EBERHARD KARLS UNIVERSITÄT TÜBINGEN

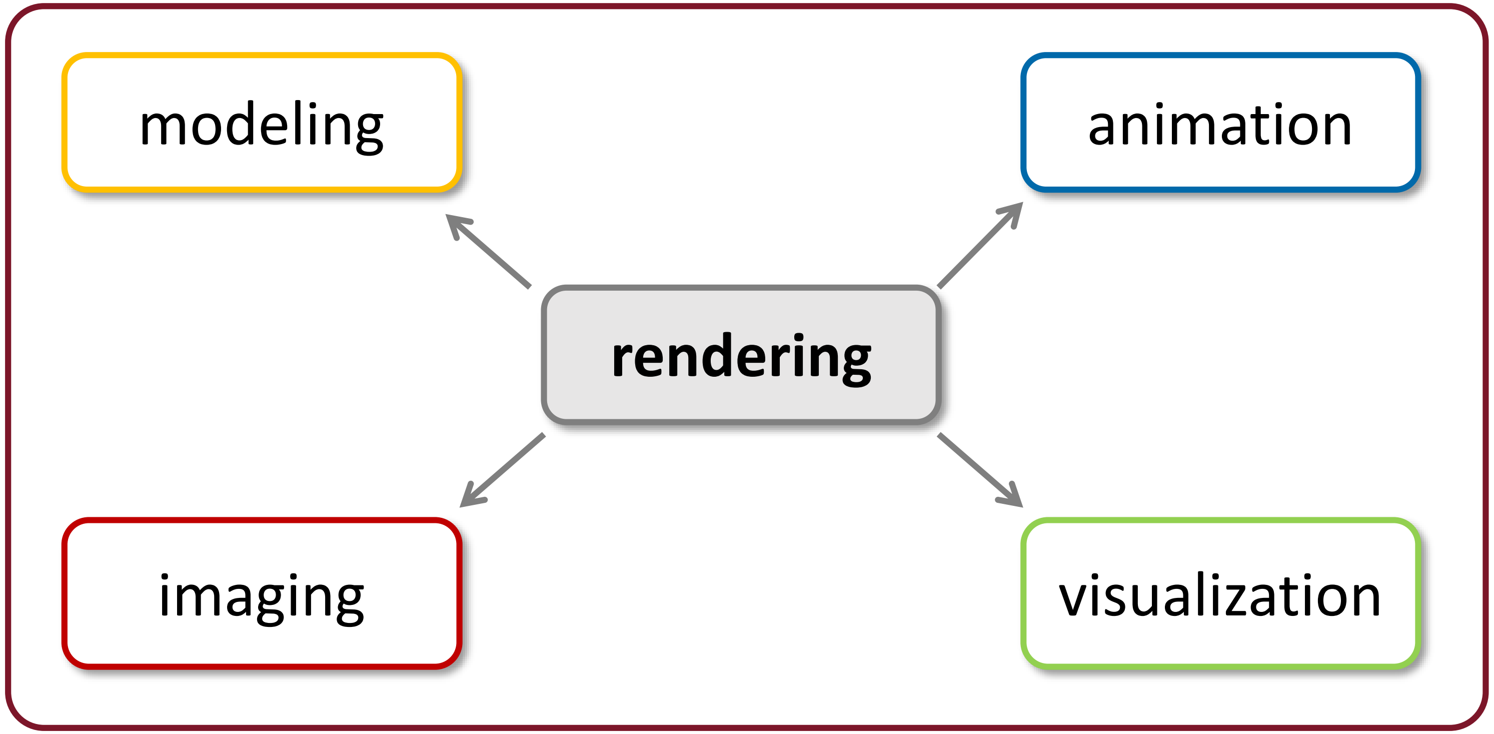
Computer Graphics

Scientific Visualization – Summer Semester 2021

Jun.-Prof. Dr. **Michael Krone**

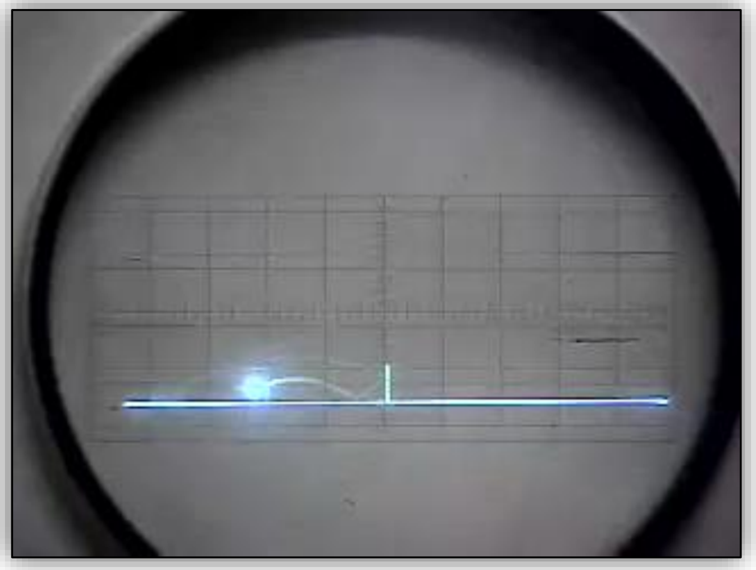
What is Computer Graphics?

- Generation and manipulation of images with computers
- Research areas:



Evolution of Computer Graphics in Video Games

- Obviously, CG development was partially motivated by a ludic drive…



Tennis for Two, 1958
William Higinbotham
Analog computer and oscillograph

Spacewar!, 1961
MIT Students
DEC PDP-1

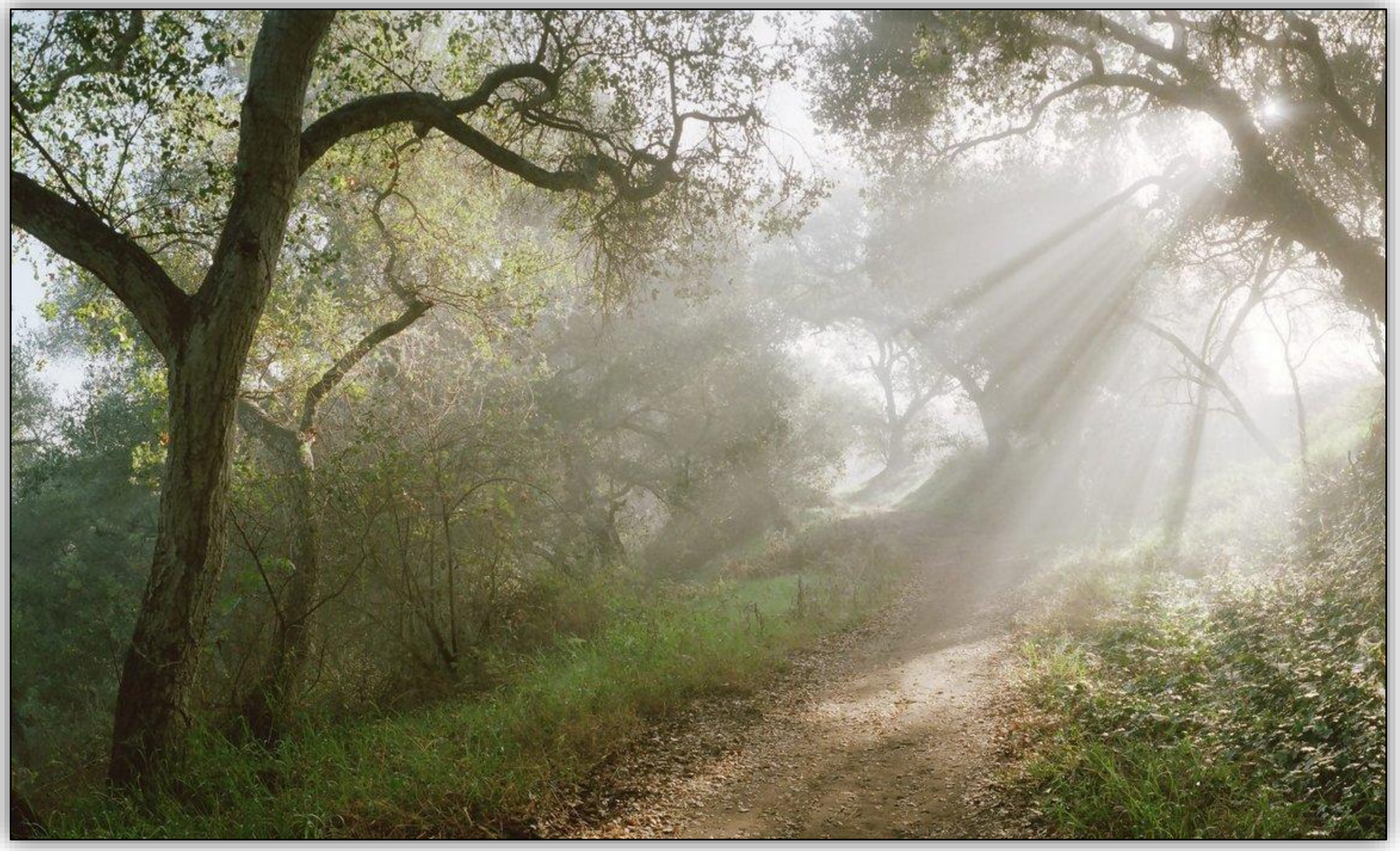
Evolution of Computer Graphics in Video Games

1962 1978 1981 1991 1998

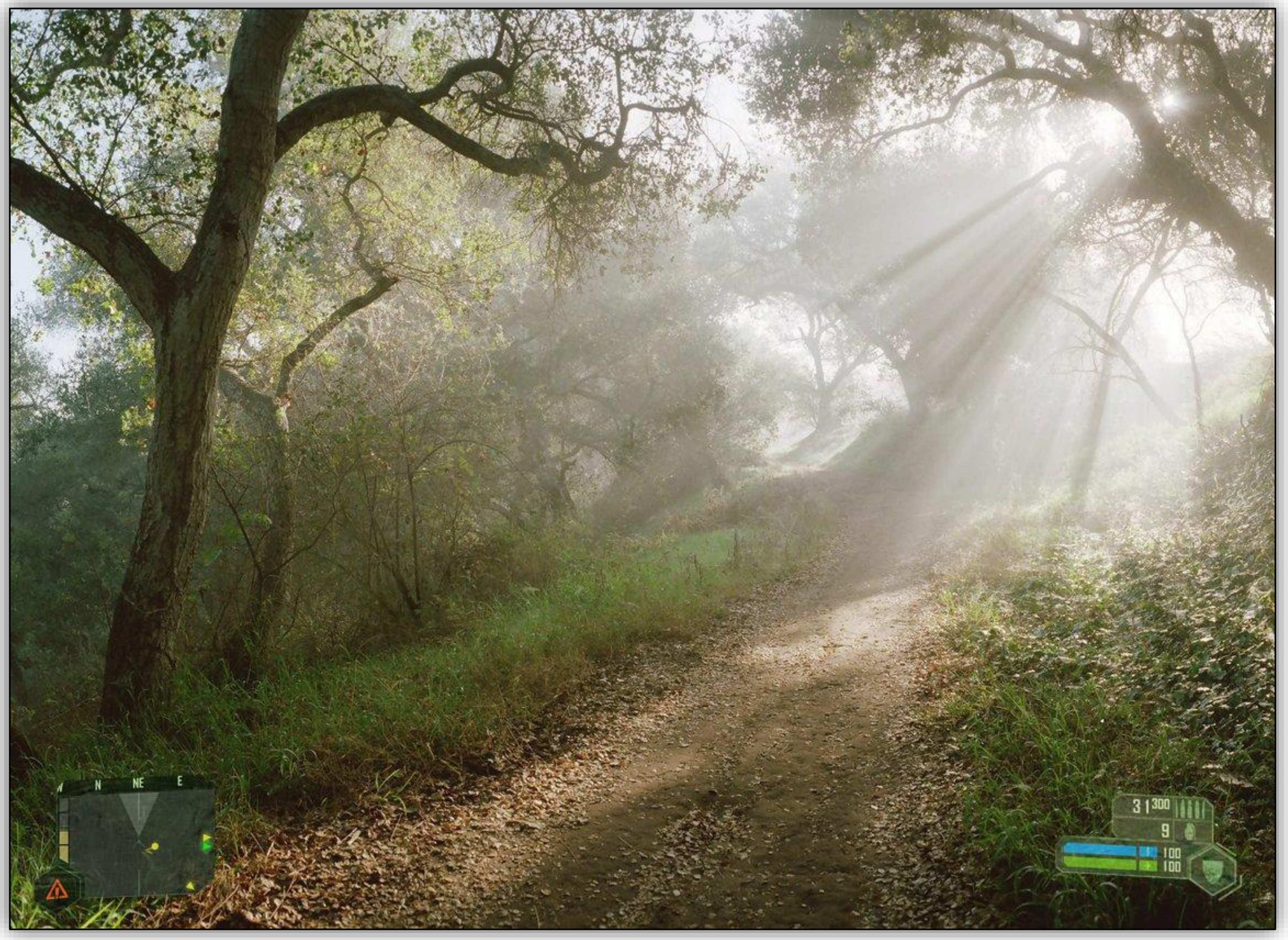
2010-2020

→ Towards photorealistic rendering

When will games reach this degree of realism?



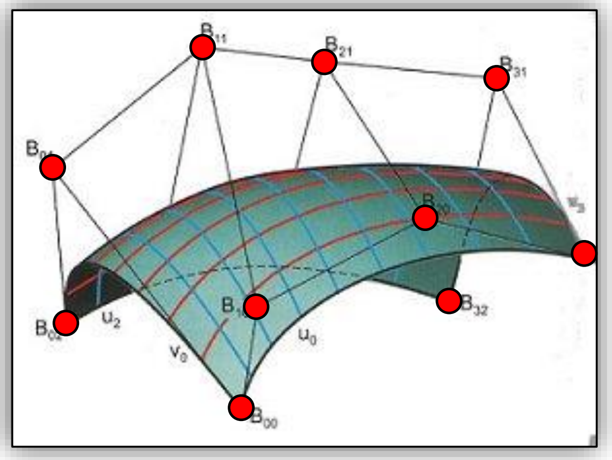
...they already have!



3D Geometry: Description of Shape of Objects

- Depiction of the surface
  - Usually via triangles
  - Tessellation (amount/granularity of triangles)

- Free form surfaces
  - Developed independently by Pierre Bézier (Renault) and Paul de Casteljau (Citroën) for the computer-aided construction of car bodies



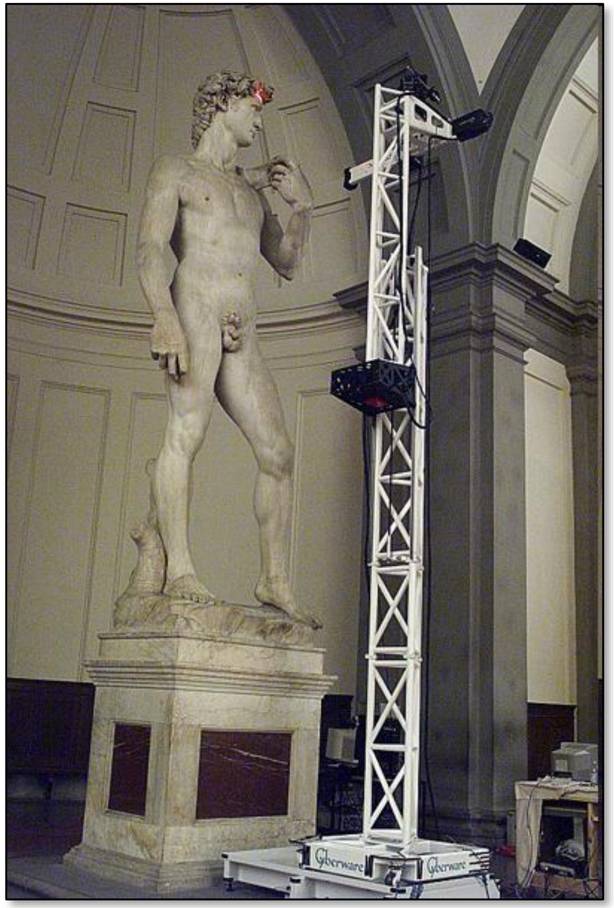
Procedural Models – Example: Rocks

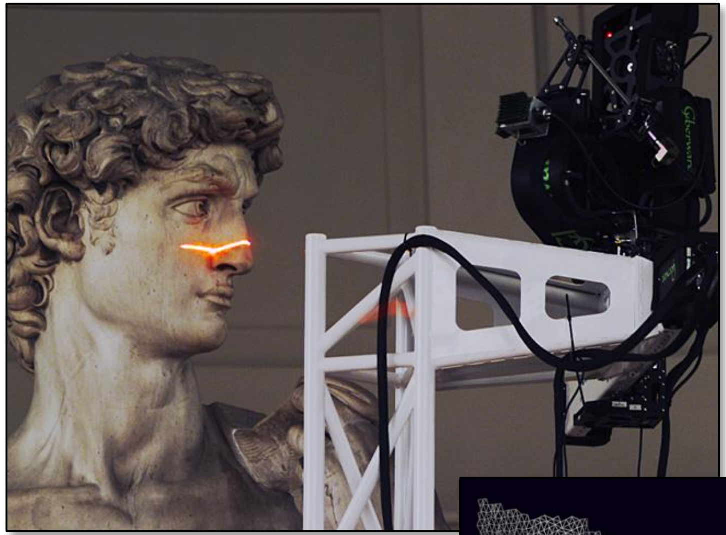
- Generate randomly distributed points and from them, coarse meshes
- Subdivid

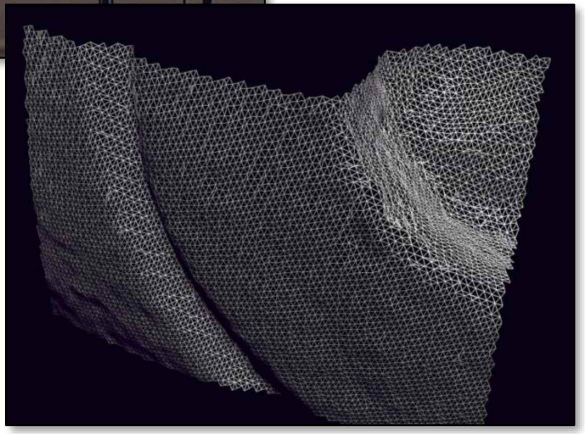
Image: Frank Doassans

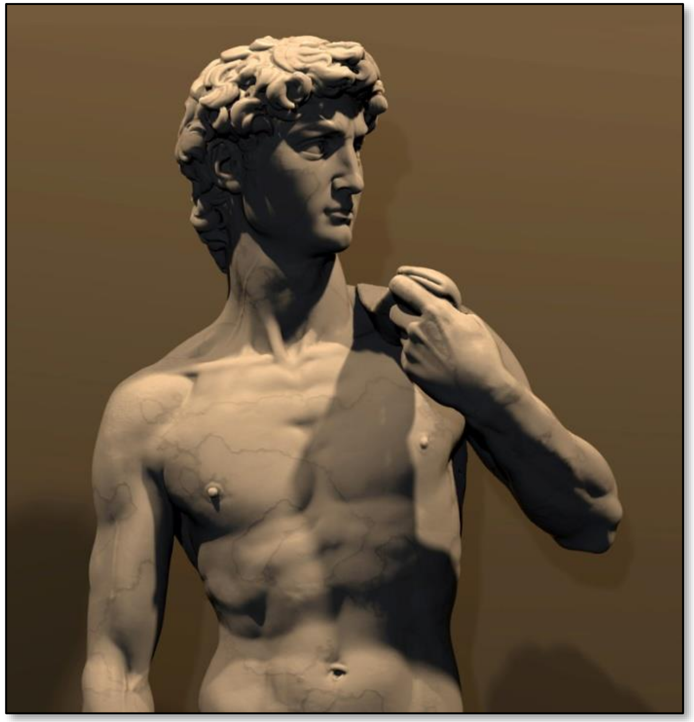
Detailed Geometry

- 3D Scanning: Acquisition of surfaces with a laser









www-graphics.stanford.edu/projects/mich/

What else do we need?

- Material properties (reflectance, opacity etc.)
- Shading, lighting (e.g., photorealistic or illustrative)
- Animation
- …

© University of Utah

Pixels vs. 2D vs. 3D Graphics

- Pixel-based graphics
  - Given resolution, describe color at each pixel
  - Basis for digital photography
  - Whole research area of image processing
- 2D graphics (*a.k.a.* vector graphics)
  - Uses 2D lines and areas to describe an image
  - 2D drawing programs: Inkscape, Adobe Illustrator, MS PowerPoint,…
  - Rasterize as pixel image for a specific resolution
- 3D graphics
  - Describe 3D objects of a scene
  - Compute what light would do to these objects
  - Compute pixel image from a virtual camera


© Maarten Boot


© KeithByrne (via DeviantArt)


© Crystal Dynamics / Square Enix

Why should I learn about Computer Graphics?

- Basis for graphical digital media
  - Core discipline for of many jobs
- Basis for recent CG movies and SFX
  - Practically no more movies without it!
- Basis for most computer games
  - Market bigger than the film industry
- **Basis for scientific visualization**
  - **Graphical depiction of scientific data**

→Lecture “Graphische Datenverarbeitung” (GDV)

- Winter semester, Prof. Dr. Hendrik Lensch

© Marvel Studios/Walt Disney Studios Motion Pictures

Image source: https://www.dirtgame.com/

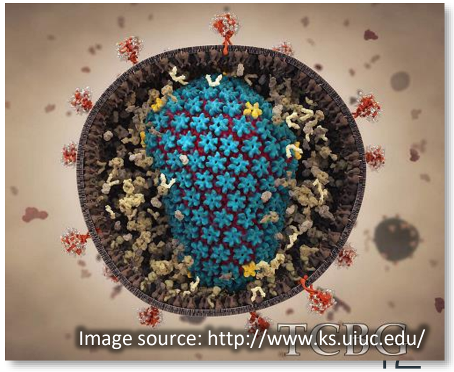
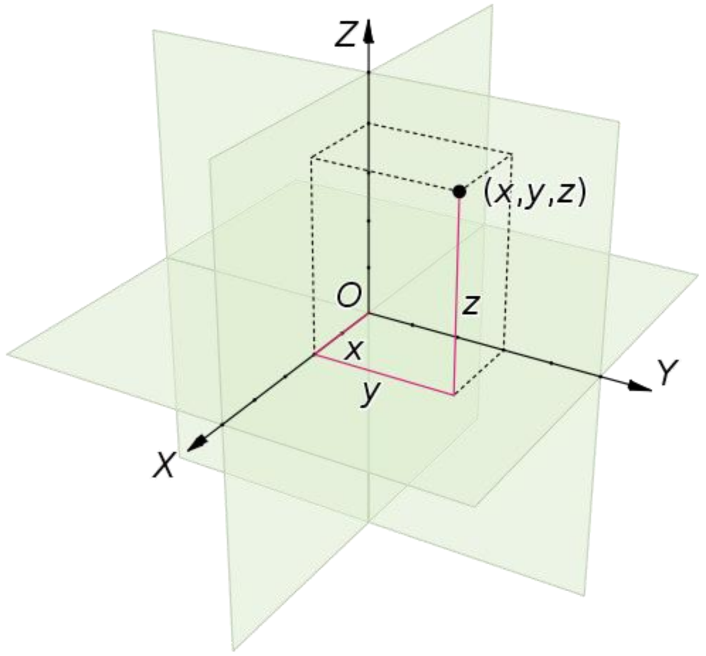


Image source: http://www.ks.uiuc.edu/

Math Recap: What We Need to Survive…

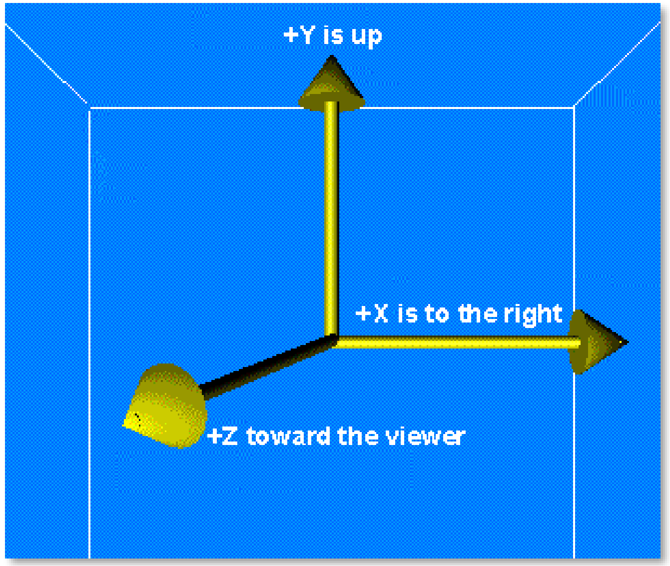
- Coordinate Reference Frames
  - Embeds all objects that we want to render
- Dimensionality
  - We will meet 2, 3 and 4 dimensions
- Types of coordinate systems
  - Usually Cartesian (rectilinear):
    Pairwise orthogonal axes with (identical) linear scale



Images: Wikipedia

Standard 3D Cartesian Coordinate Reference Frames

- Most frequently used "world coordinates" (e.g., OpenGL / WebGL): "Right handed" system, often depicted as looking from z axis



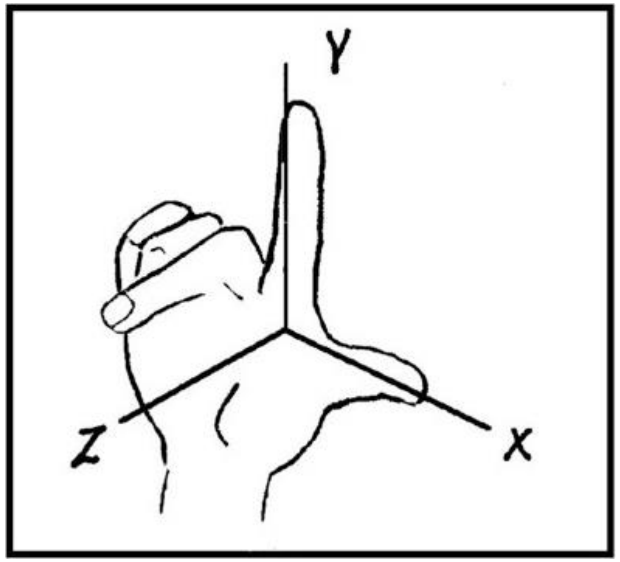


Image sources: euclidianspace.com, cornell.edu

- "Left handed" system used in special cases (e.g. 2D screen positions with additional depth information)

Points and Vectors

- Point
  - Fixed position specified with coordinate values in reference frame
    - e.g. in 3D Cartesian coordinates: (p_x, p_y, p_z)

$$P = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$$

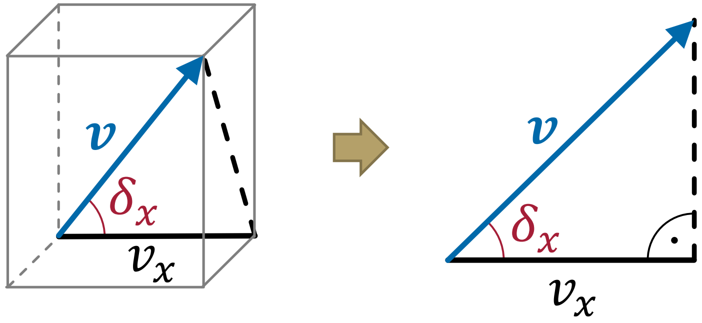
- Vector
  - Tuple of real numbers, considered as element of a vector space
  - Direction → Positions can be specified by vector from origin

$$\boldsymbol{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

$$\boldsymbol{v} = (v_x, v_y, v_z)$$

- Properties of Vectors
  - Magnitude (length)
  - Direction angles

$$\|\boldsymbol{v}\| = \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2}$$



$$\cos\delta_x = \frac{v_x}{\|\boldsymbol{v}\|}\,;\ \cos\delta_y = \frac{v_y}{\|\boldsymbol{v}\|}\,;\ \cos\delta_z = \frac{v_z}{\|\boldsymbol{v}\|}$$

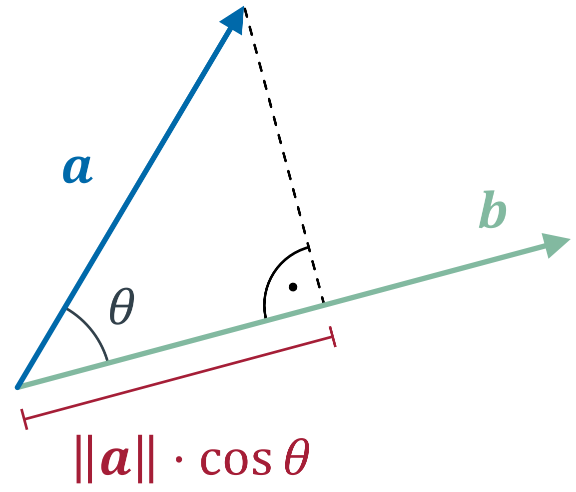
Scalar Product (Dot Product)

- The dot product computes a real (scalar) value from two coordinate vectors of equal dimension

$$\boldsymbol{a} \bullet \boldsymbol{b} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \bullet \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = a_x b_x + a_y b_y + a_z b_z = \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\| \cdot \cos\theta$$

- Applications:
  - Computation of angle between two coordinate vectors
  - Scalar projection of vector A in direction B

$$\boldsymbol{a_b} = \boldsymbol{a} \bullet \frac{\boldsymbol{b}}{\|\boldsymbol{b}\|} = \|\boldsymbol{a}\| \cdot \cos\theta$$



Cross Product (Vector Product)

- The cross product of two coordinate vectors is a vector that is perpendicular to both given vectors
  - Direction: Right-hand rule
  - Magnitude: Equals spanned parallelogram

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \times \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$$

$$\boldsymbol{a} \times \boldsymbol{b} = -(\boldsymbol{b} \times \boldsymbol{a})$$

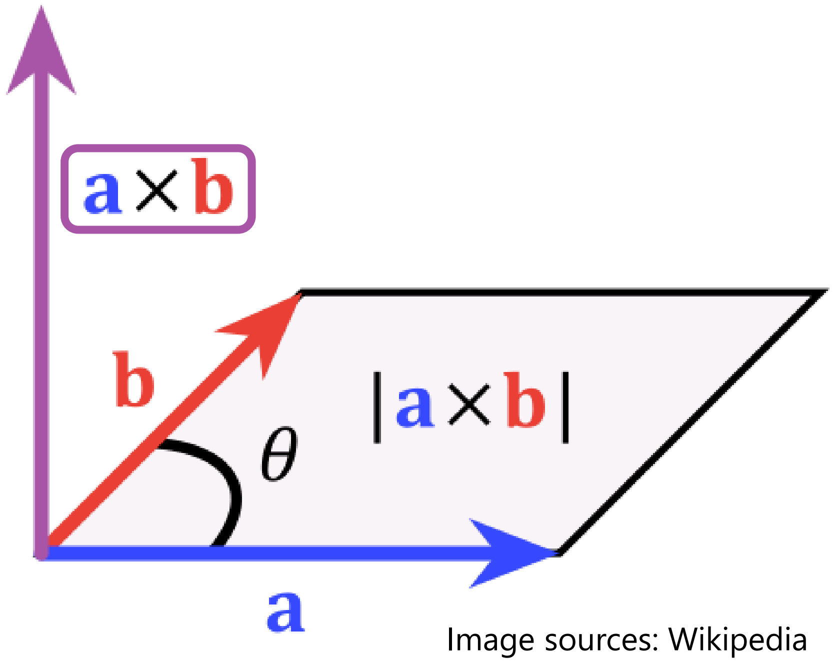
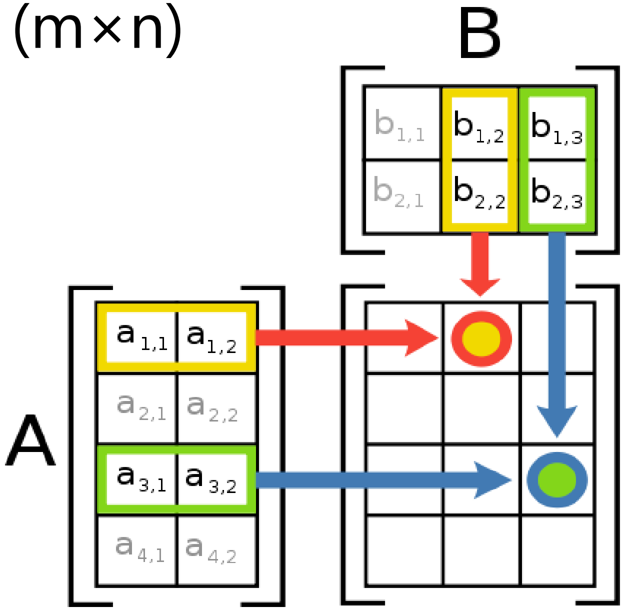


Image sources: Wikipedia

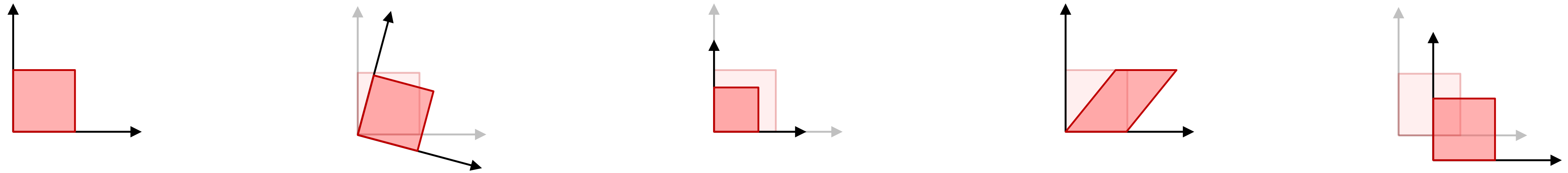
Matrices

- (m×n) arrangement of real numbers (m rows, n columns)
- A matrix can be multiplied with a real number pointwise
- Two matrices of identical dimensions can be added pointwise
- Matrix-matrix / matrix-vector multiplication:
  - $A \cdot B = C$ → A = (m×p) multiplied by B = (p×n) gives C = (m×n)
  - A vector of length p can be seen as a (p×1) matrix

$$C_{i,j} = \sum_{k=1}^{p} A_{i,k} \cdot B_{k,j} \qquad \begin{array}{c} 1 \leq i \leq m \\ 1 \leq j \leq n \end{array}$$



Affine Transformations



Identity　　Rotation　　Scaling　　Shearing　　Translation

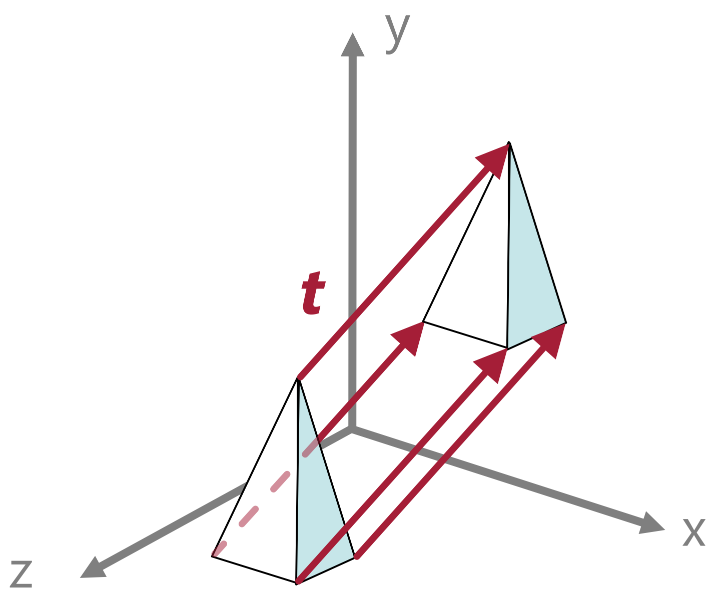
- Combinable (associative, but not commutative)
- Reversible/invertible (except scaling with zero)
- Can be expressed as linear function of the previous coordinate values:

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3^1 \end{pmatrix} = \underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}}_{\text{Rotation, Scaling, Shearing}} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} + \underbrace{\begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}}_{\text{Translation}} = A \cdot p + t$$

Affine Transformations

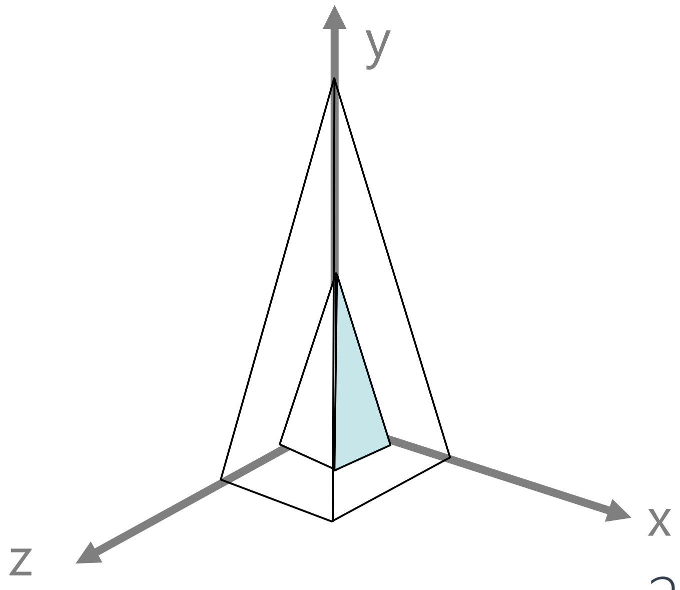
- Translation
  - Add a vector *t*
  - Geometrical meaning: shifting/moving objects

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} p_1 + t_1 \\ p_2 + t_2 \\ p_3 + t_3 \end{pmatrix}$$



- Uniform Scaling
  - Multiply with a scalar *s*
  - Geometrical meaning: Changing the size of an object

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \cdot s = \begin{pmatrix} p_1 \cdot s \\ p_2 \cdot s \\ p_3 \cdot s \end{pmatrix}$$



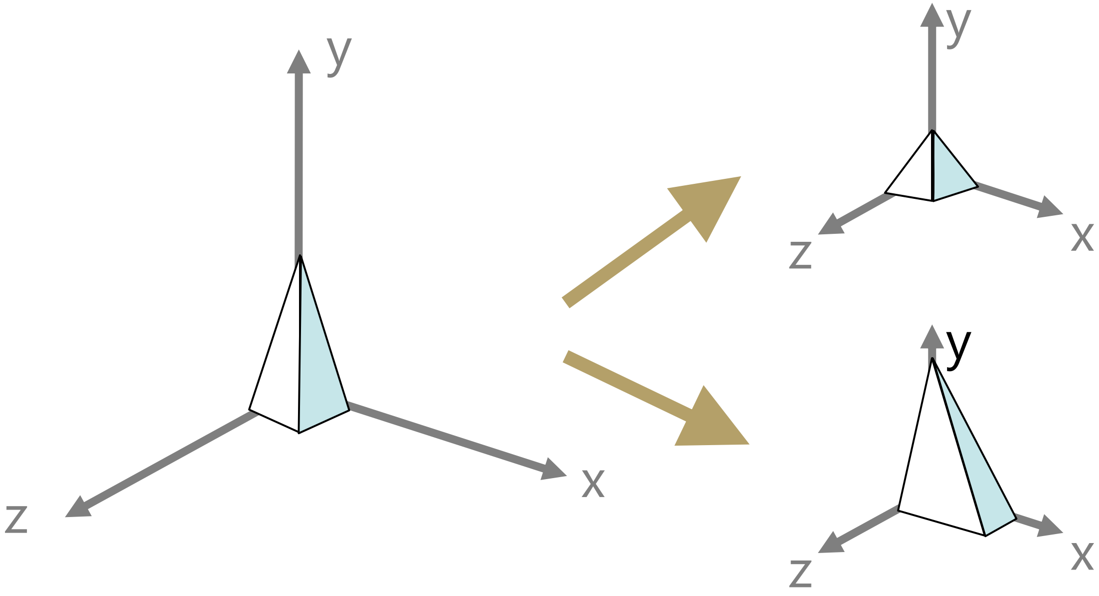
Affine Transformations

- Non-Uniform Scaling
  - Multiply with three scalars
  - One for each dimension

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \cdot s_1 \\ p_2 \cdot s_2 \\ p_3 \cdot s_3 \end{pmatrix}$$


Utah Teapot (original)

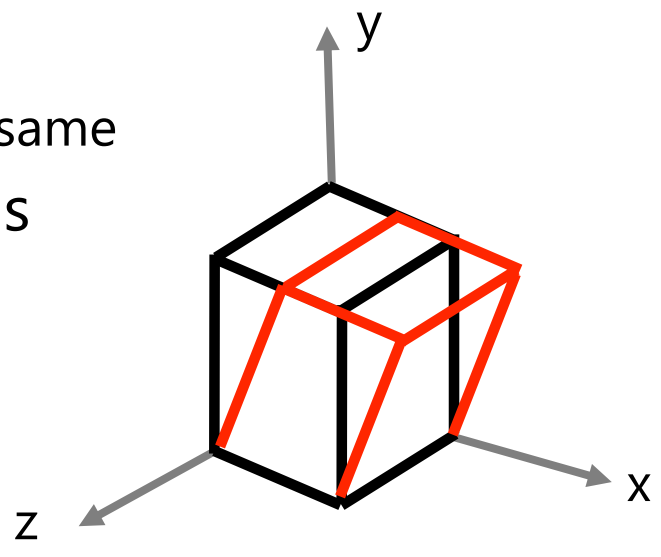
3D model, scaled along y axis



Affine Transformations

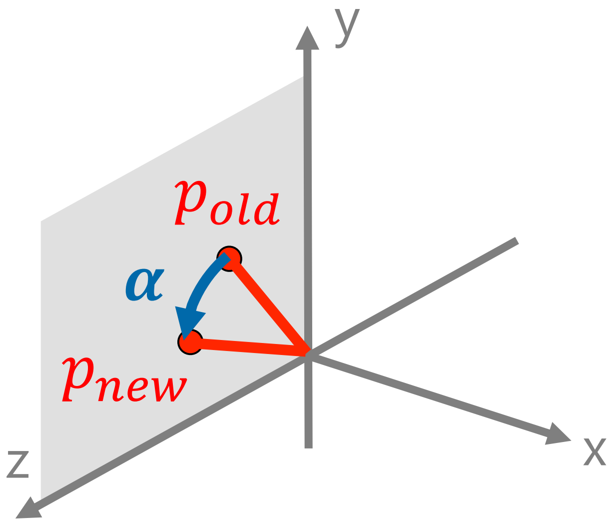
- Shearing
  - Example: Shearing along x axis
    - Only x coordinate values are modified
    - Modification depends linearly on y coordinate value
    - Areas in x/y plane and x/z plane (as well as volume) remain the same
  - Generalization to other axes and arbitrary axis: analogous

$$\begin{pmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 + m \cdot p_2 \\ p_2 \\ p_3 \end{pmatrix}$$



Affine Transformations

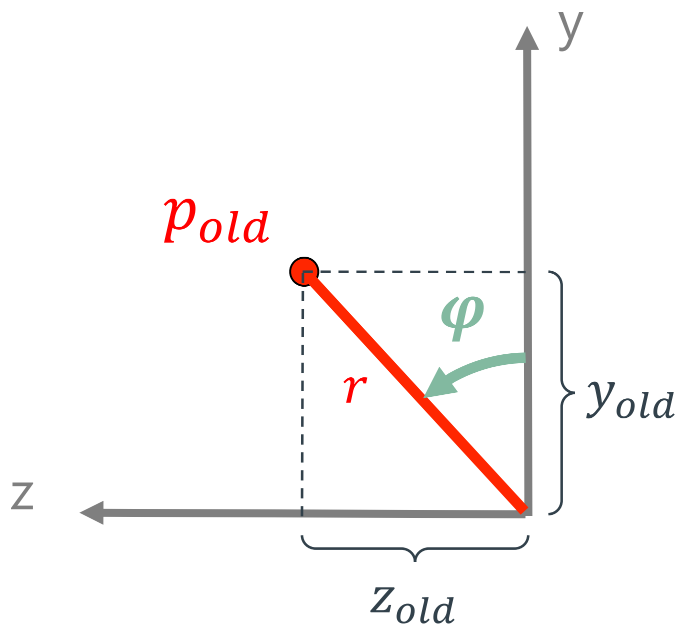
- Rotation about X Axis
  - x coordinate value remains constant
  - Rotation takes place in y/z-plane (2D)



$$p_{old} = \begin{pmatrix} x_{old} \\ y_{old} \\ z_{old} \end{pmatrix}$$

$$p_{new} = \begin{pmatrix} x_{new} \\ y_{new} \\ z_{new} \end{pmatrix}$$

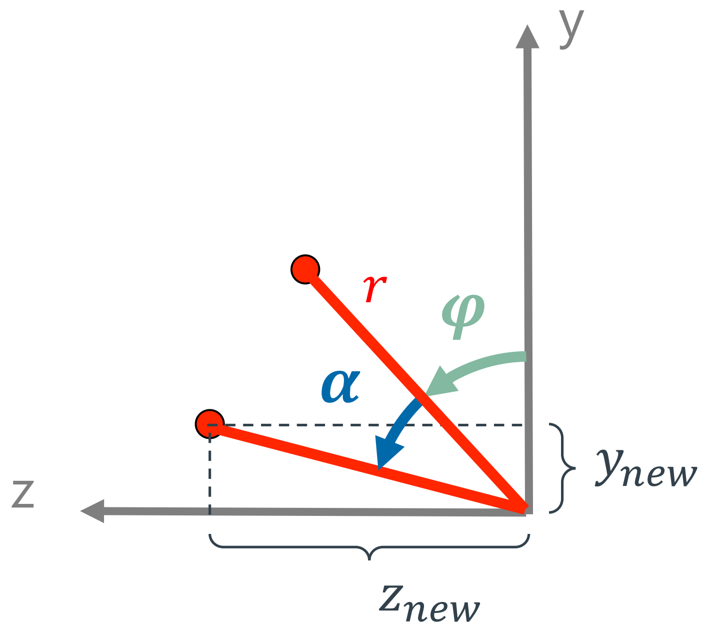
$$r = \|p_{old}\|$$



$$\cos\varphi = \frac{y_{old}}{r} \qquad y_{old} = r \cdot \cos\varphi$$

$$\sin\varphi = \frac{z_{old}}{r} \qquad z_{old} = r \cdot \sin\varphi$$

Affine Transformations – Rotation about X Axis



$$y_{old} = r \cdot \cos\varphi$$

$$z_{old} = r \cdot \sin\varphi$$

$$\cos(\alpha + \varphi) = \frac{y_{new}}{r}$$

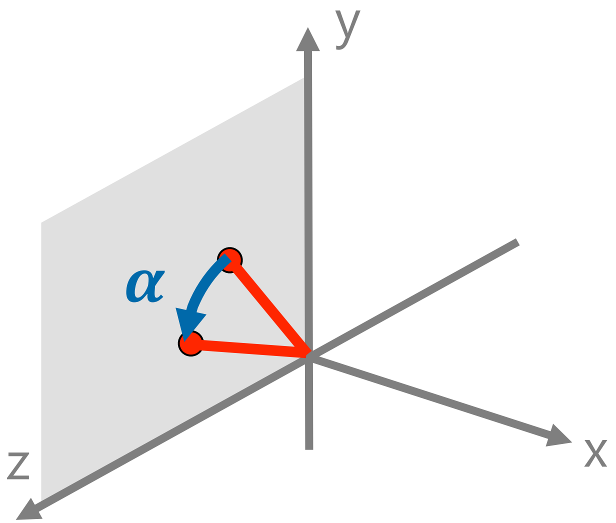
$$\begin{aligned} y_{new} &= r \cdot \cos(\alpha + \varphi) \\ &= r \cdot \cos\alpha \cdot \cos\varphi - r \cdot \sin\alpha \cdot \sin\varphi \\ &= \cos\alpha \cdot y_{old} - \sin\alpha \cdot z_{old} \end{aligned}$$

$$\sin(\alpha + \varphi) = \frac{z_{new}}{r}$$

$$\begin{aligned} z_{new} &= r \cdot \sin(\alpha + \varphi) \\ &= r \cdot \sin\alpha \cdot \cos\varphi + r \cdot \cos\alpha \cdot \sin\varphi \\ &= \sin\alpha \cdot y_{old} + \cos\alpha \cdot z_{old} \end{aligned}$$

Affine Transformations – Rotation about X Axis

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \\ \cos\alpha \cdot p_2 - \sin\alpha \cdot p_3 \\ \sin\alpha \cdot p_2 + \cos\alpha \cdot p_3 \end{pmatrix}$$



- Other axes → analogous
- Combine rotations about main axes to express arbitrary rotation
  - This is not always intuitive
- Order matters (a lot!)
  - Likely source of mistakes/bugs!

Summary: Affine Transformations in $\mathbb{R}^3$

Rotation around x axis: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \\ \cos\alpha \cdot p_2 - \sin\alpha \cdot p_3 \\ \sin\alpha \cdot p_2 + \cos\alpha \cdot p_3 \end{pmatrix}$

Non-uniform **Scaling**: $\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & S_3 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \cdot s_1 \\ p_2 \cdot s_2 \\ p_3 \cdot s_3 \end{pmatrix}$

Shearing along x axis: $\begin{pmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 + m \cdot p_2 \\ p_2 \\ p_3 \end{pmatrix}$

Translation: $\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} p_1 + t_1 \\ p_2 + t_2 \\ p_3 + t_3 \end{pmatrix}$

- “Drawback”? → Translation is a sum, all others are matrix multiplications!

→ **Homogeneous Coordinates**: all transformations ca be expressed as a matrix

Homogeneous Coordinates

- Convert n-dimensional coordinate systems to homogeneous coord's: add additional dimension: $n \rightarrow n+1$ → „scaling factor" h
- 3D position $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ is represented by $\begin{pmatrix} x_\mathrm{h} \\ y_\mathrm{h} \\ z_\mathrm{h} \\ \mathrm{h} \end{pmatrix}$ such that: $x = \frac{x_h}{h}, y = \frac{y_h}{h}, z = \frac{z_h}{h}$
- Simple choice for h is the value 1: $h = 1$ → position vector $\begin{pmatrix} x \\ y \\ z \\ \mathbf{1} \end{pmatrix}$
- 3D directions are represented by $h = 0$ → direction vector $\begin{pmatrix} x \\ y \\ z \\ \mathbf{0} \end{pmatrix}$

Homogeneous Coordinates: Transformations

Rotation around x axis:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 \\ \cos\alpha \cdot p_2 - \sin\alpha \cdot p_3 \\ \sin\alpha \cdot p_2 + \cos\alpha \cdot p_3 \\ 1 \end{pmatrix}$$

Non-uniform **Scaling**:

$$\begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 \cdot p_1 \\ s_2 \cdot p_2 \\ s_3 \cdot p_3 \\ 1 \end{pmatrix}$$

Shearing along x axis:

$$\begin{pmatrix} 1 & m & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 + m \cdot p_2 \\ p_2 \\ p_3 \\ 1 \end{pmatrix}$$

Translation:

$$\begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 + t_1 \\ p_2 + t_2 \\ p_3 + t_3 \\ 1 \end{pmatrix}$$

Position vector → is translated

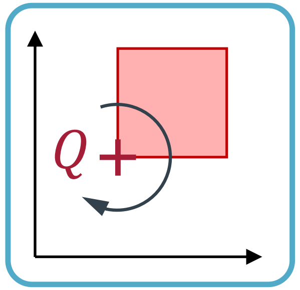
$$\begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 0 \end{pmatrix} = \begin{pmatrix} p_1 + 0 \cdot t_1 \\ p_2 + 0 \cdot t_2 \\ p_3 + 0 \cdot t_3 \\ 0 \cdot 1 \end{pmatrix}$$

Direction vector → **no effect!**

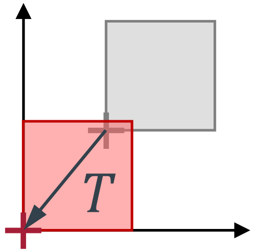
Combination of Transformations

- Combine multiple transformations in one matrix
  - Order matters! → not commutative
  - Transformation and inverse transformation

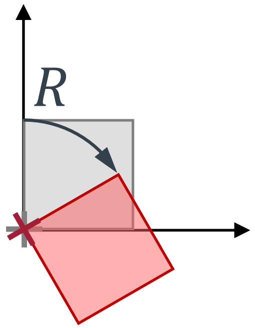
- Example (2D): Rotation ϕ around a point $Q = (q_1,\ q_2)$
  1. Translate by $-Q$ („move Q to the origin“): T
  2. Rotation around ϕ: R
  3. Translate by Q (Invert the first step): T^{-1}
  - $p' = \boxed{T^{-1} \cdot R \cdot T} \cdot p\ = M \cdot p$



1.



2.



3.