h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Visual Computing

Graphic objects and their programming

Darmstadt University

Prof. Dr Elke Hergenröther
Björn Frömmer
Prof. Dr Benjamin Meyer

CHAPTER 5

Transformations

Mathematical basics: Transformations

Points:

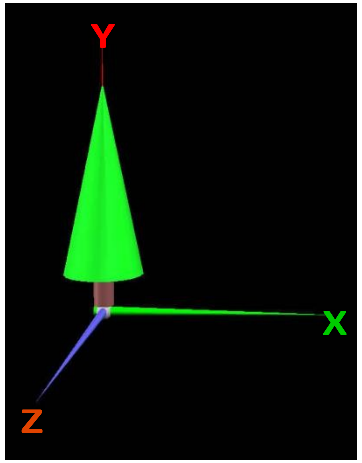
- Points (or vertices) in the plane are defined by their x and y coordinates. set.
  - In three-dimensional space corresponding to their x, y and z coordinates

- We write down points as column vectors:

$$p\text{h} = \begin{bmatrix} ? \\ y \end{bmatrix} \quad \text{resp.} \quad p\text{h} = \begin{bmatrix} ? \\ y \\ ? \end{bmatrix}$$

Arrangement of the objects in the room

1.

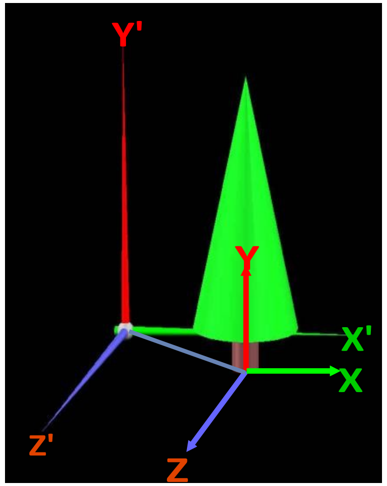


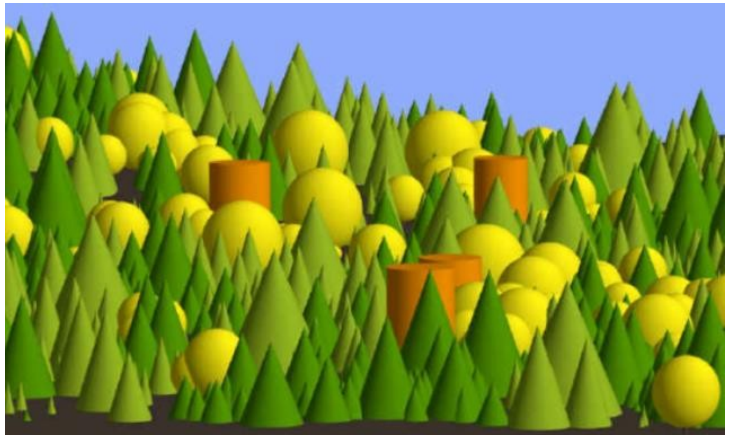
Modelling of the object in the local coordinate system

(modelling coordinate system or body coordinate system)

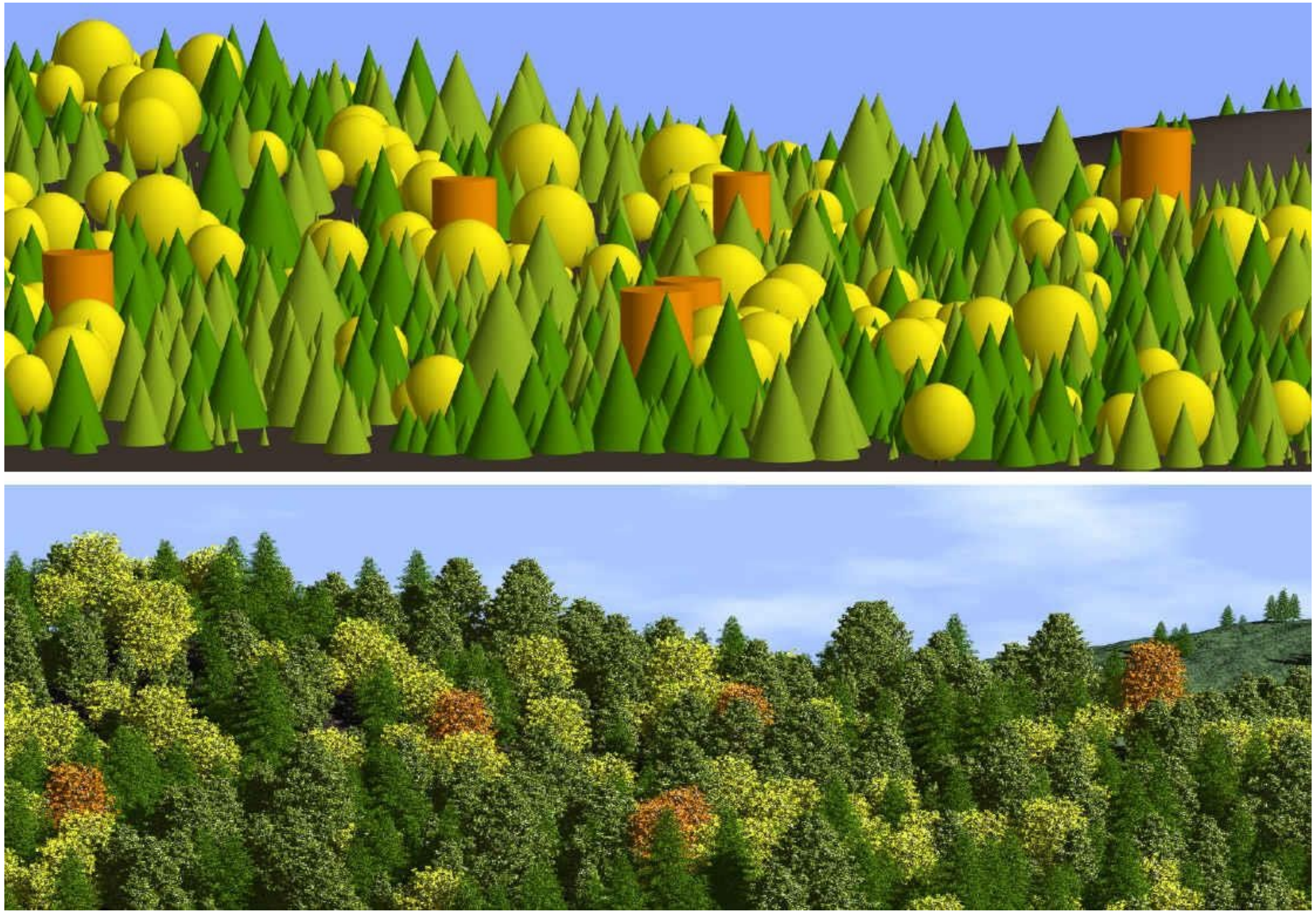
Transformation of the object into the world coordinate system

2.





Arrangement of the objects in the room II

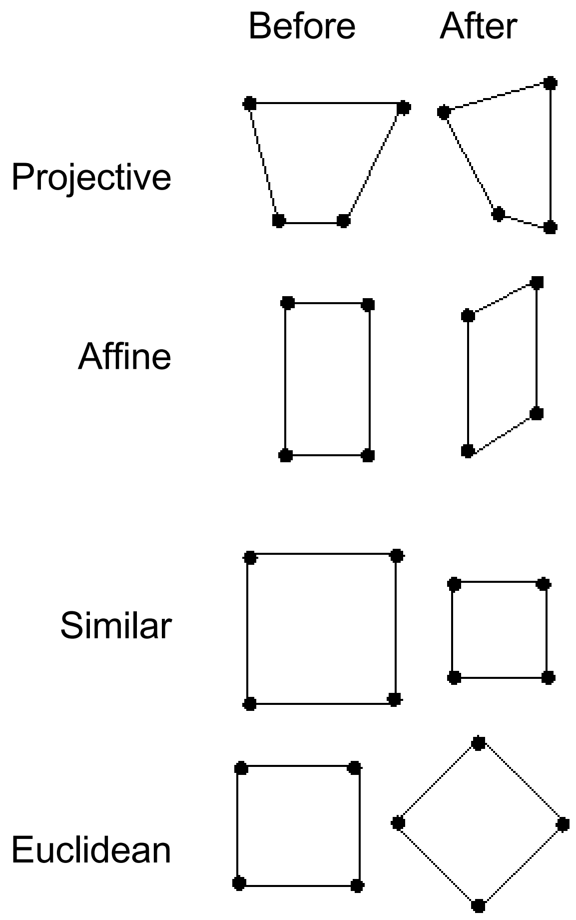


Affine transformations

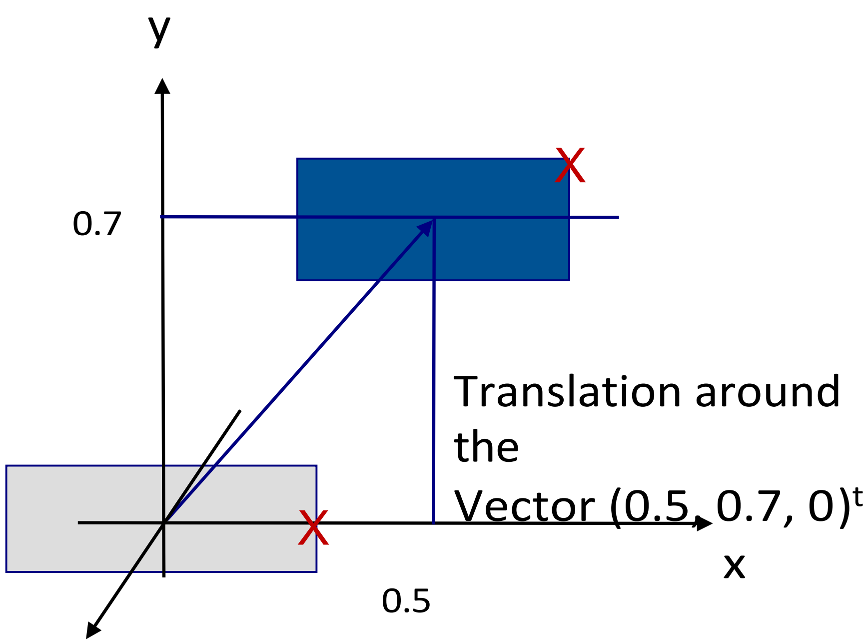
- The transformations used here
  - Translation
  - Rotation (Rotation)
  - Scaling (resizing)
- ... are also called **affine transformations**

Affin:

- What is parallel remains parallel
- The ratio of length, area and volume remains constant.



Example of a translation



- Reminder: Only the corner points of the geometry are moved!
- Mathematically, this is an addition of a Translation vector $\vec{t}$ to the original point $\vec{p}$:

$$\vec{p'} = \vec{t} + \vec{p}$$

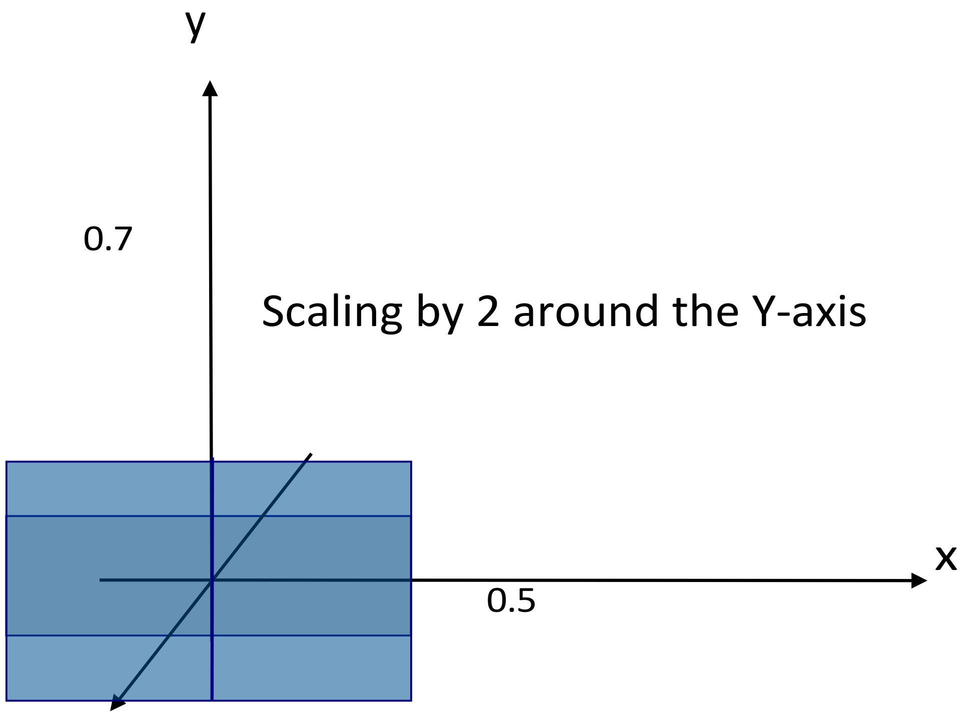
- Example for the top right corner at the coordinates $(0.3, 0.1, 0)^t$:

5. transformations

Neutral element of translation

$$t_h + p_h = \begin{bmatrix}0.5\\0.7\\0\end{bmatrix} + \begin{bmatrix}0.3\\0.1\\0\end{bmatrix} = \begin{bmatrix}0.8\\0.8\\0\end{bmatrix} = \vec{p'}$$

Example of scaling



- It applies for each point p_h at a scaling to move s_x along the X-, s_y along the Y- and s_z along the of the Z-axis:

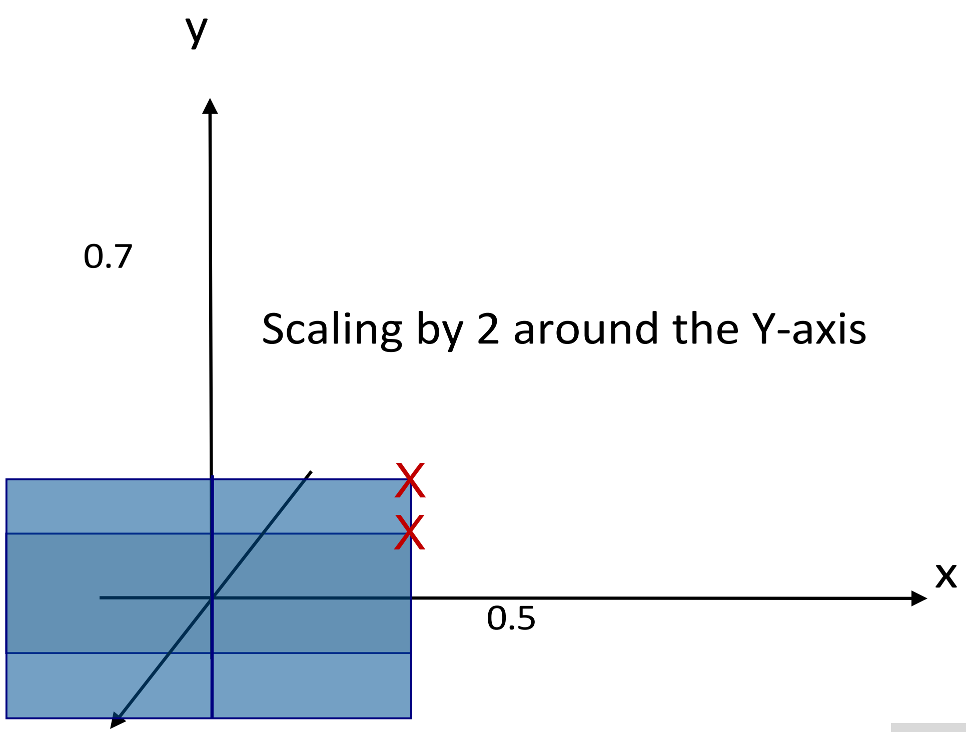
  $$p'_x = s_x * p_x \, , \quad p'_y = s_y * p_y \quad \text{and} \quad p'_z = s_z * p_z$$

- These equations can be calculated by a Summarise matrix multiplication!

$$\rightarrow \quad \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & sz \end{bmatrix} * \begin{bmatrix} px \\ py \\ pz \end{bmatrix}$$

Attention! Observe the order of multiplication!

Example of scaling 10



Scaling by 2 around the Y-axis

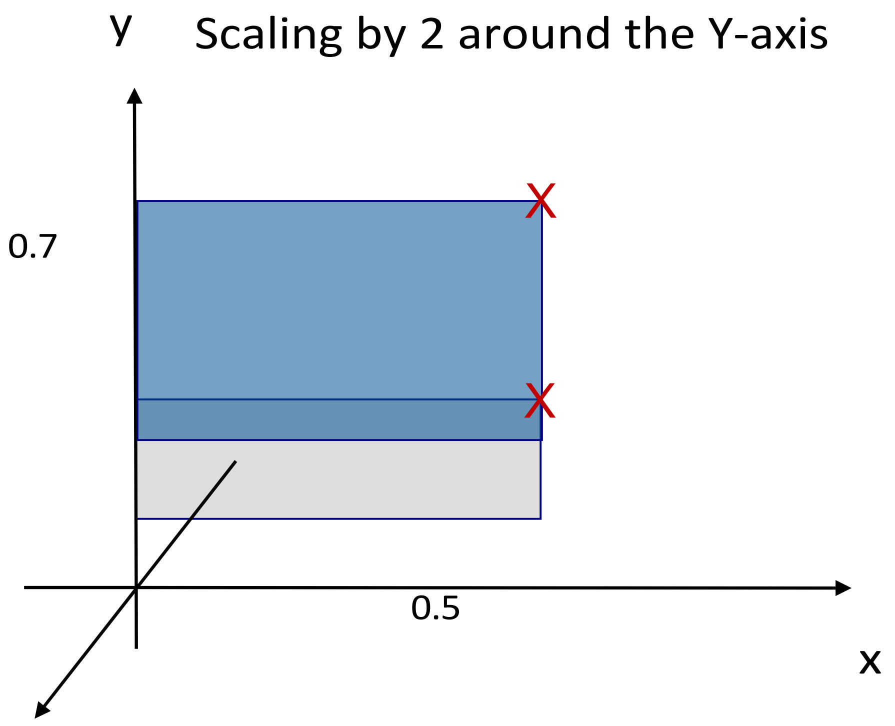
- Example for the coordinates $(0.3, 0.1, 0)^t$:

$$s_h * p_h = \begin{bmatrix} \square\square & 0 & 0 \\ 0 & \square\square & 0 \\ 0 & 0 & \square \end{bmatrix} * \begin{bmatrix} \square\square \\ py \\ \square\square \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.3 \\ 0.1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0 \end{bmatrix} = \overrightarrow{p'}$$

neutral element of the scaling!

Example of scaling III



- Again for the upper right corner, now at (0.6, 0.4, 0)^t

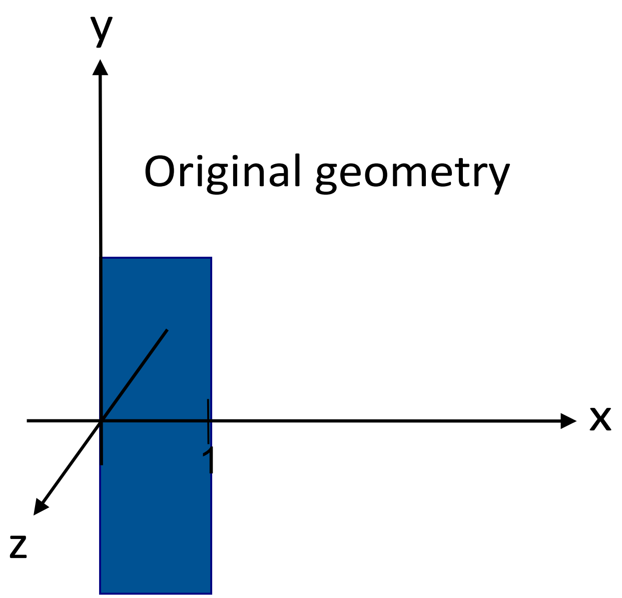
$$sh * ph = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.6 \\ 0.4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8 \\ 0 \end{bmatrix} = \vec{p'}$$

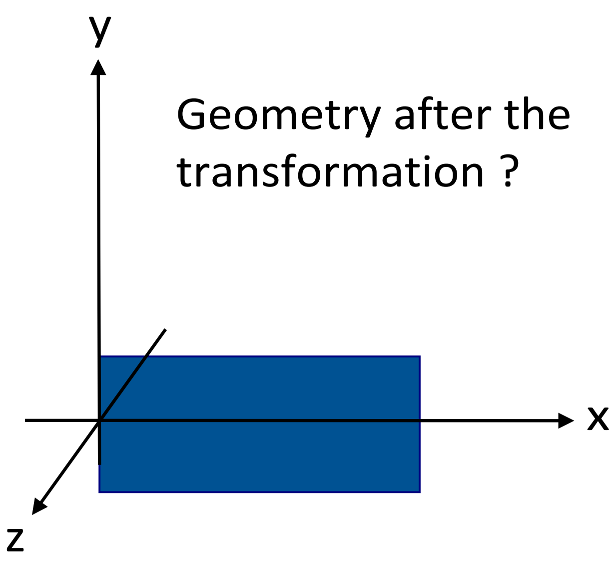
- **Observation:** The object is not only moved in the Y-direction.
  scaled, but also changes the position
  → the distance to the zero point is also scaled!

- **Therefore:** The scaling is only **valid in** relation to the Zero point invariant!

Example of scaling III

Example of scaling III

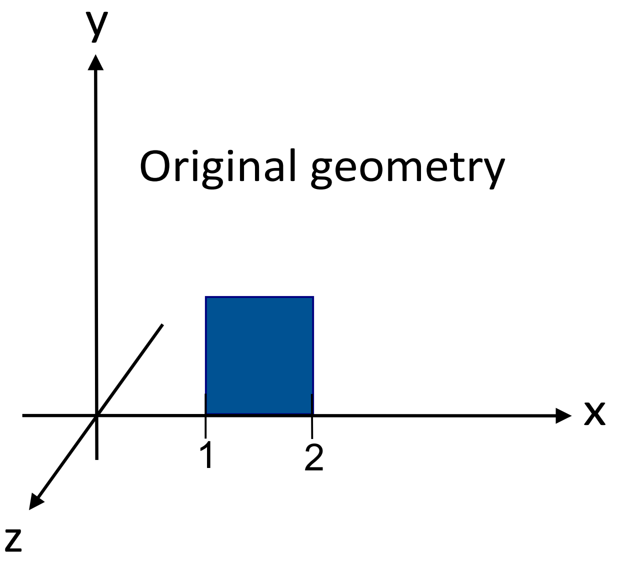


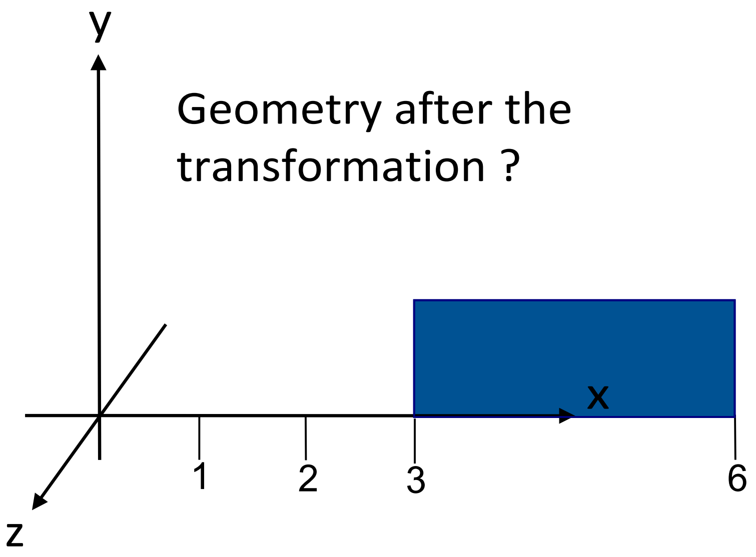


What does the geometry look like after scaling with these factors?

$x = 3$, $y = 0.3$ and $z = 1$

Example of scaling III

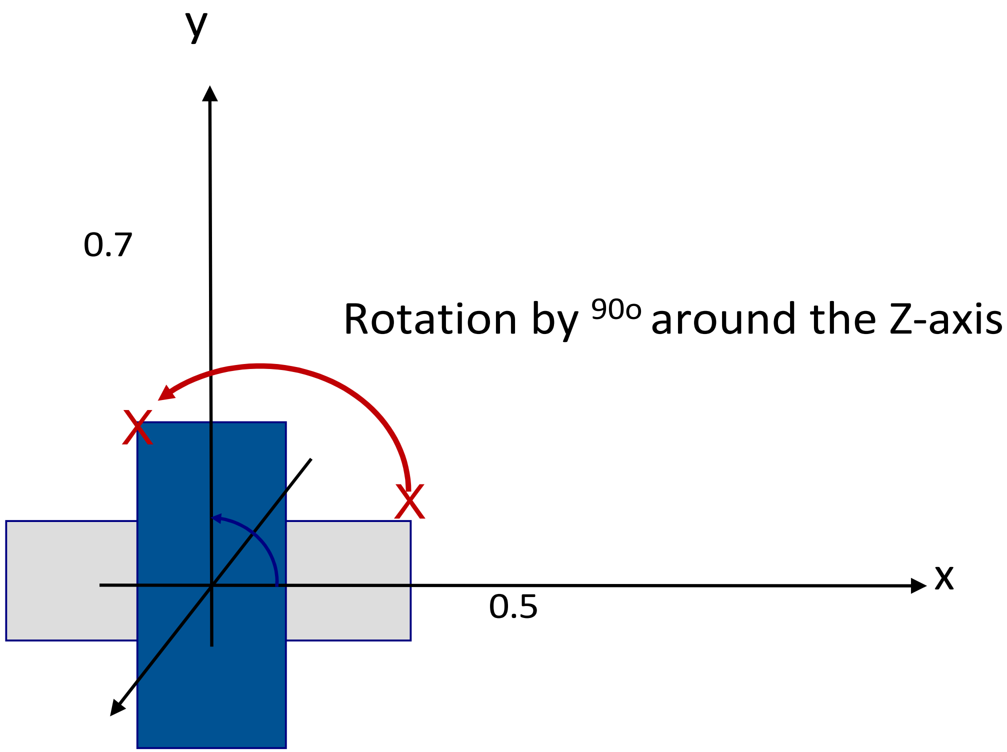




What does the geometry look like after scaling with these factors?

x = 3 , y = 1 and z = 1

Example of a rotation



- The rotation can also be formulated as a matrix multiplication of the individual corner points

- Rotation around the Z-axis by the angle α:

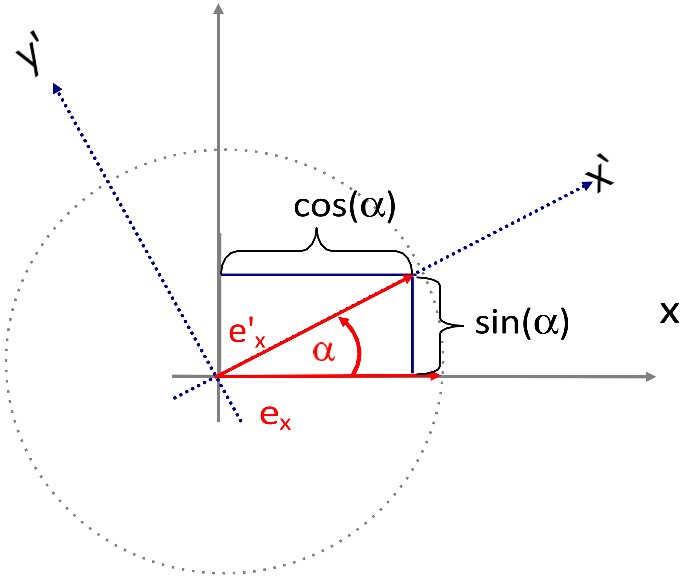
$$\vec{r} * \vec{p} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} px \\ py \\ pz \end{bmatrix} = \vec{p'}$$

- Example for the coordinates $(0.3, 0.1, 0)^t$ and 90°:

$$\vec{r} * \vec{p} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.3 \\ 0.1 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.3 \\ 0 \end{bmatrix}$$

Derivation of the rotation matrix (in 2D)

It is not the object that is rotated, but the coordinate system:



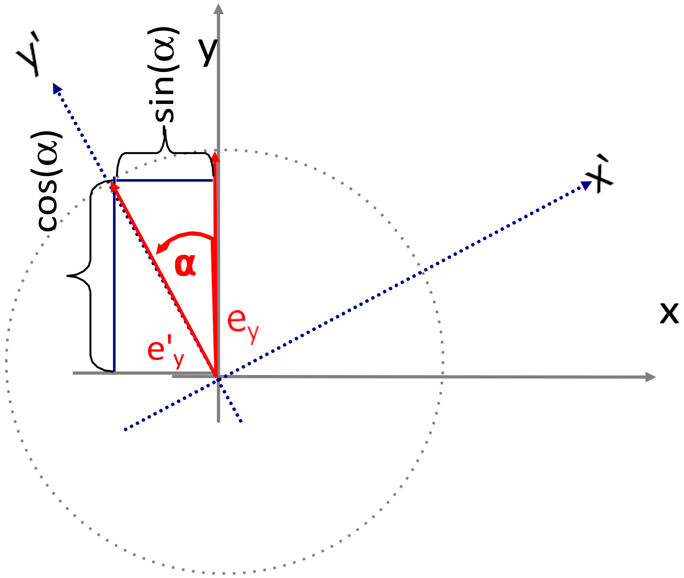
When rotating by the angle α, the unit vectors of the Cartesian coordinate system $_{ex}$ and $_{ey\ are}$ mapped onto the base vectors of the e'$_x$ and e'$_{y\ of}$ the affine coordinate system:

$$\acute{e}_{?} = T_{?} * e_{?}$$

$$\begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & ?.\ ?. \\ \sin(\alpha) & ?.\ ?. \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Derivation of the rotation matrix (in 2D)

It is not the object that is rotated, but the coordinate system:



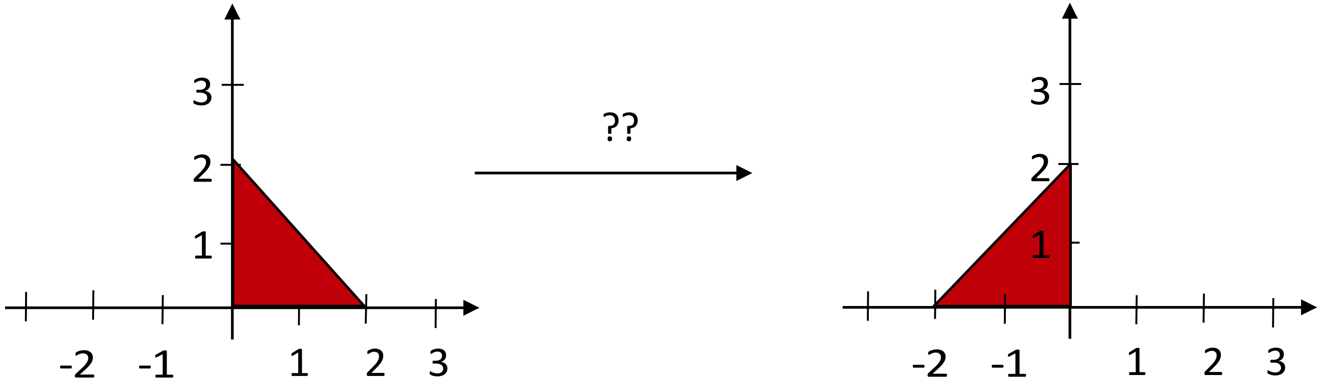
When rotating by the angle α, the unit vectors of the Cartesian coordinate system $_{ex}$ and $_{ey}$ are mapped onto the base vectors of the e'_x and e'_y of the affine coordinate system:

$$'e_{?} = T_{?} * e_{?}$$

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

Example in

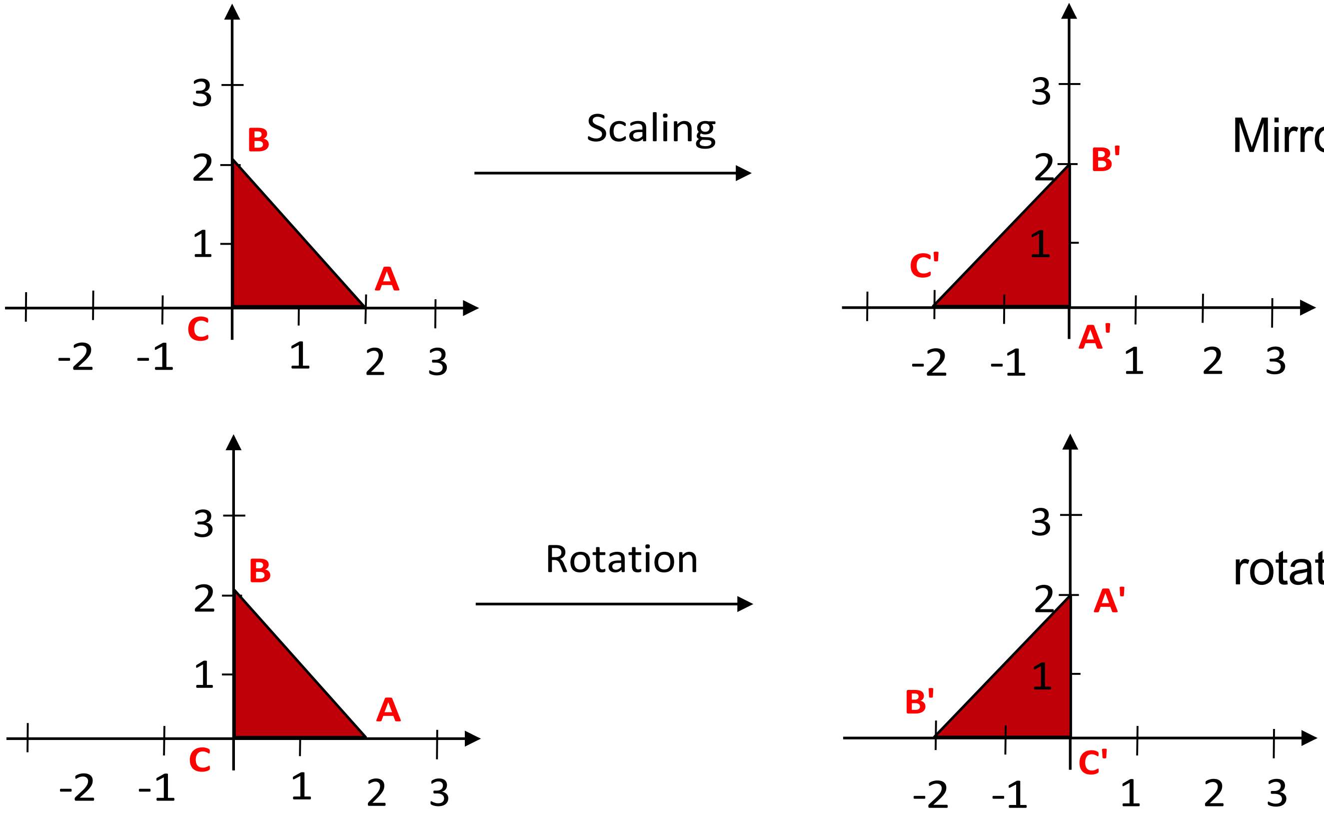
What transformation is being sought?



Without labelling the corner points, there are several solutions!

Which?

Example in

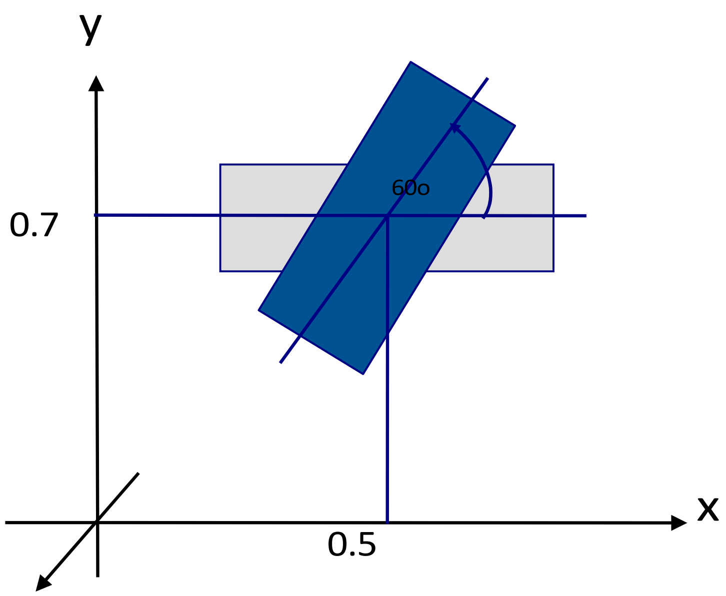


Mirrored on the Y-axis

rotated by 90

Complex transformations

- What transformations are necessary to rotate the rectangle as given?



3. transformation back to the original position

2. rotation around the Z-axis

1. Transformation into the origin

- **→ □□□ order in which the transformations are applied is important!**

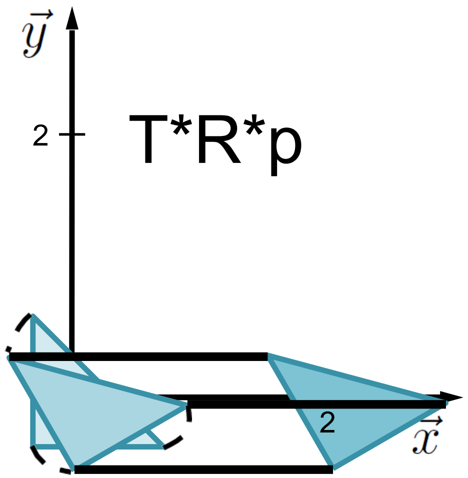
Summary up to here...

- The rotation and scaling is **always relative to the origin.**
- Rotate/Scale with Reference Point is a three-step algorithm:
  - Moving the reference point to the origin
  - Apply transformation
  - Moving the reference point back to the original position
- Chaining of transformations necessary......
- but applying one transformation after the other to each vertex **takes long** and **leads to rounding errors**
- Better concept: **Summarise transformations** and focus on the original

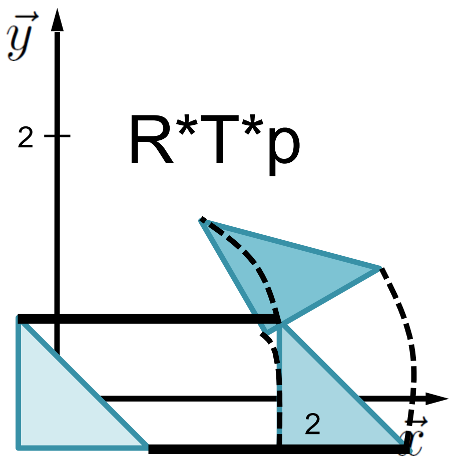
  Apply corner points → **accumulated transformation matrix**

Sequence of transformations

```
Matrix4f m = new Matrix4f();
m.translate(2, 0, 0);
m.rotate(30, 0, 0, 1);
```




```
Matrix4f m = new Matrix4f();
m.rotate(30, 0, 0, 1);
m.translate(2, 0, 0);
```



"The geometry moves backwards through the programme and collects the transformations".

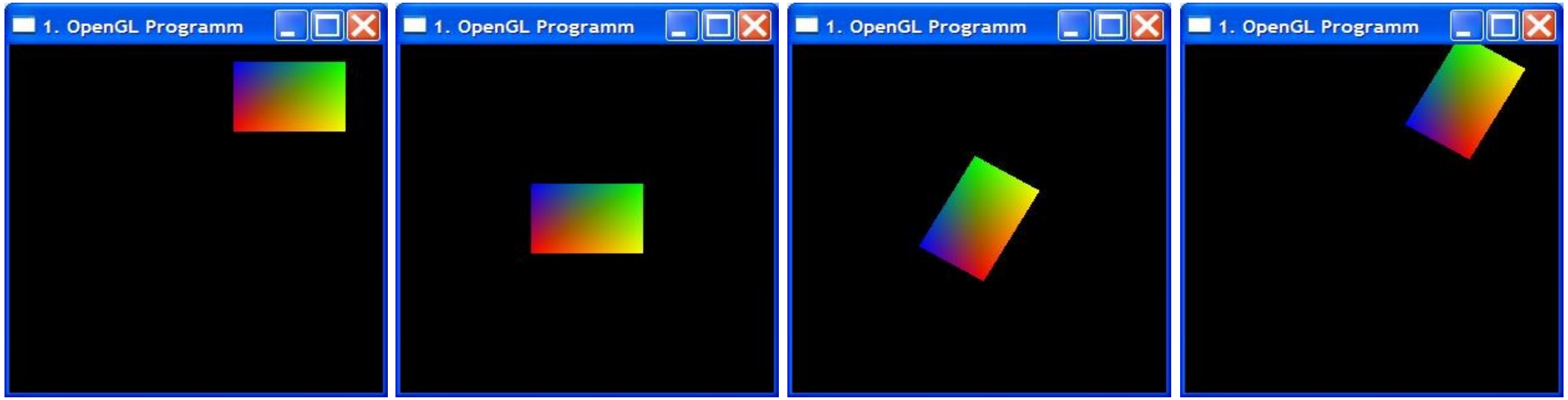
Sequence of transformations 23

Example: Rotating a square by 60°.

- Transformations are given in **reverse** order:

Order in which, the transformations are applied to the square:

```
Matrix4f m = new Matrix4f();    // create new Mattix
m.translate( 0.5, 0.7, 0.);     // Back transformation
m.rotate( 60., 0., 0., 1.);     // Rotation
m.translate( -0.5, -0.7, 0.); // To the origin
```



Sequence of transformations 24

Why are the transformations given in reverse order?

- Matrix multiplications are not commutative (i.e. they are not in their sequence interchangeable)
- Example: a) does not give the same result as b)

a)

$$\begin{bmatrix} 3 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

b)

$$\begin{bmatrix} 5 \\ -3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

Sequence of transformations 25

Why are the transformations given in reverse order?

- Matrix multiplications are not commutative (i.e. they are not in their sequence interchangeable)
- Equivalent calculations of P':

$$\boxed{P' = M_2 \cdot (M_1 \cdot P)} \Leftrightarrow \boxed{P' = (M_2 \cdot M_1) \cdot P}$$

Intuitive approach when you transformed "by hand":
P is first multiplied by M_1 and then the result vector is multiplied by M_2

Procedure of OpenGL: Create an accumulated matrix ($_{M2}$ is multiplied by $_{M1}$).

Sequence of transformations V

Example: Rotating a square by 60°.

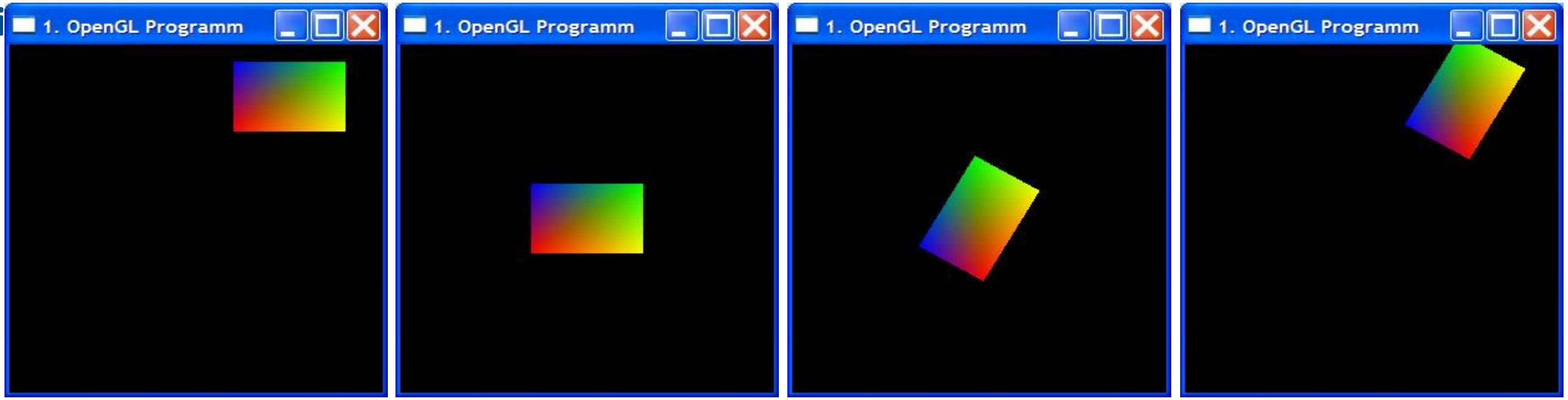
- Transformations are given in **reverse** order:

Order in which, the transformations are multiplied onto the accumulated matrix M:

```
Matrix4f m = new Matrix4f();      // create new Mattix
m.translate( 0.5, 0.7, 0.);       // Retransformation
m.rotate( 60., 0., 0., 1.);       // Rotation
m.translate( -0.5, -0.7, 0.);     // Into the origin
```

$\mathbf{M=M_{T2} * M_R * M_{T1}}$

5. transformati



Homogeneous

Problem:

Translation = vector addition

Scaling & Rotation = Matrix Multiplication

Since it makes sense to "calculate" the different transformations with each other to an accumulated matrix and then apply it to all vertices (see OpenGL), there must be a way to link the additive part (translation) and the multiplicative part (rotation, scaling) with each other.

Homogeneous

Problem:

- **Rotation** and **scaling** can be expressed by **matrix multiplication**
- To create the accumulated matrix we need a corresponding Representation of the **translation** (i.e. a **vector addition**)
- This is what such a matrix would have to look like, so that the following applies:

$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} p_x + t_x \\ p_y + t_y \\ p_z + t_z \end{pmatrix}$$

- Unfortunately, such a 3x3 matrix does not exist.
- Solution: **Homogeneous coordinates**

Homogeneous coordinates II

- Mathematical trick: adding a 4th coordinate!

$$\mathbb{R}^3 \ni \begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{P}^3 \,,$$

- Homogenisation (back to Euclidean space)

$$\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} X/W \\ Y/W \\ Z/W \end{pmatrix} \,, \forall W \neq 0$$

- Divide by the 4th coordinate

Homogeneous coordinates in

- The vectors $\vec{x'}, \vec{y'}$ and $\vec{t}$ become a matrix T (transformation matrix) summarised

$$\begin{pmatrix} p'_1 \\ p'_2 \end{pmatrix} = \begin{pmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \Leftrightarrow \begin{pmatrix} p'_1 \\ p'_2 \\ 1 \end{pmatrix} = \begin{pmatrix} x'_1 & y'_1 & t_1 \\ x'_2 & y'_2 & t_2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ 1 \end{pmatrix}$$

Homogeneous coordinates

Homogeneous coordinates in

Homogeneous coordinates in

- The vectors $\vec{x'}$, $\vec{y'}$ and $\vec{t}$ become a matrix T (transformation matrix) summarised

Multiplicative part

$$\begin{pmatrix} p'_1 \\ p'_2 \\ 1 \end{pmatrix} = \begin{pmatrix} x'_1 & y'_1 & t_1 \\ x'_2 & y'_2 & t_2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} p'_1 \\ p'_2 \\ 1 \end{pmatrix} = \begin{pmatrix} x'_1 \cdot p_1 + y'_1 \cdot p_2 + t_1 \cdot 1 \\ x'_2 \cdot p_1 + y'_2 \cdot p_2 + t_2 \cdot 1 \\ 0 \quad\quad 0 \quad 1 \end{pmatrix}$$

Homogeneous coordinates

Homogeneous coordinates in

ditive part

Overview of all transformations (in 4D)

- Rotation:

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(a) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$R_y(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(a) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

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

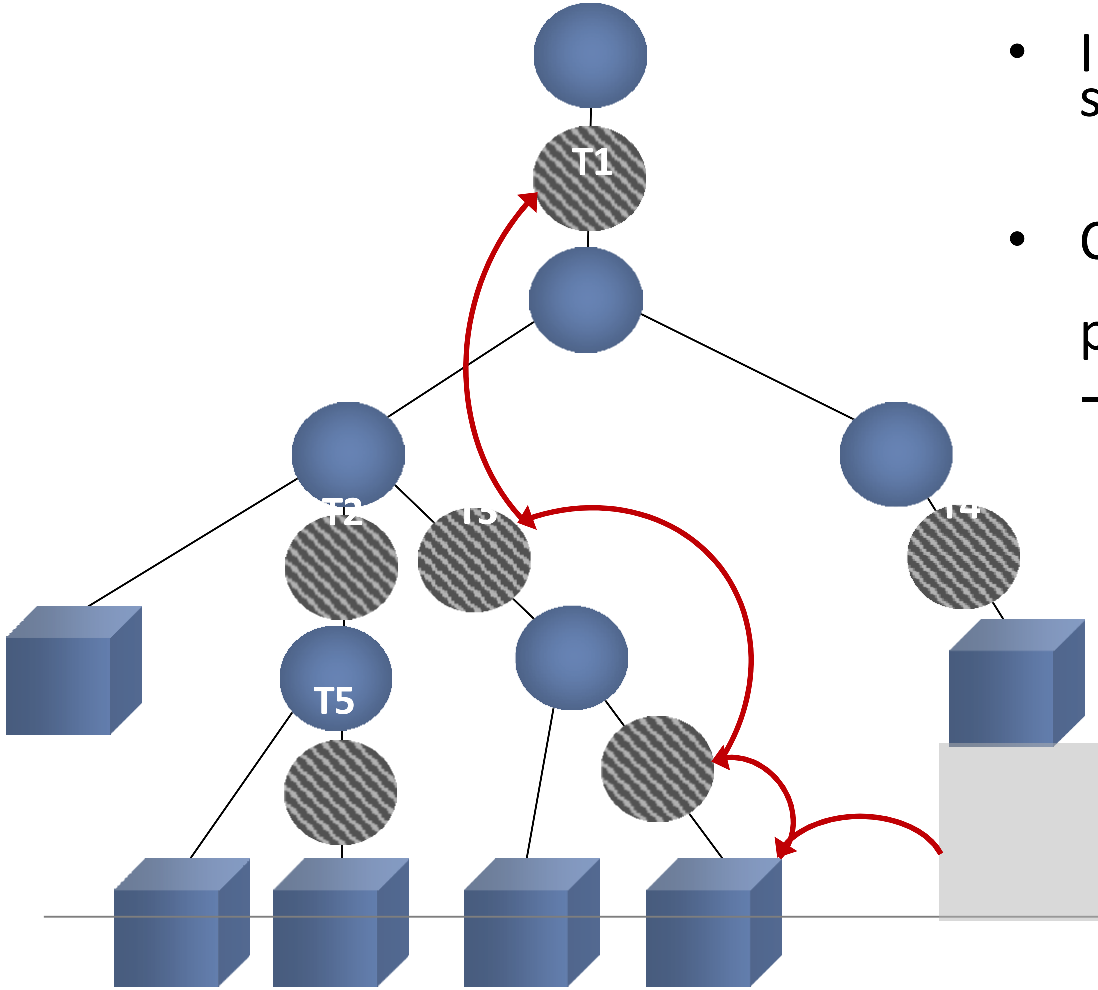
- Scaling:

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

- Translation:

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

Implementation of the scene graph



- In which order must the transformations of the scene graph be considered?
- Child objects inherit the transformations of their parents
  → the scene graph is scrolled from bottom to top!

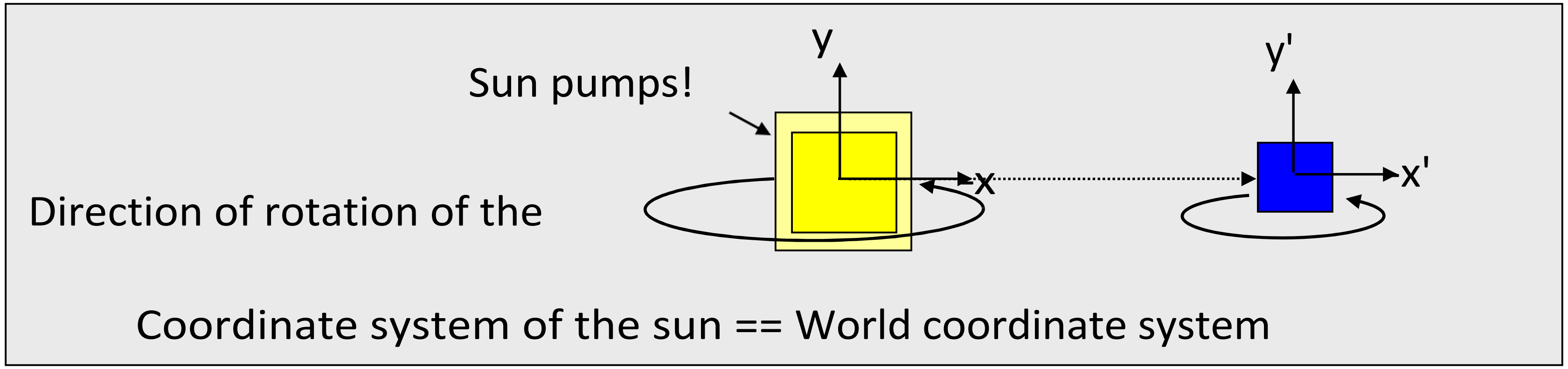
Applied transformations:

T1 * T3 * T6 * P

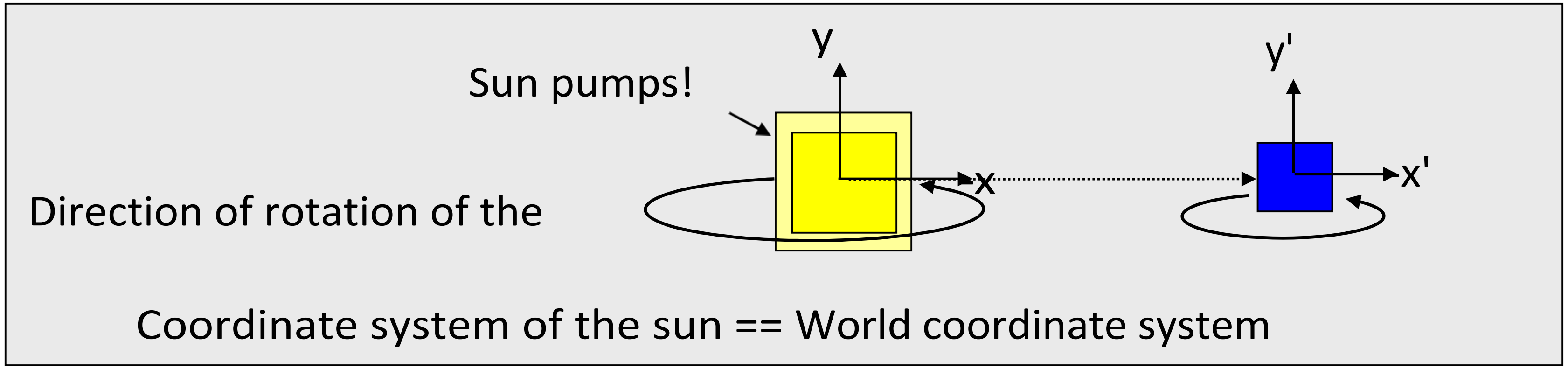
Scene graph -

Task: Miniature solar system

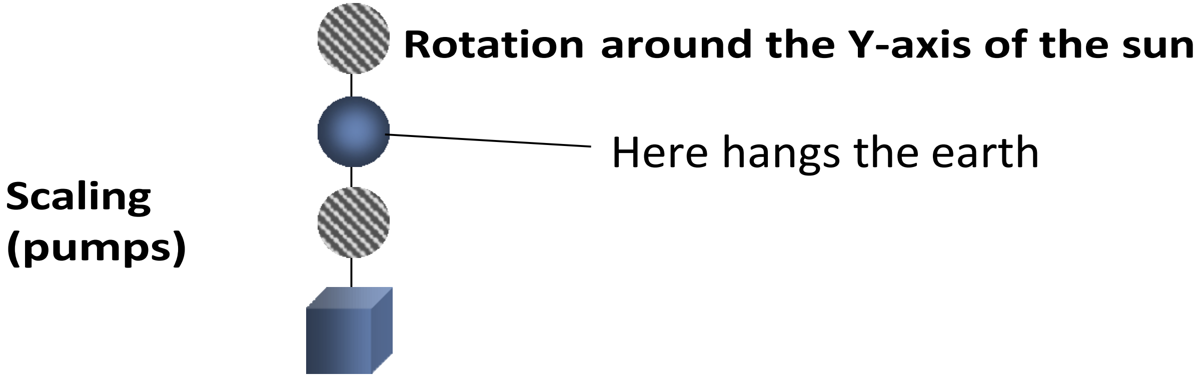
- Sun rotates around its Y-axis.
- As it rotates, it inflates and collapses again.
- The Earth rotates at some distance with the same angular velocity (i.e. swept angle per time unit is equal) around the Y-axis of the sun.
- In addition, the Earth rotates around its own Y-axis.
- Earth and sun are angular ;-)



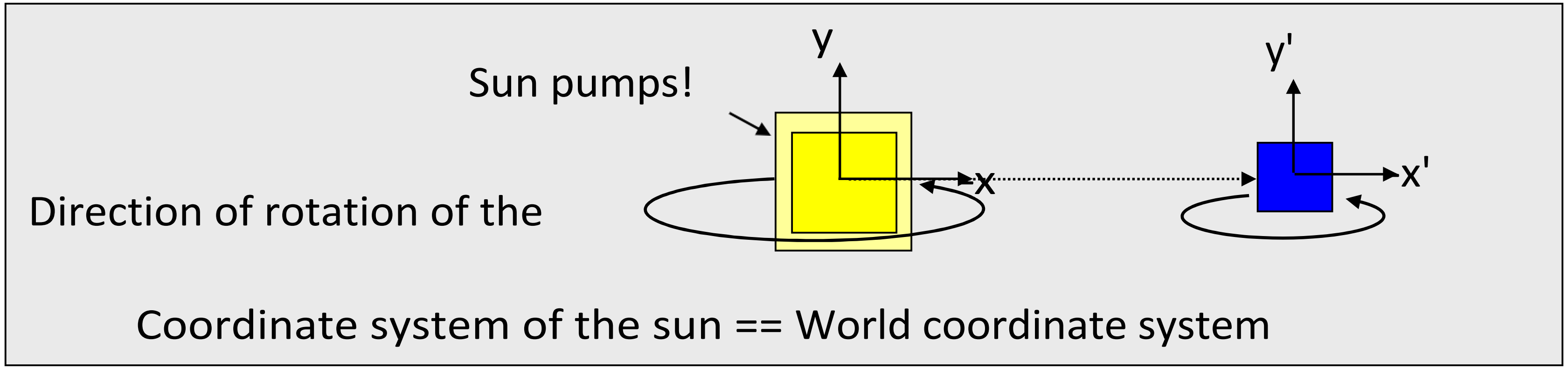
Scene graph -



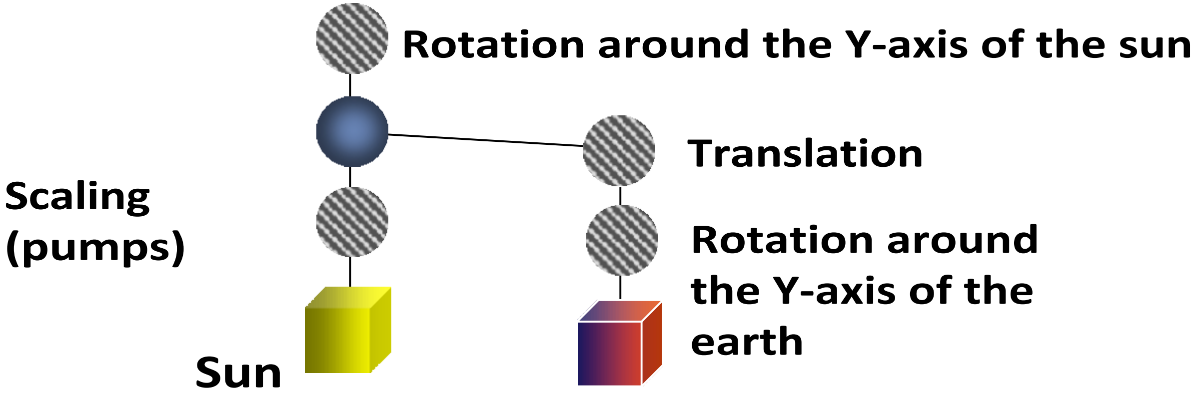
- The sun and the earth rotate at the same angular velocity around the sun's Y-axis
- But only the sun pumps up and collapses



Scene graph -



- Earth additionally rotates around its own Y-axis



Earth

Scene graph -

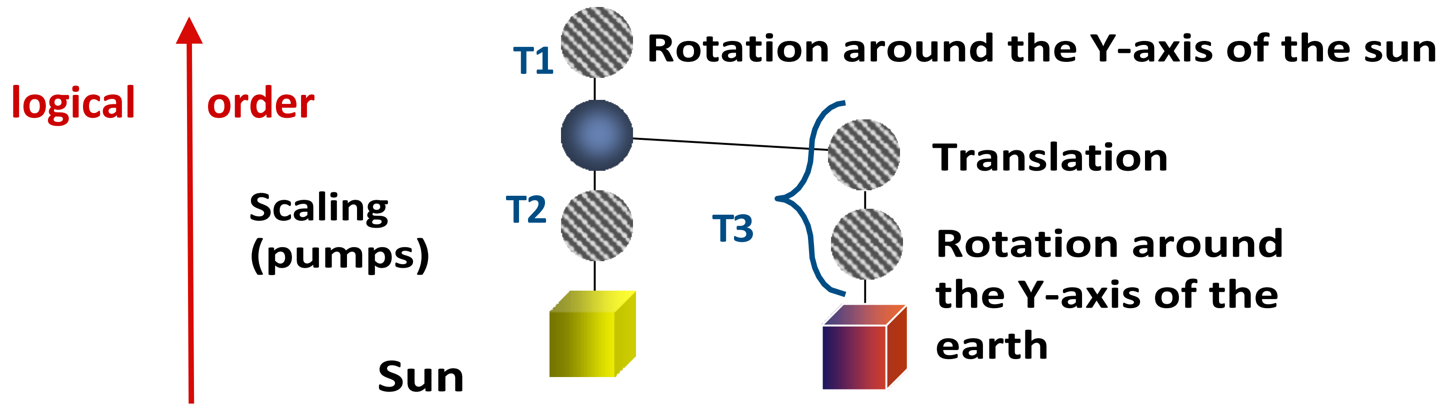
Scene graph - example

Pseudocode:

```
Matrix4f T1 = new Matrix4f();          // Create new matrix T1
T1.rotate( alpha, 0.0, 1.0, 0.0);      // Rotation with angle alpha around the vector
                                          (0,1,0)^t

Matrix4f T2 = new Matrix4f();          // Create new matrix T2
T2.scale( s, s, s);                    // uniform scaling by factor s

Matrix4f T3 = new Matrix4f();          // Create new matrix T3
T3.translate(p1, p2, p3);              // Translation relative to the sun, position (p1, p2,
                                          )
```



Earth

Scene graph - example

Accumulated matrix for.

the $_{ME}$ = **T1* T3;**

the sun: $_{MS}$ = **T1* T2;**