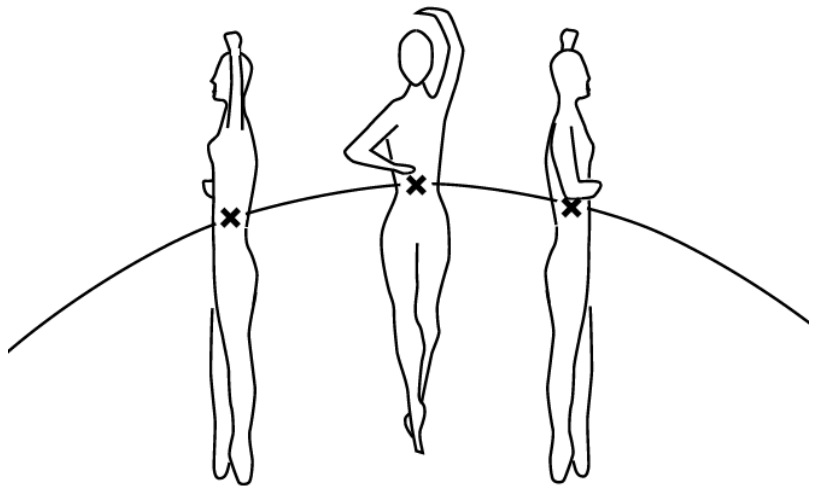
Basics for Enhanced Visualization: 3D/Data

Transformations in 2D/3D



Rodrigo Cabral

Polytech Nice - Data Science

cabral@unice.fr

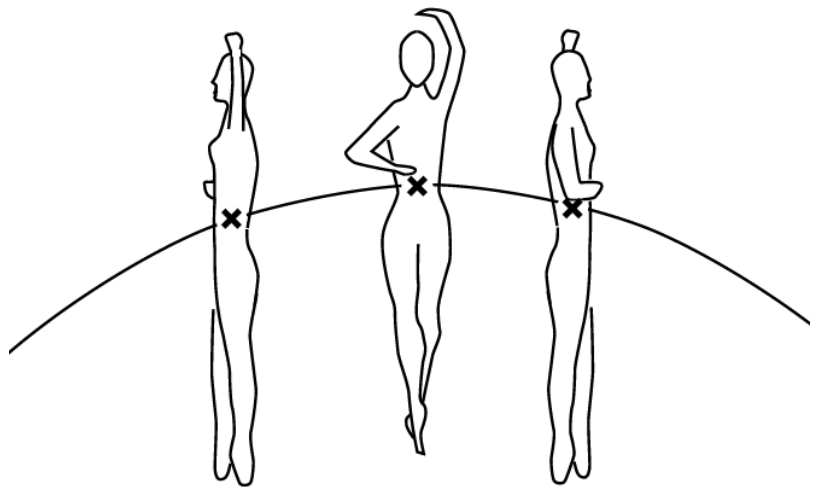
Source: http://www.lightandmatter.com/lm/

Outline

These slides are partly inspired from Lionel Fillatre's slides.

What are and why do we need 3D transformations?

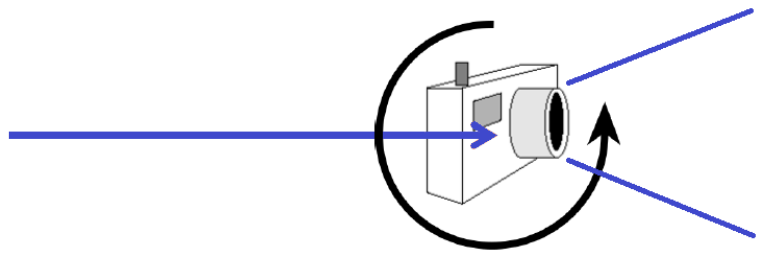
- Moving objects translate **T** and rotate **R** in space, with respect to a reference. Reference is called **world reference or world frame**.



- Objects can also deform. But here we consider mainly rigid body objects.

What are and why do we need 3D transformations in static scenes?

- The scene is observed by a camera with position, orientation and an effective size with respect to the world reference. This reference is known as **camera reference or camera frame**.
- Using the camera frame as a reference equals to a translation **T**, rotation **R** and scaling **S** of the points forming the object.

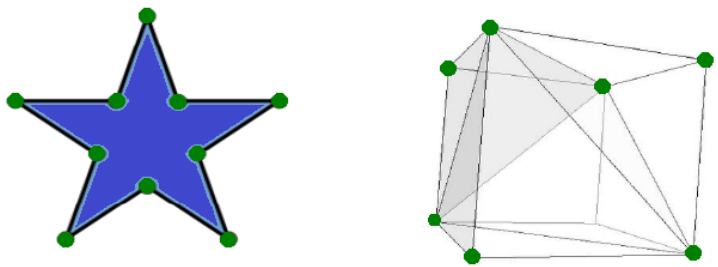


- Transformations from 3D to 2D known as projections are also required. Next class.

Introduction

Computer based approach

Lowest level description of an object is given by its vertices.



Objective: mathematically define the points and the transformations.

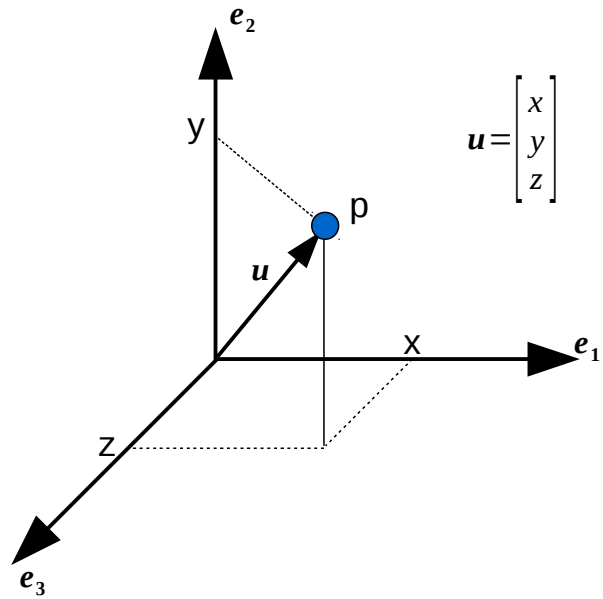
Solution: linear algebra.

Euclidean space and cartesian coordinates

A point **p** in the 3D (2D) Euclidean space is analytically represented in Cartesian coordinates by

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \in \mathbb{R}^3 \qquad (\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \in \mathbb{R}^2)$$

A vector as in linear algebra.



Keep in mind that there exist geometric objects called vectors. We will see later how to differentiate between both.

Inner product, norm and angle

Note that we choose as basis vectors for the reference frame

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ and } \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Define a metric with the inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle \triangleq \mathbf{u}^{\mathrm{T}} \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3$$

This allows to evaluate:

- Distances to the origin : Norm - $\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$
- Distances between points : $\|\mathbf{u} - \mathbf{v}\|$
- Angles with respect to the origin: $\cos(\theta) = \dfrac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\| \|\mathbf{v}\|}$

Orthogonal vector to plane: Cross-product - $\mathbf{u} \times \mathbf{v} = \mathbf{U}_{\times} \mathbf{v}$

$$\mathbf{U}_{\times} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$

Linear transformations

Definition: linear transformation

A linear transformation (or linear function) T of a vector is a function respecting two properties:

- Additivity: $T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$
- Homogeneity: $T(a\mathbf{u}) = aT(\mathbf{u})$

Linear transformations

Theorem: transformations and linear algebra

Any linear transformation T can be represented by the matrix-vector product $T(\mathbf{u}) = \mathbf{A}\mathbf{u}$.

Proof: $\mathbf{A}$ can be constructed by applying the transformation to the canonical basis $\mathbf{e}_1, \cdots, \mathbf{e}_N$ and using the two properties.

Moreover the columns of $\mathbf{A} = [\mathbf{a}_1 | \cdots | \mathbf{a}_N]$ are directly the transformed canonical basis $\mathbf{e}_1, \cdots, \mathbf{e}_N$.

Affine transformations

Similarly to linear transformations **affine transformations** T can be represented by a linear transformation plus a fixed vector **b**:

$$T(\mathbf{u}) = \mathbf{A}\mathbf{u} + \mathbf{b}.$$

- Affine transformations include linear ones.
- Is the origin invariant to linear and affine transformations?

Rigid body transformations

Definition: rigid body transformation

A rigid body transformation $\mathbf{m} : \mathbb{R}^3 \to \mathbb{R}^3$ satisfies the following properties for vectors $\mathbf{u}$ and $\mathbf{v}$, representing any two points ($\mathbf{p}$ and $\mathbf{q}$ for example) and $\mathbf{w} = \mathbf{u} - \mathbf{v}$, $\mathbf{x}$ representing differences between points:

- Distance is preserved: $\|\mathbf{u} - \mathbf{v}\| = \|m(\mathbf{u}) - m(\mathbf{v})\|$
- Cross product is preserved: $m(\mathbf{w} \times \mathbf{x}) = m_\star(\mathbf{w}) \times m_\star(\mathbf{x})$

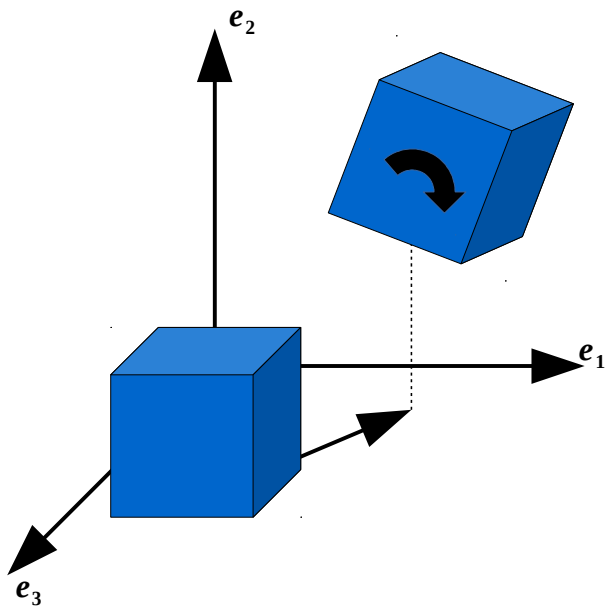
where $m_\star(\mathbf{w}) = m(\mathbf{u}) - m(\mathbf{v})$

Consequences: rigid body transformations do not change the shape neither the volume of an object.

Rigid body transformations

Rotations and translations

(like the "rigid ballet dancer")



General form: input vector **u**

$$\mathbf{y} = \mathbf{R}\mathbf{u} + \mathbf{t}$$

- **R** is a square orthogonal matrix with $\det(\mathbf{R}) = 1$.
- **t** is a vector.
- Rigid body transformations are affine transformations.

Linear transformations - 2D

2D transformation

- Vector **u** is a point in 2D space.
- u_1 is the coordinate in $\mathbf{e}_1$ - standard **x** axis coordinate.
- u_2 is the coordinate in $\mathbf{e}_2$ - standard **y** coordinate.

$$T(\mathbf{u}) = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

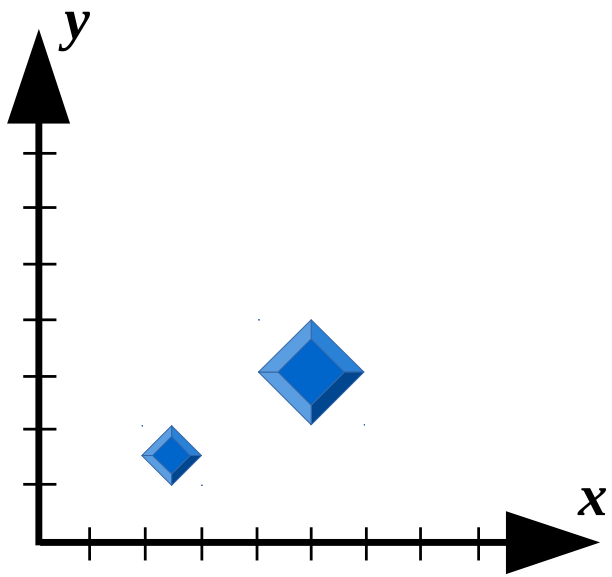
- $T(\mathbf{e}_1) = \begin{bmatrix} a \\ b \end{bmatrix}$ - transformation of a unitary vector in x axis.
- $T(\mathbf{e}_2) = \begin{bmatrix} c \\ d \end{bmatrix}$ - transformation of a unitary vector in y axis.

Linear transformations - 2D

Scaling

- First coordinate is scaled by factor s_x.
- Second coordinate is scaled by factor s_y.

- $\mathbf{T}(\mathbf{e}_1) = \begin{bmatrix} s_x \\ 0 \end{bmatrix}$ and $\mathbf{T}(\mathbf{e}_2) = \begin{bmatrix} 0 \\ s_y \end{bmatrix} \Longrightarrow \mathbf{T} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$



- Changes the size of the object
- But also translates the object if not in origin.
- It does not preserve distances.
- In general, it does not preserve angles.

Linear transformations - 2D

Scaling changes shape: not a rigid body transformation.

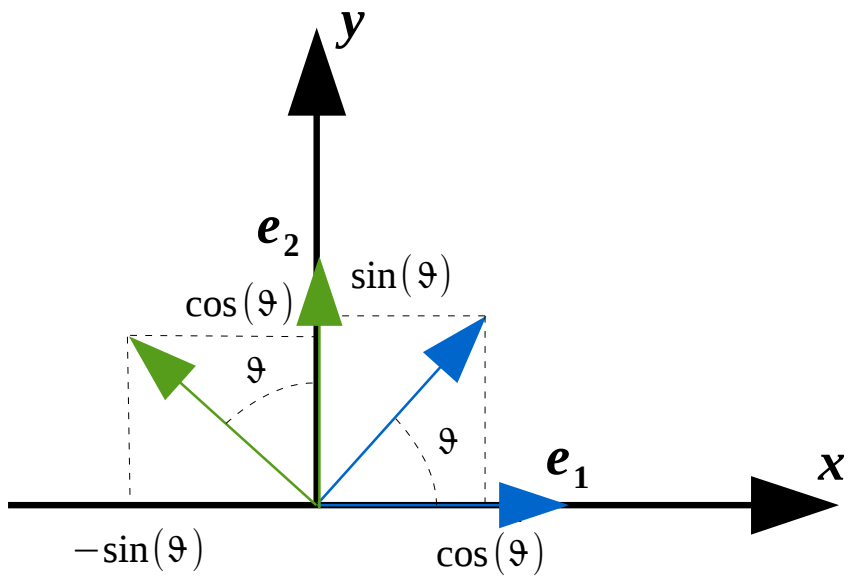
Questions:

- Why a scaling is not a rigid body transformation mathematically?
- Is the origin invariant in linear transformations?

Linear transformations - 2D

Rotation

- Rotation about origin by an angle θ.

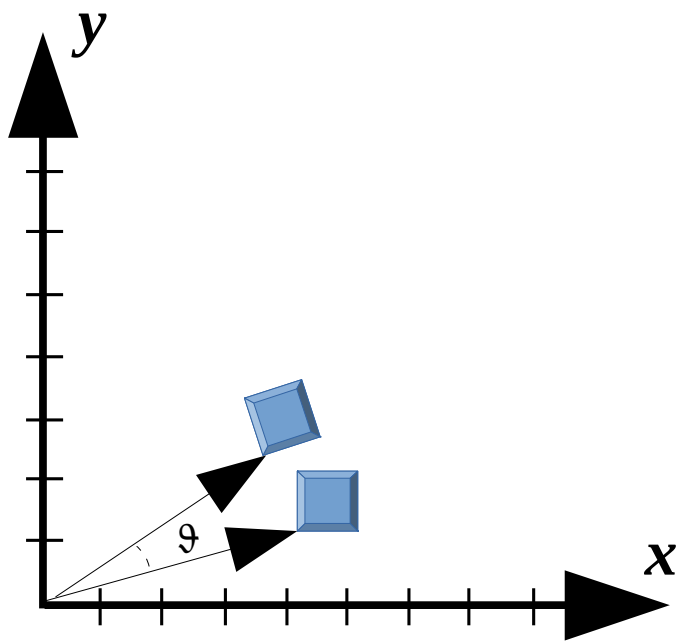


- $\mathbf{T}(\mathbf{e}_1) = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$ and $\mathbf{T}(\mathbf{e}_2) = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix}$

$$\Longrightarrow \mathbf{T} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

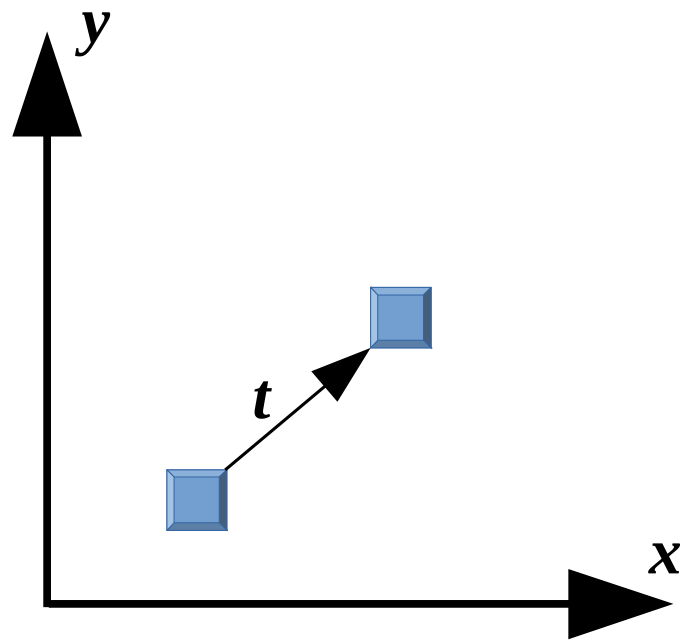
Linear transformations - 2D

- Rotations are rigid body transformations. (Proof?)
- To rotate around the center of the object a translation is required.
- Inversion is easy. (In general, what is the inverse of a rigid body transformation?)
- How to encode a sequence of rotations and scalings?



2D Translation and homogeneous coordinates

Translation



- $T(\mathbf{u}) = \mathbf{u} + \mathbf{t} = \begin{bmatrix} u_1 + t_1 \\ u_2 + t_2 \end{bmatrix}$: an affine transformation $\longrightarrow$ not linear.
- We cannot compose it with other types of transformation.
- If we want a linear framework for rigid body transformations (plus scaling)$\longrightarrow$ what do we do? ☹

Homogeneous coordinates

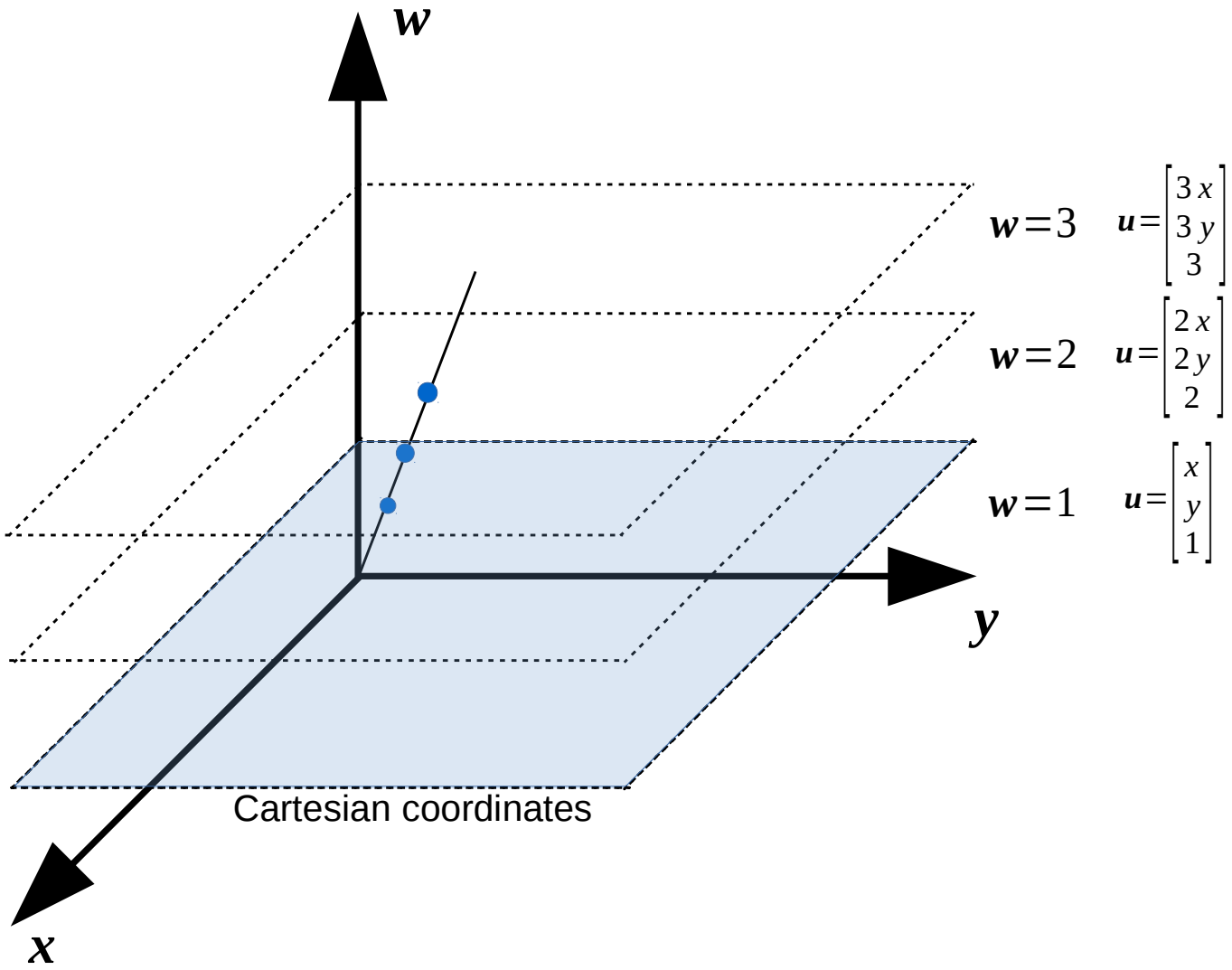
- We add a dimension corresponding to a **w** axis. This generates a supplementary coordinate.
- The 2D space is now embedded in the 3D space.
- But how to specify this third coordinate?
- Add a constant coordinate, 1 for example:

$$u_c = \begin{bmatrix} x \\ y \end{bmatrix} \Longrightarrow u_h = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Furthermore, points $\mathbf{u}_h = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$ for any $w \neq 0$ are equivalent.

2D Translation and homogeneous coordinates

Homogeneous coordinates



2D Translation and homogeneous coordinates

Homogeneous coordinates

- If you have a point in homogeneous coordinates $\mathbf{u}_h = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}$ ($w' \neq 0$) you can get Cartesian coordinates as follows

$$\mathbf{u}_h = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Longrightarrow \begin{bmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \\ 1 \end{bmatrix} \Longrightarrow \mathbf{u}_c = \begin{bmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \end{bmatrix}$$

2D Translation and homogeneous coordinates

Linear transformations and Homogeneous coordinates

To keep w unchanged ($w = 1$ for example) we can define linear transformations in homogeneous coordinates as follows:

$$T_c = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \Longrightarrow \quad T_h = \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- T_c is the transformation we want to apply in Cartesian coordinates.
- T_h is its corresponding version in homogeneous coordinates
- This works for all linear transformations: scaling, rotations *etc*.

2D Translation and homogeneous coordinates

How to do a translation?

Now we can do it!

For a translation $\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$ we have

$$T_c = \begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix} \quad \Longrightarrow \quad T_h = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

- Verify it with a vector $\mathbf{u}_h = \begin{bmatrix} u_1 \\ u_2 \\ 1 \end{bmatrix}$!

2D Translation and homogeneous coordinates

Homogenized transformations

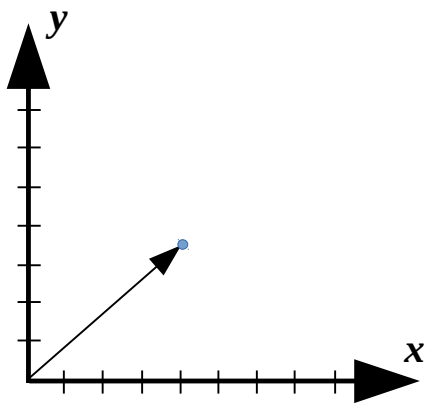
All previous transformations in homogenized form:

Transformation	Matrix
Scaling	$\mathbf{S} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Rotation	$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Translation	$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$

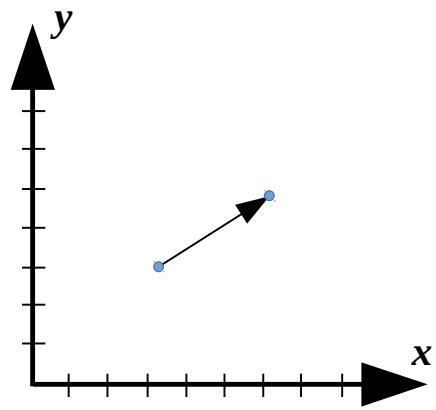
2D Translation and homogeneous coordinates

Brief comment on geometric vectors

A point



A vector



- A point is represented by $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & \mathbf{1} \end{bmatrix}^{\mathrm{T}}$ in homogeneous coordinates.
- A vector starting on a point cannot be translated. We define it as $\mathbf{v} = \begin{bmatrix} u_1 & u_2 & \mathbf{0} \end{bmatrix}^{\mathrm{T}}$.
- The last coordinate makes geometric vectors unchanged by translation.
- Example : normal vectors in lighting rendering.

Composition of transformations

Composition of functions

- How can we combine multiple transformations?
- How do we describe linear transformations sequentially with functions?

- Transformations are functions.
- Functions can be composed sequentially: $f \circ g(\mathbf{u}) = f(g(\mathbf{u}))$
- If we want to apply first $\mathbf{T}_1$ and then $\mathbf{T}_2 \Longrightarrow$

Matrix product: $\mathbf{T}_{12} = \mathbf{T}_2\mathbf{T}_1$.

Composition of transformations

Matrix product

- We are able to make more complex transformations using composition.
- Example: first scale **S**, then rotate **R** and finally translate **T** a point $\mathbf{u} = \begin{bmatrix} x & y & 1 \end{bmatrix}^{\mathrm{T}}$:

$$\mathbf{u}' = \mathbf{TRSu} =$$

$$\begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

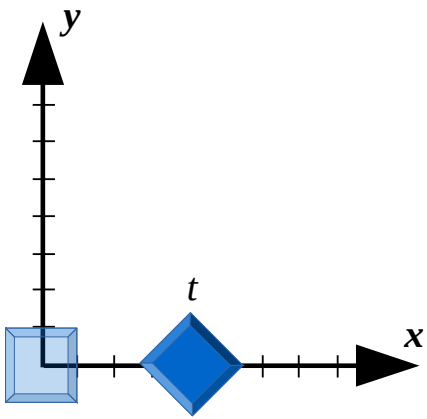
- Composition is read from **right to left**.

Composition of transformations

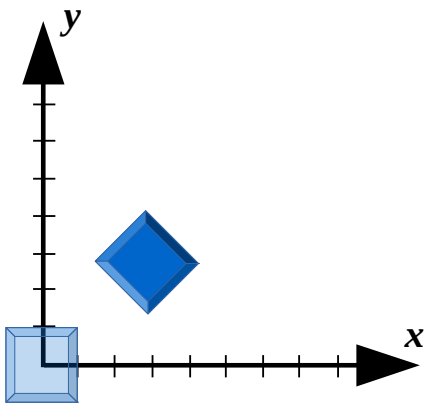
Non commutativity

- As any composition, **order matters**: matrix multiplication is not commutative.

Rotate 45° $\Longrightarrow$ Translate $\begin{bmatrix} t_1 \\ 0 \end{bmatrix}$



Translate $\begin{bmatrix} t_1 \\ 0 \end{bmatrix}$ $\Longrightarrow$ Rotate 45°



Composition of transformations

A few questions:

- Do an inverse exist for each of the previously presented transformations in homogeneous coordinates?
- What are the inverses of these transformations in homogeneous coordinates?
- How do you use translations and inverses to write the rotation of an object around its center?

3D transformations

Transformations in homogeneous coordinates (4 coord.):

Transformation	Matrix
Scaling	$\mathbf{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rotation	They can be defined around any axis. (see next slides)
Translation	$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

3D transformations

Rodrigues' formula

Rotation by angle θ around unit vector $\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$:

$$\mathbf{R}_{\mathbf{v},\theta} =$$

$$\begin{bmatrix} \cos(\theta) + v_x^2[1-\cos(\theta)] & v_x v_y[1-\cos(\theta)] - v_z\sin(\theta) & v_x v_z[1-\cos(\theta)] + v_y\sin(\theta) & 0 \\ v_x v_y[1-\cos(\theta)] + v_z\sin(\theta) & \cos(\theta) + v_y^2[1-\cos(\theta)] & v_y v_z[1-\cos(\theta)] - v_x\sin(\theta) & 0 \\ v_x v_z[1-\cos(\theta)] - v_y\sin(\theta) & v_y v_z[1-\cos(\theta)] + v_x\sin(\theta) & \cos(\theta) + v_z^2[1-\cos(\theta)] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D transformations

Rotation around each axis (Euler's angles)

Axis, Angle	Matrix
x, ψ	$\mathbf{R}_{x,\psi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) & 0 \\ 0 & \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
y, θ	$\mathbf{R}_{x,\psi} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
z, ϕ	$\mathbf{T} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

3D transformations

Rotation by θ around axis $\mathbf{v}$ with composition

1. Align $\mathbf{v}$ with x axis using rotations:
   1.1 rotate around y axis to put $\mathbf{v}$ in xy plane - $\mathbf{R}_y$.
   1.2 rotate around z axis to put $\mathbf{v}$ in x axis - $\mathbf{R}_z$.
2. Rotate by θ around x - $\mathbf{R}_{x,\theta}$.
3. Inverse the rotation used to align with x axis.

$$\mathbf{R}_{\mathbf{v},\theta} = \mathbf{R}_y^{-1}\mathbf{R}_z^{-1}\mathbf{R}_{x,\theta}\mathbf{R}_z\mathbf{R}_y$$

- How to do a rotation around a line
  (axis which does not contain the origin)?

Conclusions

- Vertices defining a 2D/3D object can be stored in vectors: 3 coordinates for 2D and 4 coordinates for 3D.
- Rigid body transformations plus scaling can be implemented by matrix multiplications.
- If the transformation is complex (a lot of compositions) and the number of vertices is large $\Longrightarrow$ Large complexity reduction by composing the matrix first.
- You need to be careful with the order in the composition.