

Data-driven computer animation

Tutorial 1

Prof. Taku Komura

TA: Huancheng Lin(lamws@connect.hku.hk), Mingyi Shi(myshi@cs.hku.hk)

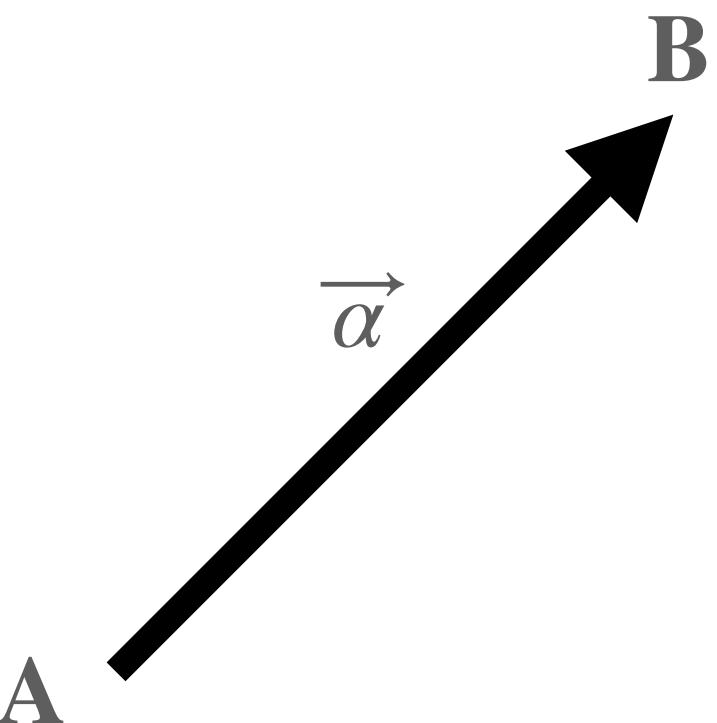
Tutorial 1.1: Basic Linear Algebra in Graphics

Computer Graphics

- More dependent on Linear Algebra
 - Vectors
 - Matrices
- Examples
 - A point is a vector.
 - An operation like translating or rotating objects can be matrix-vector multiplication.

Vectors

- Usually written as $\vec{\alpha}$ or in bold α .
- Direction and length.
- No absolute starting position.

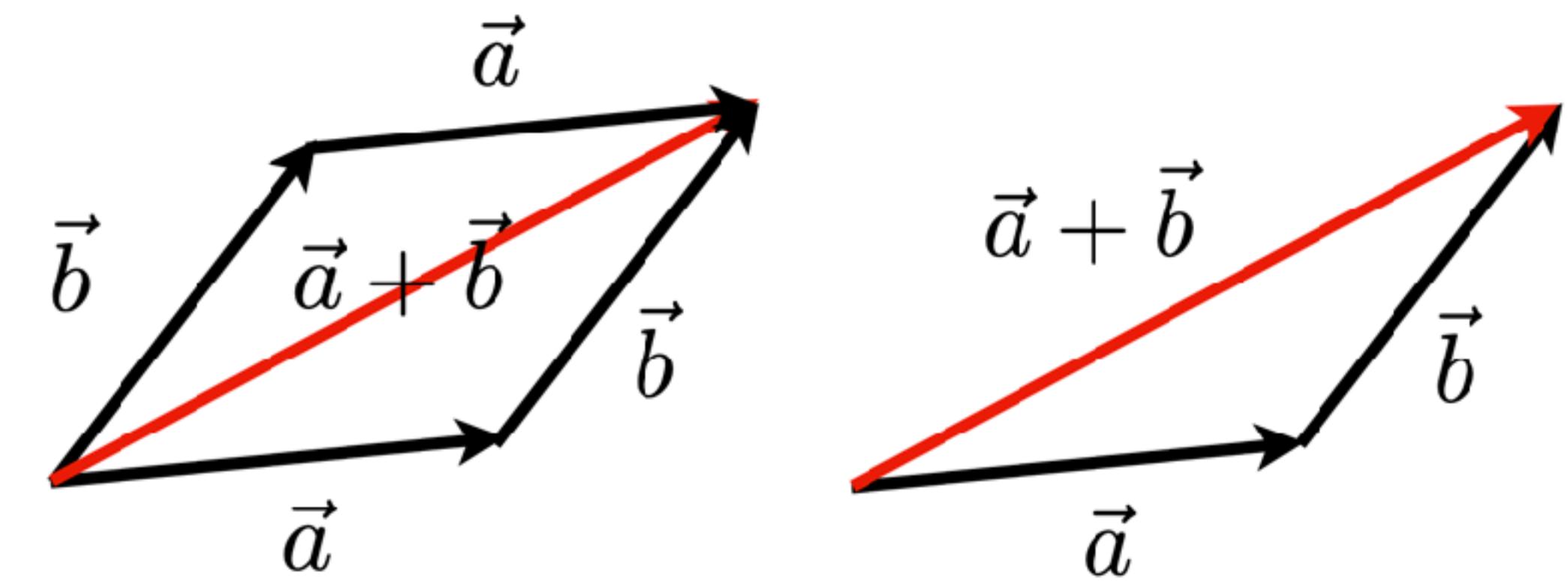


Vector Normalization

- Magnitude (length) of a vector written as $\|\vec{a}\|$
- Unit Vector
 - A vector with a magnitude of 1
 - Finding the unit vector of a vector (normalization)
 - $$\hat{a} = \frac{\vec{a}}{\|\vec{a}\|}$$
 - Used to represent directions

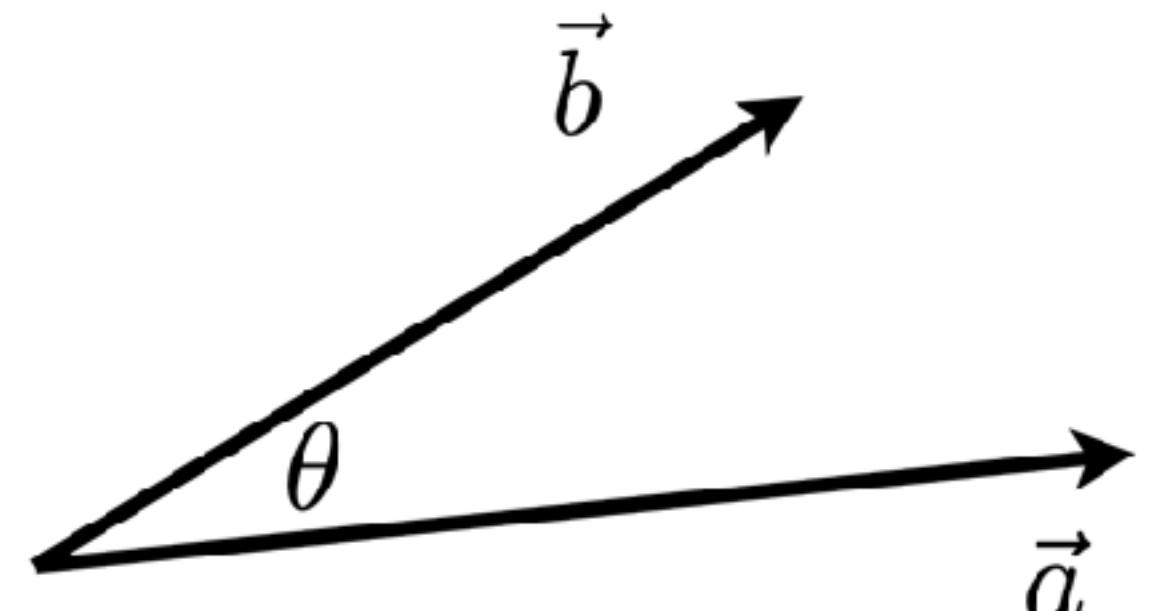
Vector Addition

- Algebraically: Simply add coordinates
- Geometrically: Parallelogram law & Triangle law



Vector Dot Product

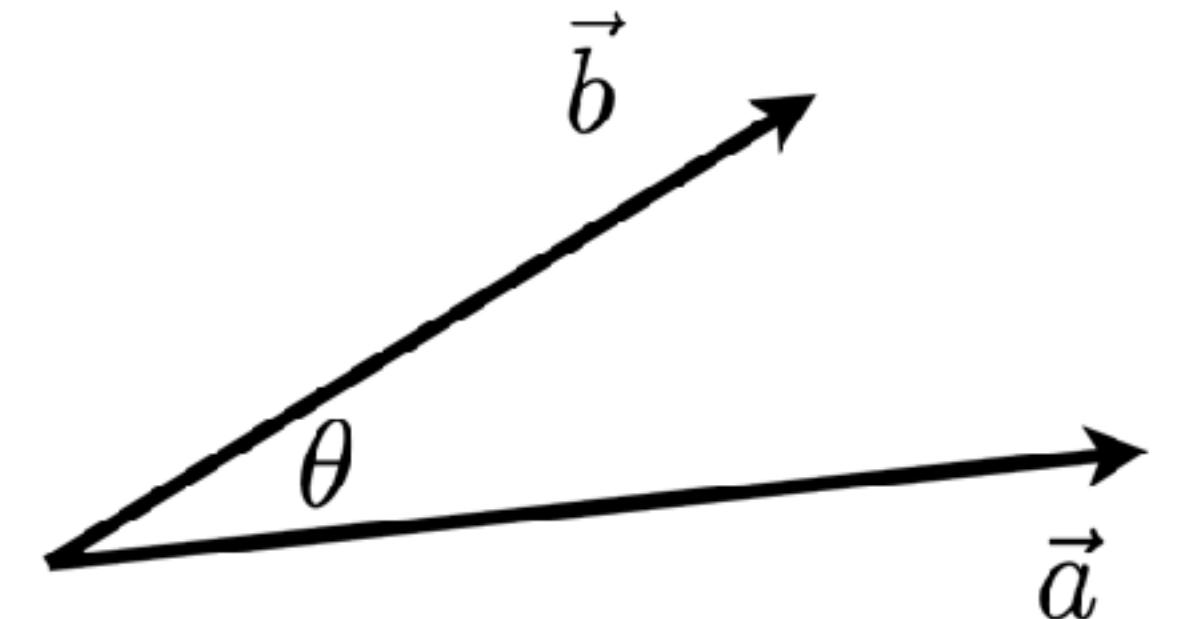
- $\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$
- Component-wise multiplication, then adding up
 - $\vec{a} \cdot \vec{b} = \begin{pmatrix} x_a \\ y_a \end{pmatrix} \cdot \begin{pmatrix} x_b \\ y_b \end{pmatrix} = x_a x_b + y_a y_b$ in 2D.
 - $\vec{a} \cdot \vec{b} = x_a x_b + y_a y_b + z_a z_b$ in 3D.



Dot Product in Graphics

- Find the angle between two vectors.

- $\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$

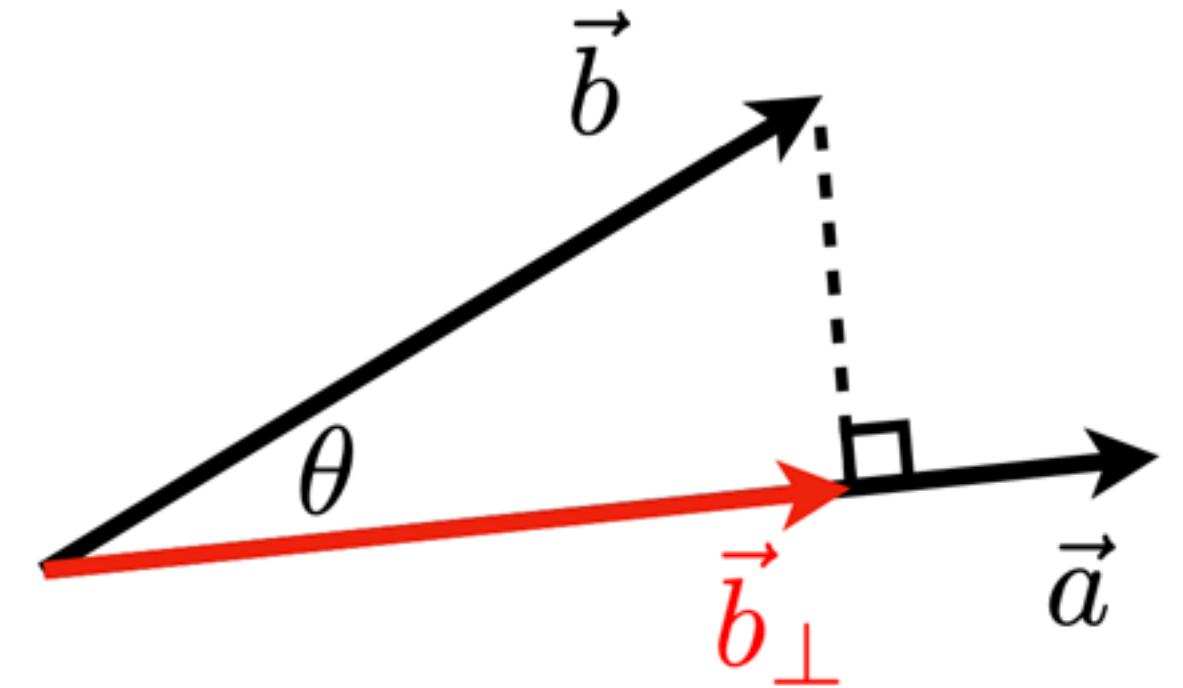


- Finding the projection of one vector on another

- \vec{b}_\perp is the projection of \vec{b} onto \vec{a} .

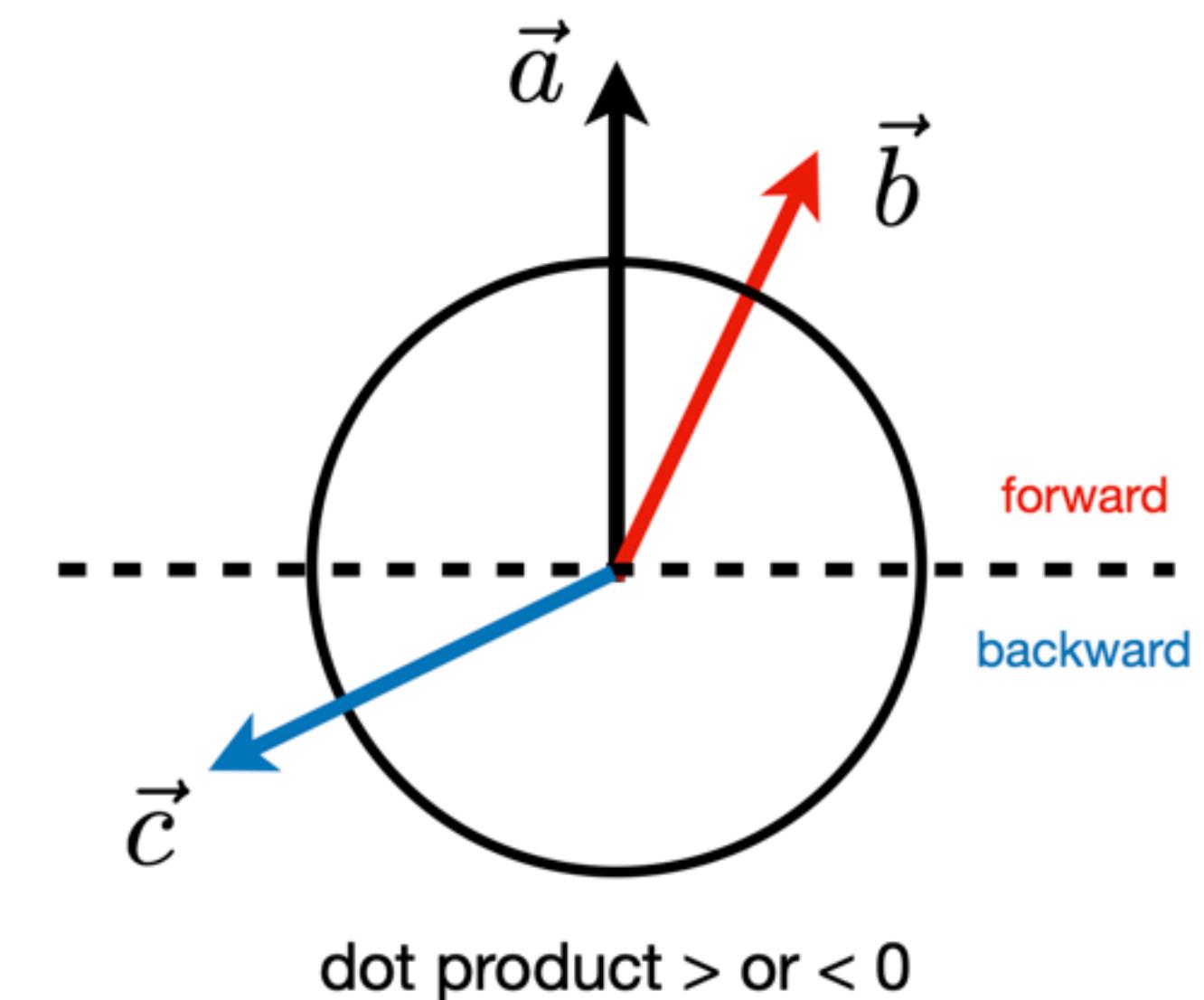
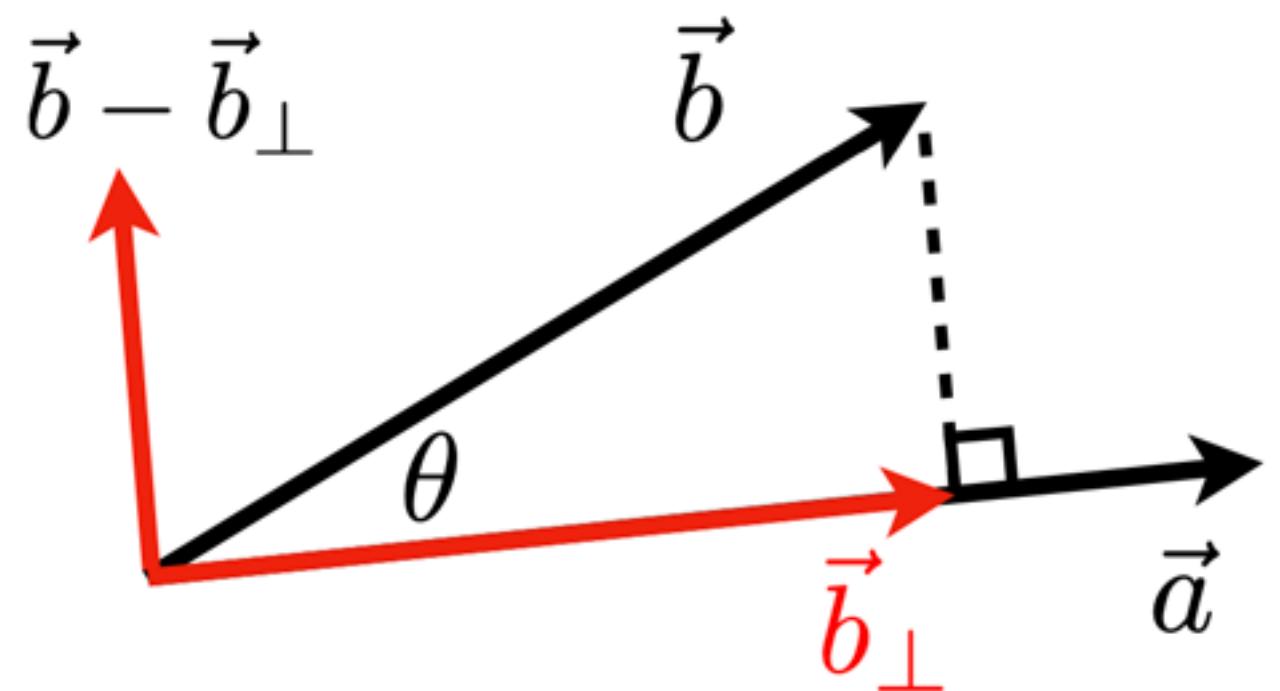
- $\|\vec{b}_\perp\| = \|\vec{b}\| \cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|}$

- $\vec{b}_\perp = \|\vec{b}_\perp\| \hat{a}$



Dot Product in Graphics

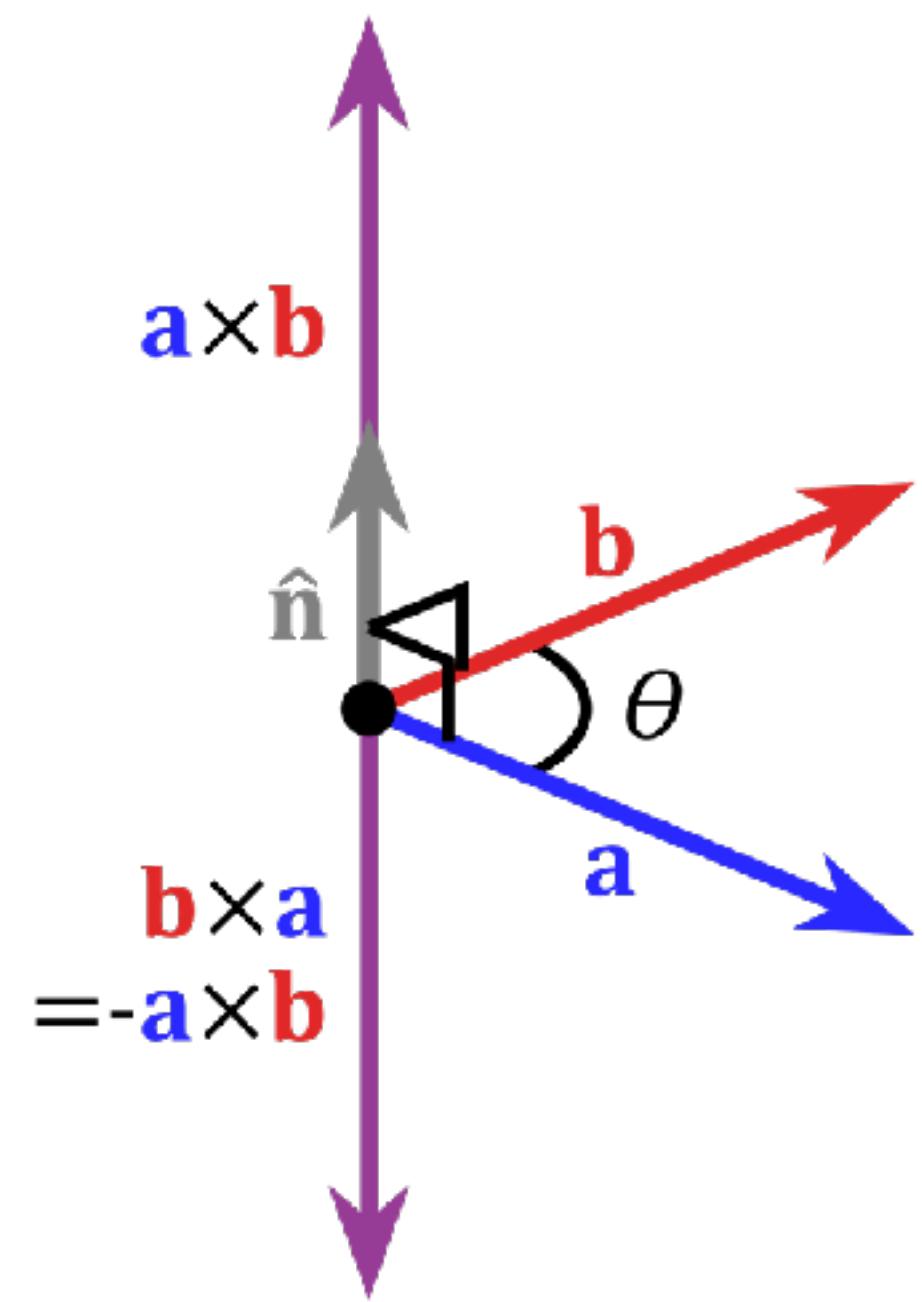
- Measure how close two directions are
- Decompose a vector
- Determine forward or backward



Cross Product

Properties

- Cross product is orthogonal to two initial vectors.
- Direction determined by the right-hand rule



Cross Product Properties

$$\vec{x} \times \vec{y} = +\vec{z}$$

$$\vec{y} \times \vec{x} = -\vec{z}$$

$$\vec{y} \times \vec{z} = +\vec{x}$$

$$\vec{z} \times \vec{y} = -\vec{x}$$

$$\vec{z} \times \vec{x} = +\vec{y}$$

$$\vec{x} \times \vec{z} = -\vec{y}$$

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

$$\vec{a} \times \vec{a} = \vec{0}$$

$$\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$$

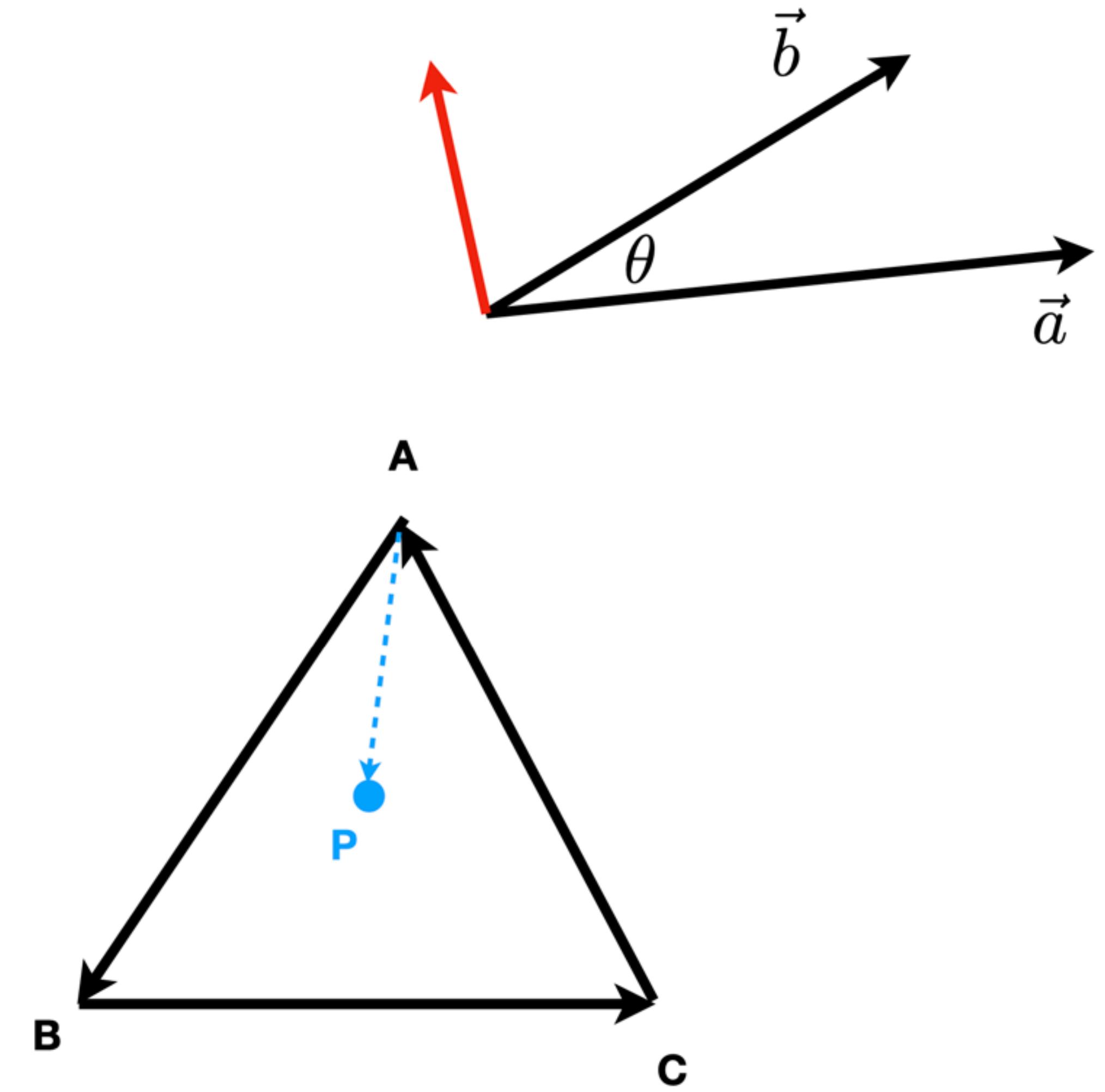
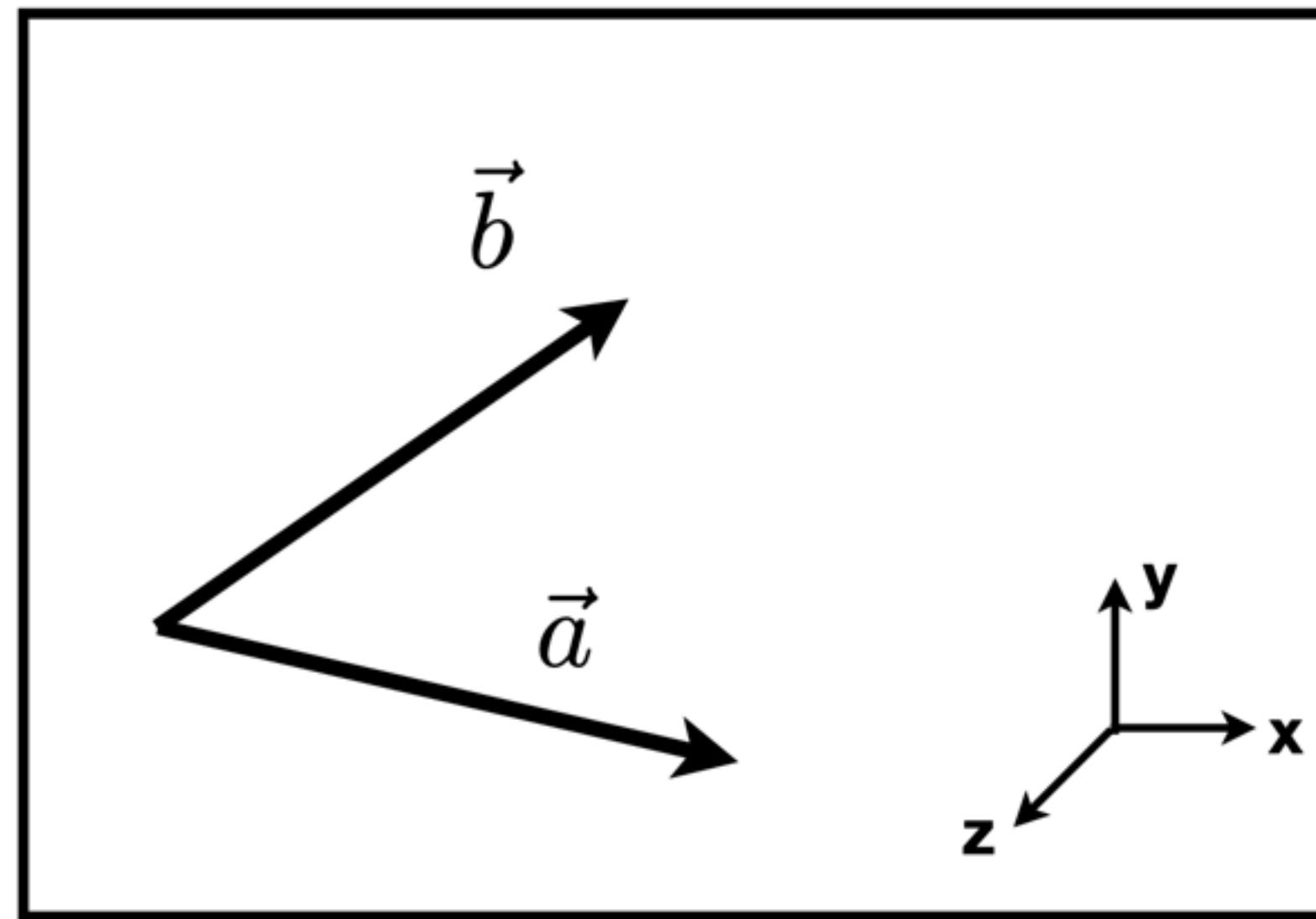
$$\vec{a} \times (k\vec{b}) = k(\vec{a} \times \vec{b})$$

Cross Product Formulation

$$\vec{a} \times \vec{b} = \begin{pmatrix} y_a z_b - y_b z_a \\ z_a x_b - x_a z_b \\ x_a y_b - y_a x_b \end{pmatrix}$$

Cross Product in Graphics

- Determine left / right
- Determine inside / outside



Matrices

- Magical 2D arrays that haunt every CS course
- In Graphics, pervasively used to represent transformations
 - Translation, rotation, shear, scale

Matrices

What is a matrix

- An $m \times n$ matrix is an array of numbers with m rows and n columns.

$$\begin{pmatrix} 1 & 3 \\ 5 & 2 \\ 0 & 4 \end{pmatrix}$$

- Addition and multiplication by a scalar are trivial:
element by element.

Matrices

Matrix-Matrix Multiplication

- # (number of) columns in A must = # rows in B ($M \times N$) ($N \times P$) = ($M \times P$)

$$AB = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} = \begin{pmatrix} 11 & 14 & 17 & 20 \\ 23 & 30 & 37 & 44 \\ 35 & 46 & 57 & 68 \end{pmatrix} = C$$

- C_{ij} = the dot product of the i-th row from matrix A and the j-th column from matrix B.

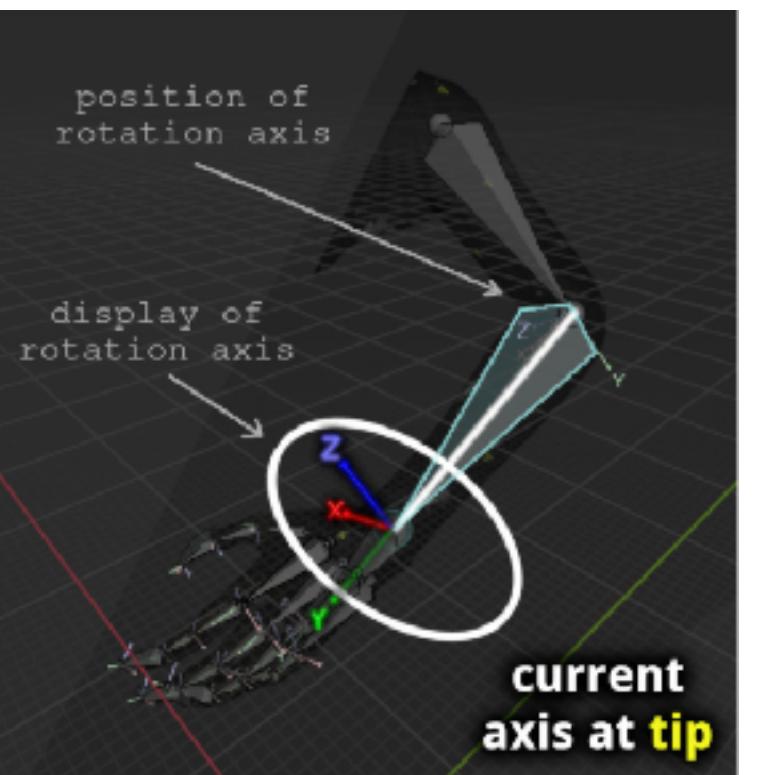
Matrices

Matrix-Vector Multiplication

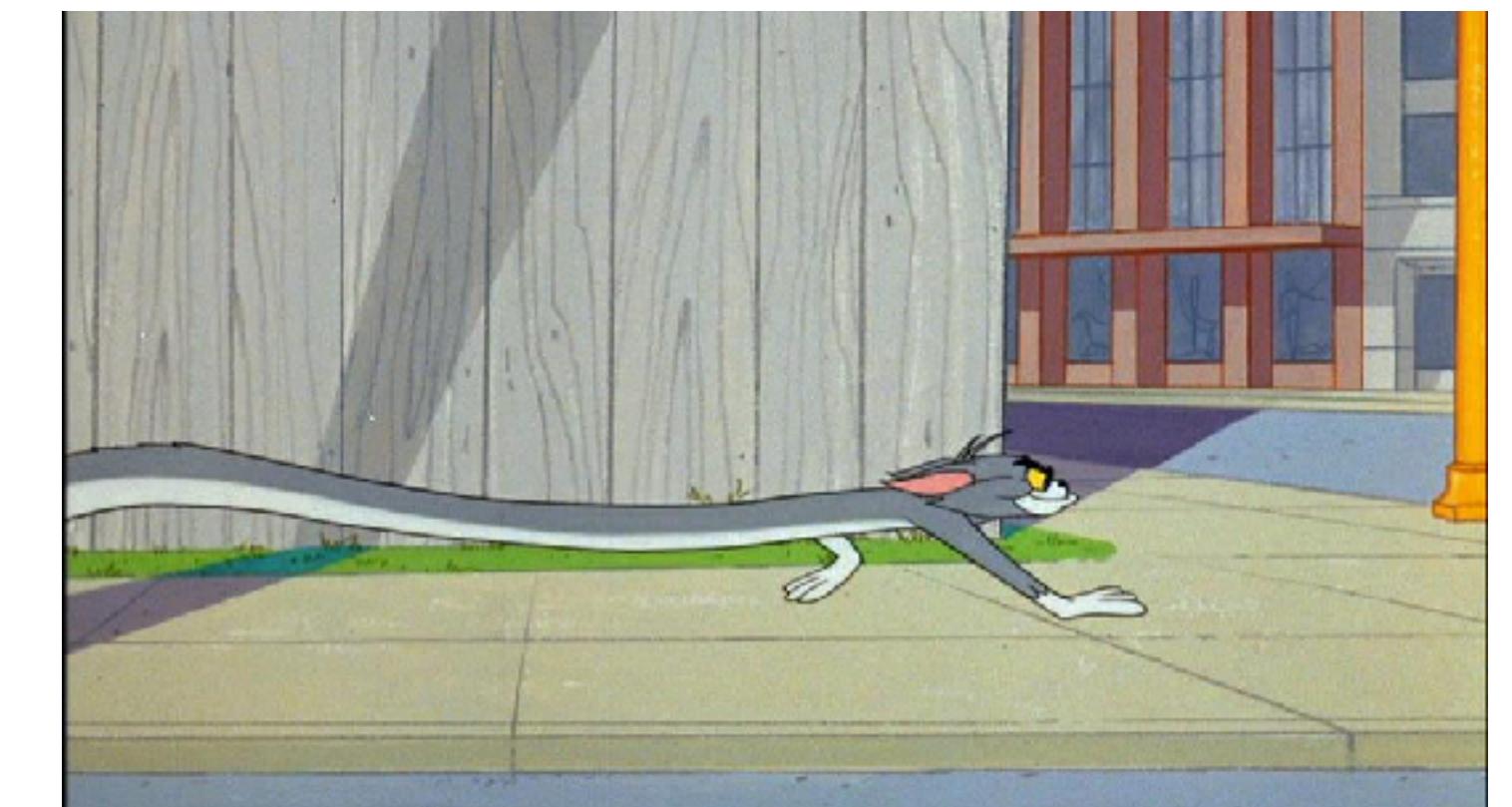
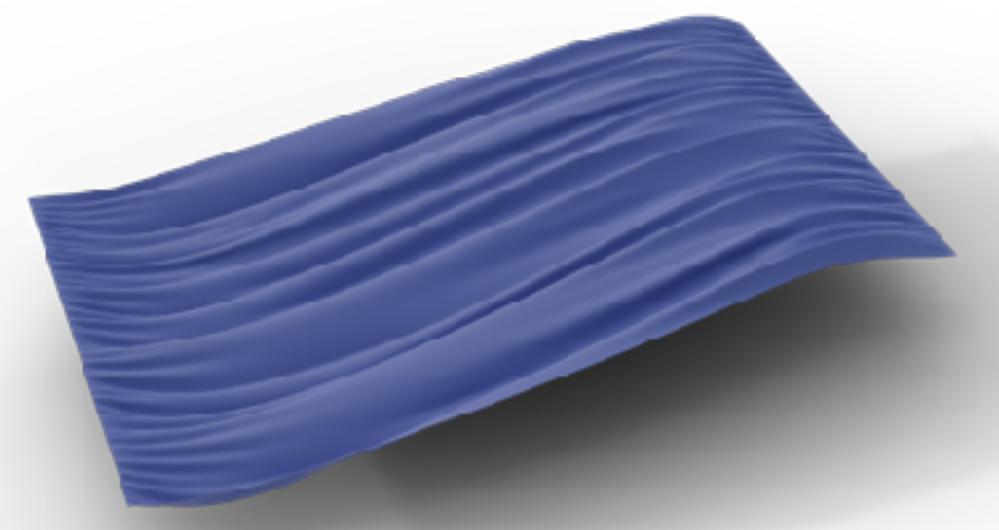
- Treat vector as a column matrix ($m \times 1$)
- Key for transforming points

Transformation

Why transformation?



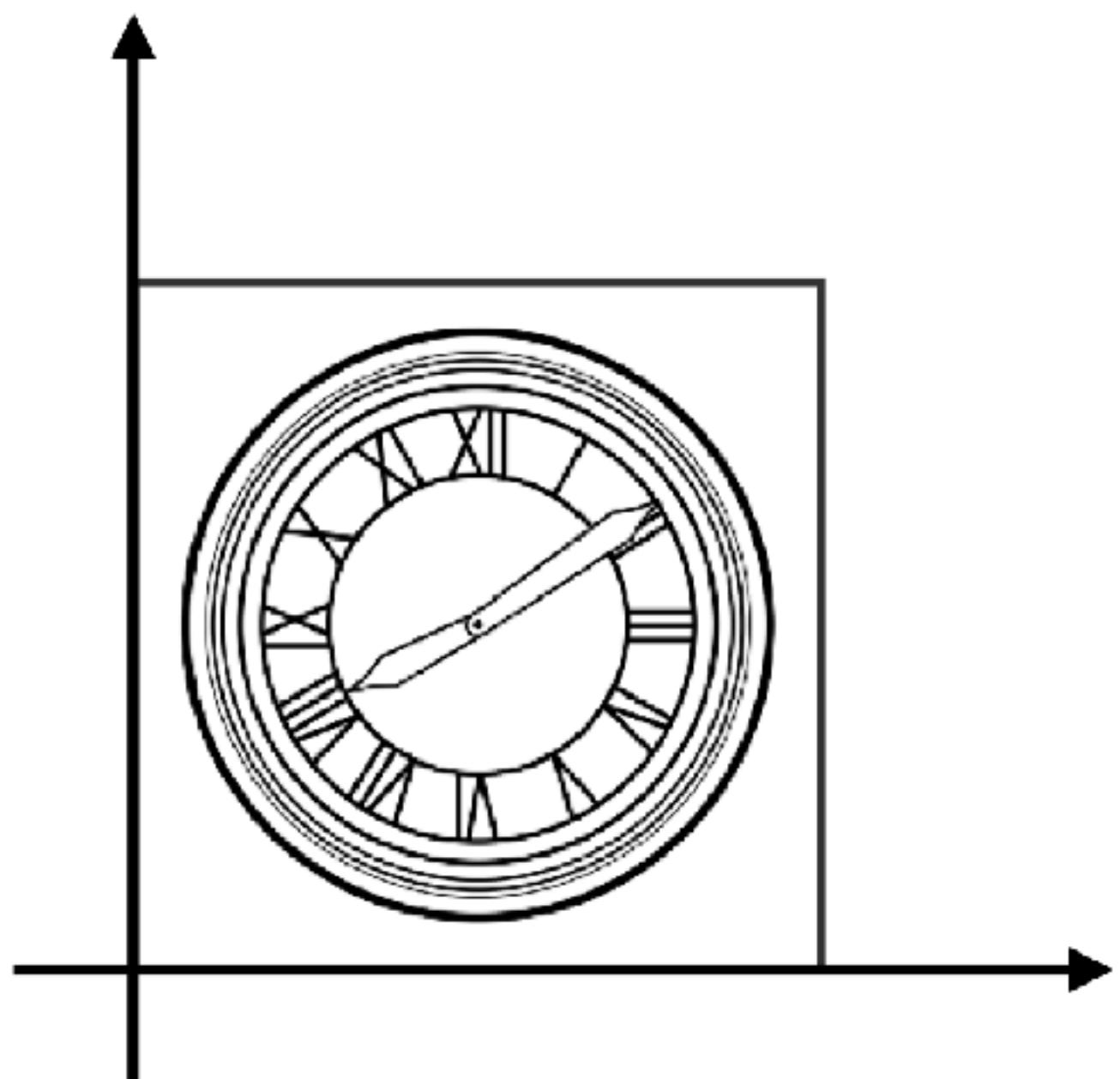
Rotation



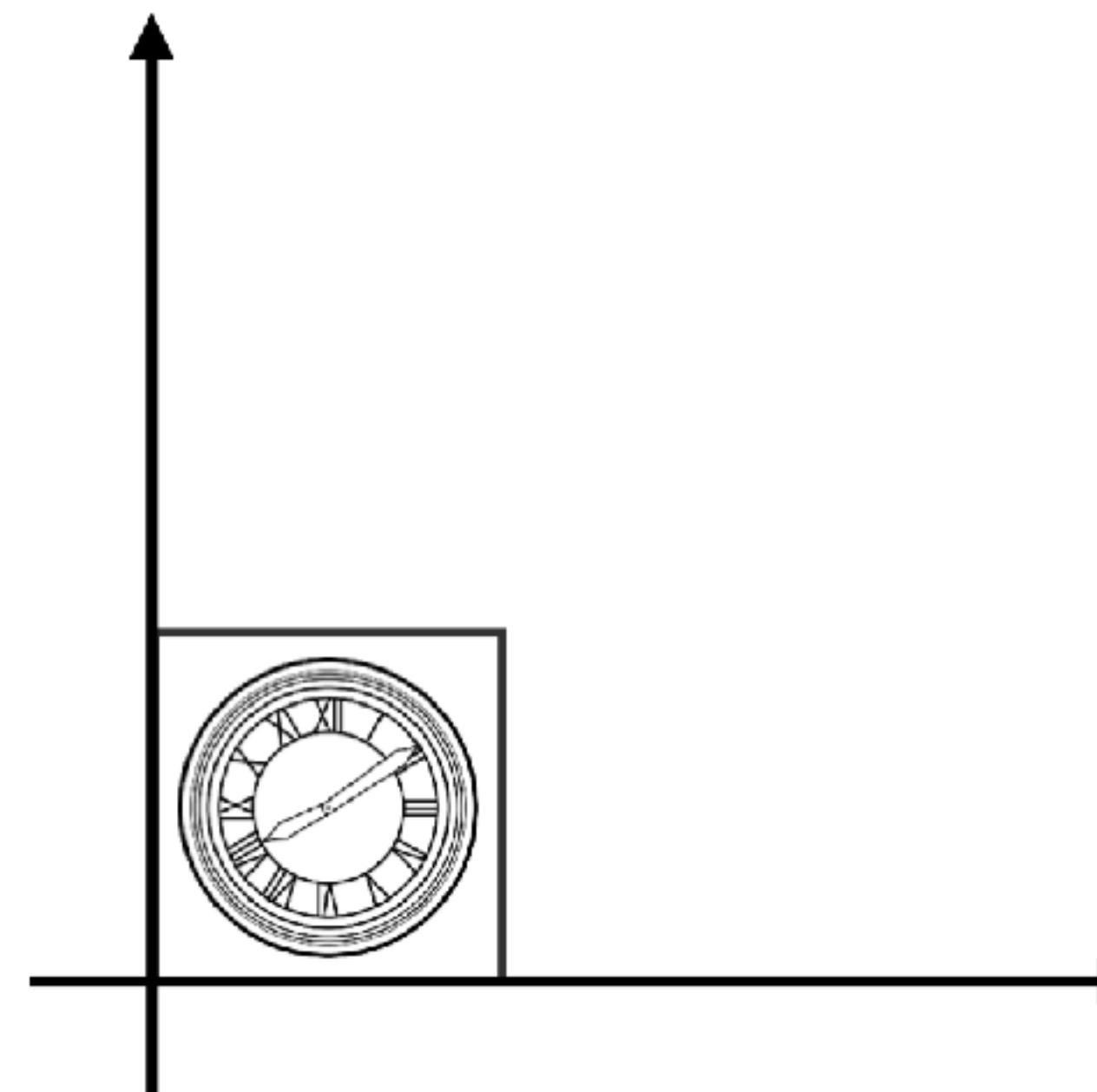
Scaling

2D transformation

Scale

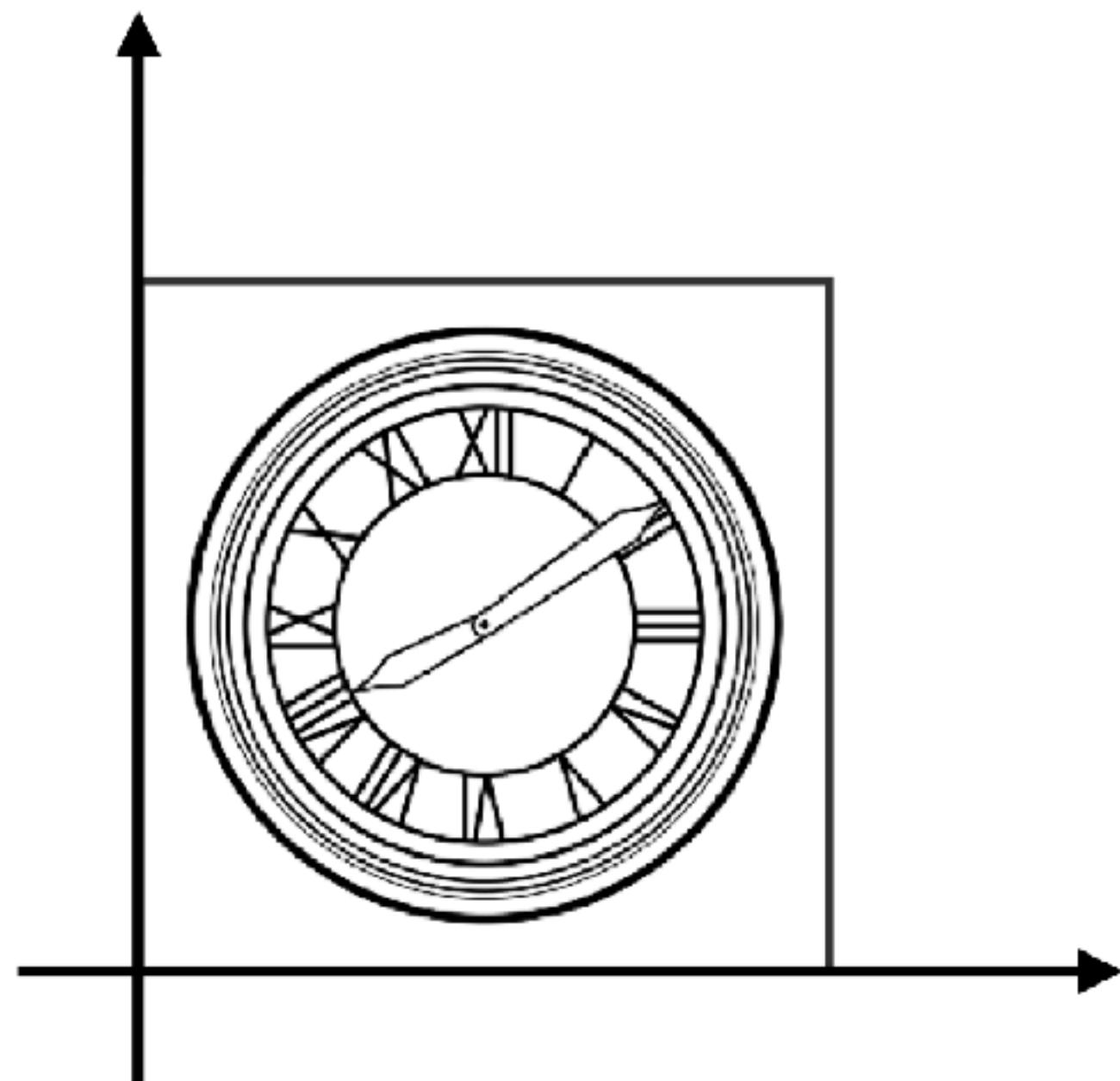


$S_{0.5}$

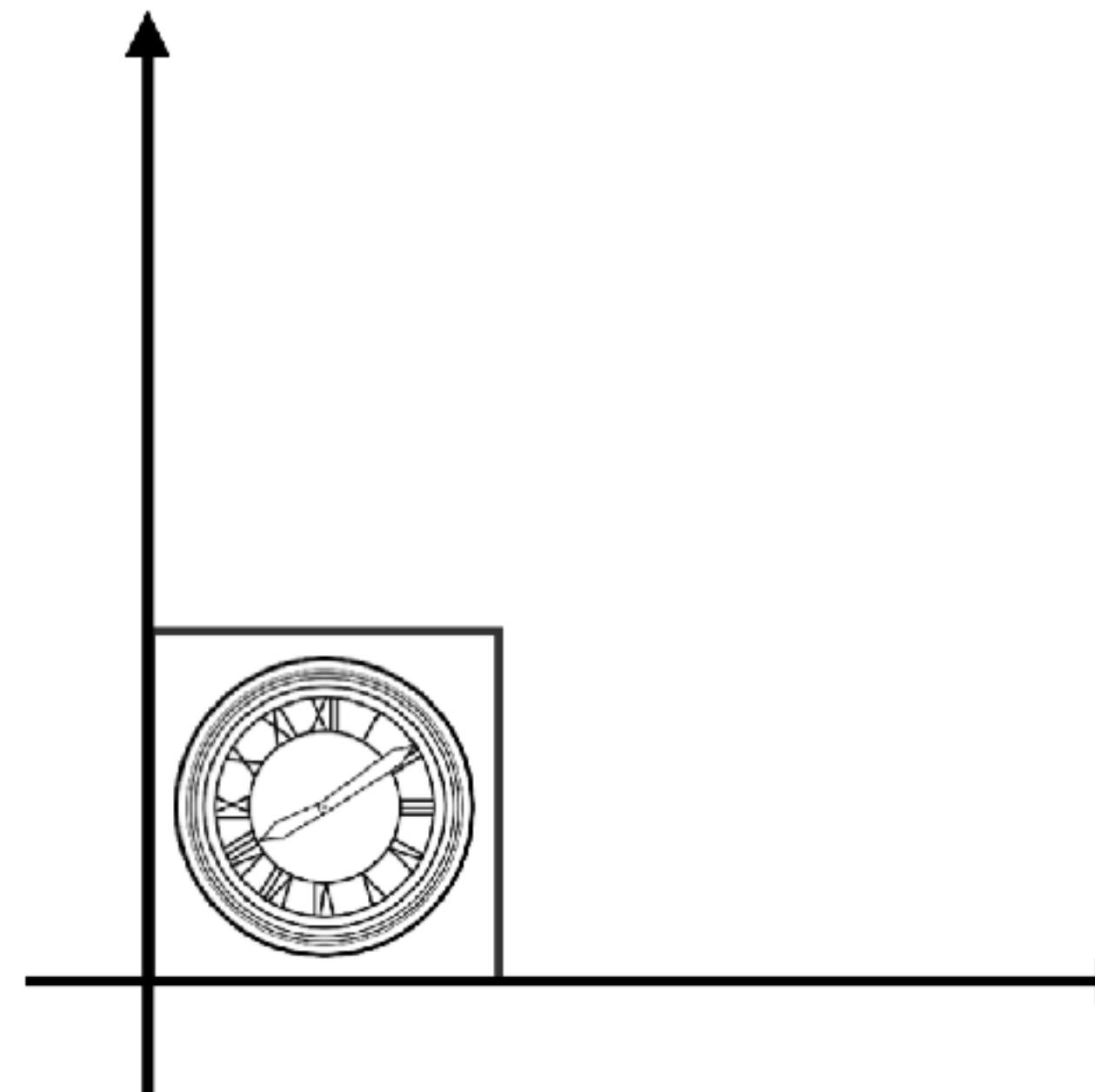
A horizontal arrow pointing from the original clock image towards the scaled version, indicating the direction of the transformation.

2D transformation

Scale



$$S_{0.5}$$
An arrow pointing from the original clock image to the scaled-down version, indicating the scaling factor.

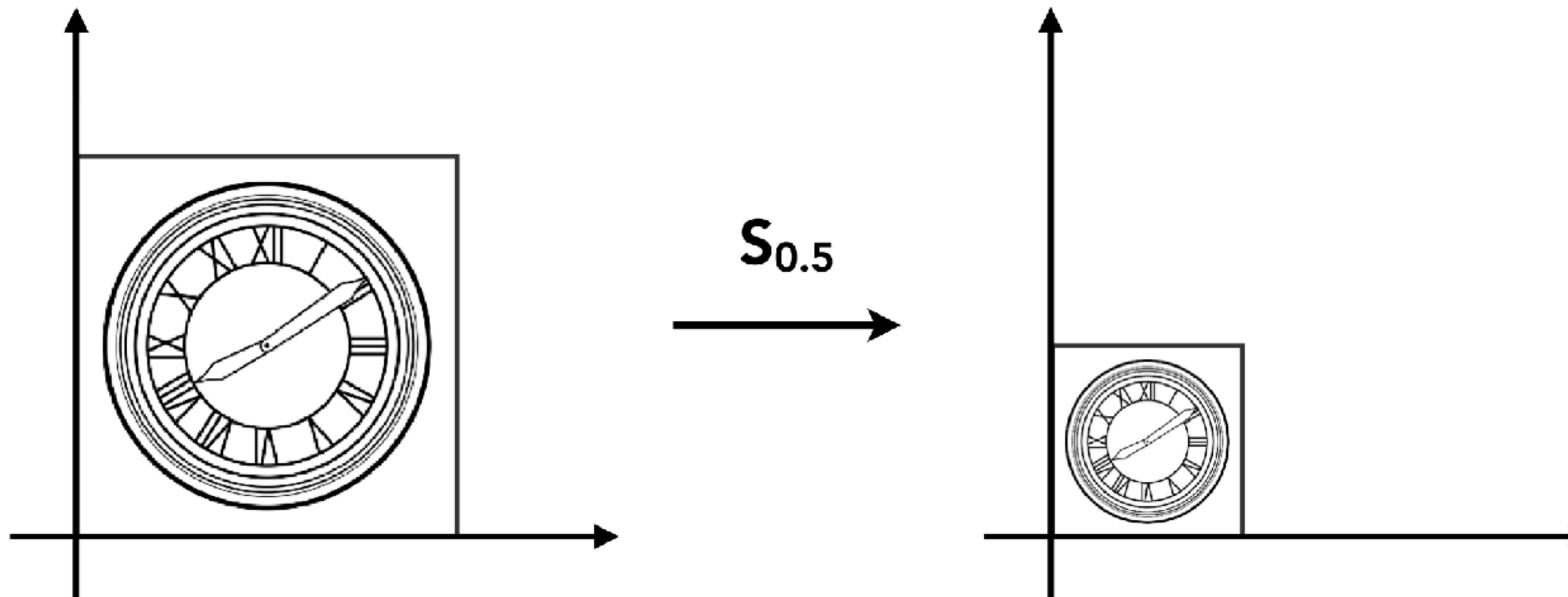


$$x' = sx$$

$$y' = sy$$

2D transformation

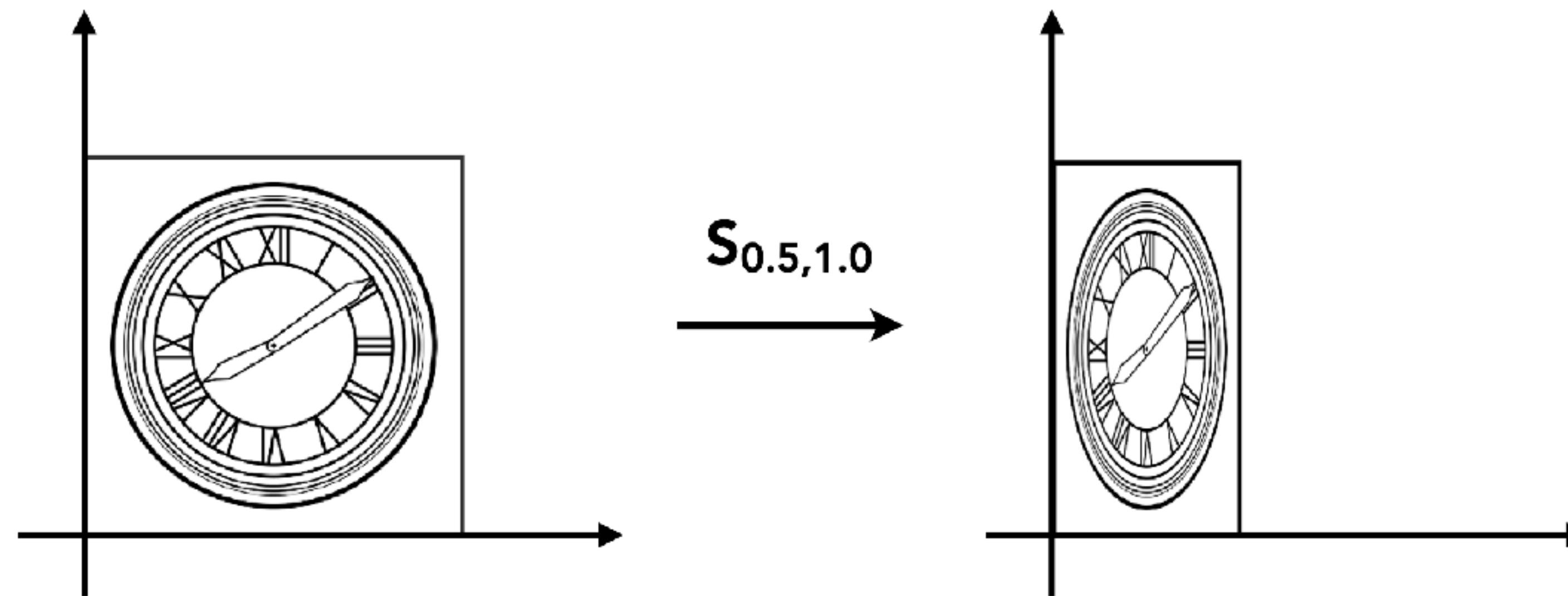
Scale



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

2D transformation

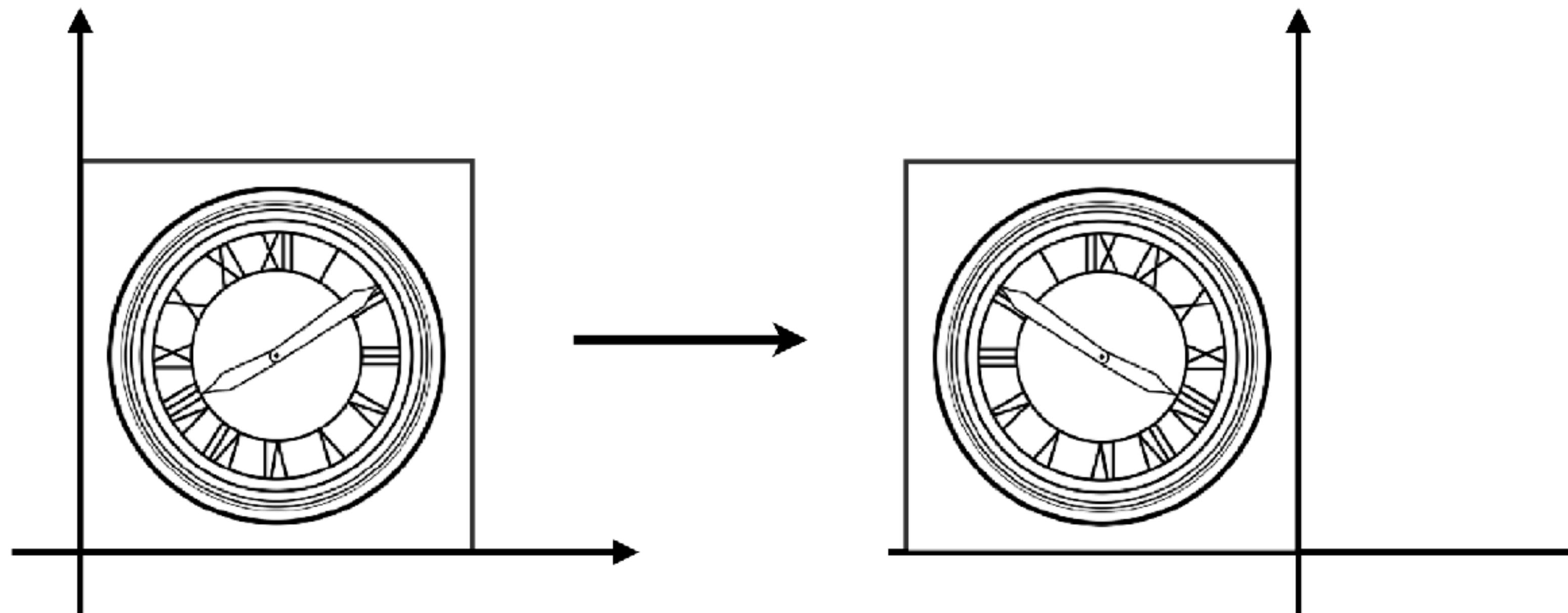
Scale (Non-Uniform)



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

2D transformation

Scale (Reflection)

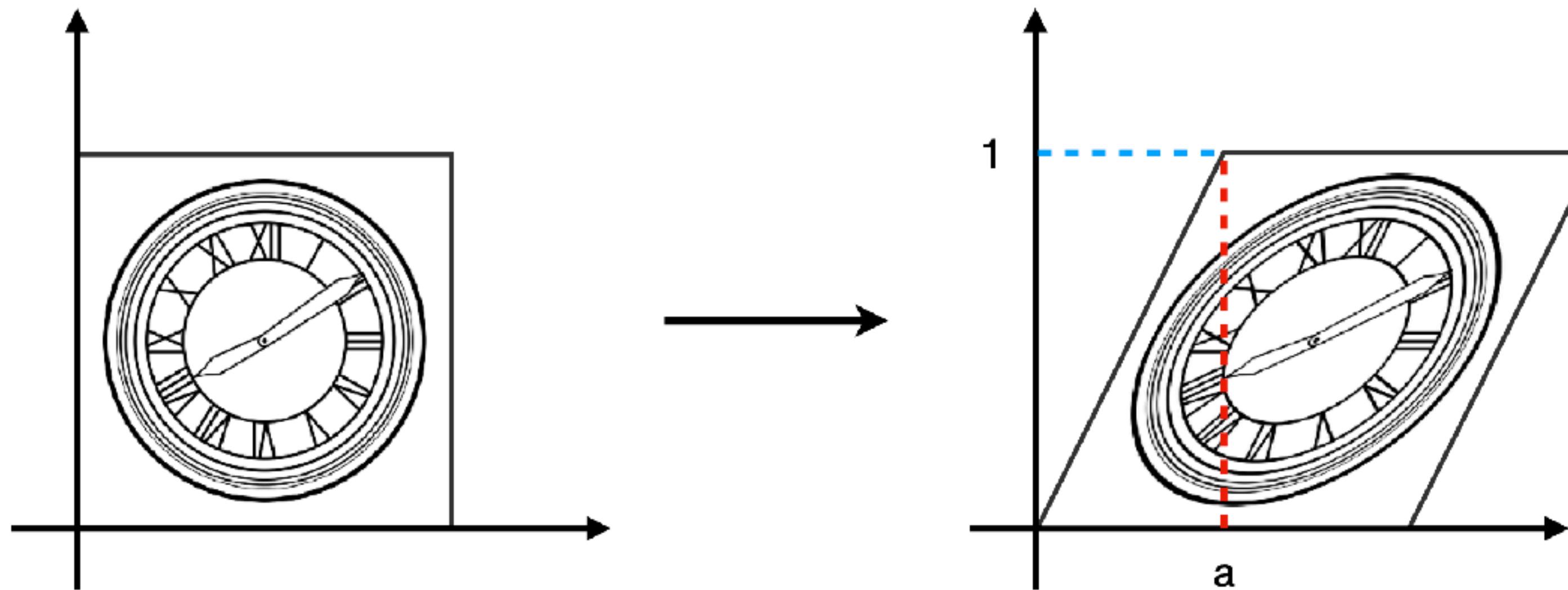


$$\begin{aligned}x' &= -x \\y' &= y\end{aligned}$$

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

2D transformation

Shear



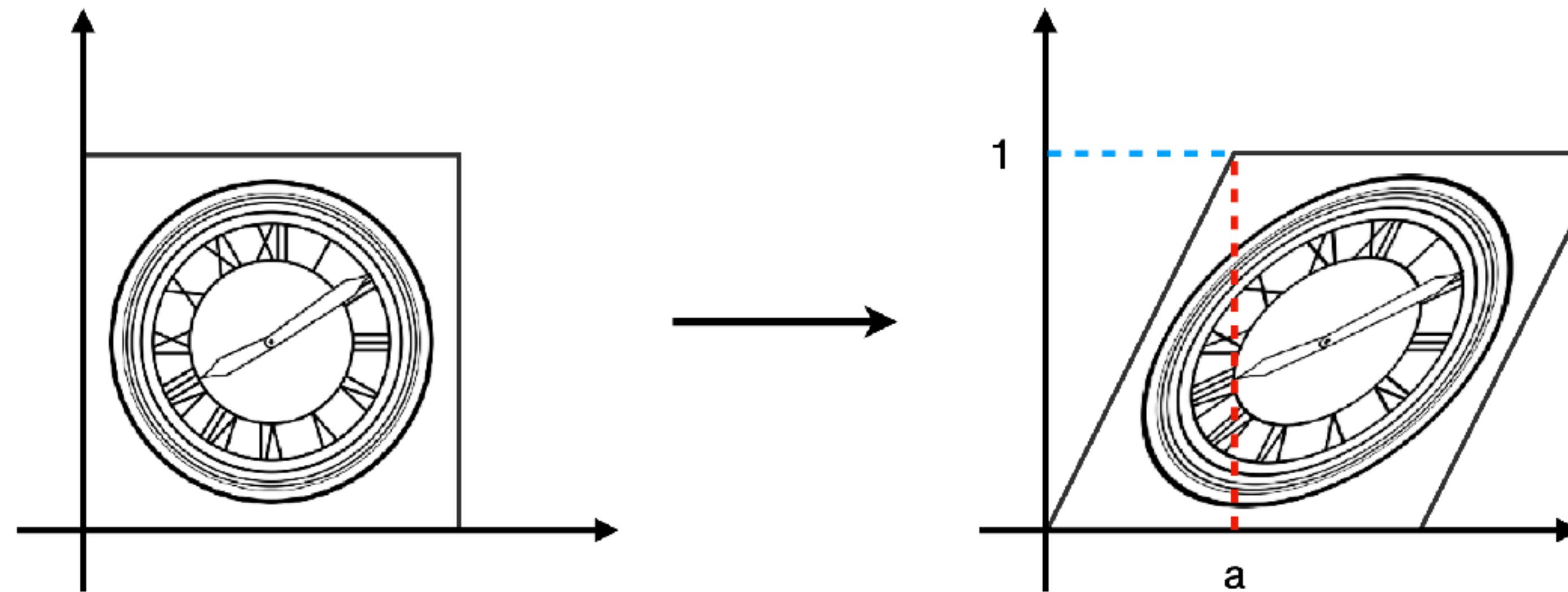
Hints:

- Horizontal shift is 0 at $y=0$
- Horizontal shift is a at $y=1$
- Vertical shift is always 0

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

2D transformation

Shear



Hints:

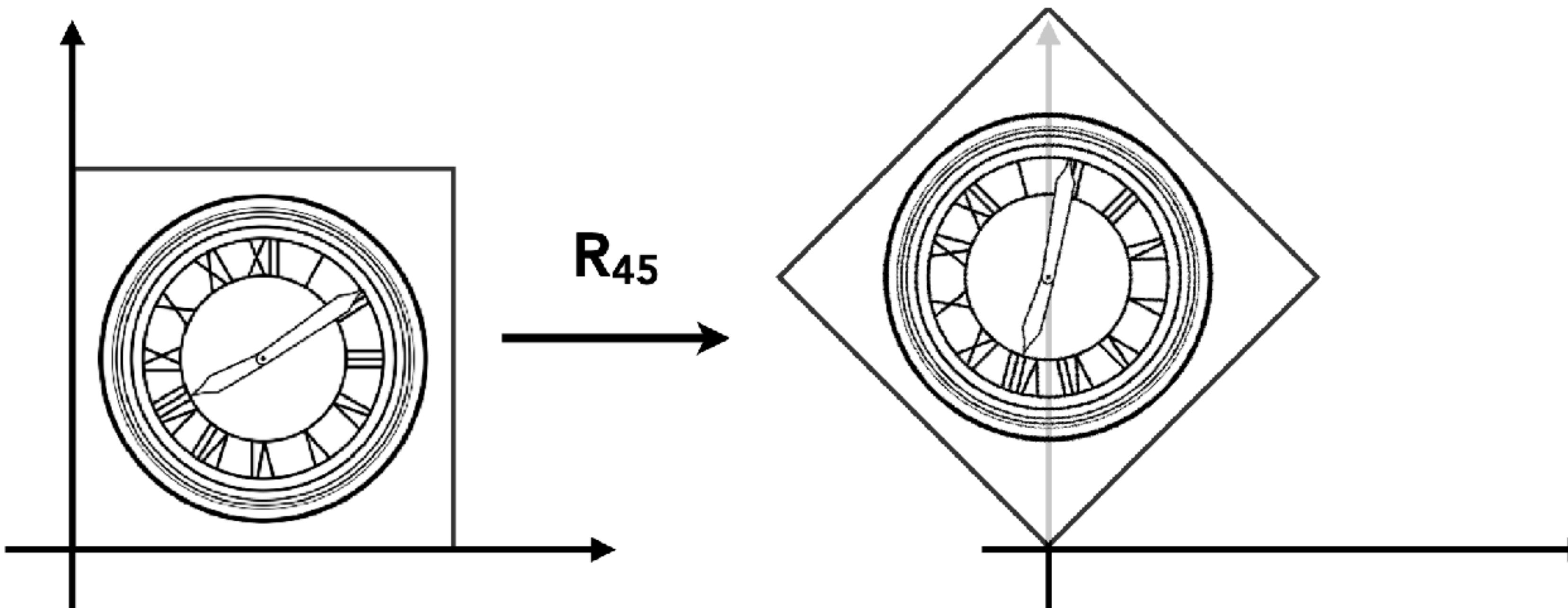
- Horizontal shift is 0 at $y=0$
- Horizontal shift is a at $y=1$
- Vertical shift is always 0

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

$$\begin{aligned} x' &= x + ay \\ y' &= y \end{aligned}$$

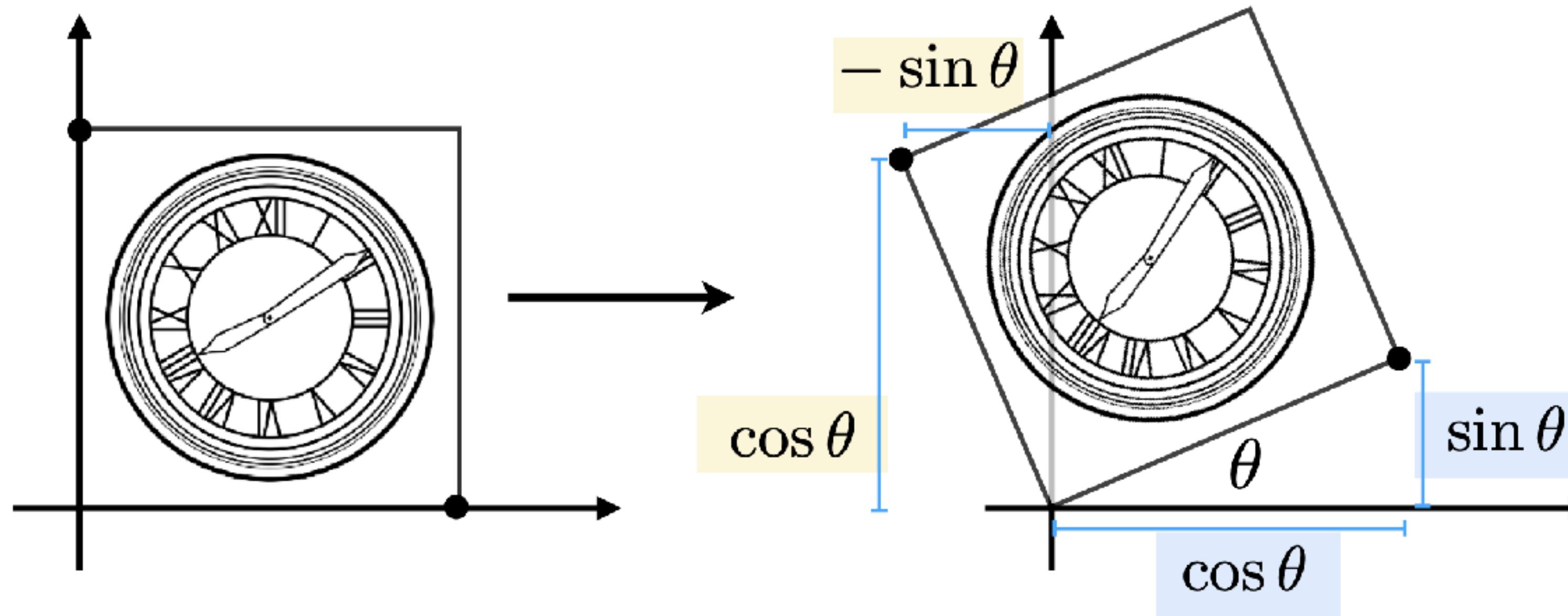
2D transformation

Rotate



2D transformation

Rotate



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

2D transformation

Linear Transformation = Matrices

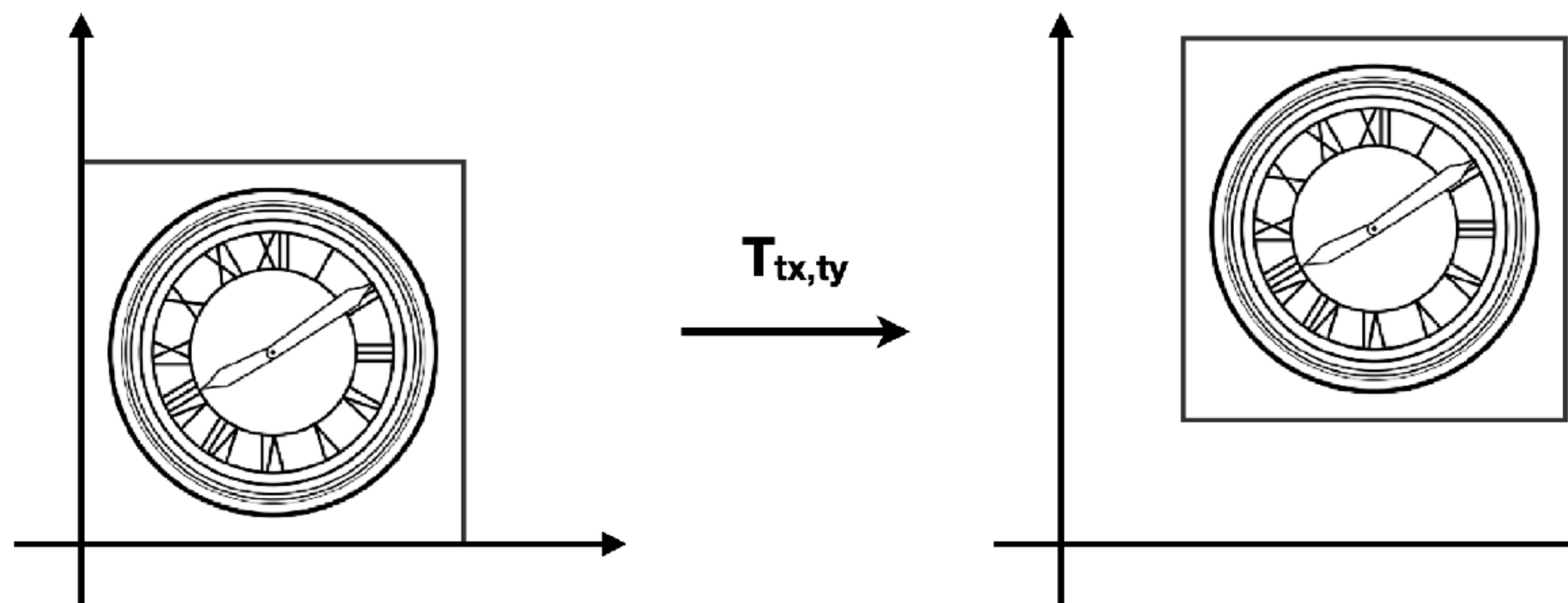
$$x' = a x + b y$$

$$y' = c x + d y$$

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

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

Translation

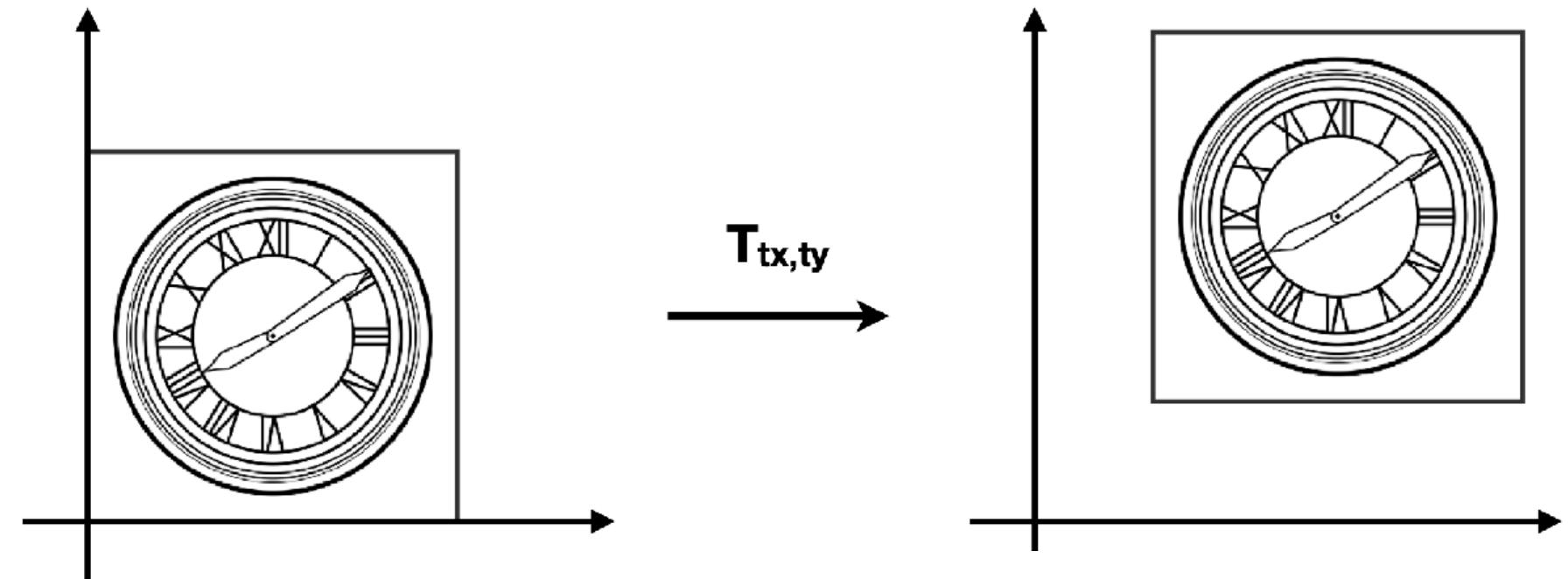


Homogeneous coordinates

Translation

- Translation cannot be represented in matrix form.
- So, translation is NOT a linear transform.

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



$$x' = x + t_x$$

$$y' = y + t_y$$

Homogeneous coordinates

Add a new coordinate (w-coordinate.)

- 2d point $(x, y, 1)^T$, 3d point $(x, y, z, 1)^T$.
- 2d vector $(x, y, 0)^T$, 3d vector $(x, y, z, 0)^T$.
- Matrix representation of translations

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

Homogeneous coordinates

Valid operation

- Valid if the w-coordinate of the result is either 0 or 1.
 - $\text{Vector}(0) + \text{Vector}(0) = \text{Vector}(0)$ VALID
 - $\text{Point}(1) + \text{Vector}(0) = \text{Point}(1)$ VALID
 - $\text{Point}(1) - \text{Point}(1) = \text{Vector}(0)$ VALID
 - $\text{Point}(1) + \text{Point}(1) = ??(\textcolor{red}{2})$ NOT VALID

Affine Transformation

- Affine map = linear map + translation

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

- Using homogenous coordinates

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

Affine Transformation

More 2D examples

Scale

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

Rotation

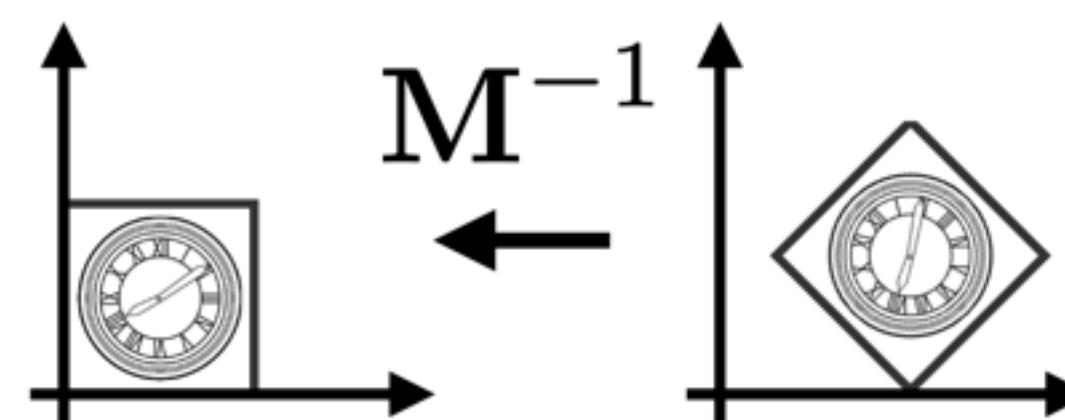
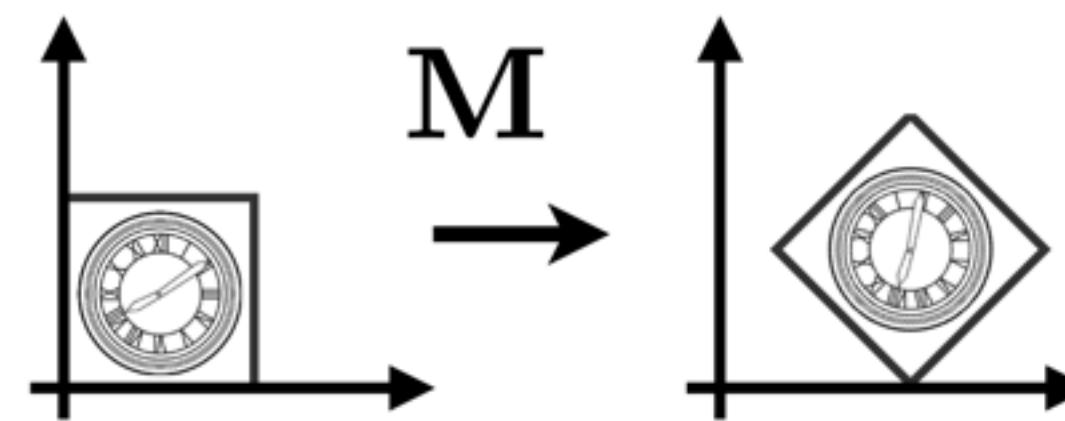
$$\mathbf{R}(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Translation

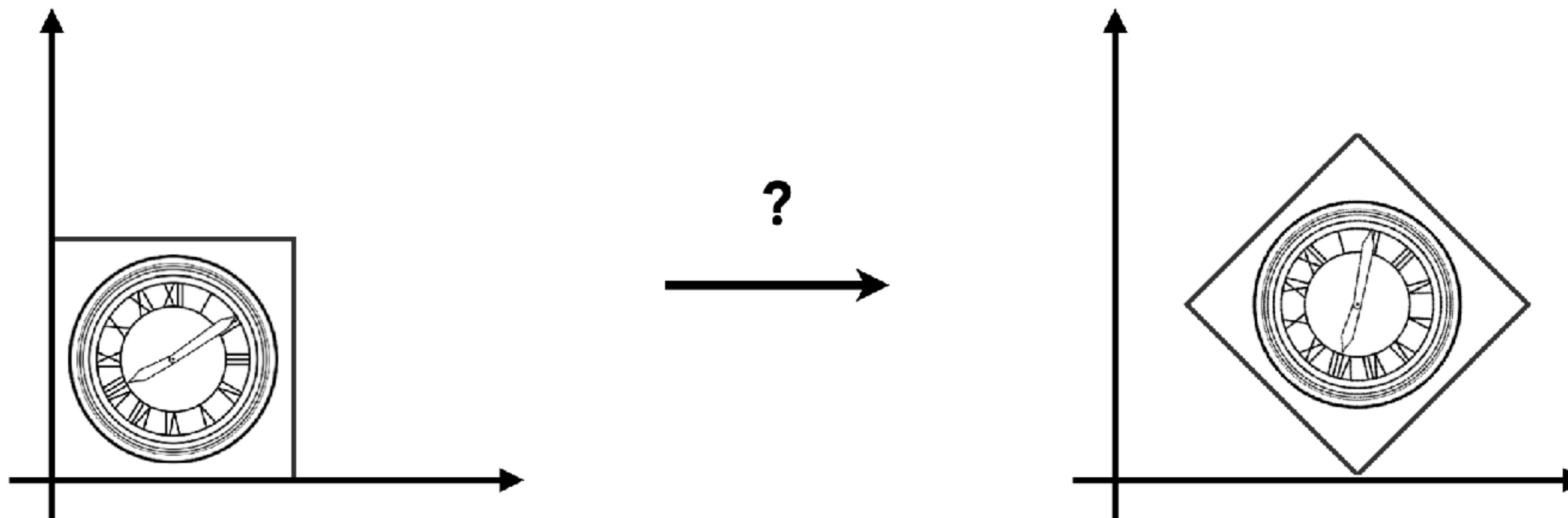
$$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Inverse Transformation

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

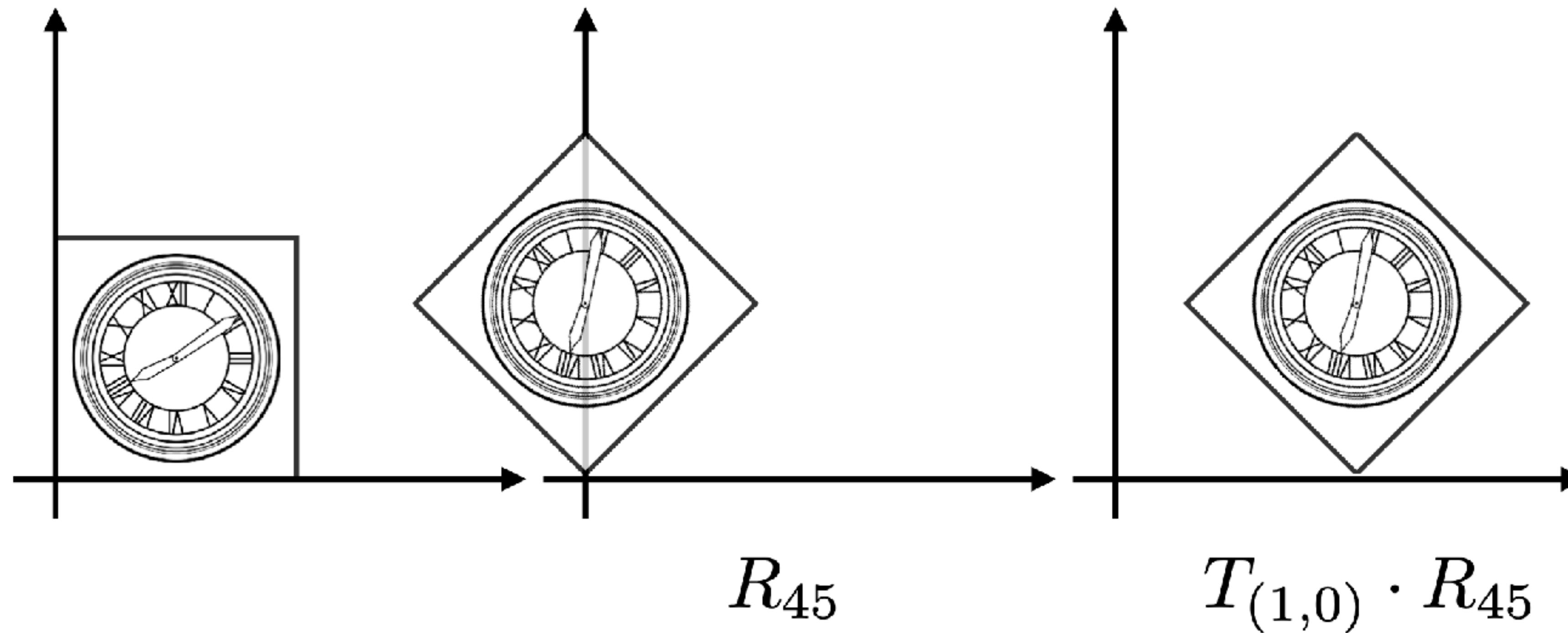


Composite Transform



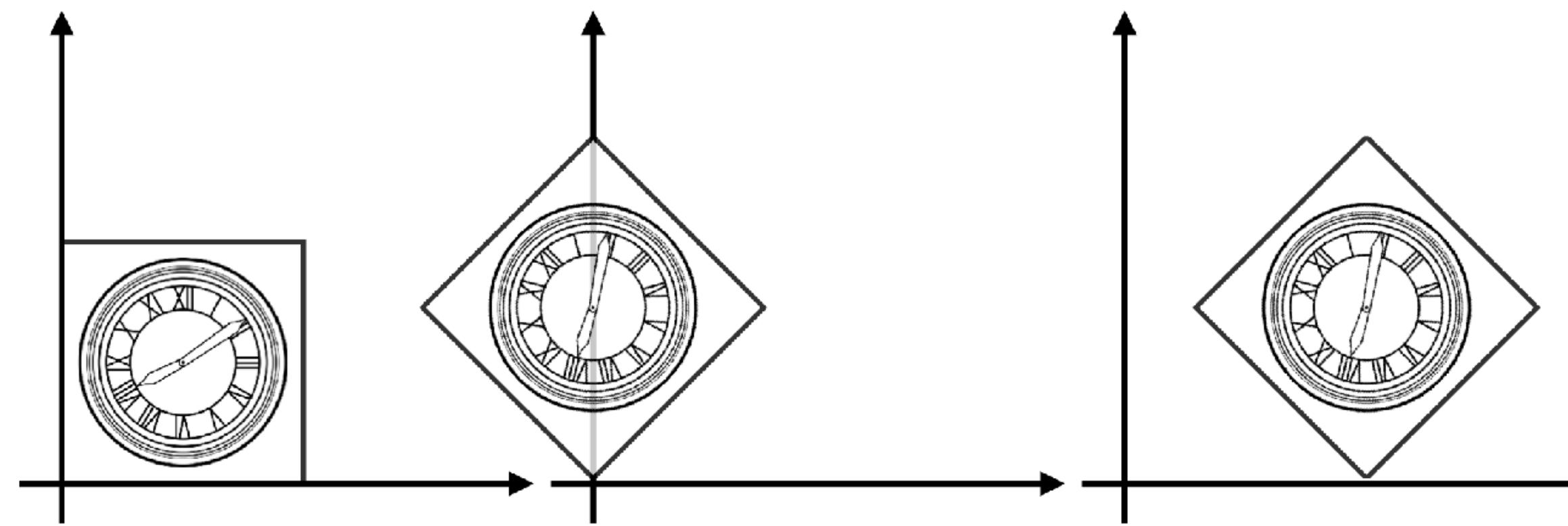
Composite Transform

Rotate Then Translate



Composite Transform

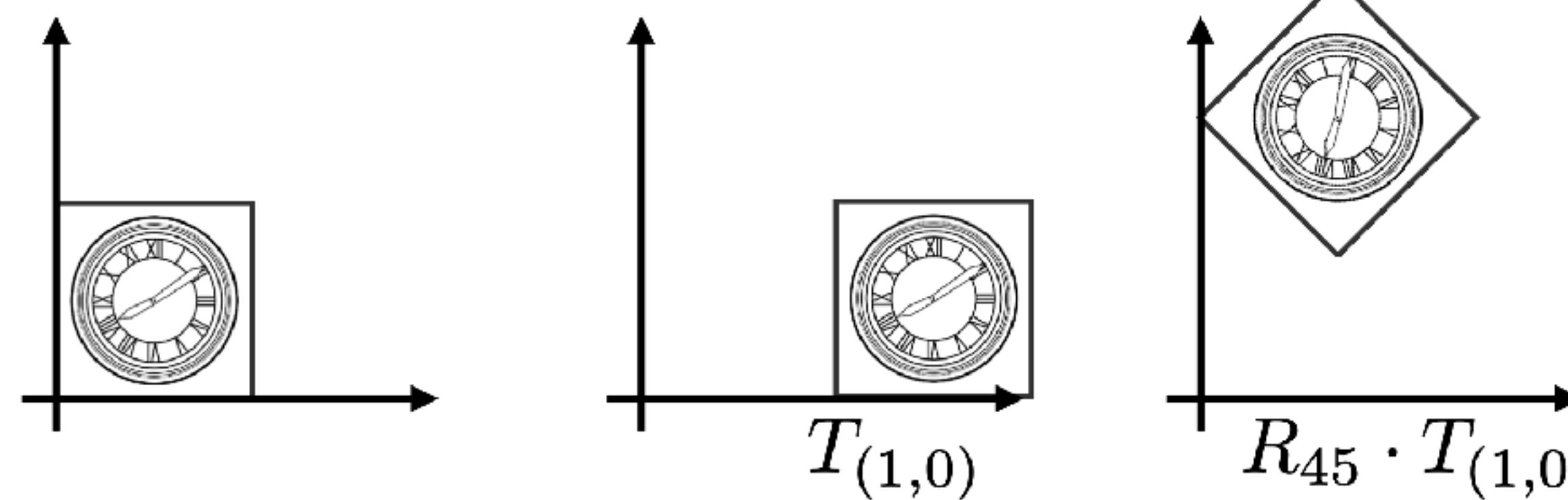
What about translate then rotate?



$$R_{45}$$

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

-



$$T_{(1,0)}$$

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

Composite Transform

Transform Ordering Matters

- Matrix multiplication is not commutative.

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

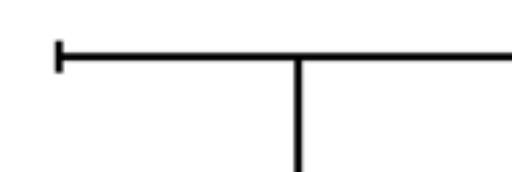
- Note that matrices are applied from right to left:

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

Composite Transform

- A Sequence of affine transforms $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\dots$

- Compose by matrix multiplication
 - Very important for performance!

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


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

Tutorial 1.2: Animation in Blender

<https://www.blender.org/download/>

Outline

A brief introduction to animation in Blender

- Data Representation
- Rigging
- Keyframe animation

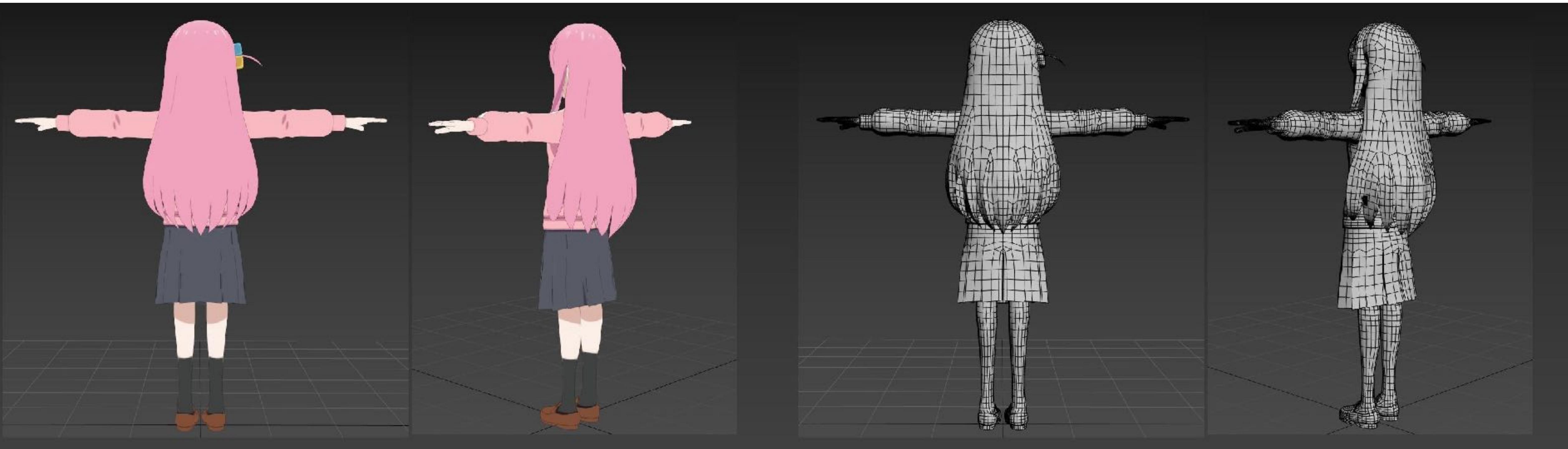
Mesh

Basic representation in digital computing

- String -> characters -> codes (ASCII)
- Image -> pixels -> 2d array / 3d array
- Voice -> sequence of electronic positions
- Human?

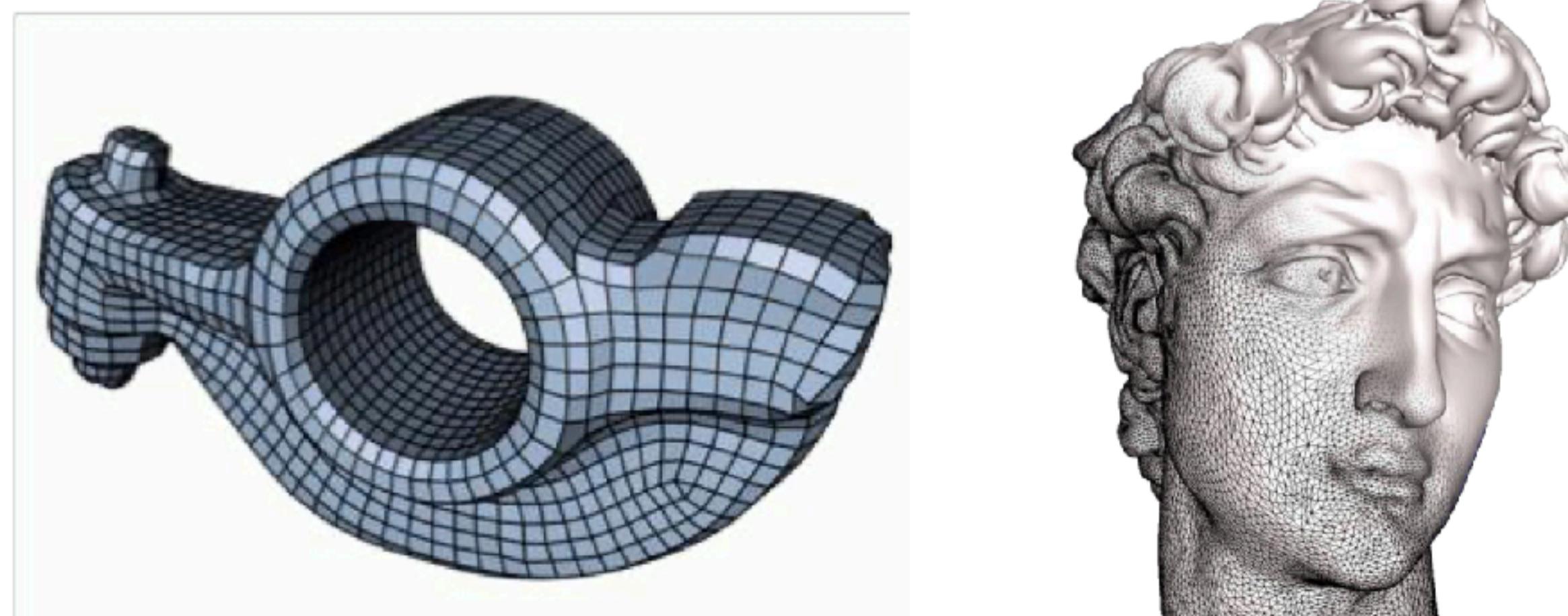
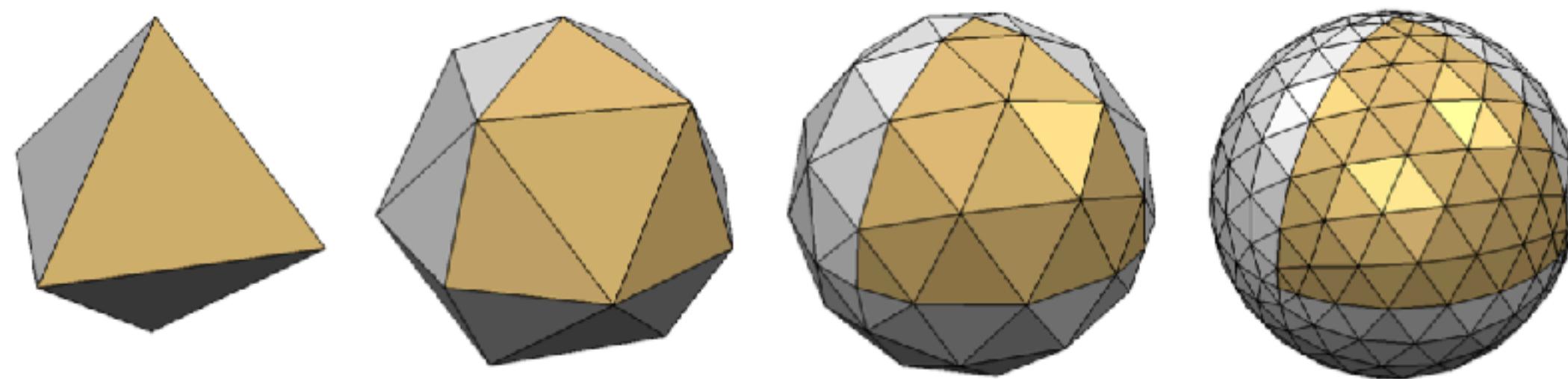


Mesh



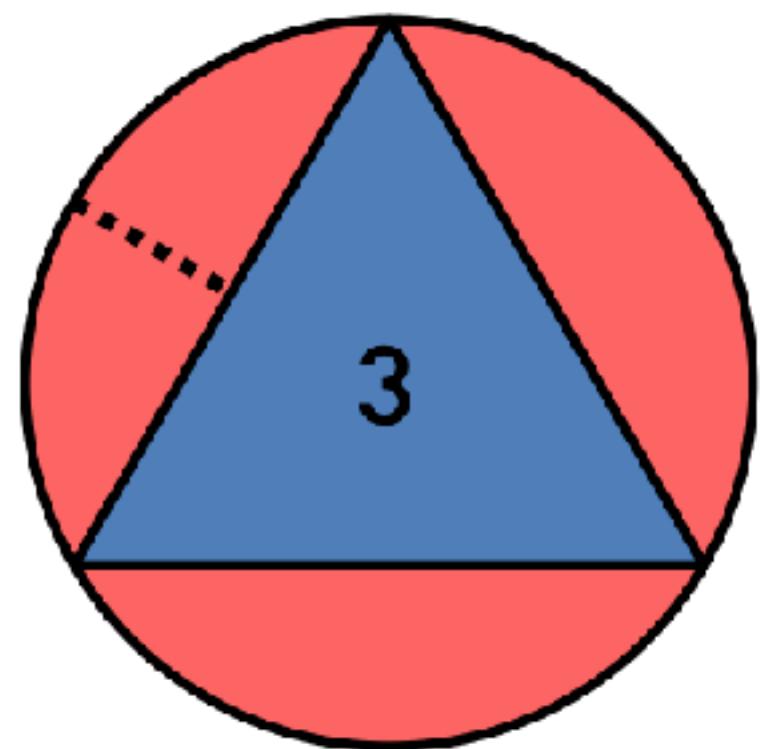
Mesh

- Boundary representations of objects

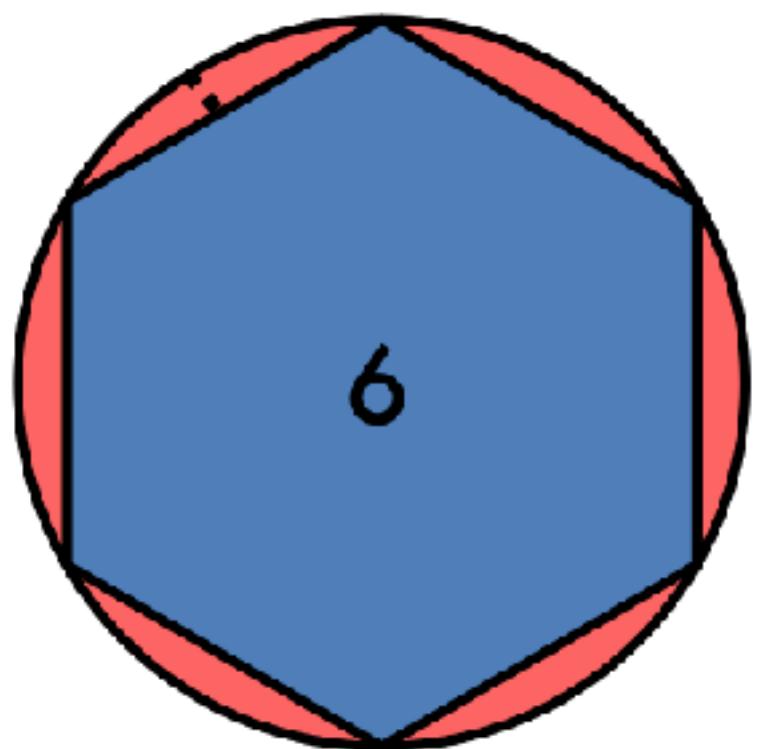


Mesh

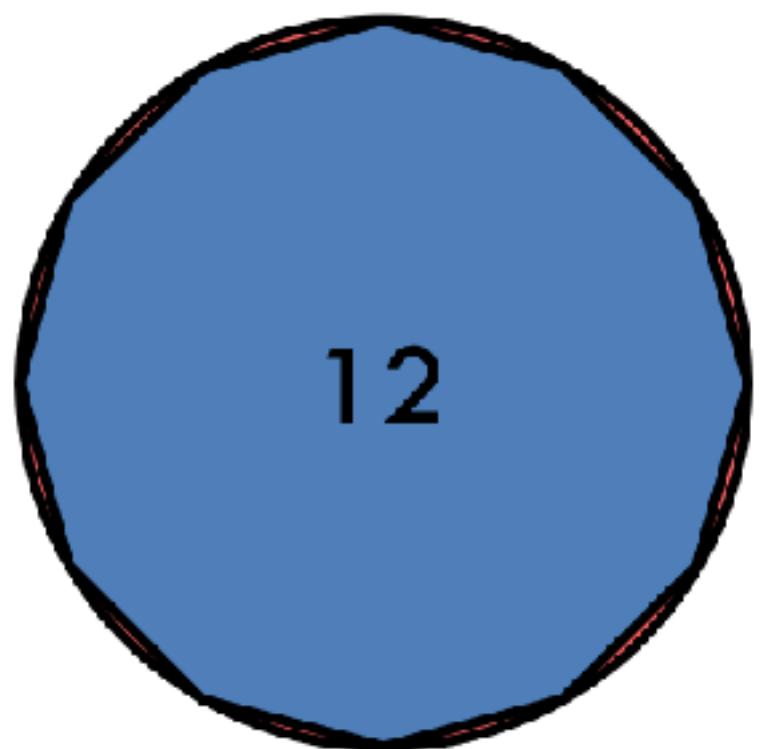
Piecewise linear approximation



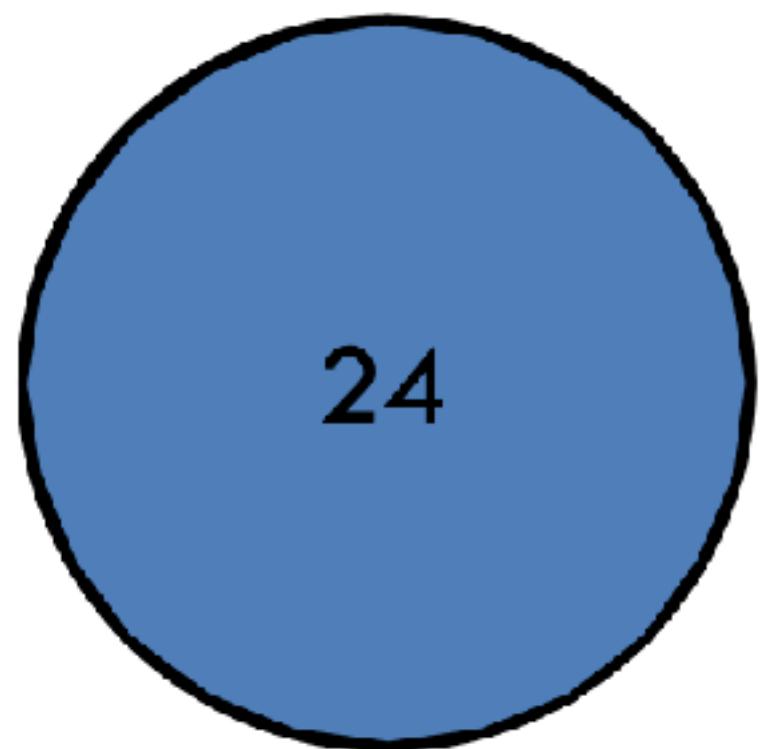
25%



6.5%



1.7%

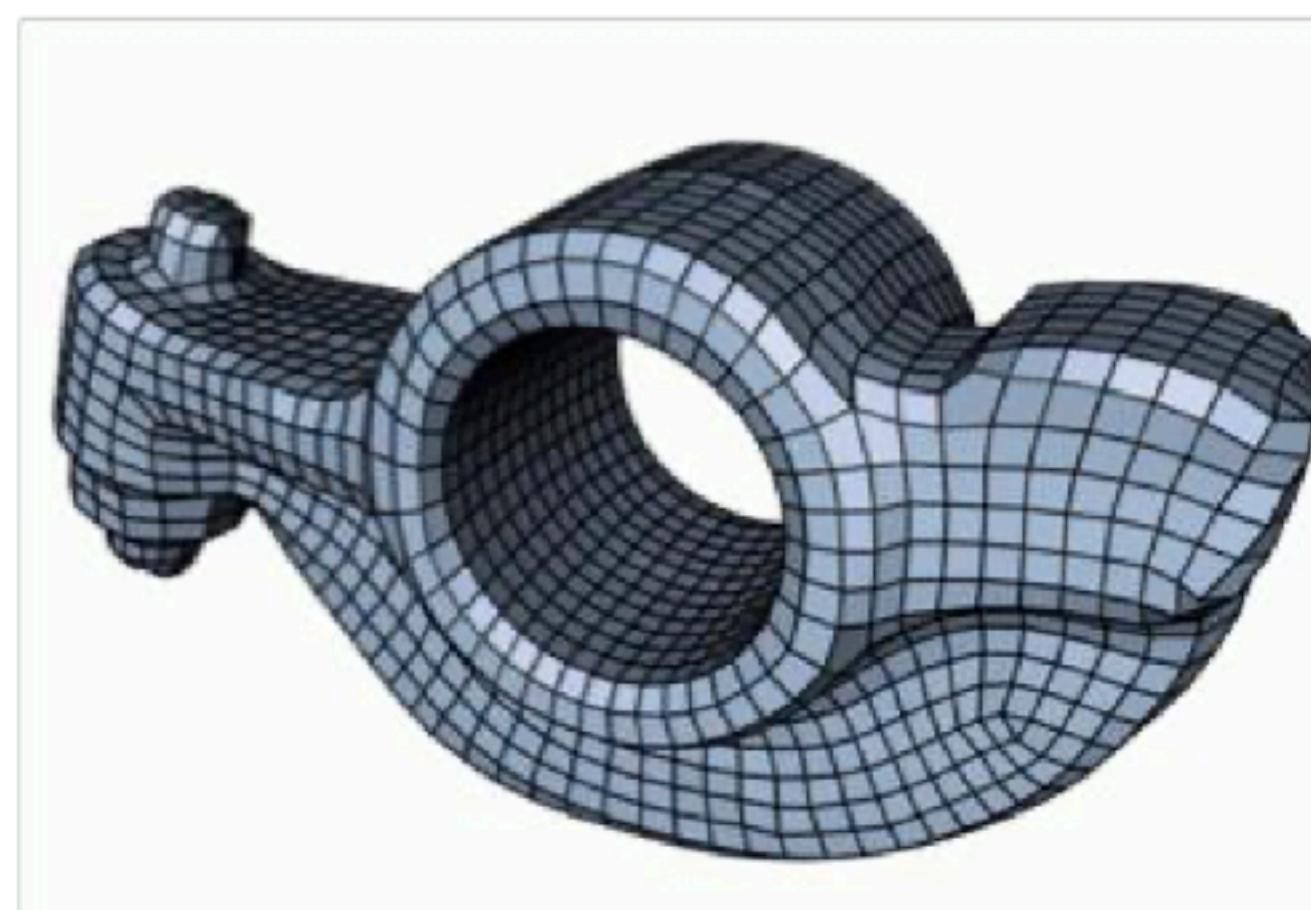
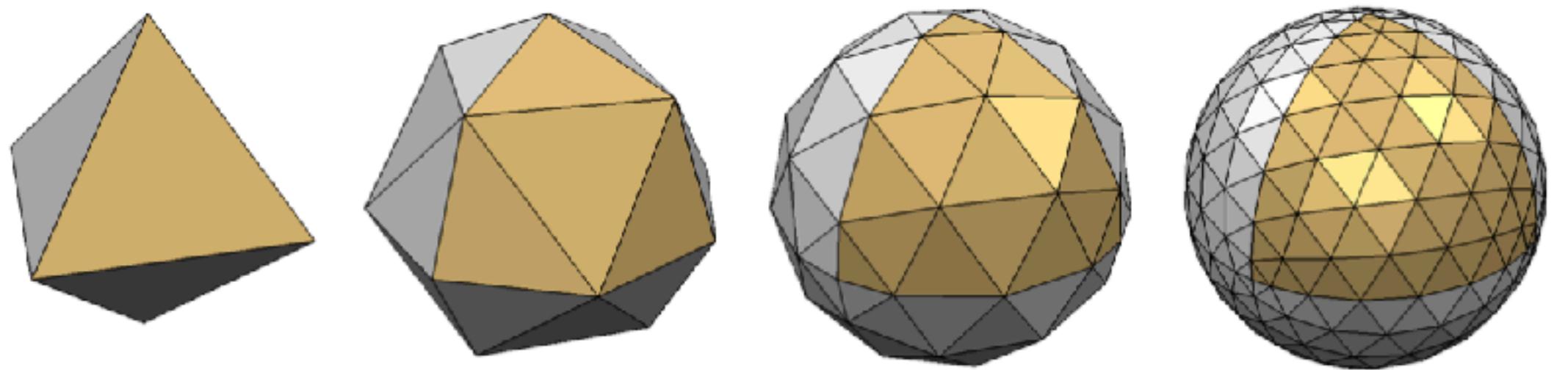


0.4%

Mesh

Polygonal Mesh

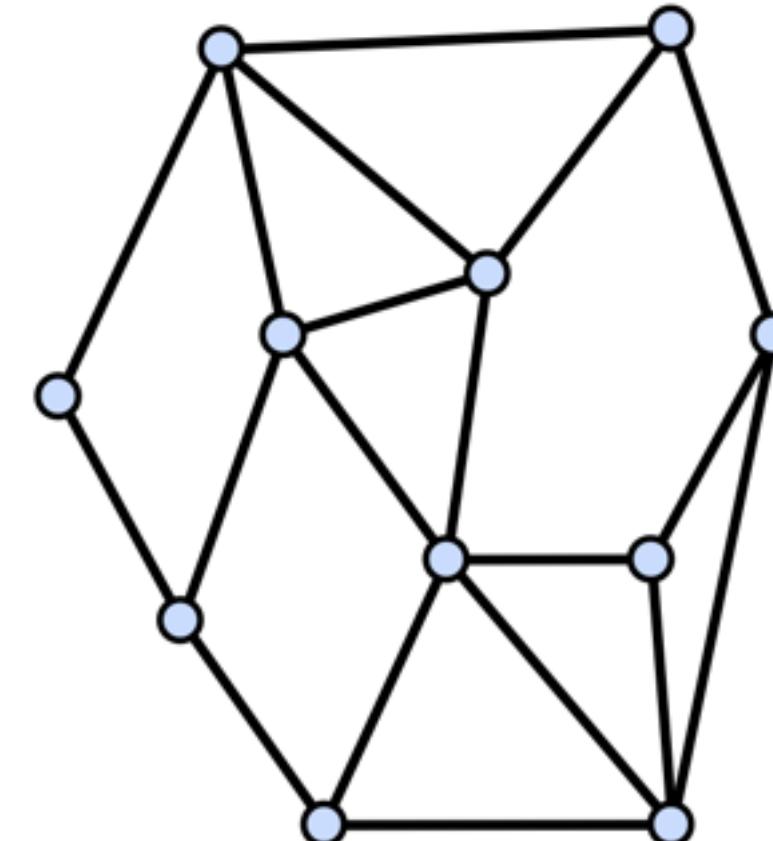
- Polygonal meshes are a good representation.
 - approximation $O(h^2)$
 - arbitrary topology
 - piecewise smooth surfaces
 - adaptive refinement
 - efficient rendering



Mesh

Polygonal Mesh

- A finite set M of closed, simple polygons is a polygonal mesh



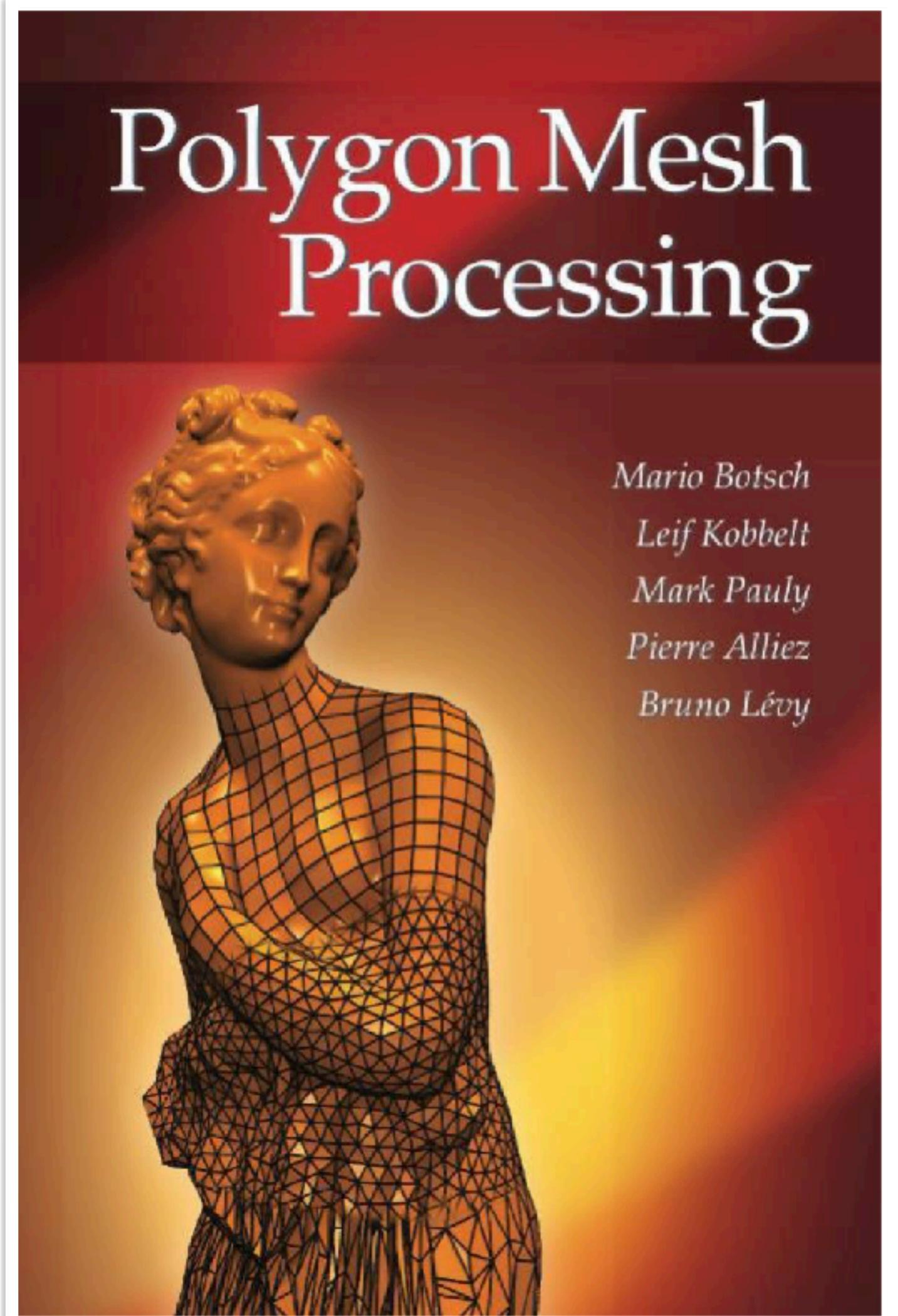
$$M = \langle V, E, F \rangle$$

vertices edges faces

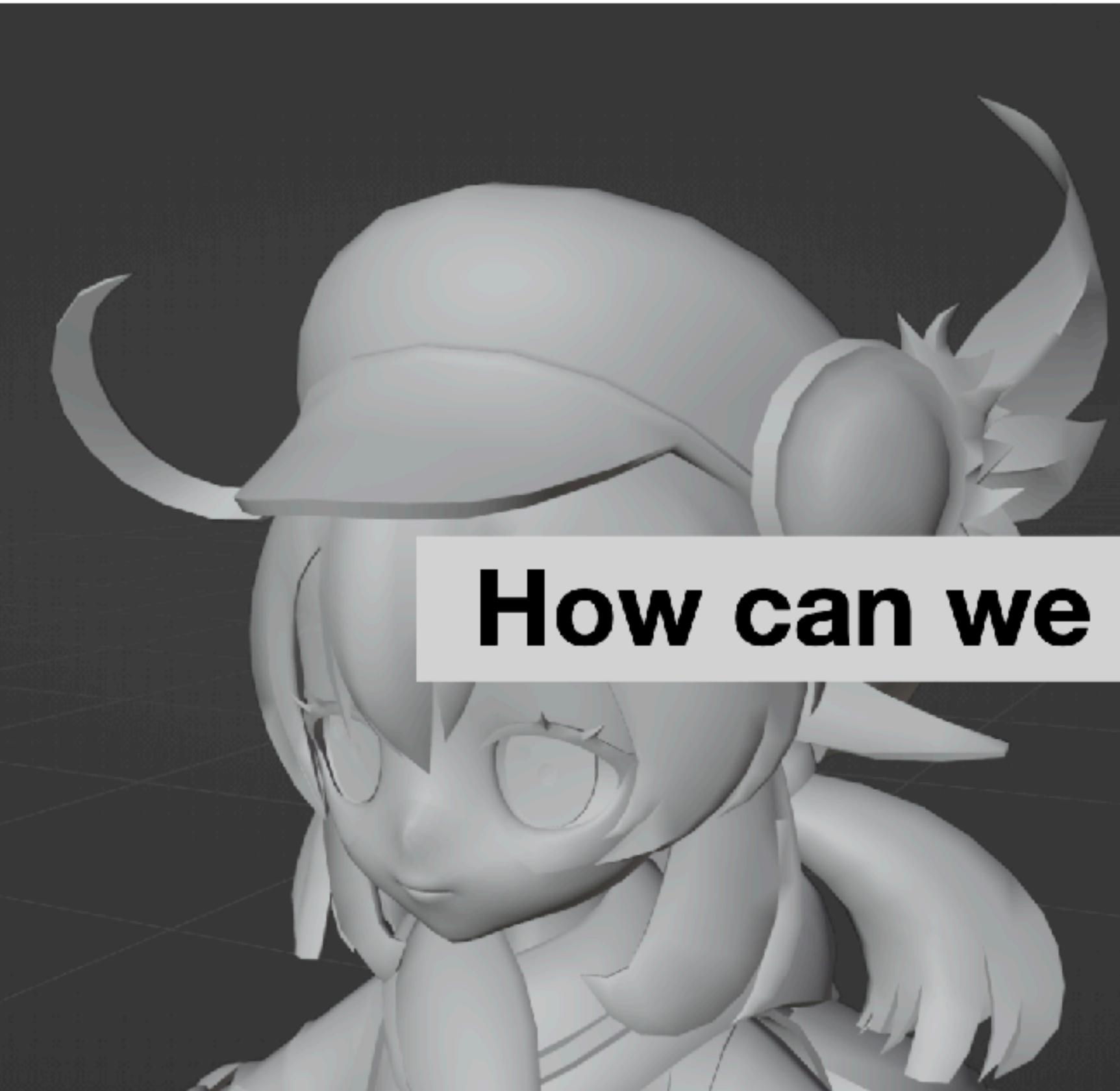
Arrows point from the labels "vertices", "edges", and "faces" to the respective components of the equation $M = \langle V, E, F \rangle$.

Reference Book

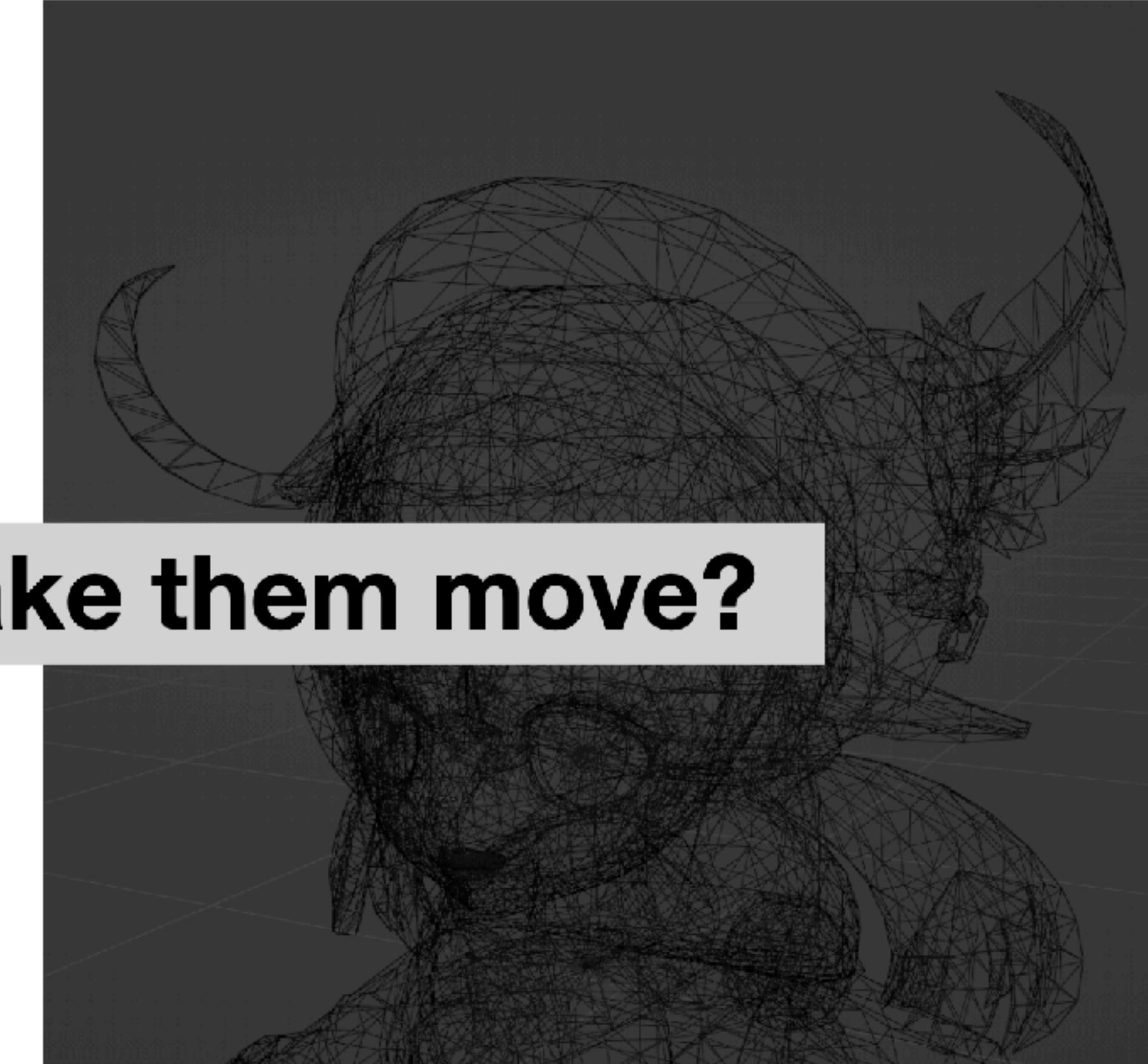
- Polygon Mesh Processing Book, Chapter 2



Skeleton



How can we make them move?



Character Mesh

Mesh (Vertices, edges and faces)

Control the key joints (skeleton)

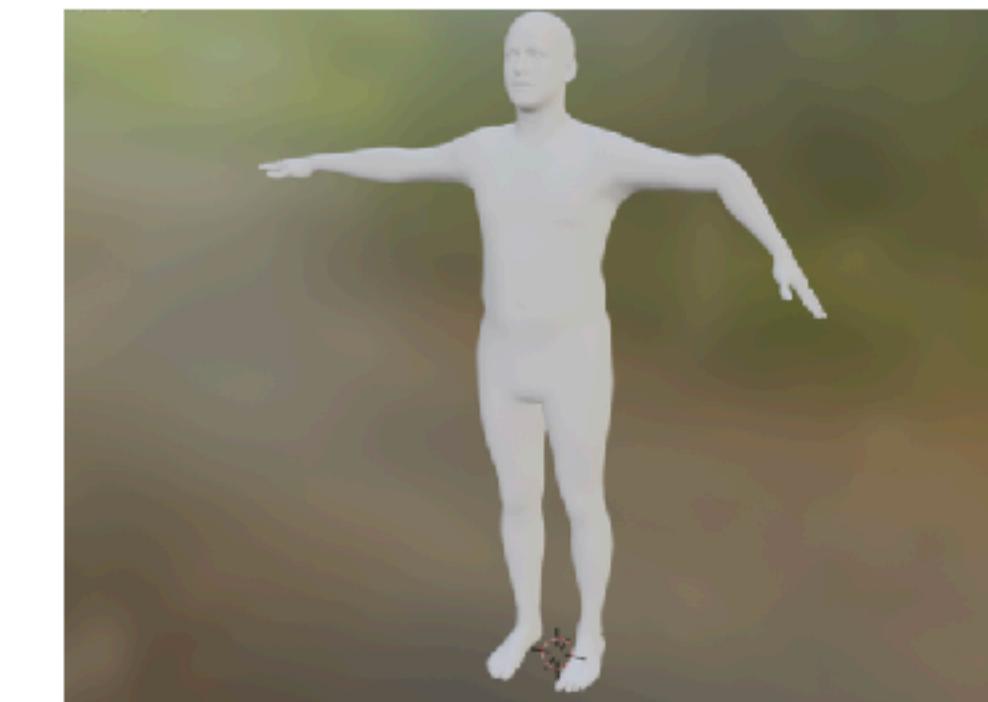


Animation Example

Skeleton
(Bones)



Mesh



Example: `skeleton_in_mesh.blend`

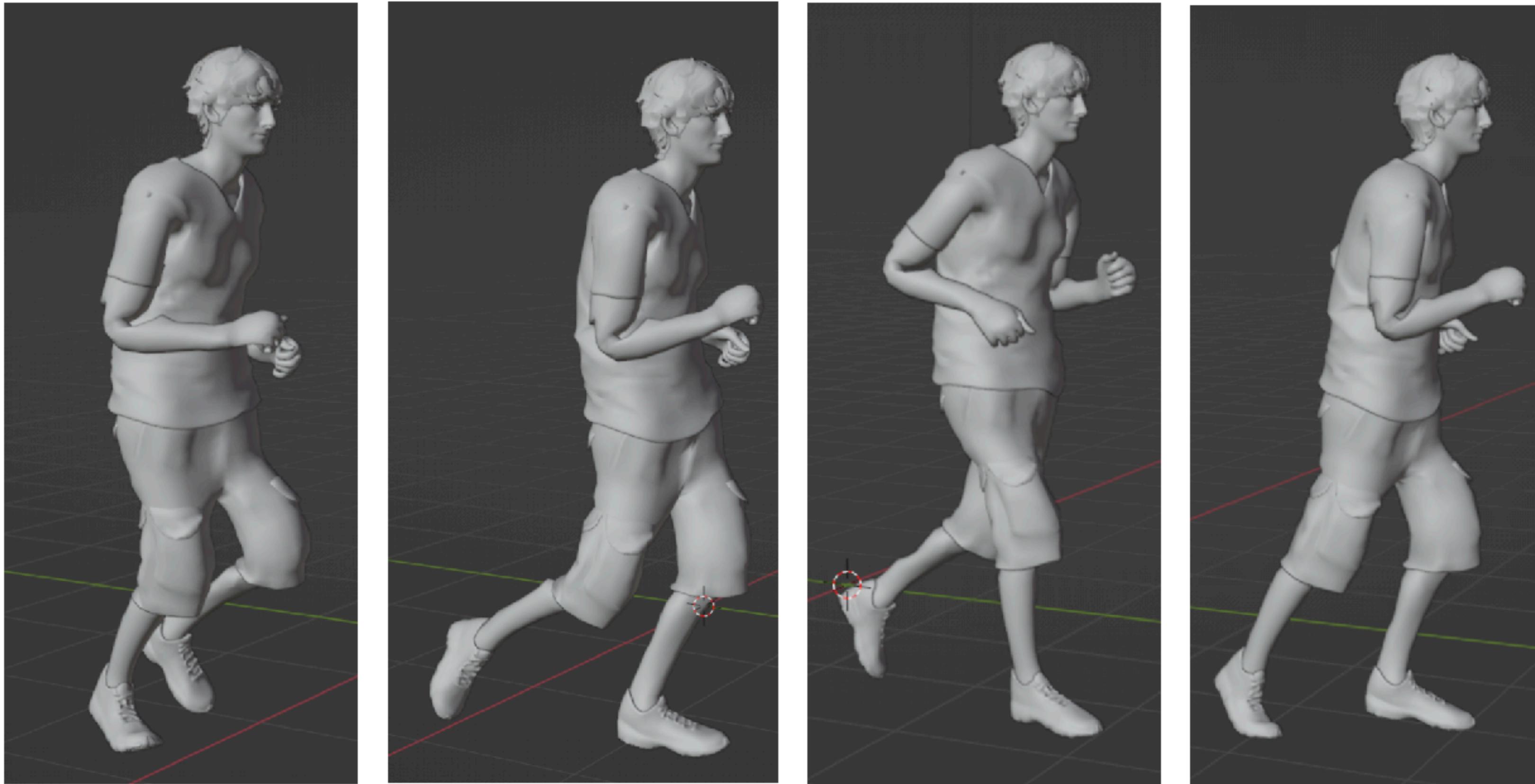
Conclusion

- A polygon mesh is used to present digital humans.
- A skeleton will be in the mesh to rig the mesh.
- A weight-binding method is used to manipulate the mesh with a few key joints.

Import Mesh -> Import Skeleton -> Bind(Rig) Mesh and Skeleton



Keyframe Animation



Example File: [keyframe_animation.blend](#)

Task1

40%

- Download Blender
- Import the provided mesh (feel free to use your mesh if you like)
- Define your key joints and skeleton
- Rigging/Skinning
- Design keyframes animation (feel free to make use of your creativity to add any objects you like)
- Render the video (make use of lights, camera)

Blender Tips

Keyframe Animation

- A new version will be welcomed (blender 3.4)
- It's free software, so please download the official version.
- Viewpoint Navigation - <https://www.youtube.com/watch?v=ILqOWe3zAbk>
- Add/delete Object - <https://www.youtube.com/watch?v=JSAobQPRlwC>
- Rigging - <https://www.youtube.com/watch?v=f2pTkW-1JkE>
- Keyframe Animation - <https://www.youtube.com/watch?v=SZJswvw9wEs>
- Google

Thank you