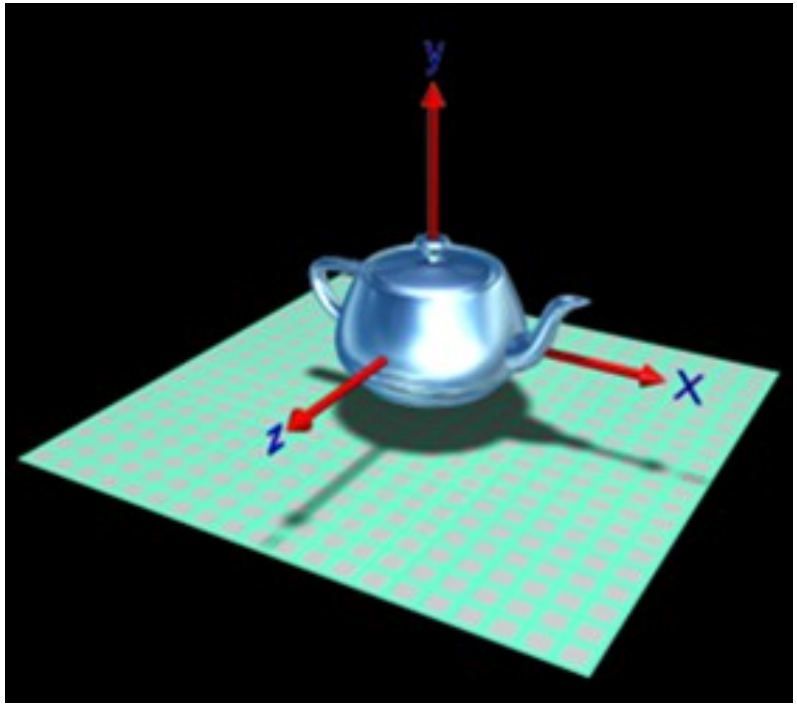
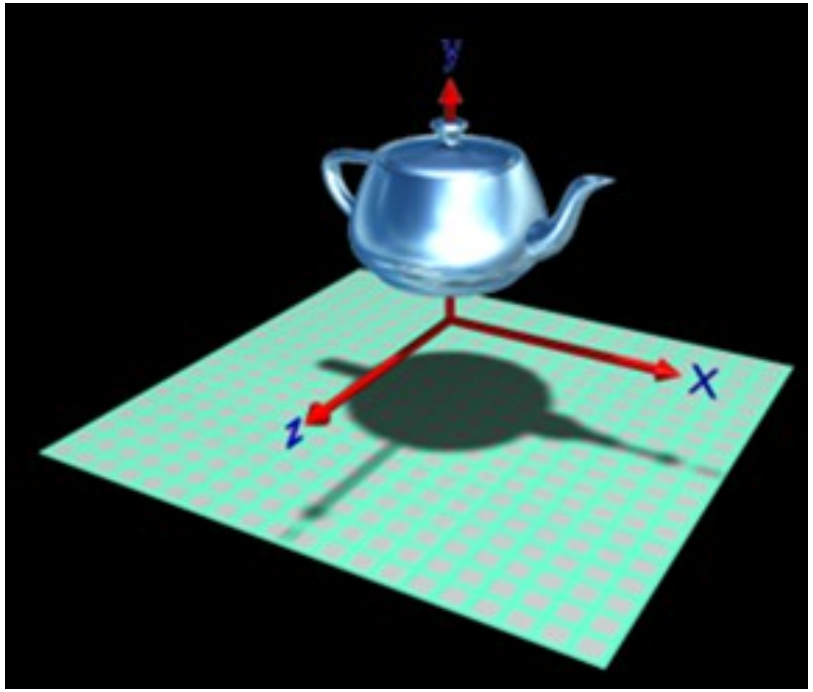
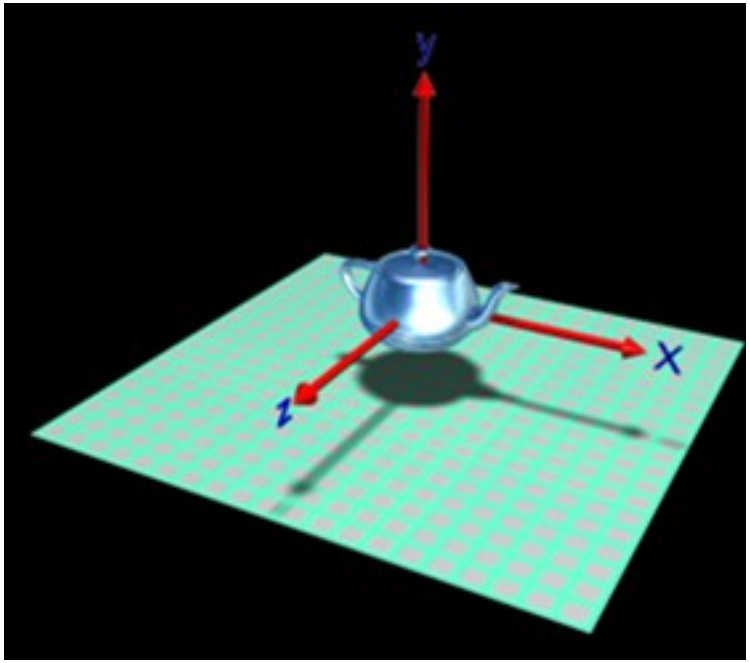
3D transformacije in gledanje

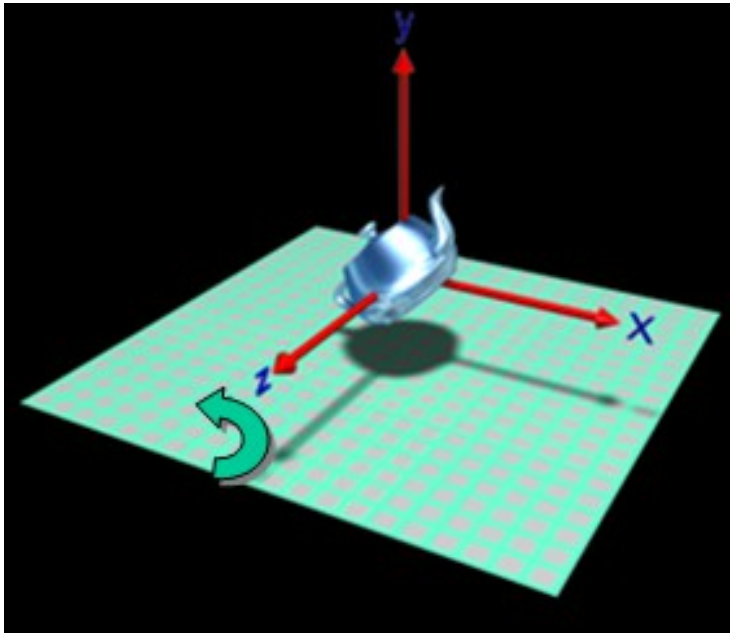
Premikanje predmeta - translacija



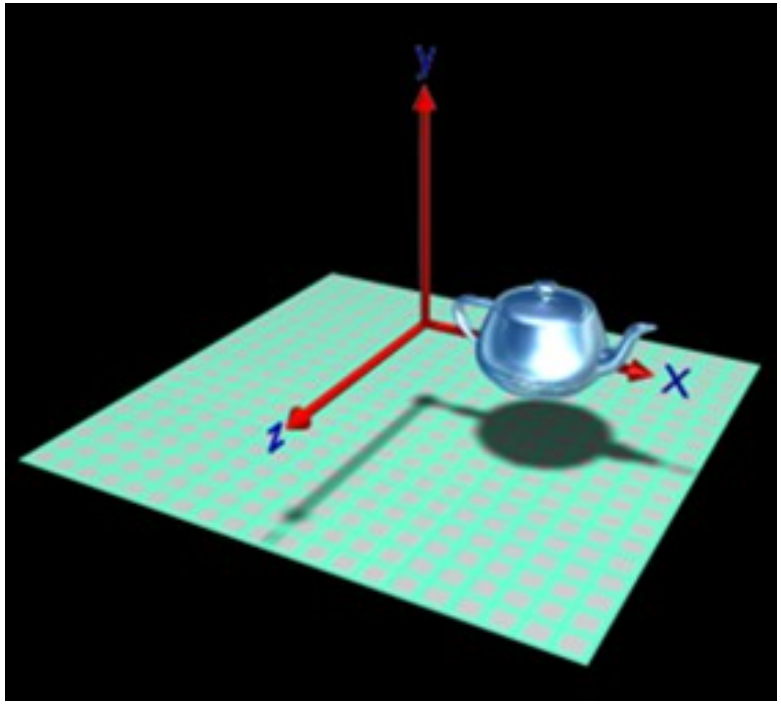


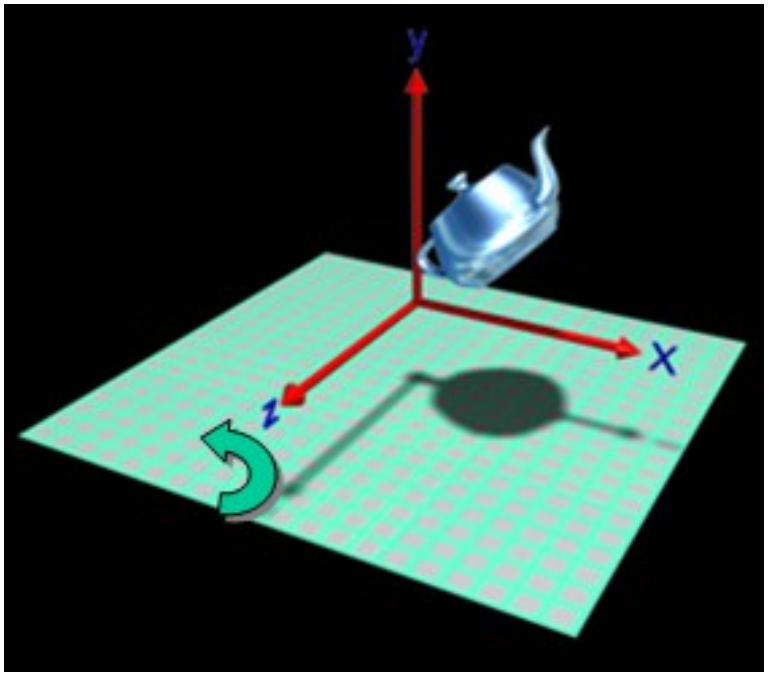
Vrtenje – rotacija okrog središča



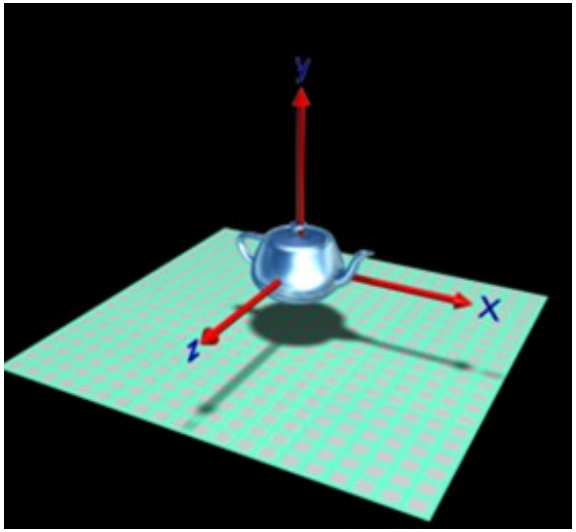


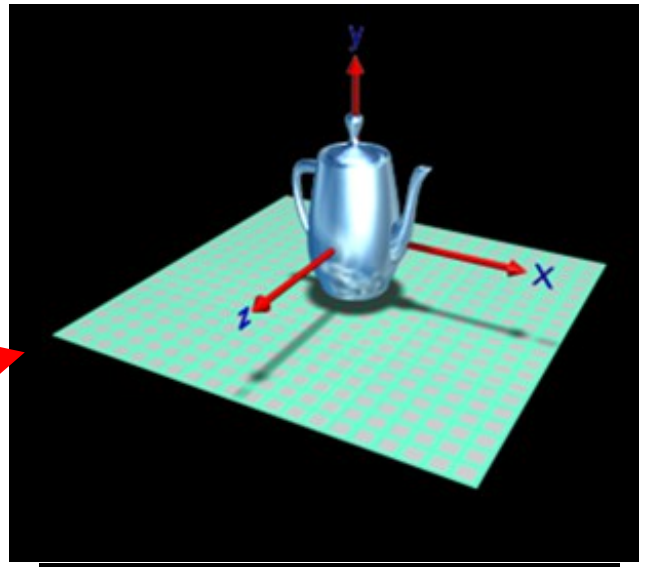
Vrtenje – rotacija okrog tečaja

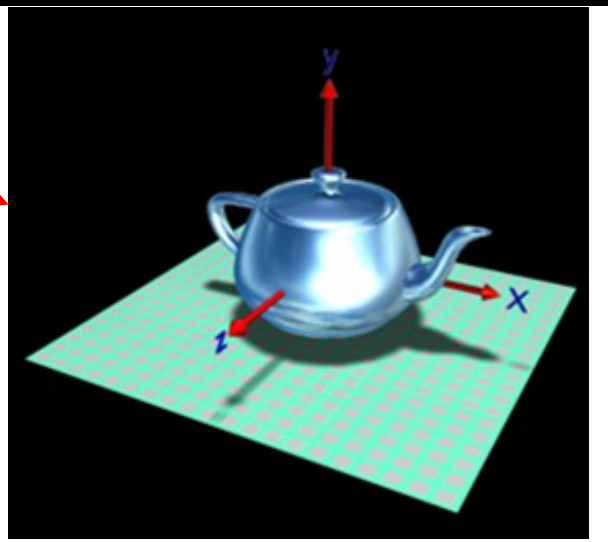




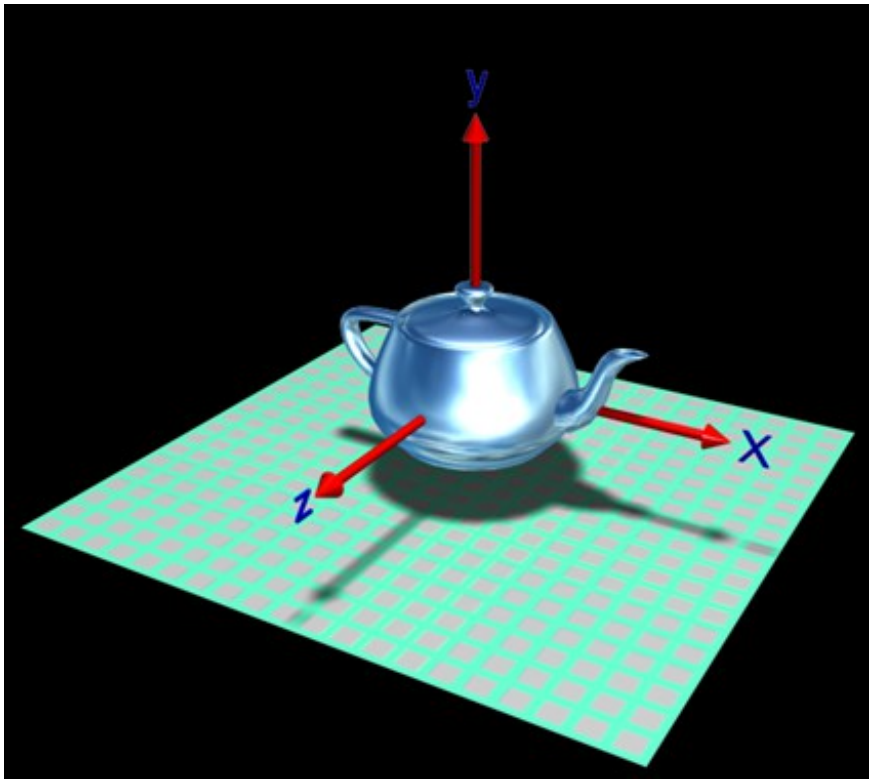
Povečava -pomanjšanje

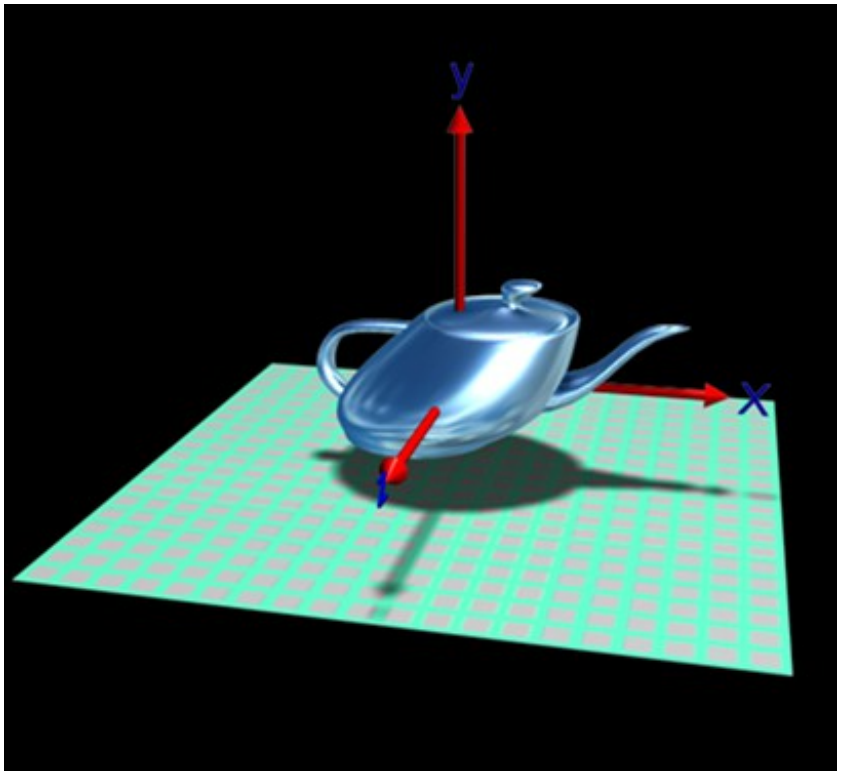






Striženje (shear)





Povečanje 2D transformacij na 3D

Primer:

<u>2D skaliranje</u>:

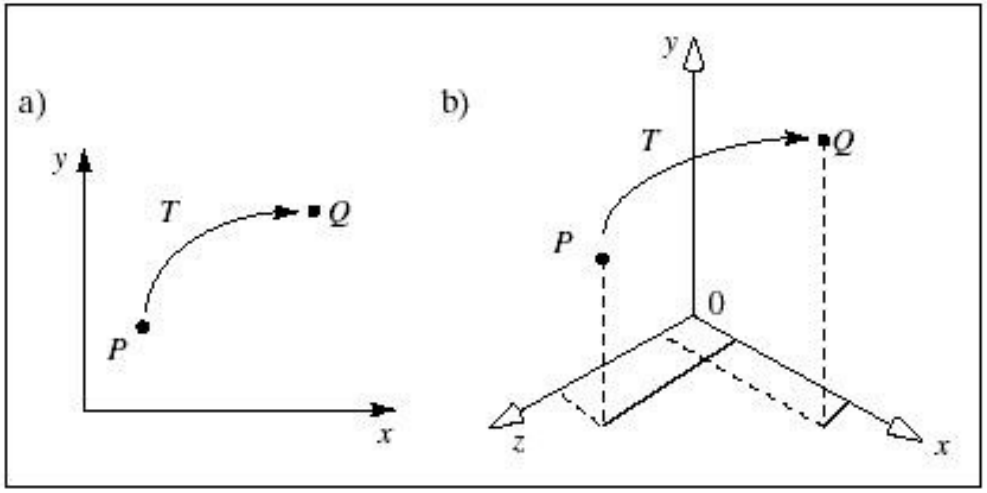
$$[x'\ y'\ 1] = [x\ y\ 1] * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

<u>3D skaliranje</u>:

$$[x'y'z'1] = [x\ y\ z\ 1] * \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Narišemo samo x in y.

Transformacija točk in predmetov



$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} = M \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

$$P = \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix} \quad Q = \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix}$$

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Homogena predstavitev

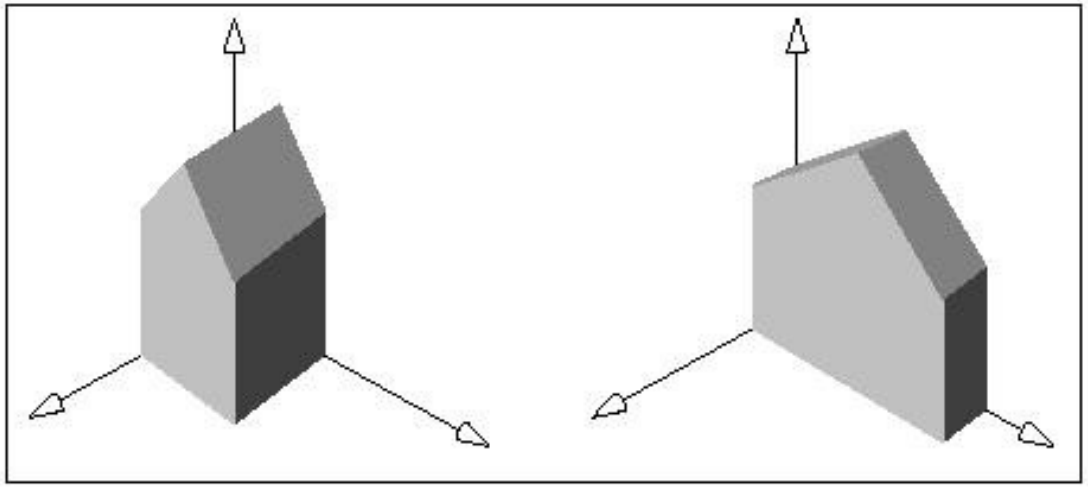
Osnovne 3D afine transformacije

Translacija

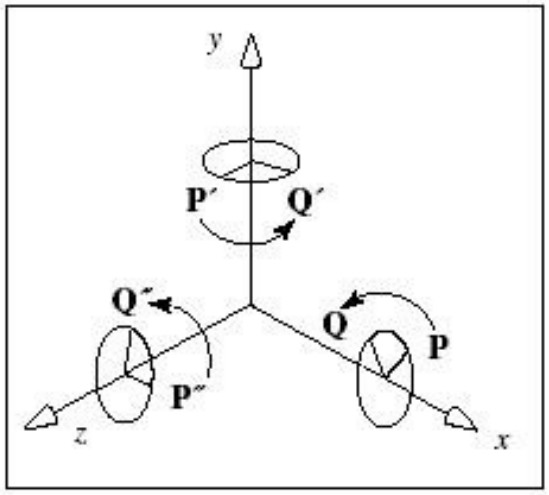
$$\begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_x \\ 0 & 0 & 1 & T_x \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Povečava (scalling)

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Rotacije



Okoli osi x:

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

Okoli osi y:

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

Okoli osi z

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

3D Rotacije

Okoli osi x

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Okoli osi y

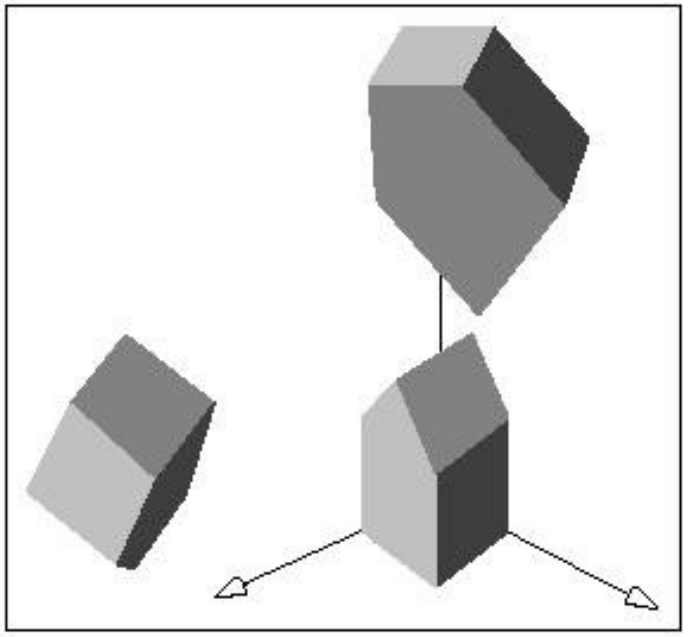
$$\begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

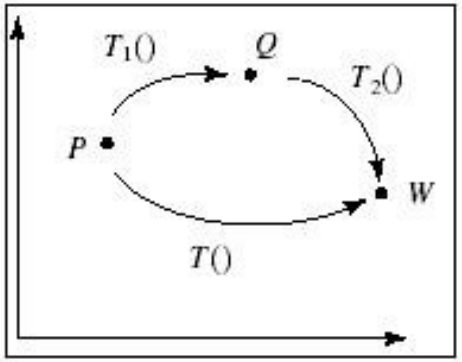
Okoli osi z

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Opazka: to je enako kot pri 2D rotaciji

Sestavljanje (veriženje): Enako kot v 2D





$$W = M(P) = (M_2 M_1)(P)$$

Sestavljanje rotacij

Vrstni red **je pomemben**

3D matrik vrtenja **ne smemo** zamenjati

Še nekaj primerov 3D transformacij

Zrcaljenje preko ravnine xy

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Enakomerna povečava

$$\begin{pmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translacija v smeri x

$$\begin{pmatrix} 1 & 0 & 0 & m \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Splošna oblika 3D transformacijske matrike

$$M = \begin{bmatrix} a & b & c & l \\ d & e & f & m \\ g & h & i & n \\ p & q & r & s \end{bmatrix}$$

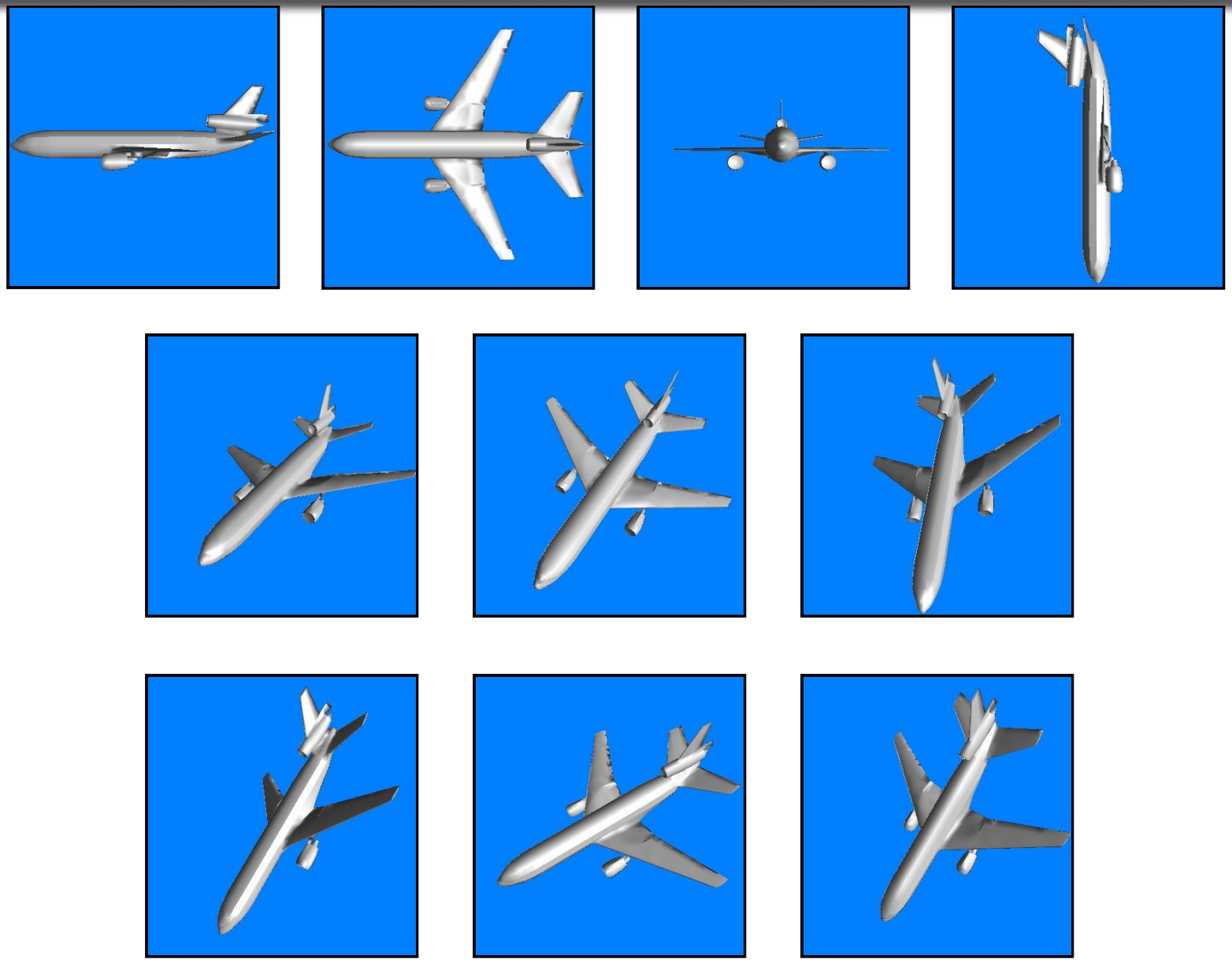
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$ ⇨ povečava, striženje, vrtenje, zrcaljenje

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix}$$ ⇨ Translacija

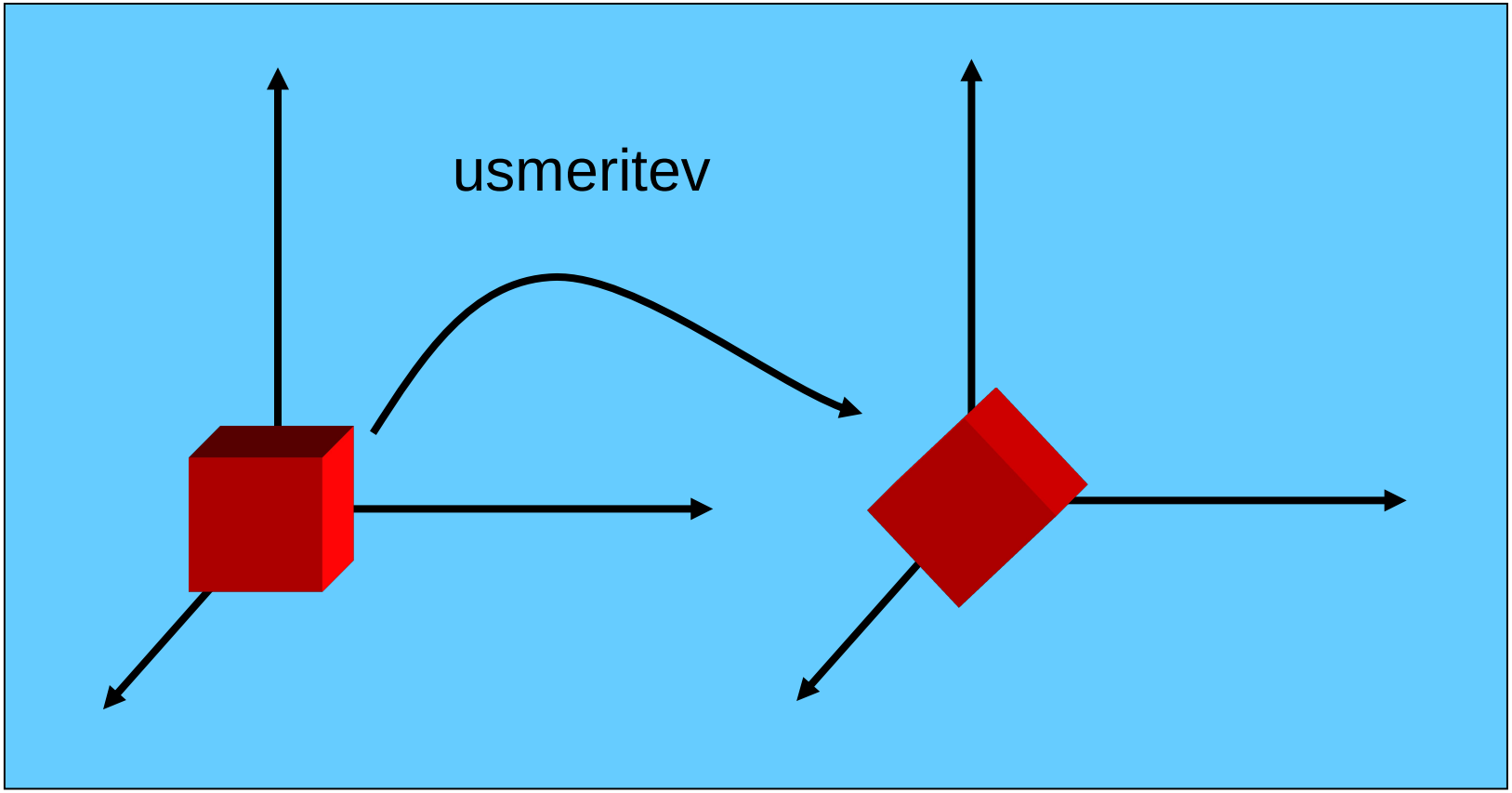
$$\begin{pmatrix} p & q & r \end{pmatrix}$$ ⇨ Perspektivna preslikava

s ⇨ Splošna povečava

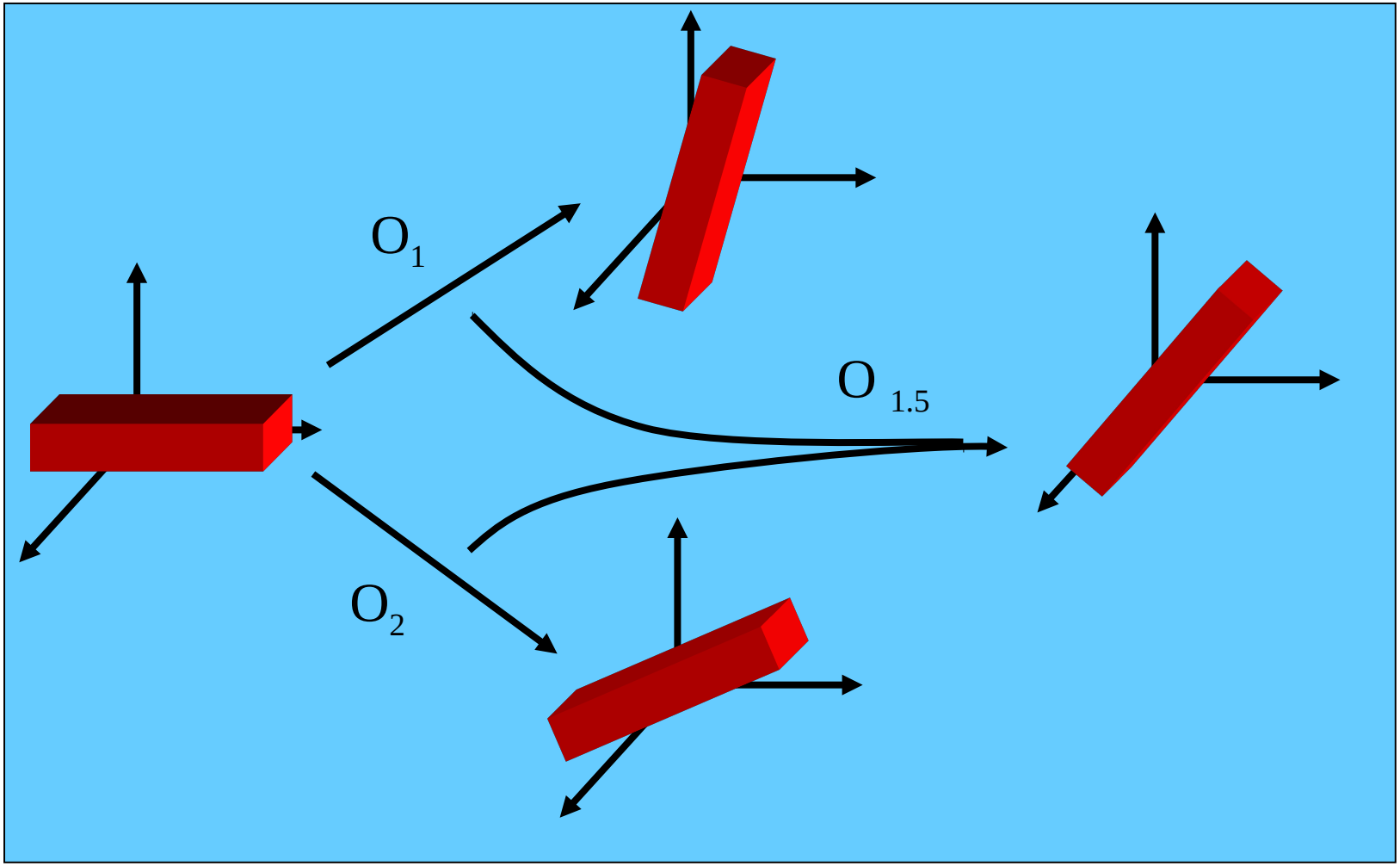
Še več o rotacijah



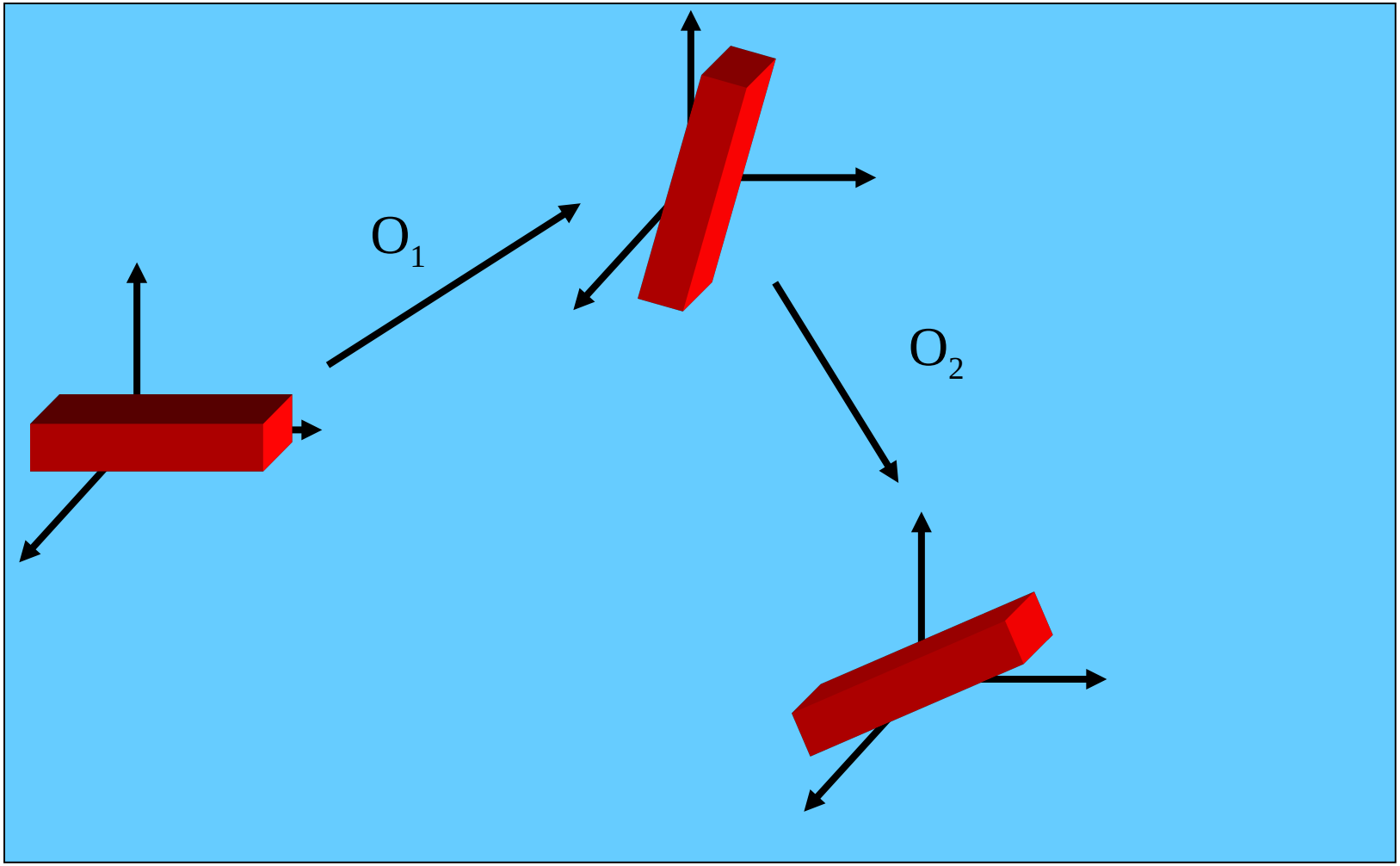
Več o rotacijah oziroma usmeritvi



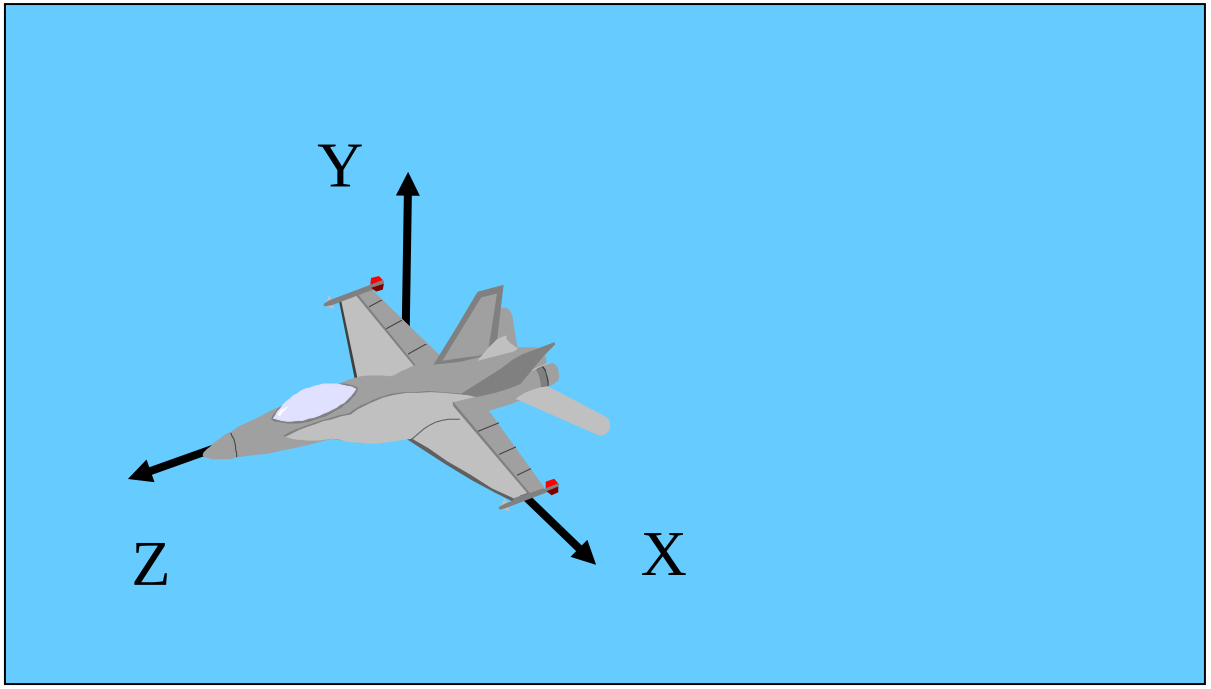
Interpolacija



Veriženje



Koti rotacije okrog fiksnih osi



$$R_x(\theta_1).\ R_y(\theta_2).\ R_z(\theta_3).\ P$$

Predstavitev fiksnih kotov

- Koti, za katere zavrtimo predmet okrog fiksnih osi
- Usmeritve določa množica 3 urejenih parametrov ki predstavljajo urejene rotacije okrog fiksnih osi: najprej okrog x, nato y nato z
- Več možnih zaporedij, ki ne uporabijo nujno vse 3 osi

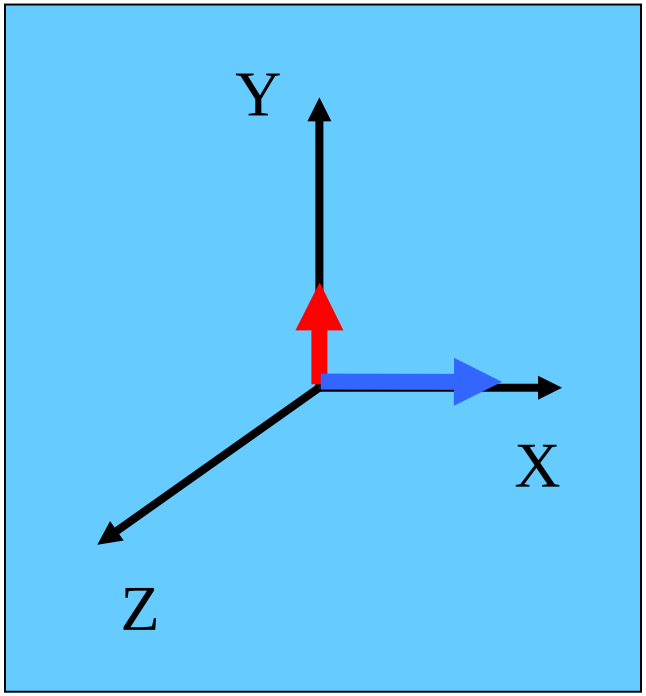
Predstavitev fiksnih kotov

- Vrtenje za 10,45, 90 bi zapisali kot
  - Rz(90) Ry(45), Rx(10), saj hočemo najprej rotacijo okrog, ki jo uporabimo na točki P…. RzRyRx P

- Do problema pride, če se dve osi vrtenja poravnata med seboj. Temu pojavu pravimo kardanska zapora (“Gimbal Lock”)

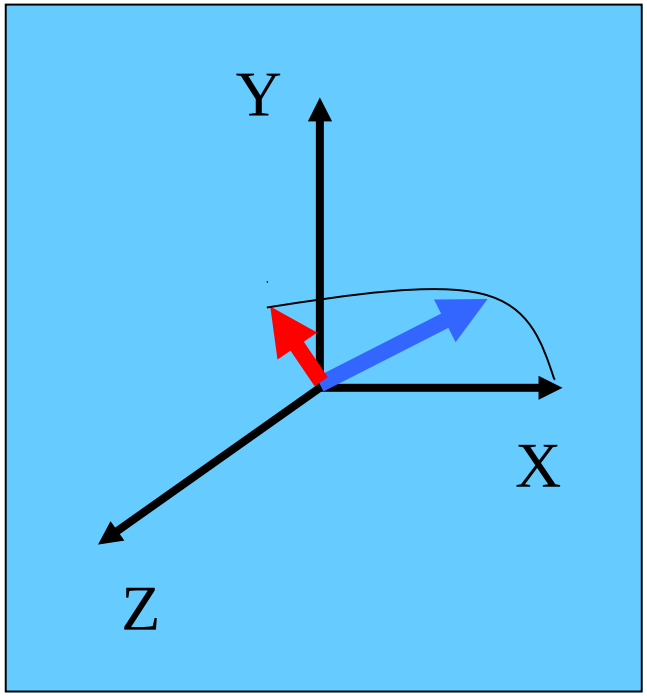
Primeri fiksnih kotov rotacije

Primer (0,90,0)



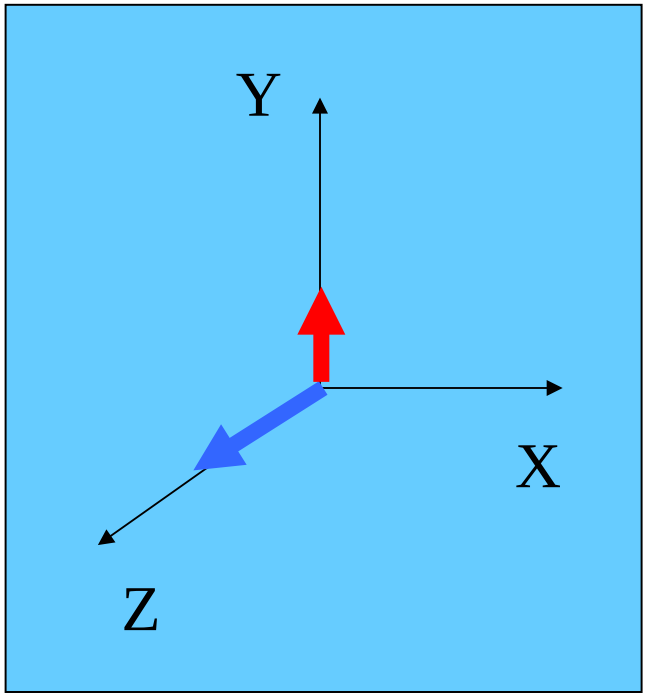
Rotacija
okoli osi z

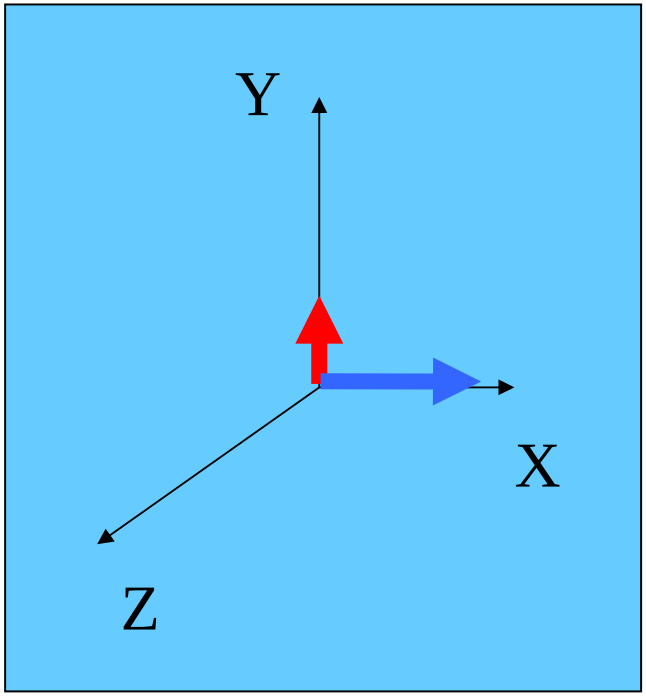
Primer (-45,90,0)



Primeri fiksnih kotov rotacije

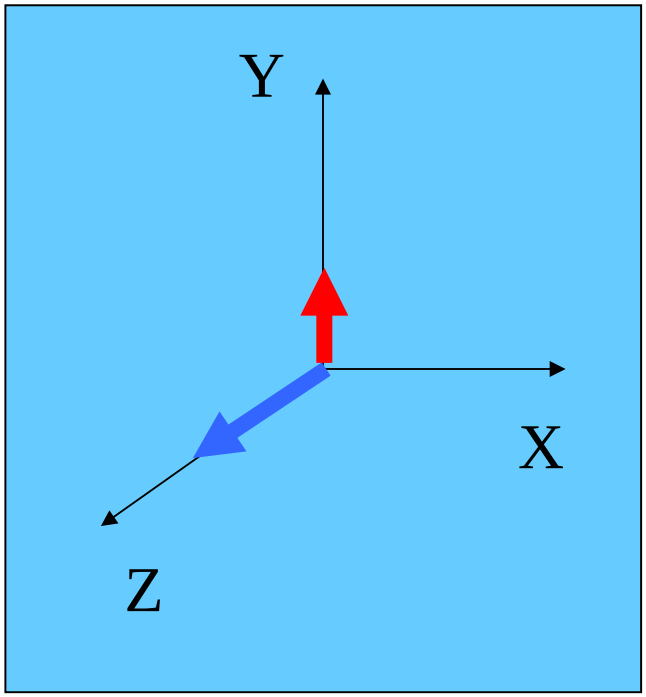
Na primer (0,90,0)

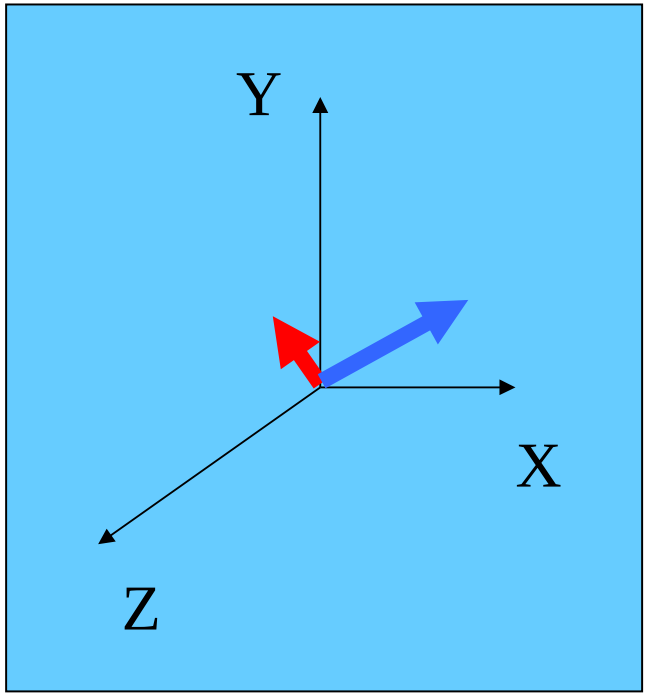




Primeri fiksnih kotov rotacije

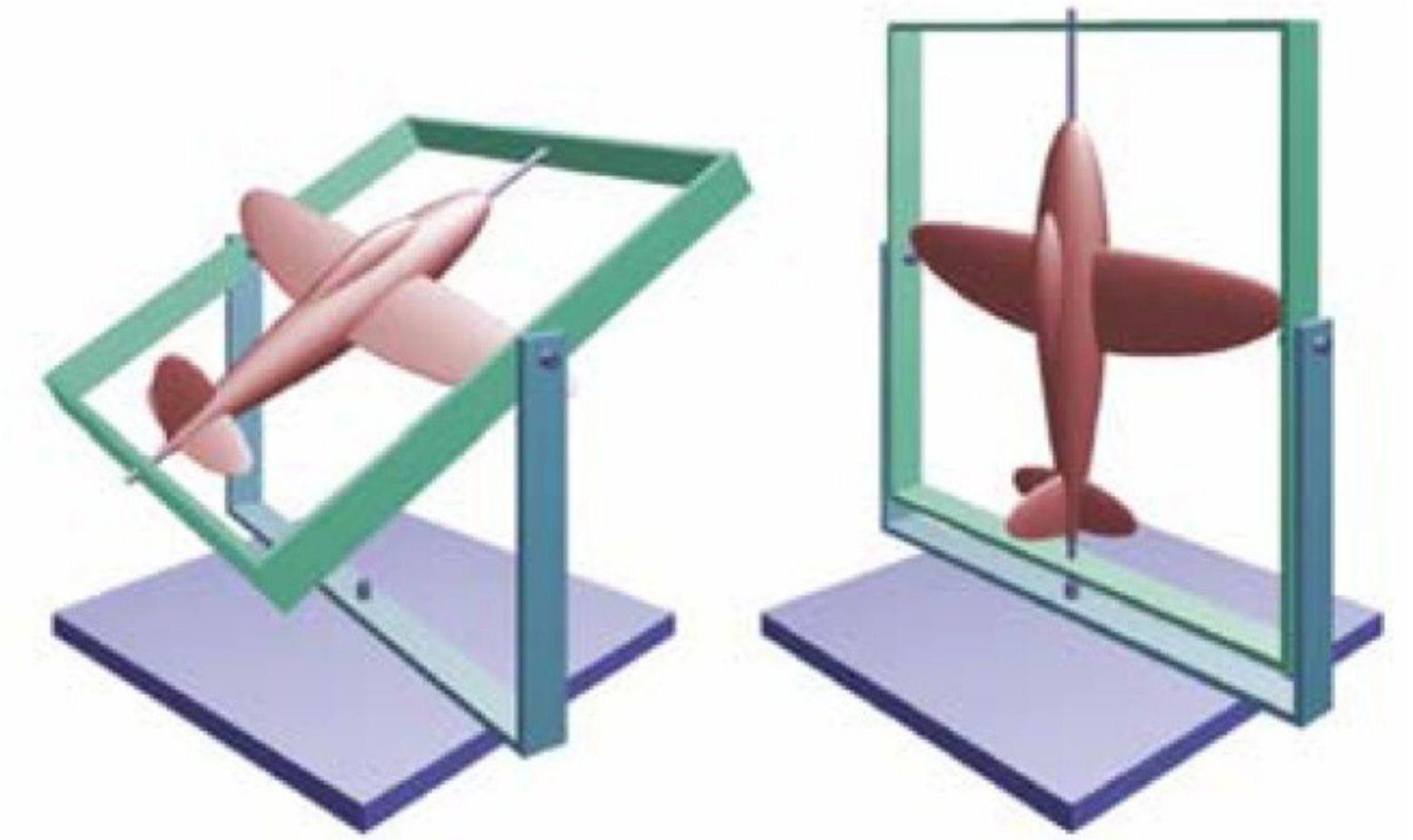
Primer (-45,90,0)





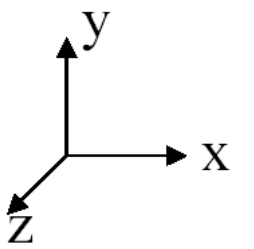
Kardanska zapora (gimbal lock)

- Poravnanje dveh ali več osi vrtenja predmeta pomeni izgubo prostostne stopnje.
- Predmet se ne bo vrtel tako, kot smo si zamislili.

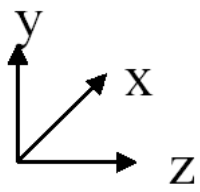


Kardanska zapora

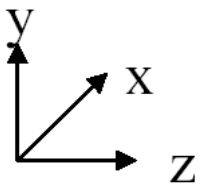
- Vrtenje za kot 90 stopinj okoli osi y v bistvu pomeni, da se prva os vrtenja poravna s tretjo.



Rotx(0)



Roty(90)



Rotz(0)

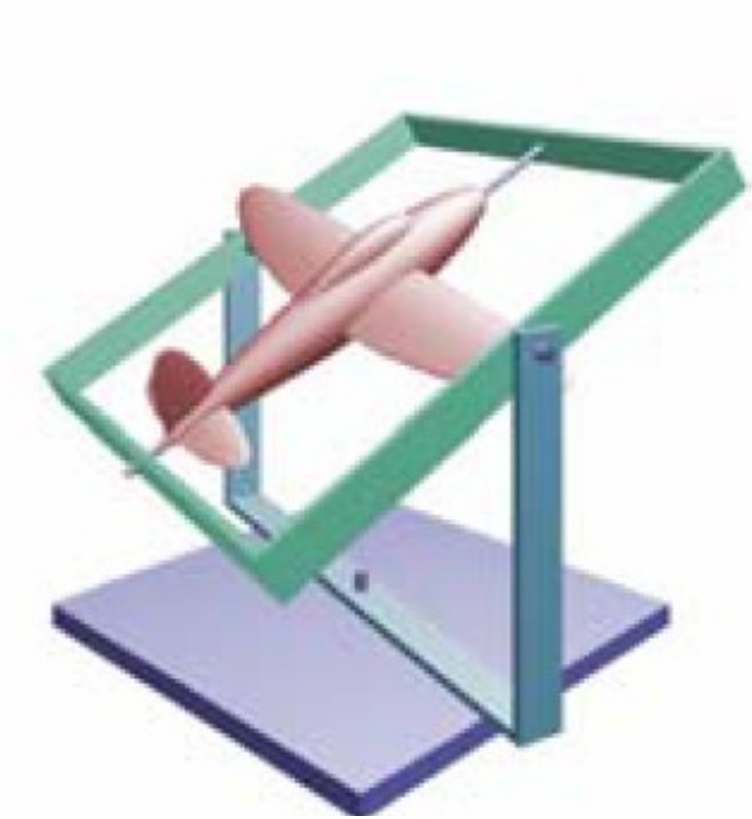
- Inkrementalne spremembe v x,z pripeljejo do enakega rezultata: izgubimo prostostno stopnjo

Eulerjevi koti

An Euler angle is a rotation about a single axis.

Any orientation can be described composing three rotation around each coordinate axis.

Roll, pitch and yaw



Eulerjev teorem

Poljubna rotacija (ali zaporedje rotacij) okrog neke točke je ekvivalentna eni sami rotaciji okrog neke osi skozi to točko.

Eulerjevi koti, β_1, β_2, β_3

$$M = R_z(\beta_3)R_y(\beta_2)R_x(\beta_1)$$

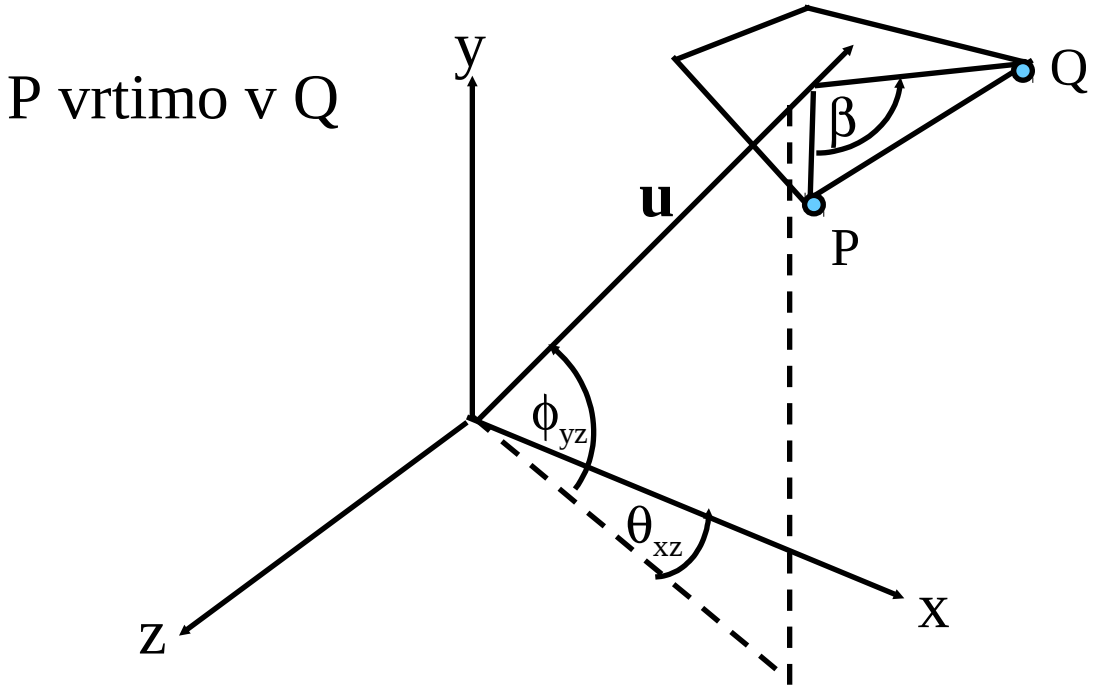
Poljubno 3D rotacijo lahko dosežemo z vrtenji okoli osi x, y in z

Rotacija točke okrog poljubne osi v 3D

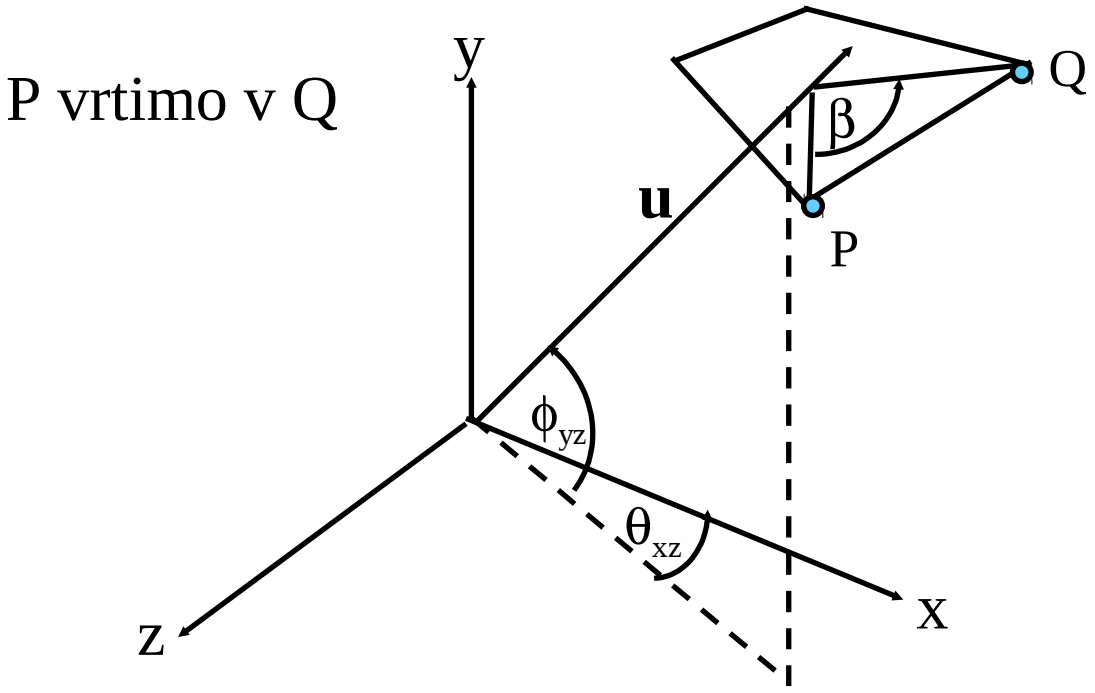
- Imamo podano:

  u = vektor enote v smeri osi vrtenja

  β = kot vrtenja okoli **u**



Rotacija točke okrog poljubne osi v 3D



$$M = R_y(-\theta_{xz})R_z(\phi_{yz})R_x(\beta)R_z(-\phi_{yz})R_y(\theta_{xz})$$

θ_{xz} = kot, projiciran na ravnino xz

ϕ_{yz} = kot, projiciran na ravnino yz

Rotacija točke okrog poljubne osi v 3D

- Imamo podano:

  u = vektor enote v smeri osi vrtenja

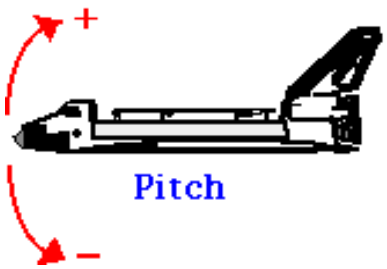
  β = kot vrtenja okoli **u**

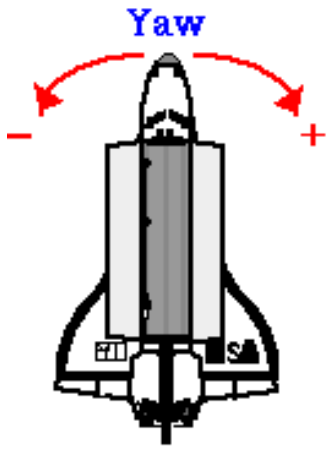
$$\begin{pmatrix} c+(1-c)u_x & (1-c)u_yu_x - su_z & (1-c)u_zu_x + su_y & 0 \\ (1-c)u_xu_y + su_z & c+(1-c)u_y^2 & (1-c)u_zu_y - su_x & 0 \\ (1-c)u_xu_z - su_y & (1-c)u_yu_z + su_x & c+(1-c)u_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

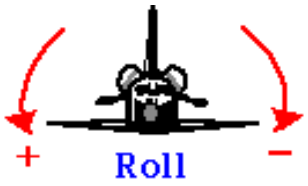
Kjer je $c \equiv \cos(\beta)$ in $s \equiv \sin(\beta)$

Eulerjevi koti

- Splošna rotacija je kombinacija treh osnovnih rotacij: okoli osi x (x-roll) , okoli osi y (y-roll) in okoli osi z (z-roll).







Eulerjevi koti in matrike vrtenja

$$x-roll(\theta_1)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta_1 & \sin\theta_1 & 0\\ 0 & -\sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

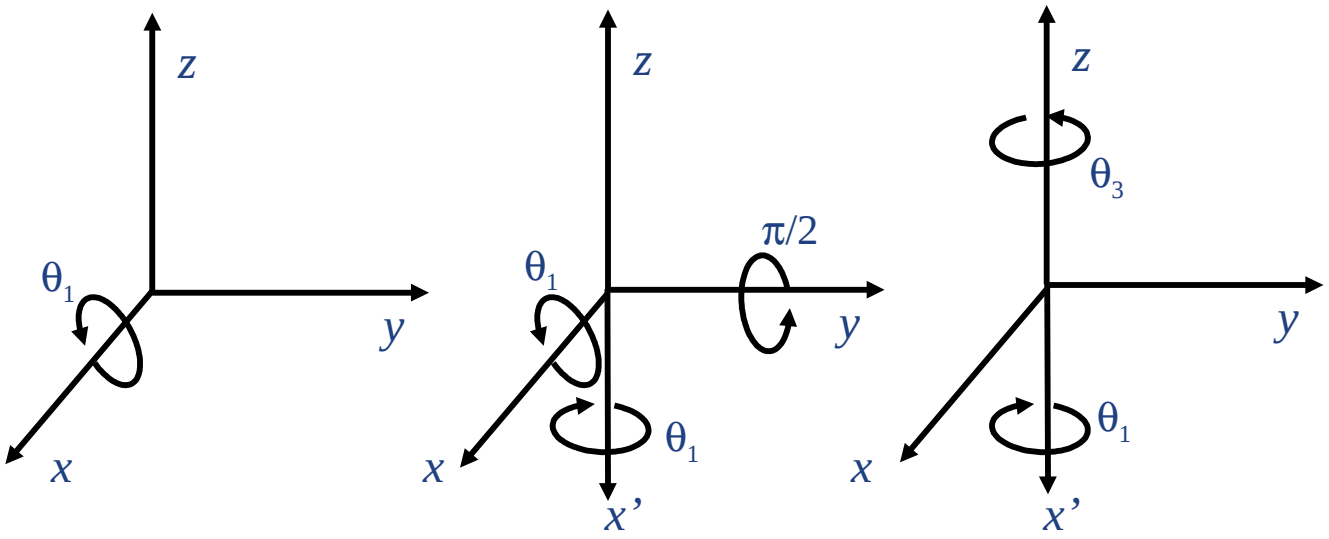
$$\text{y-roll}(\theta_2)=\begin{pmatrix}\cos\theta_2 & 0 & -\sin\theta_2 & 0\\ 0 & 1 & 0 & 0\\ \sin\theta_2 & 0 & \cos\theta_2 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$z-roll(\theta_3)=\begin{pmatrix}\cos\theta_3 & \sin\theta_3 & 0 & 0\\ -\sin\theta_3 & \cos\theta_3 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

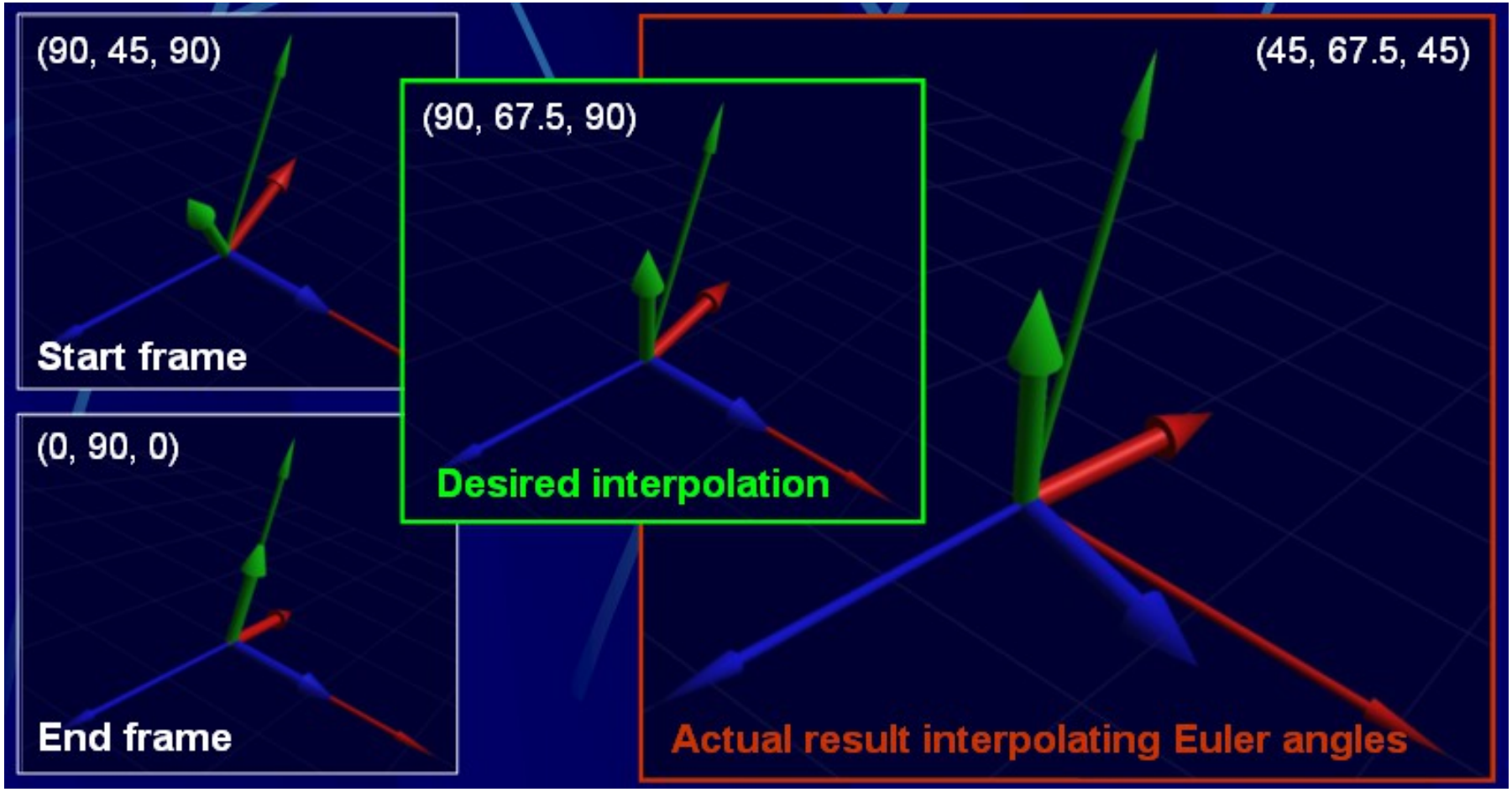
$$R(\theta_1,\theta_2,\theta_3)=\begin{pmatrix}c_2c_3 & c_2s_3 & -s_2 & 0\\ s_1s_2c_3-c_1s_3 & s_1s_2s_3+c_1c_3 & s_1c_2 & 0\\ c_1s_2c_3+s_1s_3 & c_1s_2s_3-s_1c_3 & c_1c_2 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Spet naletimo na kardansko zaporo!

- Rotacija za kot 90^0 povzroči izgubo prostostne stopnje



Interpolacija Eulerjevih kotov →nenaravno gibanje !



Cilj

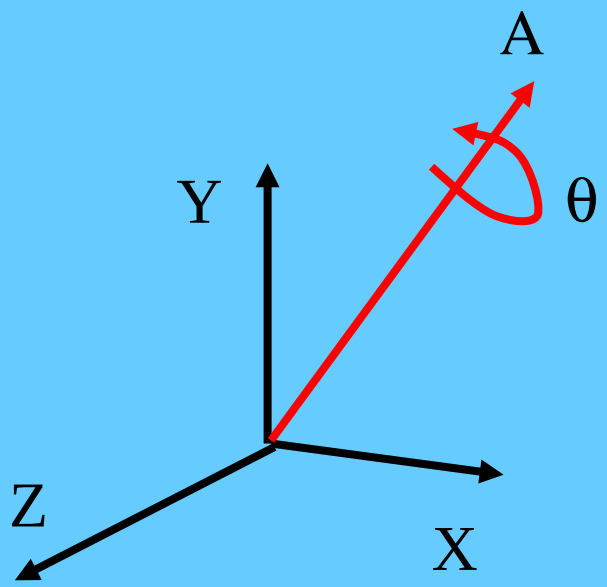
- Iščemo parametrizacijo, pri kateri
  - Obstaja med dvema ključnima rotacijama preprosta in neomajna rotacija
  - Je gibanje neodvisno od izbire koordinatnega sistema

Kot in os

- Katerokoli usmeritev lahko podamo s četvorčkom
  - kot, vektor(x,y,z), pri čemer kot pove, za koliko se zavrtimo okoli osi, ki jo definira vektor
- Ločeno lahko interpoliramo tako kot kot os
- Ni problemov s kardansko zaporo!
- Ne moremo pa učinkovito sestavljati rotacij. Moramo najprej pretvoriti v matrično obliko!

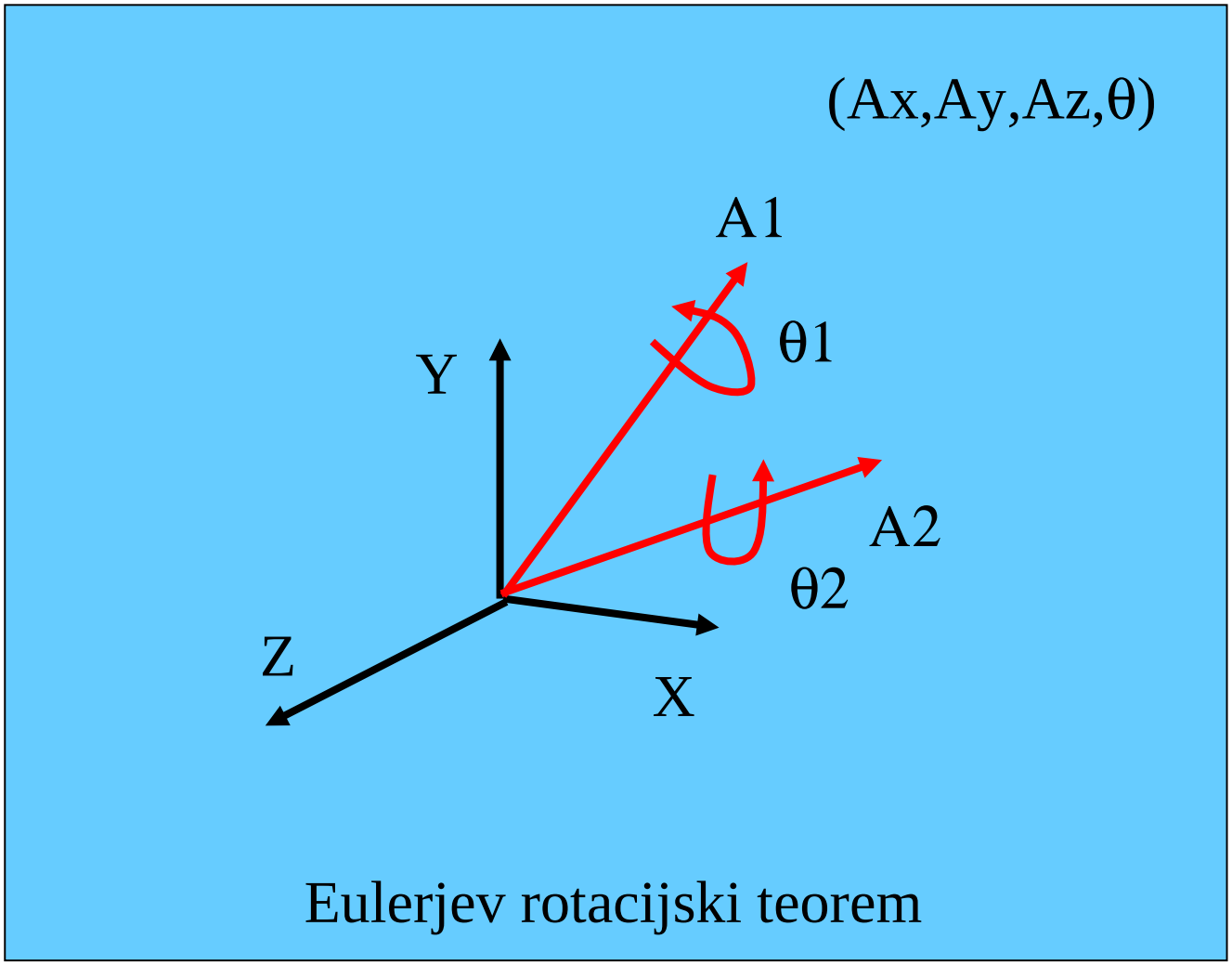
Kot in os

(Ax,Ay,Az,θ)



Eulerjev rotacijski teorem

Kot in os



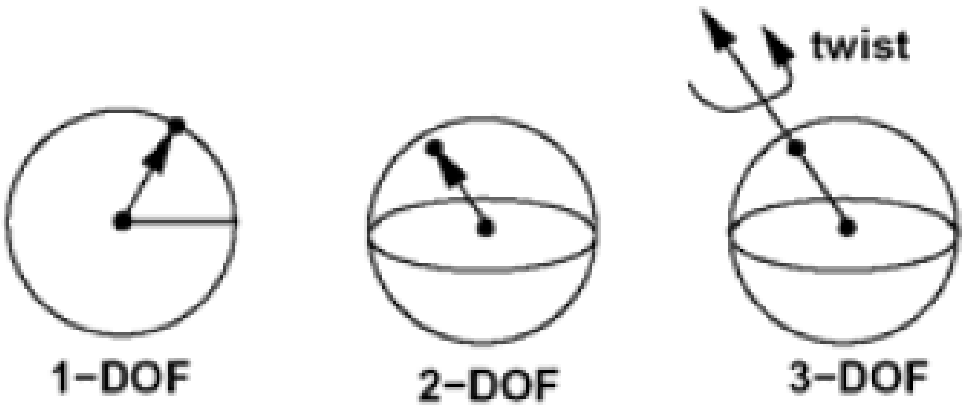
Eulerjev rotacijski teorem

Kvaternioni

Solution: Quaternion Interpolation

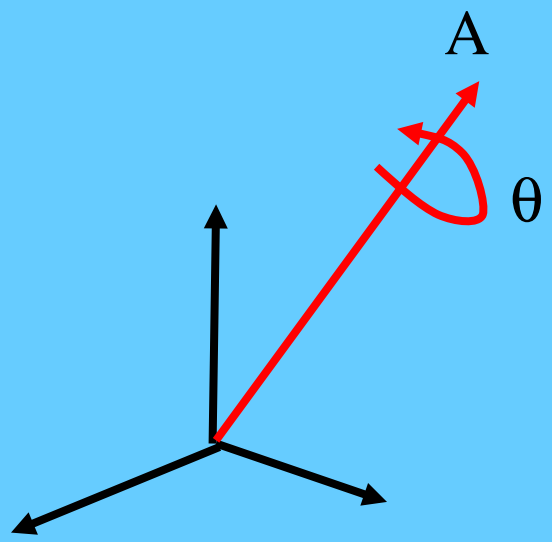
Interpolate orientation on the unit sphere

By analogy: 1-, 2-, 3-DOF rotations as constrained points on 1, 2, 3-spheres



Kvaternioni

$q = [s,v] = [s,x,y,z]$



$(\cos(\theta/2), \sin(\theta/2) * A)$

Kvaternioni

- Razširjajo koncept rotacij iz 3D na 4D.
- Izognejo se problemu “kardanske zapore” in omogočajo mehke in zvezne rotacije.
- V bistvu lahko smatramo, da dodajo še en kot vrtenja sferičnim koordinatam (na primer kotom Longitude (zemljepisna dolžina), Latitude (zemljepisna širina) in Rotation)
- Kvaternion je definiran s 4 realnimi števili |x y z w|. Ta izračunamo s kombinacijo treh koordinat osi vrtenja in kota vrtenja.

Kakšno zvezo imajo kvaternioni in 3D animacija?

- Rešitev problema s “kardansko zaporo"
- Namesto vrtenja predmeta z zaporedjem rotacij omogoča kvaternion vrtenje predmeta preko ene same poljubne osi.
- Ker je os vrtenja podana kot enotin vektor, lahko uporabimo vektorsko matematiko ali sferične koordinate (longitude/latitude).
- Interpolacija kvaternionov: mehko in napovedljivi učinki rotacije.

Motivacija

- Iskanje najbolj naravnega in kompaktnega načina za predstavitev rotacij in usmeritev
- Interpolacija usmeritve, ki vodi v naravno gibanje
- Zaključena matematična oblika, ki obravnava rotacije in usmeritve (razširitev kompleksnih števil)

Definicija kvaternionov

- Razširitev kompleksnih števil
- Četvorčki realnih števil
  - s,x,y,z ali [s,v]
  - s je skalar
  - v je vektortor
- Isti podatki kot pri podajanju kota in osi, vendar v drugi obliki
- Lahko gledamo kot na originalno usmeritev ali kot na rotacijo nekega predmeta

Od kvaternionov k matriki vrtenja

$$Q = \begin{pmatrix} X & Y & Z & W \end{pmatrix}$$

$$M = \begin{bmatrix} 1-2Y^2-2Z^2 & 2XY-2ZW & 2XZ+2YW \\ 2XY+2ZW & 1-2X^2-2Z^2 & 2YZ-2XW \\ 2XZ-2YW & 2YZ+2XW & 1-2X^2-2Y^2 \end{bmatrix}$$

Matematika s kvaternioni

$$[s_1, v_1] \cdot [s_2, v_2] = [s_1 \cdot s_2 - v_1 \bullet v_2, s_1 \cdot v_2 + s_2 \cdot v_1 \times v_2]$$

Točka v prostoru je podana tako: $[0, v]$

$[1, (0,0,0)]$ Multiplikativna identiteta

$$q^{-1} = (1/\|q\|)^2 \cdot [s, -v]$$

Pri čemer velja $\|q\| = \sqrt{s^2 + x^2 + y^2 + z^2}$

$q \cdot q^{-1} = [1, (0,0,0)]$ Kvaternion enotne dolžine

Lastnosti kvaternionov

Konjugacija in absolutna vrednost sta podobni kot pri kompleksnih številih

$$\bar{q} = (s, -v)$$

$$\|q\| = \sqrt{q * \bar{q}} = \sqrt{s^2 + v_x^2 + v_y^2 + v_z^2}$$

- Kvaternioni niso komutativni

$\mathbf{q_1} = (\mathbf{s_1}, \mathbf{v_1})$ $\mathbf{q_2} = (\mathbf{s_2}, \mathbf{v_2})$

$\mathbf{q_1 * q_2} = (\mathbf{s_1 s_2 - v_1 . v_2}\ ,\ \mathbf{s_1 v_2 + s_2 v_1 + v_1 \times v_2})$

- Inverzna vrednost: $$q^{-1} = \frac{\bar{q}}{q * \bar{q}}$$
- Kvaternion enote: $$\|q\| = 1 \Rightarrow q^{-1} = \bar{q}$$

Rotacija s kvaternioni

- S pomočjo kvaternionske matematike želimo vrteti vektor v
  - Vektor predstavimo kot [0,v]
  - Rotacijo s pomočjo kvaternionov predstavimo kot q

$$q = Rot_{\theta,(x,y,z)} = \left[\cos(\theta/2), \sin(\theta/2) \bullet (x, y, z)\right]$$

$$v' = Rot(v) = q \bullet v \bullet q^{-1}$$

Rotacija s kvaternioni

- Rotacija P=(0,r) okrog vektorja enote n za kot θ s pomočjo enotnega kvaterniona q=(s,v)

$$R_q[P] = qPq^{-1} = (0, (s^2 - v \cdot v)r + 2v(v \cdot r) + 2sv \times r)$$

toda q=(cos½θ, sin½θ•n) kjer je |n|=1

$$R_q[P] = (0\,, (\cos^2\theta/2 - \sin^2\theta/2)r + 2n(n \cdot r)\sin^2\theta/2 +$$
$$2(n \times r)\cos\theta/2 \sin\theta/2\,)$$
$$= (0\,, \cos\theta\, r + (1 - \cos\theta)\, n\, (n \cdot r) + (n \times r)\sin\theta)$$

Rotacija s kvaternioni

Veriženje rotacij – rotacija s pomočjo q_1 nato z uporabo q_2 je kot rotacija z uporabo $q_2 * q_1$

$$q_2 * (q_1 * P * q_1^{-1}) * q_2^{-1} = (q_2 * q_1) * P * (q_1^{-1} * q_2^{-1})$$
$$= (q_2 * q_1) * P * (q_2 * q_1)^{-1}$$

Rotacije v praksi

- Rotacije najlažje izrazimo v Eulerjevih kotih ali kot pare os-kot
- Med različnimi vrstami predstavitev lahko prehajamo (jih pretvarjamo)
- Izberemo tisto predstavitev, ki je za dano nalogo najbolj primerna
  - Vnos podatkov:Eulerjevi koti
  - interpolacija: kvaternioni
  - Sestavljanje rotacij: kvaternioni, matrika usmeritve

Dodatna gradiva, vezana na OpenGL

Trije načini uporabe transformacij

- Brute Force: Calculate 2D or 3D transformation matrices then perform Q = M P and draw Q points
- Using glMatrixMult: Calculate 3D transformation matrices, then

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMatrixMultd(M);
glMatrixMultd(N);
Draw P points
```

- Using Opengl transformations: Calculate 3D transformation matrices, then

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotated(angle, 0,0,1);
glTranslated(dx,dy,dz);
glScaled(sx,sy,sz);
Draw P points
```

OpenGL transformacije

- Transformacije OpenGL lahko izvedemo bodisi na matriki modelview ali na projekcijski matriki
- Tekočo matriko nastavimo na modelview ali na projection z naslednjim ukazom:

```
glMatrixMode(GL_MODELVIEW)
or
glMatrixMode(GL_PROJECTION)
```

Uporaba glMatrixMult

glMatrixMult{fd}(const TYPE*M)

Pri tem je

$$M = \begin{bmatrix} m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \\ m_4 & m_8 & m_{12} & m_{16} \end{bmatrix}$$

- Vse matrike in točke izrazimo v 3D
- Izračunamo vsak element transformacijske matrike in to shranimo v enodiman zionalno polje M[16]
- CT = CT*M
- Nrišemo originalne točke P

Primer

- Izračunamo transformacijsko matriko M
- Uporabimo MatrixMult za posodobitev CT
- Narišemo točke P

```
void myDisplay(void)
{
        Point p[10], pl[4];
        double M[16], M1[16];
        int i;
// data
        p[0].x = -1.0;
        p[0].y = 3.0;
        p[0].z = 0.0;
        p[0].r = 1;
        p[1].x = -1;
        p[1].y = -2;
        p[1].z = 0;
        p[1].r = 1;

        ...
```

```
//transformation matrix 0 by column
M[0] = cos(10*3.14159/180);
M[1] = -sin(10*3.14159/180);
M[2] = 0;
M[3] = 0;
M[4] = sin(10*3.14159/180);
M[5] = cos(10*3.14159/180);

M[6] = 0;
M[7] = 0;
M[8] = 0;
M[9] = 0;
M[10] = 1;
M[11] = 0;
M[12] = 0;
M[13] = 0;
M[14] = 0;
M[15] = 1;
```

```
//transformation matrix 1 by column
M1[0] = 1;
M1[1] = 0;
M1[2] = 0;
M1[3] = 0;
M1[4] = 0;
M1[5] = 1;

M1[6] = 0;
M1[7] = 0;
M1[8] = 0;
M1[9] = 0;

M1[10] = 1;
M1[11] = 0;
M1[12] = 3;
M1[13] = 2;

M1[14] = 0;
M1[15] = 1;
```

```
//draw transformed shape CT = CT(M)(M1)
    glColor3f(1.0,0.0,1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glMatrixMultd(M);
    glMatrixMultd(M1);

    glBegin(GL_POLYGON);
      for(i=0;i<10;i++)
              glVertex2d(p[i].x,p[i].y);
    glEnd();
    glFlush();
}
```

Opazka: rišemo originalne točke P, ne transformiranih točk Q, ker je matrika Modelview avtomatično uporabljena na vseh P točkah

Transformacije OpenGL

- glRotated (angle, 0,0,1)
- glTranslated (dx,dy, dz)
- glScaled (sx,sy,sz)

Naknadno množenje trenutne transformacijske matrike

CT = CT*Rotate*Translate*Scale

Pripelje do

Q(x,y,z) = CT*Rotate*Translate*Scale*P(x,y,z)

Isti vrstni red kot pri MatrixMult

Shranjevanje CT za kasnejšo uporabo

- Po več zaporednih matričnih množenjih se lahko kopičijo napake zaokrožanja
- Morda si želimo povratka na prejšnji CT

```
For(I=0; I<num;  I=++)
{
        glPushMatrix();              //remember the CT
        glTranslate (…);
        glRotate(…);
        drawfigure();
        glPopMatrix();               //restore the CT
}
```

Nastavitev kamere v OpenGL za paralelne projekcije

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left,right,bottom, top, near,far);
```

Volumen gledanja (view volume) postavimo z definiranjem parametrov left, right, bottom, top, near, far

Pozicioniranje in usmeritev kamere

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt (eye.x,eye.y,eye.z,look.x,look.y,look.z,up.x,up.y,up.z);
```

To nastavimo pred uporabo transformacij.