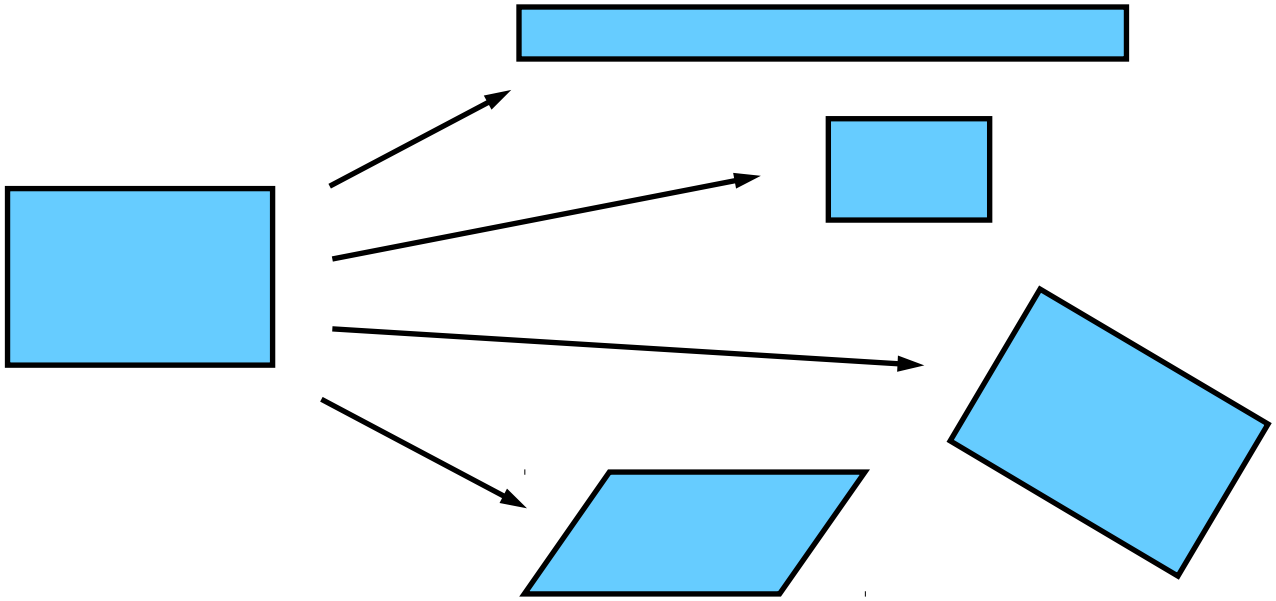
2D Transformacije

Zakaj potrebujemo transformacije?

- Animacija
- Več instanc istega predmeta, variacije istega objekta na sceni
- Tvorba kompliciranih predmetov iz bolj preprostih
- Transformacije gledanja

Kaj je transformacija v rač. grafiki?

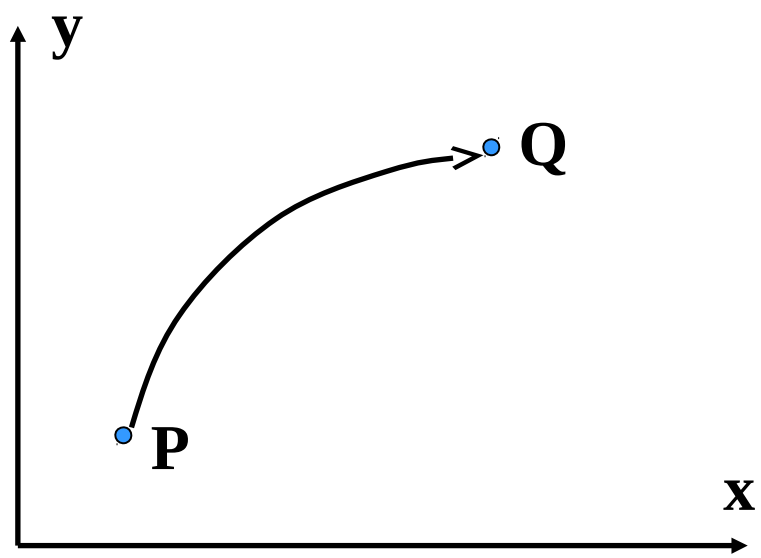
- Geometrične transformacije s pomočjo primerne formule ali algoritma spremenijo koordinate predmetov.
- Uporabljamo jih za izražanje sprememb lokacij, usmeritev, velikosti in oblike predmetov.



2D Transformacije

Z drugimi besedami:

- Transformacija je funkcija, ki vzame točko ali vektor in preslika to točko ali vektor v neko drugo točko ali vektor:



$$Q = T(P)$$

$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = T \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$

Q je slika **P**
Zaradi preslikave **T**

Osnovne transformacije

- *Translacija*: premik objekta z ene na drugo lokacijo, npr.: premik skozi sobo ali okolico.

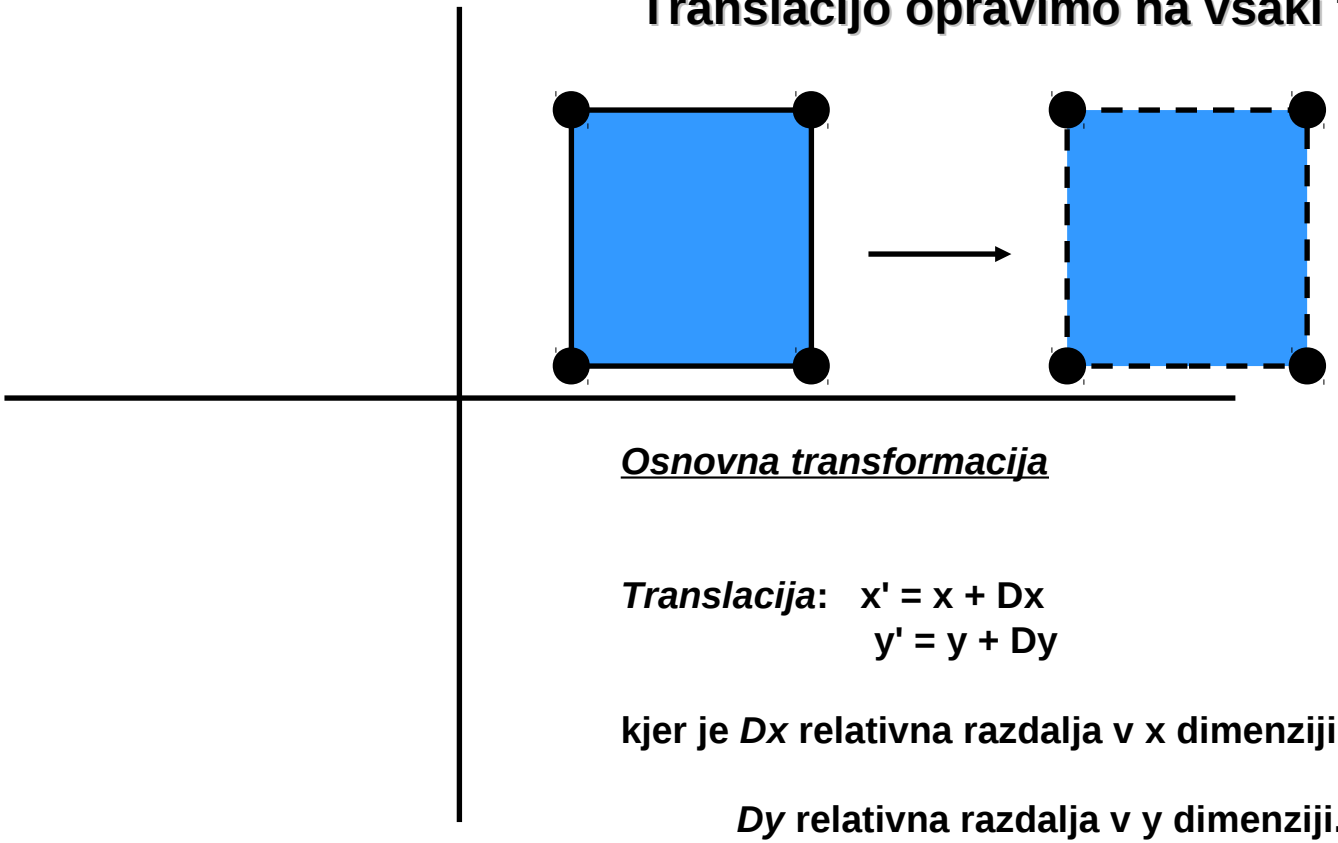
- *Rotacija*: sprememba orientacije objekta na določeni lokaciji, npr.: vrtenje okrog lastne osi.

- Povečava *(skaliranje)*: sprememba velikosti objekta, kot se pojavi na zaslonu, npr.: predmet se poveča ali zmanjša.
- Zrcaljenje
- Striženje (shear)
- Identiteta

- *"Clipping"*: vedeti, kje nehati risati predmet, ker je njegov del izven zaslona.

Translacija

Translacijo opravimo na vsaki točki



Osnovna transformacija

***Translacija*: x' = x + Dx**
y' = y + Dy

kjer je *Dx* relativna razdalja v x dimenziji,

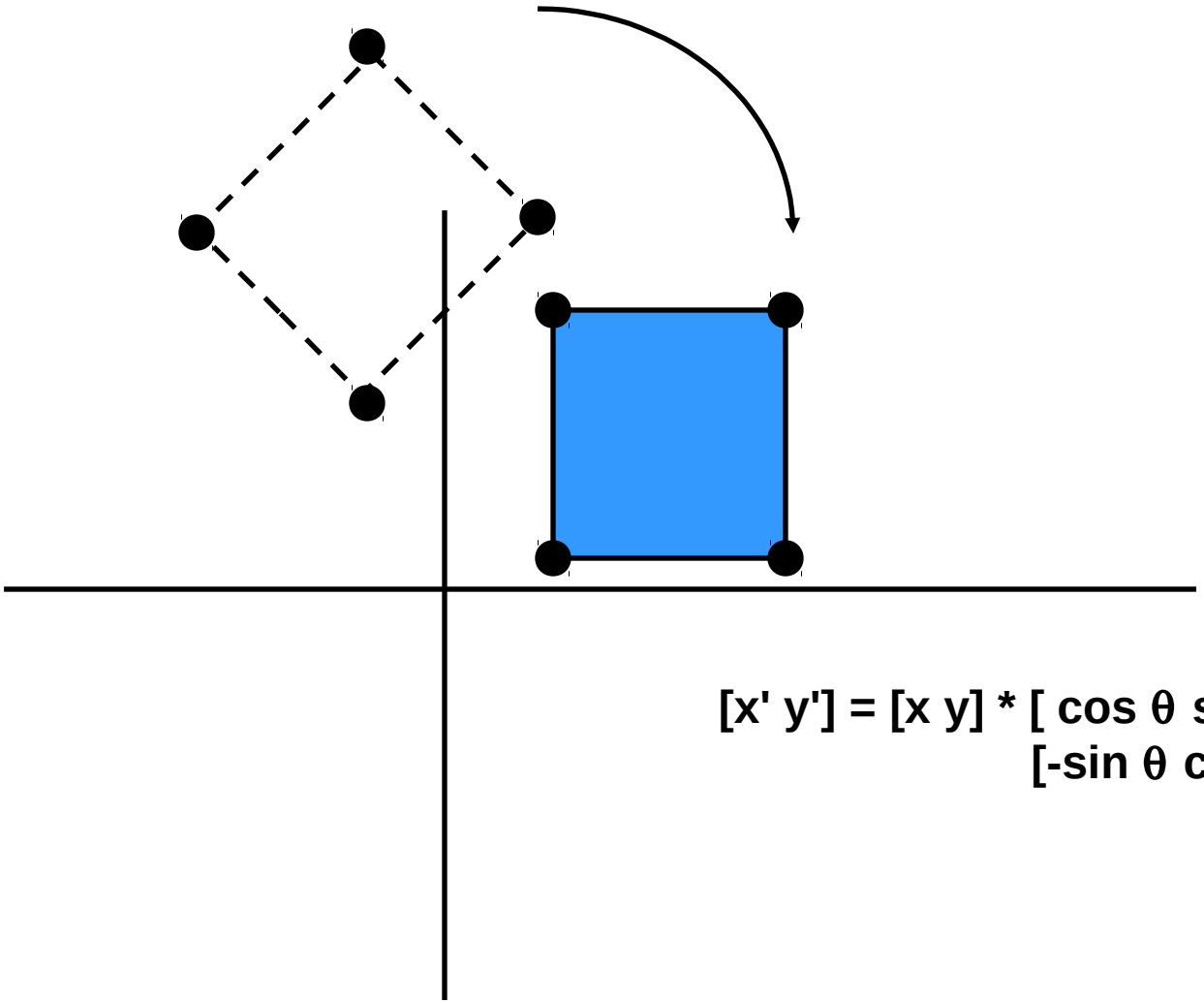
***Dy* relativna razdalja v y dimenziji.**

Izračun:

[x' y'] = [x y] + [Dx Dy]

P' = P + T

Rotacija



$$[x'\ y'] = [x\ y] * \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Rotacija

***Rotacija*:** $x' = x\cos\theta - y\sin\theta$
$y' = x\sin\theta + y\cos\theta$

kjer je θ kot rotacije.

Izračun:

$$[x'\ y'] = [x\ y] * \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$P' = P * R$$

***Pozitivni* koti so v nasprotni smeri urinega kazalca od x proti y;**
za *negatvne* uporabimo identiteti:

$$\cos(-\theta) = \cos\theta\ ,$$

in

$$\sin(-\theta) = -\sin\theta$$

Rotacija—Okrog fiksne točke

Pri rotaciji:

- **velika razlika med:**

  "rotacijo okrog središčne točke objekta"

  in

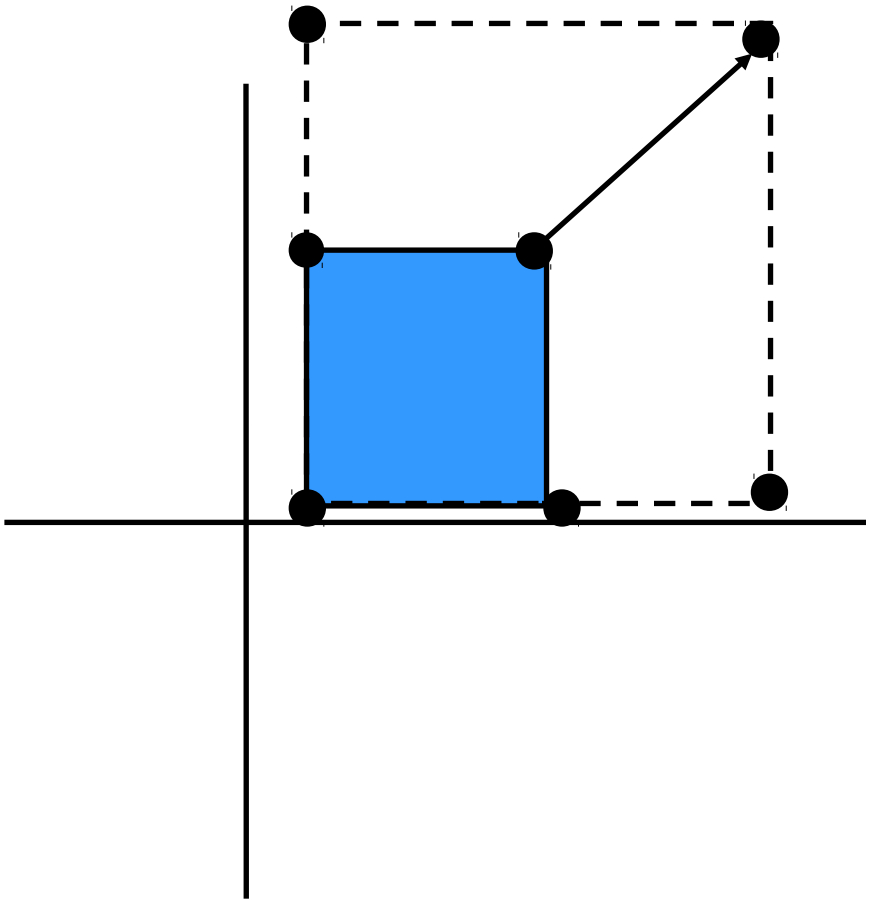
  "rotacijo okrog izhodišča kartezičnega sveta"

***Primer*:**
- **imamo kroglo z vrvjo pritrjeno na palico**
- **ali želimo, da se krogla vrti okrog svoje osi?**
- **ali pa želimo, da se krogla z vrvico vrti okrog palice?**

Za rotacijo objekta okrog središčne točke:

- **najprej translacija objekta v izhodišče,**
- **nato rotacija,**
- **nato translacija nazaj.**

Povečava (scalling)



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

***Skaliranje*:** $x' = x * Sx$

$y' = y * Sy$

kjer je *Sx* skalirni faktor za x dimenzijo,

***Sy* skalirni faktor za y dimenzijo.**

Izračun:

S je definirana kot $\begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$

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

$$P' = P * S$$

2D Transformacije: povzetek

Translacija: P' = P+T
Skaliranje: P' = P*S
Rotacija: P' = P*R

Problem:

- **Imamo *heterogene* operacije (seštevanje in množenje)**

- **Želimo *homogene* operacije (vse samo množenje), tako da lahko transformacije združujemo**

***Primer*:**

- ***Skaliranje* objekta med premikanjem (*translacijo*) in vrtenjem (*rotacija*).**

- **Želimo *eno samo* kompleksno operacijo, ne *tri* individualne.**

Homogene operacije

Koordinatni sistem lahko izboljšamo tako, da dodamo še eno dimenzijo.

- **Tako lahko razne operacije izvajamo samo z množenjem matrik.**
- **Torej: da dobimo homogene operacije, uporabimo [D+1 x D+1] matrike (D je dimenzija):**
  - **za 2D 3x3 matrike**
  - **za 3D 4x4 matrike**

2D skaliranje: matrika

enako kot prej

***2D skaliranje*:**

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

P' = P * S(Sx,Sy), S je zgornja matrika

Zaporedne operacije lahko množimo.

$$[x''\ \ y''\ \ 1] = [x\ y\ 1] * \begin{bmatrix} Sx1Sx2 & 0 & 0 \\ 0 & Sy1Sy2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2D rotacija: matrika

enako kot prej

***2D rotacija*:**

$$[x'\ y'\ 1] = [x\ y\ 1] * \begin{bmatrix} \cos B & \sin B & 0 \\ -\sin B & \cos B & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

P' = P * R(B), R je zgornja matrika

Zaporedne rotacije lahko seštevamo.

P -> P'(B) -> P"(A):

$$[x''\ y''\ 1] = [x\ y\ 1] * \begin{bmatrix} \cos B+\cos A & \sin B+\sin A & 0 \\ -\sin B-\sin A & \cos B+\cos A & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2D translacija: matrika

***2D translacija*:**

[x' y' 1] = [x y 1] * [1 0 0]
[0 1 0]
[Dx Dy 1]

P' = P * T(Dx, Dy), T je zgornja matrika.

Zaporedne operacije lahko seštevamo.

P -> P‘ -> P":

[x" y" 1] = [x y 1] * [1 0 0]
[0 1 0]
[Dx1+Dx2 Dy1+Dy2 1]

Osnovne 2D afine transformacije

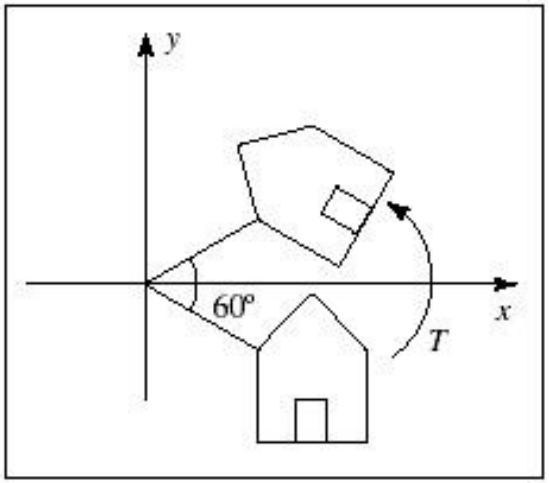
Translacija

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

Povečava

$$\begin{pmatrix} Sx & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotacija



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

Inverzne transformacije afinih transf.

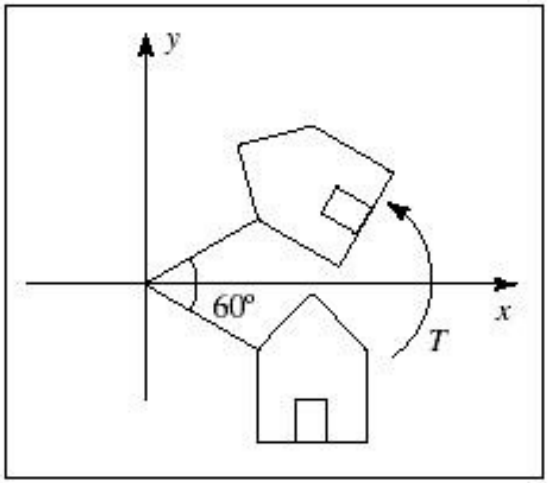
Translacija

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

Pomanjšava

$$\begin{pmatrix} \frac{1}{Sx} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotacija



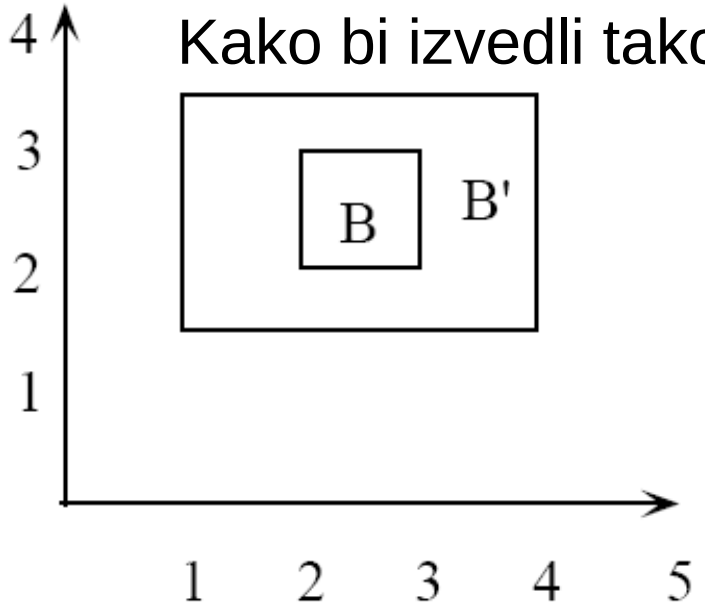
$$\begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Sestavljanje afinih transformacij

- Pogosto uporabljamo zaporedje elementarnih transformacij in tako sestavimo kompleksno transformacijo.
- Procesu zaporedne uporabe več transformacij za oblikovanje splošne transformacije pravimo sestavljanje ali veriženje transformacij.
- Rezultat sestavljanja dveh afinih transformacij je afina transformacija.

Sestavljanje afinih transformacij

Kako bi izvedli tako povečavo



- *Premik B tako, da bo poravnan z izhodiščem*

tx = , ty =

- *Povečava*

sx = , sy =

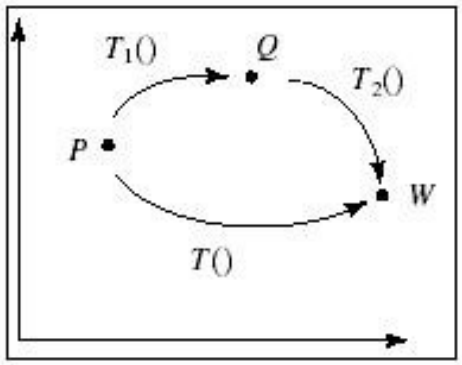
- *Premik nazaj na prejšnje mesto*

tx = , ty =

Sestavljanje (veriženje) afinih transformacij

Primer:

- *Translacija za(3, -4)*
- *Nato rotacija za 30 stopinj*
- *Nato povečava za (2, -1)*
- *Nato translacija za (0.15)*
- *In končno rotacija za –30 stopinj*

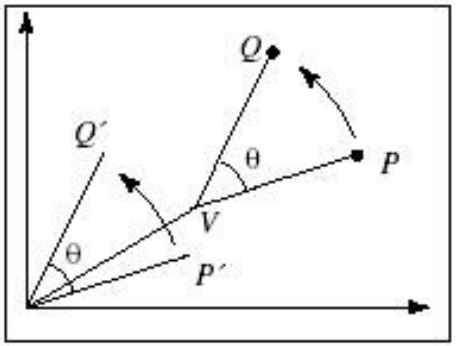


$$W = T_2\left(T_1\left(P\right)\right)$$

$$W = M\left(P\right) = \left(M_2 M_1\right)\left(P\right)$$

Če uporabljamo homogene koordinate, sestavimo afine transformacije s preprostim množenjem matrik

Primer : rotacija okoli poljubne točke



1. Premik točke **P** v izhodišče preko vektorja **v = (-Vx, -Vy)**
2. Rotacija okrog izhodišča za kot theta
3. Premik **P** nazaj za vektor **v**

$$\begin{pmatrix} 1 & 0 & V_x \\ 0 & 0 & V_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -V_x \\ 0 & 0 & -V_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & d_x \\ \sin(\theta) & \cos(\theta) & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

Sestavljanje afinih transformacij

Naj matrika $\mathbf{M}_1$ predstavlja transformacijo T1 in matrika $\mathbf{M}_2$ naj predstavlja transformacijo T2:

$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = \mathbf{M}_1 \cdot \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix} \qquad \begin{pmatrix} Wx \\ Wy \\ 1 \end{pmatrix} = \mathbf{M}_2 \cdot \begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix}$$

Potem je $\begin{pmatrix} Wx \\ Wy \\ 1 \end{pmatrix} = \mathbf{M} \cdot \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$, pri čemer je $\mathbf{M} = \mathbf{M}_2 \cdot \mathbf{M}_1$

Sestavljanje afinih transformacij

Ker množenje matrik ni kumulativno, je vrstni red transformacij pomemben

$$\mathbf{M}_2 \cdot \mathbf{M}_1 \cdot \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix} \neq \mathbf{M}_1 \cdot \mathbf{M}_2 \cdot \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$

Afine Transformacije

- Najbolj pogoste transformacije v računalniški grafiki
- Omogočajo enostavno premikanje, vrtenje in povečavo likov
- Zaporedje afinih transformacij lahko sestavimo v eno splošno afino transformacijo
- Afine transformacije nudijo kompaktno matrično predstavitev

Afine transformacije so linearne

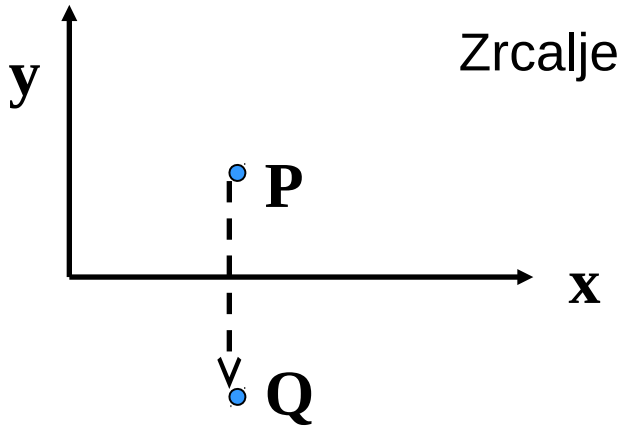
- Ravne črte ostanejo ravne
- Vzporedne črte ostanejo vzporedne
- Relativna razmerja se ohranijo
- Transformirana površina = |det M| originalne površine

Še preostale afine transformacije

Zrcaljenje (reflection)

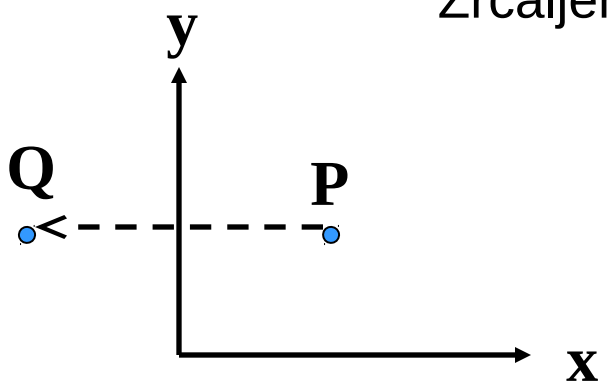
- Zrcaljenje tvori zrcalno sliko predmetov
- Običajno zrcalimo preko osi zrcaljenja
- Do tega pridemo s spremembo predznaka bodisi x bodisi y koordinat predmeta

Zrcaljenje preko koordinatne osi



Zrcaljenje preko osi x

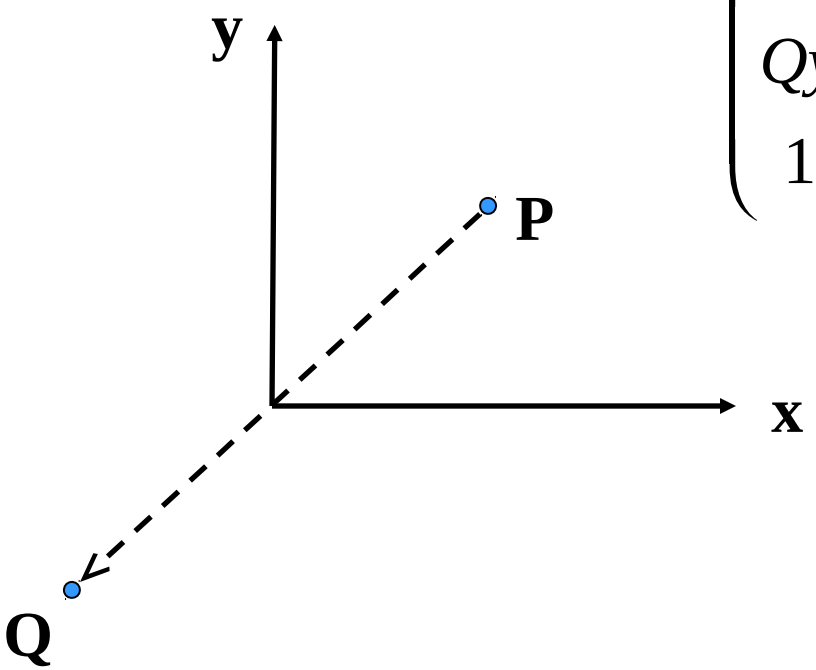
$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$



Zrcaljenje preko osi y

$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$

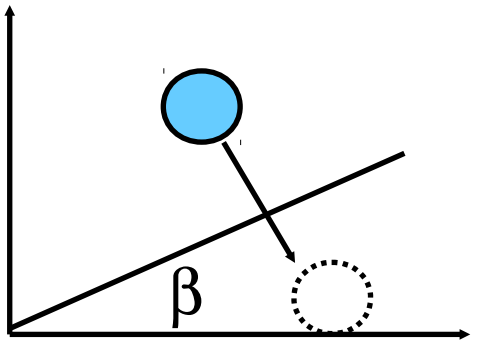
Zrcaljenje preko izhodišča



$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$

Primer zrcaljenja preko poševne črte

1. Rotacija za kot -β
2. Zrcaljenje preko osi x
3. Rotacija nazaj za kot β

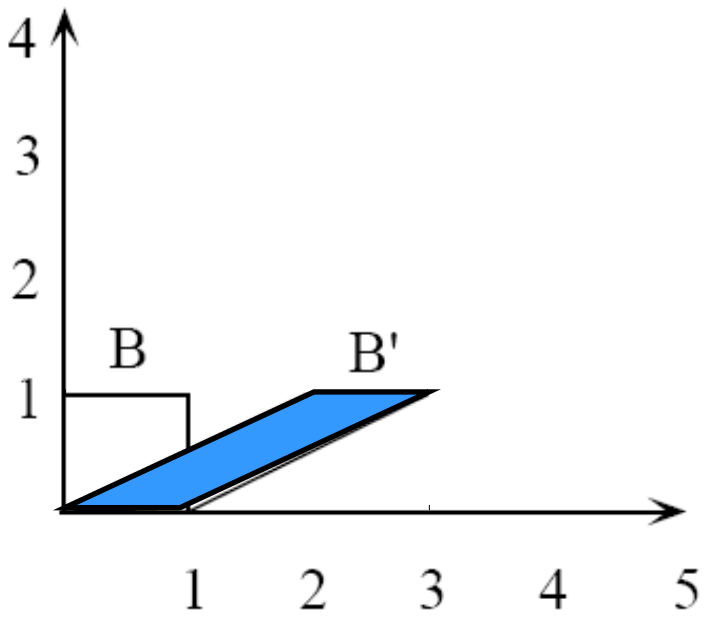


$$\begin{pmatrix}\text{Rotate back}\\ \text{through } \beta\end{pmatrix}\begin{pmatrix}\text{Reflect about}\\ \text{x}-\text{axis}\end{pmatrix}\begin{pmatrix}\text{Rotate through}\\ -\beta\end{pmatrix}$$

Vrstni red je pomemben

$$\begin{pmatrix}\text{Rotate back}\\ \text{through } \beta\end{pmatrix}\left[\begin{pmatrix}\text{Reflect about}\\ \text{x}-\text{axis}\end{pmatrix}\begin{pmatrix}\text{Rotate through}\\ -\beta\end{pmatrix}\right]$$

Striženje (shear)



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & g & 0 \\ h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

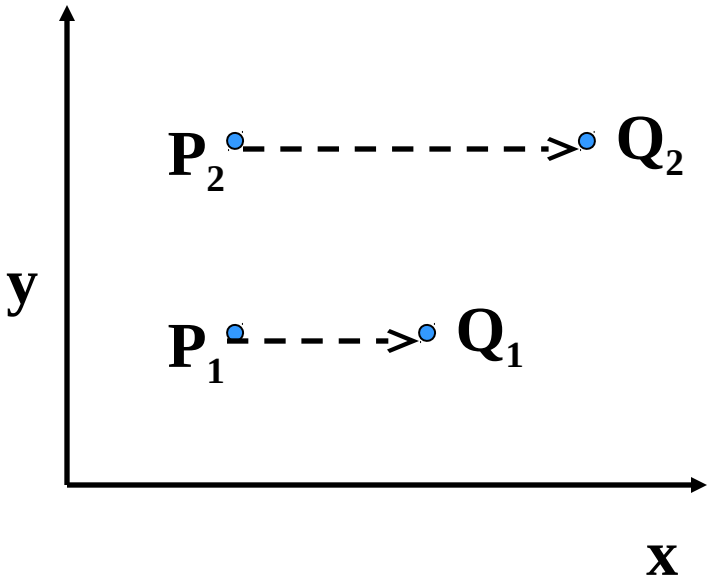
$$x' = x + g\,y$$
$$y' = y + h\,x$$

$$g = 2$$
$$h = 0$$

Striženje (shear)

Striženje spremeni podobo predmeta v odvisnosti od razdalje.

Striženje v smeri x



$$\begin{pmatrix} Qx_1 \\ Qy_1 \\ 1 \end{pmatrix} = \begin{pmatrix} Px_1 + h \cdot Py_1 \\ Py_1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} Qx_2 \\ Qy_2 \\ 1 \end{pmatrix} = \begin{pmatrix} Px_2 + h \cdot Py_2 \\ Py_2 \\ 1 \end{pmatrix}$$

Striženje (shear)

Splošna oblika matrike:

$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & h & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$

Pri tem je h faktor striženja v smeri x,
g je faktor striženja v smeri y

Identiteta

Poseben primer transformacij: nobene spremembe originalnega predmeta.

$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} Qx \\ Qy \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Px \\ Py \\ 1 \end{pmatrix}$$

Tej transformacijski matriki pravimo matrika identitete.

Programski primer

```
// Calculate the transformation matrix M
// Apply M to each point to transform P to Q
// Draw the Q points

void myDisplay(void) {
    Point p[10], q[10];
    double M[3][3], M1[3][3], tempx, tempy, tempr;
    // data
    p[0].x = -1.0;
    p[0].y = 3.0;
    p[0].r = 1;
    p[1].x = -1;
    p[1].y = -2;
    p[1].r = 1;

    . . .
```

```
//transformation matrix 0
    M[0][0] = cos(10*3.14159/180.0);
    M[0][1] = -sin(10*3.14159/180.0);
    M[0][2] = 0;
    M[1][0] = sin(10*3.14159/180.0);
    M[1][1] = cos(10*3.14159/180.0);
    M[1][2] = 0;
    M[2][0] = 0;
    M[2][1] = 0;
    M[2][2] = 1;
//transformation matrix 1
    M1[0][0] = 1;
    M1[0][1] = 0;
    M1[0][2] = 2;
    M1[1][0] = 0;
    M1[1][1] = 1;
    M1[1][2] = 1;
    M1[2][0] = 0;
    M1[2][1] = 0;
    M1[2][2] = 1;
```

```
//draw transformed shape Q = (M1)(M)P
    for( i=0;i<10;i++) {
      tempx = M[0][0]*p[i].x + M[0][1]*p[i].y + M[0][2]*p[i].r;
      tempy = M[1][0]*p[i].x + M[1][1]*p[i].y + M[1][2]*p[i].r;
      tempr = M[2][0]*p[i].x + M[2][1]*p[i].y + M[2][2]*p[i].r;
      q[i].x = M1[0][0]*tempx + M1[0][1]*tempy + M1[0][2]*tempr;
      q[i].y = M1[1][0]*tempx + M1[1][1]*tempy + M1[1][2]*tempr;
      q[i].r = M1[2][0]*tempx + M1[2][1]*tempy + M1[2][2]*tempr;
    }
    glBegin(GL_POLYGON);
    for(i=0;i<10;i++)
      glVertex2d (q[i].x, q[i].y);
    glEnd();
    glFlush();
}
```

Draw the transformed points, Q