Computational Graphics: Lecture 12

The CVDlab Team

Thu, Mar 27, 2014

1. 3D Affine transformations
   - Translation and scaling
   - Rotation
   - Shearing

Introduction

A 3D extension of plane translation and scaling is easy

A major care is only needed for 3D rotation and 3D shearing

In order to unify the tratment of linear and affine transformations, and to use the matrix product as the only geometric operator, we use normalized homogeneous coordinates and tensors in $\mathrm{lin}\ \mathbb{R}^4$

Translation

Translation tensor $\mathbf{T}_{xyz}(l, m, n)$ with parameters l, m, n (the components of the translation vector), and its matrix:

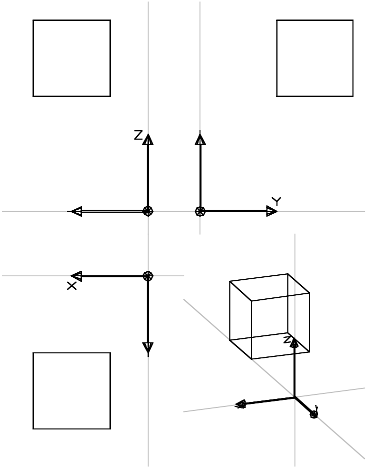
$$\mathbf{T}_{xyz}(l, m, n) = \begin{pmatrix} 1 & 0 & 0 & l \\ 0 & 1 & 0 & m \\ 0 & 0 & 1 & n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Remark

Here the homogeneous row and column are the last ones !!

Translation

Predefined operator in Pyplasm




```
T([1,2,3])([0.5,1,1.5])(CUBOID([1,1,1]))
```

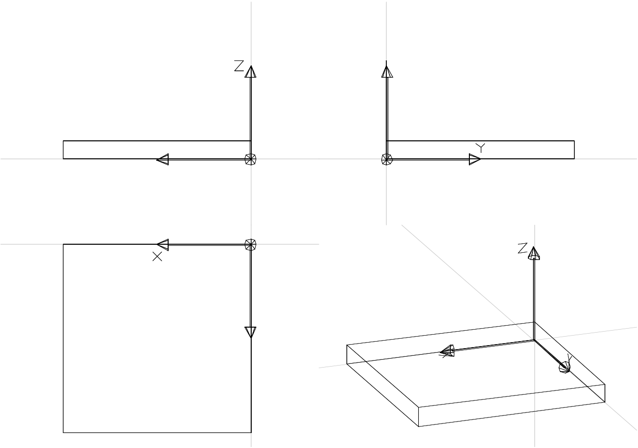
Scaling

The scaling tensor $\mathbf{S}_{xyz}(a, b, c)$ with parameters a, b, c is represented in coordinates by the matrix

$$\mathbf{S}_{xyz}(a, b, c) = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Scaling

Predefined operator in `Pyplasm`




```
S([1,2,3])([2,2,0.2])(CUBOID([1,1,1]))
```

Elementary rotations

There are $\binom{d}{2}$ different elementary rotations in $\mathbb{E}^d$

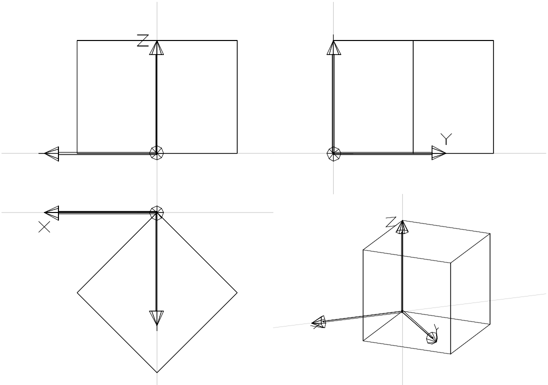
Given a Cartesian frame in $\mathbb{E}^3$, we call elementary rotations $\mathbf{R}_{yz}$,$\mathbf{R}_{xz}$ and $\mathbf{R}_{xy}$, three functions $\mathbb{R} \to \mathrm{lin}\,\mathbb{R}^3$, which give, for every angle, the rotation tensor about a cooordinate axis

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}, \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}, \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Matrices in Cartesian coordinates

Elementary rotations

Predefined operator in Pyplasm

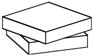


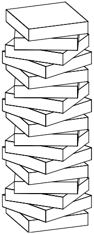

```
R([1,2])(PI/4)(CUBOID([1,1,1]))
```

Elementary rotations

example

Here we define a parallelepiped `element`, translated in x, y by a tensor `T([1,2])([-5,-5])` to move its center upon the z axis





```
element = COMP([T([1,2])([-5,-5]), CUBOID])([10,10,2])
element = T([1,2])([-5,-5])( CUBOID([10,10,2]) )

column = STRUCT( NN(17)([element, T(3)(2), R([1,2])(PI/8)]) )
column = STRUCT( CAT(N(17)([element, T(3)(2), R([1,2])(PI/8)])) )

VIEW(column)
```

equivalent expressions

General rotation

A rotation of $\mathbb{E}^3$ is a linear orthogonal transformation with a set of fixed points (eigenspace in linear algebra) of dimension 1, known as rotation axis

In this transformation, every point (not on the axis) is mapped in the other extreme of a circumference arc of constant angle, centered on the axis, and contained in a plane orthogonal to it.

We will compute the rotation matrix of the tensor $\mathbf{R}_{xyz}(\mathbf{n}, \alpha)$, with

$$\mathbf{R}_{xyz} : \mathbb{R}^3 \times \mathbb{R} \to \operatorname{lin} \mathbb{R}^4 : (\mathbf{n}, \alpha) \mapsto \mathbf{R}_{xyz}(\mathbf{n}, \alpha),$$

where the vector $\mathbf{n}$ is parallel to the rotation axis, and α is the angle of rotation

General rotation

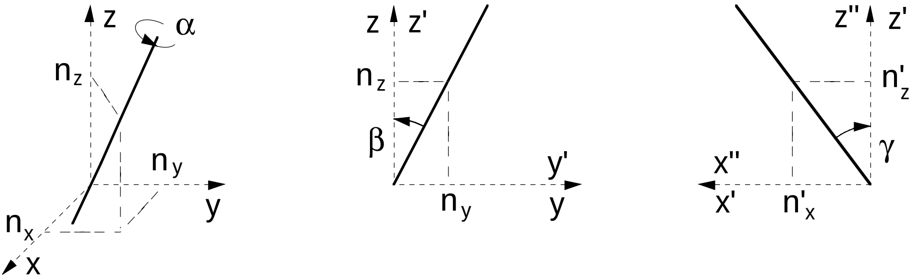
by composition of elementary rotations

A 3D non elementary rotation $\mathbf{R}_{xyz}(\mathbf{n}, \alpha)$, with axis n and α angle, can be reduced to the composition of elementary rotations:

$$
\begin{aligned}
\mathbf{R}_{xyz}(\mathbf{n}, \alpha) &= (\mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta))^{-1} \circ \mathbf{R}_z(\alpha) \circ (\mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta)) \\
&= \mathbf{R}_x(\beta)^{-1} \circ \mathbf{R}_y(\gamma)^{-1} \circ \mathbf{R}_z(\alpha) \circ \mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta) \\
&= \mathbf{R}_x(-\beta) \circ \mathbf{R}_y(-\gamma) \circ \mathbf{R}_z(\alpha) \circ \mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta).
\end{aligned}
$$

General rotation

by composition of elementary rotations



(a) **n** axis; (b) rotation about x; (c) rotation about y

$$\beta = \arctan\left(\frac{n_y}{n_z}\right) \qquad \gamma = -\arctan\left(\frac{n'_x}{n'_z}\right)$$

where $\mathbf{n}' = \mathbf{R}_x(\beta)\ \mathbf{n}$.

General rotation

by transformation of coordinates

The tensor $\mathbf{R}_{xyz}(\mathbf{n}, \alpha)$ of a general rotation may be computed by composition of three tensors:

$$\mathbf{R}_{xyz}(\mathbf{n}, \alpha) = \mathbf{Q}_{\mathbf{n}}^{-1} \circ \mathbf{R}_z(\alpha) \circ \mathbf{Q}_{\mathbf{n}}.$$

such that:

1. a coordinate transformation $\mathbf{Q}_{\mathbf{n}}$ that maps the unit vector $\frac{\mathbf{n}}{|\mathbf{n}|}$ and two orthogonal versors to the elements of a new basis;
2. a rotation $\mathbf{R}_z(\alpha)$ about the z axis of this new basis;
3. the inverse coordinate transformation $\mathbf{Q}_{\mathbf{n}}^{-1}$.

General rotation

by transformation of coordinates

We choose a triple $\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z$ of orthonormal vectors, with an element oriented as the rotation axis

such vectors are mapped to the basis $\{\mathbf{e}_i\}$ by the unknown matrix $\mathbf{Q_n}$:

$$\begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{pmatrix} = \mathbf{Q_n} \begin{pmatrix} \mathbf{q}_x & \mathbf{q}_y & \mathbf{q}_z \end{pmatrix}.$$

so that

$$\mathbf{Q_n} = \begin{pmatrix} \mathbf{q}_x & \mathbf{q}_y & \mathbf{q}_z \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{q}_x & \mathbf{q}_y & \mathbf{q}_z \end{pmatrix}^T = \begin{pmatrix} \mathbf{q}_x^T \\ \mathbf{q}_y^T \\ \mathbf{q}_z^T \end{pmatrix}$$

General rotation

by transformation of coordinates

Let us start by setting

$$\mathbf{q}_z = \frac{\mathbf{n}}{\mathbf{n}},$$

We suppose that $\mathbf{n} \neq \mathbf{e}_3$ is verified — if false, it would imply $\mathbf{R}(\mathbf{n}, \alpha) = \mathbf{R}_z(\alpha)$.

Therefore:

$$\mathbf{q}_x = \frac{\mathbf{e}_3 \times \mathbf{n}}{\mathbf{e}_3 \times \mathbf{n}}, \qquad \text{and} \qquad \mathbf{q}_y = \mathbf{q}_z \times \mathbf{q}_x.$$

General rotation

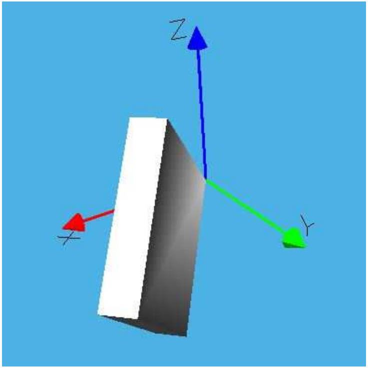
Implementation by transformation of coordinates

```
def ROTN (args):
    alpha, n = args
    n  = UNITVECT(n)
    qx = UNITVECT((VECTPROD([[0,0,1], n])))
    qz = UNITVECT(n)
    qy = VECTPROD([qz,qx])
    Q  = MATHOM([qx, qy, qz])
    if n[0]==0 and n[1]==0:
        return R([1, 2])(alpha)
    else:
        return COMP([MAT(TRANS(Q)),R([1,2])(alpha),MAT(Q)])

VIEW( ROTN([PI/4, [0,0,1]])(CUBE(1)) )
VIEW( ROTN([PI/4, [1,1,1]])(CUBE(1)) )
```

General rotation

example




```
obj = ROTN([ pi/2, [1,1,0] ])(CUBOID([1,1,0.2]))
VIEW(obj)
```

Elementary shearing

A 3D elementary shearing is a tensor that does'nt change one coordinate of $\mathbb{E}^3$ points, and maps the others as linear functions of the non-transformed coordinate

We may distinguish three elementary shearing tensors $\mathbf{H}_{yz}(a, b)$, $\mathbf{H}_{xz}(a, b)$ and $\mathbf{H}_{xy}(a, b)$, whose matrices differ from the identity only by the elements of a single column

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Elementary shearing

- Let consider the 3D space as a bundle of planes parallel to a coordinate plane, that remains fixed
- The other planes are translated on theirselves, by a linear function of their distance from the fixed plane

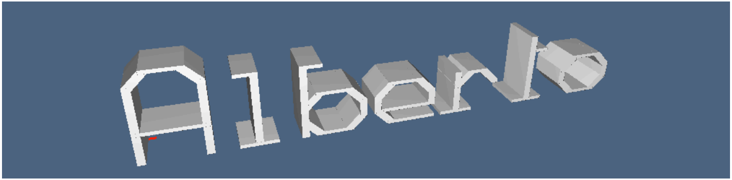
$$
\begin{aligned}
\mathbf{p}^* &= \mathbf{H}_x(a,b)\,\mathbf{p} = (x,\ y+ax,\ z+bx,\ 1)^T \\
\mathbf{p}^* &= \mathbf{H}_y(a,b)\,\mathbf{p} = (x+ay,\ y,\ z+by,\ 1)^T \\
\mathbf{p}^* &= \mathbf{H}_z(a,b)\,\mathbf{p} = (x+az,\ y+bz,\ z,\ 1)^T
\end{aligned}
$$

with respect to the tensor $\mathbf{H}_z = \mathbf{H}_{xy}(a,b)$:

1. the $z = 0$ plane is invariant;
2. the $z = 1$ plane translates by the translation vector $\mathbf{t} = (a,b,0)^T$;
3. each plane $z = c$ translates by a vector $\mathbf{t}' = c(a,b,0)^T$.

Elementary shearing

`from myfont import *`



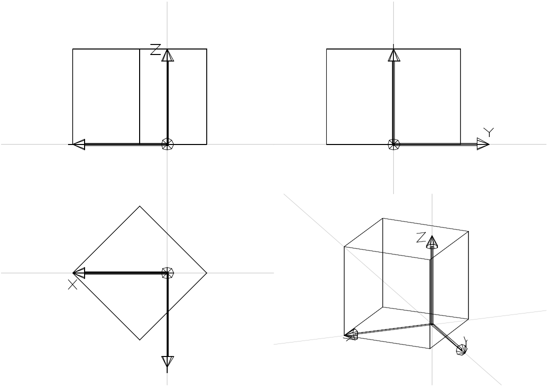

```
obj = PROD([  OFFSET([0.5,0.25])(TEXT("Alberto")) , Q(3) ])
VIEW(obj)
tensor = MAT([[1,0,0,0],[0,1,0.5,0],[0,0,1,0],[0,0,0,1]])
VIEW(tensor(obj))
```

QUESTION: what shearing was applied ?

Composite transformations

by function composition, NO matrix product !!

$\boxed{\text{rotation of } \pi/4 \text{ about the axis for the edge } ((1,0,0),(1,0,1))}$




```
(T:1:1  ~  R:<1,2>:(PI/4)  ~  T:1:-1):(CUBOID:<1,1,1>);
```

Rotation about an affine axis

a more general rotation tensor of $\mathbb{E}^3$, with fixed affine subspace of dimension 1, i.e. a line out the origin, is obtained by composition of transformations in $\mathrm{lin}\ \mathbb{R}^4$:

$$\mathbf{R}^*_{xyz}(\mathbf{n}, \mathbf{p}, \alpha) = \mathbf{T}_{xyz}(\mathbf{p} - \mathbf{o}) \circ \mathbf{R}_{xyz}(\mathbf{n}, \alpha) \circ \mathbf{T}_{xyz}(\mathbf{o} - \mathbf{p})$$

where $\mathbf{R}^*_{xyz}(\mathbf{n}, \mathbf{p}, \alpha)$ denotes a *rotation about the* **n** *axis though the point* $\mathbf{p} \in \mathbb{E}^3$, and **o** is the origin of the Cartesian frame $\mathbb{E}^3$.

Reflection about an affine plane

analogously a reflection $\mathbf{Z}_{xyz}(\mathbf{n}, \mathbf{p})$ about to a plane (think to a mirror) with normal $\mathbf{n}$ passing for the point $\mathbf{p}$ need the composition of:

1. a translation $\mathbf{T}_{xyz}(\mathbf{o} - \mathbf{p})$ moving $\mathbf{p}$ to the origin $\mathbf{o}$
2. a rotation $\mathbf{R}_{xyz}(\mathbf{n} \times \mathbf{e}_3, \alpha)$ moving $\mathbf{n}$ on the axis $\mathbf{e}_3$, with $\alpha =$
3. a reflection $\mathbf{S}(1, 1, -1)$ w.r.t. a normal coordinate plane
4. the inverse rotation $\mathbf{R}_{xyz}(\mathbf{n} \times \mathbf{e}_3, -\alpha)$
5. the inverse translation $\mathbf{T}_{xyz}(\mathbf{p} - \mathbf{o})$

$$\mathbf{Z}_{xyz}(\mathbf{n}, \mathbf{p}) =$$

$$\mathbf{T}_{xyz}(\mathbf{p} - \mathbf{o}) \circ \mathbf{R}_{xyz}(\mathbf{n} \times \mathbf{e}_3, -\alpha) \circ \mathbf{S}(1, 1, -1)\mathbf{R}_{xyz}(\mathbf{n} \times \mathbf{e}_3, \alpha) \circ \mathbf{T}_{xyz}(\mathbf{o} - \mathbf{p})$$

Uniform scaling

definition

A uniform scaling $\mathbf{S}_{xyz}(a, a, a)$ is represented by a matrix $(s_{ij}) \in \mathbb{R}^4_4$, different from the identity 4×4 only by s_{44}:

$$\mathbf{S}_{xyz}(a, a, a) \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{a} \end{pmatrix}$$

it is easy to verify that:

$$p^* = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{a} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ \frac{1}{a} \end{pmatrix} = \begin{pmatrix} ax \\ ay \\ az \\ 1 \end{pmatrix}$$

Structure of an affine tensor

The affine transformations of $\mathbb{E}^3$, represented in homogeneous coordinates by real matrices 4×4, have the structure:

$$\mathbf{Z} = \begin{pmatrix} \mathbf{Q} & \mathbf{m} \\ \mathbf{0}^T & a \end{pmatrix}$$

- where $\mathbf{Q}$ is an invertible matrix 3×3
- if $\mathbf{m} \neq \mathbf{0}$, then $\mathbf{Z}$ has a translation component
- if $a \neq 1$, then we say that $\mathbf{Z}$ is *non normalised*. In this case it contains a uniform scaling with parameter $\frac{1}{a}$.

Action of a tensor on covectors

A linear equation of type $ax + by + cz + d = 0$ (Cartesian equation of a plane in E^3) can be written as:

$$\mathbf{q}\mathbf{p} = 0$$

where $\mathbf{q} = (a, b, c, d)$ e $\mathbf{p} = (x, y, z, 1)^T$
what is the action of an affine tensor $\mathbf{M}$ on the affine plane? We know that it changes lines to lines and planes to planes, ergo

$$\mathbf{q}^*\mathbf{p}^* = \mathbf{q}\mathbf{Q}\mathbf{M}\mathbf{p} = 0$$

ovvero

$$\mathbf{q}\mathbf{Q}\mathbf{M}\mathbf{p} = \mathbf{q}\mathbf{p}$$

so $\mathbf{Q}\mathbf{M} = \mathbf{I}$, and hence, for the unknown tensor $\mathbf{Q}$ to be applied to covectors we have: $\boxed{\mathbf{Q} = \mathbf{M}^{-1}}$

Properties of tensors

composition or product ??

When a succession of tensors $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_n$ is applied to a point $\mathbf{p}$, we can write either

$$\mathbf{p}^* = (\mathbf{Q}_n \circ \cdots \circ \mathbf{Q}_2 \circ \mathbf{Q}_1)(\mathbf{p}),$$

or

$$\mathbf{p}^* = \mathbf{Q}_n \;\cdots\; \mathbf{Q}_2 \; \mathbf{Q}_1 \; \mathbf{p},$$

depending on the meaning (tensor or matrix) of $\mathbf{Q}_i$ symbol.

Properties of tensors

Associativity

Both the tensor composition and the product of matrices are associative operations (left-and or right-hand)

$$(\mathbf{Q}_3 \circ \mathbf{Q}_2) \circ \mathbf{Q}_1 = \mathbf{Q}_3 \circ (\mathbf{Q}_2 \circ \mathbf{Q}_1) = \mathbf{Q}_3 \circ \mathbf{Q}_2 \circ \mathbf{Q}_1$$

$$(\mathbf{Q}_3\mathbf{Q}_2)\ \mathbf{Q}_1 = \mathbf{Q}_3\ (\mathbf{Q}_2\mathbf{Q}_1) = \mathbf{Q}_3\mathbf{Q}_2\mathbf{Q}_1$$

The use of parentheses is hence not needed to specify the order of operations

Properties of tensors

Non commutativity

In general, the composition of tensors and the product of matrices are not commutative:

$$\mathbf{Q}_1 \circ \mathbf{Q}_2 \neq \mathbf{Q}_2 \circ \mathbf{Q}_1 \quad \text{e} \quad \mathbf{Q}_1\mathbf{Q}_2 \neq \mathbf{Q}_2\mathbf{Q}_1,$$

But there are important exceptions to this rule:
e.g., are commutative (the list is not complete):

1. the composition (product) of rotations about the same axes;
2. the composition (product) of translations;
3. the composition (product) of scalings;
4. the composition (product) of rotations and uniform scalings;.

Properties of tensors

Composition

Rotation and translation tensors additive composability:

$$\mathbf{T}_{xy}(m_1, n_1) \circ \mathbf{T}_{xy}(m_2, n_2) = \mathbf{T}_{xy}(m_1 + m_2, n_1 + n_2),$$

$$\mathbf{R}_{xy}(\alpha_1) \circ \mathbf{R}_{xy}(\alpha_2) = \mathbf{R}_{xy}(\alpha_1 + \alpha_2)$$

conversely, scaling tensors have multiplicative composability:

$$\mathbf{S}_{xy}(a_1, b_1) \circ \mathbf{S}_{xy}(a_2, b_2) = \mathbf{S}_{xy}(a_1 a_2, b_1 b_2)$$

Properties of tensors

Inverse

It follows, for inverse transformations:

$$(\mathbf{T}_{xy}(m,n))^{-1} = \mathbf{T}_{xy}(-m,-n)$$

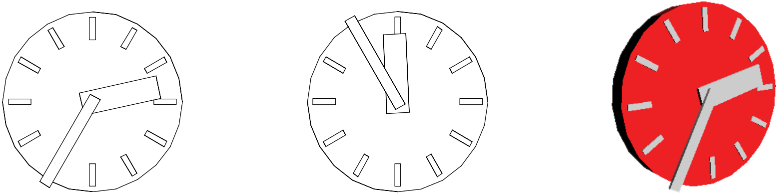
$$(\mathbf{R}_{xy}(\alpha))^{-1} = \mathbf{R}_{xy}(-\alpha)$$

$$(\mathbf{S}_{xy}(a,b))^{-1} = \mathbf{S}\left(\frac{1}{a},\frac{1}{b}\right)$$

Examples

2D/3D Clock model

Circular `background`, 12 `ticks` of hours, and `hour` and `minute` hands, given in their local coordinate systems



```
background = COLOR(RED)(CIRCLE(0.8)([48,1]))
minute = T([1,2])([-0.05,-0.05])(CUBOID([0.9,0.1]))
hour = T([1,2])([-0.1,-0.1])(CUBOID([0.7,0.2]))
tick = T([1,2])([-0.025,0.55])(CUBOID([0.05,0.2]))
ticks = STRUCT(NN(12)([ tick, R([1,2])(PI/6) ]))
```

Examples

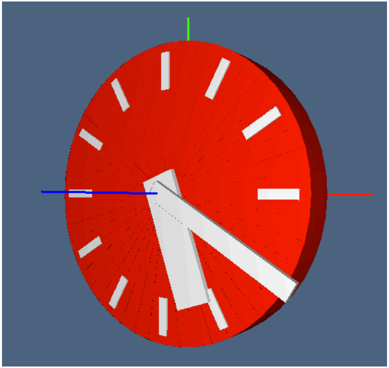
2D/3D Clock model

```
def clock2D (h,m):
    return STRUCT([ background, ticks,
    R([1,2])( PI/2 - (h + m/60)*PI/6 )(hour),
    R([1,2])( PI/2 - m*PI/30 )(minute) ])
```

```
def clock3D (h,m):
    return STRUCT([
     COLOR(RED)(PROD([ background, Q(0.2) ])),
     T(3)(0.2)(PROD([ ticks, Q(0.01) ])), T(3)(0.2),
     R([1,2])( PI/2 - (h + m/60.)*PI/6 )(PROD([ hour, Q(0.03) ])),
         T(3)(0.03),
     R([1,2])( PI/2 - m*PI/30 )(PROD([ minute, Q(0.03) ])) ])
```

Examples

2D/3D Clock model



```
VIEW(clock3D(5,20))
```