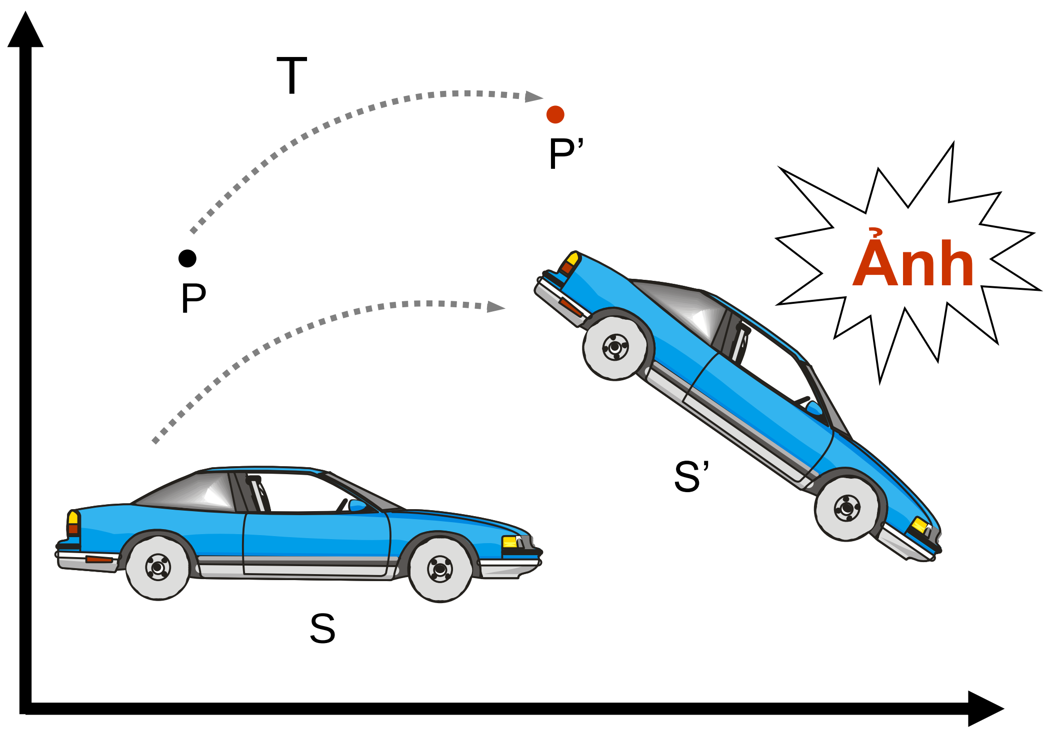
ĐỒ HỌA 2D

CÁC PHÉP BIẾN ĐỔI

Phép biến đổi là gì ?

Phép biến đổi là một ánh xạ từ không gian R^2 vào R^2 :

- Biến một điểm P thành một điểm P'
- Biến một đối tượng S thành đối tượng S'



Công thức

Biến đổi dạng ánh xạ

$$T : R^2 \rightarrow R^2$$
$$P \mapsto P'$$

Biến đổi dạng hàm

$$P' = T(P)$$

hay

$$\begin{cases} P_x^{'} = T_x(P_x, P_y) \\ P_y^{'} = T_y(P_x, P_y) \end{cases}$$

Định nghĩa biến đổi affine

Phép biến đổi affine là phép biến đổi mà các hàm biến đổi T_x, T_y có dạng tuyến tính.

$$\begin{cases} T_x(x, y) = \mathbf{a}x + \mathbf{c}y + \mathbf{e} \\ T_y(x, y) = \mathbf{b}x + \mathbf{d}y + \mathbf{f} \end{cases}$$

Hệ tọa độ thuần nhất

Hệ tọa độ Đề các	Hệ tọa độ thuần nhất
$P(P_x, P_y)$	$P(P_x, P_y, 1)$
$P(P_x, P_y)$	$P(wP_x, wP_y, w)$

Công thức xác định ảnh của một điểm

Dạng hàm

$$\begin{cases} P'_x = aP_x + cP_y + e \\ P'_y = bP_x + dP_y + f \end{cases}$$

Dạng ma trận

$$P' = P.M$$

Ma trận biến đổi

hoặc

$$\begin{pmatrix} P'_x & P'_y & 1 \end{pmatrix} = \begin{pmatrix} P_x & P_y & 1 \end{pmatrix} . \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix}$$

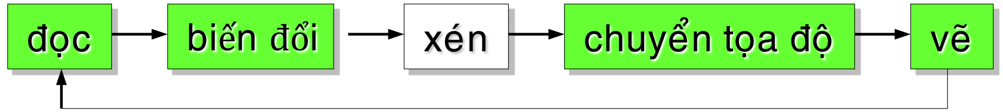
Cấu trúc dữ liệu

```
// Lưu thông tin phep bien doi affine
struct TAffine2D {
      double M[3][3];
};
```

Cài đặt

```
TPoint2D TransformPoint2D(TAffine2D T, TPoint2D P)
{
	TPoint2D Q;

	Q.x = T.M[0][0]*P.x + T.M[1][0]*P.y + T.M[2][0];
	Q.y = T.M[0][1]*P.x + T.M[1][1]*P.y + T.M[2][1];

	return Q;
}
```

Cài đặt




```
void Read_Transform_Convert_Draw_2D(CDC *pDC, char *filename, TAffine2D T)
{
        . . .
        TPoint2D P1, P2;
        CPoint   Q1, Q2;
        int      r, g, b;
        // Doc doan thang tu tap tin
        f >> P1.x >> P1.y >> P2.x >> P2.y >> r >> g >> b;
        // Bien doi doan thang
        P1 = TransformPoint2D(T, P1);
        P2 = TransformPoint2D(T, P2);
        // Chuyen toa do doan thang
        Q1 = ConvertWorldToScreen2D(P1);
        Q2 = ConvertWorldToScreen2D(P2);
        // Ve doan thang
        pDC->MoveTo(Q1);
        pDC->LineTo(Q2);
        . . .
}
```

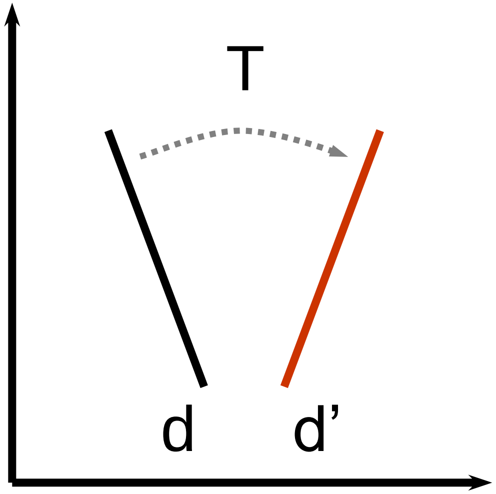
Tính chất của biến đổi affine

Một phép biến đổi affine luôn có 3 tính chất

- Bảo toàn tính thẳng
- Bảo toàn tỉ lệ
- Bảo toàn song song

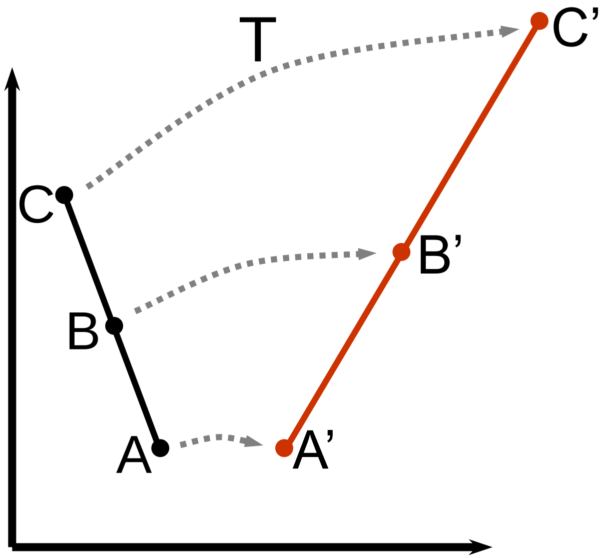
Bảo toàn tính thẳng

GT	d là đường thẳng d' = T(d)
KL	d' là đường thẳng



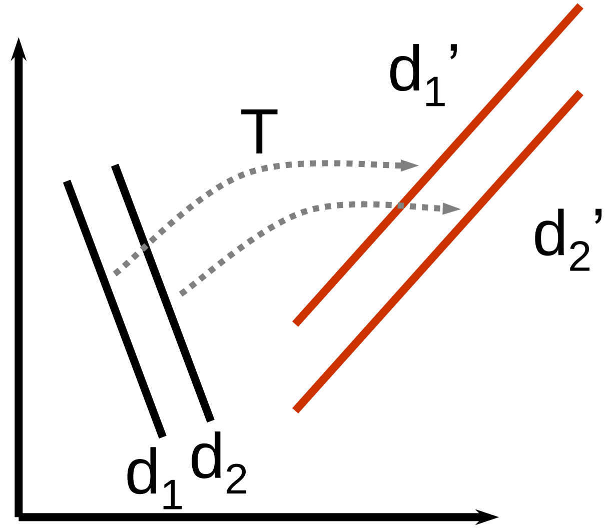
Bảo toàn tỉ lệ

GT	A, B, C là 3 điểm thẳng hàng A' = T(A) B' = T(B) C' = T(C)
KL	A':B':C' = A:B:C



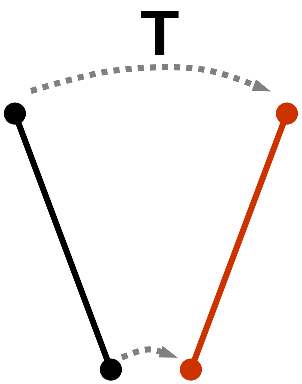
Bảo toàn song song

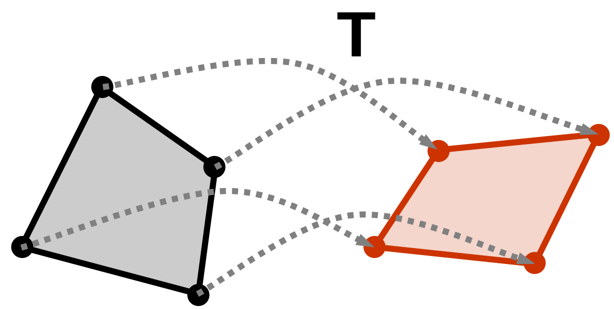
GT	$d_1 // d_2$ $d_1' = T(d_1)$ $d_2' = T(d_2)$
KL	$d_1' // d_2'$

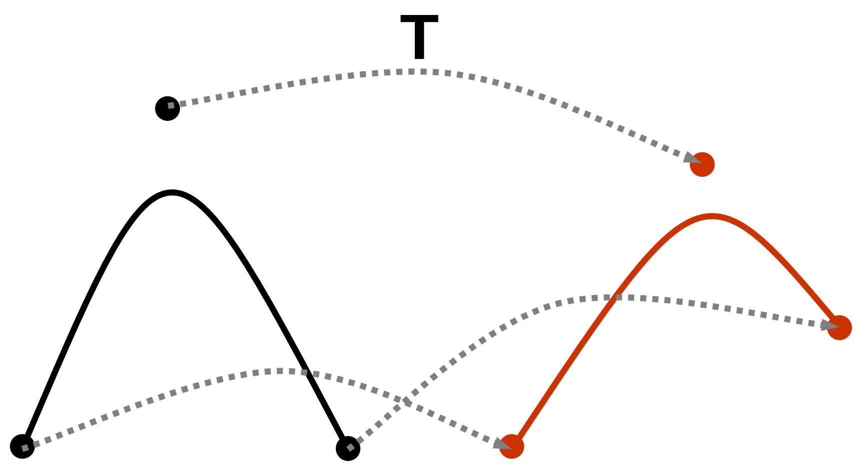


Hệ quả

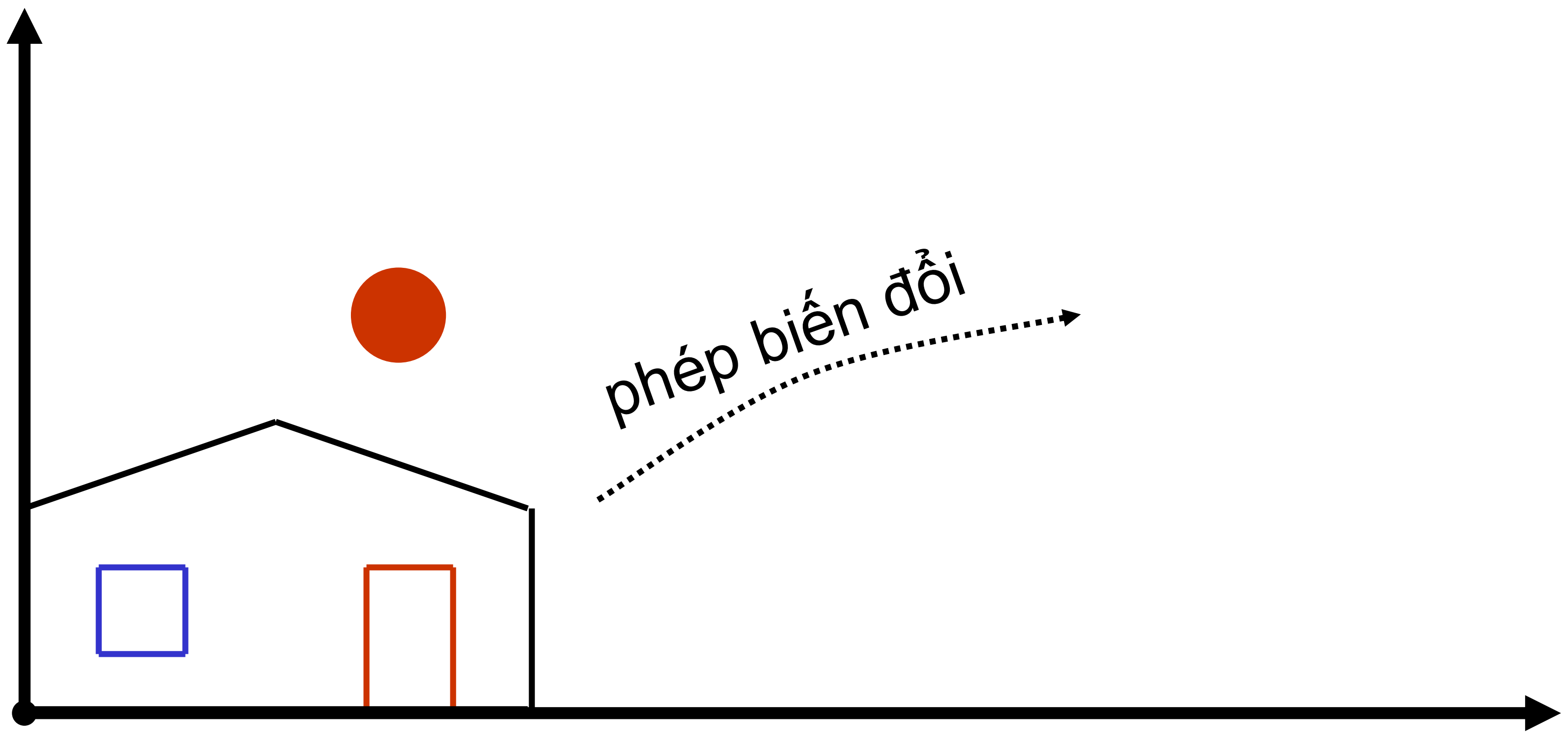
- Đoạn thẳng biến thành đoạn thẳng.
- Đa giác biến thành đa giác.
- Đường cong Bezier biến thành đường cong Bezier.



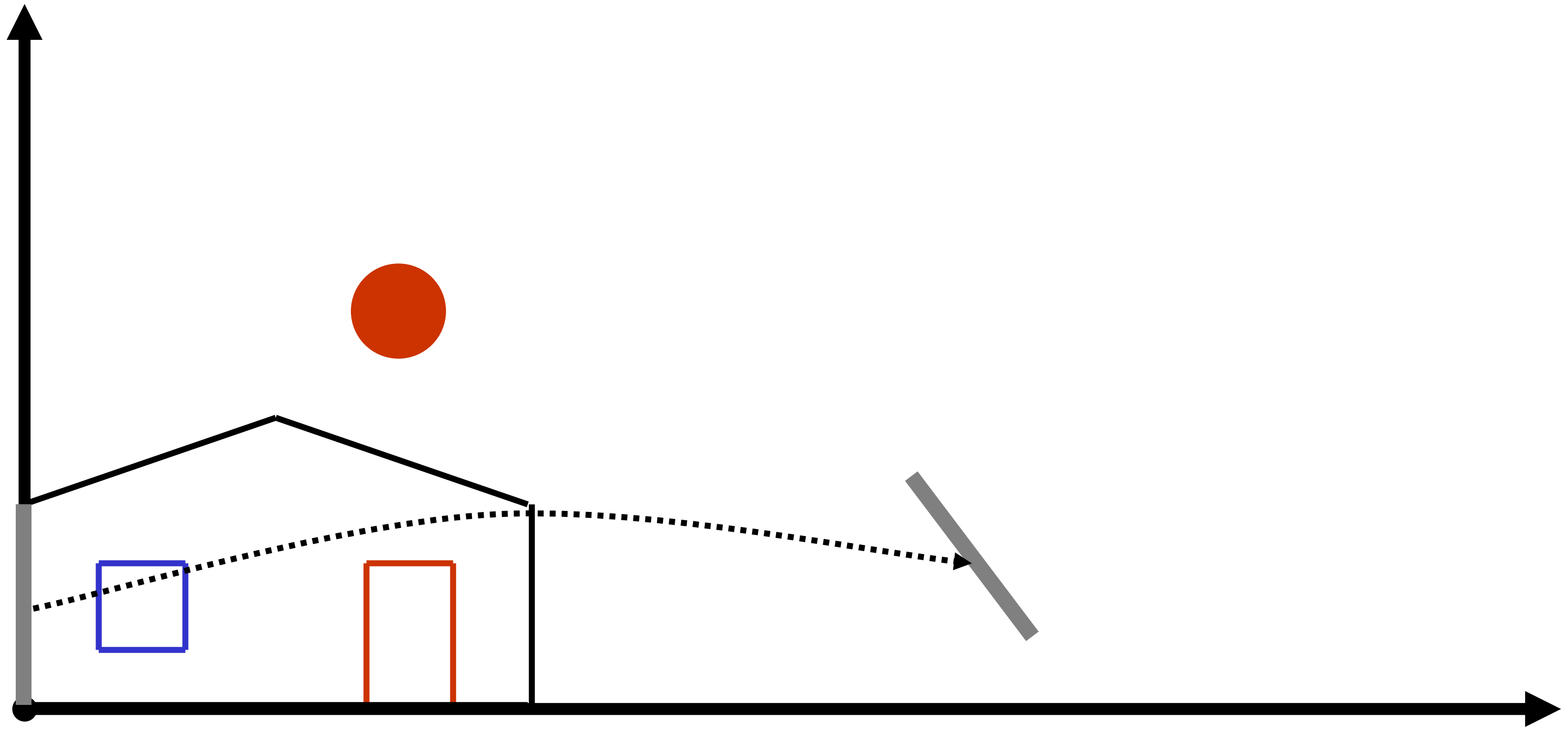




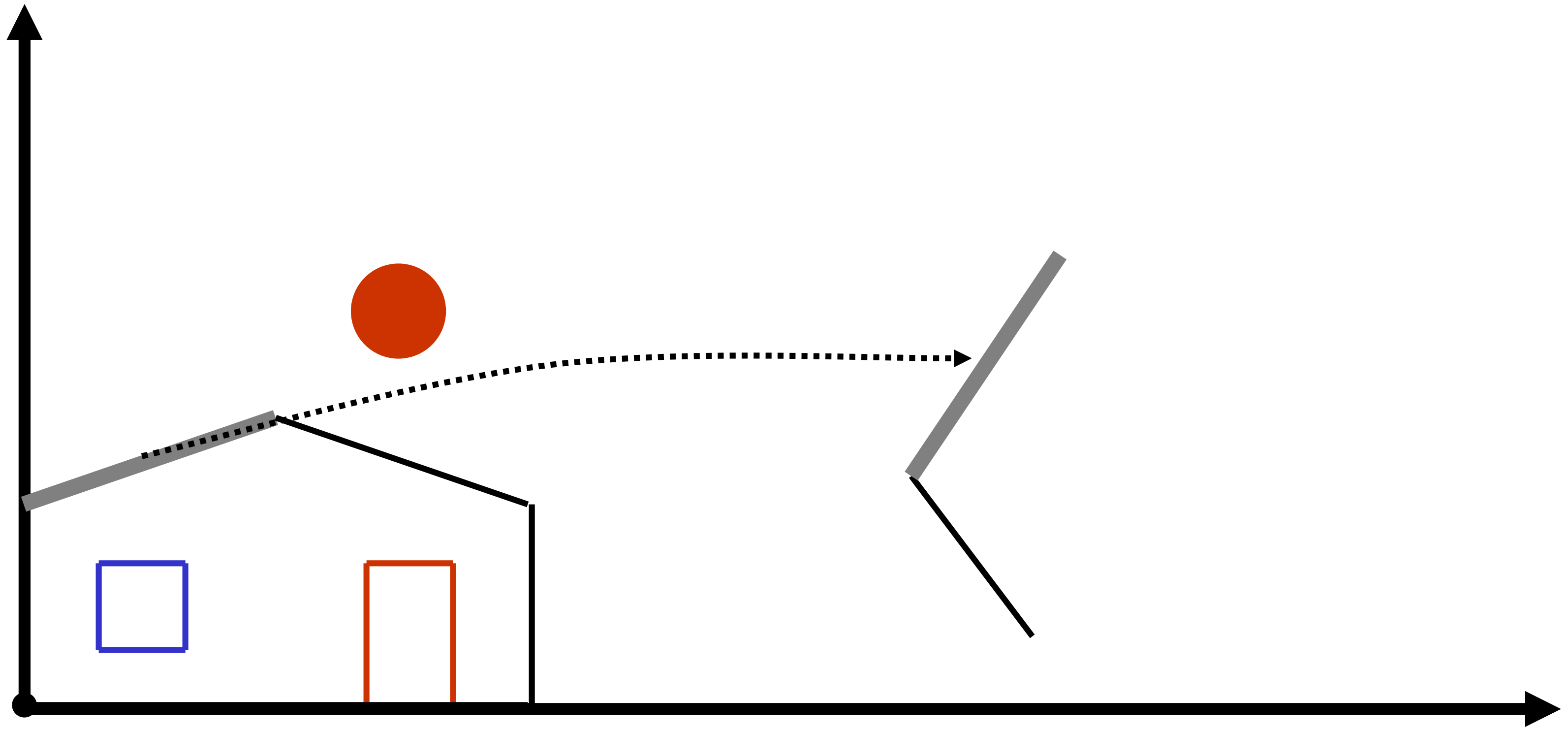
Biến đổi đối tượng



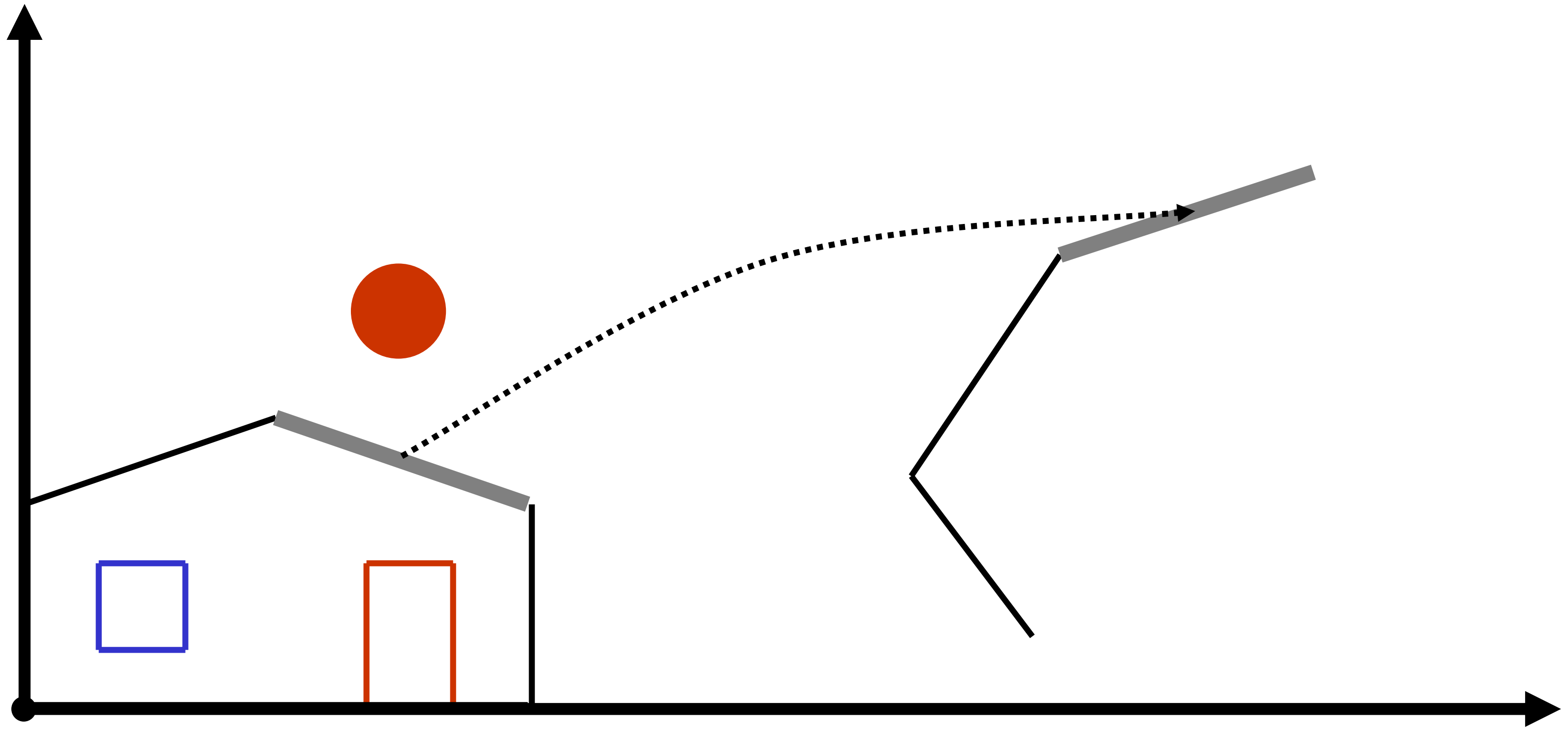
Biến đổi đối tượng



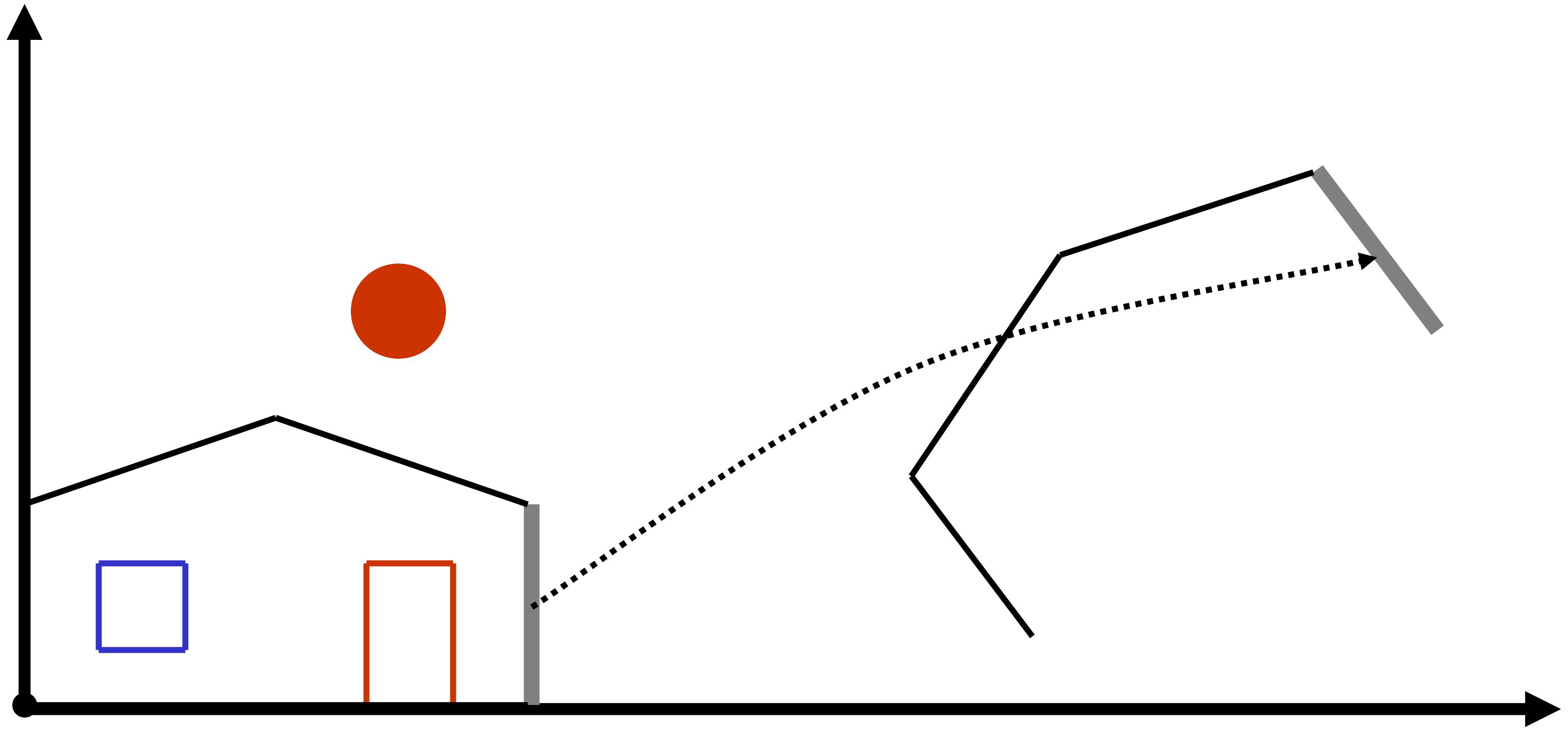
Biến đổi đối tượng



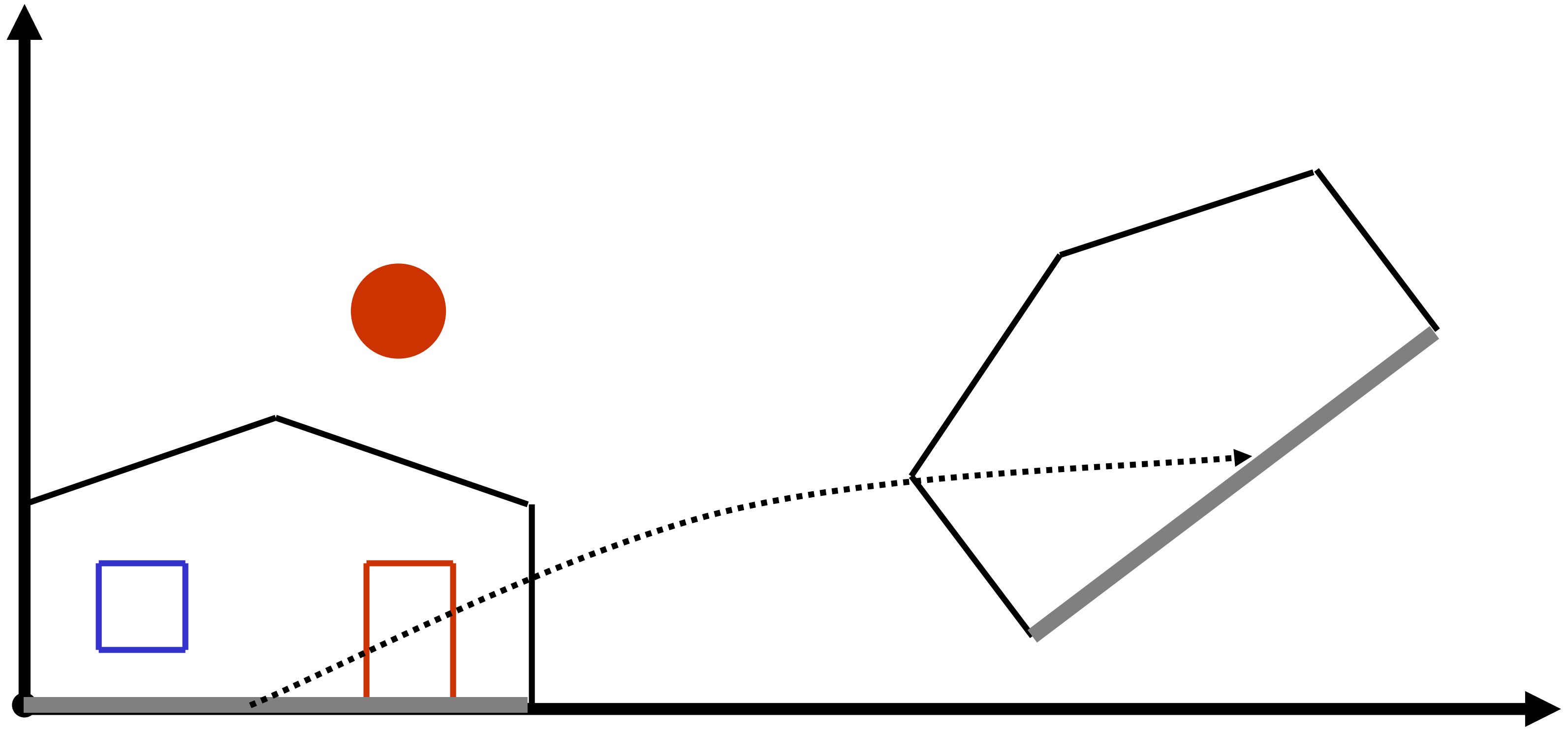
Biến đổi đối tượng



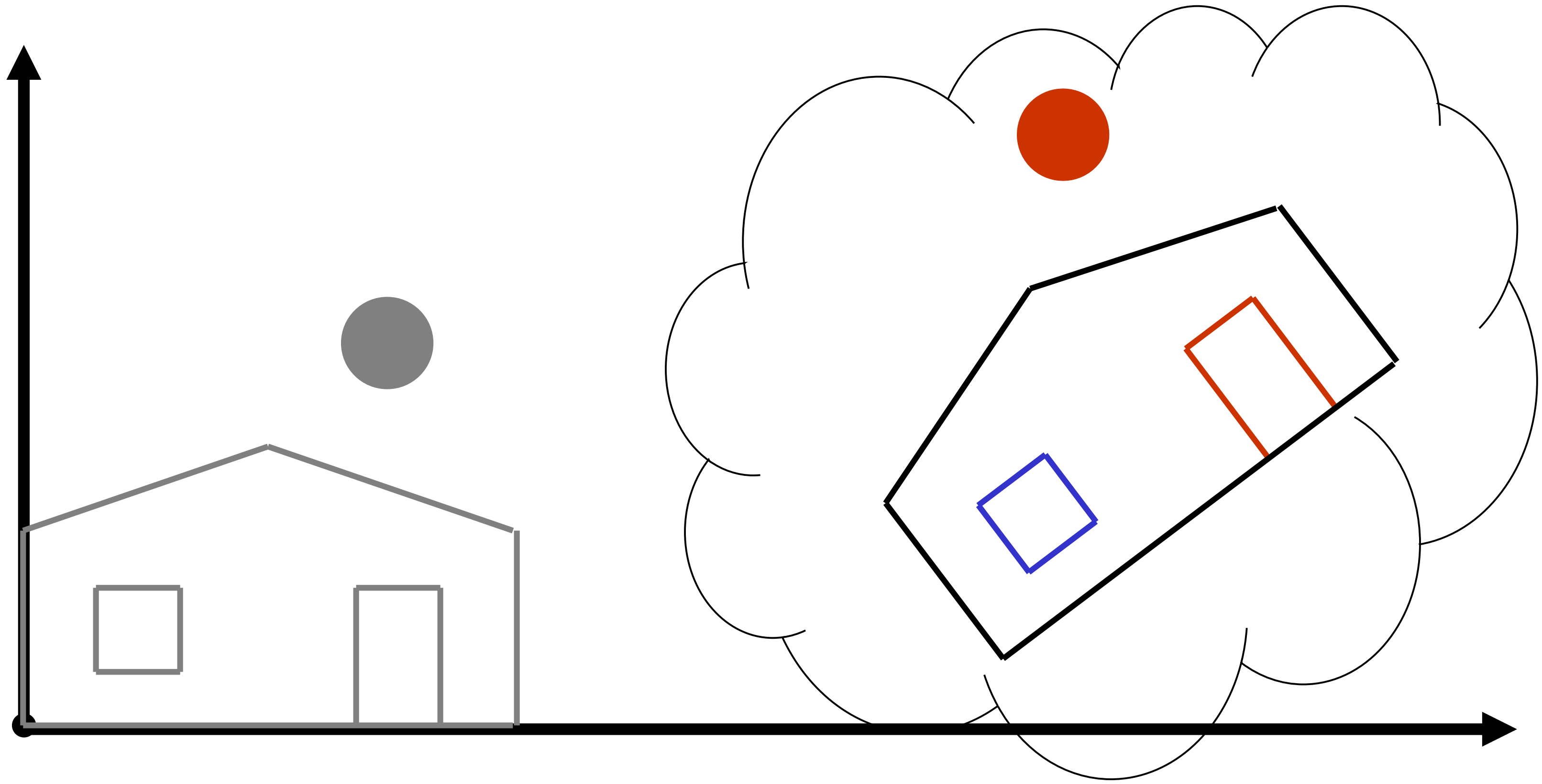
Biến đổi đối tượng



Biến đổi đối tượng



Biến đổi đối tượng



Các phép biến đổi affine cơ sở

Có 3 phép biến đổi affine cơ sở

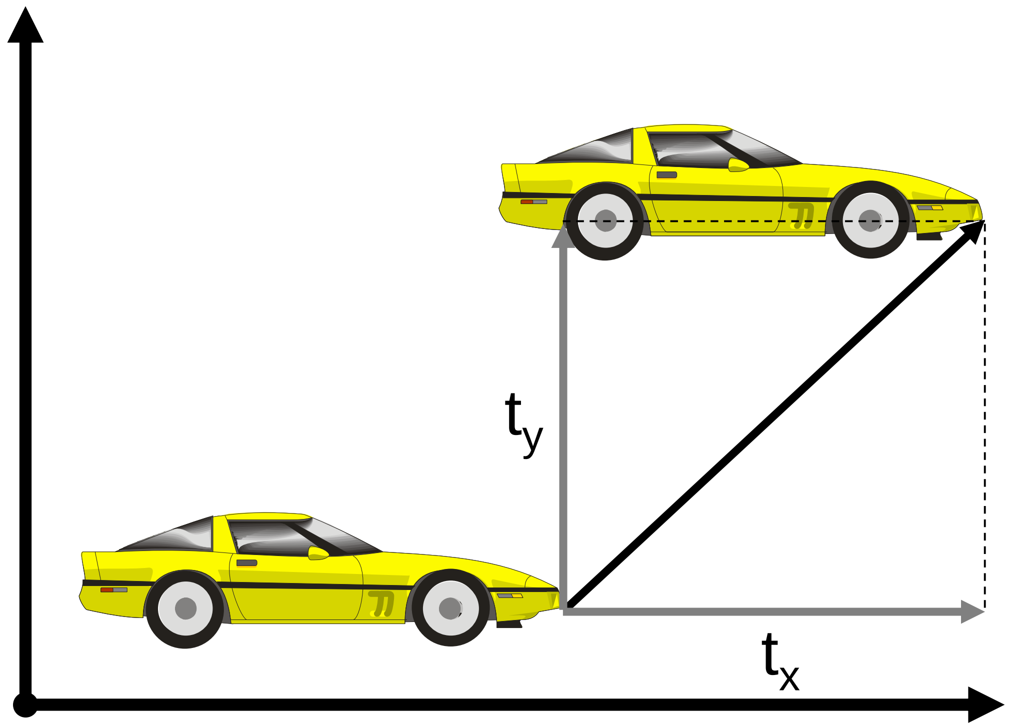
- Phép tịnh tiến
- Phép quay
- Phép tỉ lệ

Phép tịnh tiến (translation)

Dùng để thay đổi vị trí của đối tượng từ vị trí này sáng vị trí khác.

Tham số

- Độ dịch chuyển trên trục Ox : t_x
- Độ dịch chuyển trên trục Oy : t_y



Công thức

Dạng hàm

$$\begin{cases} P'_x = P_x + t_x \\ P'_y = P_y + t_y \end{cases}$$

Dạng ma trận

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix}$$

Cài đặt

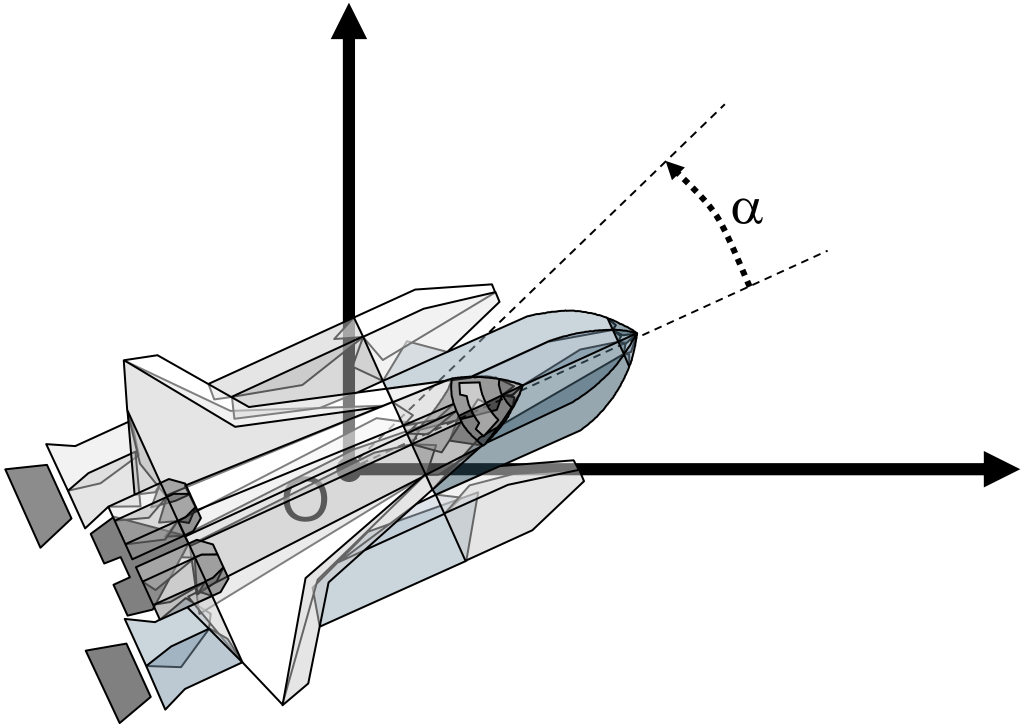
```
TAffine2D BuildTranslation2D(double tx, double ty)
{
        TAffine2D T;

        T.M[0][0]=1;  T.M[0][1]=0;  T.M[0][2]=0;
        T.M[1][0]=0;  T.M[1][1]=1;  T.M[1][2]=0;
        T.M[2][0]=tx; T.M[2][1]=ty; T.M[2][2]=1;

        return T;
}
```

Phép quay (rotation)

Dùng để thay đổi hướng của đối tượng

Tham số

- Tâm quay : O
- Góc quay : α



Công thức

Dạng hàm

$$\begin{cases} P'_x = \cos\alpha P_x - \sin\alpha P_y \\ P'_y = \sin\alpha P_x + \cos\alpha P_y \end{cases}$$

Dạng ma trận

$$M = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Cài đặt

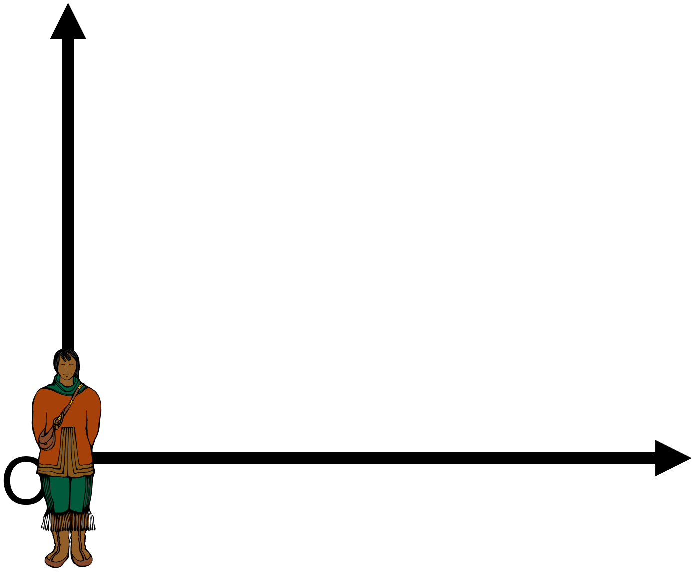
```
TAffine2D BuildRotation2D(double alpha)
{
        TAffine2D T;

        alpha *= PI/180;
        T.M[0][0]=cos(alpha);  T.M[0][1]=sin(alpha); T.M[0][2]=0;
        T.M[1][0]=-sin(alpha); T.M[1][1]=cos(alpha); T.M[1][2]=0;
        T.M[2][0]=0;           T.M[2][1]=0;          T.M[2][2]=1;

        return T;
}
```

Phép tỉ lệ (scaling)

Dùng để thay đổi kích thước của đối tượng

Tham số

- Tâm tỉ lệ : O
- Hệ số tỉ lệ : s_x, s_y



Công thức

Dạng hàm

$$\begin{cases} P'_x = s_x P_x \\ P'_y = s_y P_y \end{cases}$$

Dạng ma trận

$$M = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Cài đặt

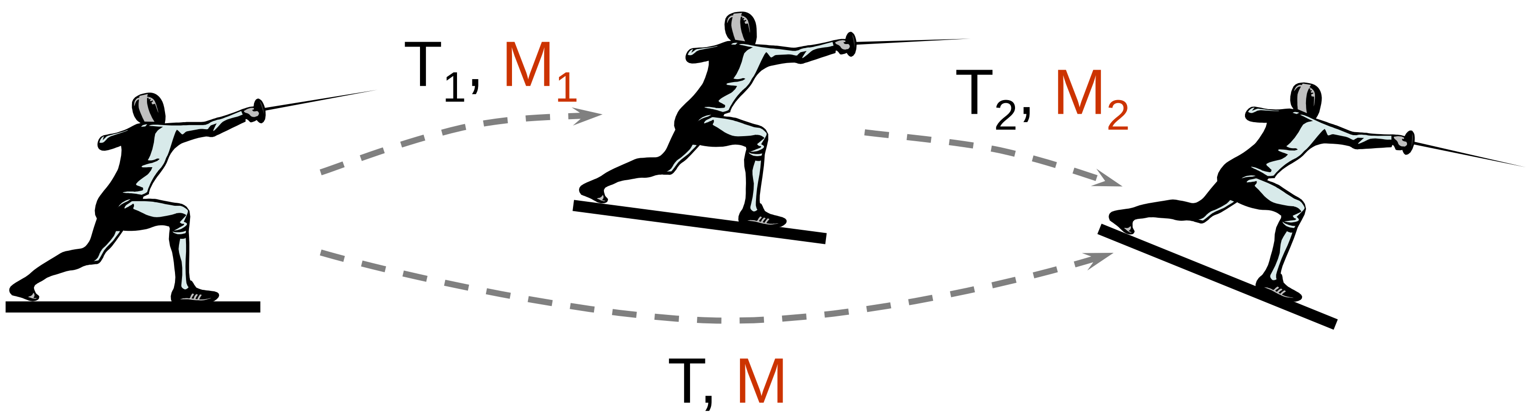
```
TAffine2D BuildScaling2D(double sx, double sy)
{
        TAffine2D T;

        T.M[0][0]=sx; T.M[0][1]=0;  T.M[0][2]=0;
        T.M[1][0]=0;  T.M[1][1]=sy; T.M[1][2]=0;
        T.M[2][0]=0;  T.M[2][1]=0;  T.M[2][2]=1;

        return T;
}
```

Nguyên lý kết hợp các phép biến đổi

Nếu T_1, T_2 là Phép biến đổi affine

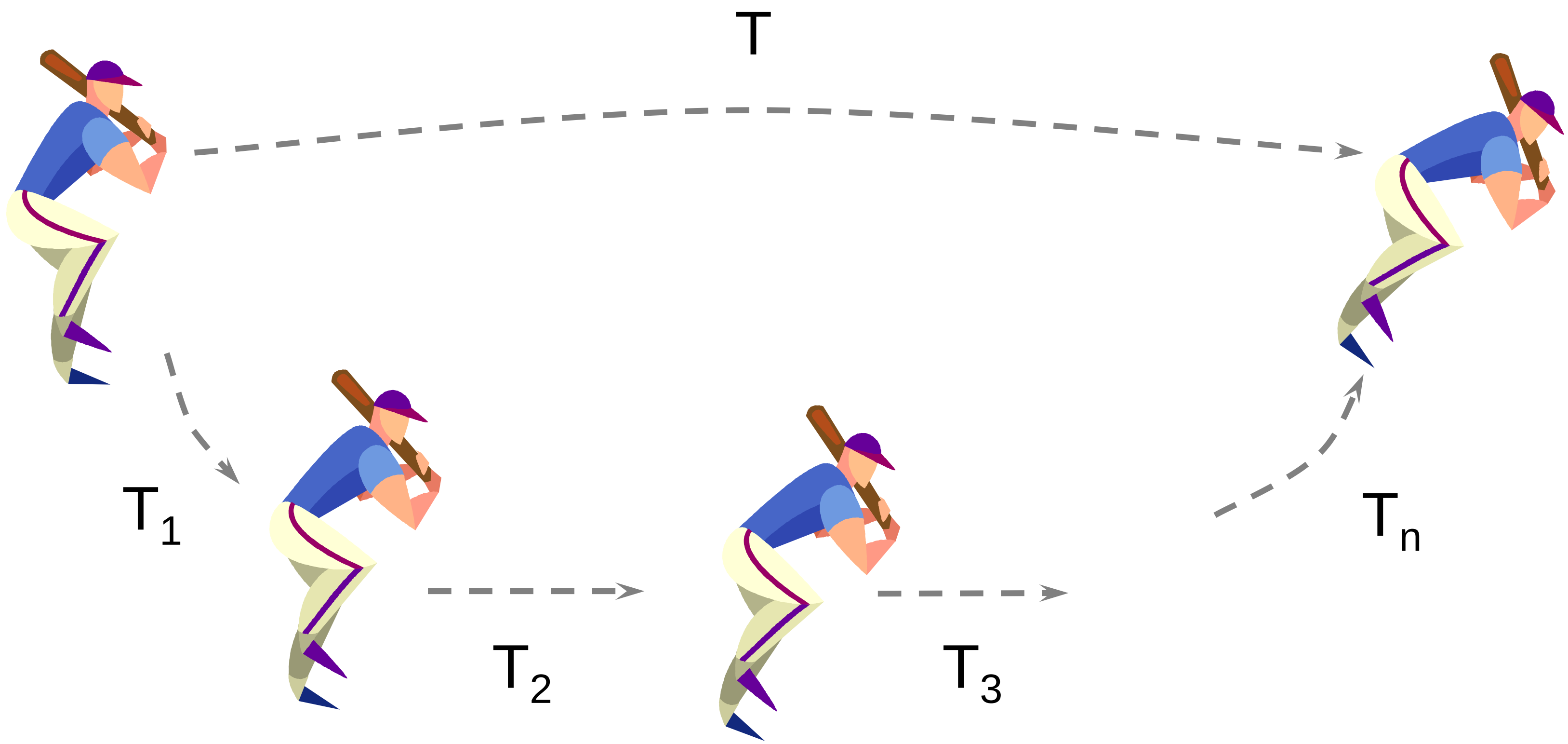
Thì

- $T = T_1 + T_2$ là Phép biến đổi affine
- $M = M_1 \times M_2$



Nguyên lý phân rã phép biến đổi

Mọi phép biến đổi affine đều có thể phân rã thành một chuỗi các phép biến đổi cơ sở

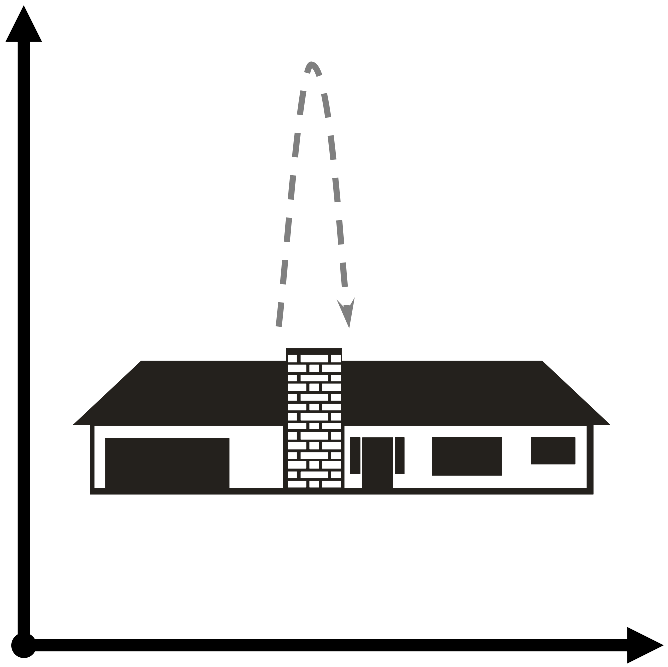


Cài đặt nguyên lý kết hợp

```
TAffine2D operator + (TAffine2D T1, TAffine2D T2)
{
        TAffine2D T;

        for(int i=0; i<3; i++)
                for(int j=0; j<3; j++)
                {
                        T.M[i][j] =
                                T1.M[i][0]*T2.M[0][j] +
                                T1.M[i][1]*T2.M[1][j] +
                                T1.M[i][2]*T2.M[2][j];
                }

        return T;
}
```

Phép đồng nhất (Identity)

Biến "nó" thành chính "nó"



Dạng hàm

$$\begin{cases} P'_x = P_x \\ P'_y = P_y \end{cases}$$

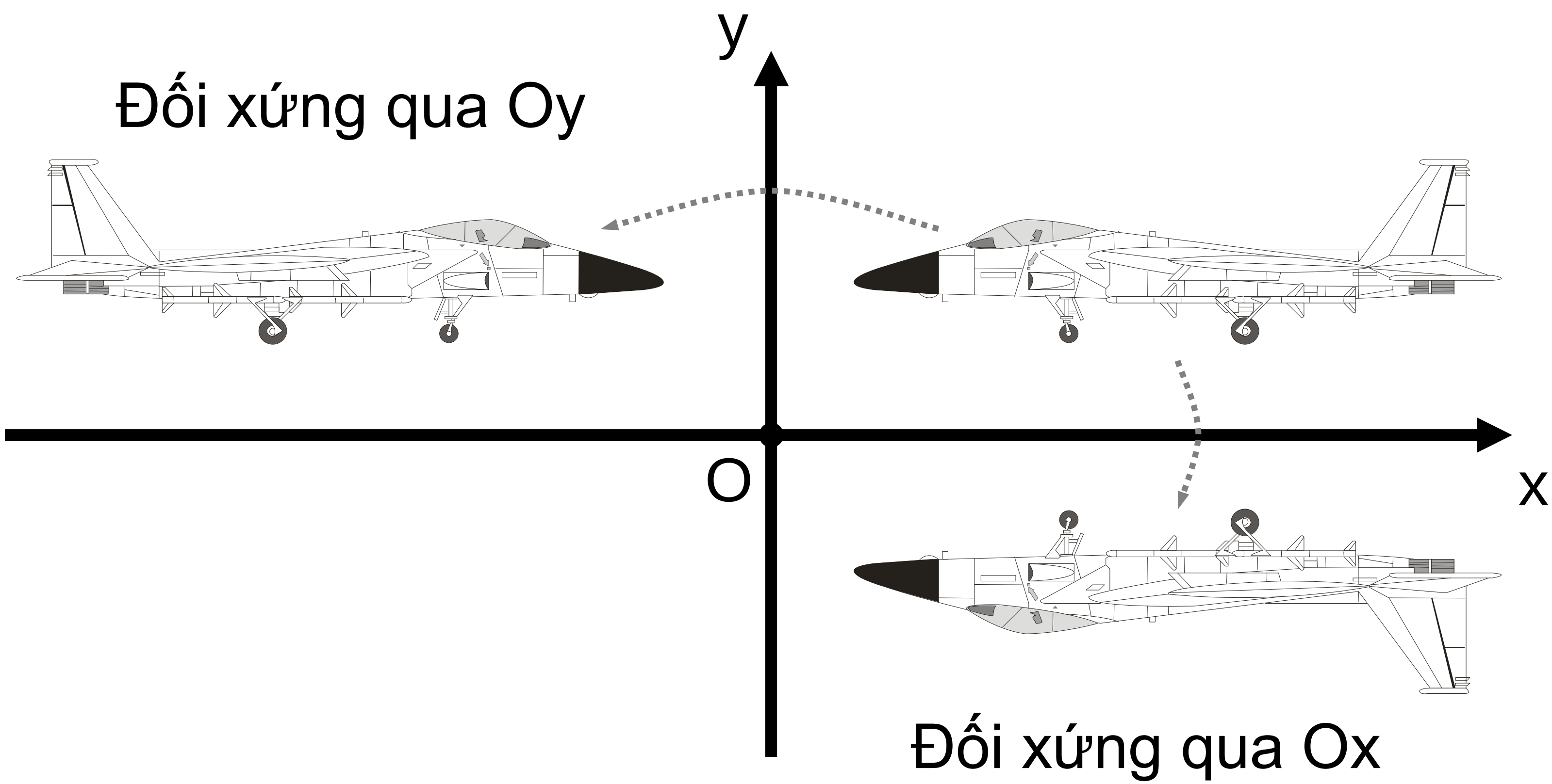
Dạng ma trận

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Cài đặt

```
TAffine2D BuildIdentity2D()
{
        TAffine2D T;

        T.M[0][0] = 1; T.M[0][1] = 0; T.M[0][2] = 0;
        T.M[1][0] = 0; T.M[1][1] = 1; T.M[1][2] = 0;
        T.M[2][0] = 0; T.M[2][1] = 0; T.M[2][2] = 1;

        return T;
}
```

Phép đối xứng (reflection)



Công thức

Đối xứng qua trục Ox

Dạng hàm

$$\begin{cases} P_x^{'} = P_x \\ P_y^{'} = -P_y \end{cases}$$

Dạng ma trận

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

Đối xứng trục Oy

Dạng hàm

$$\begin{cases} P_x^{'} = -P_x \\ P_y^{'} = P_y \end{cases}$$

Dạng ma trận

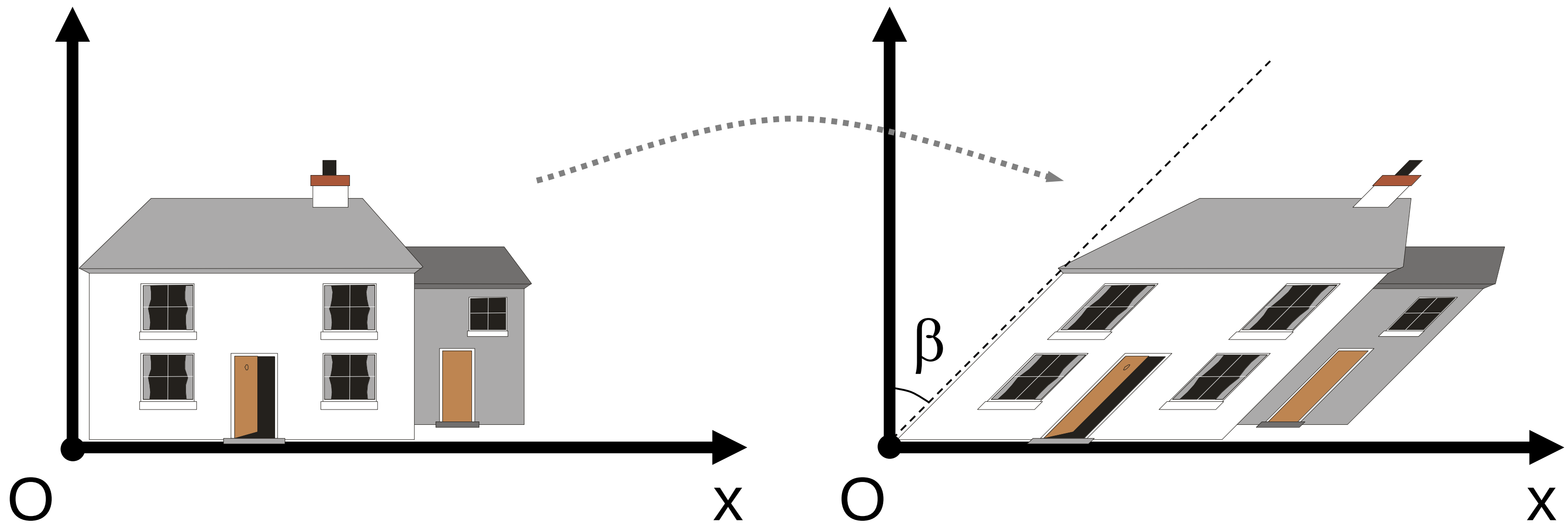
$$M = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Phép biến dạng (shearing)

Dùng để làm nghiêng đối tượng theo một trục nào đó.

Tham số

- Trục nghiêng : Ox
- Góc nghiêng : β



Công thức

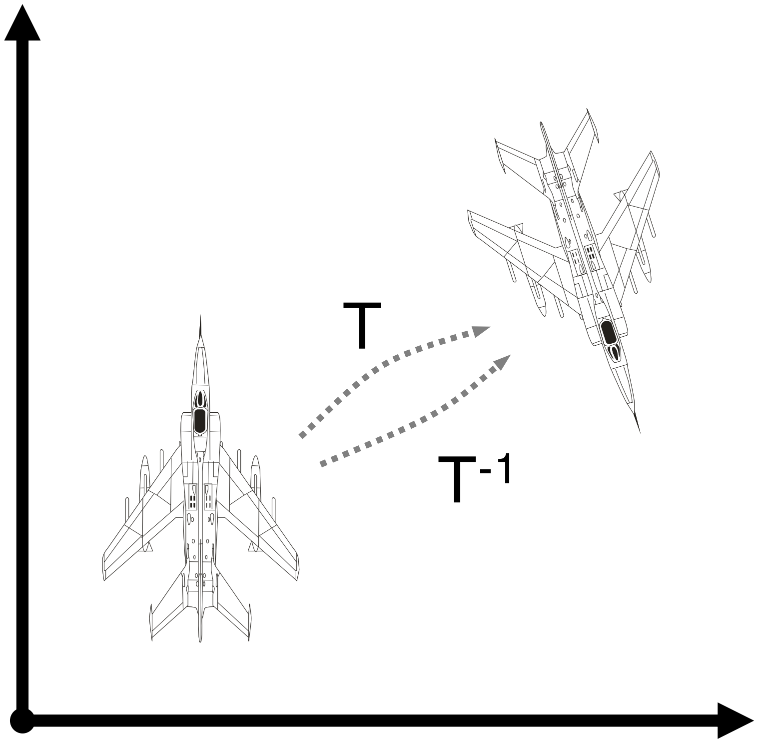
Dạng hàm

$$\begin{cases} P_x^{'} = P_x + tg\beta P_y \\ P_y^{'} = P_y \end{cases}$$

Dạng ma trận

$$M = \begin{pmatrix} 1 & 0 & 0 \\ tg\beta & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ hoặc } M = \begin{pmatrix} 1 & 0 & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Phép nghịch đảo



Công thức

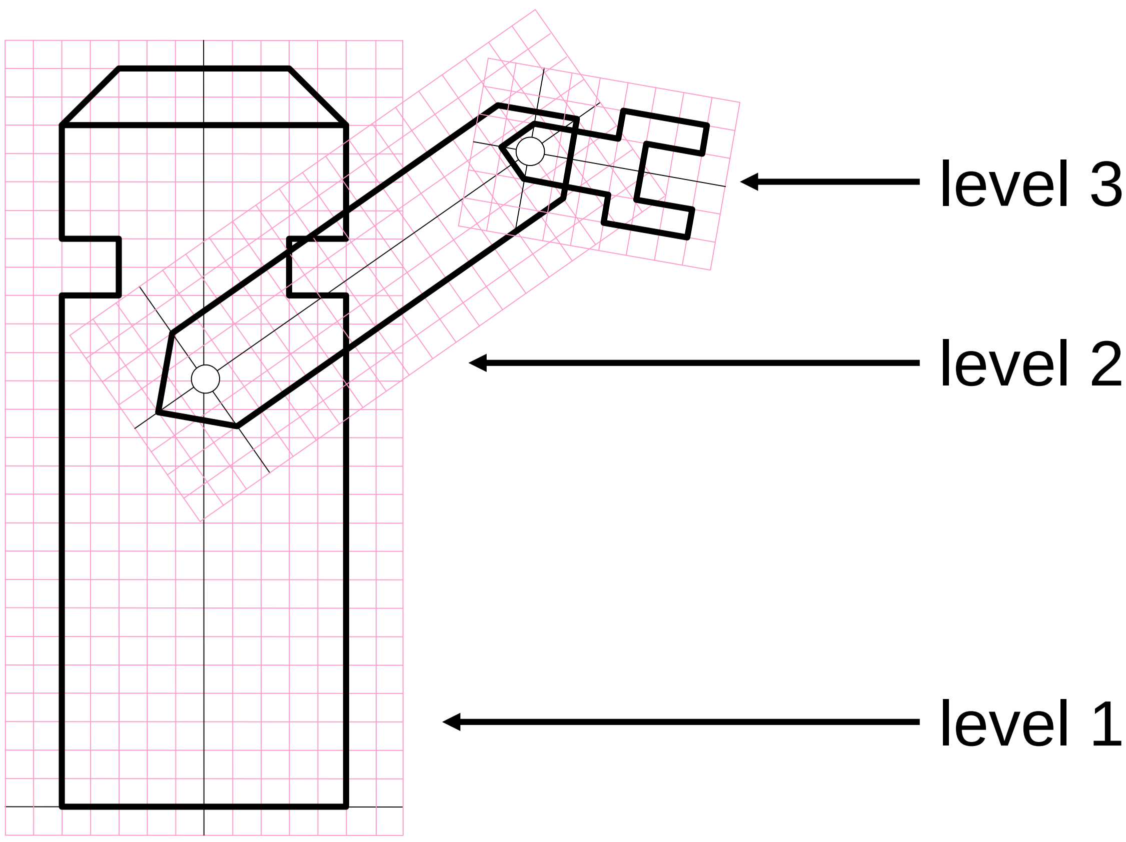
Nếu T có

$$M = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix}, \text{với } ad - bc \neq 0$$

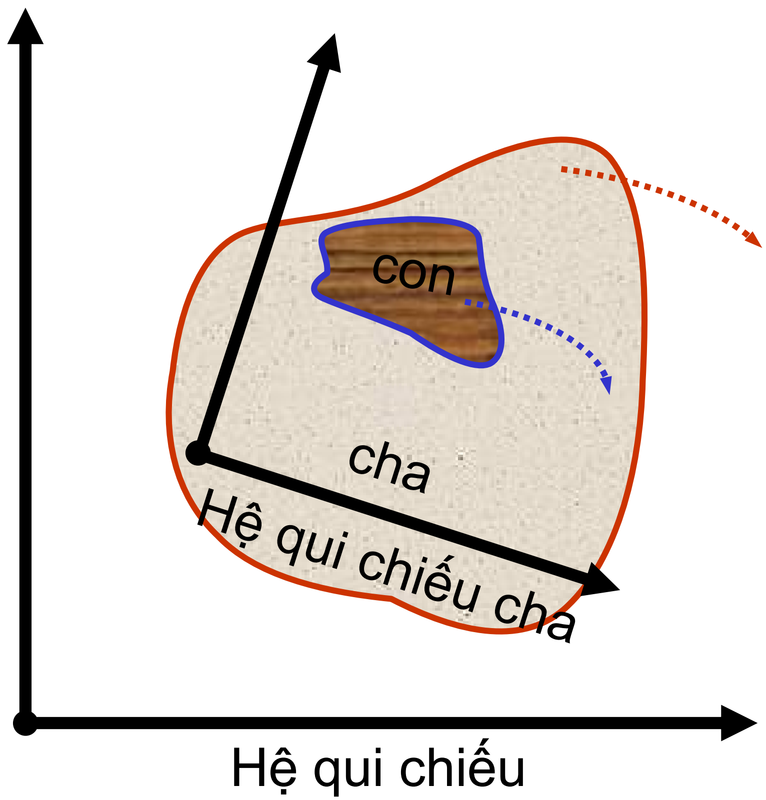
thì T^{-1} có

$$M^{-1} = \begin{pmatrix} \dfrac{d}{ad-bc} & \dfrac{-b}{ad-bc} & 0 \\ \dfrac{-c}{ad-bc} & \dfrac{a}{ad-bc} & 0 \\ \dfrac{cf-de}{ad-bc} & \dfrac{be-af}{ad-bc} & 1 \end{pmatrix}$$

Biến đổi cho đối tượng phân cấp



Công thức



$$T_{con} = T_{con\text{-}cha} + T_{cha}$$