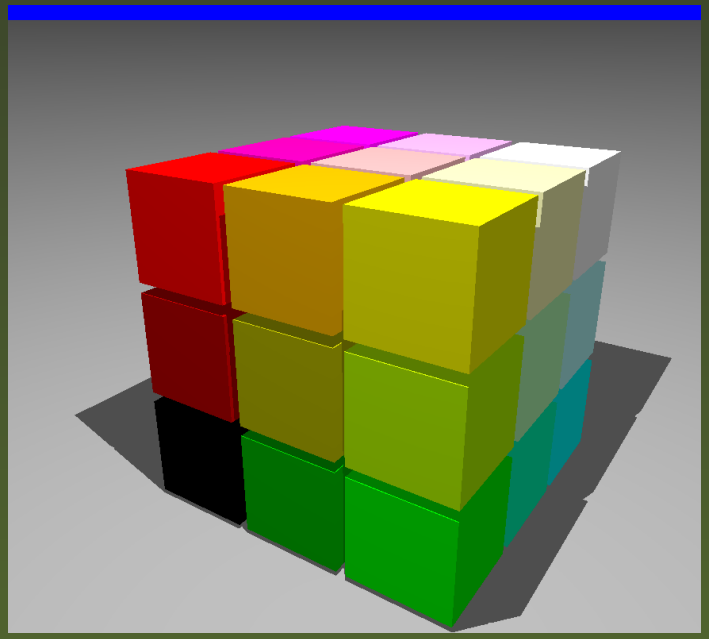
3D-modelling and the Pycao software

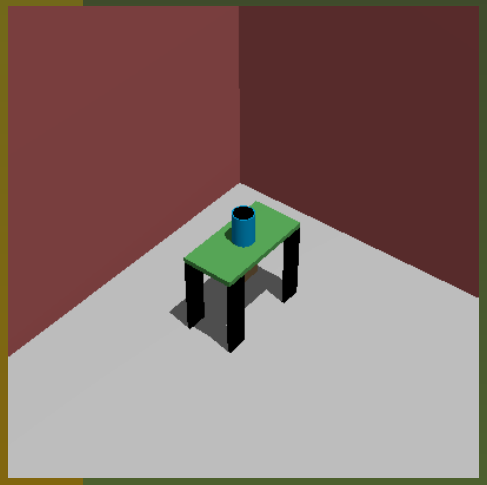
Laurent Evain



3D-modelling versus 2D-modelling

- Description of the scene as 3D objects
- Possibility of extraction of 2D information (Example: photo)
-

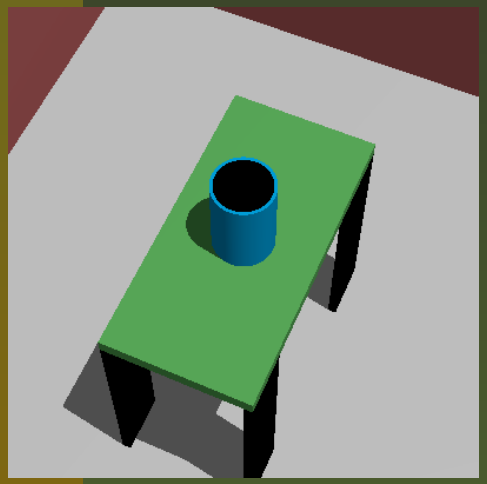
```
camera.location=origin+5*Y+6*Z+6.2*X
camera.lookAt=tableTray.center
```

- 

3D-modelling versus 2D-modelling

- Same description of the scene
-

```
camera.location=origin+2*Y+4*Z+3.2*X
camera.lookAt=tableTray.center
camera.zoom(.881)
```

- 

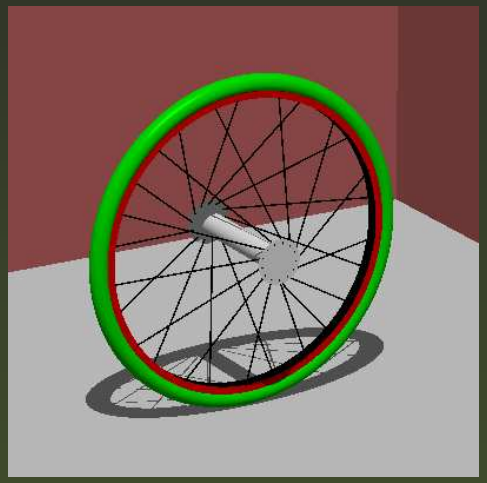
Main applications

- CAD software in industry
- Engineering (Example: thermal study of a house)
- Mathematical illustration (Ex: 3D-combinatorics)
- Mathematical verification (Ex: stability of a bike, temperature evolution...)
- Movies, Architecture

Description of a scene

- Difficult, whatever the tool
- Several reasons:
  - $Isom(\mathbb{R}^3) \neq Isom(\mathbb{R}^2)$
  - $2D$ screen
  - Integrated tools (versatility vs simplicity)
  - Documentation (povray vs Blender)
  - New paradigms required
- One year to master cad software - 2 weeks for basic skills

Graphical tools



- Time consuming (exemple: simple wheel)
- 2 clicks for a point, (de)zooming, unseen pieces
- Required precision: movie vs science
- Interfacing with Numpy/scipy.
- Latex vs Word

Code with coordinates

- Povray, .obj format
-

```
//Unnamed Object
plane {
<0.0,1.0,0.0>,0.0  material{texture{ pigment {color Brown} finish{metallic phong 1} }}
 }

//Unnamed Object
box {
<0.0,0.0,0.0>,<1.0,0.5,0.05>  material{texture{ pigment {color ForestGreen} finish{metallic phong 1} }}
 matrix <1.0 , 0.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.0 , 0.0 , 1.0 , 1.0 , 1.0 , 0.8>}



//Unnamed Object
box {
<0.0,0.0,0.0>,<0.2,0.03,0.8>  material{texture{ pigment {color Copper} finish{metallic phong 1} }}
 matrix <1.0 , 0.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.0 , 0.0 , 1.0 , 1.0 , 1.0 , 0.0>}
```

- Difficult to write, to read, to maintain
- Ray tracer $\neq$ Modeller

Towards a short description

- Mathematical proof with $\forall$, $\exists$, ...
- What is the shortest proof ?
- Description of a 3D scene with words
- What is the shortest code ?
- **Pycao Objective:** shortest possible description (time, length)

Architecture to not reinvent the wheel

3d- Modelisation with Pycao

⇓

Object obtained after compilation

⇓

Pycao Plugin to the mathematical software

⇓

Thermal study, visualisation ...

Paradigms of Pycao

- **Pycao Objective:** shortest possible description (time, length)
- Choices
  - Mathematical affine geometry framework
  - Coordinate free
  - Python
  - Dynamical description
  - Box paradigm
  - Markers, CSG

The massic space

- unification of points and vectors
- massic point $p = (x, y, z, w)$
- $w = 0 : \ vector, w = 1 : \ point$
- internally: all points are massic points
- $0.7 * p_1 + v + 0.3 * p_0 = point.$

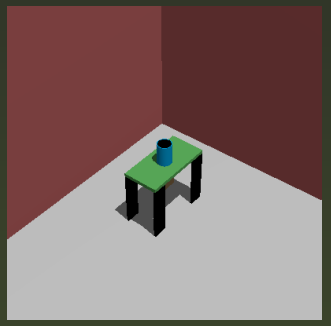
The massic space for the end-user

- Transparent use of the language of affine geometry
- $p_1 - p_2,\ 0.3 * p_1 + 0.2 * p_2 + 0.5 * p_3$ known
- consistency verification by the compiler
- absolute and relative definition:
  $[p_1, p_2, p_3, p_4] \simeq [p_1, v, p3, w]$ if $v = p2 - p1$,
  $w = p4 - p3$.
- implicit conversion. f affine, $f(v) = \vec{f}(v)$.

The massic space for the developper

- Unification of the affine and vectorial base changes

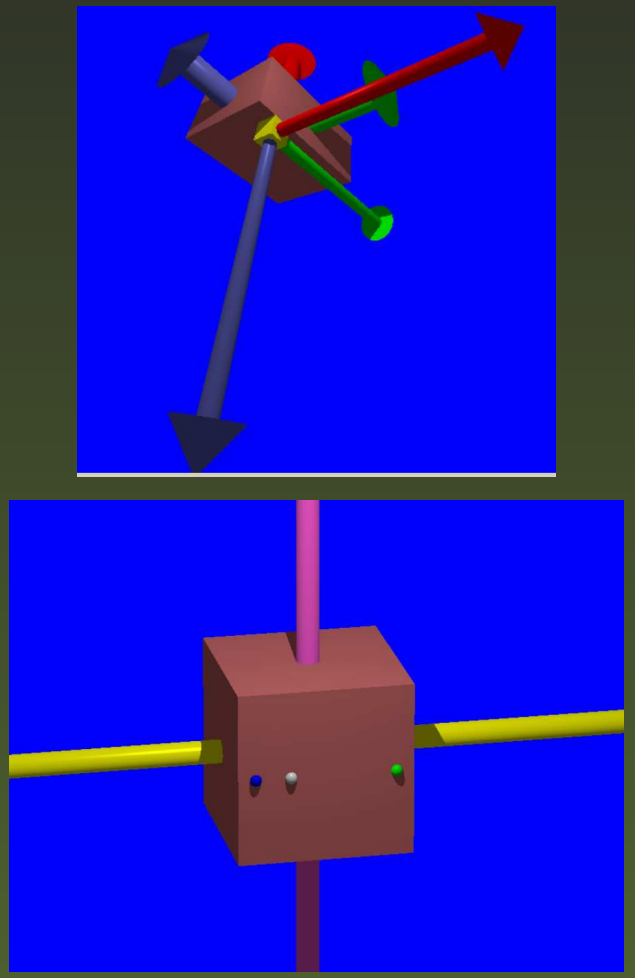
The carpenter paradigm



- movePieces, coordinate free, box paradigm, gluing

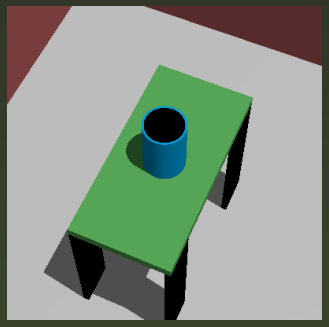
```
tableTray=Cube(tableTrayDimensions)
tableLeg1=Cube(tableLegDimensions)
tableLeg2=tableLeg1.copy()
tableLeg3=tableLeg1.copy()
tableLeg4=tableLeg1.copy()
# The next line moves tableLeg1 to the tray at adjusting the cubes at t
tableLeg1.move_against(tableTray,Z,Z,X,X,offset=(0,0,0),adjustEdges=-X-
# The next line glues the leg on the table
tableLeg1.transplant_on(tableTray)
```

The box paradigm



- Deeply in our mind (child drawings , natural language)

CSG



```
flowerPot=Cylinder(origin,origin+flowerPotHeight*Z,radius=flowerPotRadius)
toCut=Cylinder(origin+flowerPotThickness*Z,origin+2*flowerPotHeight*Z,radius=flowerPotRadius-flowerPotThickness)
flowerPot.amputed_by(toCut)
```

- 1 type of intersection
- 1 type of difference
- 2 types of union (parenting+symmetric)

markers

- points
- lines
- center

Objectives

- code in a high level language (write,read,maintain)
- a documentation with precise mathematics
- shortest possible description of the scene
- a language of moderate size (no redundancies, no rarely used features) towards not full time cad developers
- modular architecture
- tools to build and share library of objects