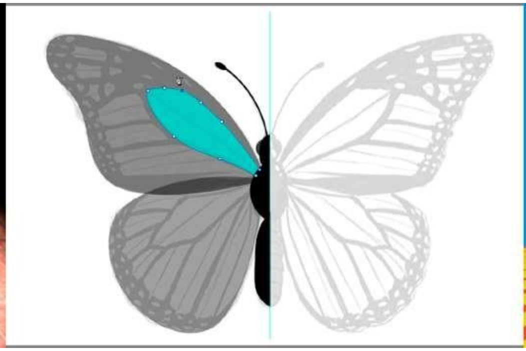
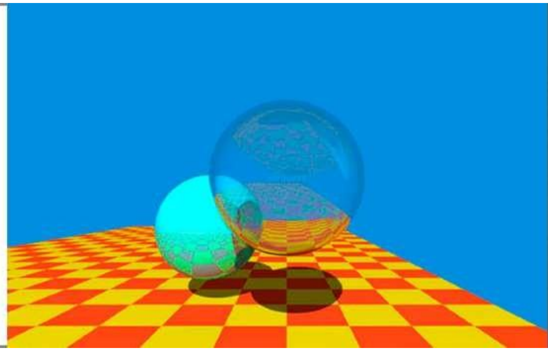
Computer Graphics

Transformation



Last Lecture

- Vectors
  - Basic operations: addition, multiplication
- Dot Product
  - Forward / backward (dot product positive / negative)
- Cross Product
  - Left / right (cross product outward / inward)
- Matrices

多选题 1分

设置

向量加法满足以下哪种定律?

A 交换律

B 结合律

C 分配律

D 三角律

提交

多选题 1分

设置

Dot Product的作用有哪些?

- A 计算向量夹角
- B 计算向量的长度
- C 评估两个向量的相近程度
- D 做向量加法
- E 向量分解
- F 判断forward / backward

提交

多选题 1分

设置

对于Cross Product，下来哪些公式或说法是正确的?

A $\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$

B $\vec{a} \times \vec{a} = \vec{1}$

C $\vec{a} \times (k\vec{b}) = k(\vec{a} \times \vec{b})$

D 若两向量相互垂直，$\vec{a} \times \vec{b} = \vec{0}$

E Cross Product满足左手法则

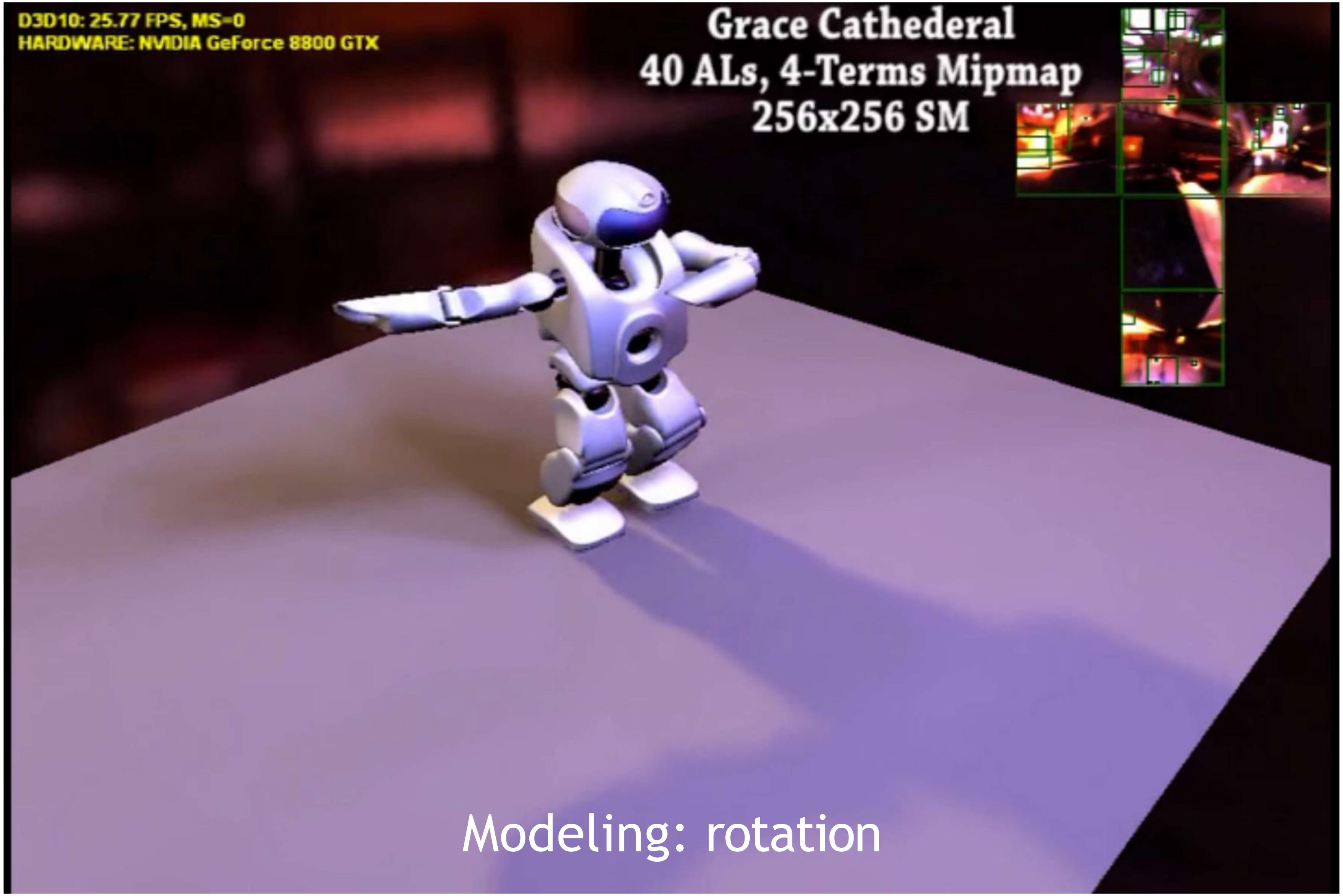
提交

Outline

- Why study transformation
  - Modeling
  - Viewing
- 2D transformations
- Homogeneous coordinates

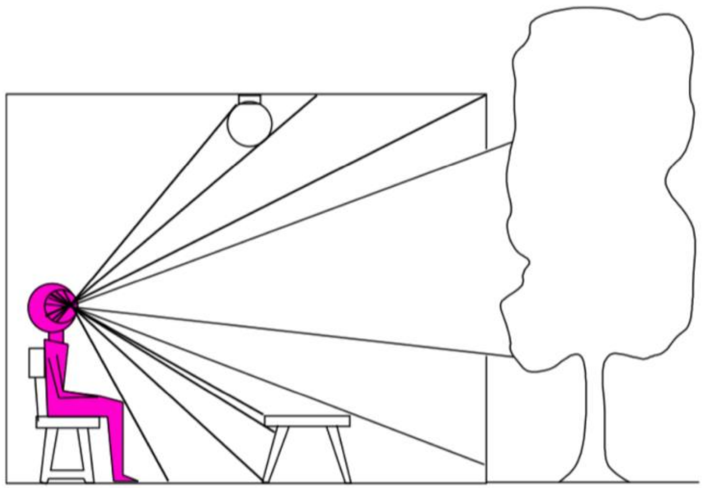
Why Transformation?

Why Transformation?



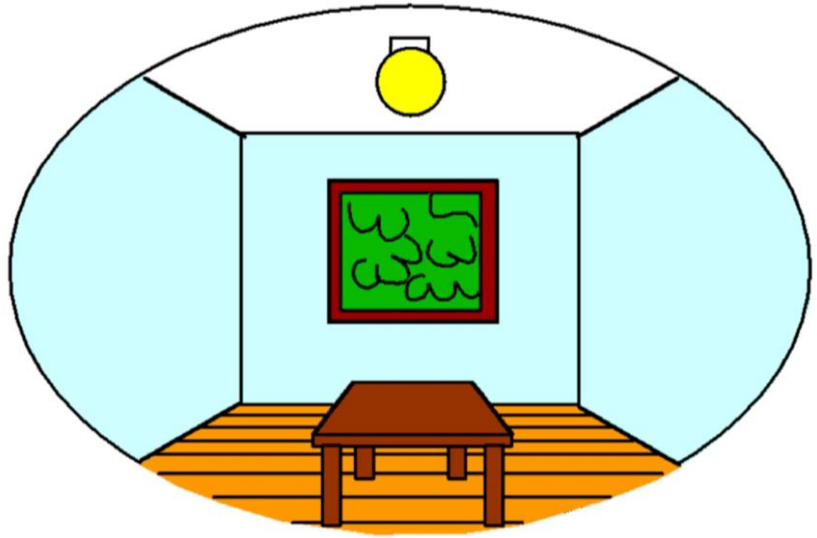
Why Transformation?

3D world



Point of observation

2D image



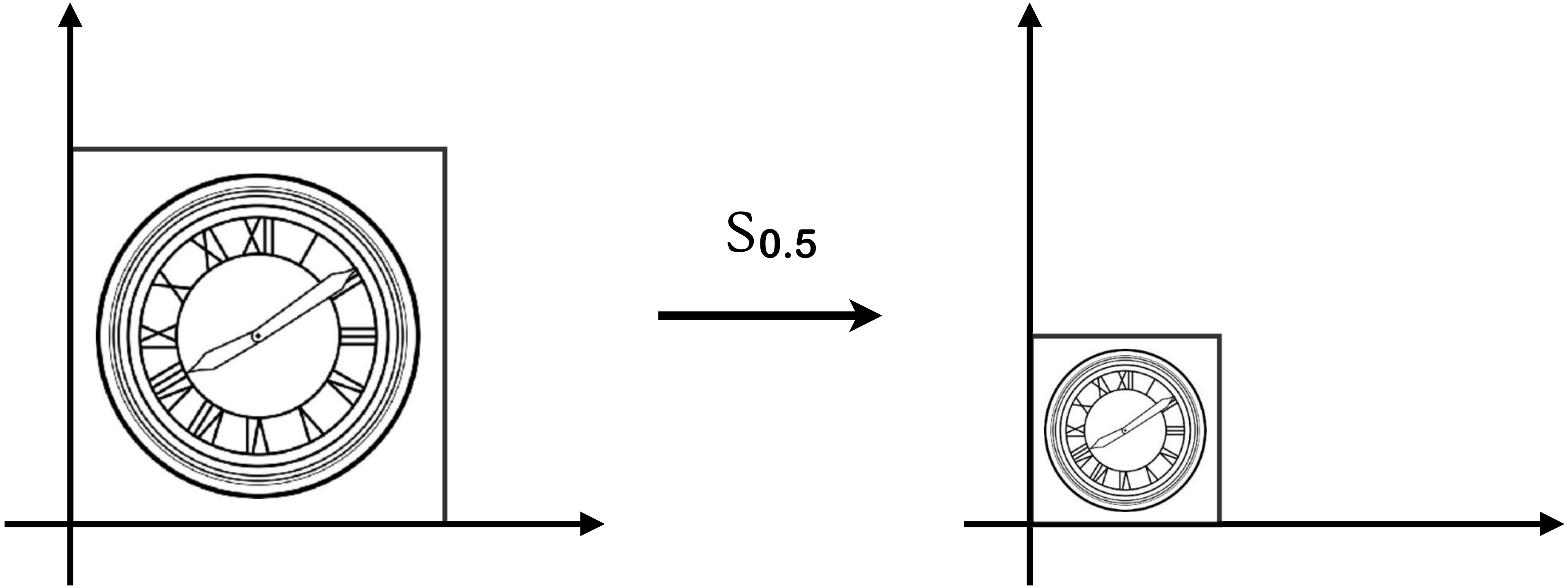
Figures © Stephen E. Palmer, 2002

Viewing: (3D to 2D) projection

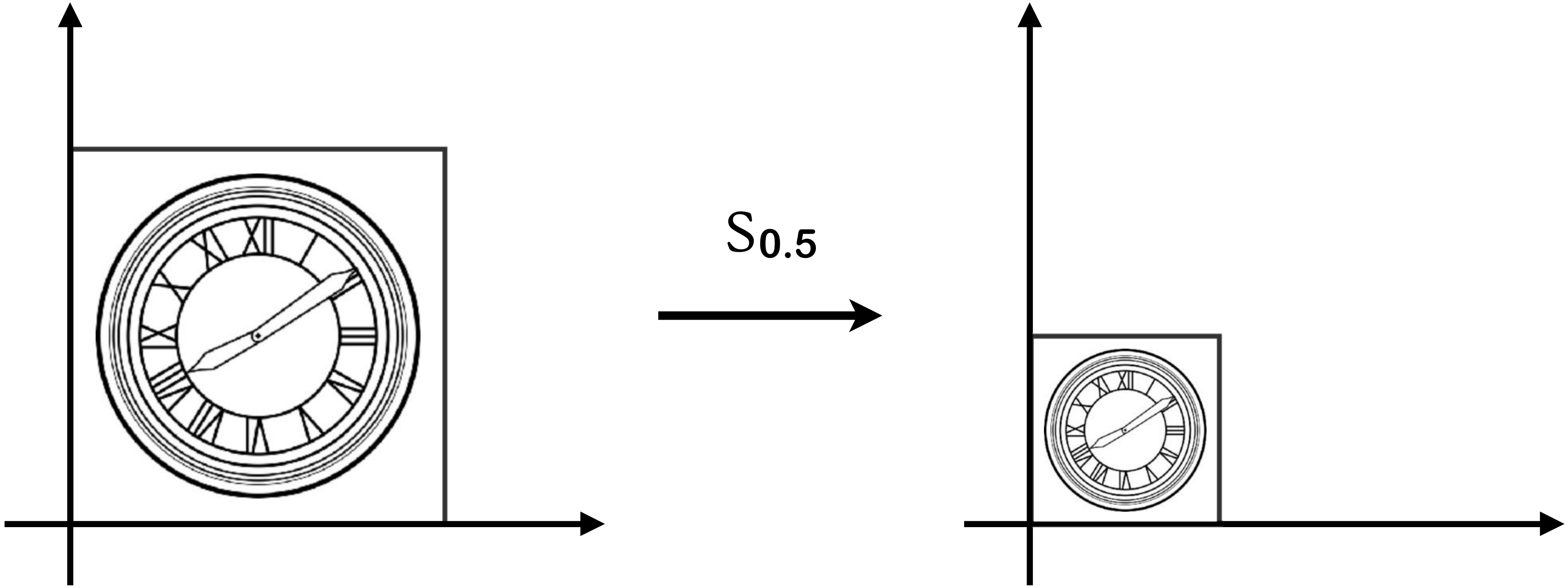
Outline

- Why study transformation
- 2D transformations
  - Representing transformations using matrices
  - Rotation, scale, shear
- Homogeneous coordinates

Scale



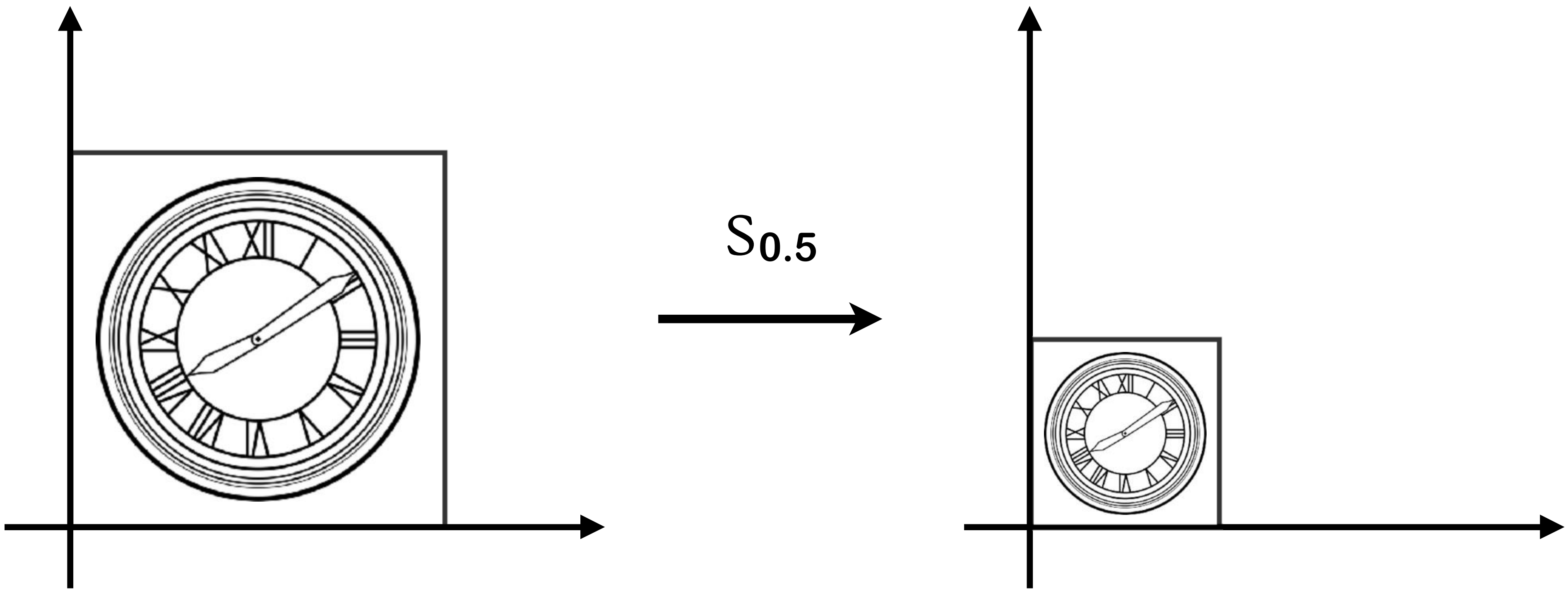
Scale Transform



$$x' = sx$$

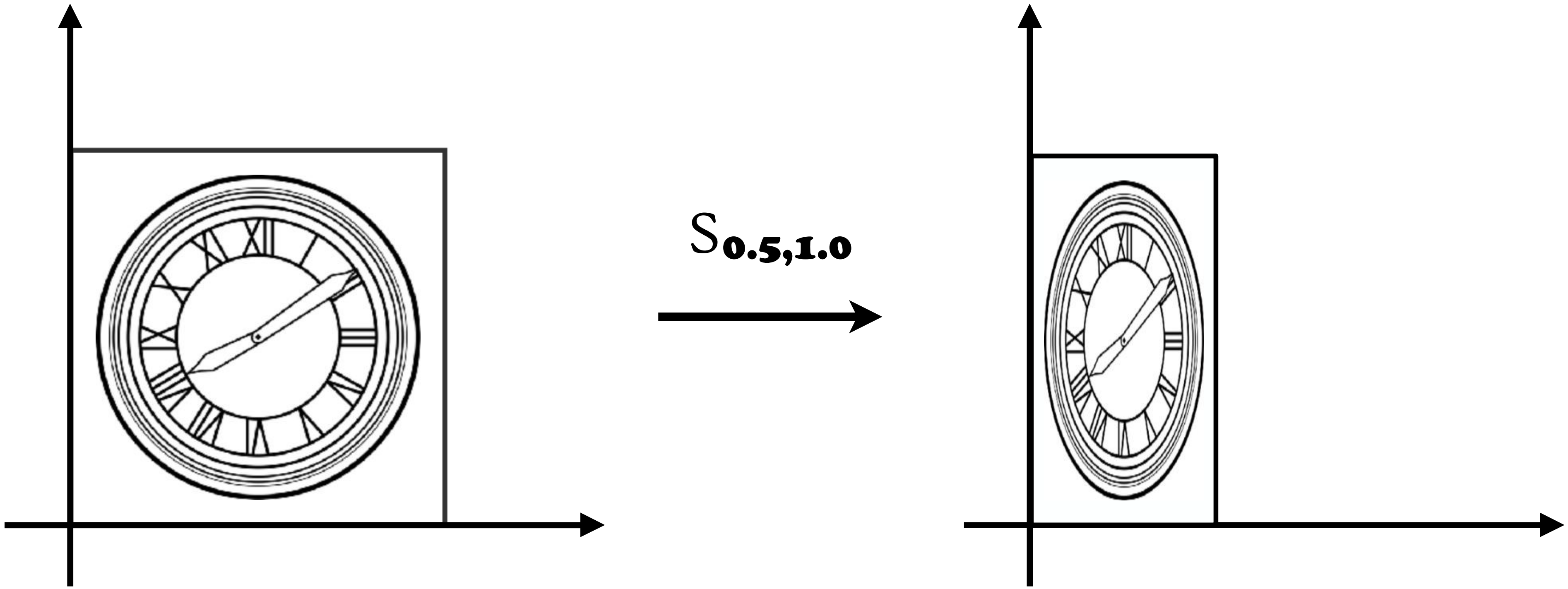
$$y' = sy$$

Scale Matrix



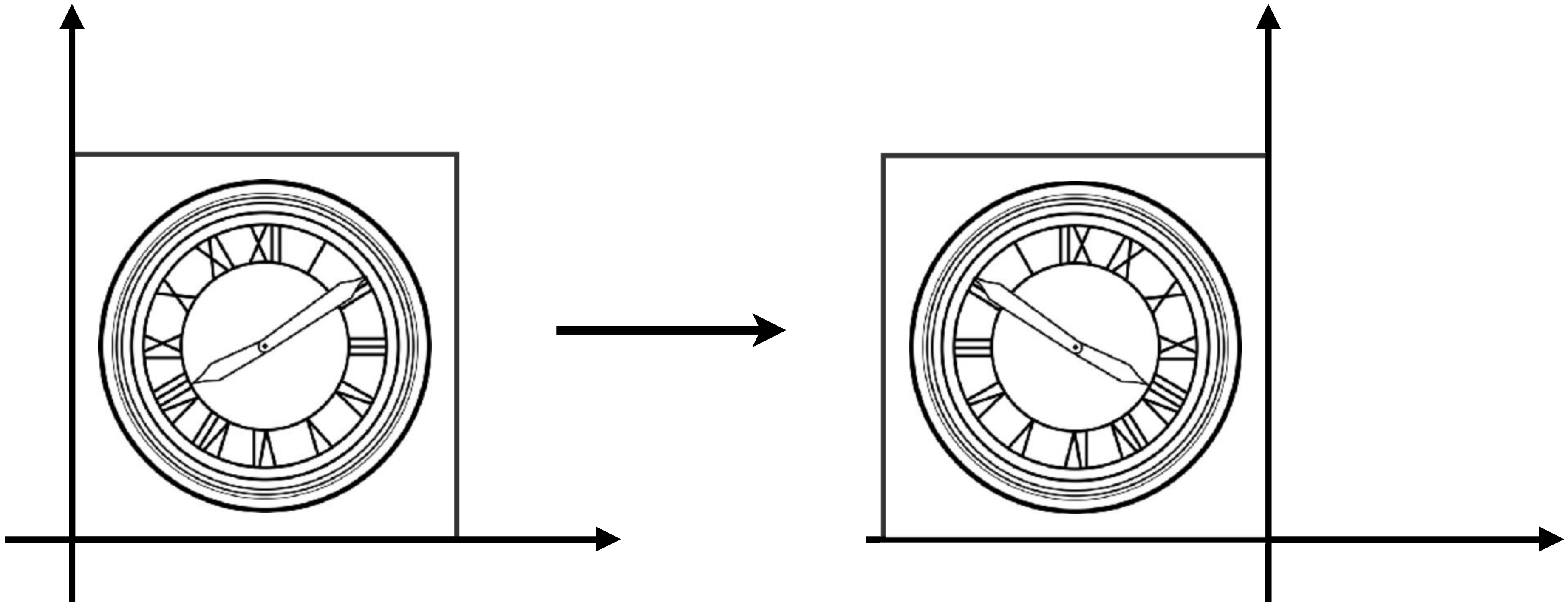
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale (Non-Uniform)



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Reflection Matrix

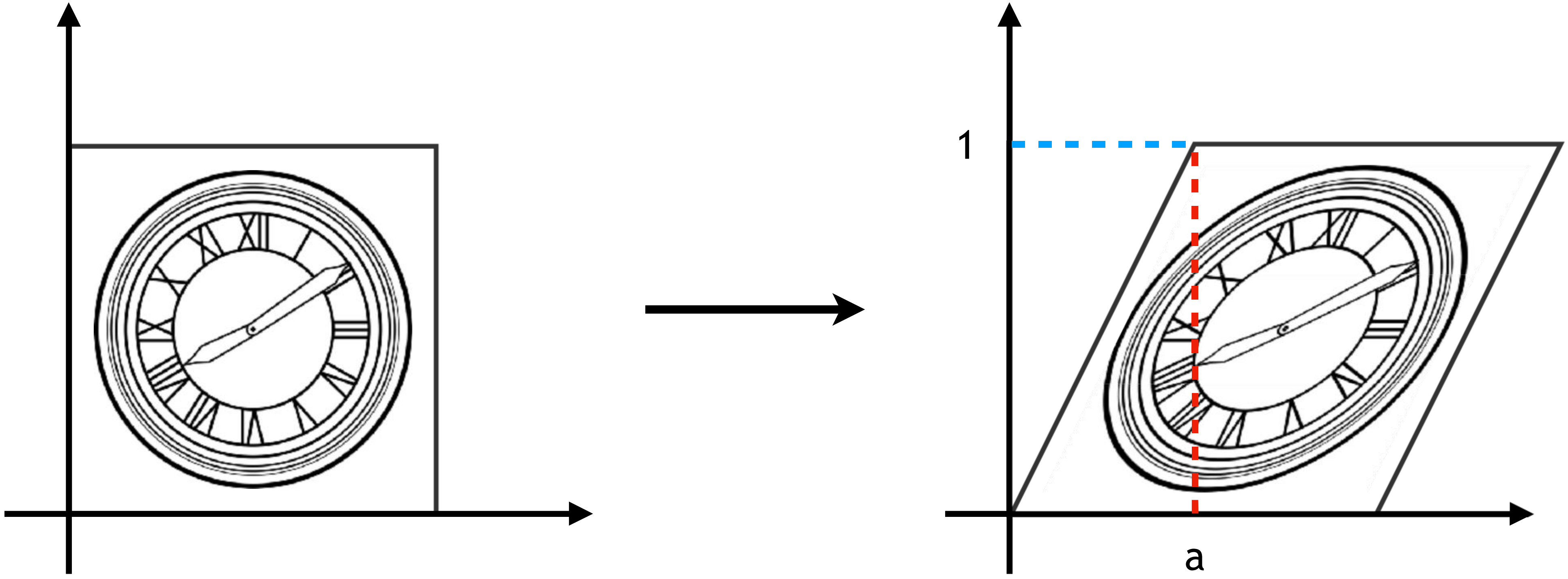


Horizontal reflection:

$$x' = -x$$
$$y' = y$$

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

Shear Matrix



Hints:

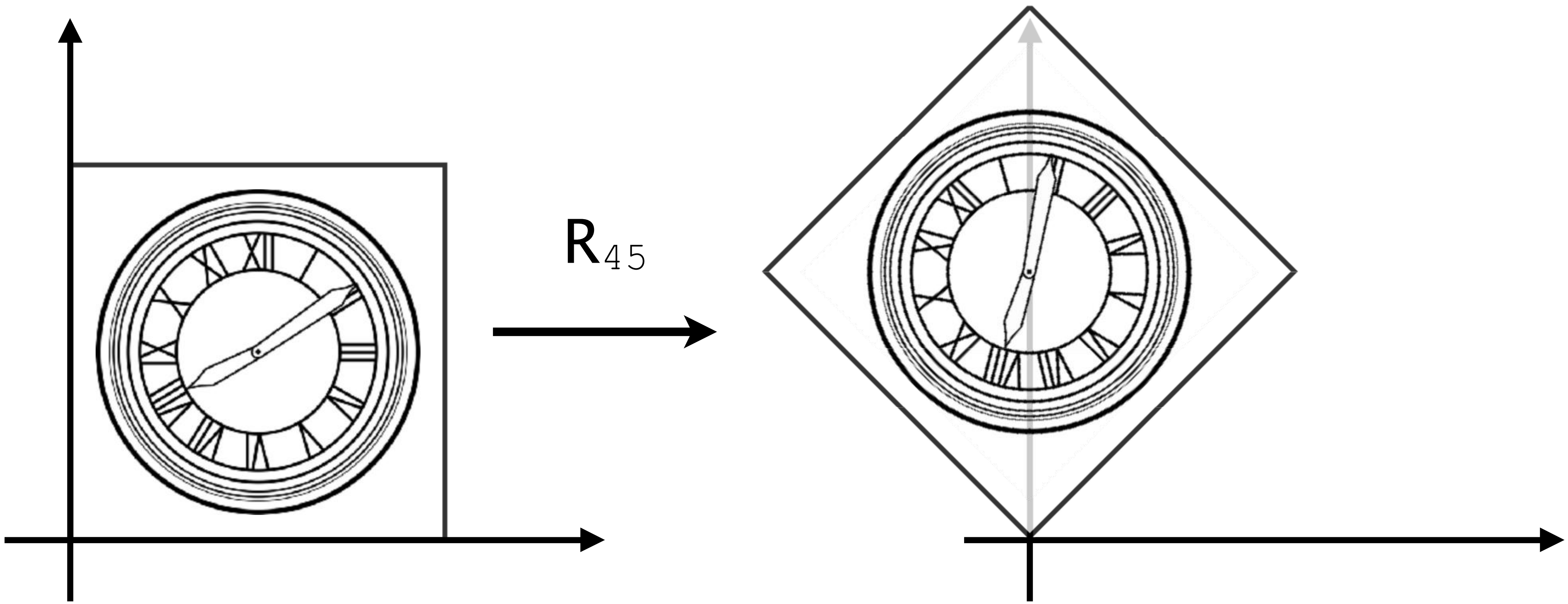
Horizontal shift is 0 at y=0

Horizontal shift is a at y=1

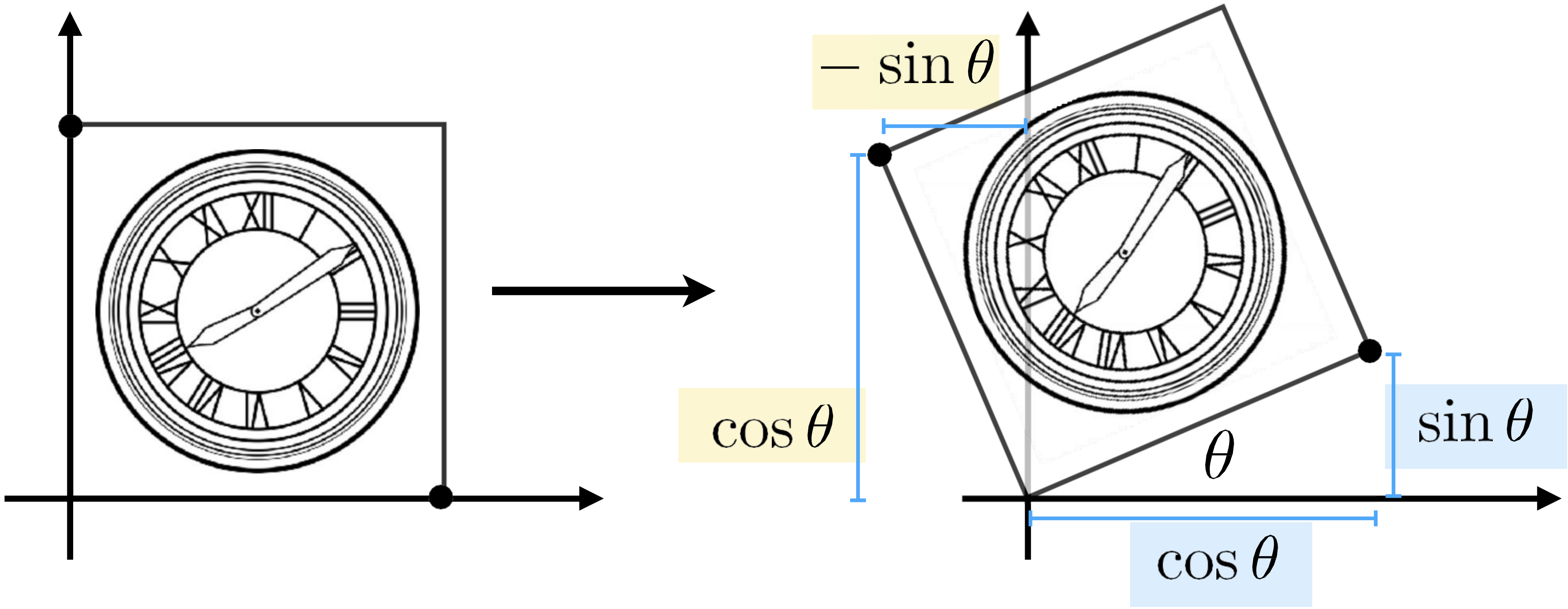
Vertical shift is always 0

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

Rotate (about the origin (0, 0), CCW by default)



Rotation Matrix



$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Linear Transforms （线性变换） = Matrices

(of the same dimension)

$$x' = a\,x + b\,y$$
$$y' = c\,x + d\,y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

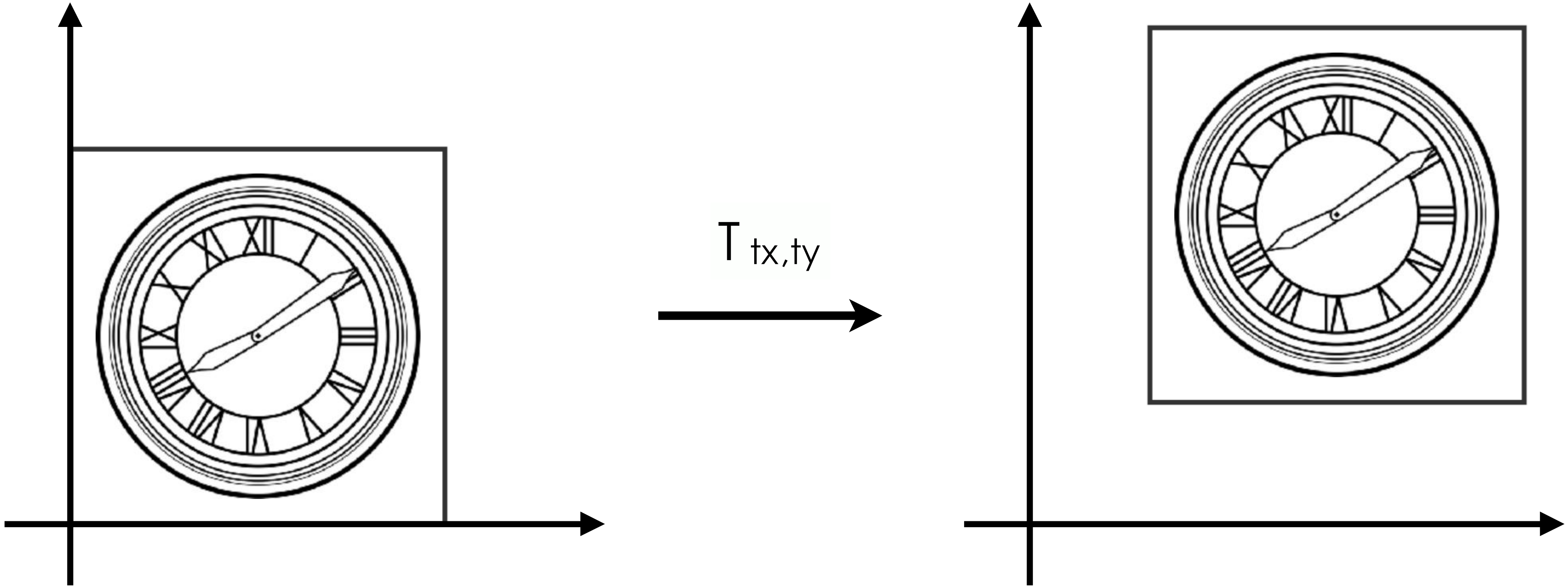
$$\mathbf{x}' = \mathbf{M}\,\mathbf{x}$$

Questions?

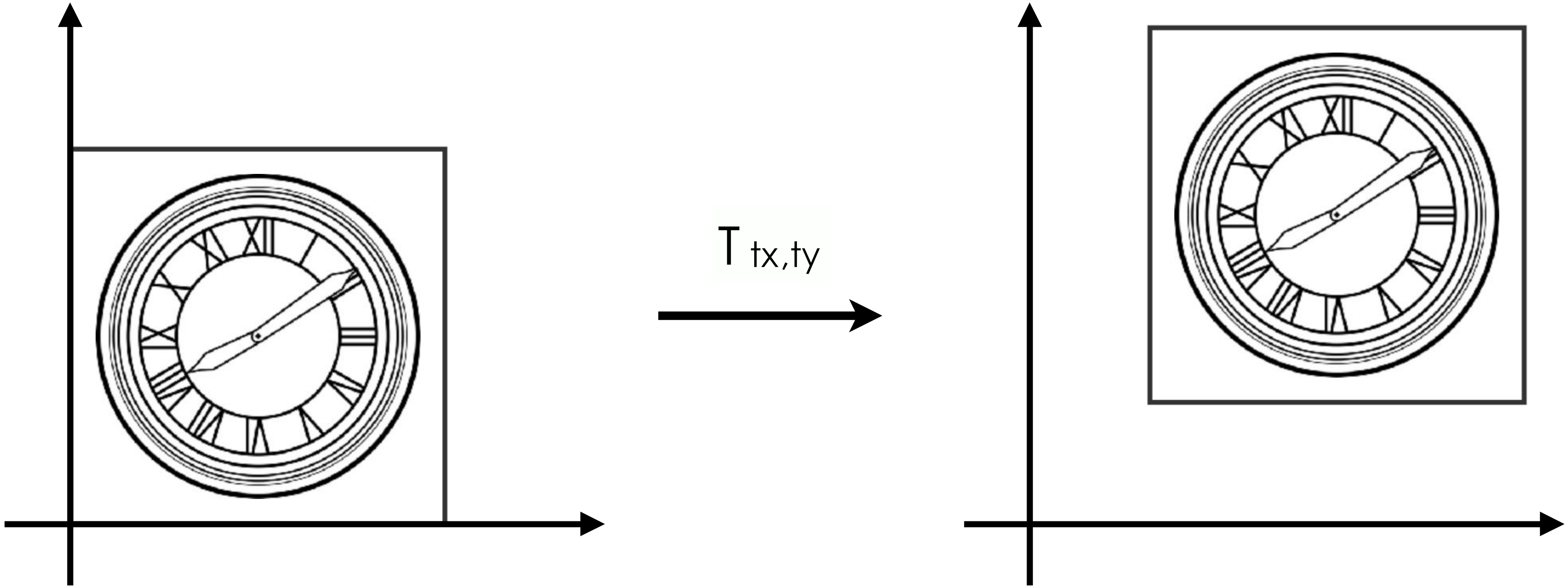
Outline

- Why study transformation
- 2D transformations
- Homogeneous coordinates
  - Why homogeneous coordinates
  - Affine transformation

Translation



Translation??



$$x' = x + t_x$$
$$y' = y + t_y$$

Why Homogeneous Coordinates

- Translation cannot be represented in matrix form

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

(So, translation is NOT linear transform!)

- But we don't want translation to be a special case

- Is there a unified way to represent all transformations? (and what's the cost?)

Solution: Homogenous Coordinates

Add a third coordinate (*w-coordinate*)

- 2D point = $(x, y, 1)^T$
- 2D vector = $(x, y, 0)^T$

Matrix representation of translations

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

What if you translate a vector?

Homogenous Coordinates

Valid operation if w-coordinate of result is 1 or 0

- vector + vector = vector
- point – point = vector
- point + vector = point
- point + point = ??

In homogeneous coordinates,

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \text{ is the 2D point } \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix}, \quad w \neq 0$$

Affine Transformations （仿射变换）

Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Using homogenous coordinates:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D Transformations

What's the order?
Linear Transform first or Translation first?

Scale

$$\mathbf{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation

$$\mathbf{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Translation

$$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

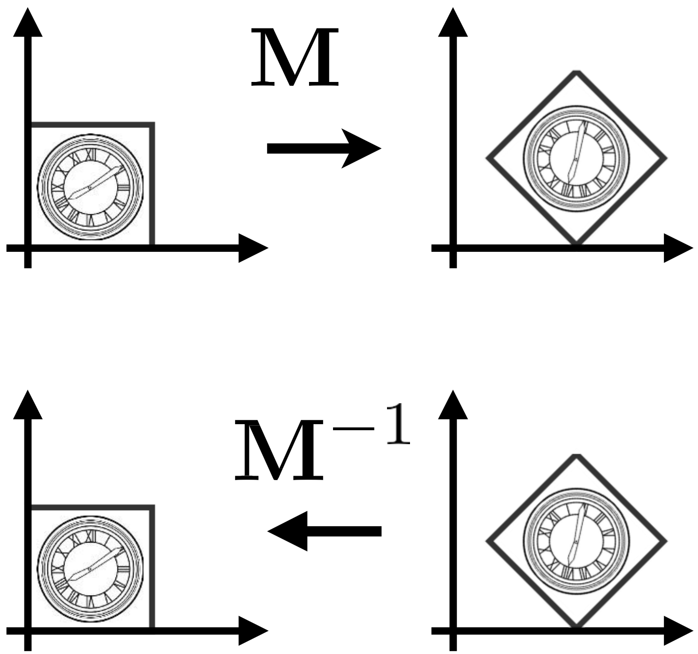
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

homogenous coordinates

Inverse Transform

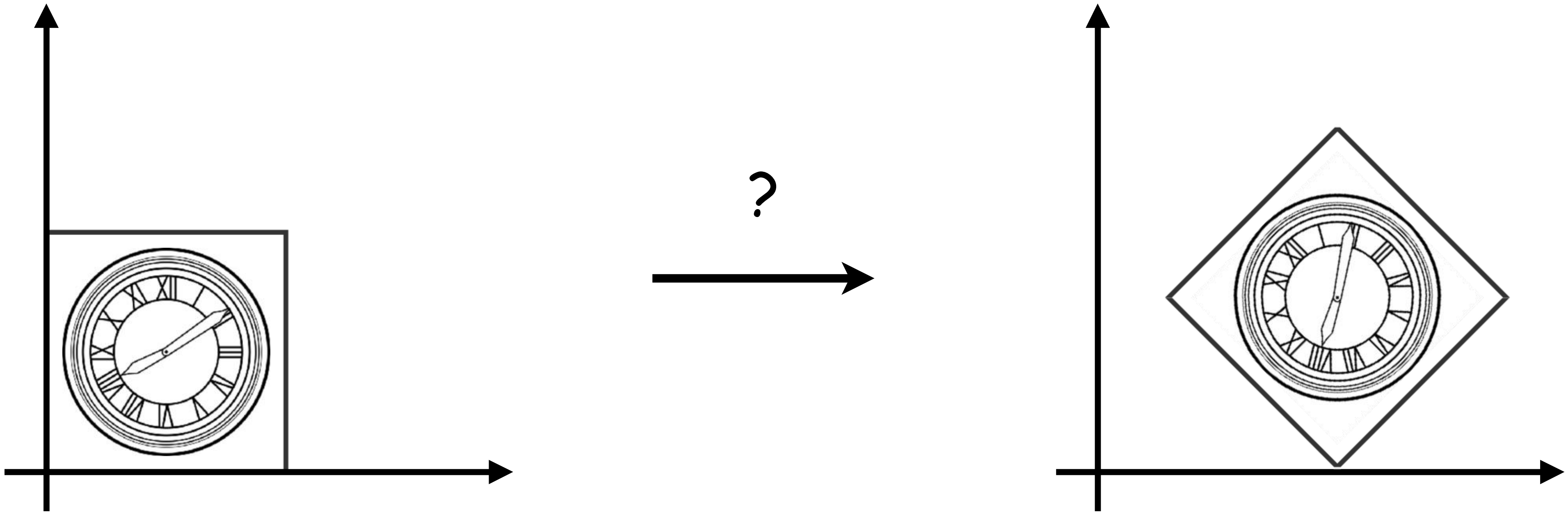
$$\mathbf{M}^{-1}$$

$\mathbf{M}^{-1}$ is the inverse of transform $\mathbf{M}$ in both a matrix and geometric sense

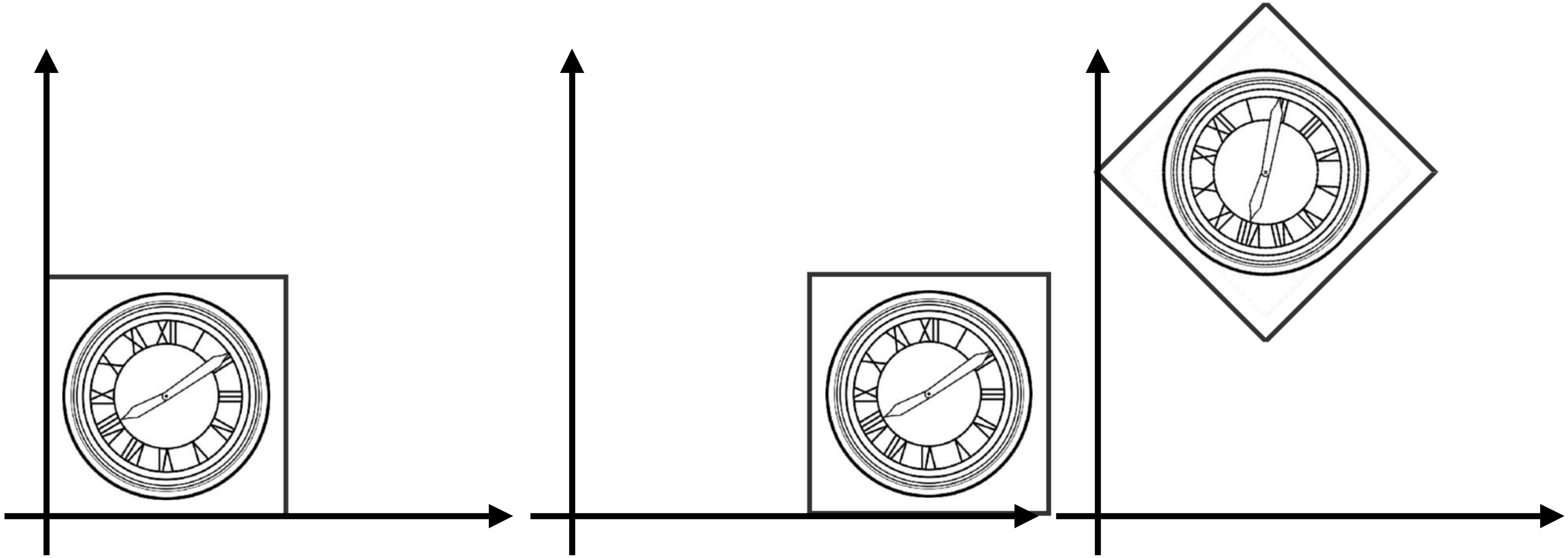


Composing Transforms
组合变换

Composite Transform

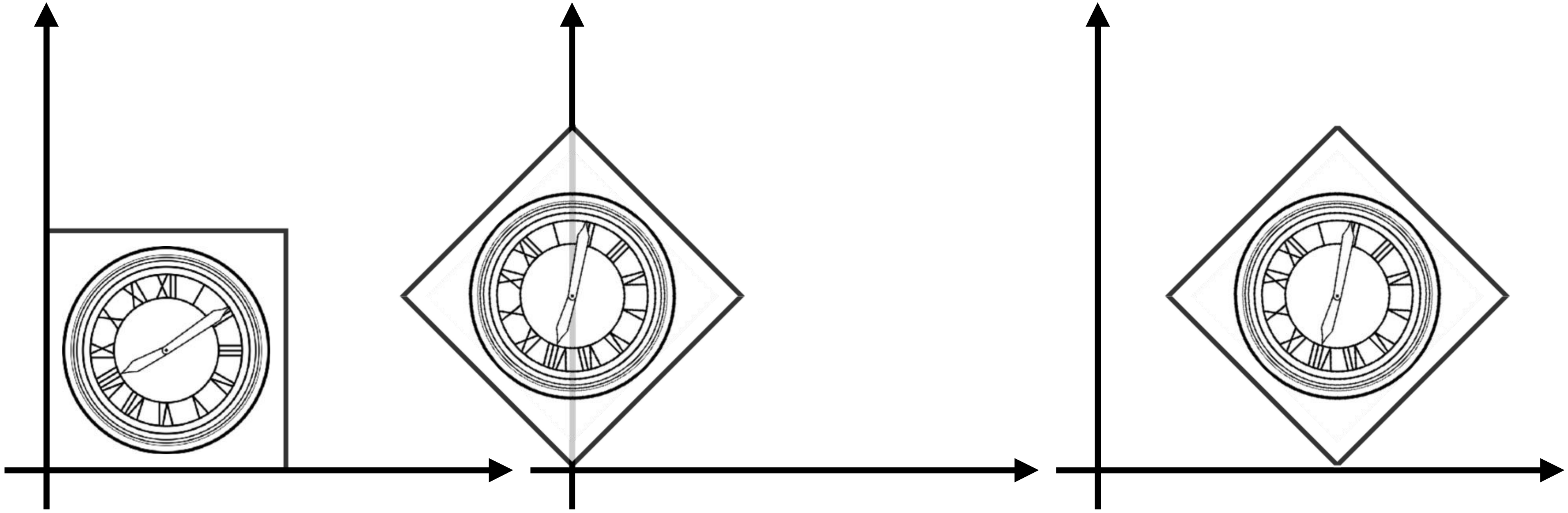


Translate Then Rotate?



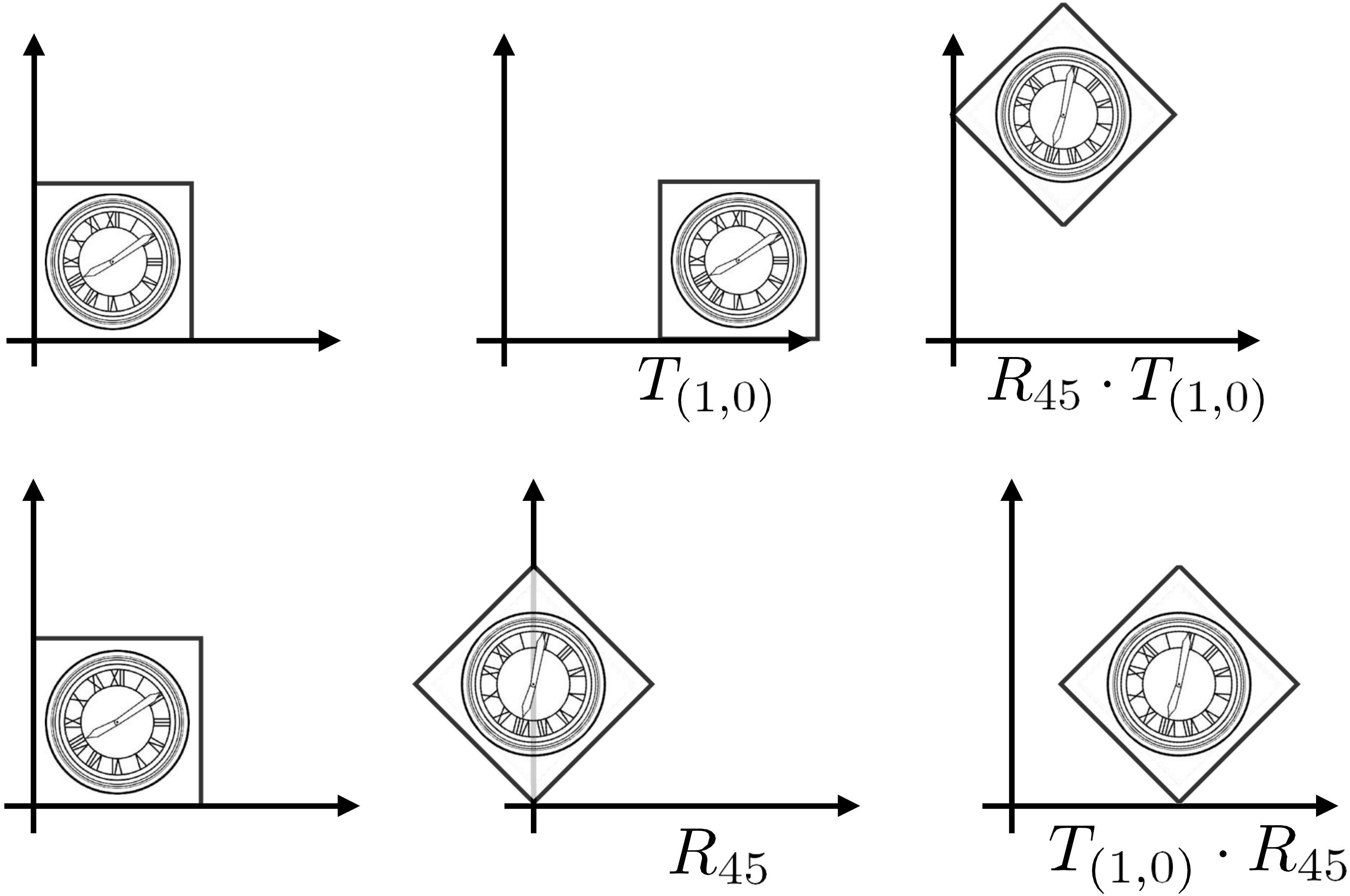
$T_{(1,0)}$ $\qquad$ $R_{45} \cdot T_{(1,0)}$

Rotate Then Translate!



R_{45} $\qquad$ $T_{(1,0)} \cdot R_{45}$

Transform Ordering Matters!



Transform Ordering Matters!

Matrix multiplication is <u>not</u> commutative

$$R_{45} \cdot T_{(1,0)} \neq T_{(1,0)} \cdot R_{45}$$

Note that matrices are applied right to left:

$$T_{(1,0)} \cdot R_{45} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Composing Transforms

Sequence of affine transforms A_1, A_2, A_3, ...

- Compose by matrix multiplication

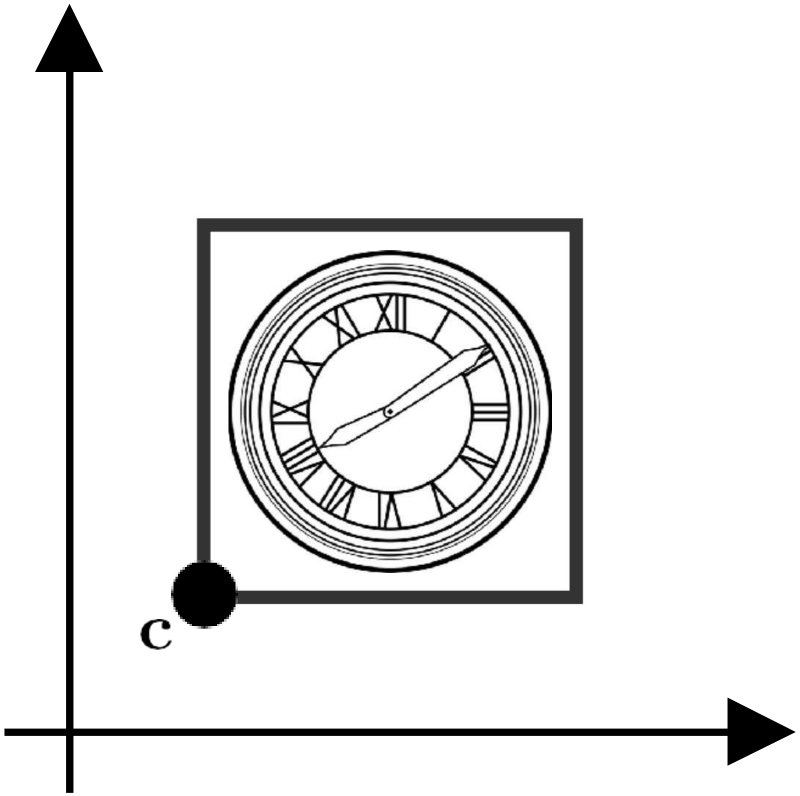
$$A_n(\ldots A_2(A_1(\mathbf{x}))) \;=\; \underbrace{\mathbf{A}_n \cdots \mathbf{A}_2 \cdot \mathbf{A}_1}_{} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Pre-multiply n matrices to obtain a single matrix representing combined transform

Very important for performance!

Decomposing Complex Transforms

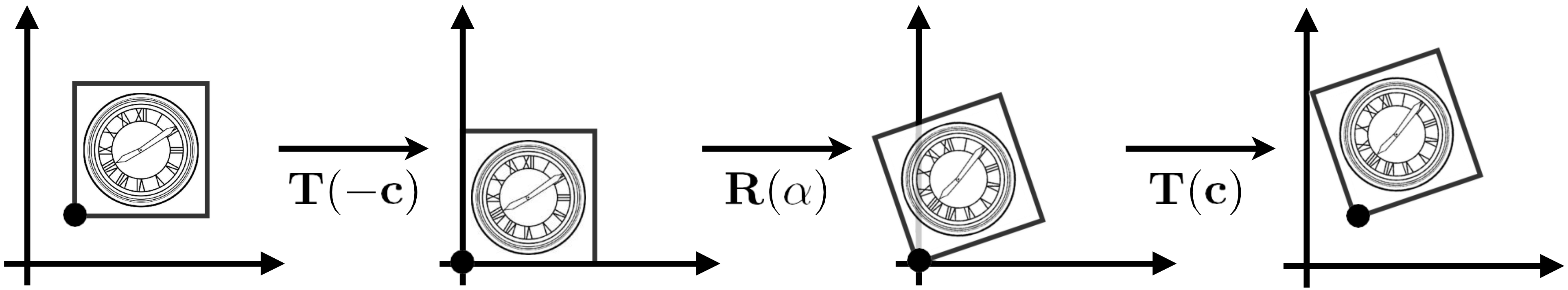
How to rotate around a given point c?



Decomposing Complex Transforms

How to rotate around a given point c?

1. Translate center to origin
2. Rotate
3. Translate back



Matrix representation?

$$\mathbf{T}(\mathbf{c}) \cdot \mathbf{R}(\alpha) \cdot \mathbf{T}(-\mathbf{c})$$

3D Transforms

3D Transformations

Use homogeneous coordinates again:

- 3D point = $(x, y, z, 1)^T$
- 3D vector = $(x, y, z, 0)^T$

In general, (x, y, z, w) ($w \neq 0$) is the 3D point:

$$(x/w, y/w, z/w)$$

3D Transformations

Use 4×4 matrices for affine transformations

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D Transformations

Scale

$$\mathbf{S}(s_x, s_y, s_z) \;=\; \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation

$$\mathbf{T}(t_x, t_y, t_z) \;=\; \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

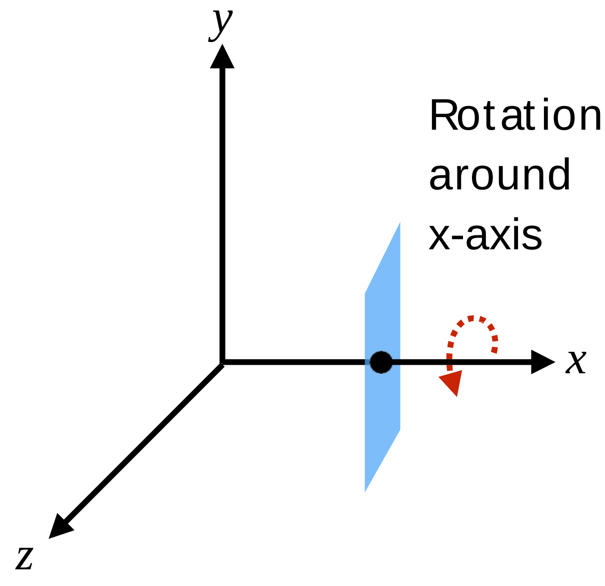
3D Transformations

Rotation around x-, y-, or z-axis

$$\mathbf{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



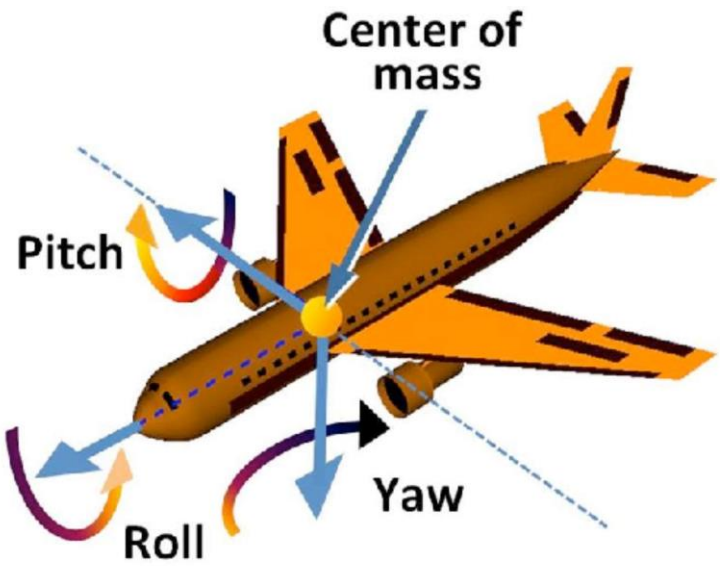
Anything strange about R_y?

3D Rotations

Compose any 3D rotation from R_x, R_y, R_z?

$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) \;=\; \mathbf{R}_x(\alpha)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\gamma)$$

- So-called *Euler angles*
- Often used in flight simulators: roll, pitch, yaw



Rodrigues' Rotation Formula

Rotation by angle α around axis n

$$\mathbf{R}(\mathbf{n},\alpha) \;=\; \cos(\alpha)\,\mathbf{I} \;+\; (1-\cos(\alpha))\,\mathbf{n}\mathbf{n}^T \;+\; \sin(\alpha)\underbrace{\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}}_{\mathbf{N}}$$

How to prove this magic formula?

Check out the supplementary material on the course website!

Rodrigues’ Rotation Formula

发现历程和定义

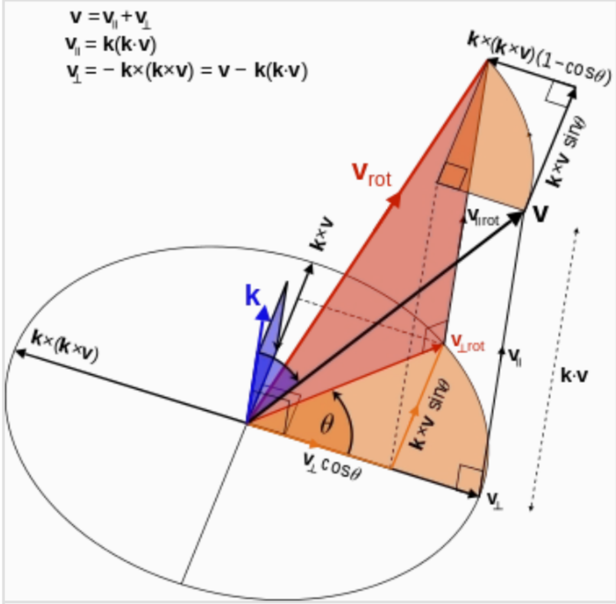
播报 编辑

在向量旋转公式发现以前，瑞士数学家列昂哈德·欧拉(Leonhard Euler(1707-1783))为了证明四平方和定理，发现了四平方和恒等式。然而这个恒等式的构造过程非常繁琐。直到后来，四元数被引入，使得这个恒等式的推导大大简化。

四元数可以很方便地表示旋转变换。但在很多场合中，使用矩阵形式和向量形式表达旋转更有利于推导。向量旋转公式最早由法国数学家本杰明·奥伦德·罗德里格(Benjamin Olinde Rodrigues(1795–1851))导出，后来被应用在很多领域。

设***v***是一个三维空间向量，***k***是旋转轴的单位向量，则***v***在右手螺旋定则意义下绕旋转轴***k***旋转角度*θ*得到的向量可以由三个不共面的向量***v***, ***k***和***k***×***v***构成的标架表示：

$$\mathbf{v}_{rot} = \cos\theta\mathbf{v} + (1-\cos\theta)(\mathbf{v}\cdot\mathbf{k})\mathbf{k} + \sin\theta\mathbf{k}\times\mathbf{v}$$



罗德里格旋转公式

Thank you!