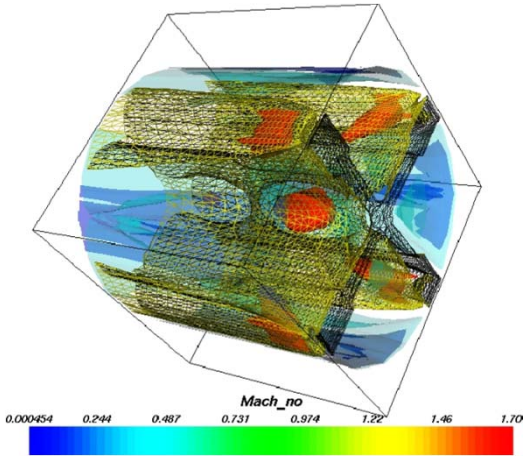
Fakultät Maschinenwesen Institut für Verarbeitungsmaschinen und mobile Arbeitsmaschinen

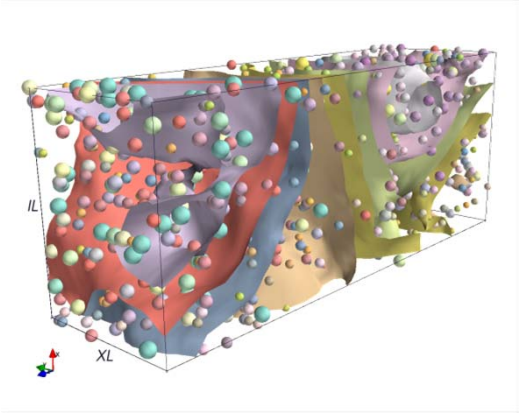
3D Visualisierung mit Python

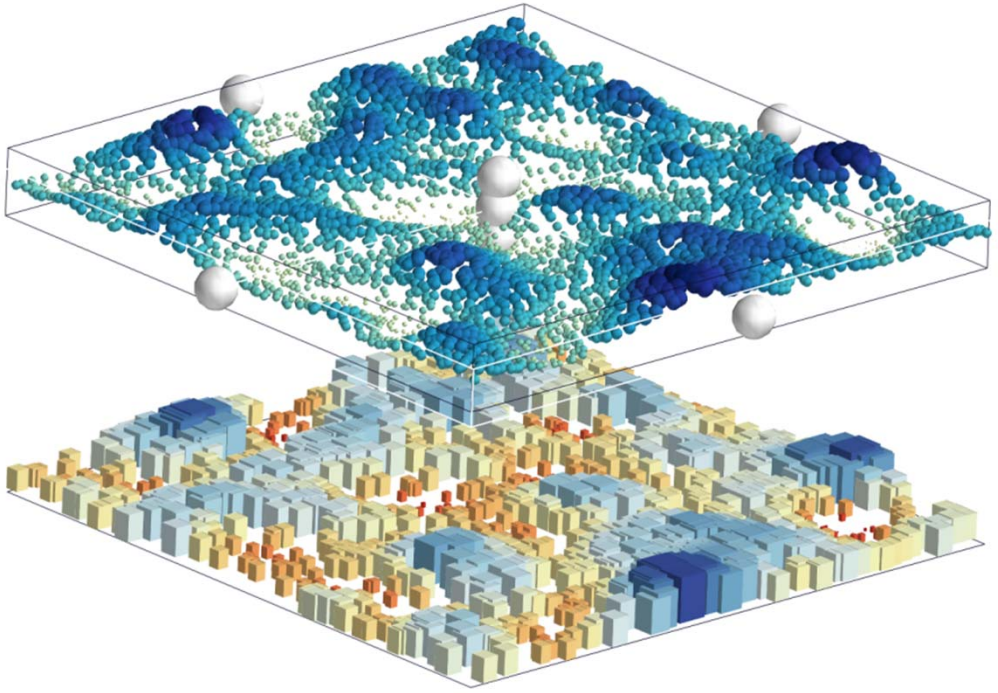
04.07.2011 Sebastian Voigt

Wie was wo warum? - I

3D-Visualisierung von Messdaten, Simulationsergebnissen etc.

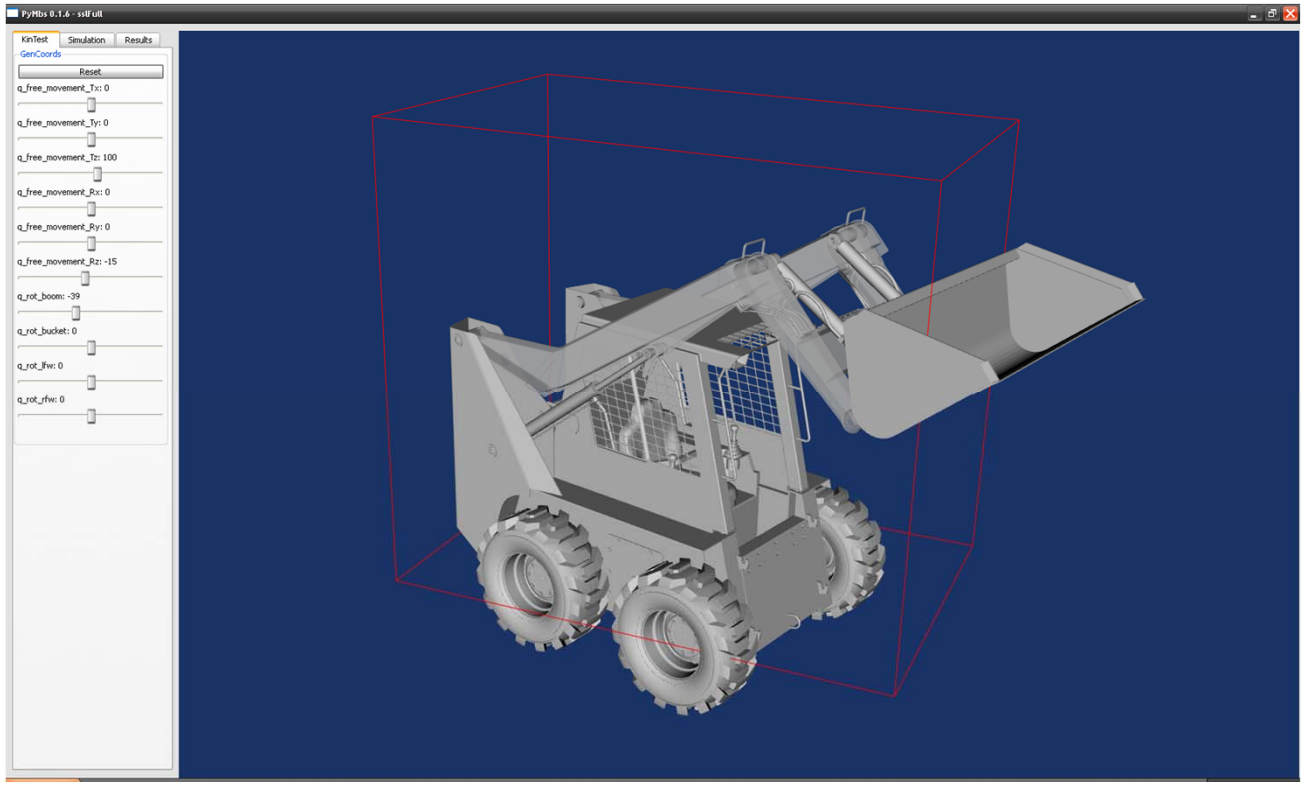






Wie was wo warum? - II

3D-Szenen mit Körpern, Beleuchtung, Animationen etc.



Python Pakete 3D-Visualisierung

- PyOpenGL - http://pyopengl.sourceforge.net/
- Mayavi2 - http://github.enthought.com/mayavi/mayavi/
- Vtk - http://www.vtk.org/
- Vpython - http://www.vpython.org/
- Python-Ogre - http://www.pythonogre.com/
- OpenGL Modul in PyQt
- Wrapper für allerlei Scenegraphen, z.B. osgpython

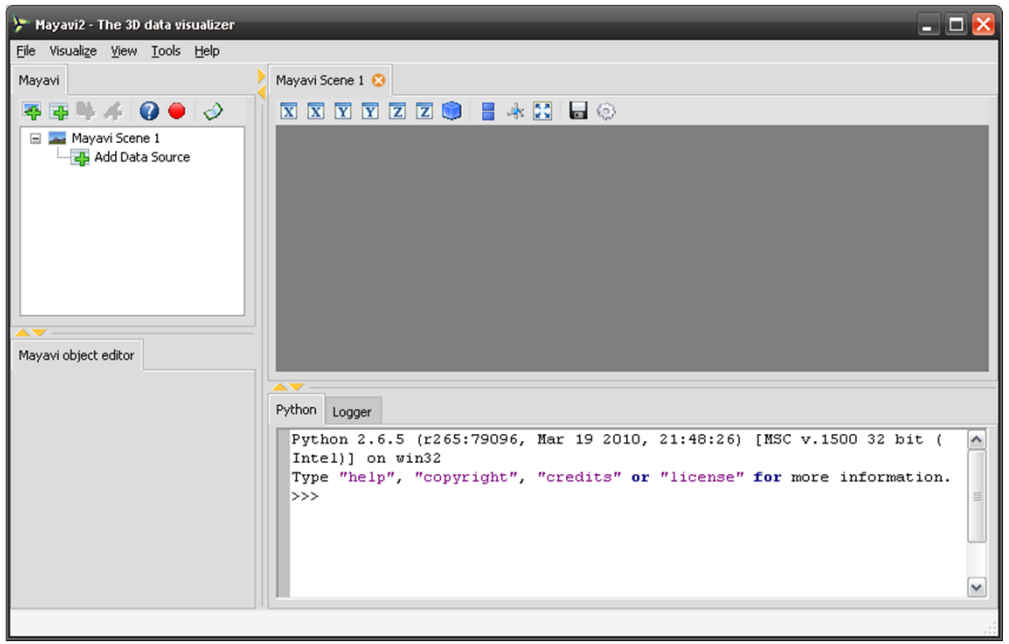
Blender – teils in Python geschrieben, umfangreiche Python Skripting API

Mayavi2 - Einführung

- Kann als standalone Anwendung oder per Skript genutzt werden
- Intern auf numpy arrays basierend
- Sehr mächtig und umfangreich

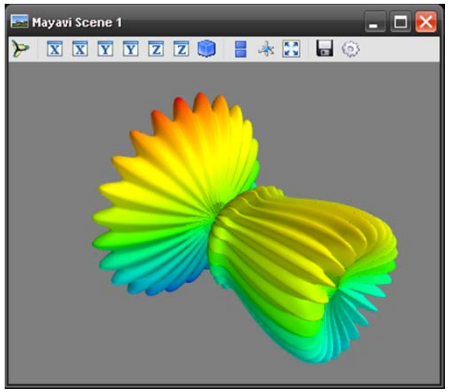
Mayavi2 als Applikation starten:

- Konsole öffnen
- mayavi2 eingeben und Enter



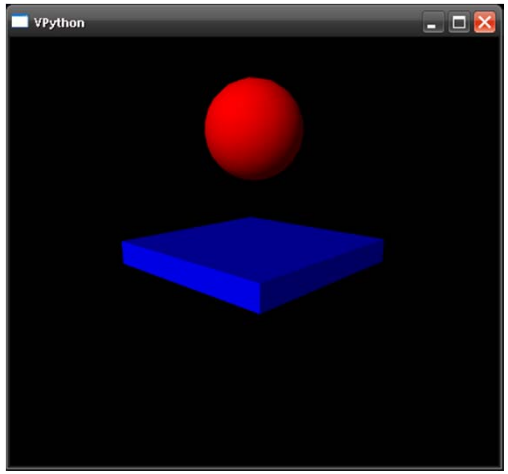
Mayavi - mlab

```
from numpy import pi, sin, cos, mgrid
from enthought.mayavi import mlab


# Daten erzeugen
dphi, dtheta = pi/250.0, pi/250.0
[phi,theta] = mgrid[0:pi+dphi*1.5:dphi,0:2*pi+dtheta*1.5:dtheta]

m0 = 4; m1 = 3; m2 = 2; m3 = 3; m4 = 6; m5 = 2; m6 = 6; m7 = 4;
r = sin(m0*phi)**m1+cos(m2*phi)**m3+sin(m4*theta)**m5+cos(m6*theta)**m7

x = r*sin(phi)*cos(theta)
y = r*cos(phi)
z = r*sin(phi)*sin(theta)

# anschauen
s = mlab.mesh(x, y, z)
mlab.show()
```



Vpython - Simulationsumgebung

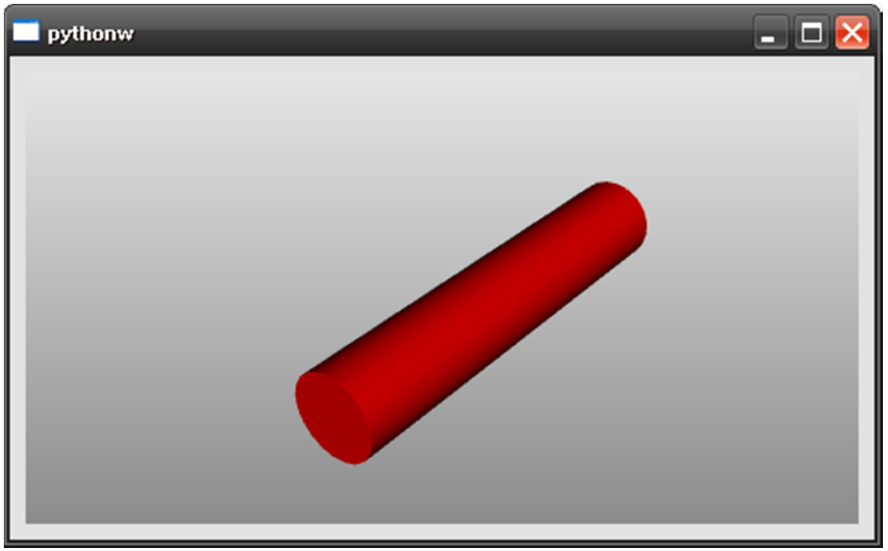
```
from visual import *

# Koerper anlegen
floor = box(pos=(0,0,0), length=4, height=0.5, width=4, color=color.blue)
ball = sphere(pos=(0,4,0), radius=1, color=color.red)

# Startwert fuer Geschwindigkeit und Schrittweite
ball.velocity = vector(0,-1,0)
dt = 0.01

# Simulation
while 1:
    rate(100)
    ball.pos = ball.pos + ball.velocity*dt
    if ball.y < ball.radius:
        ball.velocity.y = abs(ball.velocity.y)
    else:
        ball.velocity.y = ball.velocity.y - 9.81*dt
```



Vtk

- Highlevel-Wrapper für OpenGL
- Unterbau fuer Mayavi2
- Hilfe: http://www.vtk.org/VTK/help/documentation.html
- Classdocs: http://www.vtk.org/doc/release/4.0/html/hierarchy.html
- Importer u.a. für 3D-Formate (obj, 3ds, vrml, stl, ...)

- Beispiel: vtkbasic.py



Optische Eigenschaften werden über das Property-Objekt des actors eingestellt:

```
propObj = actor.GetProperty()
```

Transparenz:
```
propObj.SetOpacity(0.5)
```
Farbe:
```
propObj.SetColor(0,0,1)
```
→ rgb

Reflexion:
```
propObj.SetAmbient(0.2)
propObj.SetDiffuse(0.8)
propObj.SetSpecular(0.5)
propObj.SetSpecularPower(0.5)
```
Kanten sichtbar:
```
propObj.SetEdgeVisibility(1)
```

Größe:
```
propObj.SetScale(0.1)
```

Texturen gehen auch, leider kein Beispiel ☹

Vtk - Geometrieprimitive

```
part = vtk.vtkCubeSource()
part.SetXLength(10)
part.SetYLength(1)
part.SetZLength(1)
```

```
part = vtk.vtkSphereSource()
part.SetRadius(5)
part.SetThetaResolution(20)
part.SetPhiResolution(20)
```

```
part = vtk.vtkLineSource()
part.SetPoint1(0,0,0)
part.SetPoint2(10,0,0)
```

```
part = vtk.vtkTextSource()
part.SetText('Hallo Welt')
```

```
part = vtk.vtkConeSource()
part.SetHeight(10)
part.SetRadius(5)
part.SetResolution(20)
part.SetAngle(30)
part.Capping(2)
```

Vtk – Actors positionieren

- Via Poke-Matrix:

```
P = [ . . . . ]
    [ . R . p ]
    [ . . . . ]
    [ 0 0 0 1 ]

R = eye(3)        → keine Rotation
p = [1,0,0]       → Verschiebung in x-Richtung

poke = vtk.vtkMatrix4x4()
for i in range(3):
   for j in range(3):
      poke.SetElement(i,j, R[i,j])
   poke.SetElement(i,3, p[i])
actor.PokeMatrix(poke)
```

→

```
[1 0 0 0
 0 1 0 0
 0 0 1 0
 0 0 0 1]
```