CG – T6 - Transformations

L:CC, MI:ERSI

Miguel Tavares Coimbra

(course and slides designed by Verónica Costa Orvalho)

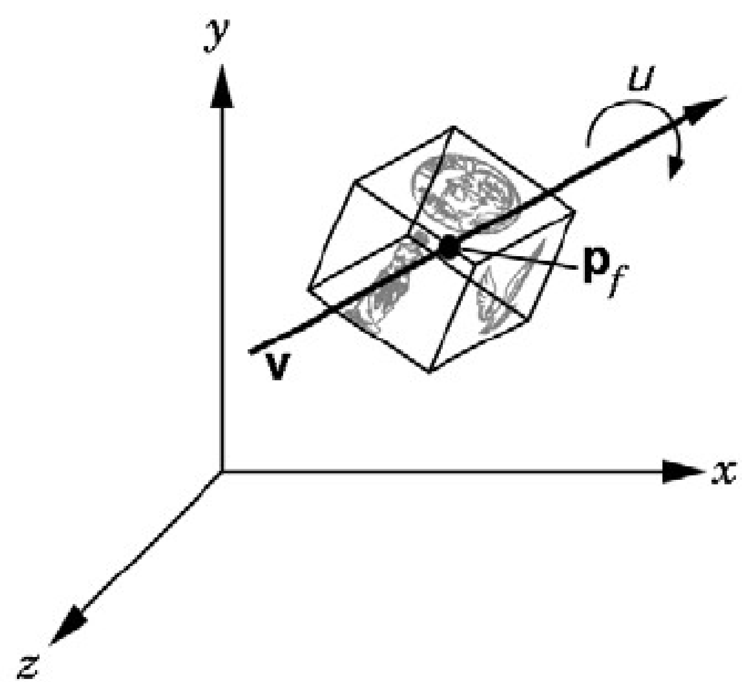
U. PORTO FC

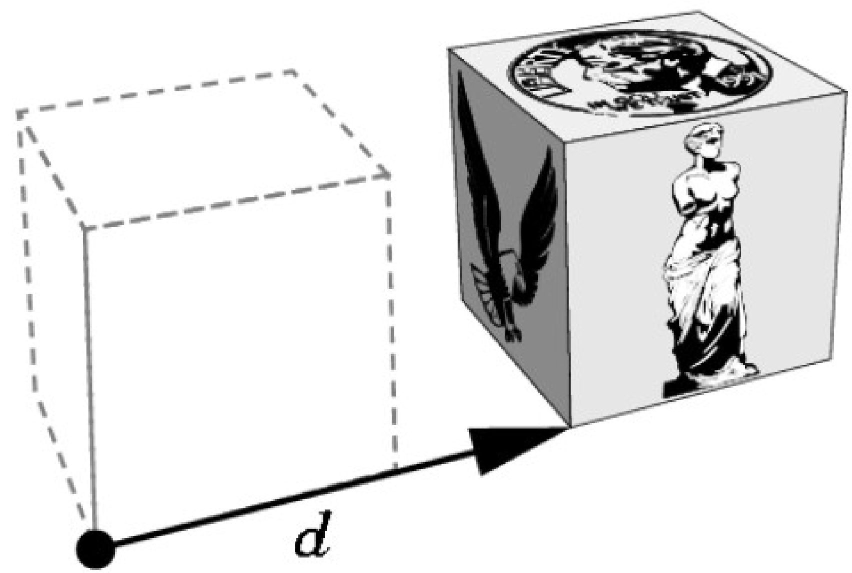
agenda

. introduction
. transform
. linear transform
. affine transformation
. homogeneous notation
. what is a matrix?
. 3D homogeneous transformations

introduction

rigid body transformations



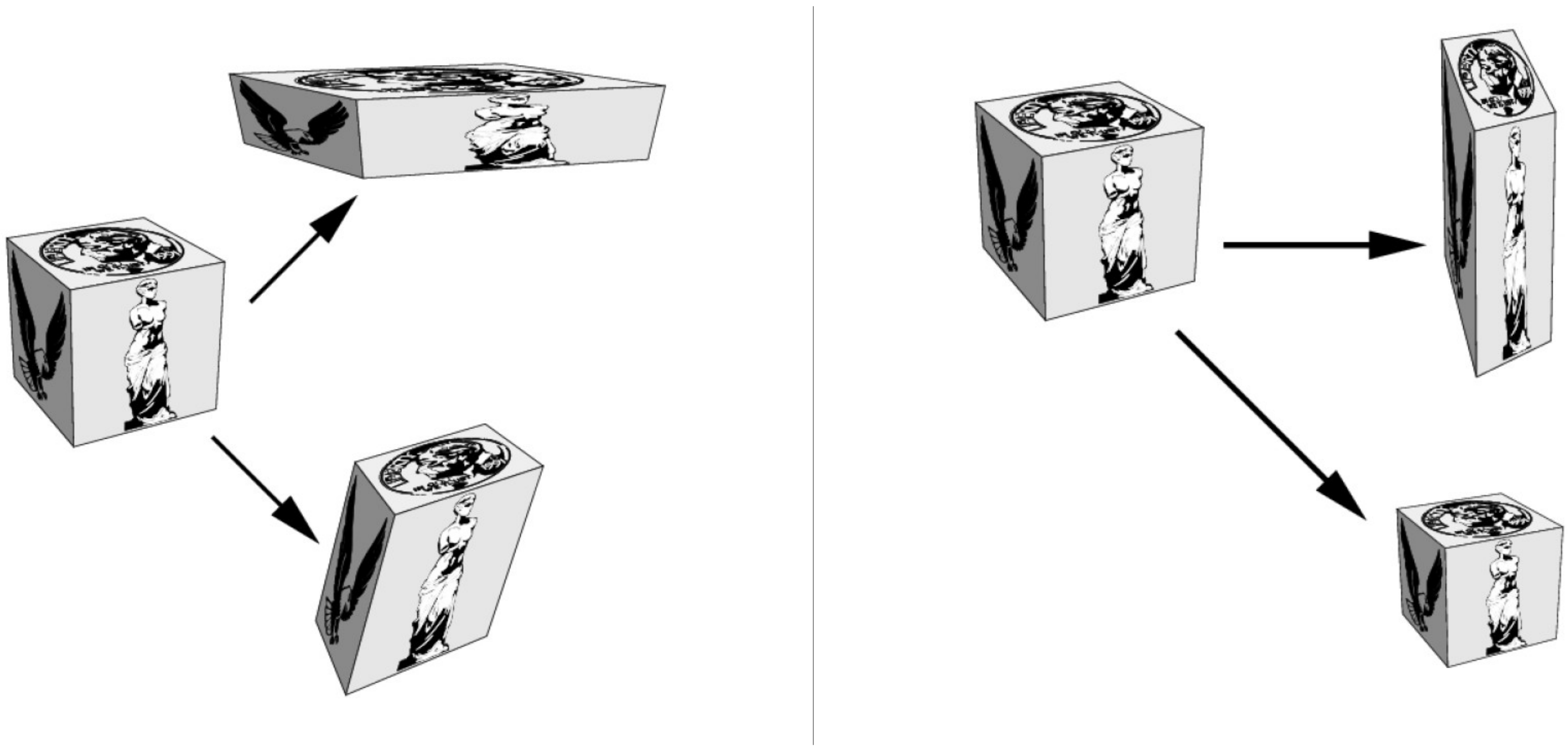


rotation

translation

introduction

non-rigid body transformations



distance between points on objects
DO NOT remain constant

transform

transform:

operation that takes an attribute: points, vectors or colors

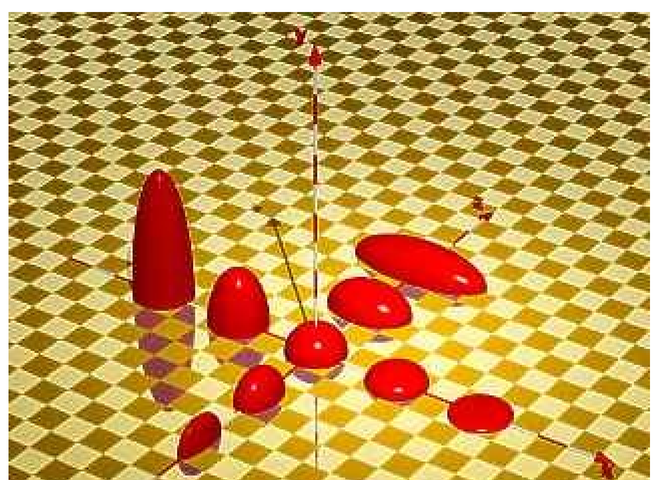
transform

transform:

operation that takes an attribute:
points, vectors or colors

↓

<u>converts</u> them in some way



http://www.lohmueller.business.t-online.de/pov_tut/trans/scale1t.jpg

http://www.ltutech.com

transform

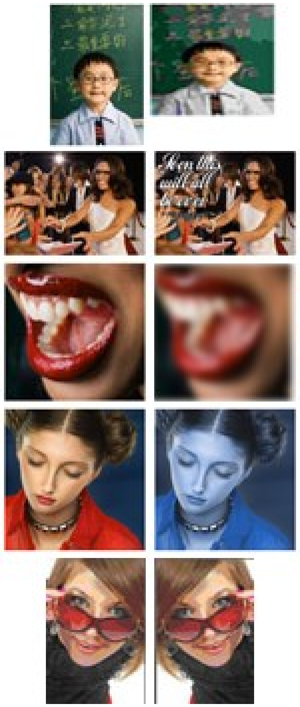
transform:

operation that takes an attribute: points, vectors or colors

↓

converts them in some way

basic tool for manipulating geometry



http://www.ltutech.com

transform

transform:

. position, reshape, animate { objects, lights, cameras }

transform

transform:

. position, reshape, animate { objects, lights, cameras }

. ensure that all computations are performed in the same coord. system, etc.

linear transform

linear transform:

linear transform

linear transform:

. preserves vector **addition**

. and **scalar multiplication**

linear transform

linear transform:

. preserves vector **addition**

$$\mathbf{f(x) + f(y) = f(x + y)}$$

. and **scalar multiplication**

linear transform

linear transform:

. preserves vector **addition**

$$\mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{y}) = \mathbf{f}(\mathbf{x} + \mathbf{y})$$

. and **scalar multiplication**

$$k\mathbf{f}(\mathbf{x}) = \mathbf{f}(k\mathbf{x})$$

linear transform

linear transform:

. preserves vector **addition**

$$\mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{y}) = \mathbf{f}(\mathbf{x} + \mathbf{y})$$

. and **scalar multiplication**

$$k\mathbf{f}(\mathbf{x}) = \mathbf{f}(k\mathbf{x}) \rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{f}(2\mathbf{x})$$

linear transform

linear transform:

. preserves vector **addition**

$$\mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{y}) = \mathbf{f}(\mathbf{x} + \mathbf{y})$$

. and **scalar multiplication**

$$k\mathbf{f}(\mathbf{x}) = \mathbf{f}(k\mathbf{x}) \rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{f}(2\mathbf{x})\ ?$$

linear transform

linear transform:

. **scalar multiplication**

$$k\mathbf{f}(\mathbf{x}) = \mathbf{f}(k\mathbf{x}) \rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{f}(2\mathbf{x})$$

takes a vector and multiplies each element by **2**

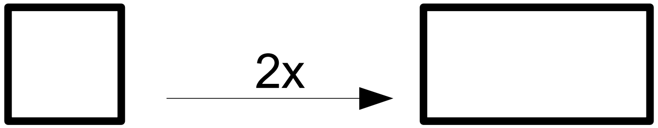
linear transform

linear transform:

. <u>**scalar multiplication**</u>

$$k\mathbf{f}(\mathbf{x}) = \mathbf{f}(k\mathbf{x}) \rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{f}(2\mathbf{x})$$

takes a vector and multiplies each element by **2**



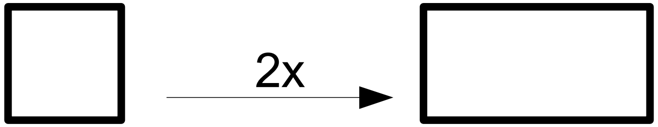
linear transform

linear transform:

. **scalar multiplication**

$$k\mathbf{f}(\mathbf{x}) = \mathbf{f}(k\mathbf{x}) \rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{f}(2\mathbf{x})$$

takes a vector and multiplies each element by **2**



scaling transform

linear transform

linear transform:

- **scaling transform**
  changes the scale (size) of the object

linear transform

linear transform:

- **scaling transform**
  changes the scale (size) of the object

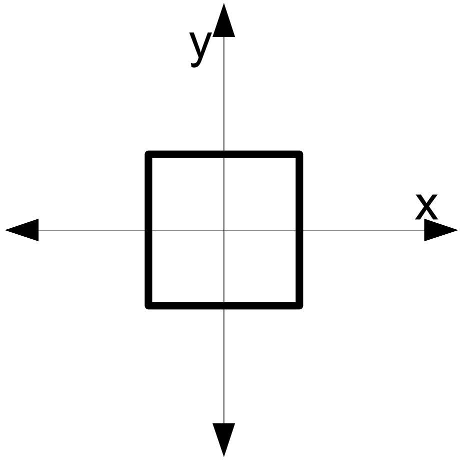
- **rotation transform**

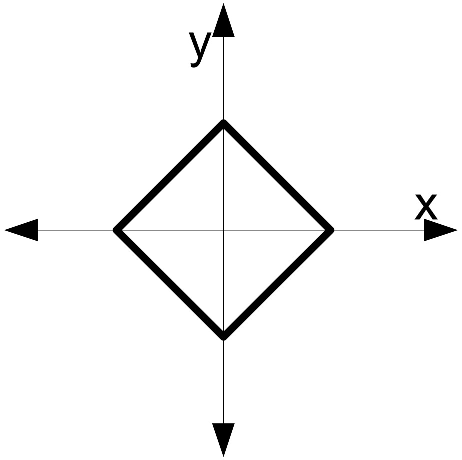
linear transform

linear transform:

. **scaling transform**
changes the scale (size) of the object

. **rotation transform**
rotates a vector about the origin





linear transform

linear transform:

. **scaling transform**
changes the scale (size) of the object

. **rotation transform**
rotates a vector about the origin

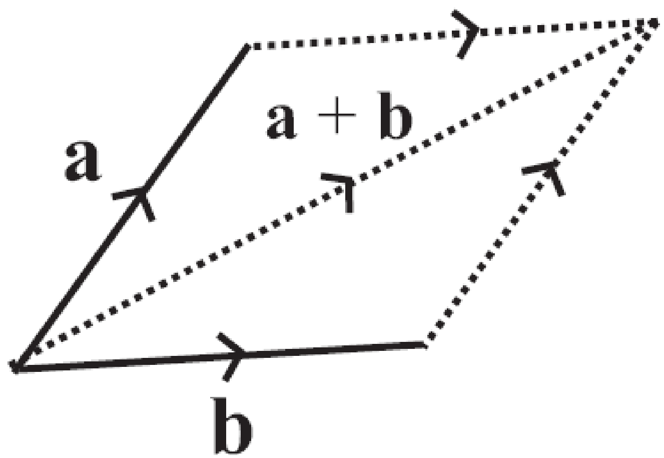
represented by: **3 x 3 matrix**

linear transform

what happens if we would like to **add** a fixed vector to another vector **?**

linear transform

what happens if we would like to **add** a <u>fixed vector</u> to <u>another vector</u> **?**

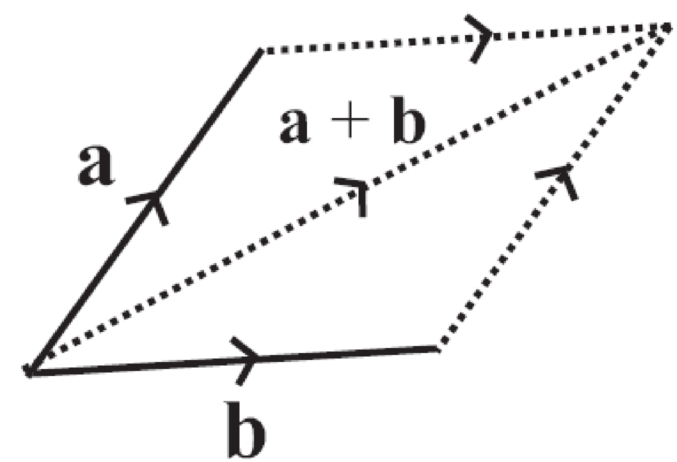


linear transform

what happens if we would like to **add** a fixed vector to another vector **?**

$\mathbf{f}(\mathbf{x}) = \mathbf{x} + (5,3,6)$

[not linear]

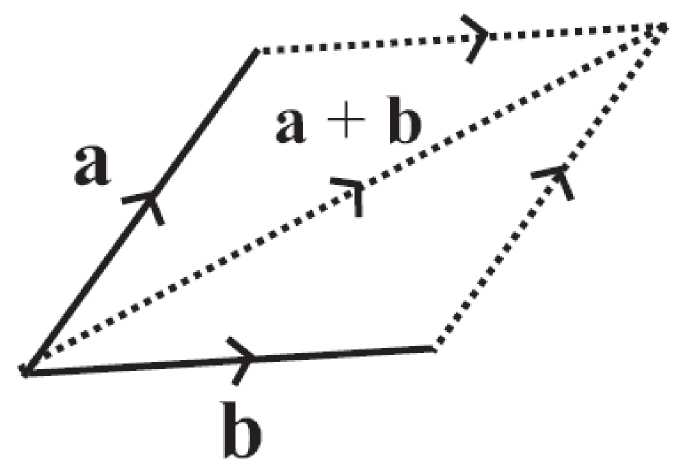


linear transform

what happens if we would like to **add** a fixed vector to another vector **?**

perform a translation

f(**x**) = **x** + (5,3,6)
[not linear]



linear transform

what if we would like to **scale** an object to be half as large,

1. $\mathbf{f}(\mathbf{x}) = \mathbf{f}(1/2\,\mathbf{x})$

linear transform

what if we would like to **scale** an object to be half as large, then **move** it to a different location ?

1. $\mathbf{f}(\mathbf{x}) = \mathbf{f}(1/2\mathbf{x})$
2. $\mathbf{f}(\mathbf{x}) = \mathbf{x} + (5,3,6)$

linear transform

what if we would like to **scale** an object to be half as large, then **move** it to a different location ?

1. $\mathbf{f}(\mathbf{x}) = \mathbf{f}(1/2\,\mathbf{x})$
2. $\mathbf{f}(\mathbf{x}) = \mathbf{x} + (5,3,6)$

using these functions makes it **difficult** to easily **combine** them

solution

affine transformations

solution:

affine transformations

affine transformations

solution:

affine transformations is one that performs a **linear transformation** and then a **translation**

affine transformations

solution:

affine transformations is one that performs a **linear transformation** and then a **translation**

represented by: **4 x 4 matrix**

affine transformations

solution:

affine transformations is one that performs a **linear transformation** and then a **translation**

represented by: **4 x 4 matrix**

homogeneous notation

homogeneous notation

. useful for transforming both:
vectors and **points**

homogeneous notation

. useful for transforming both:
vectors and **points**

. allows translation only on points

homogeneous notation

p = (pX,pY,pZ,pW)

pW = 1 → points

pW = 0 → vectors

homogeneous notation

p = (pX,pY,pZ,pW)

- **pW = 1** → points
- **pW = 0** → vectors

if **pW != 1** && **pW != 0**

homogeneous notation

p = (pX,pY,pZ,pW)

- **pW = 1** → points
- **pW = 0** → vectors

if **pW != 1** && **pW != 0**

→ then the actual point is obtained by **<u>homogenization</u>**

p = (pX/pW, pY/pW, pZ/pW, pW/pW)

more on matrix

now we can concatenate
individual affine transforms:

more on matrix

now we can concatenate
individual affine transforms:

. translation
. rotation
. scale
. reflection
. shearing
. rigid body
. etc.

what is a matrix?

matrix $\mathbf{M}$:

tool for manipulating **vectors** and **points**

what is a matrix?

matrix $\mathbf{M}$:

tool for manipulating **vectors** and **points**

a **point** describes a location in space

what is a matrix?

matrix $\mathbf{M}$:

tool for manipulating **vectors** and **points**

a **point** describes a location in space

a **vector** describes a direction, has no location

what is a matrix?

matrix $\mathbf{M}$:

tool for manipulating **vectors** and **points**

a **point** describes a location in space

a **vector** describes a direction, has no location

$$\mathbf{M}_{4x4} = \begin{pmatrix} m00 & m01 & m02 & 0 \\ m10 & m11 & m12 & 0 \\ m20 & m21 & m22 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

in homogeneous coordinates

what is a matrix?

matrix M:

$$\mathbf{M}_{4x4} = \begin{pmatrix} m00 & m01 & m02 & 0 \\ m10 & m11 & m12 & 0 \\ m20 & m21 & m22 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

what is a matrix?

matrix M:

rotate, scale, sharing

$$
\mathbf{M}_{4x4} = \begin{pmatrix} m00 & m01 & m02 & 0 \\ m10 & m11 & m12 & 0 \\ m20 & m21 & m22 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

what is a matrix?

matrix M:

rotate, scale, sharing

$$M_{4x4} = \begin{pmatrix} m00 & m01 & m02 & 0 \\ m10 & m11 & m12 & 0 \\ m20 & m21 & m22 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

translate

what is a matrix?

unit matrix or identity matrix $\mathbf{I}$:

it is square and contains ones in the diagonal and zeros elsewhere

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

in homogeneous coordinates

2D homogeneous transformations

2D homogeneous transformations

scale

$$M_{3x3} = \begin{pmatrix} s\_x & 0 & 0 \\ 0 & s\_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

rotate

$$M_{3x3} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

translate

$$M_{3x3} = \begin{pmatrix} 1 & 0 & t\_x \\ 0 & 1 & t\_y \\ 0 & 0 & 1 \end{pmatrix}$$

in homogeneous coordinates

2D homogeneous transformations

scale

$$M_{3x3} = \begin{pmatrix} s\_x & 0 & 0 \\ 0 & s\_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

rotate

$$M_{3x3} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

translate

$$M_{3x3} = \begin{pmatrix} 1 & 0 & t\_x \\ 0 & 1 & t\_y \\ 0 & 0 & 1 \end{pmatrix}$$

1. can we combine these matrix?

2. How?

3. why?

2D homogeneous transformations

scale

$$\mathbf{M}_{3x3} = \begin{pmatrix} s\_x & 0 & 0 \\ 0 & s\_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

rotate

$$\mathbf{M}_{3x3} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

translate

$$\mathbf{M}_{3x3} = \begin{pmatrix} 1 & 0 & t\_x \\ 0 & 1 & t\_y \\ 0 & 0 & 1 \end{pmatrix}$$

any sequence of translate/scale/rotate can be **combined** into a single homogeneous matrix by **multiplication**. For **efficiency**

2D homogeneous transformations

scale

$$M_{3x3} = \begin{pmatrix} s\_x & 0 & 0 \\ 0 & s\_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

rotate

$$M_{3x3} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

translate

$$M_{3x3} = \begin{pmatrix} 1 & 0 & t\_x \\ 0 & 1 & t\_y \\ 0 & 0 & 1 \end{pmatrix}$$

$$M_{3x3} = \begin{pmatrix} m00 & m01 & t\_x \\ m10 & m11 & t\_y \\ m20 & m21 & 1 \end{pmatrix}$$

3D homogeneous transformations

3D homogeneous transformations

translate

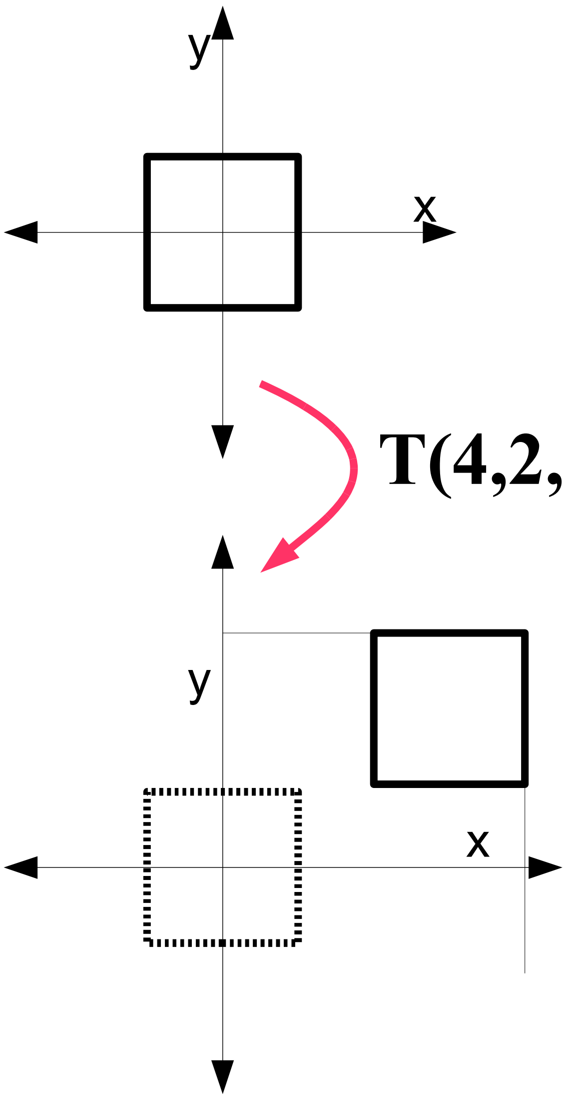
$$\mathbf{T}_{4x4} = \begin{pmatrix} 1 & 0 & 0 & t\_x \\ 0 & 1 & 0 & t\_y \\ 0 & 0 & 1 & t\_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

translate **T** an entity by a vector **t = (t_x ,t_y ,t_z)**

3D homogeneous transformations

translate

$$T_{4x4} = \begin{pmatrix} 1 & 0 & 0 & t\_x \\ 0 & 1 & 0 & t\_y \\ 0 & 0 & 1 & t\_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

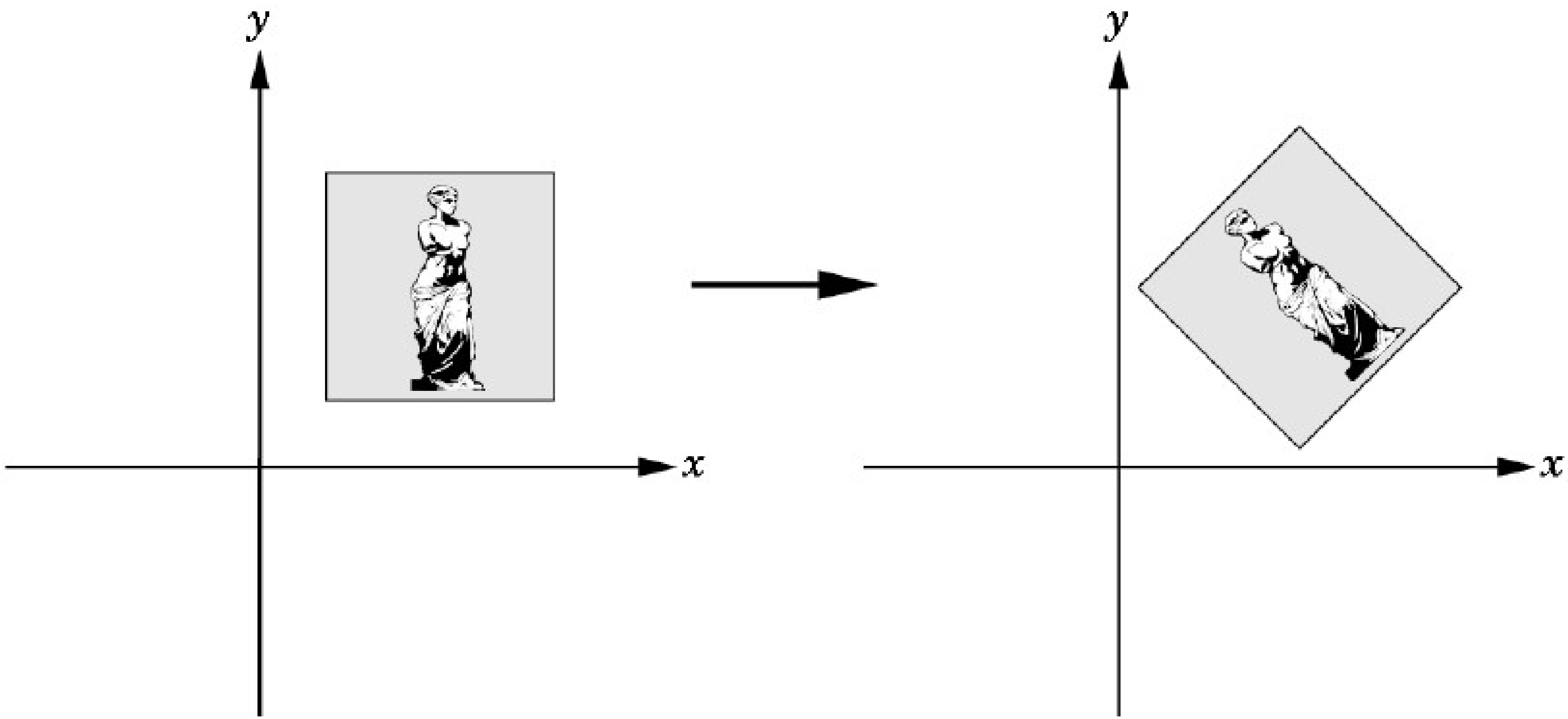


p = (p_x ,p_y ,p_z,1) with T(t) yields a new point **p'**

p'= (p_x+ t_x , p_y+t_y , p_z+t_z,1)

3D homogeneous transformations

rotate



$R_x\alpha$, $R_y\alpha$, $R_z\alpha$, which rotate an entity

α radians around XYZ

3D homogeneous transformations

rotate

$$\mathbf{R_x}\boldsymbol{\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{R_y}\boldsymbol{\alpha} = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R_z}\boldsymbol{\alpha} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\mathbf{R_x}\boldsymbol{\alpha}$**,** $\mathbf{R_y}\boldsymbol{\alpha}$**,** $\mathbf{R_z}\boldsymbol{\alpha}$**,** which rotate an entity
$\boldsymbol{\alpha}$ radians around XYZ

3D homogeneous transformations

rotate inverse

$$\mathbf{R}_i^{-1}(\alpha) = \mathbf{R}_i(-\alpha)$$

rotate in the opposite direction around the same axis

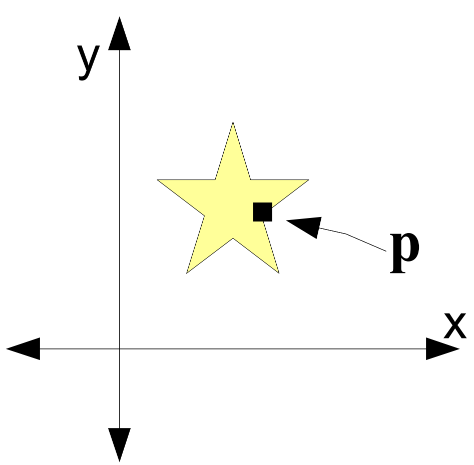
3D homogeneous transformations

rotation around a point

3D homogeneous transformations

rotation around a point

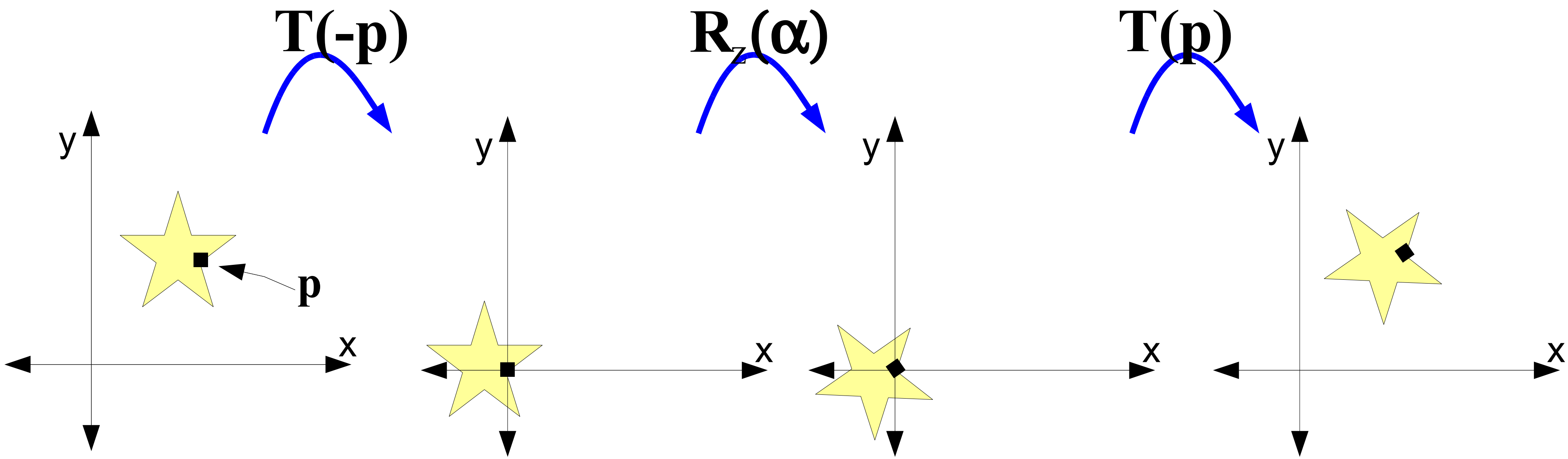
rotate an object **α radians** around the **z-axis**, with the center of rotation being point **p**



3D homogeneous transformations

rotation around a point

rotate an object **α radians** around the **z-axis**, with the center of rotation being point **p**

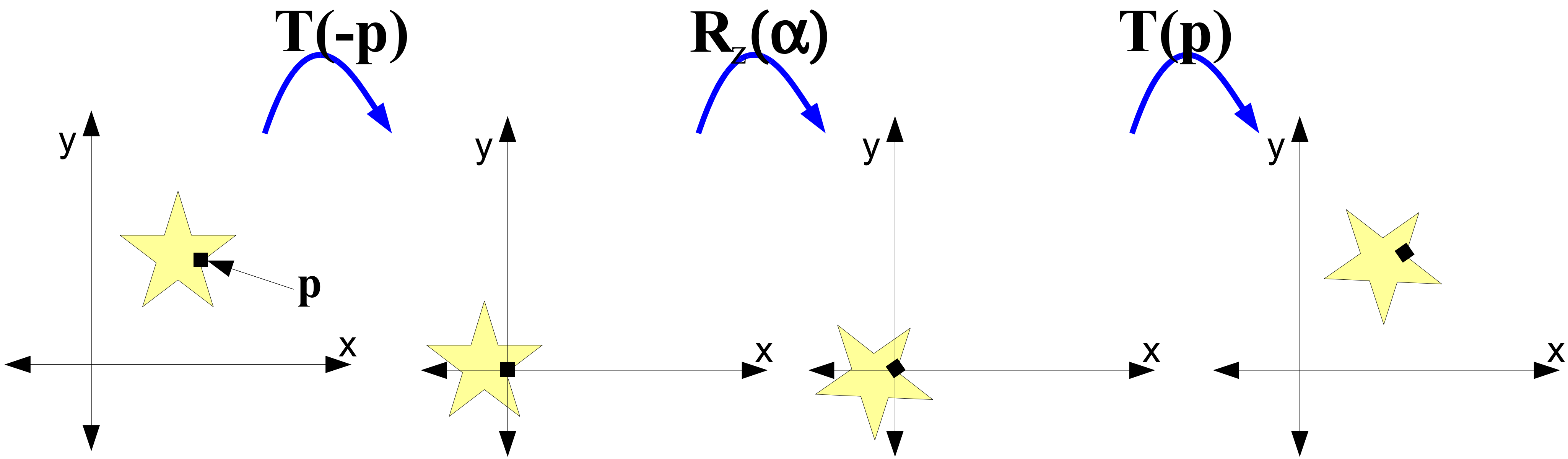


3D homogeneous transformations

rotation around a point

rotate an object **α radians** around the **z-axis**, with the center of rotation being point **p**

$$\mathbf{X = T(p)R_z(\alpha)\, T(-p)}$$



3D homogeneous transformations

scale

$$\mathbf{S}_{4x4} = \begin{pmatrix} s\_x & 0 & 0 & 0 \\ 0 & s\_y & 0 & 0 \\ 0 & 0 & s\_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D homogeneous transformations

scale

$$S_{4x4} = \begin{pmatrix} s\_x & 0 & 0 & 0 \\ 0 & s\_y & 0 & 0 \\ 0 & 0 & s\_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

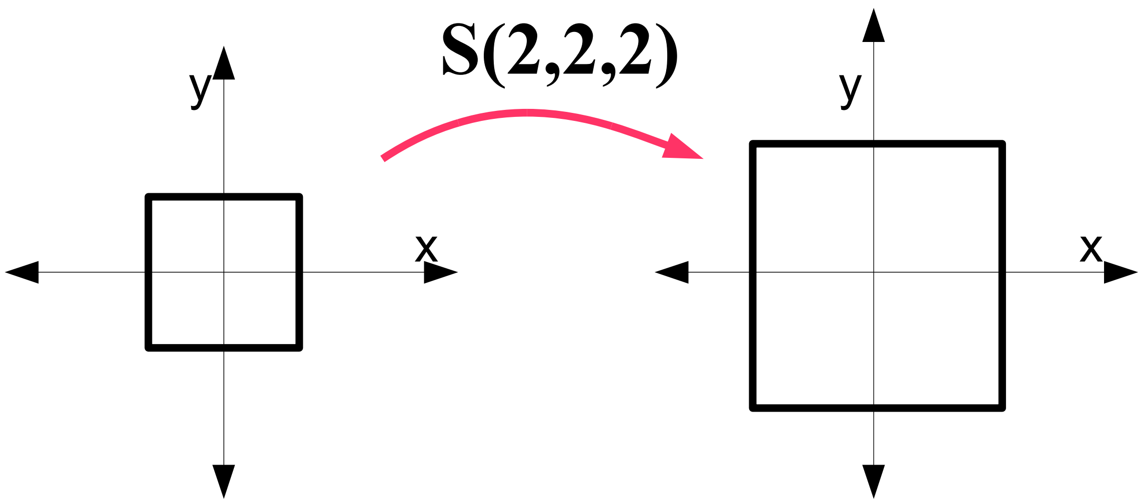
inverse

$$S^{-1}(s) = S\ (1/s\_x,\ 1/s\_y,\ 1/s\_z)$$

3D homogeneous transformations

scale (example)

$$S_{4x4} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



a 2D representation of a 3D object

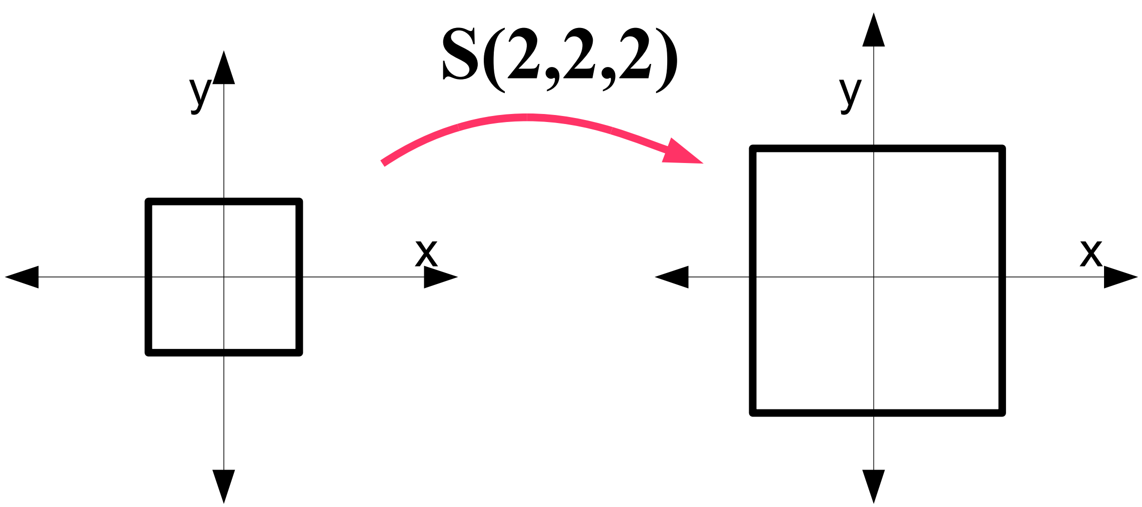
3D homogeneous transformations

scale (example)

$$S_{4x4} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

alternative

$$S_{4x4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/2 \end{pmatrix}$$



a 2D representation of a 3D object

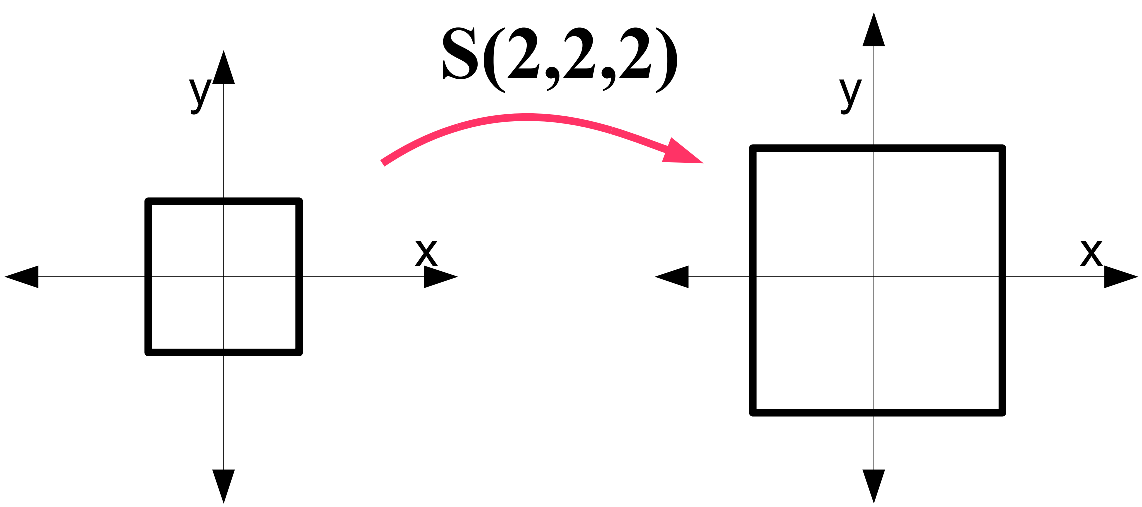
3D homogeneous transformations

scale (example)

$$S_{4x4} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

alternative

$$S_{4x4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/2 \end{pmatrix}$$



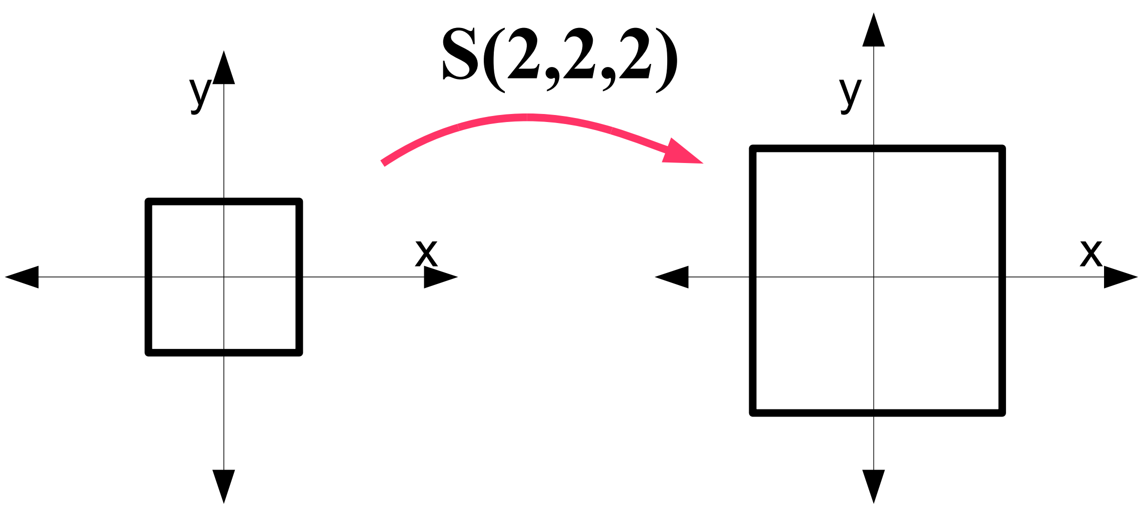
a 2D representation of a 3D object

needs homogenization

3D homogeneous transformations

scale (example)

$$S_{4x4} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



a 2D representation of a 3D object

alternative

$$S_{4x4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/2 \end{pmatrix}$$

needs homogenization

is this solution efficient ?

3D homogeneous transformations

how should the **matrix** be if you would like to **mirror an object**?

3D homogeneous transformations

how should the **matrix** be if you would like to **mirror an object**?

mirror on y-axis

$$\mathbf{S}_{4x4} = \begin{pmatrix} s\_x & 0 & 0 & 0 \\ 0 & \boxed{-s\_y} & 0 & 0 \\ 0 & 0 & s\_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D homogeneous transformations

what happens to the triangles of a 3D model when they are mirrored?

3D homogeneous transformations

what happens to the triangles of a 3D model when they are mirrored?

check if triangles vertices are clockwise or counterclockwise.

incorrect lighting and backface culling may occur.

3D homogeneous transformations

concatenation of transforms

multiplication operation on matrices is **noncommutative**

3D homogeneous transformations

concatenation of transforms

multiplication operation on matrices is **noncommutative**

↓

order in which matrices occur matters

3D homogeneous transformations

concatenation of transforms

↓

order-dependent

3D homogeneous transformations

concatenation of transforms

↓

order-dependent

gain efficiency

eg. several thousand of vertices all scale, rotate and translate at once

3D homogeneous transformations

the rigid-body transform

3D homogeneous transformations

the rigid-body transform

what happens if we take a box from a table and move it to another location?

what attributes change?

3D homogeneous transformations

the rigid-body transform

what happens if we take a box from a table and move it to another location?

what attributes change?

. object orientation and location **change**

. the shape of the object is **not affected**

3D homogeneous transformations

the rigid-body transform

preserves: lengths, angles and handedness

3D homogeneous transformations

the rigid-body transform

concatenation of a translation matrix **T(t)**
and a rotation matrix **R**

preserves: lengths, angles and handedness

3D homogeneous transformations

the rigid-body transform

$$\mathbf{X} = \mathbf{T}(\mathbf{t})\mathbf{R}$$

concatenation of a translation matrix $\mathbf{T}(\mathbf{t})$
and a rotation matrix $\mathbf{R}$

preserves: lengths, angles and handedness

3D homogeneous transformations

the rigid-body transform

$$\mathbf{X} = \mathbf{T}(\mathbf{t})\mathbf{R}$$

concatenation of a translation matrix $\mathbf{T}(\mathbf{t})$
and a rotation matrix $\mathbf{R}$

which is the appearance of the matrix?

preserves: lengths, angles and handedness

3D homogeneous transformations

the rigid-body transform

$$\mathbf{X} = \mathbf{T(t)R}$$

concatenation of a translation matrix **T(t)**
and a rotation matrix **R**

$$\mathbf{X}=\mathbf{T(t)R}=\begin{pmatrix} r00 & r01 & r02 & t\_x \\ r10 & r11 & r12 & t\_y \\ r20 & r21 & r22 & t\_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

preserves: lengths, angles and handedness

advance topics & references

Advance topics

. non rigid-body transform

. quaternions

References

. chapter 5 & 6

Book: Fundamentals of Computer Graphics (P. Shirley et al.)