Lecture 2: Geometric Transformation

Yuhao Zhu
http://yuhaozhu.com

CSC 292/572, Fall 2020
Mobile Visual Computing

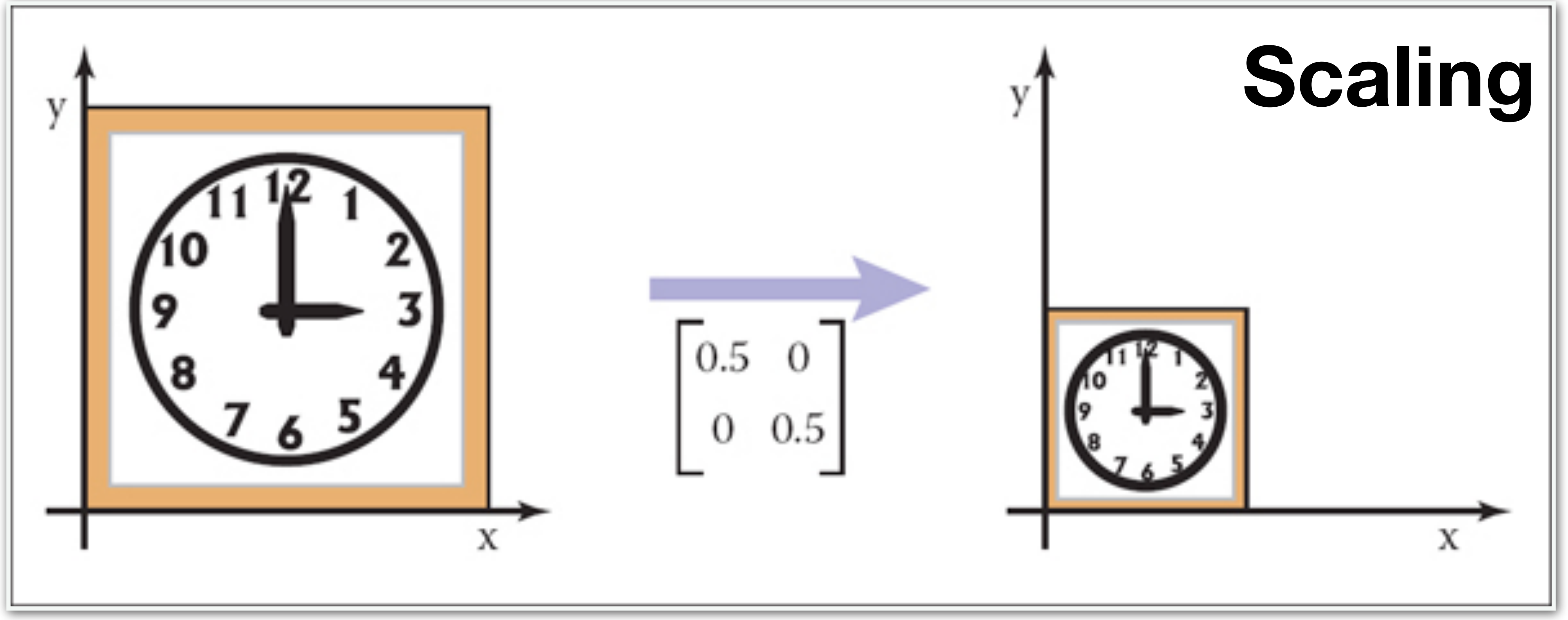
Logistics

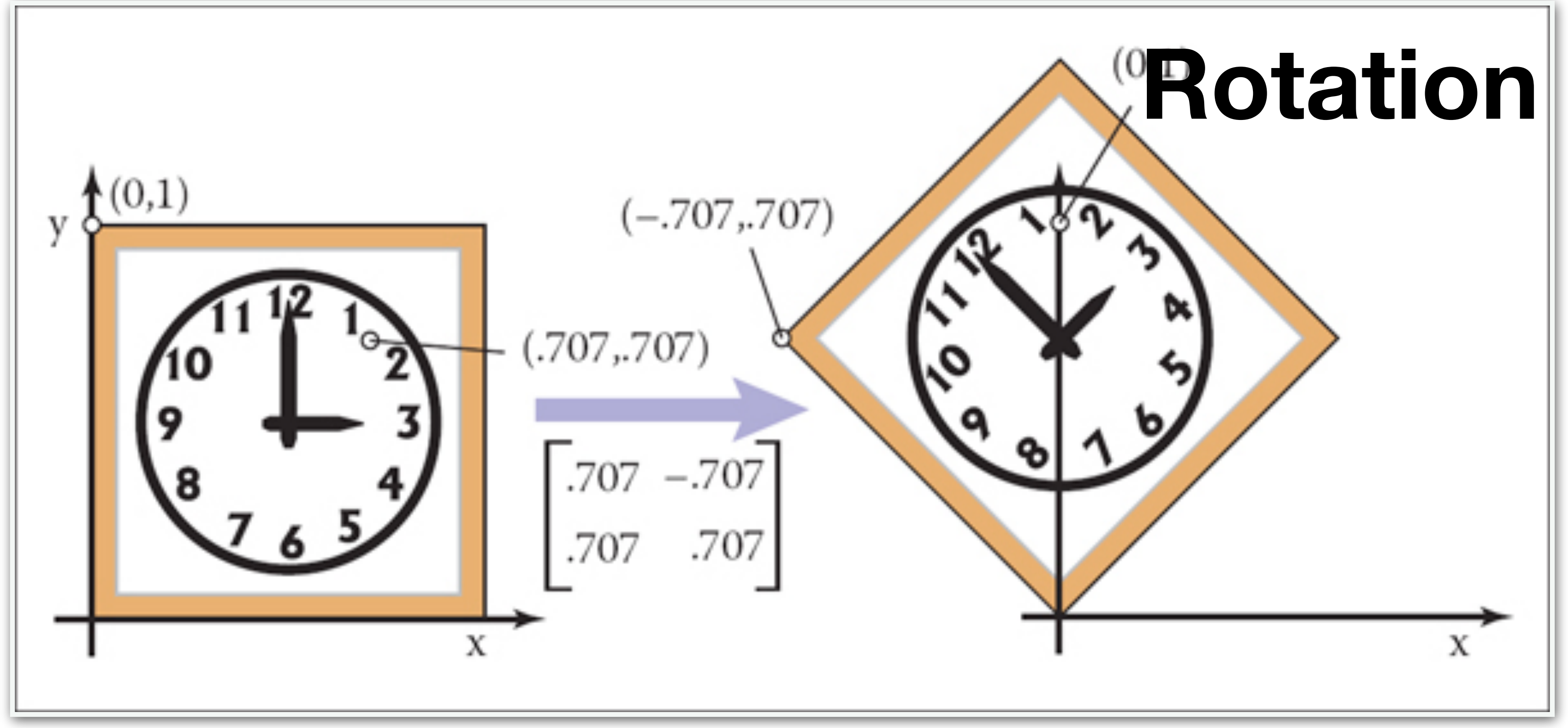
- Written assignment 0 is up and is due Sept. 2 11:30 AM.
- Course schedule: https://www.cs.rochester.edu/courses/572/fall2020/schedule.html. You will find reading assignments and slides.
- Don’t forget to sign up for the news flash presentation. Link sent in BB.
- Come and introduce yourself in the first 2 weeks during my office hour.
- Start thinking and talking to me about your final project idea.

Geometric Transformation

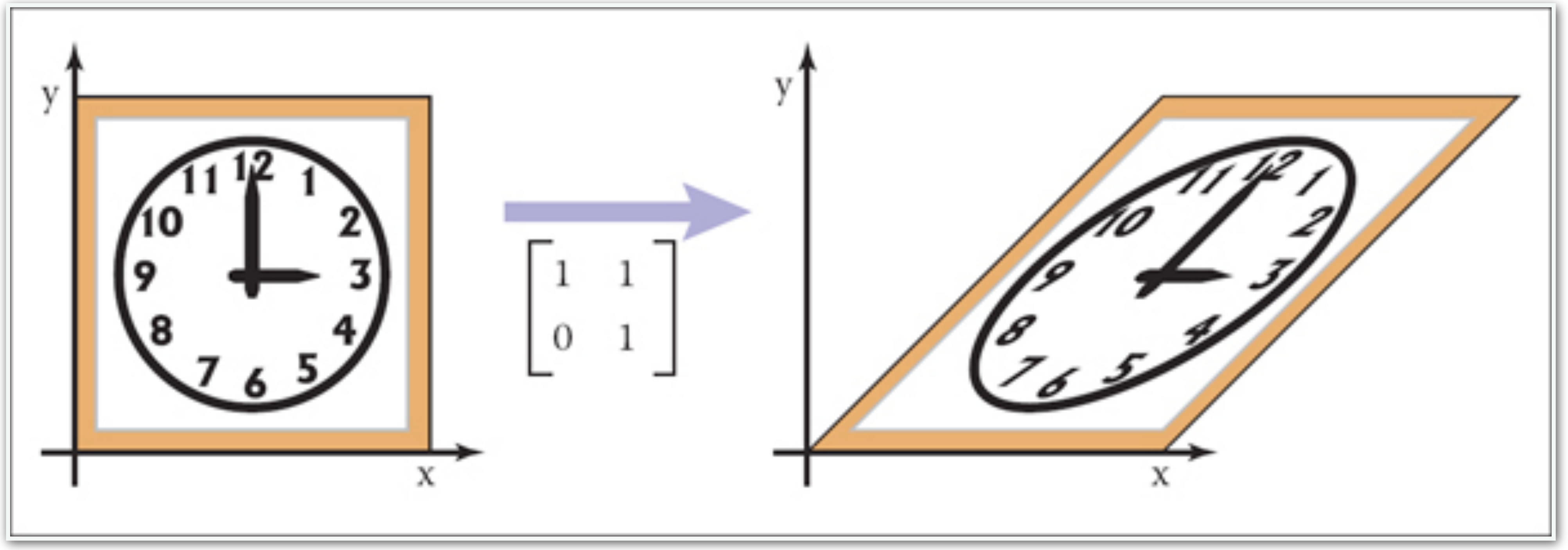
- Perhaps the single most important math for this course and for understanding all visual computing applications.
- This lecture is an introduction and overview of the concepts that will be used in later lectures, *not* a systematic treatment of geometric transformation.
- Assumes basic understanding of linear algebra. A nice and gentle introduction to linear algebra could be found in Chapter 5 of Fundamentals of Computer Graphics (log in using your UR NetID).
- Required reading: Chapter 6 of Fundamentals of Computer Graphics.

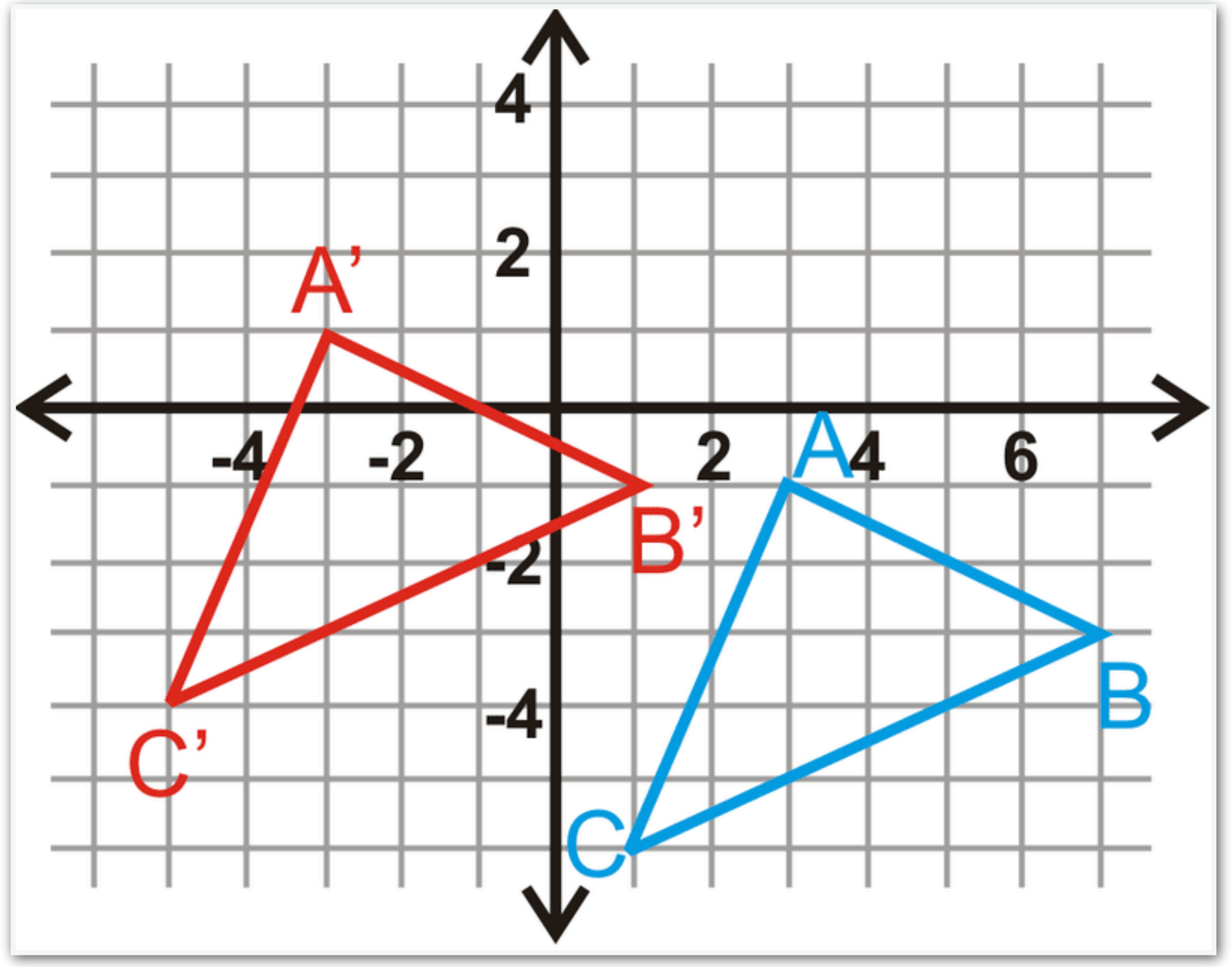
Geometric Transformation: What Is It?





Shearing





Translation

There are many more: reflection, dilation, projection, etc.

Geometric Transformation: What Is It?

- Change the position of ***all the points*** in the space ***in the same way***.
- A point P [x, y, z]. Think of it as a 1x3 matrix
- **Transformation**: change P [x, y, z] to P' [x', y', z']
- Mathematically, transforming P to P' is multiplying P with a 3x3 matrix T.
- Different transformations require different matrices.

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Point Transformation using Matrix Multiplication

- What should T be like if we want to keep x the same before and after the transformation — regardless of where P [x, y, z] is.

$$\mathbf{x' = xT_{00} + yT_{10} + zT_{20} = x,}\ \textit{for } \forall\ x, y, z$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} \boxed{\begin{matrix} T_{00}, \\ T_{10}, \\ T_{20}, \end{matrix}} & \begin{matrix} T_{01}, & T_{02} \\ T_{11}, & T_{12} \\ T_{21}, & T_{22} \end{matrix} \end{bmatrix} \quad = \quad [x', y', z']$$

Point Transformation using Matrix Multiplication

- What should T be like if we want to keep x the same before and after the transformation — regardless of where P [x, y, z] is.

$$x' = x\underset{\substack{\uparrow \\ 1}}{T_{00}} + y\underset{\substack{\uparrow \\ 0}}{T_{10}} + z\underset{\substack{\uparrow \\ 0}}{T_{20}} = x, \textit{ for } \forall\, x, y, z$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} \boxed{\begin{matrix} 1 \\ 0 \\ 0 \end{matrix}} & \begin{matrix} T_{01}, T_{02} \\ T_{11}, T_{12} \\ T_{21}, T_{22} \end{matrix} \end{bmatrix} \quad = \quad [x', y', z']$$

Point Transformation using Matrix Multiplication

- What should T be like if we want to keep y the same before and after the transformation — regardless of where P [x, y, z] is.

$$y' = x\underset{0}{\underset{\uparrow}{T_{01}}} + y\underset{1}{\underset{\uparrow}{T_{11}}} + z\underset{0}{\underset{\uparrow}{T_{21}}} = y, \textit{ for } \forall\, x, y, z$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, & 0 & T_{02} \\ T_{10}, & 1 & T_{12} \\ T_{20}, & 0 & T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Point Transformation using Matrix Multiplication

- What should T be like if we want to keep z the same before and after the transformation — regardless of where P [x, y, z] is.

$$z' = x\underset{\uparrow \atop 0}{T_{02}} + y\underset{\uparrow \atop 0}{T_{12}} + z\underset{\uparrow \atop 1}{T_{22}} = z, \textit{ for } \forall\, x, y, z$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, & T_{01}, & 0 \\ T_{10}, & T_{11}, & 0 \\ T_{20}, & T_{21}, & 1 \end{bmatrix} \quad = \quad [x', y', z']$$

Identity Matrix

- What should T be like if we want to keep a point unchanged before and after the transformation — regardless of where P [x, y, z] is?
- That matrix is called the identity matrix

$$[x, y, z] \quad \times \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad = \quad [x', y', z']$$

Scaling

- Changing from P [x, y, z] to P' [$S_0 \cdot x$, $S_1 \cdot y$, $S_2 \cdot z$]
- The "scaling factor": [S_0, S_1, S_2]
- How should the transformation matrix look like?

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Scaling

- Changing from P [x, y, z] to P' [$S_0 \cdot x$, $S_1 \cdot y$, $S_2 \cdot z$]
- The "scaling factor": [S_0, S_1, S_2]
- How should the transformation matrix look like?

$$\mathbf{x' = xT_{00} + yT_{10} + zT_{20} = S_0 x}$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Scaling

- Changing from P [x, y, z] to P' [$S_0 \cdot x$, $S_1 \cdot y$, $S_2 \cdot z$]
- The "scaling factor": [S_0, S_1, S_2]
- How should the transformation matrix look like?

$$x' = xT_{00} + yT_{10} + zT_{20} = S_0x$$

$$[x, y, z] \quad x \quad \begin{bmatrix} S_0 & T_{01}, T_{02} \\ 0 & T_{11}, T_{12} \\ 0 & T_{21}, T_{22} \end{bmatrix} = [x', y', z']$$

Scaling

- Changing from P [x, y, z] to P' [$S_0 \cdot x$, $S_1 \cdot y$, $S_2 \cdot z$]
- The "scaling factor": [S_0, S_1, S_2]
- How should the transformation matrix look like?

$$x' = xT_{00} + yT_{10} + zT_{20} = S_0 x$$

$$[x, y, z] \quad x \quad \begin{bmatrix} S_0 & 0 & T_{02} \\ 0 & S_1 & T_{12} \\ 0 & 0 & T_{22} \end{bmatrix} = [x', y', z']$$

Scaling

- Changing from P [x, y, z] to P' [$S_0 \cdot x$, $S_1 \cdot y$, $S_2 \cdot z$]
- The "scaling factor": [S_0, S_1, S_2]
- How should the transformation matrix look like?

$$x' = xT_{00} + yT_{10} + zT_{20} = S_0 x$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} S_0 & 0 & 0 \\ 0 & S_1 & 0 \\ 0 & 0 & S_2 \end{bmatrix} = [x', y', z']$$

Scaling

- Changing from P [x, y, z] to P' [$S_0 \cdot x$, $S_1 \cdot y$, $S_2 \cdot z$]
- The "scaling factor": [S_0, S_1, S_2]
- How should the transformation matrix look like?

$$x' = xT_{00} + yT_{10} + zT_{20} = S_0 x$$

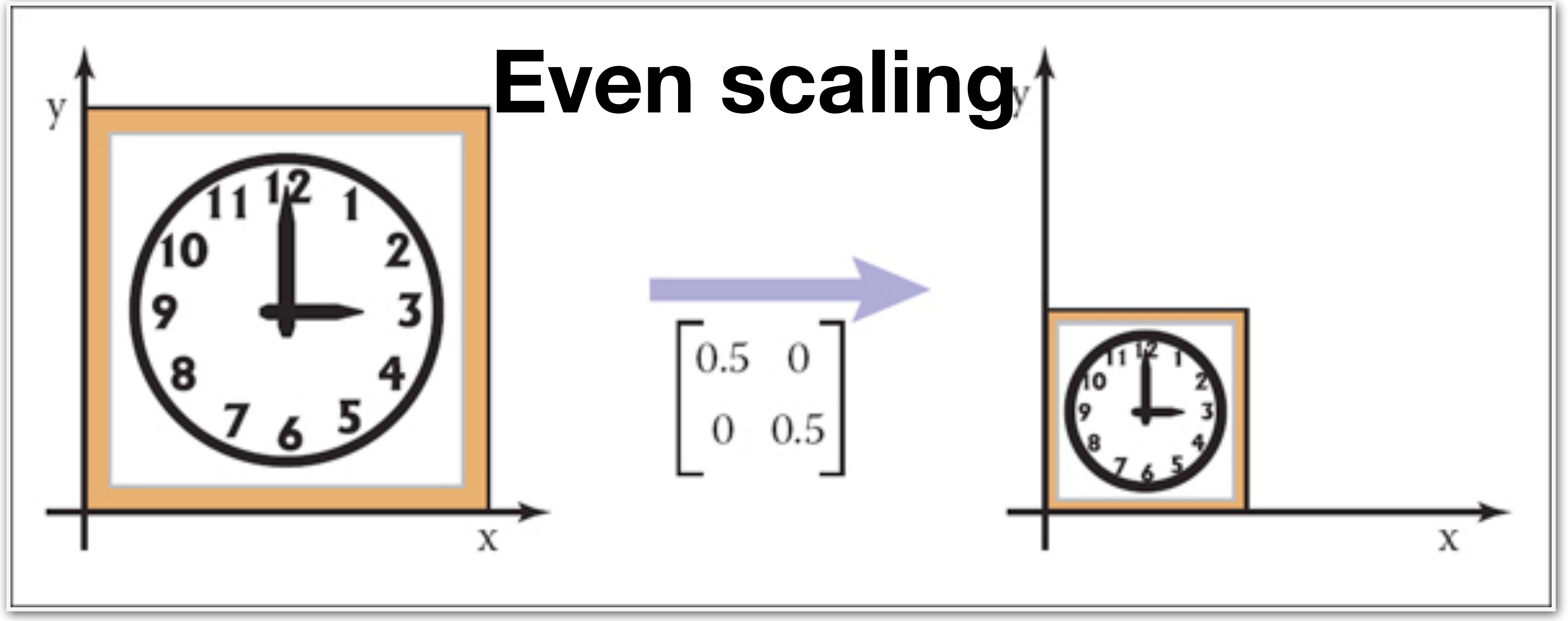
Scaling matrix is a diagonal matrix

$$[x, y, z] \times \begin{bmatrix} S_0 & 0 & 0 \\ 0 & S_1 & 0 \\ 0 & 0 & S_2 \end{bmatrix} = [x', y', z']$$

Scaling

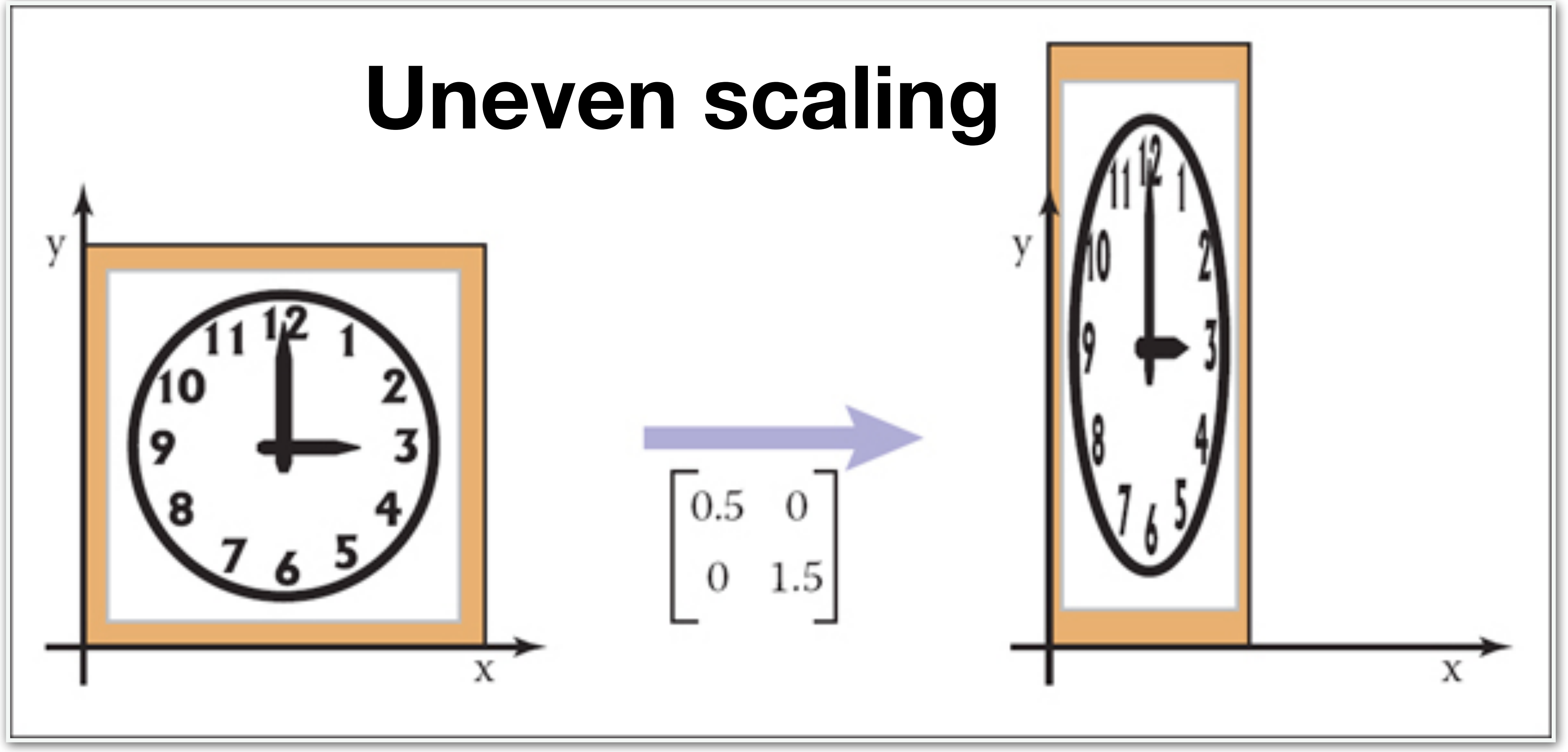
$$[x, y] \times \begin{bmatrix} 0.5, 0 \\ 0, 0.5 \end{bmatrix} = [x', y']$$

$$x' = 0.5x$$
$$y' = 0.5y$$



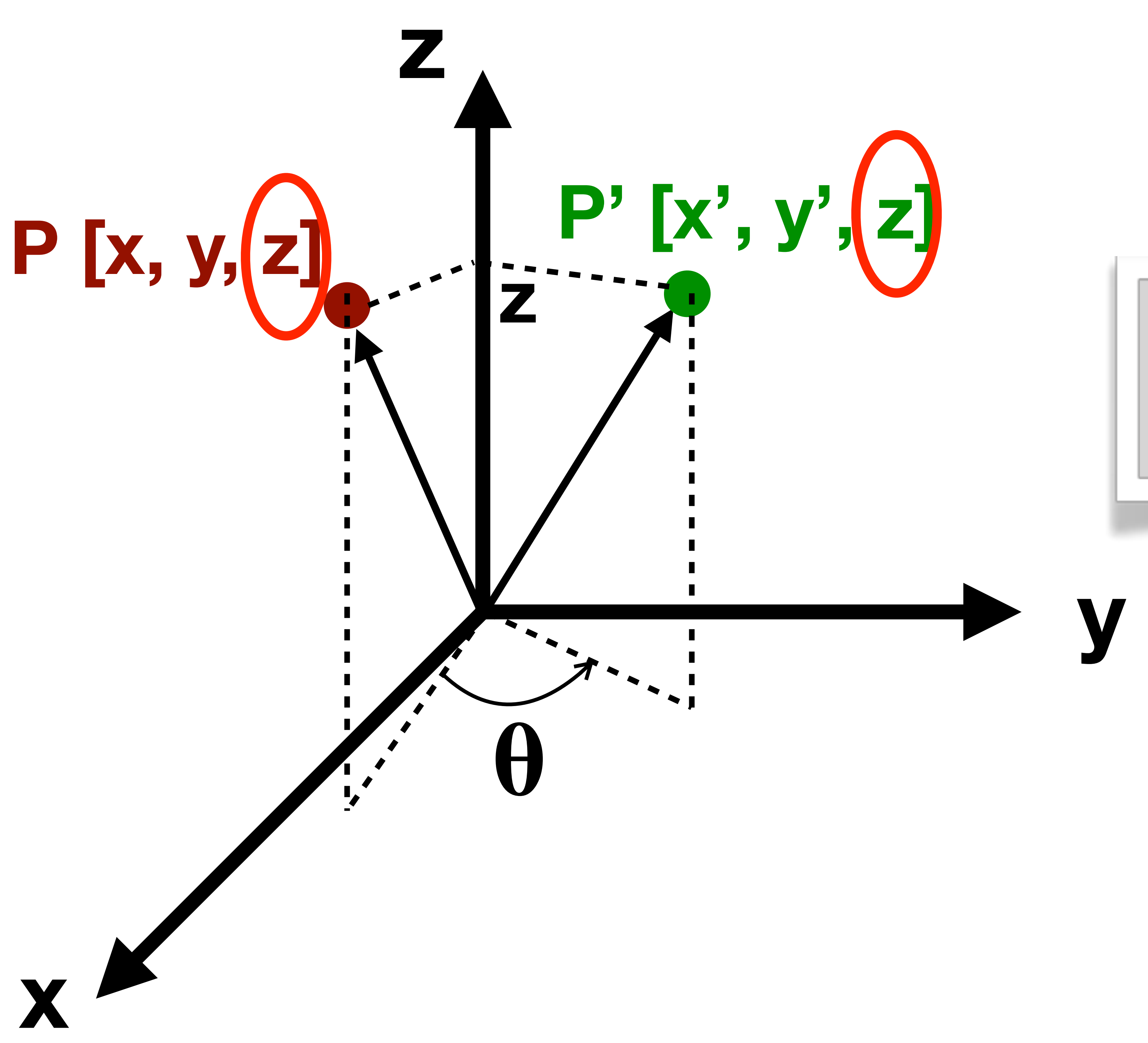
$$[x, y] \times \begin{bmatrix} 0.5, 0 \\ 0, 1.5 \end{bmatrix} = [x', y']$$

$$x' = 0.5x$$
$$y' = 1.5y$$

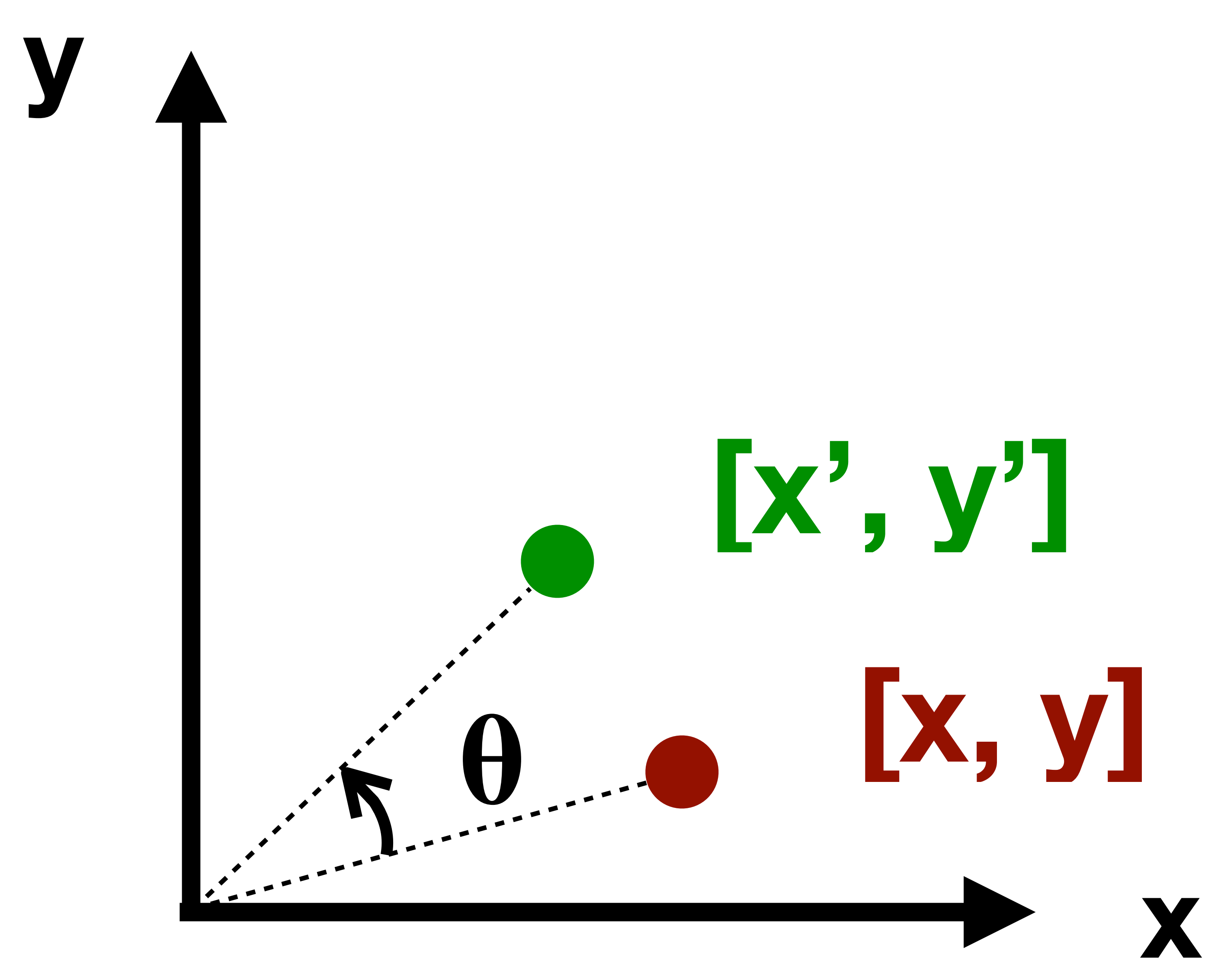


Rotation

- What should the transformation matrix be to rotate P around the z-axis by θ, regardless what P is?



Keep z the same, rotate within the x-y plane

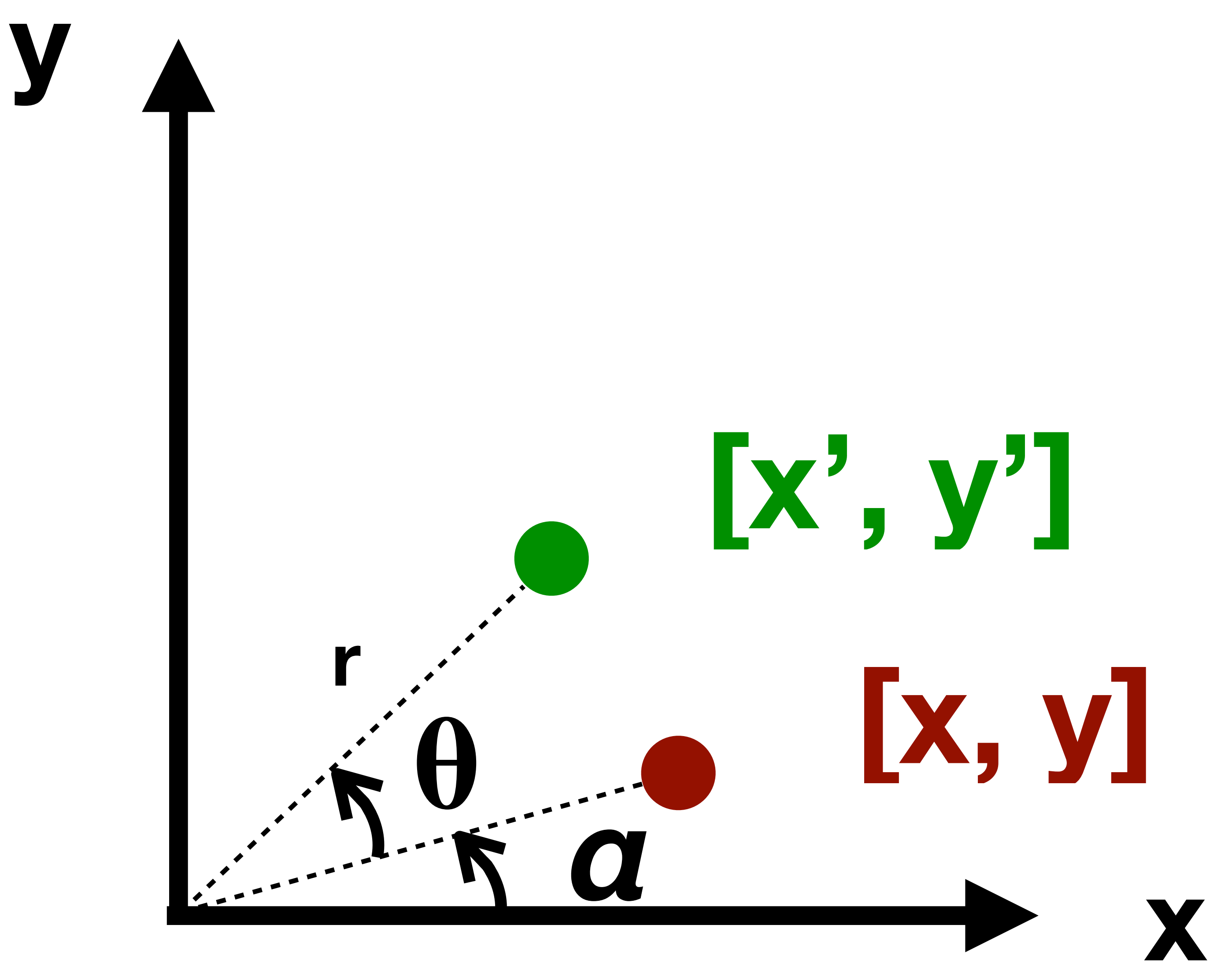


Rotation

Rotation

$\sin \alpha = y / r$

$\cos \alpha = x / r$

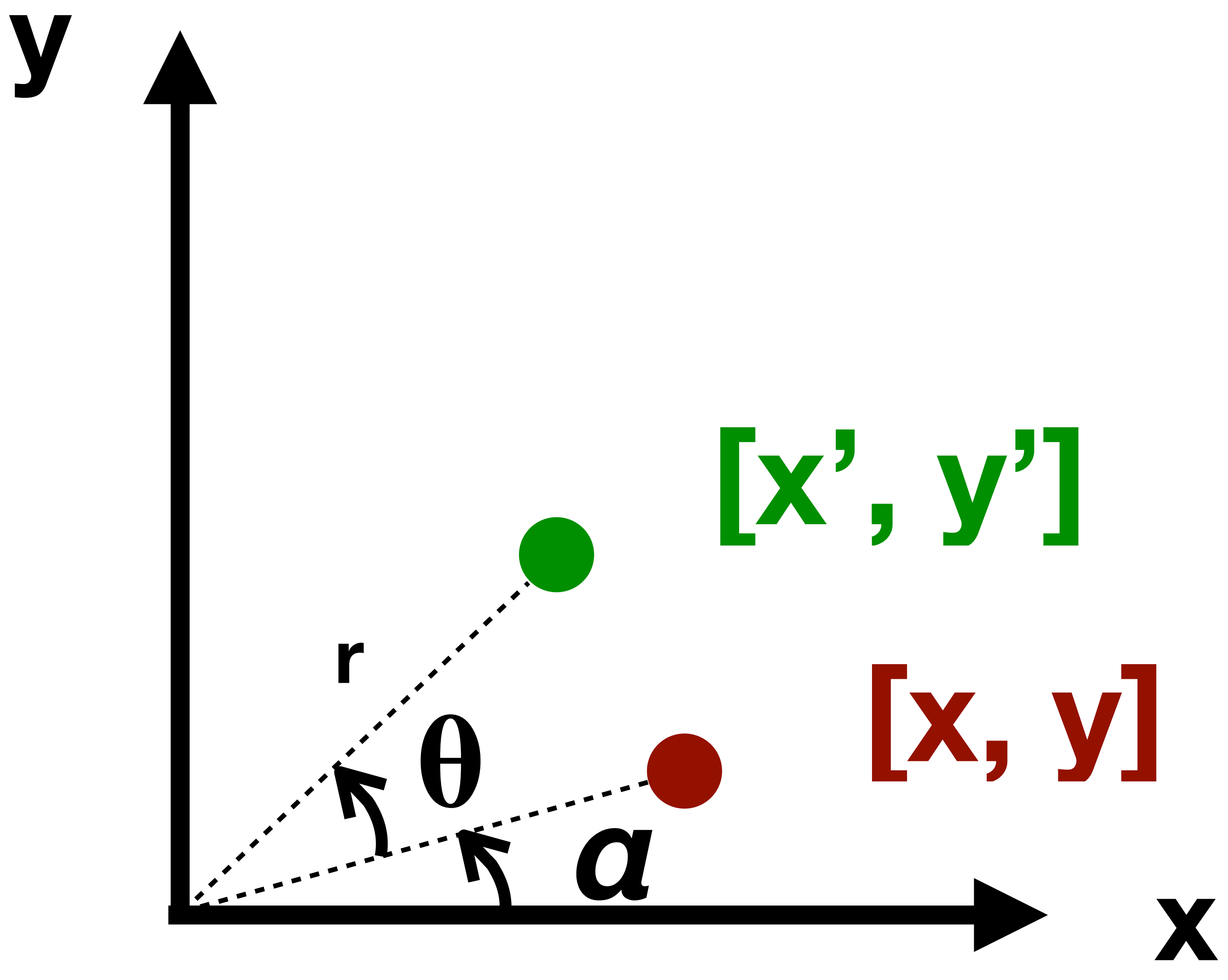


Rotation

Rotation

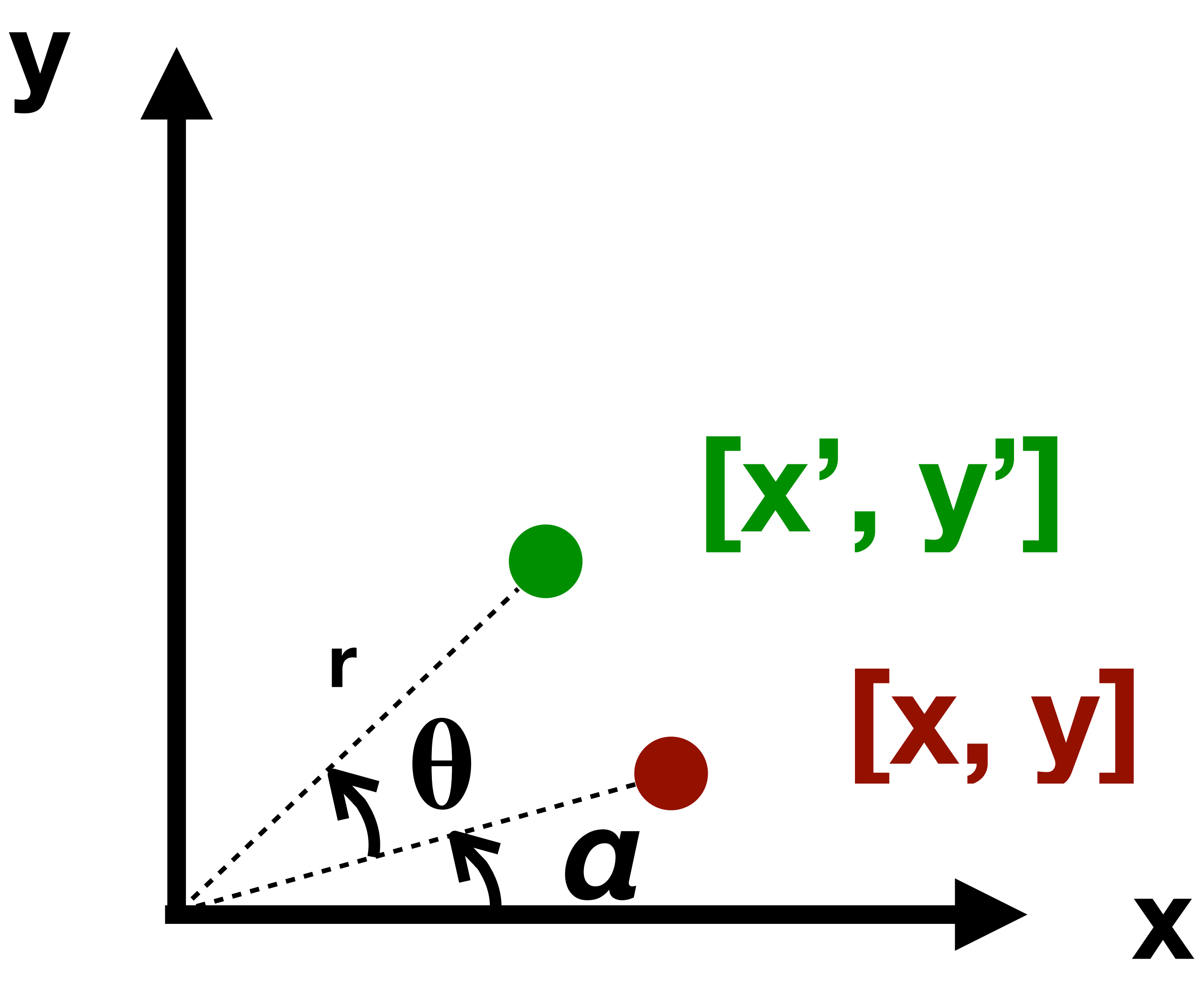
$\sin \alpha = y / r$
$\cos \alpha = x / r$

$\sin(\alpha + \theta) = y' / r = \sin \alpha * \cos \theta + \cos \alpha * \sin \theta$
$= y / r * \cos \theta + x / r * \sin \theta$



Rotation

Rotation



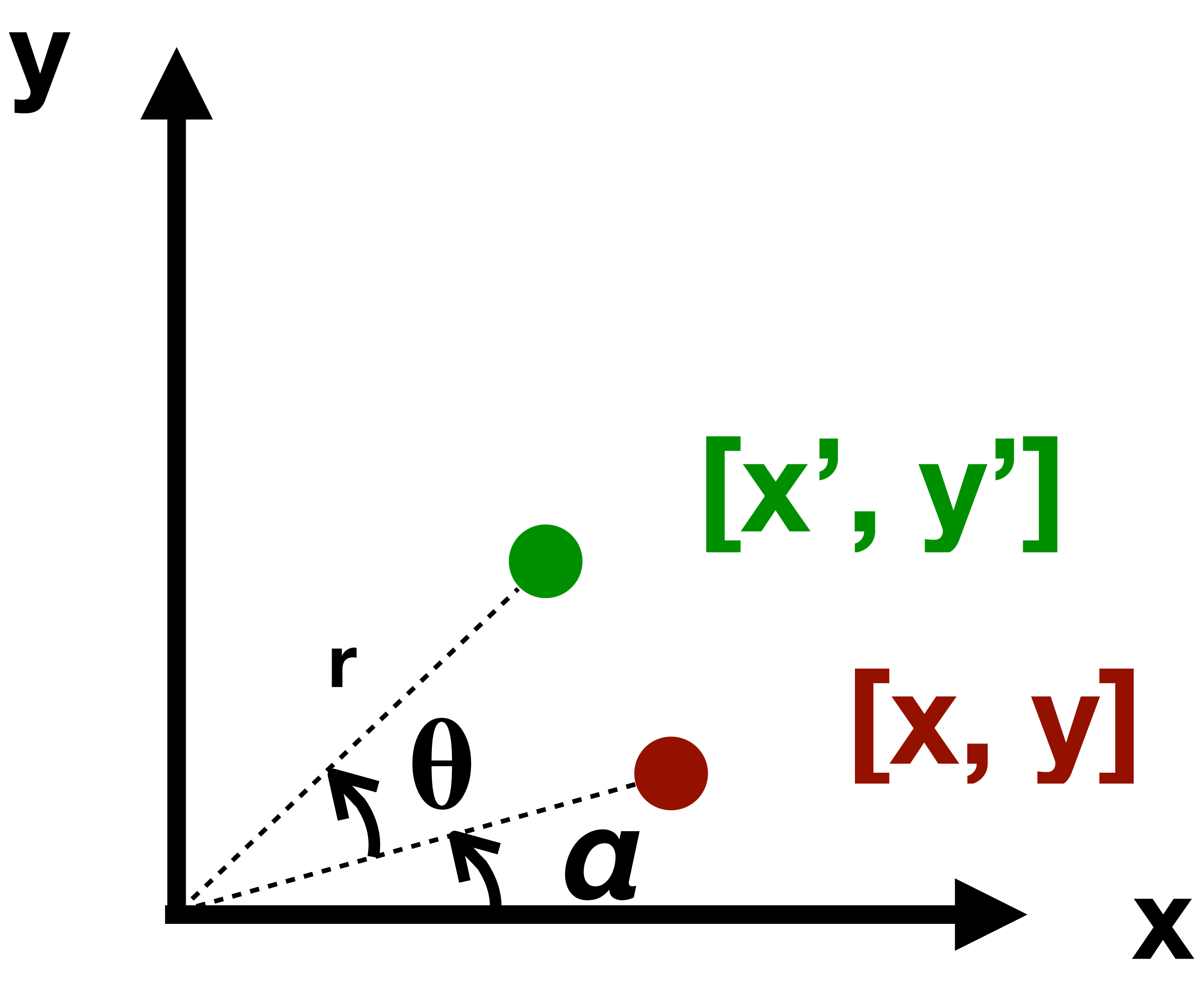
Rotation

$\sin \alpha = y / r$
$\cos \alpha = x / r$

$\sin(\alpha + \theta) = y' / r = \sin \alpha * \cos \theta + \cos \alpha * \sin \theta$
$= y / r * \cos \theta + x / r * \sin \theta$

$y' / r = y / r * \cos \theta + x / r * \sin \theta$

Rotation



Rotation

$\sin \alpha = y / r$

$\cos \alpha = x / r$

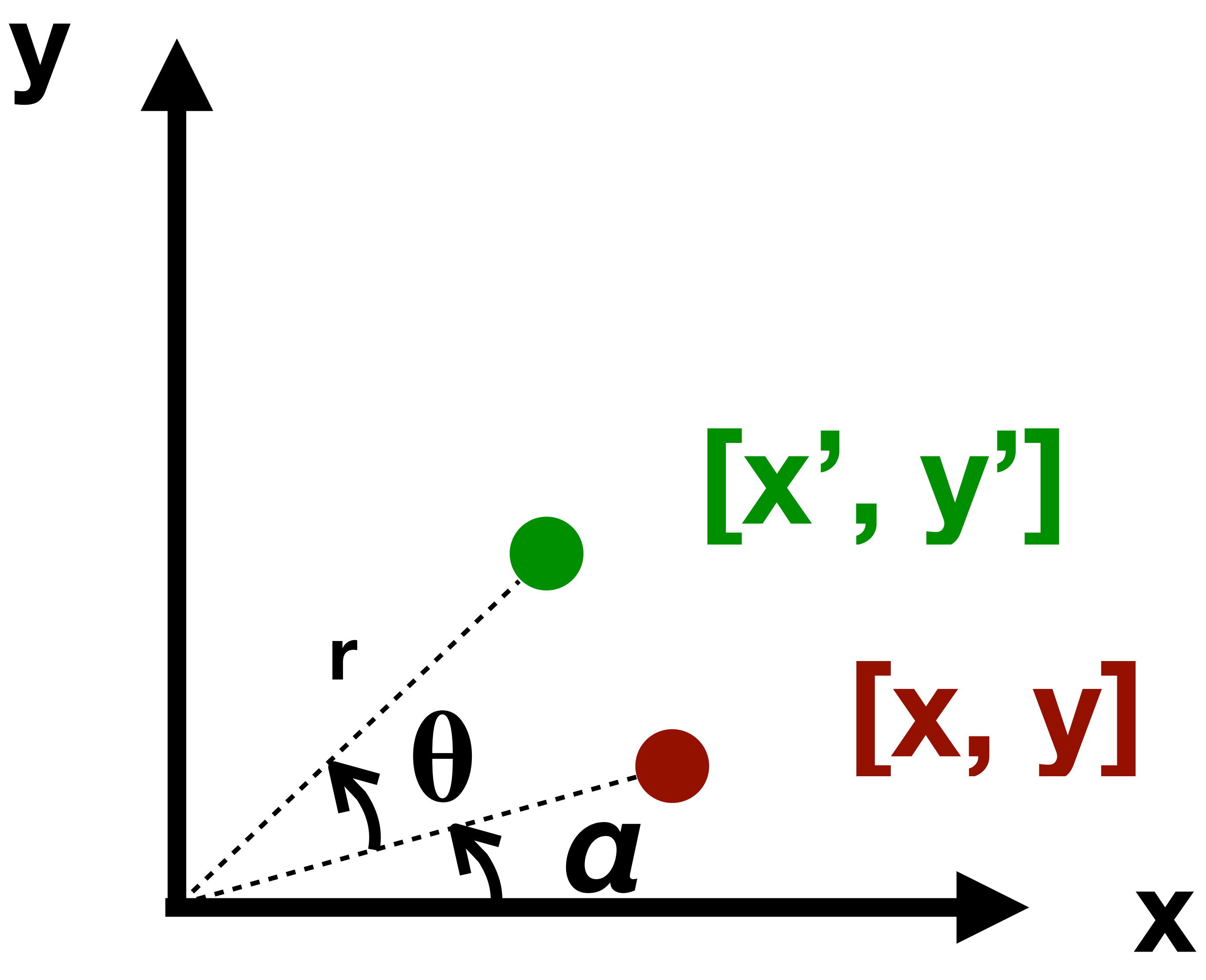
$\sin(\alpha + \theta) = y' / r = \sin \alpha * \cos \theta + \cos \alpha * \sin \theta$

$= y / r * \cos \theta + x / r * \sin \theta$

$y' / r = y / r * \cos \theta + x / r * \sin \theta$

$y' = y * \cos \theta + x * \sin \theta$

Rotation



Rotation

$\sin \alpha = y / r$
$\cos \alpha = x / r$

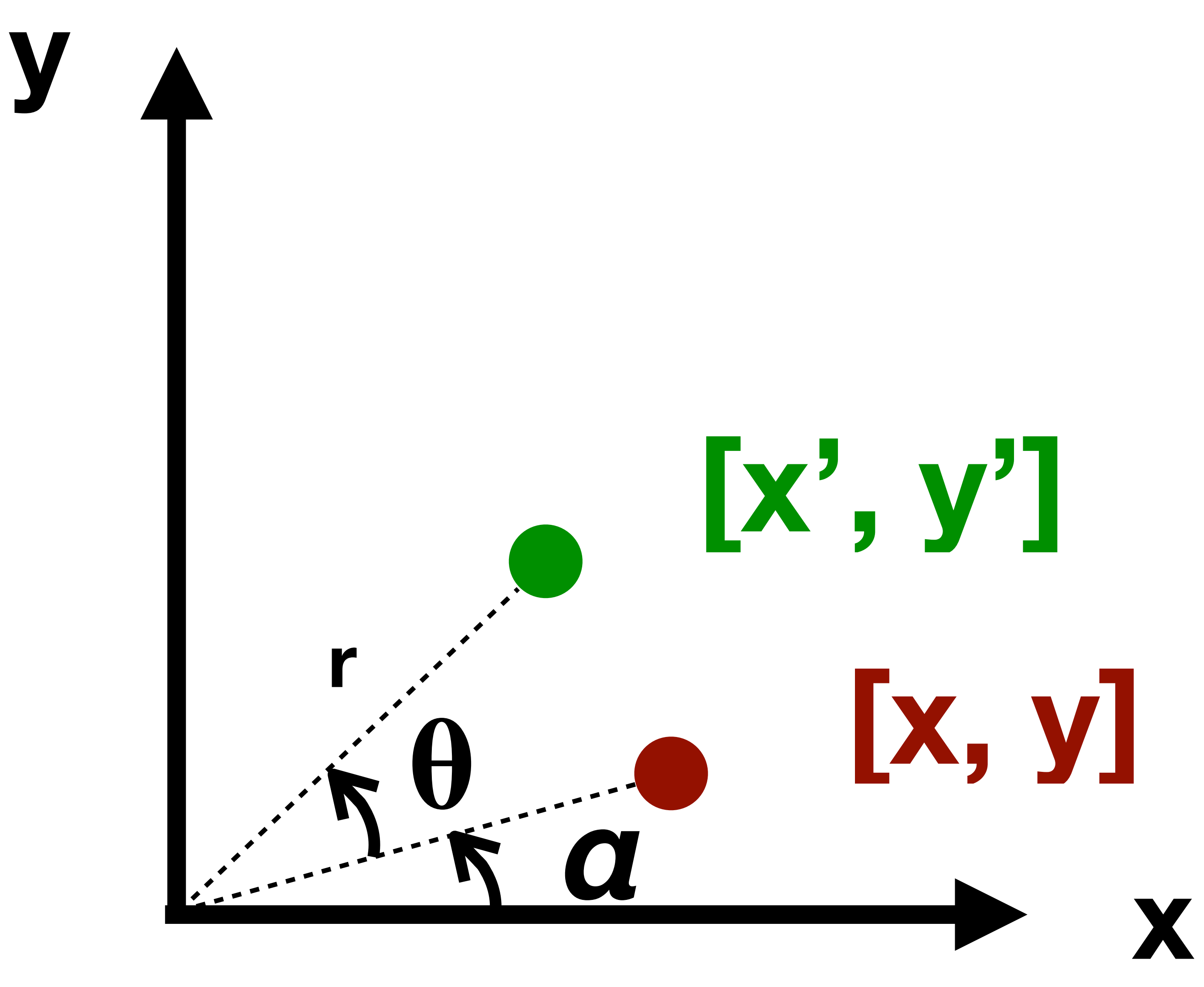
$\sin(\alpha + \theta) = y' / r = \sin \alpha * \cos \theta + \cos \alpha * \sin \theta$
$= y / r * \cos \theta + x / r * \sin \theta$

$y' / r = y / r * \cos \theta + x / r * \sin \theta$

$y' = y * \cos \theta + x * \sin \theta$

$\cos(\alpha + \theta) = x' / r = \cos \alpha * \cos \theta - \sin \alpha * \sin \theta$
$= x / r * \cos \theta - y / r * \sin \theta$

Rotation



Rotation

$\sin \alpha = y / r$
$\cos \alpha = x / r$

$\sin(\alpha + \theta) = y' / r = \sin \alpha * \cos \theta + \cos \alpha * \sin \theta$
$= y / r * \cos \theta + x / r * \sin \theta$

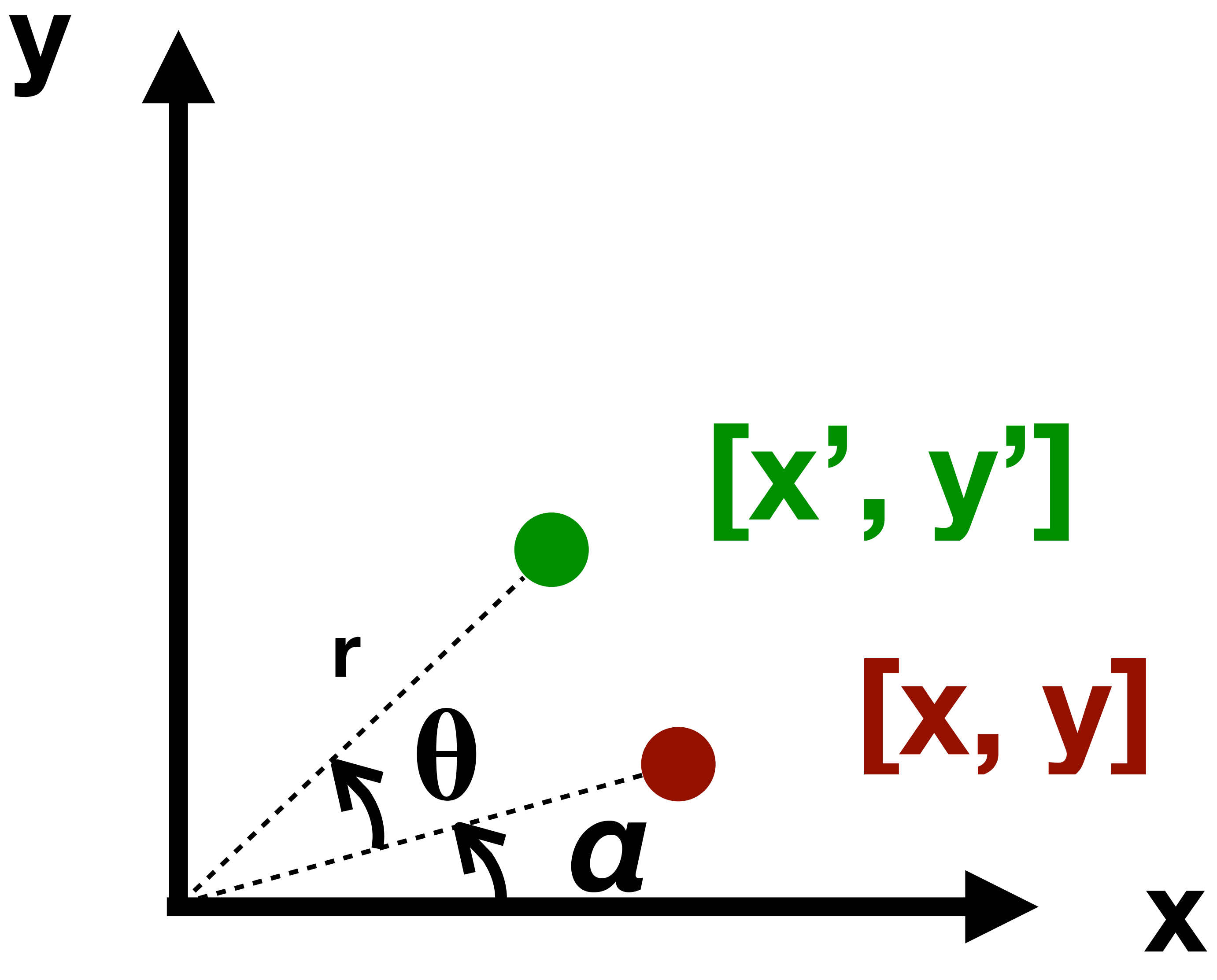
$y' / r = y / r * \cos \theta + x / r * \sin \theta$

$y' = y * \cos \theta + x * \sin \theta$

$\cos(\alpha + \theta) = x' / r = \cos \alpha * \cos \theta - \sin \alpha * \sin \theta$
$= x / r * \cos \theta - y / r * \sin \theta$

$x' / r = x / r * \cos \theta - y / r * \sin \theta$

Rotation



Rotation

$\sin \alpha = y / r$
$\cos \alpha = x / r$

$$\sin(\alpha + \theta) = y' / r = \sin \alpha * \cos \theta + \cos \alpha * \sin \theta$$
$$= y / r * \cos \theta + x / r * \sin \theta$$

$$y' / r = y / r * \cos \theta + x / r * \sin \theta$$

$$y' = y * \cos \theta + x * \sin \theta$$

$$\cos(\alpha + \theta) = x' / r = \cos \alpha * \cos \theta - \sin \alpha * \sin \theta$$
$$= x / r * \cos \theta - y / r * \sin \theta$$

$$x' / r = x / r * \cos \theta - y / r * \sin \theta$$

$$x' = x * \cos \theta - y * \sin \theta$$

Rotation Matrix (Around Z-axis)

$x' = x \cos \theta - y \sin \theta$

$y' = x \sin \theta + y \cos \theta$

$z' = z$

$$[x, y, z] \quad x \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} = [x', y', z']$$

Rotation Matrix (Around Z-axis)

$x' = x \cos\theta - y \sin\theta \qquad = xT_{00} + yT_{10} + zT_{20},$ *for $\forall$ x, y, z*

$y' = x \sin\theta + y \cos\theta \qquad = xT_{01} + yT_{11} + zT_{21},$ *for $\forall$ x, y, z*

$z' = z \qquad = xT_{02} + yT_{12} + zT_{22},$ *for $\forall$ x, y, z*

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Rotation Matrix (Around Z-axis)

$x' = x \cos\theta - y \sin\theta \quad = xT_{00} + yT_{10} + zT_{20},$ *for ∀ x, y, z*

$y' = x \sin\theta + y \cos\theta \quad = xT_{01} + yT_{11} + zT_{21},$ *for ∀ x, y, z*

$z' = z \quad = xT_{02} + yT_{12} + zT_{22},$ *for ∀ x, y, z*

$$[x, y, z] \quad \times \quad \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x', y', z']$$

Rotation Matrix (Around Z-axis)

$x' = x \cos\theta - y \sin\theta \quad = xT_{00} + yT_{10} + zT_{20},$ *for ∀ x, y, z*

$y' = x \sin\theta + y \cos\theta \quad = xT_{01} + yT_{11} + zT_{21},$ *for ∀ x, y, z*

$z' = z \quad = xT_{02} + yT_{12} + zT_{22},$ *for ∀ x, y, z*

$$[x, y, z] \quad \times \quad \begin{bmatrix} \cos\theta & \sin\theta & \boxed{\begin{matrix} 0 \\ 0 \\ 1 \end{matrix}} \\ -\sin\theta & \cos\theta & \\ 0 & 0 & \end{bmatrix} \quad = \quad [x', y', z']$$

z indeed doesn't change!

Rotation Matrix

Around X
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

Around Y
$$\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

Around Z
$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Derive the rest in your homework.

Fixed after lecture

Orthogonal Matrix

- Rotation matrix is **orthogonal matrix**, whose transpose is the same as inversion.
  - The length of each row is 1.
  - The rows are orthogonal vectors to each other.
- **Orthogonal vectors**:
  - $\mathbf{v1}$ $[x_1, y_1, z_1]$ and $\mathbf{v2}$ $[x_2, y_2, z_2]$ are orthogonal if $\mathbf{v1} \cdot \mathbf{v2} = x_1x_2 + y_1y_2 + z_1z_2 = 0$.
  - $\mathbf{v1} \cdot \mathbf{v2}$ is called the **dot (inner) product**.

$$\begin{array}{cl} \left[\begin{array}{ccc} \boxed{\cos\theta \quad \sin\theta \quad 0} \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{array}\right] & \begin{array}{l} \leftarrow \mathbf{v1} \\ \leftarrow \mathbf{v2} \\ \leftarrow \mathbf{v3} \end{array} \end{array}$$

Length of $\mathbf{v1}$ = sqrt($\mathbf{v1} \cdot \mathbf{v1}$)
= $\cos^2\theta + \sin^2\theta + 0 = 1$

Fixed after lecture

Orthogonal Matrix

- Rotation matrix is **orthogonal matrix**, whose transpose is the same as inversion.
  - The length of each row is 1.
  - The rows are orthogonal vectors to each other.
- **Orthogonal vectors**:
  - $\mathbf{v1}$ $[x_1, y_1, z_1]$ and $\mathbf{v2}$ $[x_2, y_2, z_2]$ are orthogonal if $\mathbf{v1} \cdot \mathbf{v2} = x_1x_2 + y_1y_2 + z_1z_2 = 0$.
  - $\mathbf{v1} \cdot \mathbf{v2}$ is called the **dot (inner) product**.
- Any orthogonal matrix can be used to represent a rotation (might be around an arbitrary axis though — more on this later).

$$\begin{array}{cl} \begin{bmatrix} \boxed{\cos\theta \quad \sin\theta \quad 0} \\ -\sin\theta \quad \cos\theta \quad 0 \\ 0 \quad\quad 0 \quad\quad 1 \end{bmatrix} & \begin{array}{l} \leftarrow \mathbf{v1} \\ \leftarrow \mathbf{v2} \\ \leftarrow \mathbf{v3} \end{array} \end{array}$$

Length of $\mathbf{v1}$ = sqrt($\mathbf{v1} \cdot \mathbf{v1}$)
$= \cos^2\theta + \sin^2\theta + 0 = 1$

Fixed after lecture

Orthogonal Matrix

- Rotation matrix is **orthogonal matrix**, whose transpose is the same as inversion.
  - The length of each row is 1.
  - The rows are orthogonal vectors to each other.
- **Orthogonal vectors**:
  - $\mathbf{v1}$ $[x_1, y_1, z_1]$ and $\mathbf{v2}$ $[x_2, y_2, z_2]$ are orthogonal if $\mathbf{v1} \cdot \mathbf{v2} = x_1x_2 + y_1y_2 + z_1z_2 = 0$.
  - $\mathbf{v1} \cdot \mathbf{v2}$ is called the **dot (inner) product**.
- Any orthogonal matrix can be used to represent a rotation (might be around an arbitrary axis though — more on this later).

An important property of orthogonal matrix: $Q^T = Q^{-1}$ $Q\,Q^T = I$

$$\begin{array}{cl} \left[\begin{array}{ccc} \boxed{\cos\theta & \sin\theta & 0} \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{array}\right] & \begin{array}{l} \leftarrow \mathbf{v1} \\ \leftarrow \mathbf{v2} \\ \leftarrow \mathbf{v3} \end{array} \end{array}$$

Length of $\mathbf{v1}$ = sqrt($\mathbf{v1} \cdot \mathbf{v1}$)
$= \cos^2\theta + \sin^2\theta + 0 = 1$

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.

1. First rotation: $P_{t1} = P \times T_z$

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.

1. **First rotation: $P_{t1} = P \times T_z$**
2. **Second rotation: $P_{t2} = P_{t1} \times T_y$**

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.

1. **First rotation: $P_{t1} = P \times T_z$**
2. **Second rotation: $P_{t2} = P_{t1} \times T_y$**
3. **Scaling: $P' = P_{t2} \times T_s$**

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.

1. **First rotation: $P_{t1} = P \times T_z$**
2. **Second rotation: $P_{t2} = P_{t1} \times T_y$**
3. **Scaling: $P' = P_{t2} \times T_s$**
4. **Overall: $P' = P \times T_z \times T_y \times T_s$**

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.

1. **First rotation: $P_{t1} = P \times T_z$**
2. **Second rotation: $P_{t2} = P_{t1} \times T_y$**
3. **Scaling: $P' = P_{t2} \times T_s$**
4. **Overall: $P' = P \times \boxed{T_z \times T_y \times T_s}$**

Since matrix multiplication is **associative**, let $\mathbf{T = T_z \times T_y \times T_s}$, which represents the combination effect of the three transformations

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.

1. **First rotation: $P_{t1} = P \times T_z$**
2. **Second rotation: $P_{t2} = P_{t1} \times T_y$**
3. **Scaling: $P' = P_{t2} \times T_s$**
4. **Overall: $P' = P \times \boxed{T_z \times T_y \times T_s}$**
5. **$P' = P \times T$**

Since matrix multiplication is **associative**, let $\mathbf{T = T_z \times T_y \times T_s}$, which represents the combination effect of the three transformations

Combining Transformations

- For instance: rotate P around the z-axis, then around y-axis, and scale it.
- Generally, combining transformations can be done by multiplying individual transformation matrices together first to derive a composite matrix, which is then applied once in the end.

1. **First rotation: $P_{t1} = P \times T_z$**
2. **Second rotation: $P_{t2} = P_{t1} \times T_y$**
3. **Scaling: $P' = P_{t2} \times T_s$**
4. **Overall: $P' = P \times \boxed{T_z \times T_y \times T_s}$**
5. **$P' = P \times T$**

Since matrix multiplication is **associative**, let **$T = T_z \times T_y \times T_s$**, which represents the combination effect of the three transformations

Combining Transformations

- Can we reorder the individual transformations?

Combining Transformations

- Can we reorder the individual transformations?
- Is rotating P around the z-axis, then around y-axis, and scaling P the same as rotating around y, then z, then scaling P?

Combining Transformations

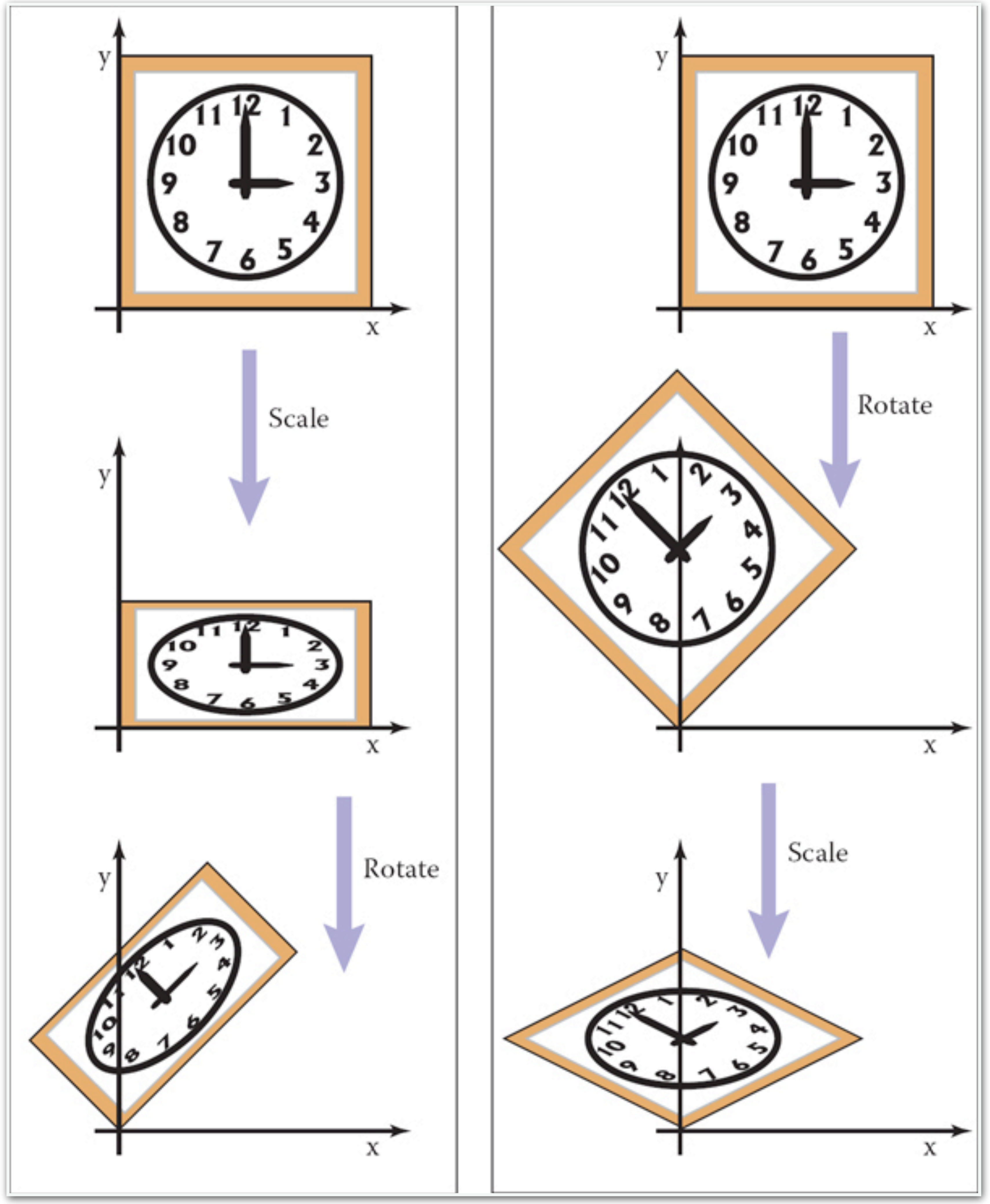
- Can we reorder the individual transformations?
- Is rotating P around the z-axis, then around y-axis, and scaling P the same as rotating around y, then z, then scaling P?
  - **T_z x T_y x T_s = T_y x T_z x T_s?**

Combining Transformations

- Can we reorder the individual transformations?
- Is rotating P around the z-axis, then around y-axis, and scaling P the same as rotating around y, then z, then scaling P?
  - **$T_z \times T_y \times T_s = T_y \times T_z \times T_s$?**
- No. Matrix multiplication is not commutative.

Combining Transformations

- Can we reorder the individual transformations?
- Is rotating P around the z-axis, then around y-axis, and scaling P the same as rotating around y, then z, then scaling P?
  - $T_z \times T_y \times T_s = T_y \times T_z \times T_s$?
- No. Matrix multiplication is not commutative.



Decomposing Transformations

- Can we decompose any arbitrary transformation into a sequence of basic transformations?
- Yes. There are multiple ways. One common way is through **singular value decomposition** (SVD).
- Any arbitrary transformation can be composed as a rotation, a scaling, and another rotation.
- There are other ways to decompose a matrix and thus other ways to decompose a transformation.

Arbitrary matrix

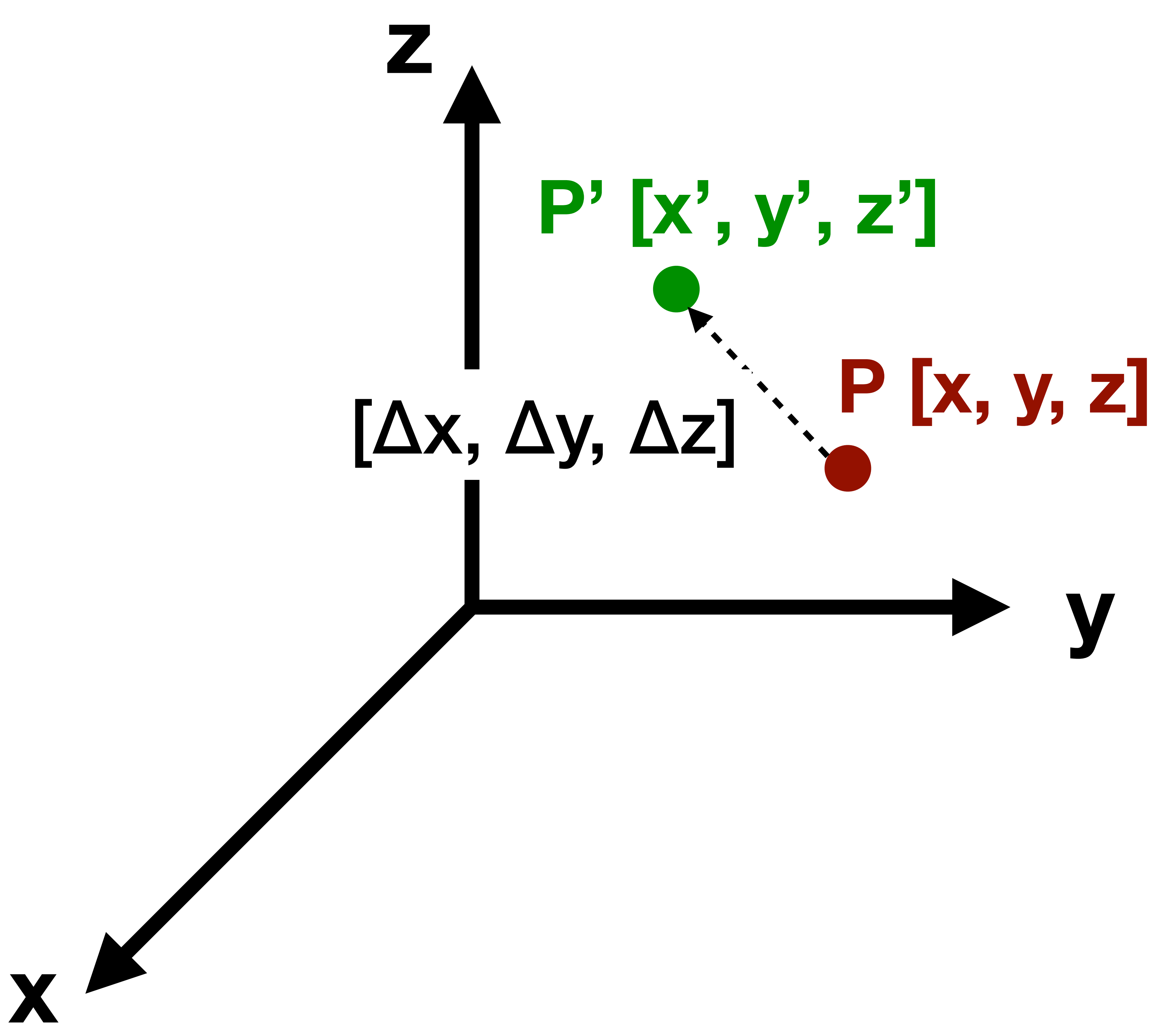
S is a diagonal matrix

$$A = USV^T$$

Both U and V^T are orthogonal matrices

Translation

- Move P [x, y, z] along the x-axis by Δx
- Move P [x, y, z] along the y-axis by Δy
- Move P [x, y, z] along the z-axis by Δz
- P [x, y, z] becomes P’ [x + Δx, y + Δy, z + Δz]



Translation

- What should the transformation matrix be if we want to move P [x, y, z] to P' [x + Δx, y + Δy, z + Δz] regardless of where P is?

Translation

- What should the transformation matrix be if we want to move P [x, y, z] to P' [x + Δx, y + Δy, z + Δz] regardless of where P is?
- Can we treat it as scaling? What would the scaling factor be?
  - $S_0 = (x + \Delta x) / x = 1 + \Delta x / x$
  - $S_1 = (y + \Delta y) / y = 1 + \Delta y / y$
  - $S_2 = (z + \Delta z) / z = 1 + \Delta z / z$
  - The scaling factor depends on [x, y, z].
  - So there is no single scaling factor that applies to all points P.
  - Reducing translation to scaling isn't a general approach.

Translation

- What should the transformation matrix be if we want to move P [x, y, z] to P' [x + Δx, y + Δy, z + Δz] regardless of where P is?
- Can we treat it as scaling? What would the scaling factor be?
  - $S_0 = (x + \Delta x) / x = 1 + \Delta x / x$
  - $S_1 = (y + \Delta y) / y = 1 + \Delta y / y$
  - $S_2 = (z + \Delta z) / z = 1 + \Delta z / z$
  - The scaling factor depends on [x, y, z].
  - So there is no single scaling factor that applies to all points P.
  - Reducing translation to scaling isn't a general approach.

$$\begin{bmatrix} \mathbf{1+\Delta x/x} & 0 & 0 \\ 0 & \mathbf{1+\Delta y/y} & 0 \\ 0 & 0 & \mathbf{1+\Delta z/z} \end{bmatrix}$$

Translation

- What should the transformation matrix be?

$$x' = xT_{00} + yT_{10} + zT_{20} = x + \Delta x$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Translation

- What should the transformation matrix be?

$$x' = xT_{00} + yT_{10} + zT_{20} = x + \Delta x$$

A 3x3 matrix can't express the Δx term!

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Translation

- We could make it work by adding one new term: T_{30}

$$x' = xT_{00} + yT_{10} + zT_{20} + T_{30} = x + \Delta x$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} = [x', y', z']$$

Translation

- We could make it work by adding one new term: T_{30}

$$x' = xT_{00} + yT_{10} + zT_{20} + T_{30} = x + \Delta x$$

$$\begin{array}{cccc} \uparrow & \uparrow & \uparrow & \uparrow \\ 1 & 0 & 0 & \Delta x \end{array}$$

$$[x, y, z] \quad \times \quad \begin{bmatrix} T_{00}, T_{01}, T_{02} \\ T_{10}, T_{11}, T_{12} \\ T_{20}, T_{21}, T_{22} \end{bmatrix} \quad = \quad [x', y', z']$$

Translation

- Effectively, the matrix becomes 4x3, and P needs to be 1x4.

$$x' = xT_{00} + yT_{10} + zT_{20} + T_{30} = x + \Delta x$$

$$\begin{array}{cccc} \uparrow & \uparrow & \uparrow & \uparrow \\ 1 & 0 & 0 & \Delta x \end{array}$$

$$[x, y, z, 1] \quad x \quad \begin{bmatrix} \boxed{\begin{matrix} 1 \\ 0 \\ 0 \\ \Delta x \end{matrix}} & \begin{matrix} T_{01}, T_{02} \\ T_{11}, T_{12} \\ T_{21}, T_{22} \\ T_{31}, T_{32} \end{matrix} \end{bmatrix} \quad = \quad [x', y', z']$$

Translation

$$[x, y, z, 1] \quad \mathbf{x} \quad \begin{bmatrix} \boxed{\begin{matrix}1\\0\\0\\\Delta x\end{matrix}} & \boxed{\begin{matrix}0\\1\\0\\\Delta y\end{matrix}} & \boxed{\begin{matrix}0\\0\\1\\\Delta z\end{matrix}} \end{bmatrix} \quad = \quad [x', y', z']$$

Translation

- But, P’ is still 1x3, which prevents **further translations** on P’!
- So P’ needs to be 1x4 as well, which means T needs to be 4x4.

$$[x, y, z, 1] \quad \times \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \Delta x & \Delta y & \Delta z \end{bmatrix} \quad = \quad [x', y', z']$$

Translation

- What should the additional column be?

$$[x, y, z, 1] \times \begin{bmatrix} 1 & 0 & 0 & T_{03} \\ 0 & 1 & 0 & T_{13} \\ 0 & 0 & 1 & T_{23} \\ \Delta x & \Delta y & \Delta z & T_{33} \end{bmatrix} = [x', y', z', 1]$$

Translation

- What should the additional column be?

$$xT_{03} + yT_{13} + zT_{23} + T_{33} = 1, \textit{ for } \forall\, x, y, z$$

$$[x, y, z, 1] \times \begin{bmatrix} \boxed{\begin{matrix} 1 \\ 0 \\ 0 \\ \Delta x \end{matrix}} & \boxed{\begin{matrix} 0 \\ 1 \\ 0 \\ \Delta y \end{matrix}} & \boxed{\begin{matrix} 0 \\ 0 \\ 1 \\ \Delta z \end{matrix}} & \boxed{\begin{matrix} T_{03} \\ T_{13} \\ T_{23} \\ T_{33} \end{matrix}} \end{bmatrix} = [x', y', z', 1]$$

Translation

- What should the additional column be?

$$xT_{03} + yT_{13} + zT_{23} + T_{33} = 1, \textit{ for } \forall\, x, y, z$$

$$\begin{array}{cccc} \uparrow & \uparrow & \uparrow & \uparrow \\ 0 & 0 & 0 & 1 \end{array}$$

$$[x, y, z, 1] \times \begin{bmatrix} 1 & 0 & 0 & T_{03} \\ 0 & 1 & 0 & T_{13} \\ 0 & 0 & 1 & T_{23} \\ \Delta x & \Delta y & \Delta z & T_{33} \end{bmatrix} = [x', y', z', 1]$$

Translation

- What should the additional column be?

$$xT_{03} + yT_{13} + zT_{23} + T_{33} = 1, \textit{ for } \forall\, x, y, z$$

$$\begin{array}{cccc} \uparrow & \uparrow & \uparrow & \uparrow \\ 0 & 0 & 0 & 1 \end{array}$$

$$[x, y, z, 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix} = [x', y', z', 1]$$

Homogeneous Coordinates

- [x, y, z] is the **cartesian coordinates** of P.
- [x, y, z, 1] is the **homogeneous coordinates** of P.
- Homogeneous coordinates are introduced so that translation could be expressed as matrix multiplication.

$$[x, y, z, 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix} = [x', y', z', 1]$$

Homogeneous Coordinates

- For translation to work:
  - The last element in the homogeneous coordinates has to be 1.
  - The last column of the matrix has to be $[0, 0, 0, 1]^T$ (We will see what would happen if this is not the case later in the semester when we talk about perspective transformations.)
- But do they generally apply to other transformations?

$$[x, y, z, 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix} = [x', y', z', 1]$$

The Identity Matrix in Homogeneous Coordinates

- The top-left 3x3 sub-matrix is the same identity matrix as before.

$$[x, y, z, 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x', y', z', 1]$$

Scaling in Homogeneous Coordinates

- Scaling P [x, y, z, 1] to P' [$S_0 \cdot x$, $S_1 \cdot y$, $S_2 \cdot z$, 1].
- The top-left 3x3 sub-matrix is the same as before.

$$[x, y, z, 1] \times \begin{bmatrix} S_0 & 0 & 0 & 0 \\ 0 & S_1 & 0 & 0 \\ 0 & 0 & S_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x', y', z', 1]$$

Rotation in Homogeneous Coordinates

- Rotate P around the z-axis by θ?
- The top-left 3x3 sub-matrix is the same as before.

$$[x, y, z, 1] \times \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x', y', z', 1]$$

Composite Transformation in Homogeneous Coordinates

- Rotate P around the z-axis by θ and translate by [Δx, Δy, Δz].

Responsible for rotation

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & -\cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & -\cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix}$$

Responsible for translation

Composite Transformation in Homogeneous Coordinates

- Easily composes translation and rotational transformations

$$\begin{bmatrix} R_{3x3} & 0_{3x1} \\ T_{1x3} & 1_{1x1} \end{bmatrix}$$

Responsible for rotation

$$[x, y, z, 1] \ \text{x} \begin{bmatrix} T_{00} & T_{01} & T_{02} & 0 \\ T_{10} & T_{11} & T_{12} & 0 \\ T_{20} & T_{21} & T_{22} & 0 \\ T_{30} & T_{31} & T_{32} & 1 \end{bmatrix} = [x', y', z', 1]$$

Responsible for translation

Block Matrix Perspective

$$[[x, y, z]_{1x3}, [1]_{1x1}] \ \text{x} \ \begin{bmatrix} R_{3x3} & 0_{3x1} \\ T_{1x3} & 1_{1x1} \end{bmatrix} = [[x', y', z']_{1x3}, [1]_{1x1}]$$

$$[x', y', z']_{1x3} = \underbrace{\boxed{[x, y, z]_{1x3} \ \text{x} \ R_{3x3}}}_{\text{Rotation}} + \underbrace{\boxed{[1]_{1x1} \ \text{x} \ T_{1x3}}}_{\text{Translation}}$$

Where transformation actually happens

$$[1]_{1x1} = [x, y, z]_{1x3} \ \text{x} \ 0_{3x1} + [1]_{1x1} \ \text{x} \ 1_{1x1}$$

Invariant. Just there to make sure matrix multiplication works.

Affine Transformation

- Matrix that has this form is called an affine transformation matrix.
- Intuitively, affine transformation preserves straight lines.
  - Translation, scale, rotation all do not bend straight lines.

$$\begin{bmatrix} T_{00} & T_{01} & T_{02} & 0 \\ T_{10} & T_{11} & T_{12} & 0 \\ T_{20} & T_{21} & T_{22} & 0 \\ T_{30} & T_{31} & T_{32} & 1 \end{bmatrix}$$

Cartesian-Homogeneous Coordinates Conversion

- [x, y, z] <==> [x, y, z, 1]

Cartesian-Homogeneous Coordinates Conversion

- [x, y, z] <==> [x, y, z, 1]
- In fact, [x, y, z] <==> [kx, ky, kz, k]

Cartesian-Homogeneous Coordinates Conversion

- [x, y, z] <==> [x, y, z, 1]
- In fact, [x, y, z] <==> [kx, ky, kz, k]
- If [x, y, z, 1] after transformation T becomes [x’, y’, z’, 1], then [kx, ky, kz, k] after the same transformation T will become [kx’, ky’, kz’, k], because T is linear (matrix multiplication).

Cartesian-Homogeneous Coordinates Conversion

- [x, y, z] <==> [x, y, z, 1]
- In fact, [x, y, z] <==> [kx, ky, kz, k]
- If [x, y, z, 1] after transformation T becomes [x’, y’, z’, 1], then [kx, ky, kz, k] after the same transformation T will become [kx’, ky’, kz’, k], because T is linear (matrix multiplication).
  - In this case, we get the Cartesian coordinates by [kx’/k, ky’/k, kz’/k, k/k].

Cartesian-Homogeneous Coordinates Conversion

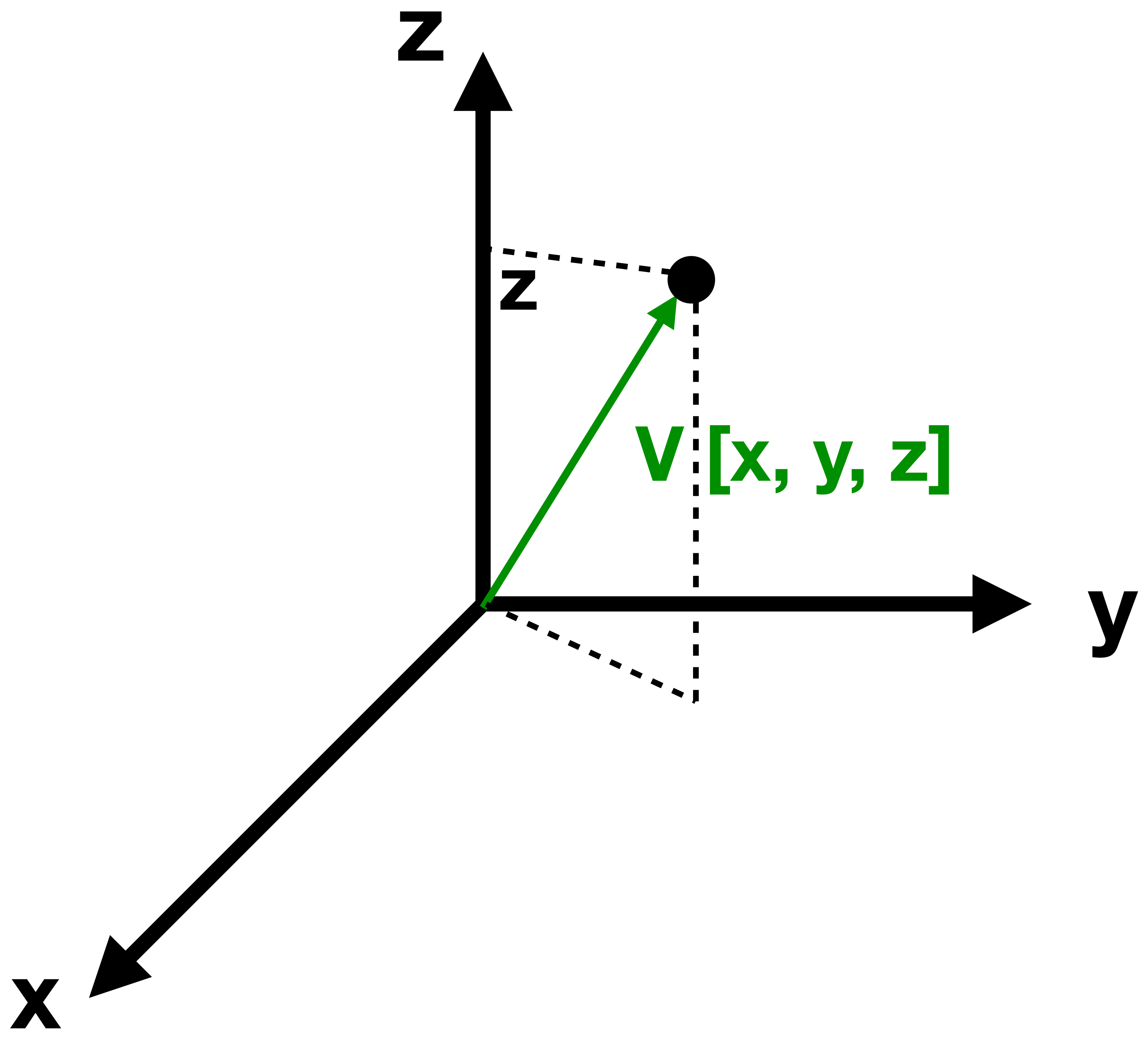
- [x, y, z] <==> [x, y, z, 1]
- In fact, [x, y, z] <==> [kx, ky, kz, k]
- If [x, y, z, 1] after transformation T becomes [x’, y’, z’, 1], then [kx, ky, kz, k] after the same transformation T will become [kx’, ky’, kz’, k], because T is linear (matrix multiplication).
  - In this case, we get the Cartesian coordinates by [kx’/k, ky’/k, kz’/k, k/k].
  - Usually k is 1, but k could be set to other values later (e.g., in perspective transformation).

Cartesian-Homogeneous Coordinates Conversion

- [x, y, z] <==> [x, y, z, 1]
- In fact, [x, y, z] <==> [kx, ky, kz, k]
- If [x, y, z, 1] after transformation T becomes [x', y', z', 1], then [kx, ky, kz, k] after the same transformation T will become [kx', ky', kz', k], because T is linear (matrix multiplication).
  - In this case, we get the Cartesian coordinates by [kx'/k, ky'/k, kz'/k, k/k].
  - Usually k is 1, but k could be set to other values later (e.g., in perspective transformation).
  - The kx, ky, kz, and k in [kx, ky, kz, k] don't have physical meanings. When you convert it back to [x, y, z], it then corresponds to a point in the physical world.

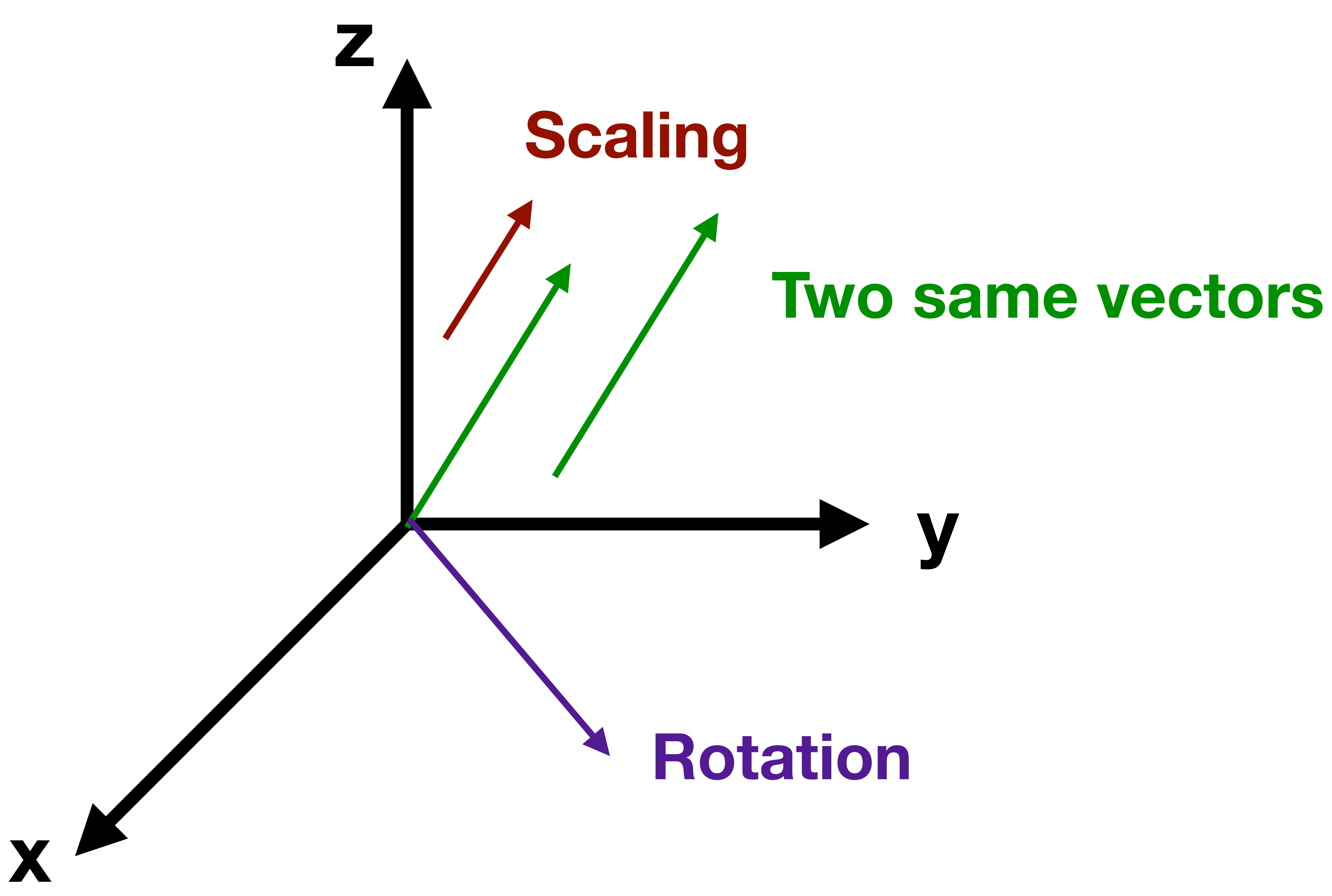
Vector

- A vector has the same representation of a point: [x, y, z].
- A vector represents a **direction** between [0, 0, 0] and [x, y, z] with a **length**.
- A unit vector or normalized vector is one whose length sqrt($x^2+y^2+z^2$) is 1.



Vector

- A vector is “positionless”, so translating vectors is meaningless.
- Rotation and scaling are meaningful vector transformations.



Vector Transformation in Homogeneous Coordinates

- Vector and point transformations are almost the same, but:
- V [x, y, z] in Cartesian coordinates is represented by [x, y, z, **0**] in homogeneous coordinates. 0 ensures that translation doesn't change the vector.
- The homogeneous transformation matrix is the same.

$$[x, y, z, 0] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \boxed{\Delta x \quad \Delta y \quad \Delta z} & & & 1 \end{bmatrix} = [x', y', z', 0]$$

Vector Transformation in Homogeneous Coordinates

- Rotate V [x, y, z] around z-axis by θ.
- Same transformation matrix as before. The only difference is that the last element in the homogeneous coordinate is 0 now.

$$[x, y, z, 0] \times \begin{bmatrix} \boxed{\begin{matrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{matrix}} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix} = [x', y', z', 0]$$

Block Matrix Perspective of Vector Transformation

$$[[x, y, z]_{1x3}, [0]_{1x1}] \text{ x } \begin{bmatrix} R_{3x3} & 0_{3x1} \\ T_{1x3} & 1_{1x1} \end{bmatrix} = [[x', y', z']_{1x3}, [0]_{1x1}]$$

$$[x', y', z']_{1x3} = \boxed{[x, y, z]_{1x3} \text{ x } R_{3x3}} + \boxed{[0]_{1x1} \text{ x } T_{1x3}}$$

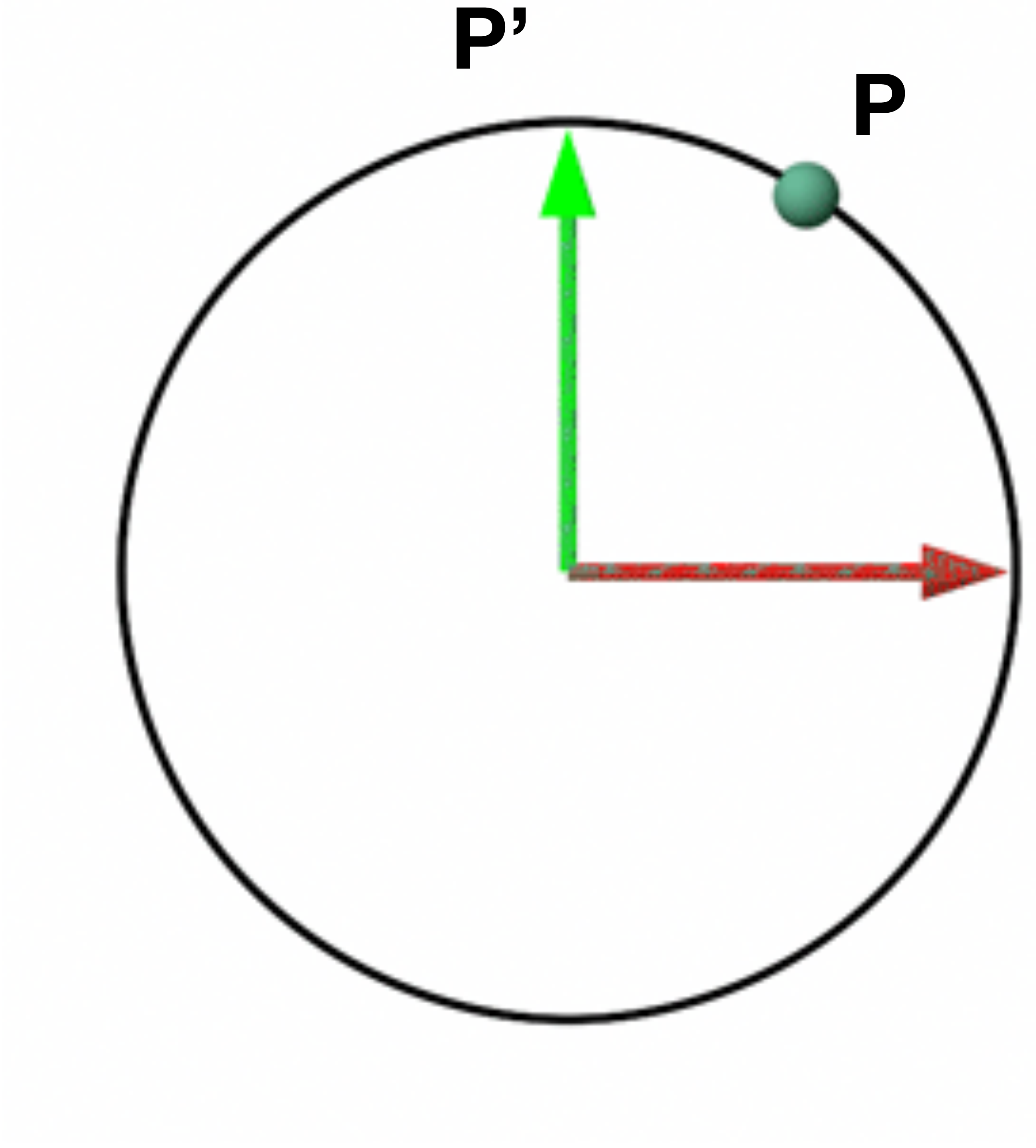
Rotation

Translation, which is always 0
(i.e., translation-invariant)

$$[0]_{1x1} = [x, y, z]_{1x3} \text{ x } 0_{3x1} + [0]_{1x1} \text{ x } 1_{1x1}$$ ← Invariant

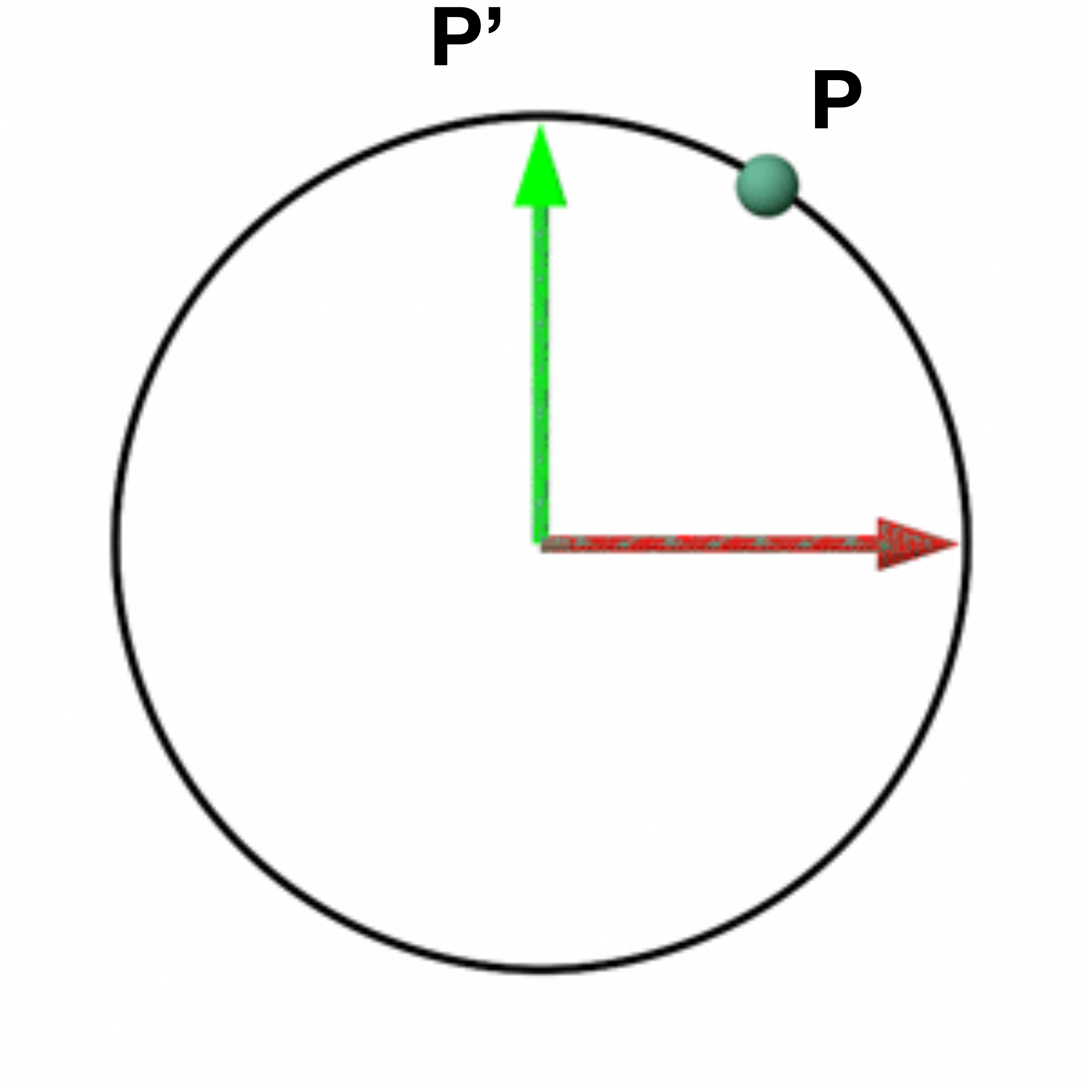
Another Way to Think About Point Transformation

- Rotating P to P': rotating P in the current coordinate system (a.k.a., frame) F_0'; or:
- We could think of it as rotating the current coordinating system F_0 to a new coordinate system F_1, while keeping the relative position of P unchanged.
  - The coordinates of P in F_0 and F_1 are the same.
- Transforming coordinate systems is very useful in all visual computing domains, as we will see later.



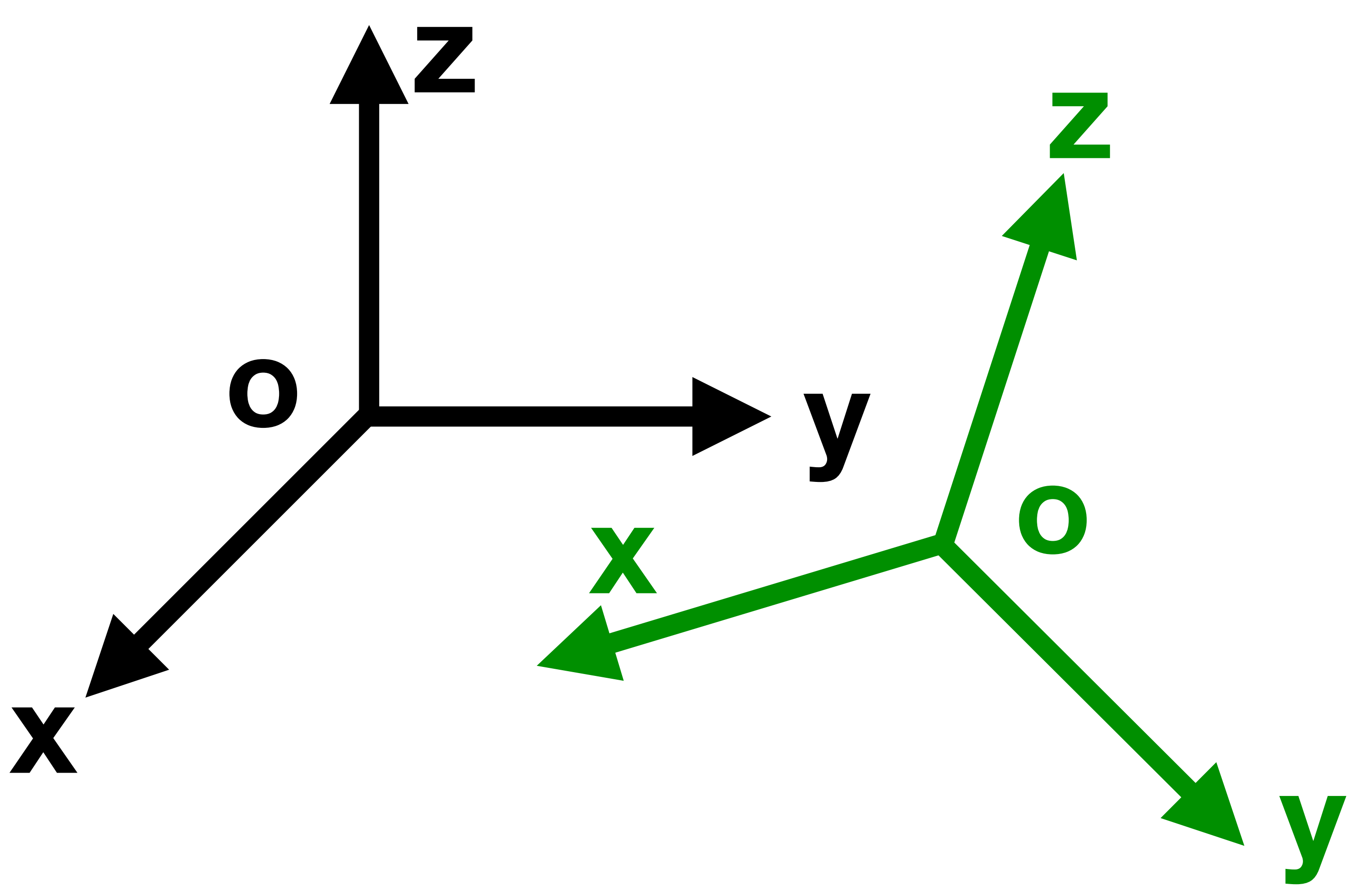
Another Way to Think About Point Transformation

- Rotating P to P': rotating P in the current coordinate system (a.k.a., frame) F_0'; or:
- We could think of it as rotating the current coordinating system F_0 to a new coordinate system F_1, while keeping the relative position of P unchanged.
  - The coordinates of P in F_0 and F_1 are the same.
- Transforming coordinate systems is very useful in all visual computing domains, as we will see later.



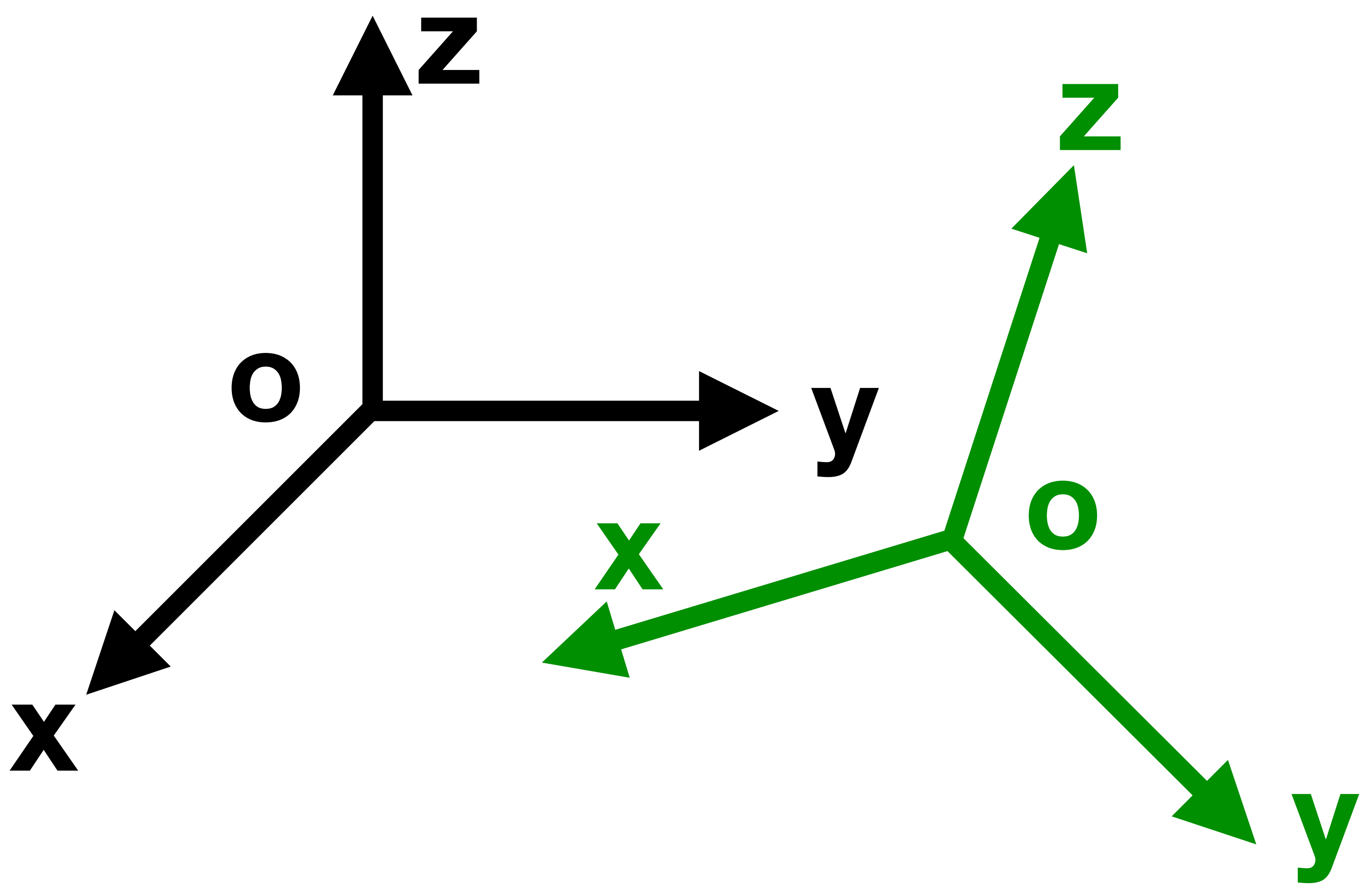
How to Transform a Frame/Coordinate System?

- A Cartesian coordinate system/frame is define by its **origin** [0, 0, 0] and three **basis vectors**: the x axis [1, 0, 0], y axis [0, 1, 0] and z axis [0, 0, 1].



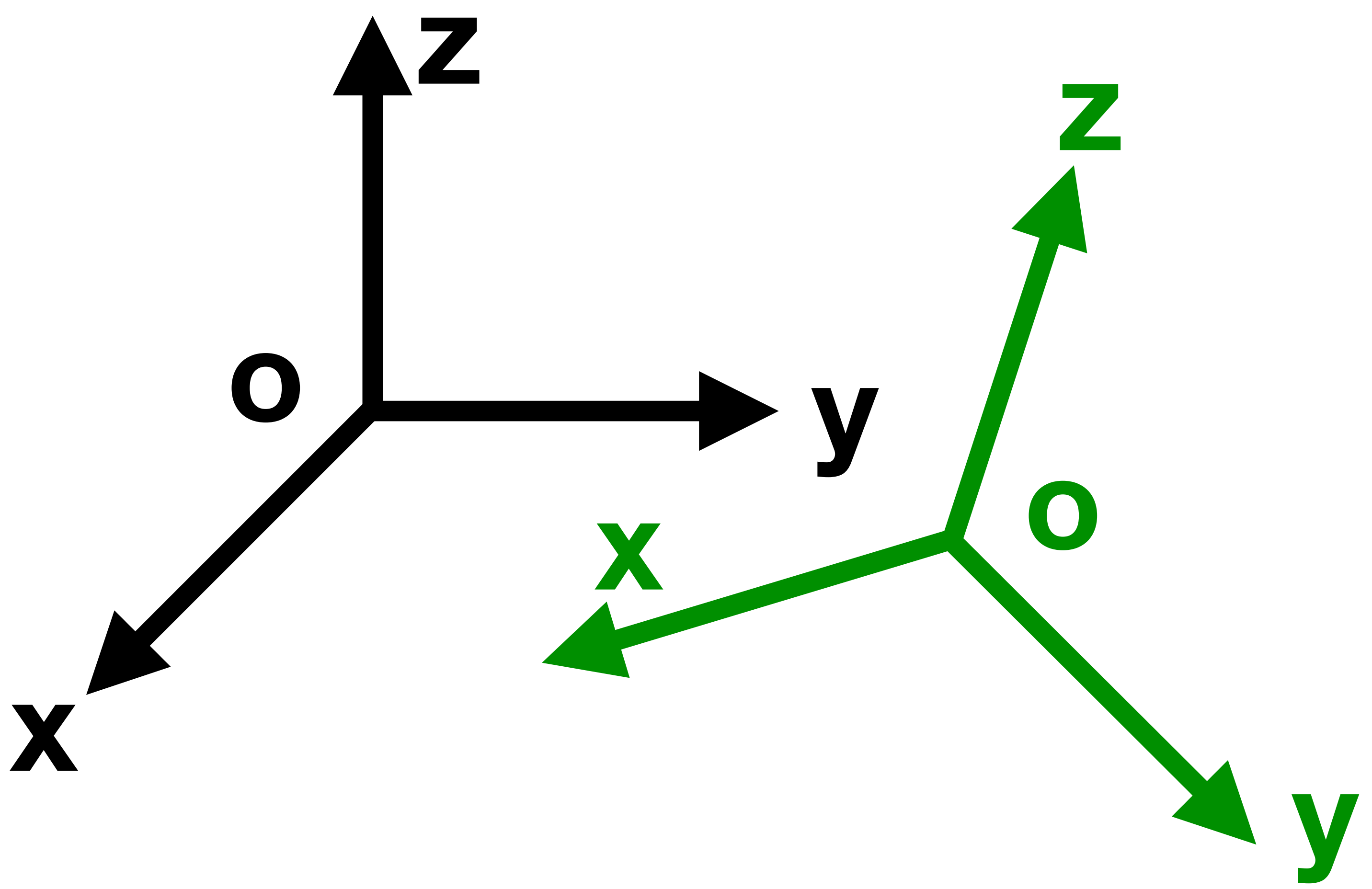
How to Transform a Frame/Coordinate System?

- A Cartesian coordinate system/frame is define by its **origin** [0, 0, 0] and three **basis vectors**: the x axis [1, 0, 0], y axis [0, 1, 0] and z axis [0, 0, 1].
- When we create a new frame, we can think of it transforming three original basis vectors and the origin to three new basis vectors and a new origin.



How to Transform a Frame/Coordinate System?

- A Cartesian coordinate system/frame is define by its **origin** [0, 0, 0] and three **basis vectors**: the x axis [1, 0, 0], y axis [0, 1, 0] and z axis [0, 0, 1].
- When we create a new frame, we can think of it transforming three original basis vectors and the origin to three new basis vectors and a new origin.
- **Key**: 4 transformations (3 vector transformations + 1 point transformation).



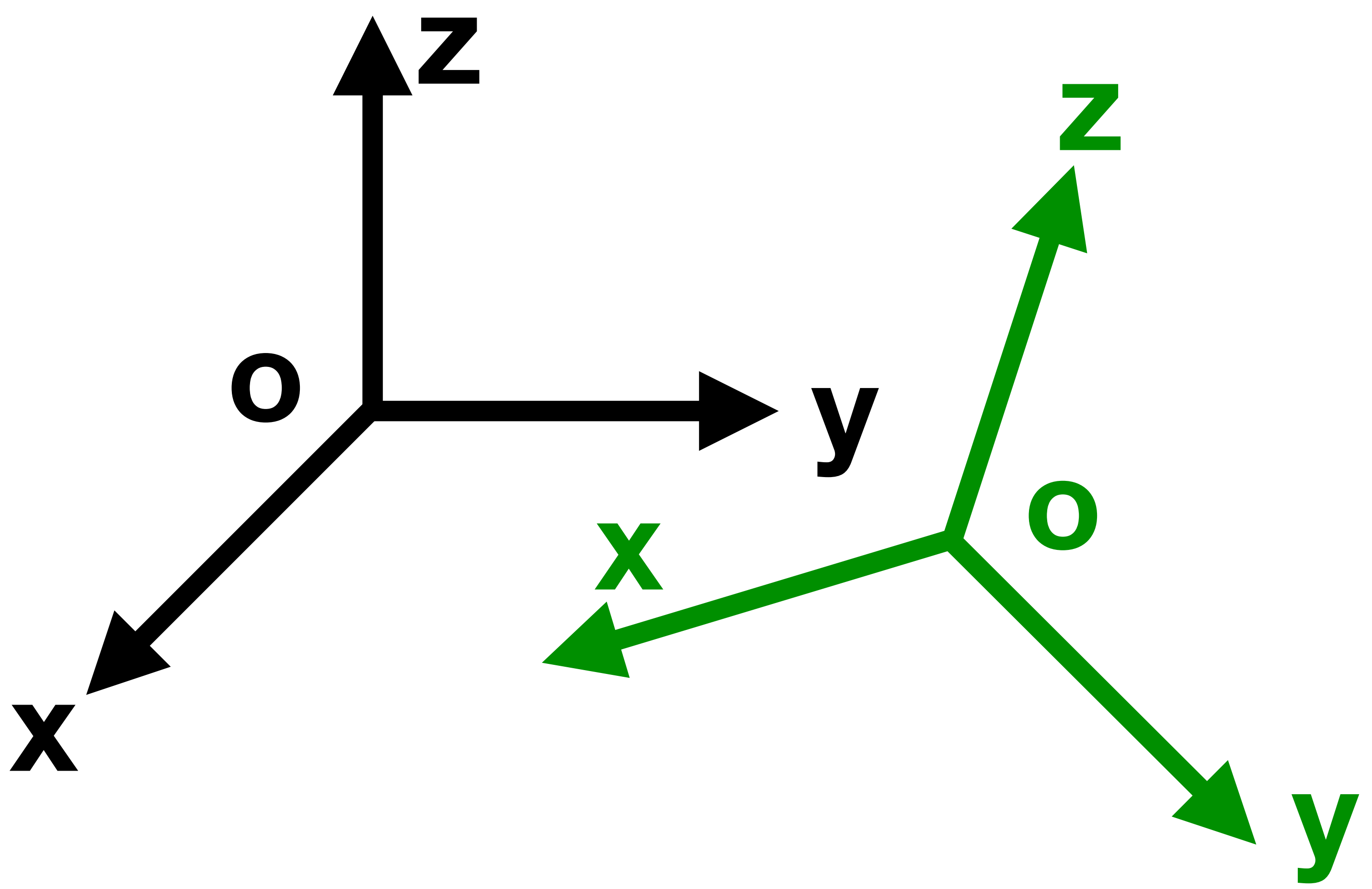
How to Transform a Frame/Coordinate System?

- One single transformation matrix can express all 4 transformations. How?

$$[1, 0, 0, 0] \times \begin{bmatrix} T_{00}, T_{01}, T_{02}, T_{03} \\ T_{10}, T_{11}, T_{12}, T_{13} \\ T_{20}, T_{21}, T_{22}, T_{23} \\ T_{30}, T_{31}, T_{32}, T_{33} \end{bmatrix} = [T_{00}, T_{01}, T_{02}, T_{03}]$$

Original x-axis

This is the new x-axis. Only the first row is used to transform x-axis.

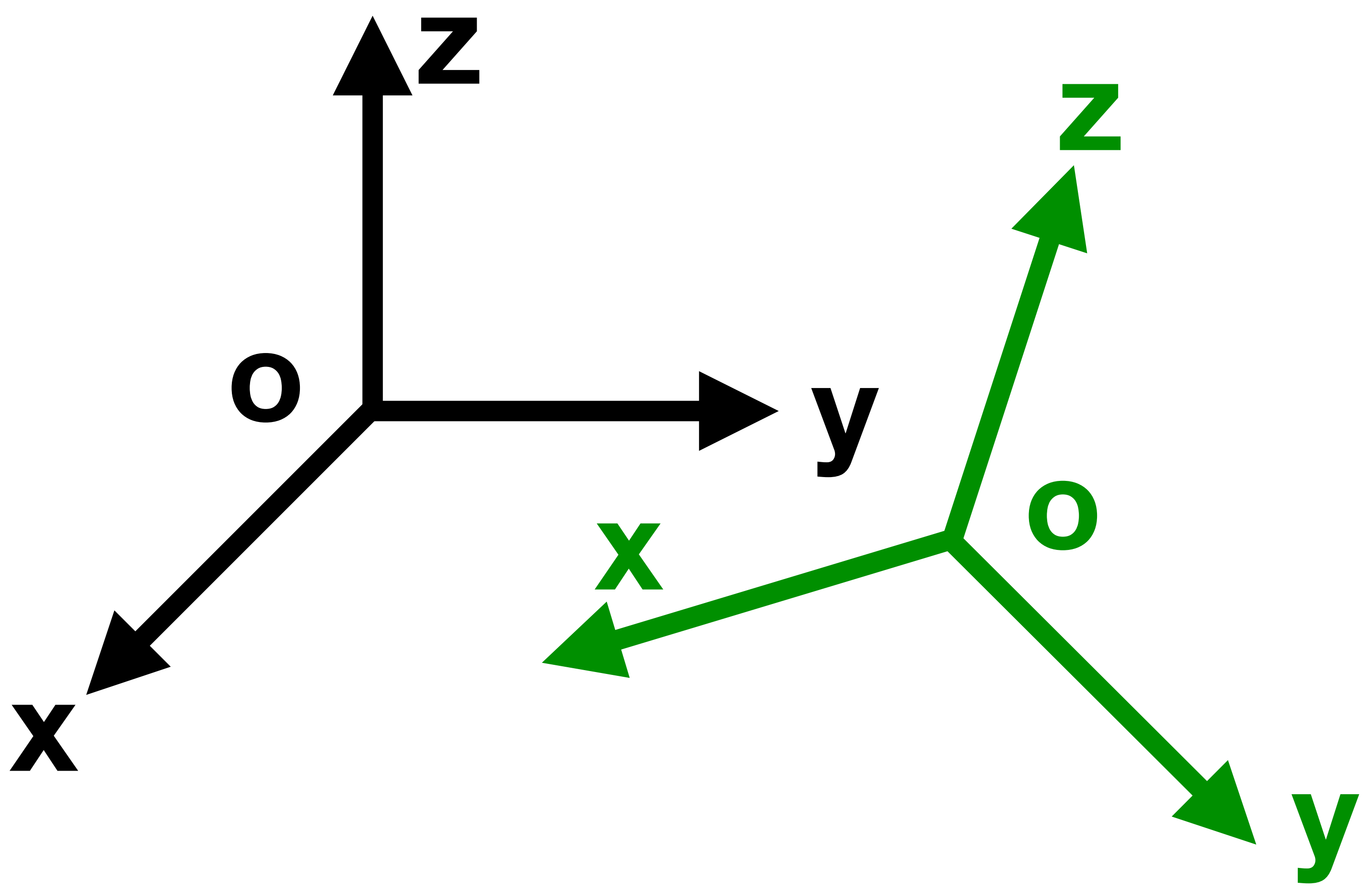


How to Transform a Frame/Coordinate System?

$$[0, 1, 0, 0] \times \begin{bmatrix} T_{00}, T_{01}, T_{02}, T_{03} \\ \boxed{T_{10}, T_{11}, T_{12}, T_{13}} \\ T_{20}, T_{21}, T_{22}, T_{23} \\ T_{30}, T_{31}, T_{32}, T_{33} \end{bmatrix} = [T_{10}, T_{11}, T_{12}, T_{13}]$$

Original y-axis

This is the new y-axis.
Only the second row is used to transform y-axis.

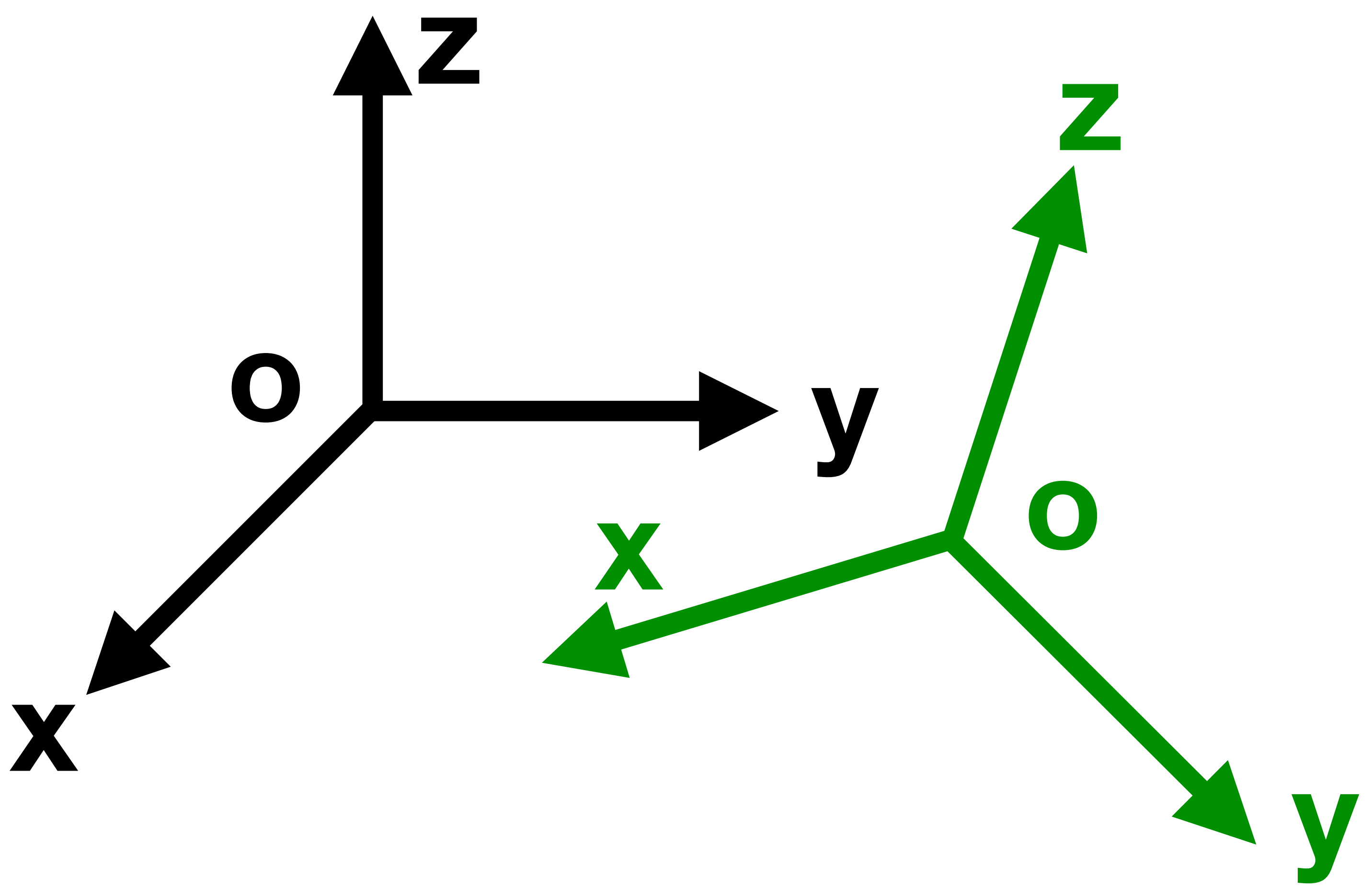


How to Transform a Frame/Coordinate System?

$$[0, 0, 1, 0] \times \begin{bmatrix} T_{00}, T_{01}, T_{02}, T_{03} \\ T_{10}, T_{11}, T_{12}, T_{13} \\ T_{20}, T_{21}, T_{22}, T_{23} \\ T_{30}, T_{31}, T_{32}, T_{33} \end{bmatrix} = [T_{20}, T_{21}, T_{22}, T_{23}]$$

Original z-axis

This is the new z-axis.
Only the third row is used to transform z-axis.

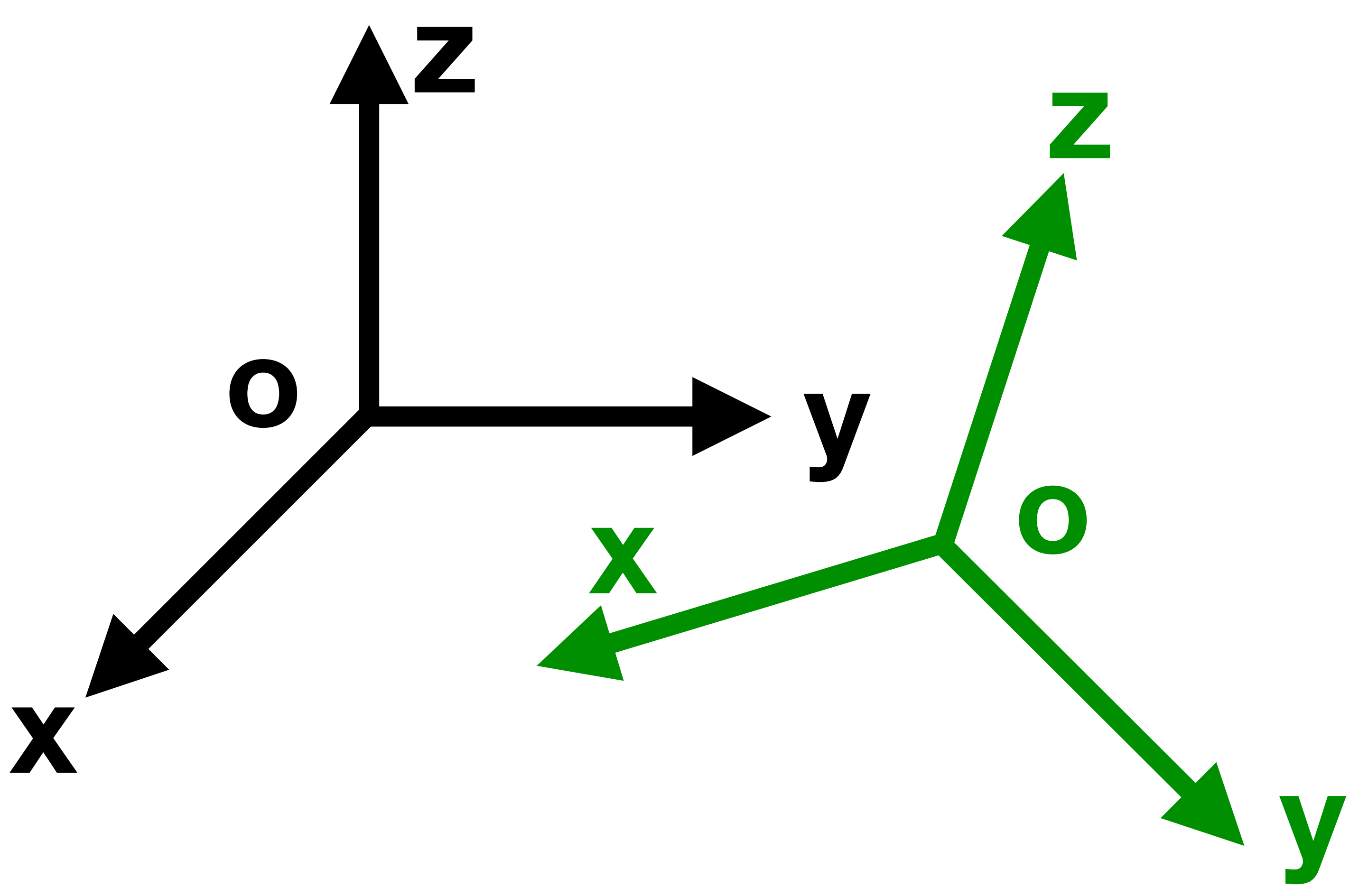


How to Transform a Frame/Coordinate System?

$$[0, 0, 0, 1] \times \begin{bmatrix} T_{00}, T_{01}, T_{02}, T_{03} \\ T_{10}, T_{11}, T_{12}, T_{13} \\ T_{20}, T_{21}, T_{22}, T_{23} \\ \boxed{T_{30}, T_{31}, T_{32}, T_{33}} \end{bmatrix} = [T_{30}, T_{31}, T_{32}, T_{33}]$$

Original origin

This is the new origin.
Only the fourth row is used to transform origin.



How to Transform a Frame/Coordinate System?

- The transformation matrix directly encodes the new basis vectors and the new origin!
- The last column needs to be $[0, 0, 0, 1]^T$
- The identity matrix basically encodes the original frame.

Deciding new basis vectors

$$\begin{bmatrix} \boxed{\begin{matrix} T_{00}, & T_{01}, & T_{02}, \\ T_{10}, & T_{11}, & T_{12}, \\ T_{20}, & T_{21}, & T_{22}, \end{matrix}} & \begin{matrix} T_{03} \\ T_{13} \\ T_{23} \end{matrix} \\ \boxed{\begin{matrix} T_{30}, & T_{31}, & T_{32}, \end{matrix}} & T_{33} \end{bmatrix} \begin{matrix} \leftarrow \text{New x-axis basis vector} \\ \leftarrow \text{New y-axis basis vector} \\ \leftarrow \text{New z-axis basis vector} \\ \leftarrow \text{New origin} \end{matrix}$$

Deciding new origin

All in homogeneous coordinates

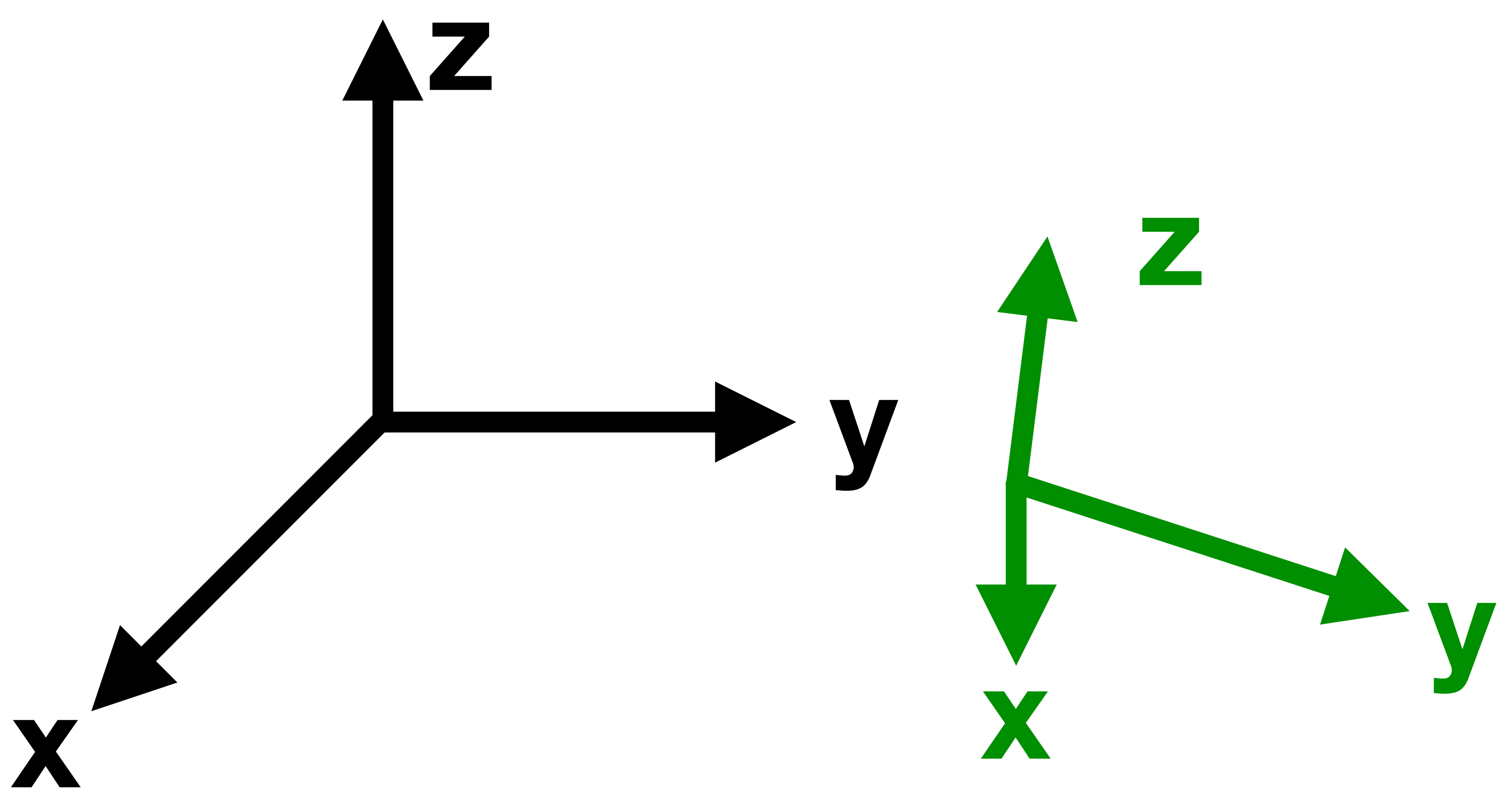
Identity matrix encodes the canonical frame's information!

$$\begin{bmatrix} 1, 0, 0, 0 \\ 0, 1, 0, 0 \\ 0, 0, 1, 0 \\ 0, 0, 0, 1 \end{bmatrix}$$

How to Transform a Frame/Coordinate System?

- Does any transformation matrix work?
- Yes, but for the transformed frame to be used as a Cartesian coordinate system, the top 3x3 matrix must be an **orthogonal matrix**: the three basic vectors must be mutually orthogonal and their lengths must be 1.
- Intuition: an orthogonal matrix rotates the three basis vector together, so mutual orthogonality and unit length requirements are naturally met.

$$\begin{bmatrix} 2, 4, 0, 0 \\ 1, 5, -1, 0 \\ -1, 0, 1, 0 \\ 1, 4, 0, 1 \end{bmatrix}$$

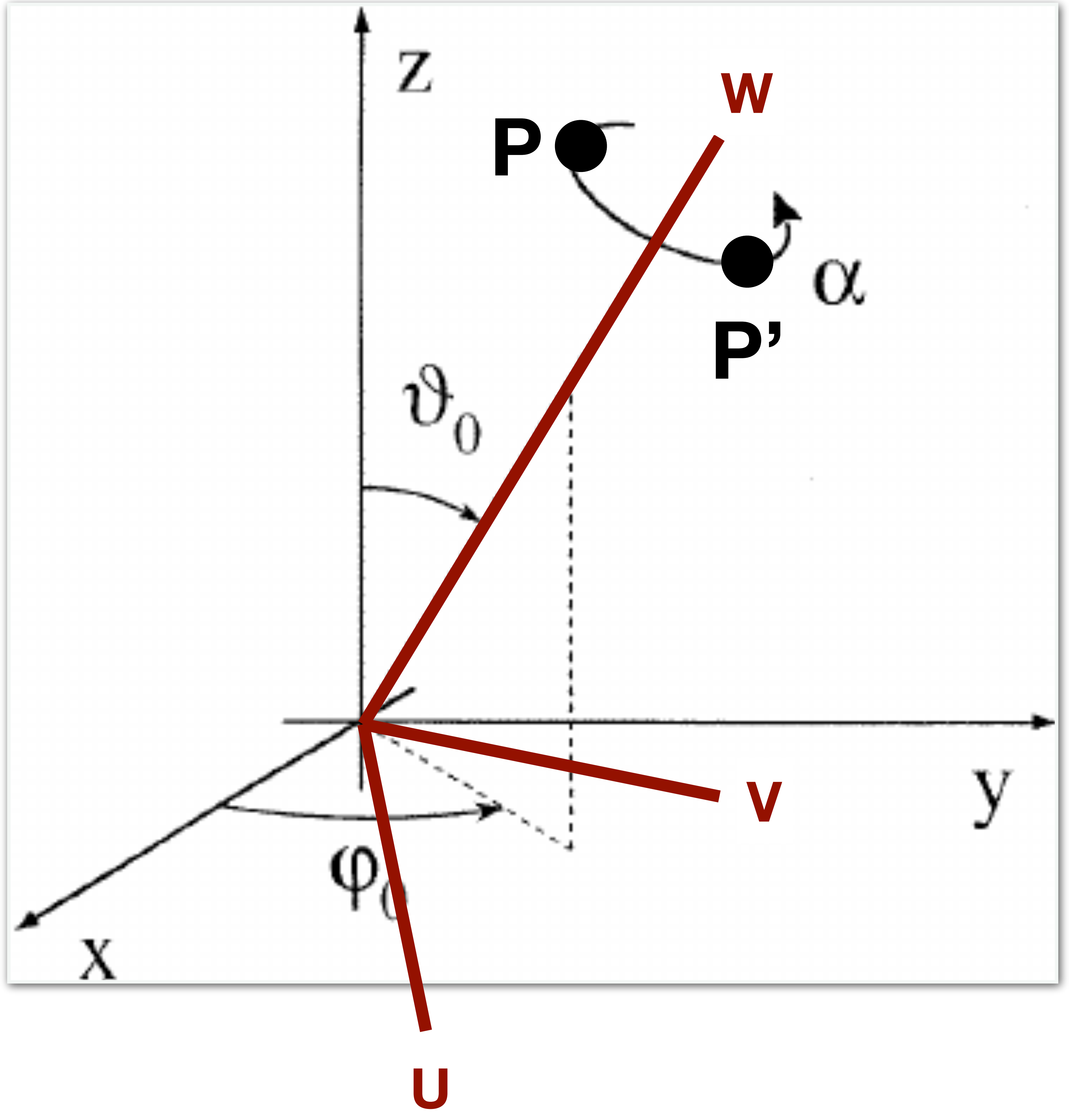


A legal transformation of the frame, but the new frame can’t be used as a Cartesian coordinate system.

Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?

$$P' = P \times R_1 \times R_2 \times R_1^{-1}$$

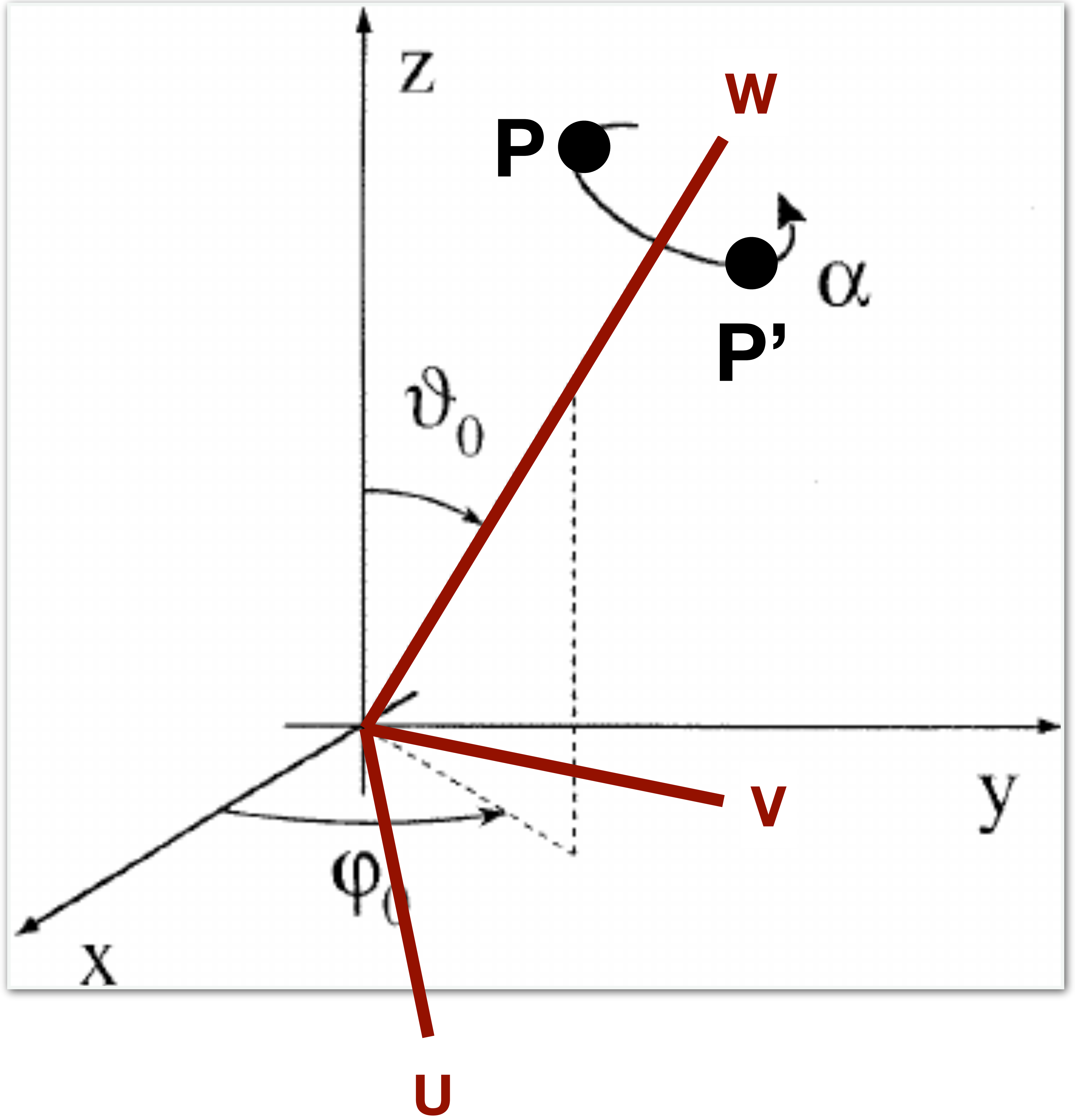


https://www.researchgate.net/figure/Notation-used-for-the-analysis-of-the-rotation-around-an-arbitrary-axis-r_fig5_230934580

Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.

$P' = P \times R_1 \times R_2 \times R_1^{-1}$

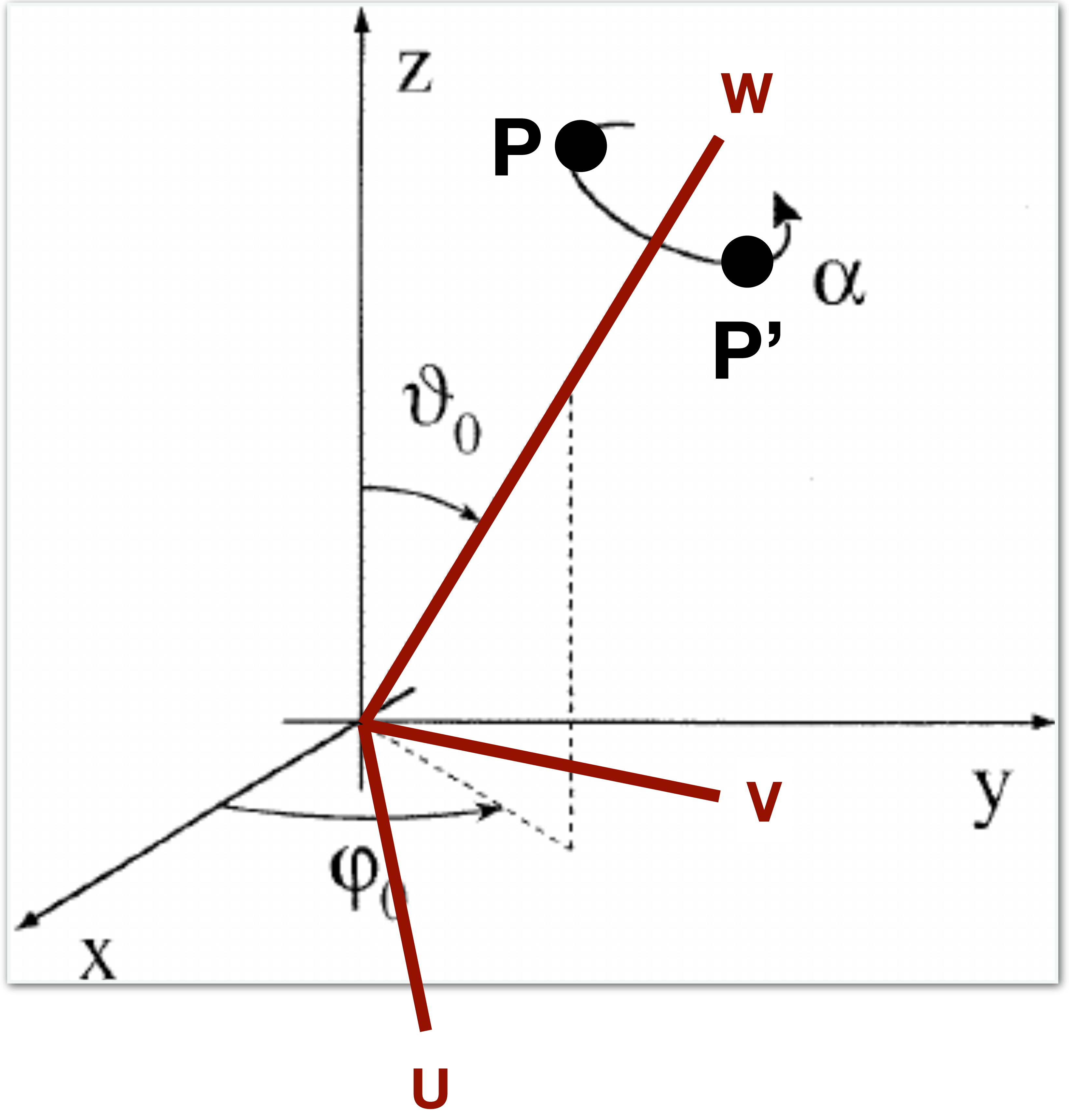


https://www.researchgate.net/figure/Notation-used-for-the-analysis-of-the-rotation-around-an-arbitrary-axis-r_fig5_230934580

Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.
- Second, rotate **UVW** to be **XYZ**; let the rotation matrix be R_1. P becomes P1.

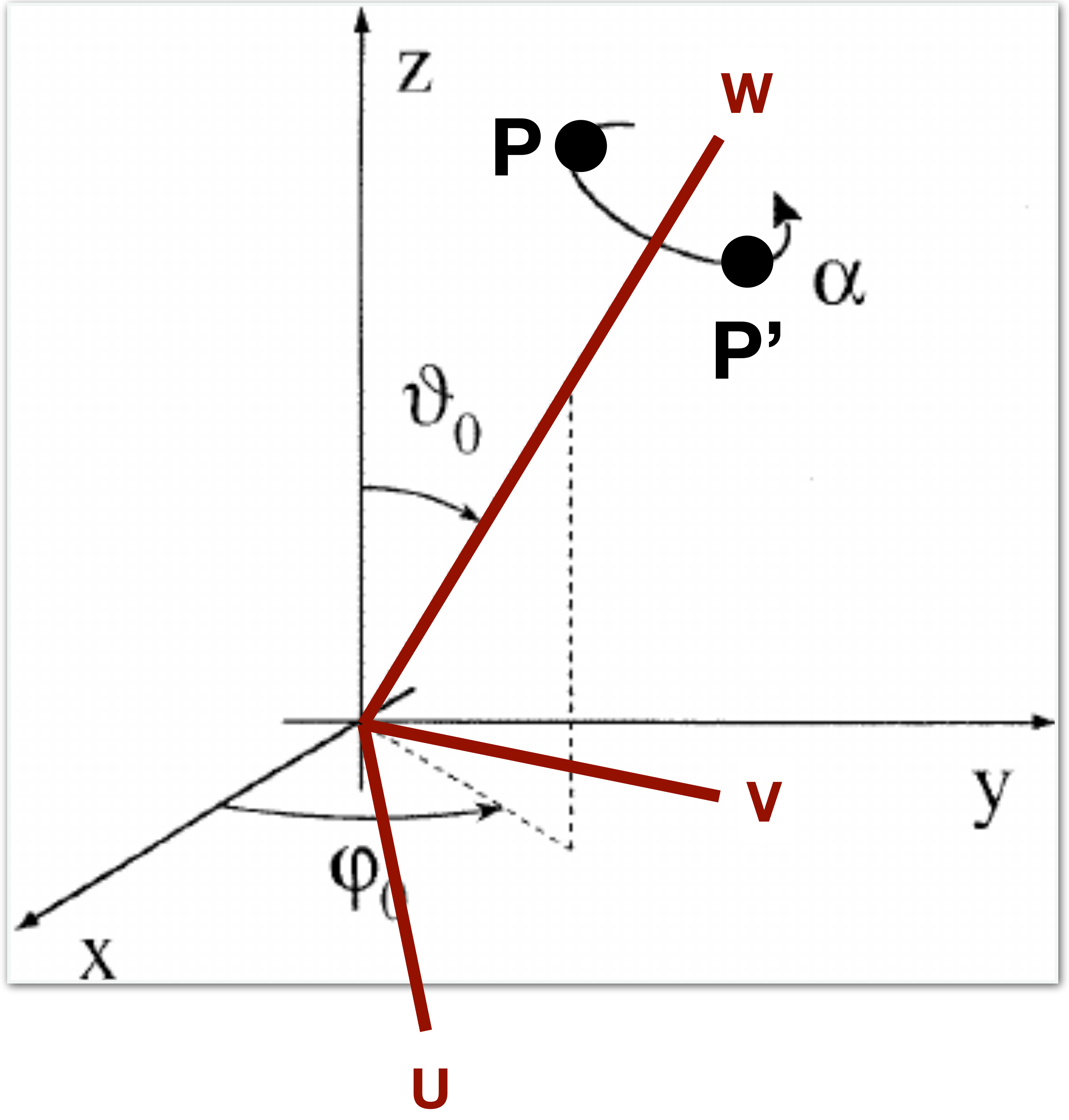
$$P' = P \times R_1 \times R_2 \times R_1^{-1}$$



Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.
- Second, rotate **UVW** to be **XYZ**; let the rotation matrix be R_1. P becomes P1.
- Third, rotate P1 around Z. Let the rotation matrix be R_2. P1 becomes P2.

$$P' = P \times R_1 \times R_2 \times R_1^{-1}$$

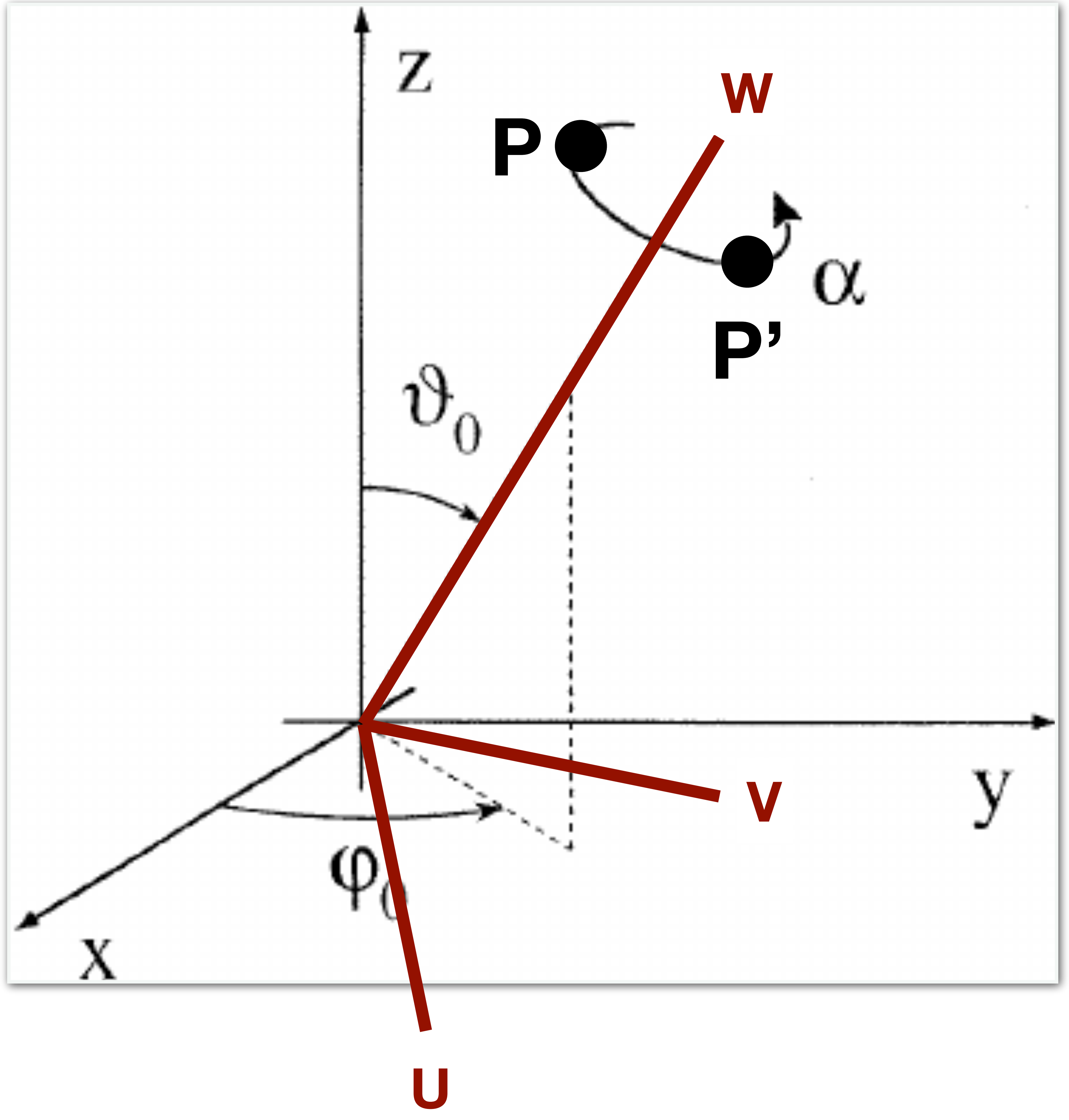


https://www.researchgate.net/figure/Notation-used-for-the-analysis-of-the-rotation-around-an-arbitrary-axis-r_fig5_230934580

Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.
- Second, rotate **UVW** to be **XYZ**; let the rotation matrix be R_1. P becomes P1.
- Third, rotate P1 around Z. Let the rotation matrix be R_2. P1 becomes P2.
- Finally, rotate P2 from **XYZ** to **UVW** to get P'. The rotation matrix is necessarily R_1^{-1} which is R_1^T since R_1 is necessarily orthogonal.

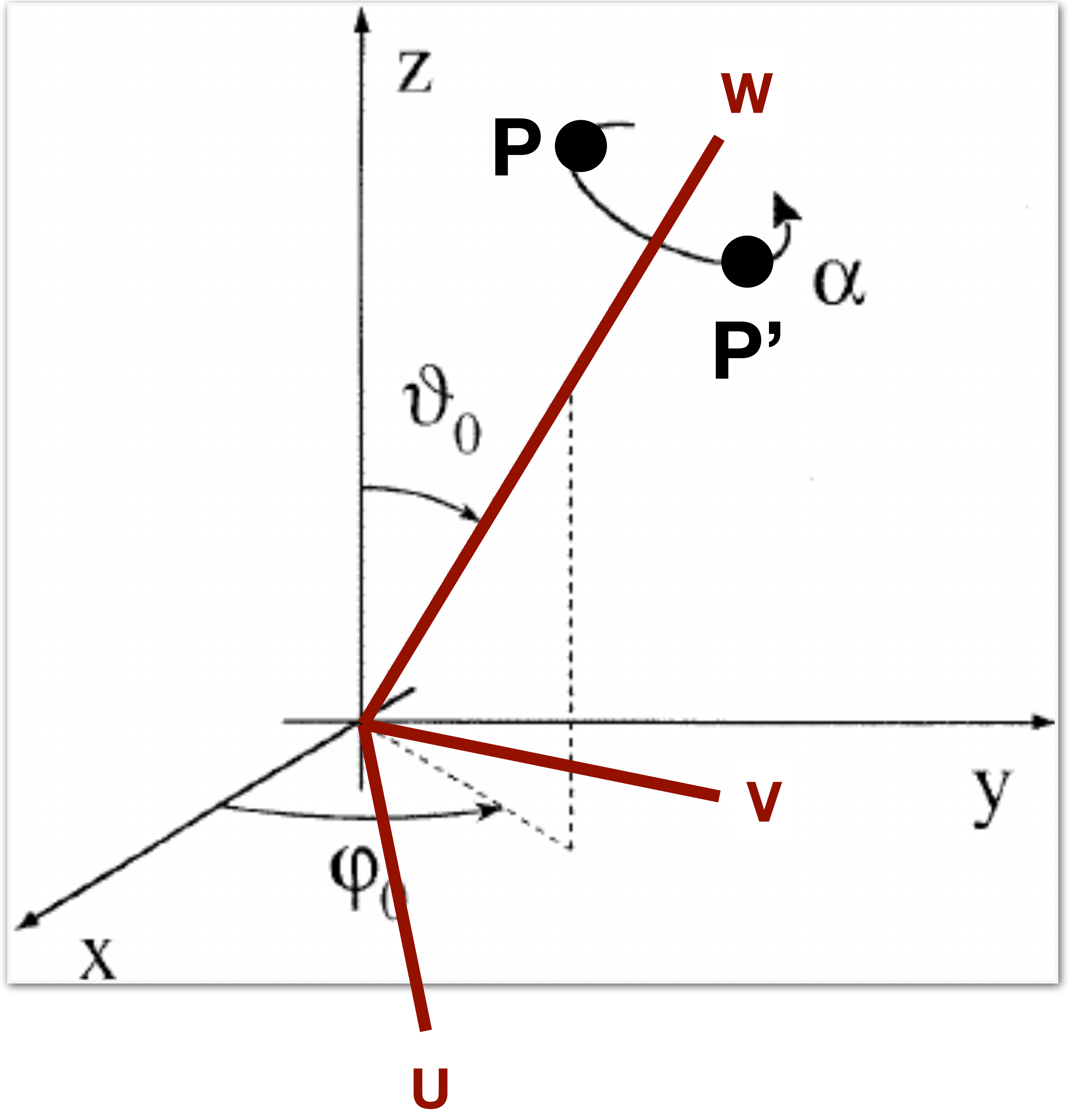
$$P' = P \times R_1 \times R_2 \times R_1^{-1}$$



Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?

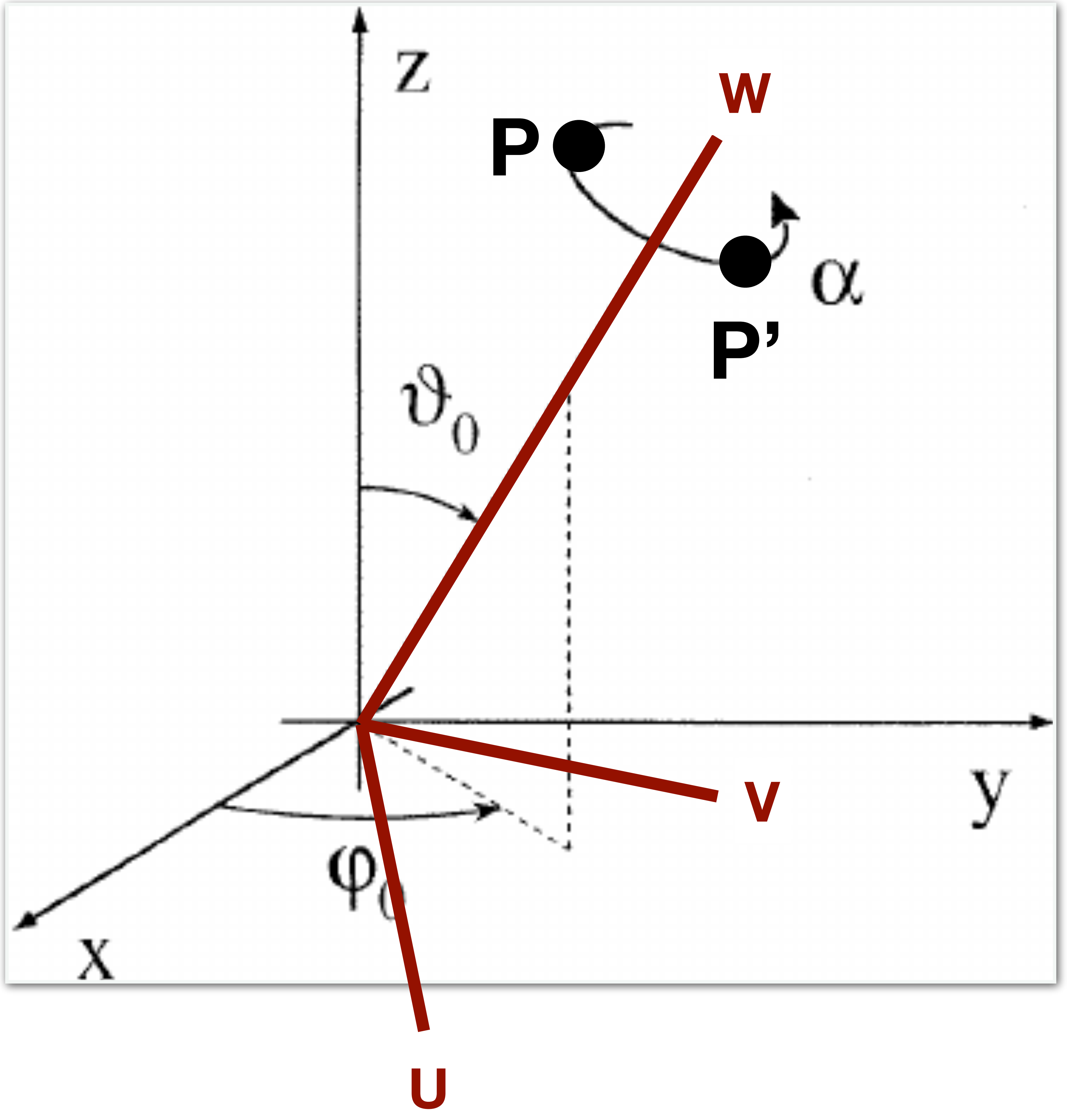
$$P' = P \times R_1 \times R_2 \times R_1^{-1}$$



Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.

$P' = P \times R_1 \times R_2 \times R_1^{-1}$

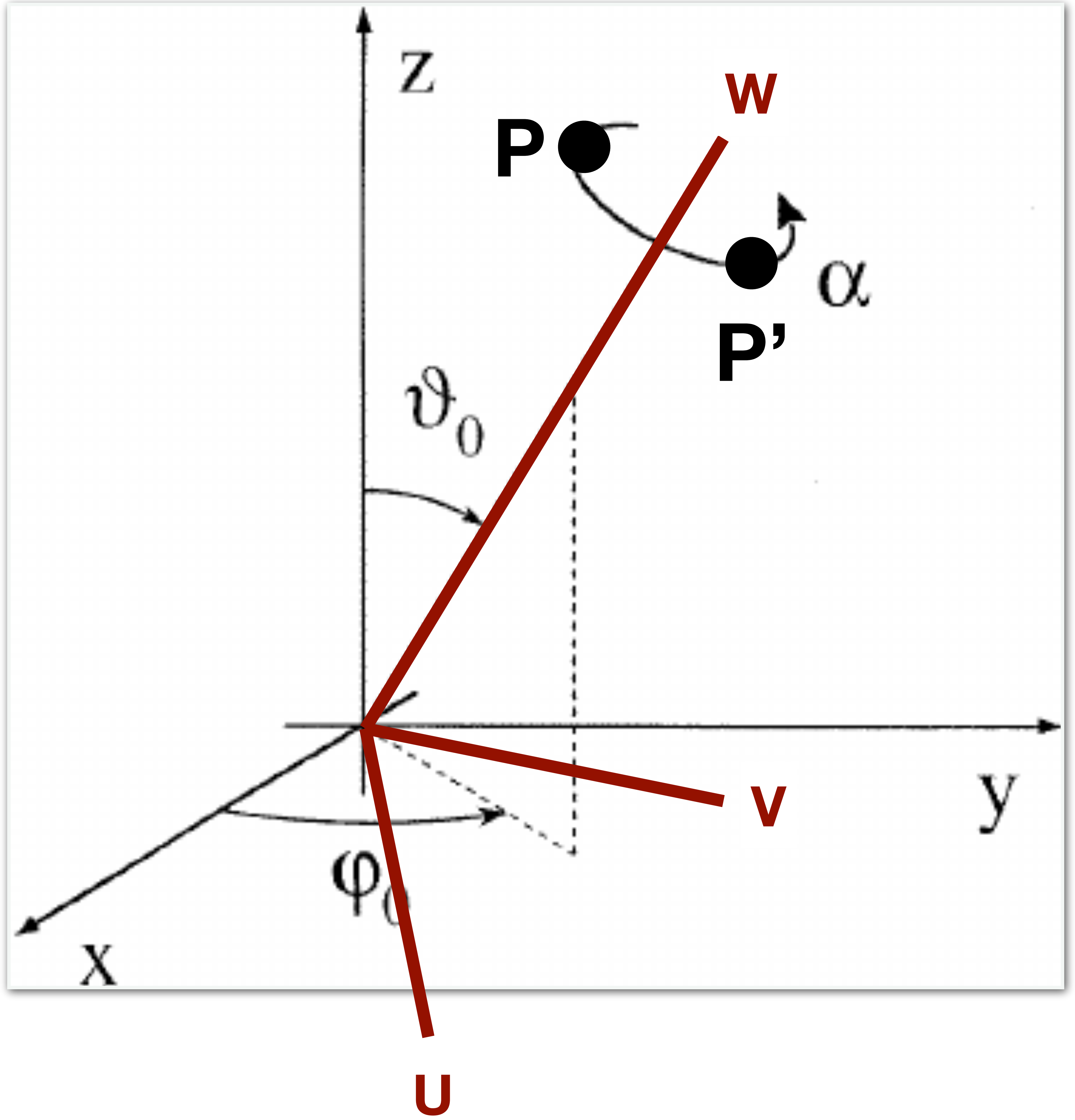


https://www.researchgate.net/figure/Notation-used-for-the-analysis-of-the-rotation-around-an-arbitrary-axis-r_fig5_230934580

Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.
- Second, rotate **UVW** to be **XYZ**; let the rotation matrix be R_1. P becomes P1.

$$P' = P \times R_1 \times R_2 \times R_1^{-1}$$

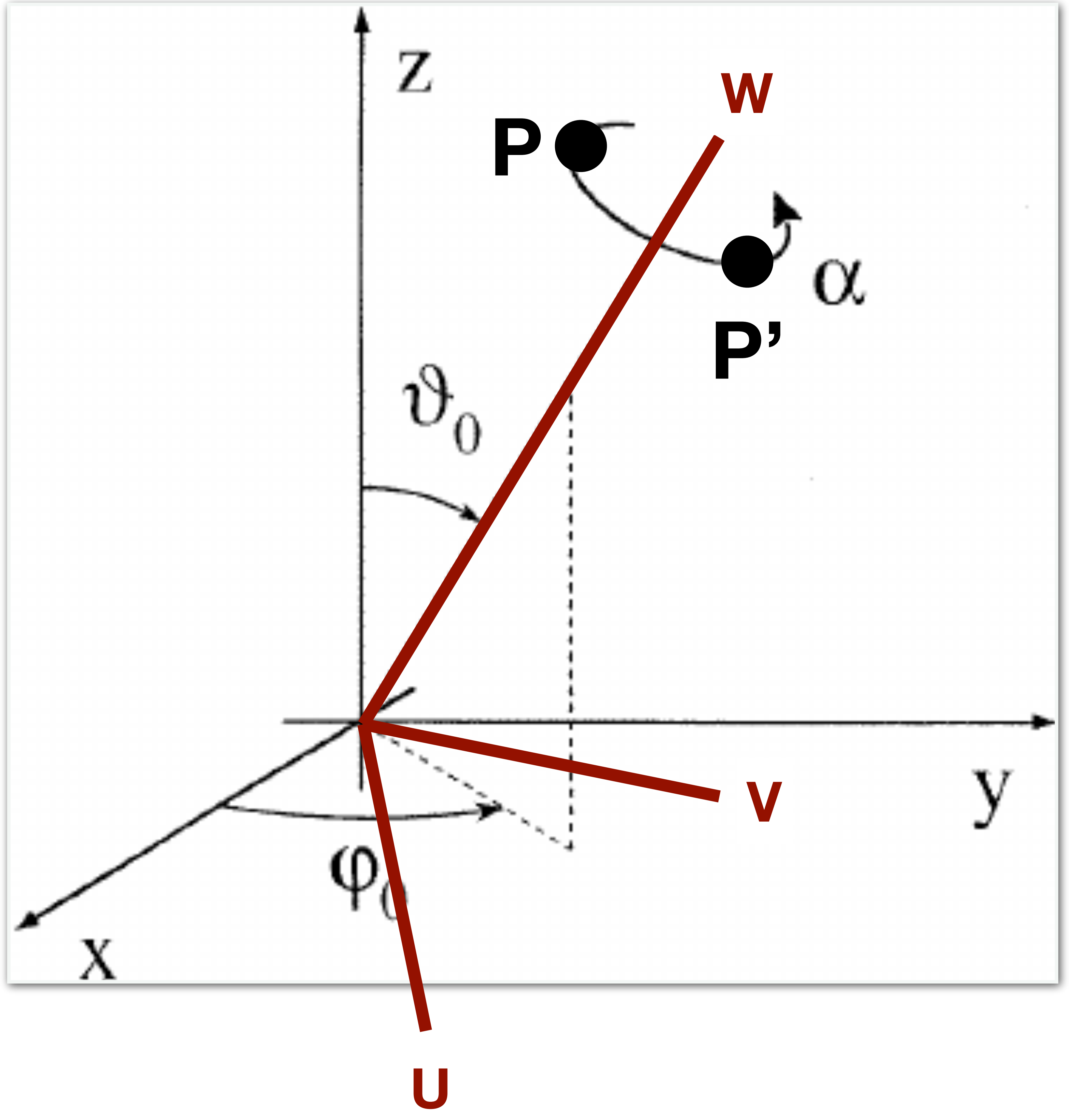


https://www.researchgate.net/figure/Notation-used-for-the-analysis-of-the-rotation-around-an-arbitrary-axis-r_fig5_230934580

Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.
- Second, rotate **UVW** to be **XYZ**; let the rotation matrix be R_1. P becomes P1.
- Third, rotate P1 around Z. Let the rotation matrix be R_2. P1 becomes P2.

$$P' = P \times R_1 \times R_2 \times R_1^{-1}$$



Rotation Around an Arbitrary Axis

- How to rotate around an arbitrary vector **w**?
- First, create a Cartesian coordinate system **UVW**. There are infinite many (only **W** is given); any one will work in principle.
- Second, rotate **UVW** to be **XYZ**; let the rotation matrix be R_1. P becomes P1.
- Third, rotate P1 around Z. Let the rotation matrix be R_2. P1 becomes P2.
- Finally, rotate P2 from **XYZ** to **UVW** to get P'. The rotation matrix is necessarily R_1^{-1} which is R_1^T since R_1 is necessarily orthogonal.

$P' = P \times R_1 \times R_2 \times R_1^{-1}$