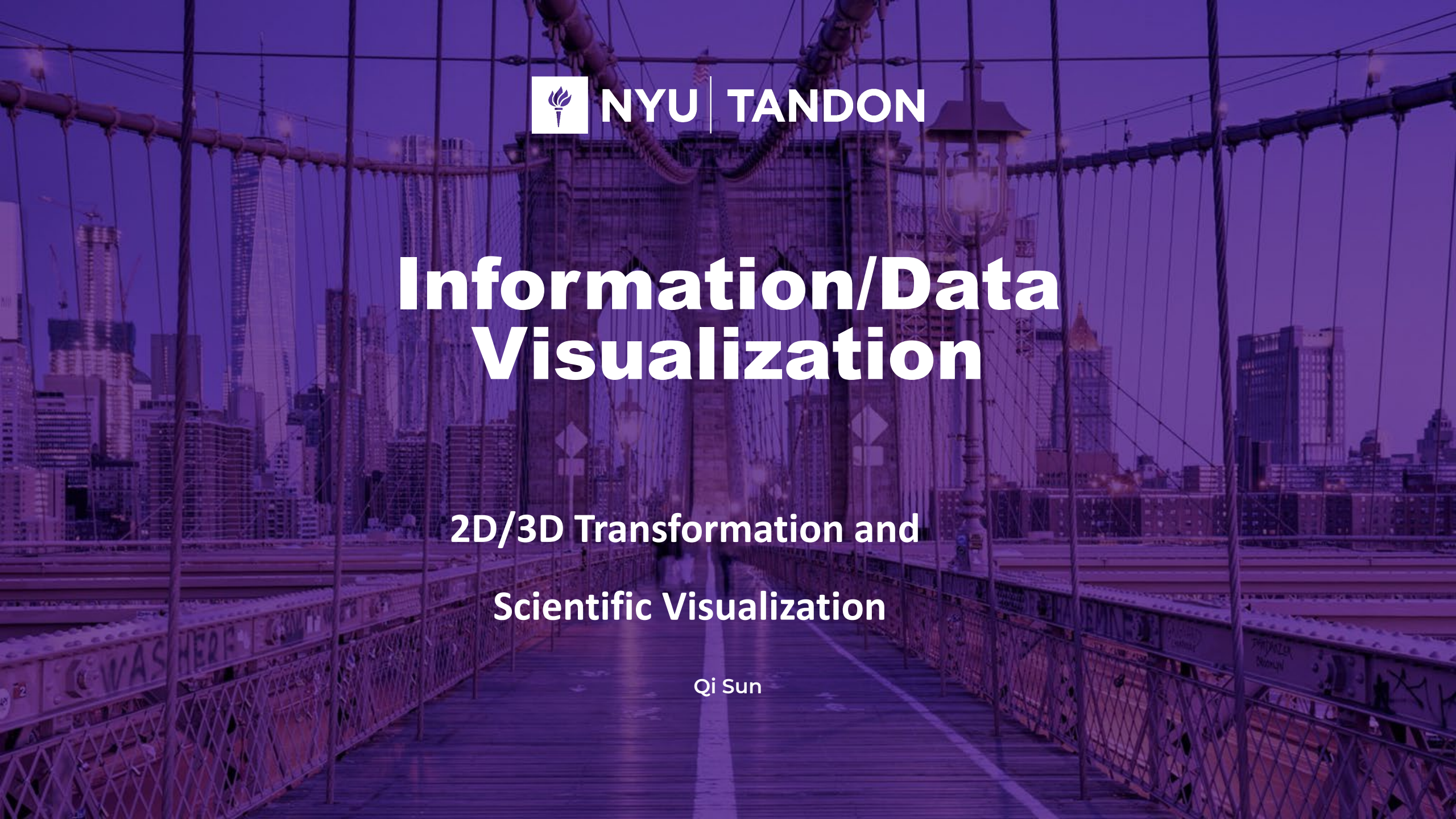


NYU TANDON

Information/Data Visualization

2D/3D Transformation and Scientific Visualization

Qi Sun

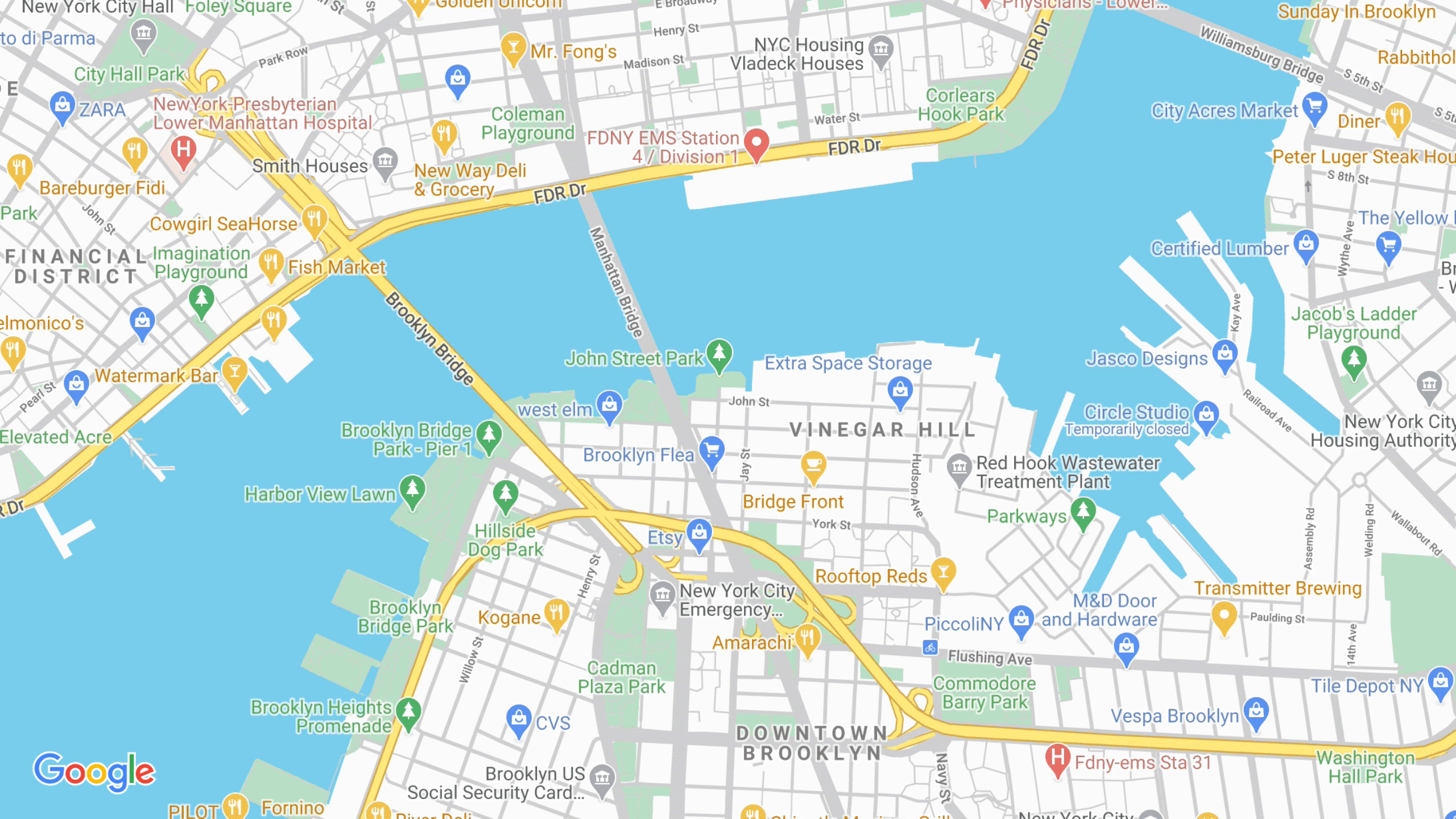
Important Logistics

- Final Project Presentation
  - 3min each: on-site presentation vs pre-recorded video + Q&A
  - engineering volume? complete/end-to-end system >> complexity
  - come to the office hours if you need help!
  - due date: April 30th (video) for May 2nd presentation

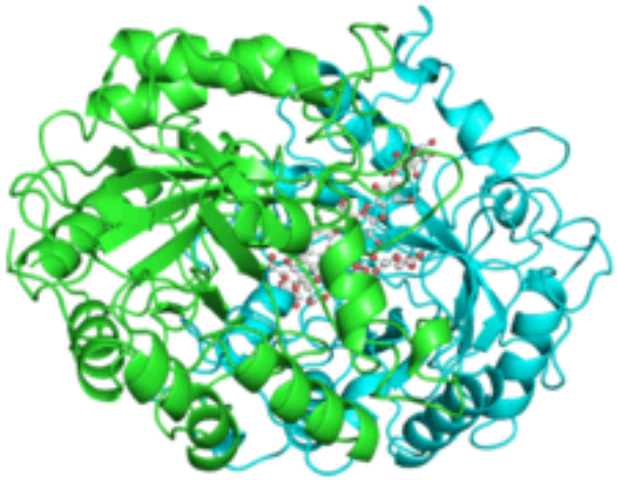
- The last guest lecture next week
  - Wenqi (Wendy) Xian (https://www.cs.cornell.edu/~wenqixian/) from Cornell Tech
  - AI-based neural video synthesis

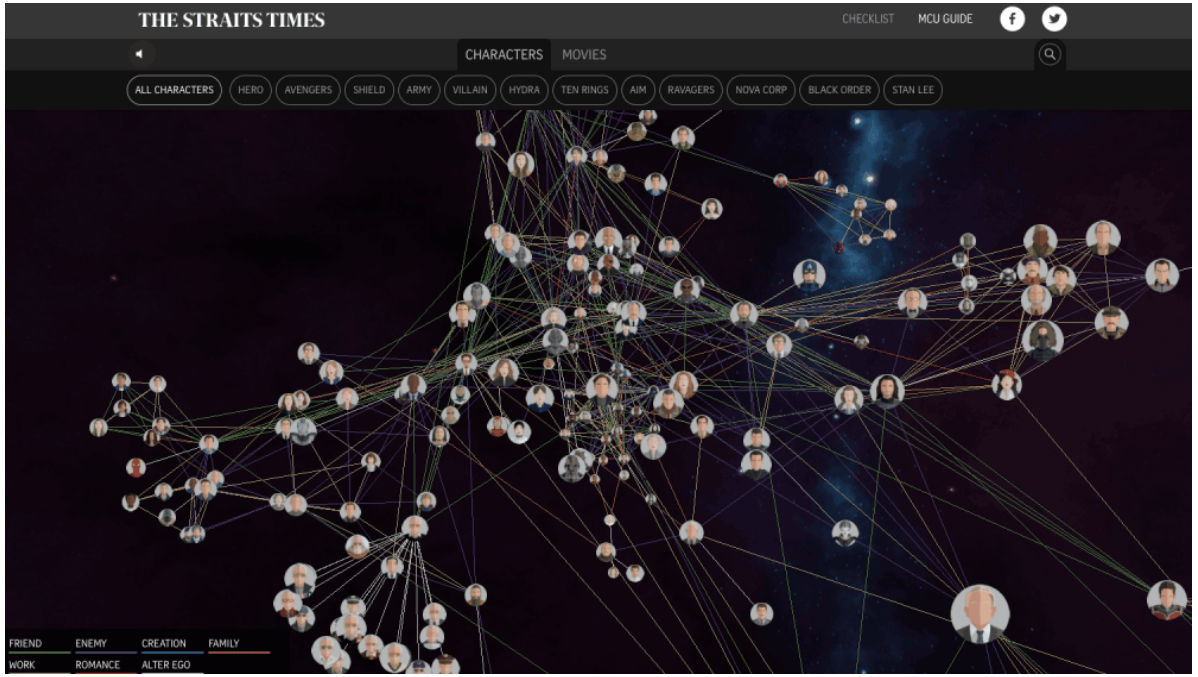
PART 01

Interaction: (2D) Transformation

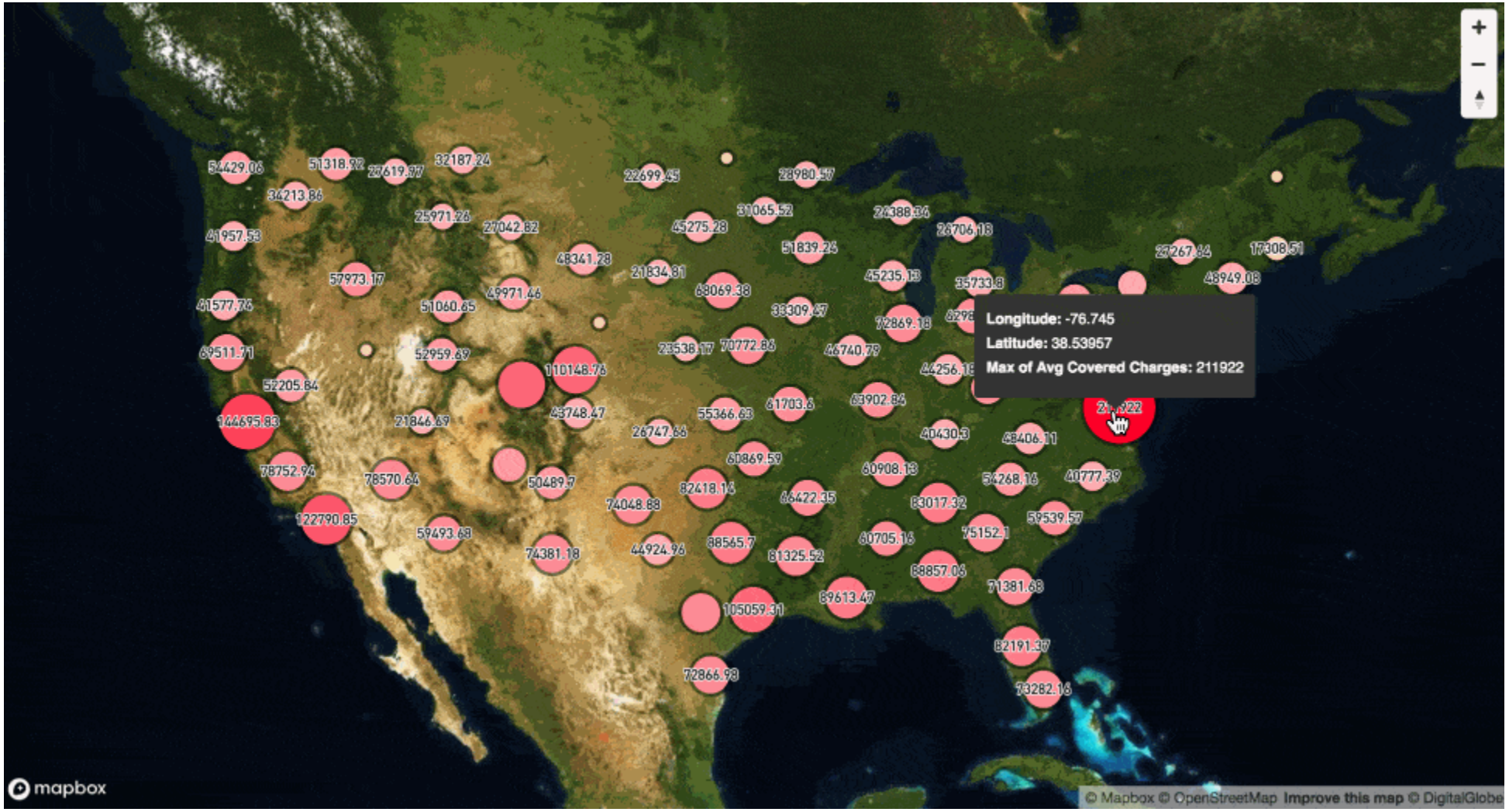

New York City Hall
Foley Square
to di Parma
City Hall Park
Park Row
Mr. Fong's
Henry St
Madison St
NYC Housing
Vladeck Houses
FDR Dr
Williamsburg Bridge
Sunday In Brooklyn
Rabbithol
ZARA
NewYork-Presbyterian
Lower Manhattan Hospital
Coleman
Playground
Water St
Corlears
Hook Park
City Acres Market
Diner
FDNY EMS Station
4 / Division 1
Peter Luger Steak Hou
S 8th St
Smith Houses
New Way Deli
& Grocery
Bareburger Fidi
Park
John St
Cowgirl SeaHorse
FINANCIAL
DISTRICT
Imagination
Playground
Fish Market
Manhattan Bridge
Brooklyn Bridge
The Yellow
Certified Lumber
Wythe Ave
elmonico's
Jacob's Ladder
Playground
Kay Ave
Jasco Designs
John Street Park
Extra Space Storage
Watermark Bar
Pearl St
west elm
John St
Circle Studio
Temporarily closed
Railroad Ave
New York City
Housing Authority
Elevated Acre
Brooklyn Bridge
Park - Pier 1
Brooklyn Flea
VINEGAR HILL
Jay St
Hudson Ave
Red Hook Wastewater
Treatment Plant
Harbor View Lawn
Bridge Front
Parkways
York St
Hillside
Dog Park
Etsy
Assembly Rd
Welding Rd
Wallabout Rd
Rooftop Reds
Transmitter Brewing
Henry St
New York City
Emergency...
M&D Door
and Hardware
Brooklyn
Bridge Park
Kogane
PiccoliNY
Paulding St
Amarachi
Flushing Ave
14th Ave
Cadman
Plaza Park
Commodore
Barry Park
Tile Depot NY
Willow St
Brooklyn Heights
Promenade
CVS
Vespa Brooklyn
DOWNTOWN
BROOKLYN
Fdny-ems Sta 31
Washington
Hall Park
Navy St
Brooklyn US
Social Security Card...
PILOT
Fornino
Google

Transformation - Rotation

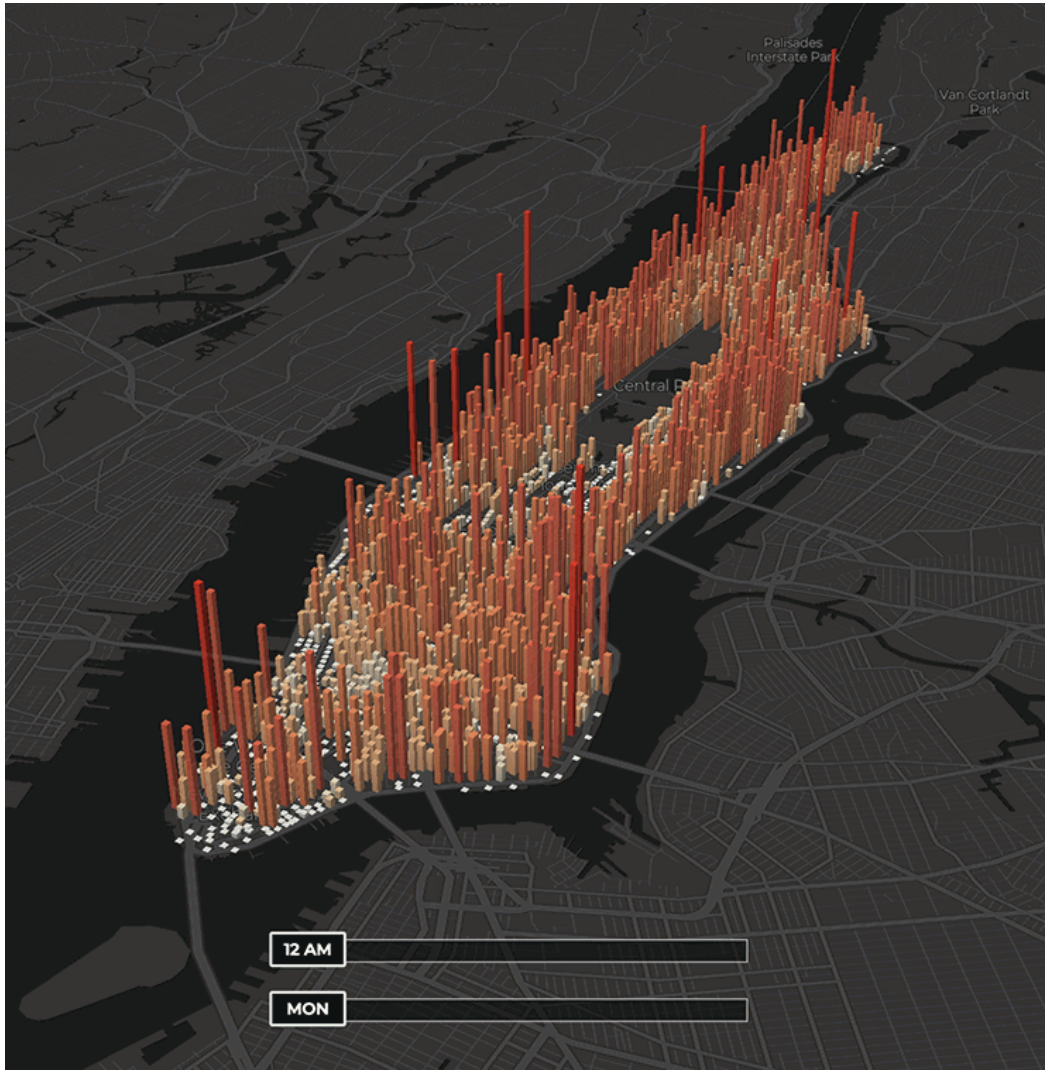




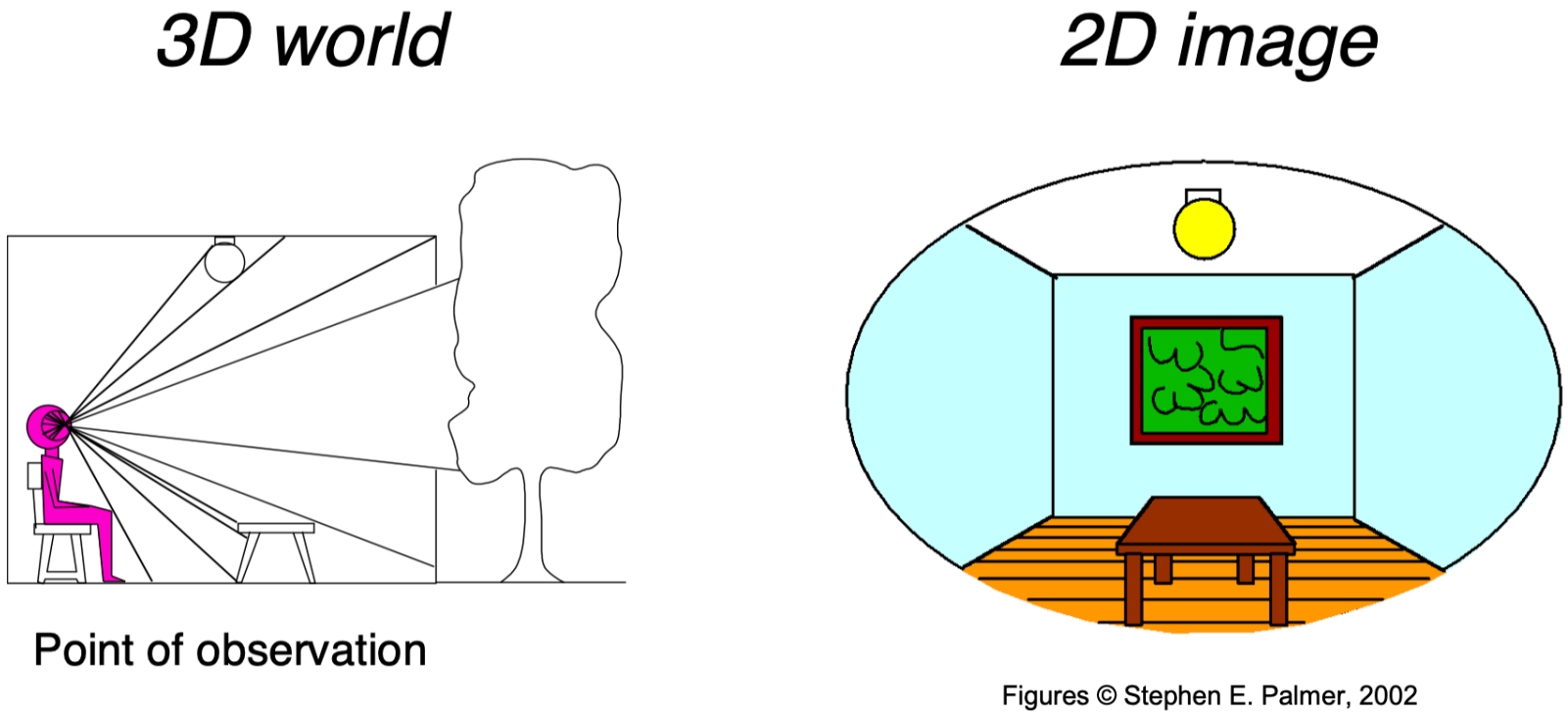
Transformation - Scaling



Transformation – Non-Rigid



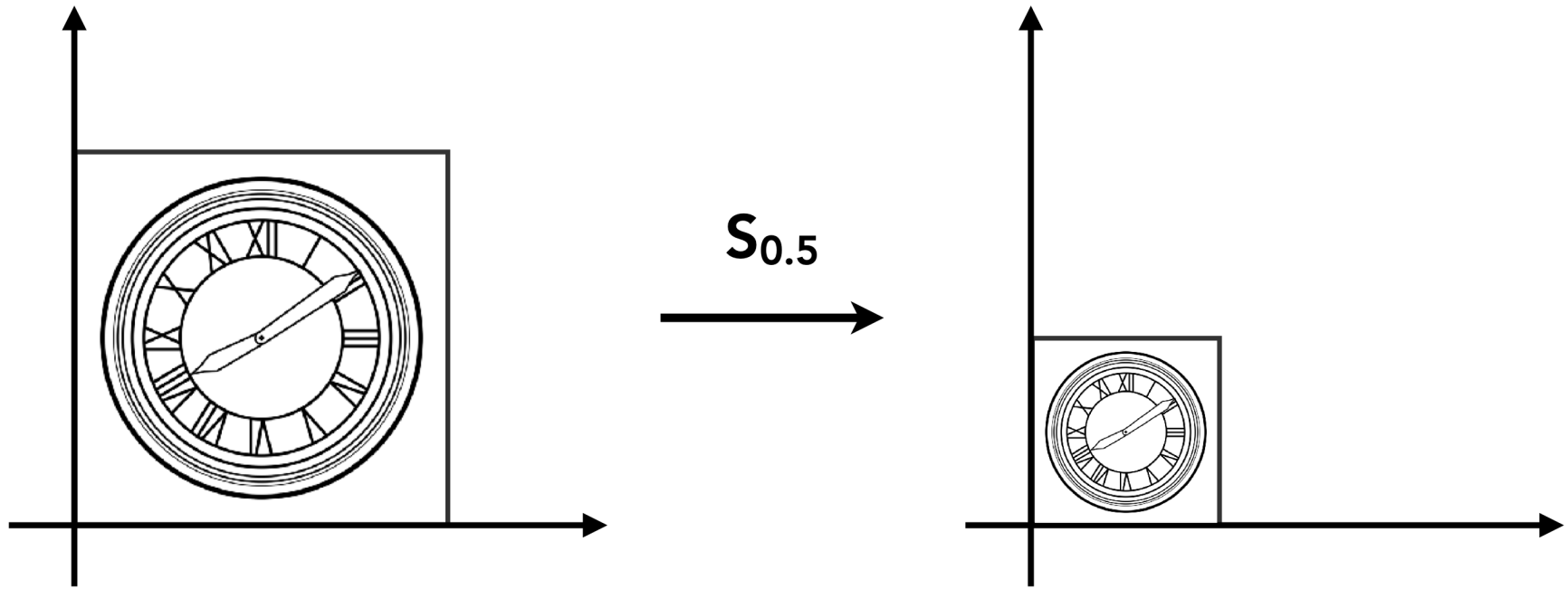
Transformation



Viewing: (3D to 2D) projection

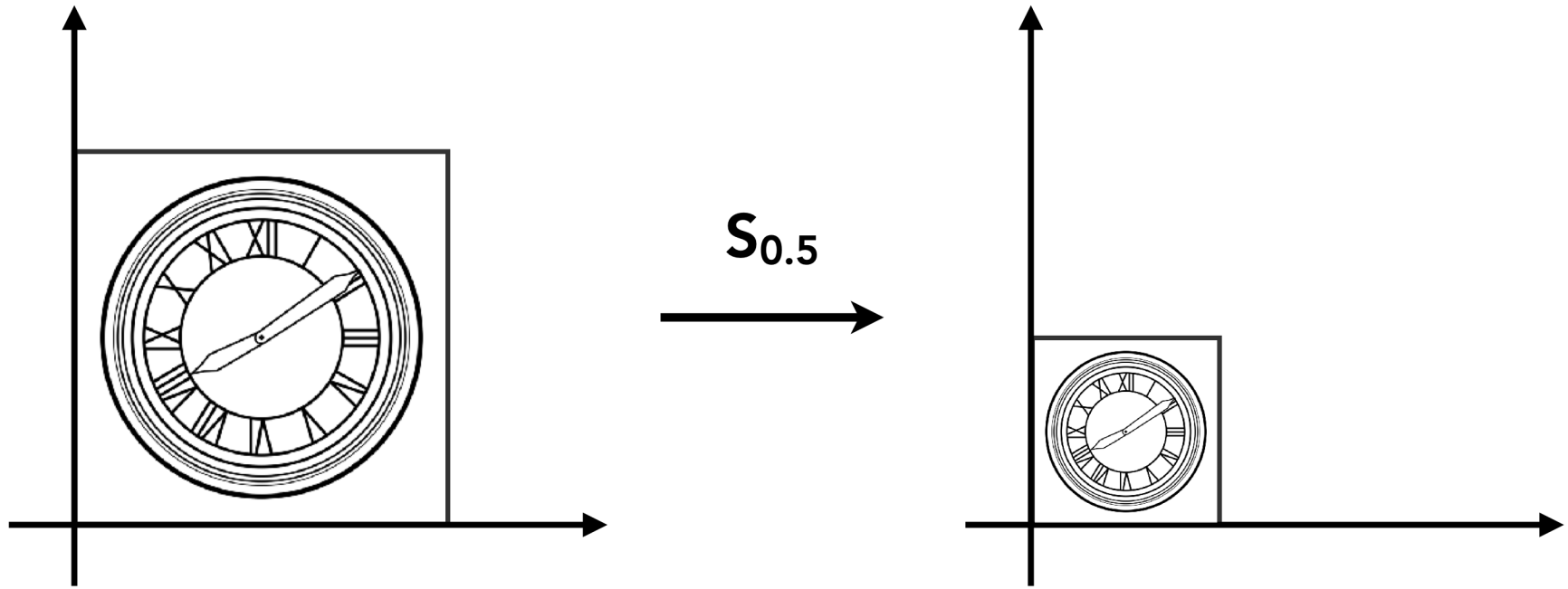
- 2D transformations
  - Representing transformations using matrices
  - Rotation, scale, shear
- Homogeneous coordinates

Scale



slide courtesy of Ren Ng

Scale Transform

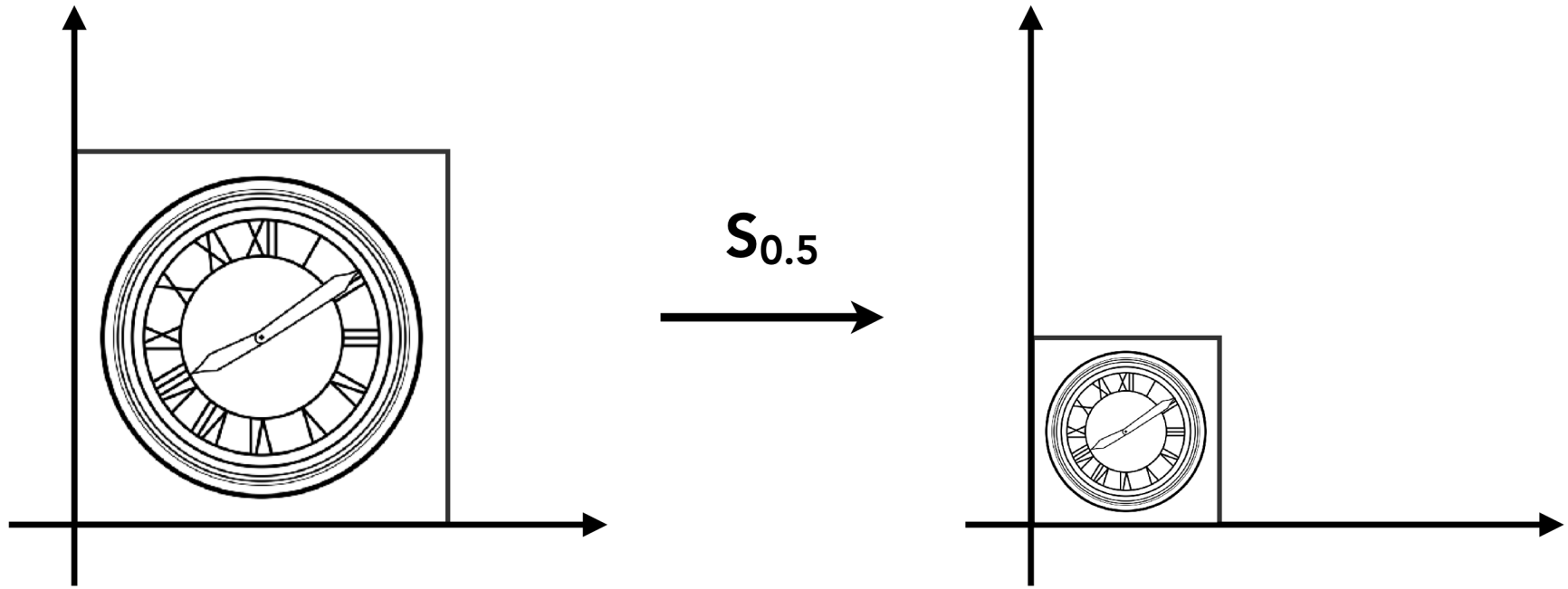


$$x' = sx$$

$$y' = sy$$

slide courtesy of Ren Ng

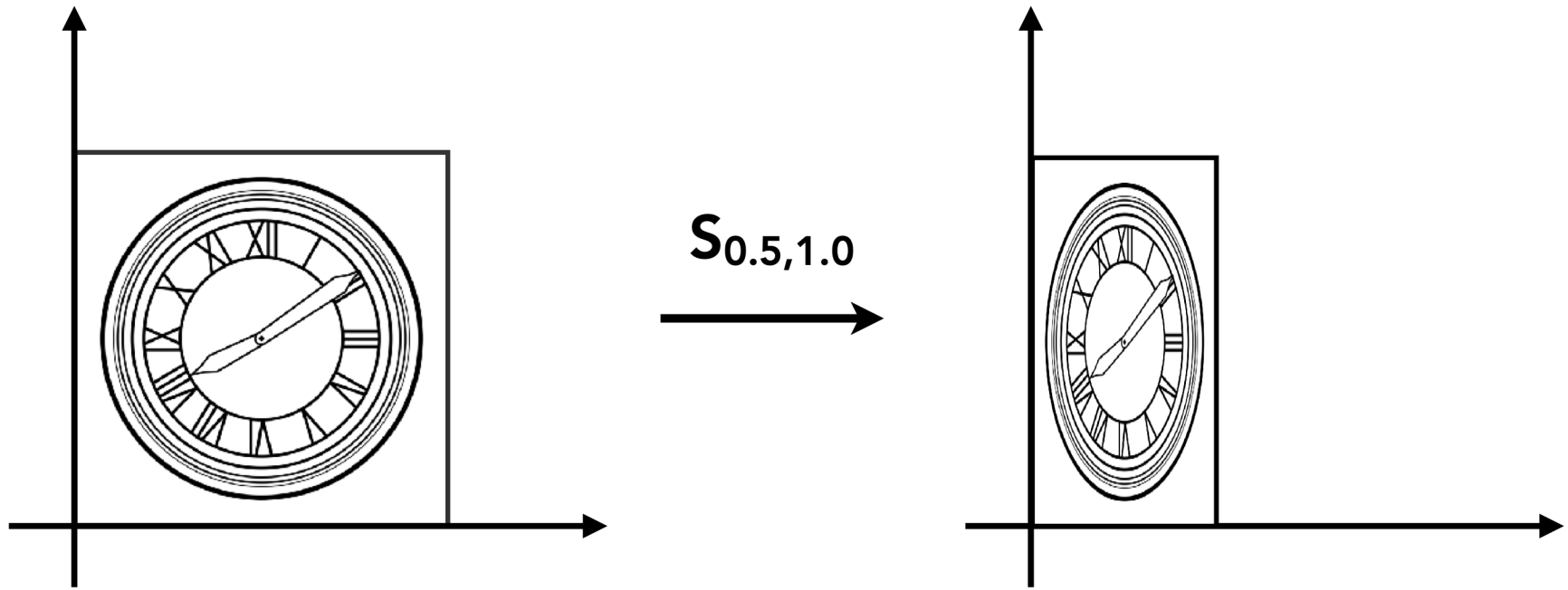
Scale Matrix



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

slide courtesy of Ren Ng

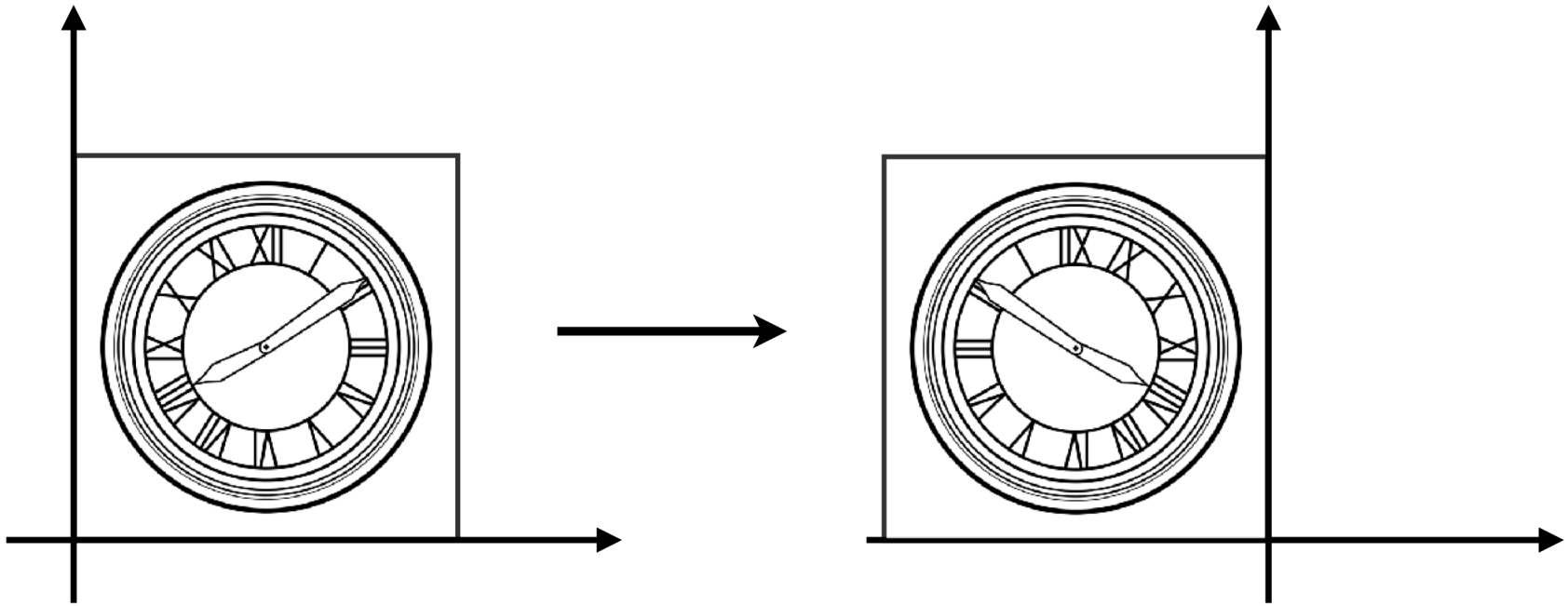
Scale (Non-Uniform)



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

slide courtesy of Ren Ng

Reflection Matrix



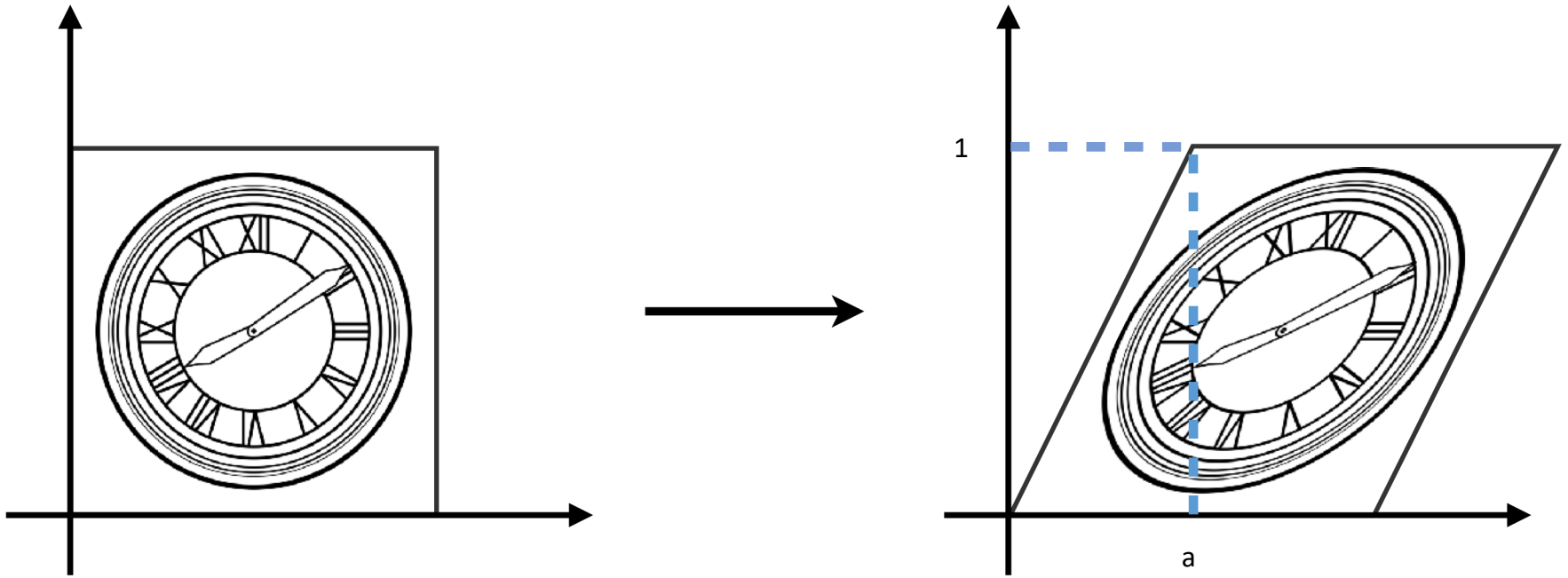
Horizontal reflection:

$$x' = -x$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

slide courtesy of Ren Ng

Shear Matrix



Hints:

Horizontal shift is 0 at y=0

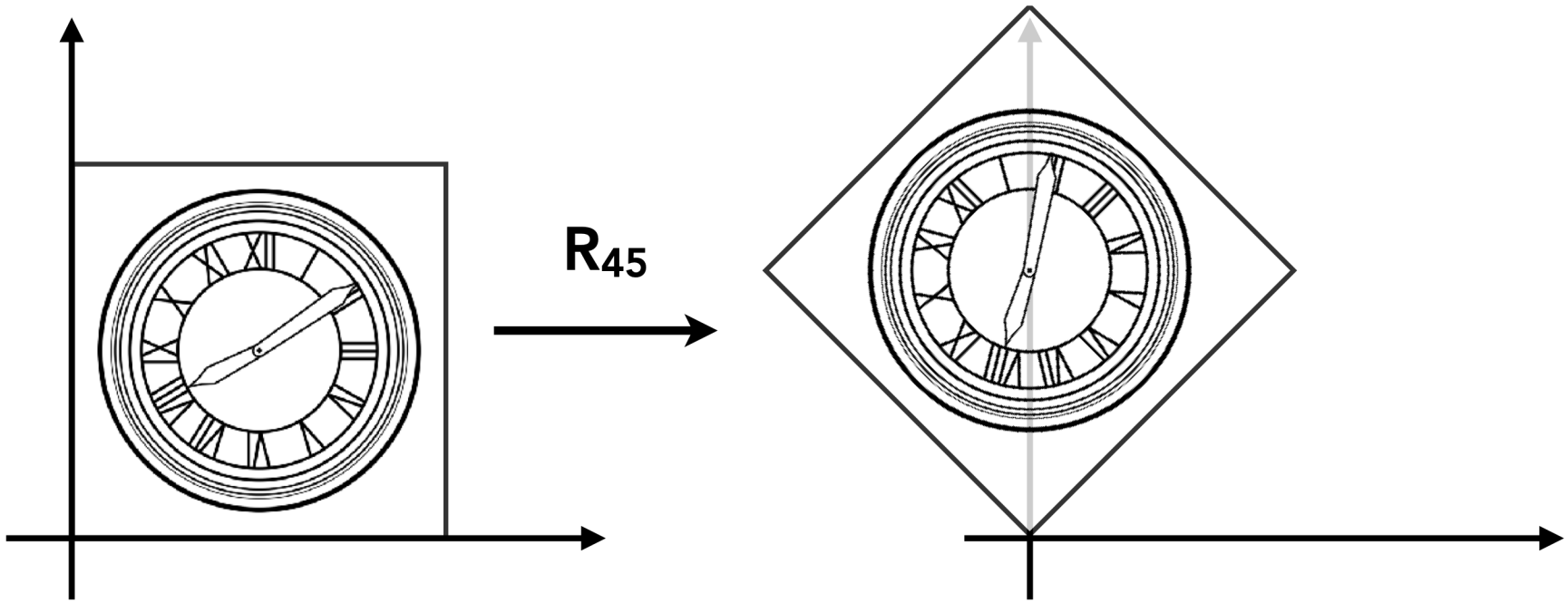
Horizontal shift is a at y=1

Vertical shift is always 0

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

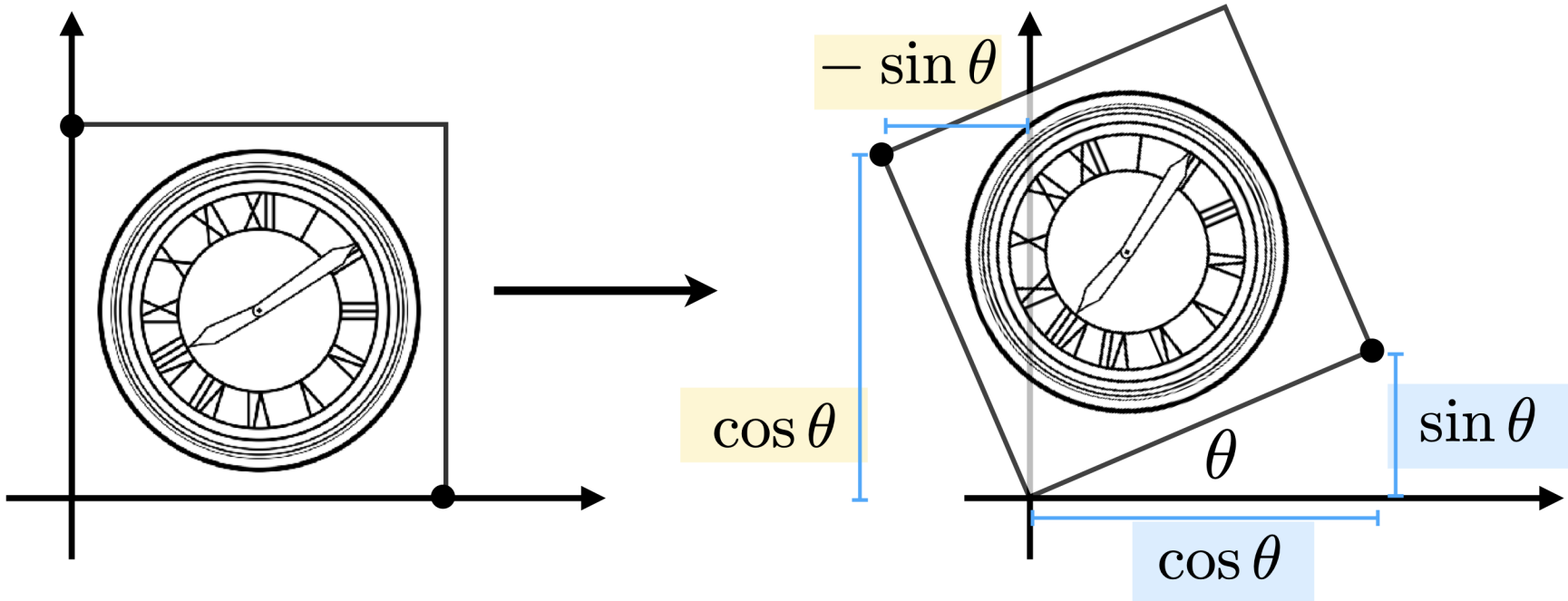
slide courtesy of Ren Ng

Rotate



slide courtesy of Ren Ng

Rotation Matrix



$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

slide courtesy of Ren Ng

Linear Transforms = Matrices

(of the same dimension)

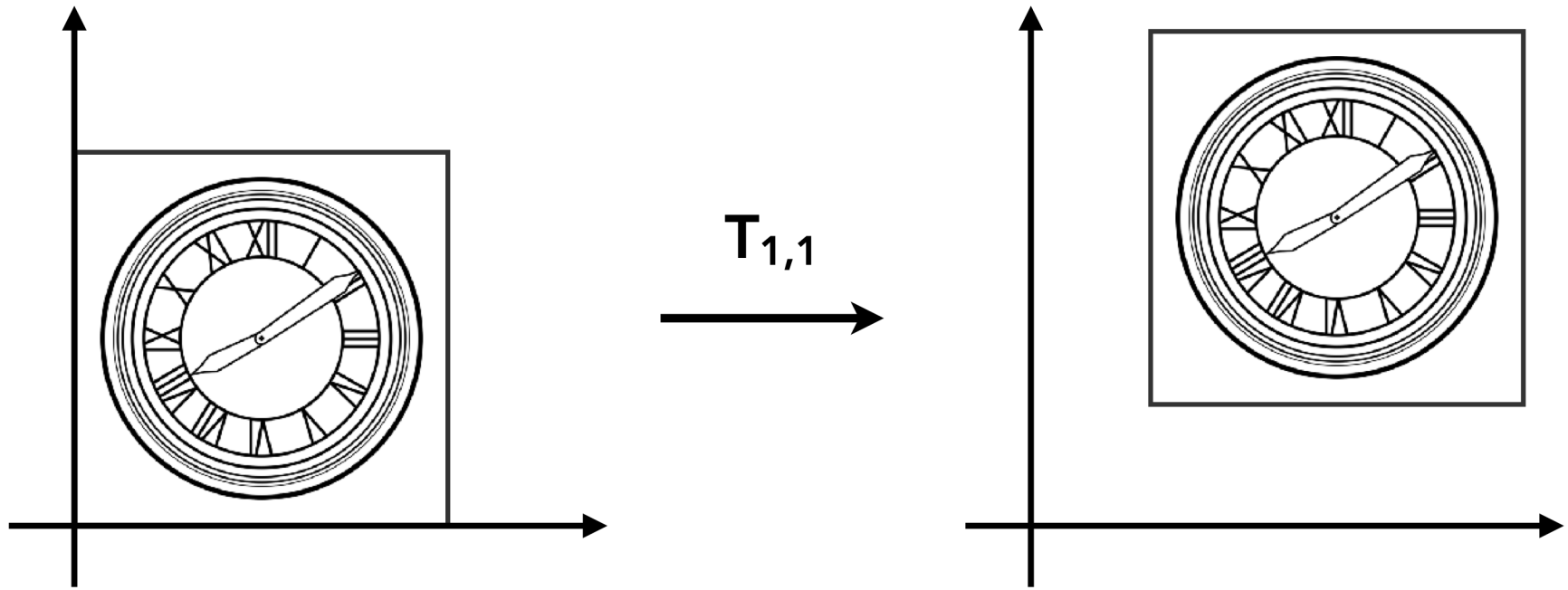
$$x' = a\,x + b\,y$$
$$y' = c\,x + d\,y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{x}' = \mathbf{M}\,\mathbf{x}$$

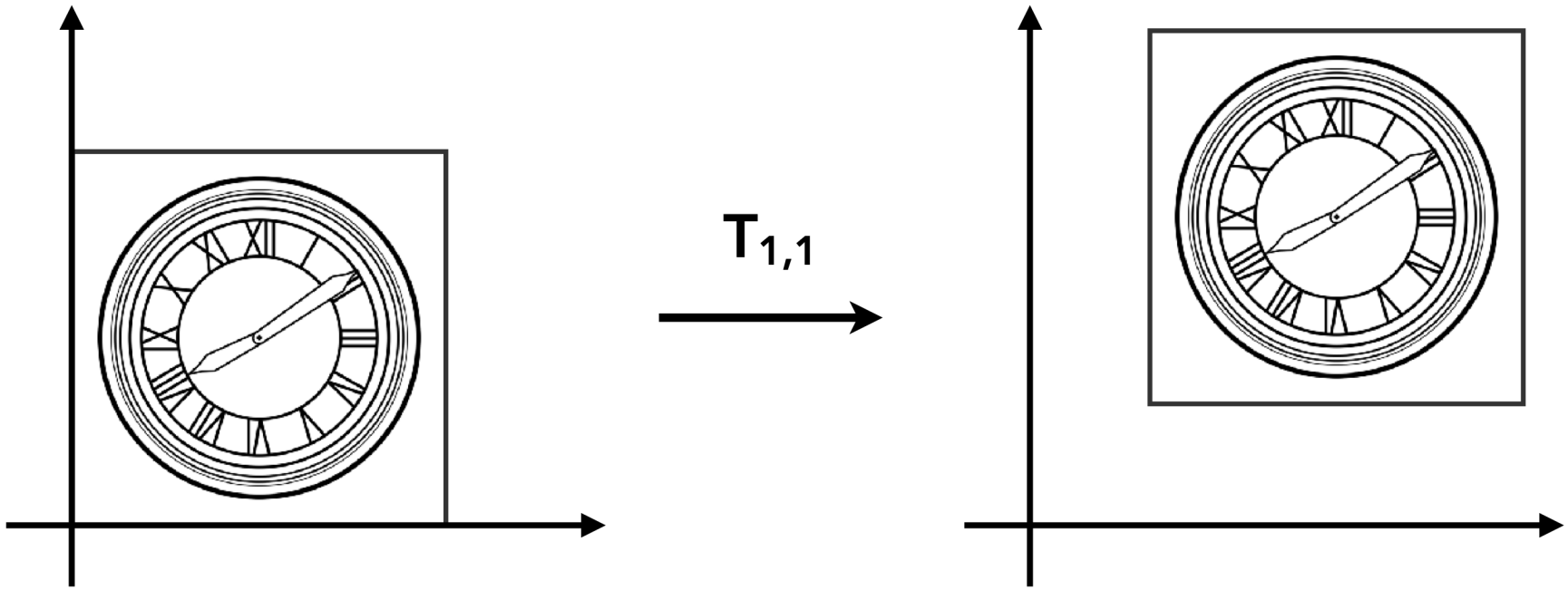
slide courtesy of Ren Ng

Translate



slide courtesy of Ren Ng

Translation??



$$x' = x + t_x$$
$$y' = y + t_y$$

slide courtesy of Ren Ng

Why Homogeneous Coordinates

- Translation cannot be represented in matrix form

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

(So, translation is NOT linear transform)

- But we don't want translation to be a special case
- Is there a unified way to represent all transformations?
  (and what's the cost?)

Solution: Homogenous Coordinates

Add a third coordinate (*w-coordinate*)

- **2D point = (x, y, 1)^T**
- **2D vector = (x, y, 0)^T**

Matrix representation of translations

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

slide courtesy of Ren Ng

Affine Transformations

Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Using homogenous coordinates:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

slide courtesy of Ren Ng

2D Transformations

Scale

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

Rotation

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

Translation

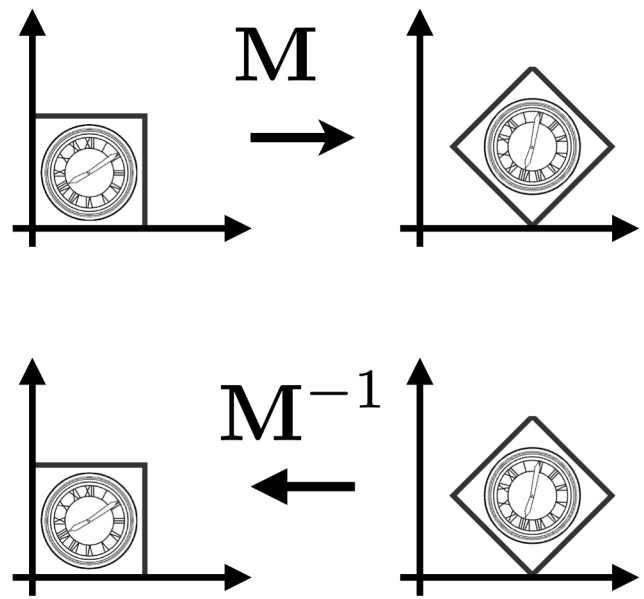
$$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

slide courtesy of Ren Ng

Inverse Transform

$$\mathbf{M}^{-1}$$

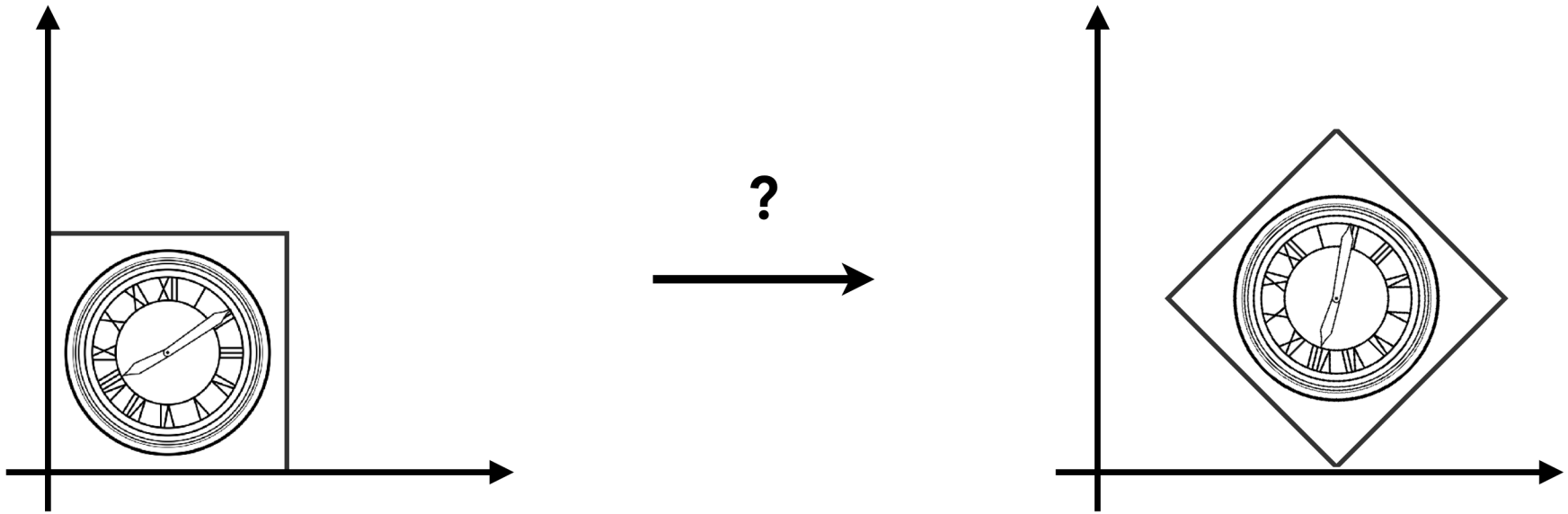
$\mathbf{M}^{-1}$ **is the inverse of transform** $\mathbf{M}$ **in both a matrix and geometric sense**



slide courtesy of Ren Ng

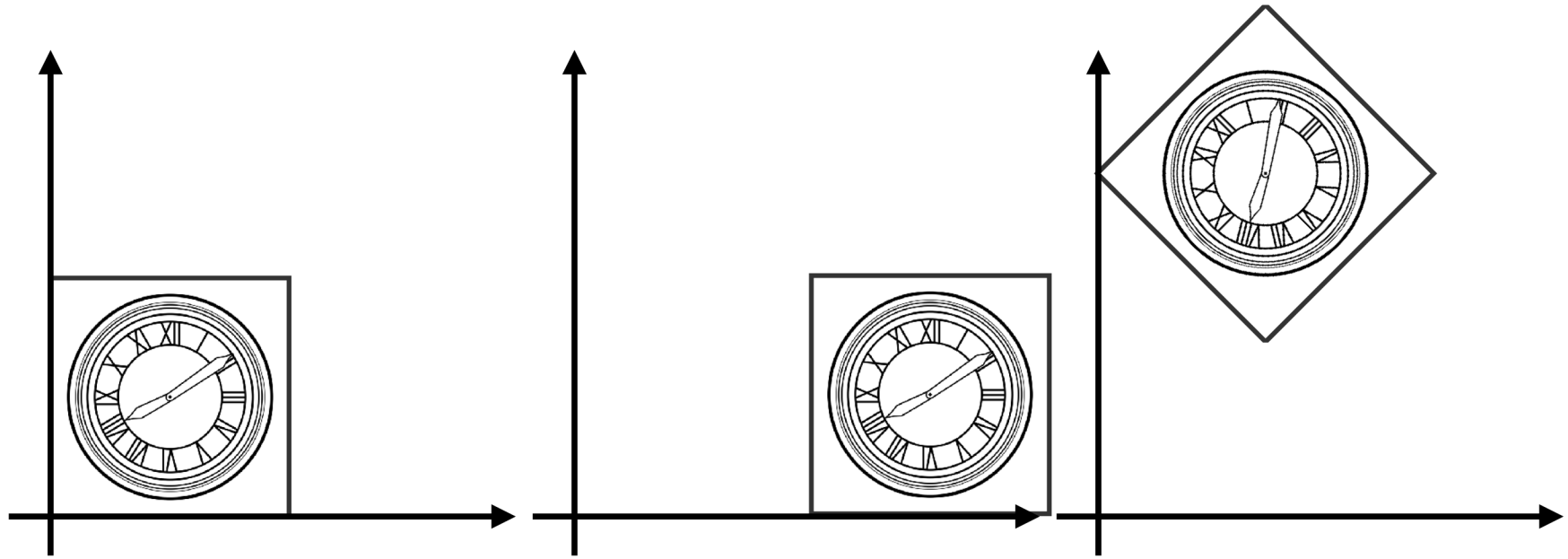
Composing Transforms

Composite Transform



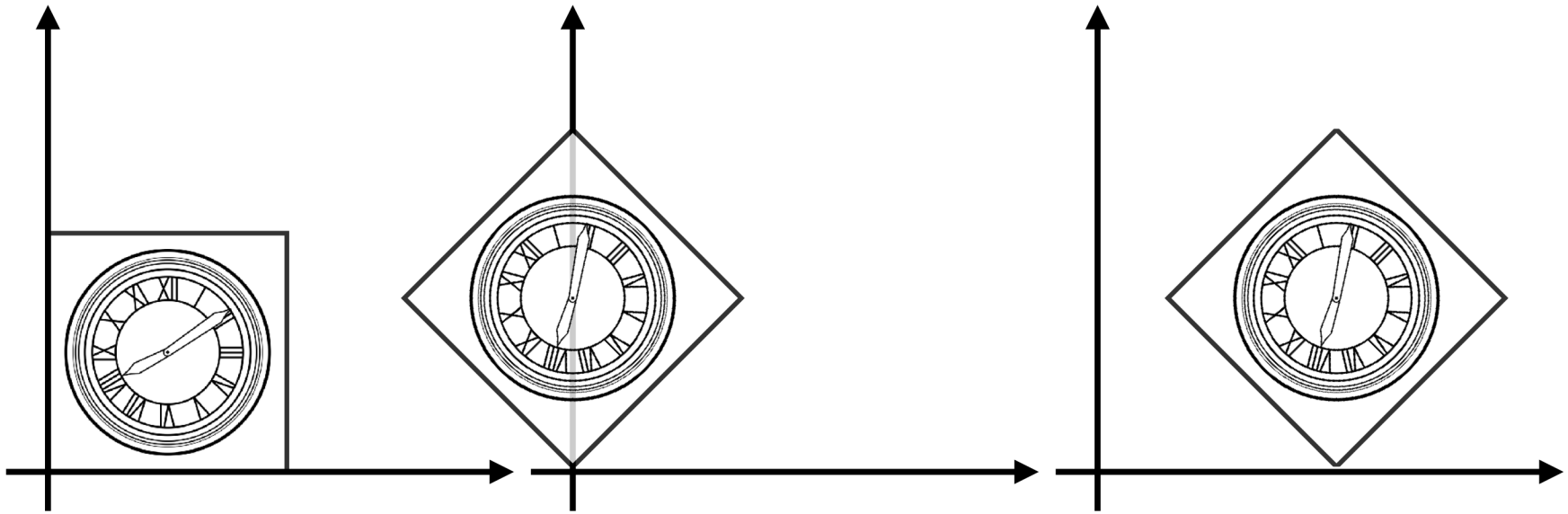
slide courtesy of Ren Ng

Translate Then Rotate?



slide courtesy of Ren Ng

Rotate Then Translate

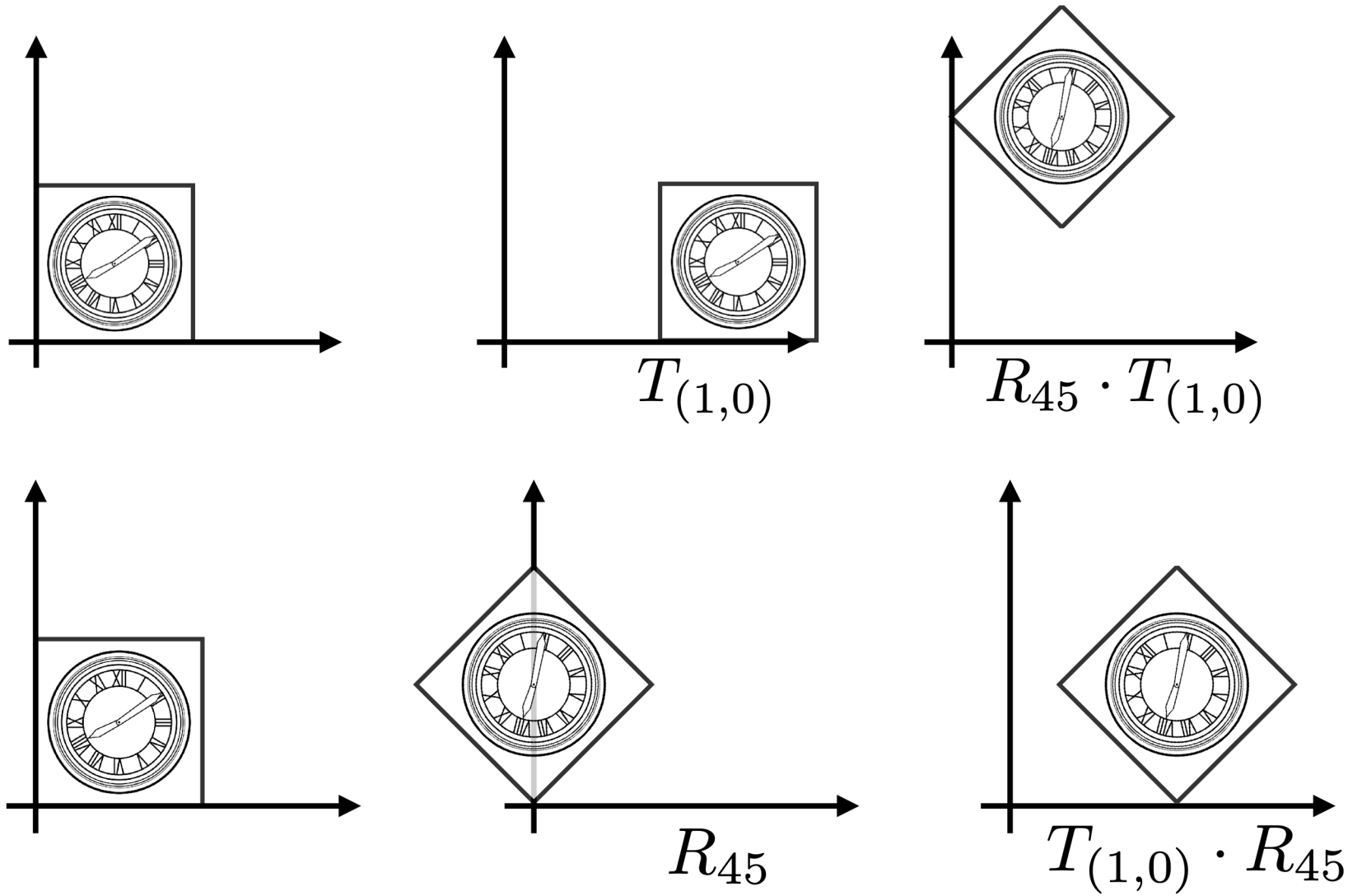


R_{45}

$T_{(1,0)} \cdot R_{45}$

slide courtesy of Ren Ng

Transform Ordering Matters!



slide courtesy of Ren Ng

Transform Ordering Matters!

Matrix multiplication is <u>not</u> commutative

$$R_{45} \cdot T_{(1,0)} \neq T_{(1,0)} \cdot R_{45}$$

Recall the matrix math represented by these symbols:

$$\begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Note that matrices are applied right to left:

$$T_{(1,0)} \cdot R_{45} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

slide courtesy of Ren Ng

Composing Transforms

Sequence of affine transforms A_1, A_2, A_3, ...

- **Compose by matrix multiplication**
  - **Very important for performance!**

$$A_n(\ldots A_2(A_1(\mathbf{x}))) = \underbrace{\mathbf{A}_n \cdots \mathbf{A}_2 \cdot \mathbf{A}_1}_{} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

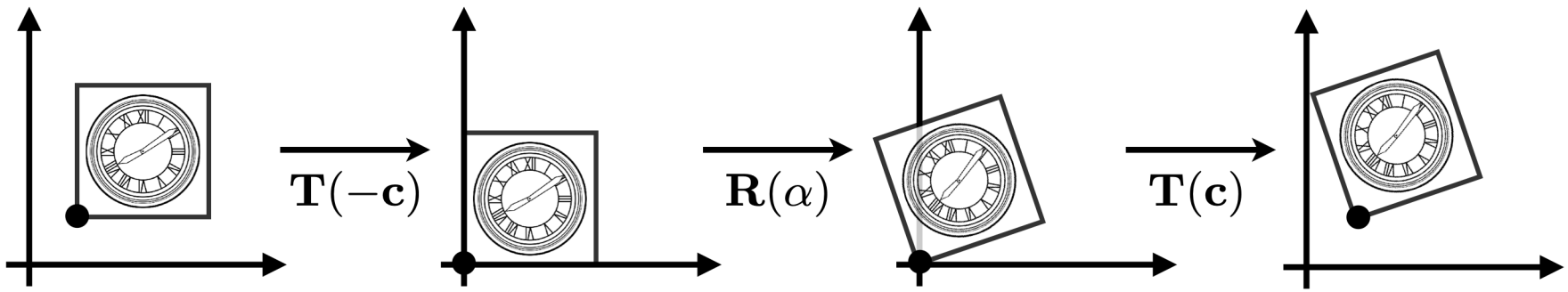
Pre-multiply *n* matrices to obtain a single matrix representing combined transform

slide courtesy of Ren Ng

Decomposing Complex Transforms

How to rotate around a given point c?

1. Translate center to origin
2. Rotate
3. Translate back



Matrix representation?

$$\mathbf{T}(\mathbf{c}) \cdot \mathbf{R}(\alpha) \cdot \mathbf{T}(-\mathbf{c})$$

slide courtesy of Ren Ng

Where We Are

- Basic mathematical tools

- Color: definition, space, map, and design
- 2D visualization: data -> spatial, temporal, network -> interaction in 2D

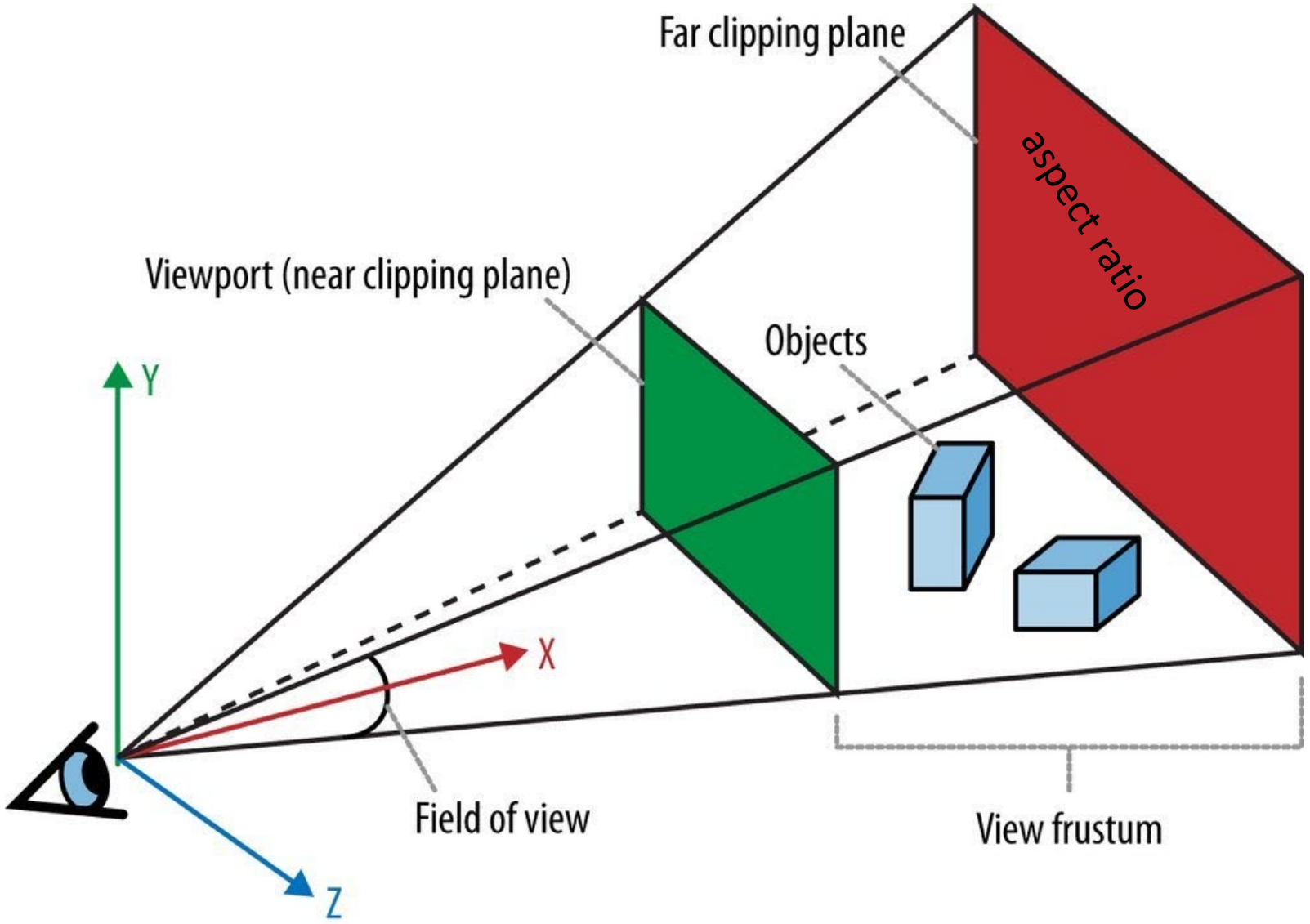
- Interaction in 3D
- Scientific and 3D visualization

- Application case studies

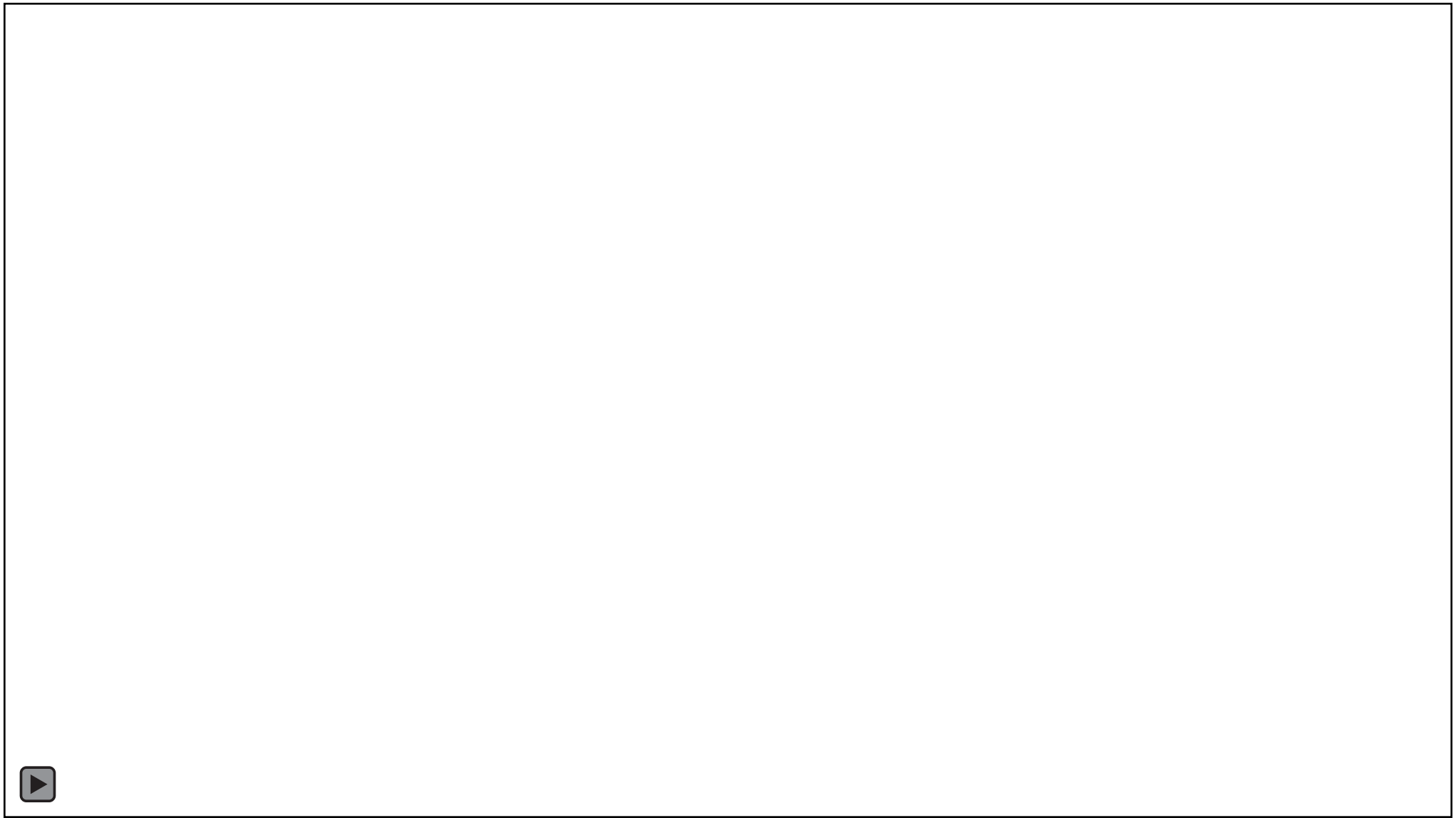
PART 02

Interaction: (3D) Transformation

“Camera” in 3D Representation



Why 3D Transformation?



3D Transformations

Use homogeneous coordinates again:

- 3D point = (x, y, z, 1)^T
- 3D vector = (x, y, z, 0)^T

Use 4×4 matrices for affine transformations

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

slide courtesy of Ren Ng

3D Transformations

Scale

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

Translation

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

Coordinate Change (Frame-to-world)

$$\mathbf{F}(\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{o}) = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{o} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

slide courtesy of Ren Ng

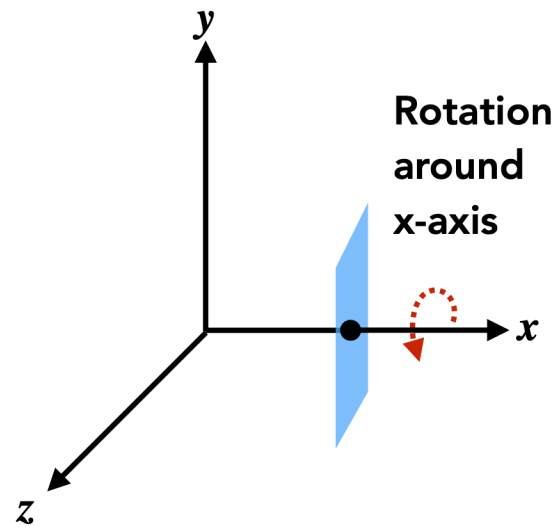
3D Transformations

Rotation around x-, y-, or z-axis

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

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

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



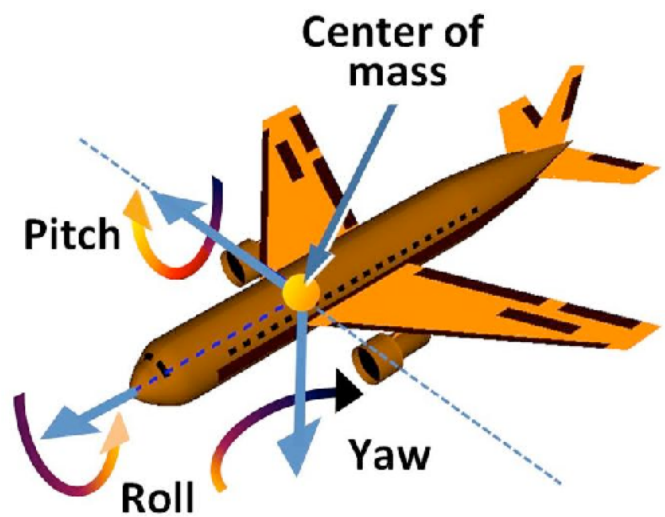
slide courtesy of Ren Ng

3D Rotations

Compose any 3D rotation from $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$?

$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\gamma)$$

- So-called *Euler angles*
- Often used in flight simulators: roll, pitch, yaw



slide courtesy of Ren Ng

3D Rotations

Compose any 3D rotation from $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$?

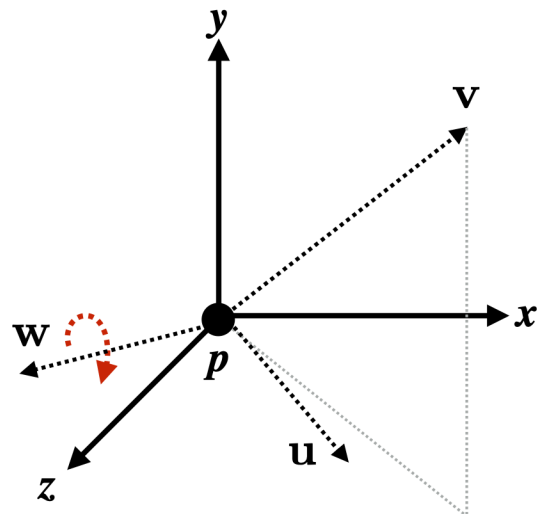
$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\gamma)$$

- So-called *Euler angles*
- Often used in flight simulators: roll, pitch, yaw

slide courtesy of Ren Ng

3D Rotation Around Arbitrary Axis

Construct orthonormal frame transformation F with p, u, v, w, where p and w match the rotation axis



Apply the transform ($F\ R_z(\theta)\ F^{-1}$)

Interpretations (both valid):

- Move to Z axis, rotate, then move back
- Cast w-axis rotation in new coordinate frame

slide courtesy of Ren Ng

Rodrigues' Rotation Formula

Rotation by angle α around axis $\mathbf{n}$

$$\mathbf{R}(\mathbf{n},\alpha) \;=\; \cos(\alpha)\,\mathbf{I} \;+\; (1-\cos(\alpha))\,\mathbf{n}\mathbf{n}^T \;+\; \sin(\alpha)\underbrace{\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}}_{\mathbf{N}}$$

How to prove this magic formula?

- Matrix $\mathbf{N}$ computes a cross-product: $\mathbf{N}\,\boldsymbol{x} = \mathbf{n} \times \boldsymbol{x}$
- Assume orthonormal system $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{n}$

$$\begin{aligned} \mathbf{R}\mathbf{n} &= \mathbf{n} \\ \mathbf{R}\mathbf{e}_1 &= \cos\alpha\,\mathbf{e}_1 + \sin\alpha\,\mathbf{e}_2 \\ \mathbf{R}\mathbf{e}_2 &= -\sin\alpha\,\mathbf{e}_1 + \cos\alpha\,\mathbf{e}_2 \end{aligned}$$

slide courtesy of Ren Ng

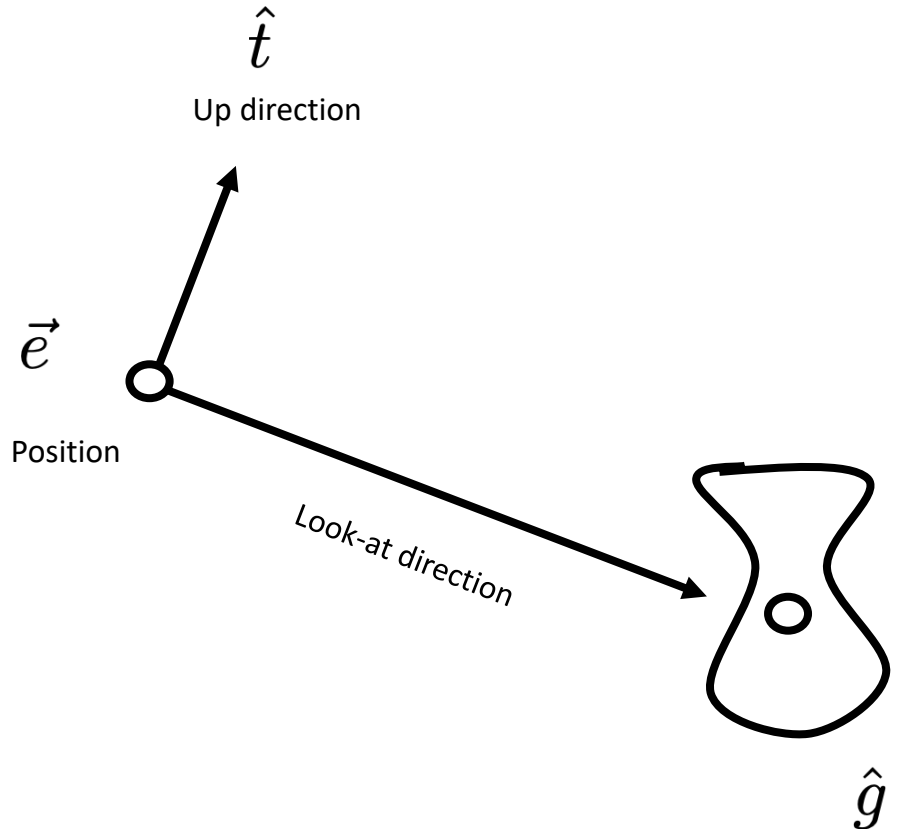
3D Transforms

Model-View-Projection

- While taking a selfie
  - Find a good place to stand (**model** transformation)
  - Find a good "angle" to place the camera (**view** transformation)
  - Cheeese! (**projection** transformation)

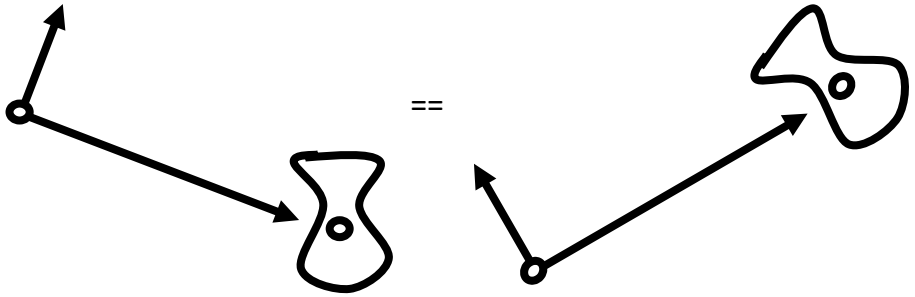
View / Camera Transformation

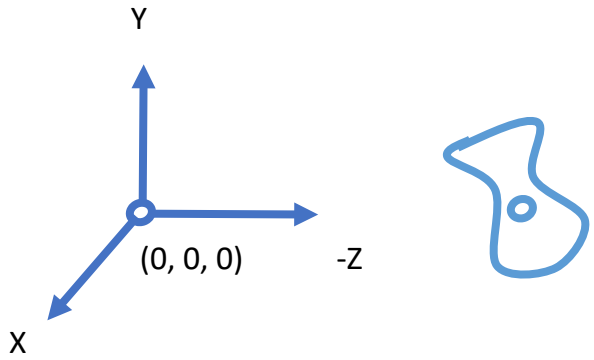
- How to perform view transformation?
- Define the camera first
  - Position
  - Look-at / gaze direction
  - Up direction (assuming perp. to look-at)



View / Camera Transformation

- Key observation
  - If the camera and objects move together, the “photo” will be the same



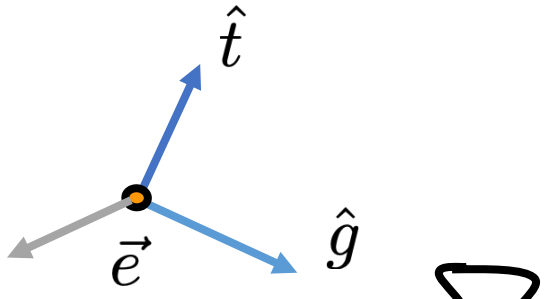


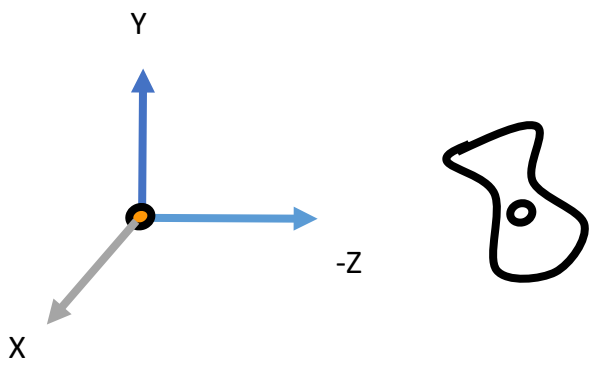
- How about that we move the camera to
  - The origin, up at Y, look at -z
  - Move the objects along with the camera

View / Camera Transformation

- Transform the camera by M_{view}
  - So that it's located at the origin, up at Y, look at -z
  - Transform the objects along with the camera

- M_{view} in math?
  - Translates e to origin
  - Rotates g to -z
  - Rotates t to Y
  - Rotates (g x t) To X





View / Camera Transformation

- M_{view} in math?
  - Let's write $M_{view} = R_{view}T_{view}$
  - Translate e to origin

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotate g to -Z, t to Y, (g x t) To X
  - Consider its inverse rotation: X to (g x t), Y to t, Z to -g

$$R_{view}^{-1} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\hat{g}\times\hat{t}} & y_t & y_{-g} & 0 \\ z_{\hat{g}\times\hat{t}} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

WHY?

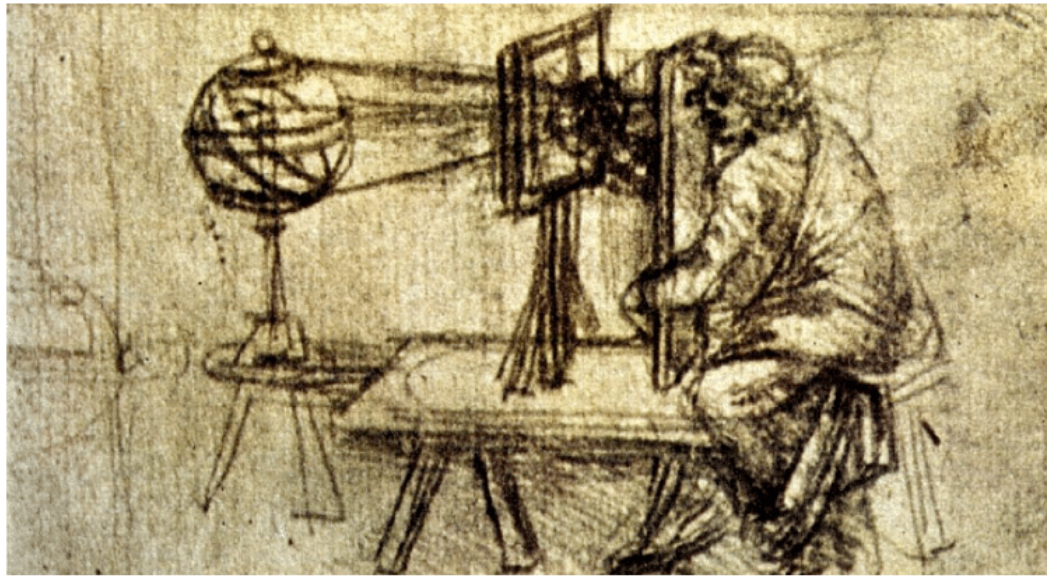
$$\Rightarrow \quad R_{view} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & y_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

View / Camera Transformation

- Summary
  - Transform objects together with the camera
  - Until camera's at the origin, up at Y, look at -Z
- But why?
  - For projection transformation!

3D Transforms

Model-View-Projection

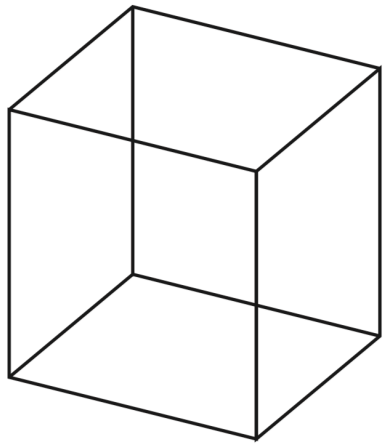


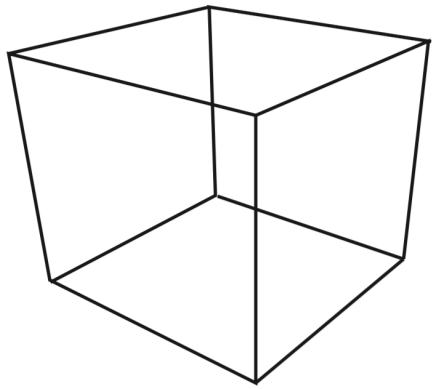
Leonardo da Vinci, 1515

Projection Transformation

- Projection in 3D Visualization
  - 3D to 2D
  - Orthographic projection
  - Perspective projection

Orthographic projection



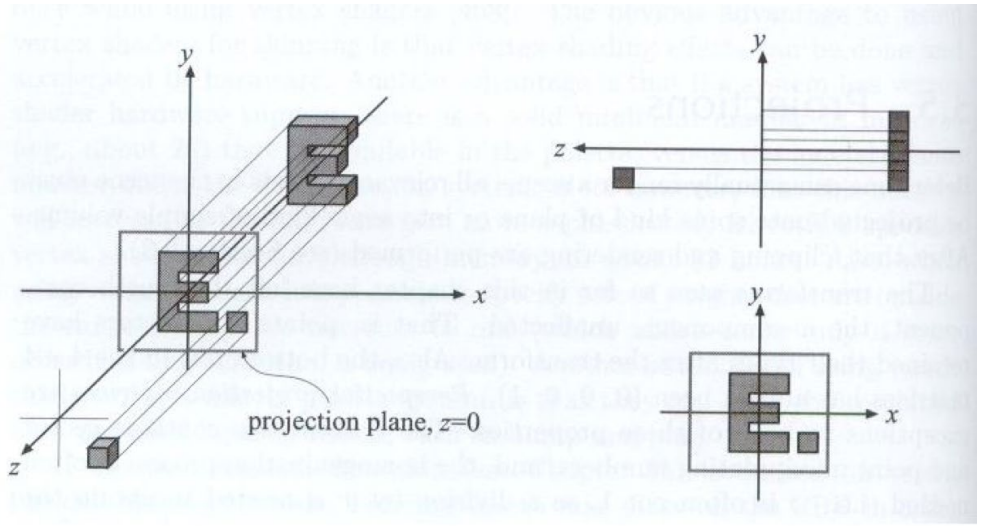


Perspective projection

Fig. 7.1 from *Fundamentals of Computer Graphics, 4th Edition*

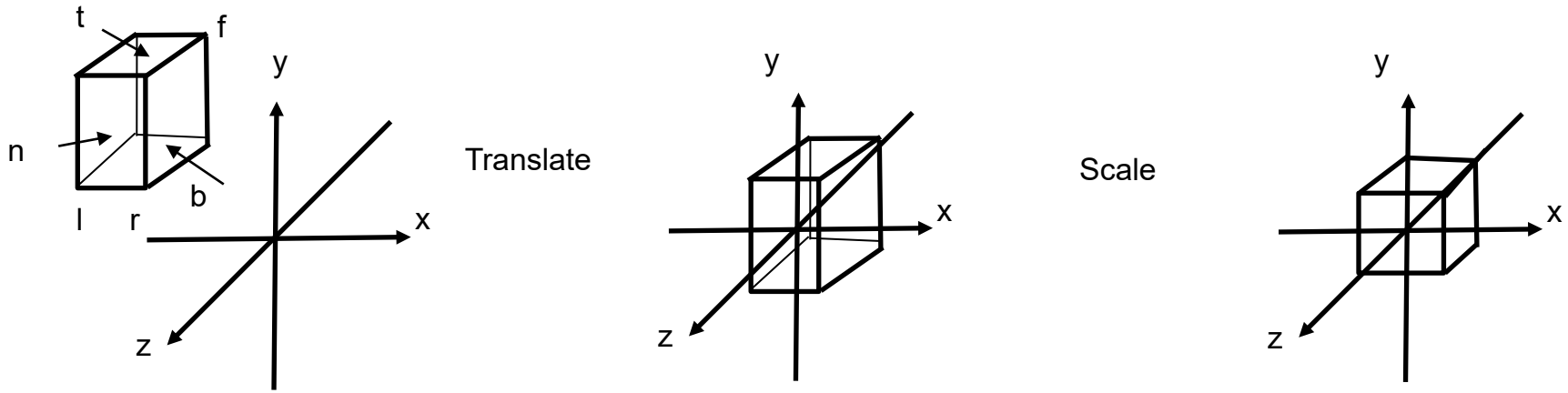
Orthographic Projection

- A simple way of understanding
  - Camera located at origin, looking at -Z, up at Y (looks familiar?)
  - Drop Z coordinate
  - Translate and scale the resulting rectangle to "canonical" view within $[-1, 1]^2$



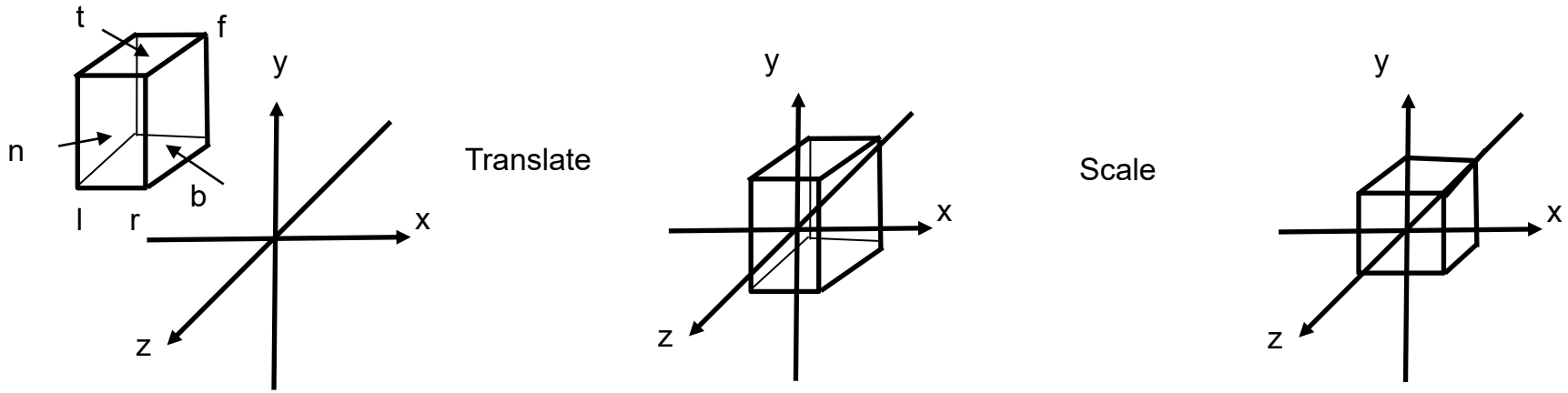
Orthographic Projection

- In general
  - We want to map a cuboid [l, r] x [b, t] x [**f, n**] to the “canonical” cube $[-1, 1]^3$



Orthographic Projection

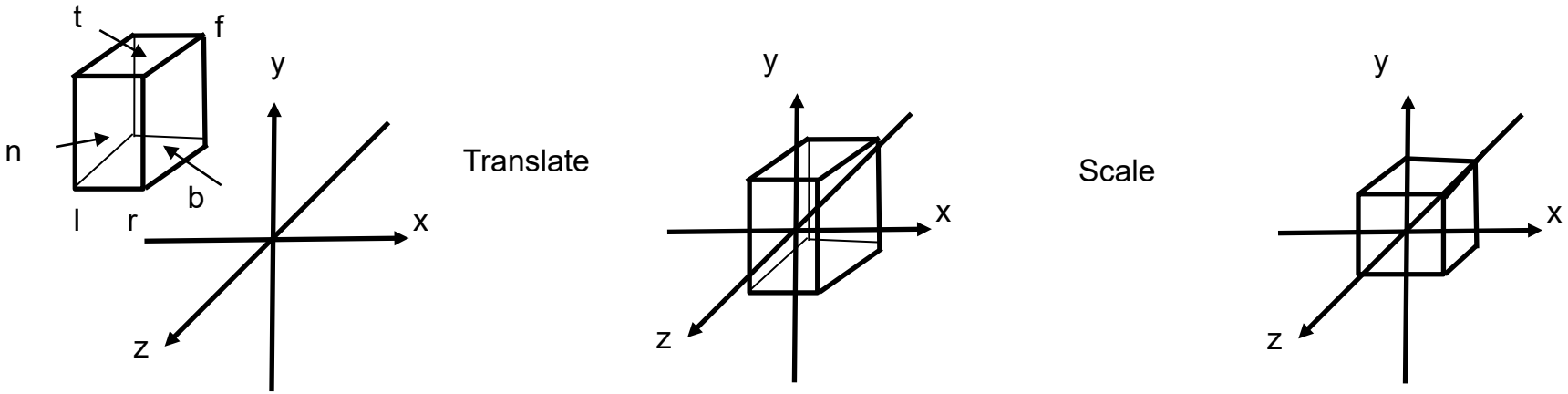
- First center cuboid by translating
- Then scale into “canonical” cube



Orthographic Projection

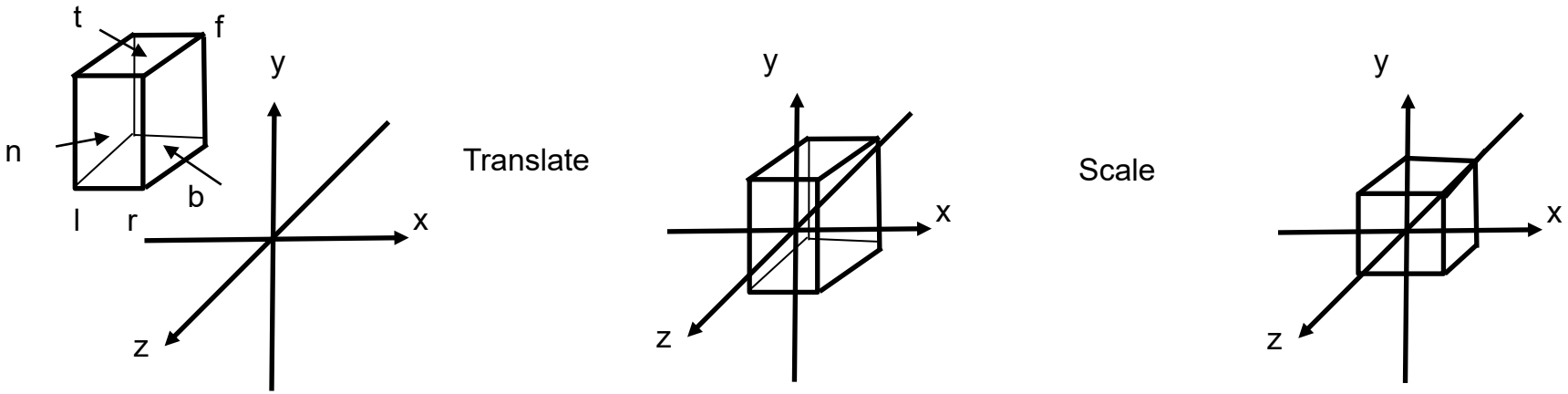
- Transformation matrix?
  - Translate (**center** to origin) first, then scale (length/width/height to **2**)

$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



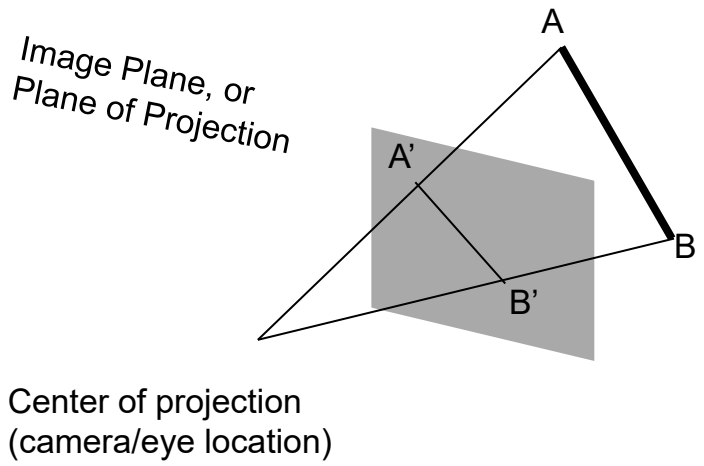
Orthographic Projection

- Caveat
  - Looking at / along -Z is making near and far not intuitive (n > f)
  - FYI: that's why OpenGL (a Graphics API) uses left hand coords.



Perspective Projection

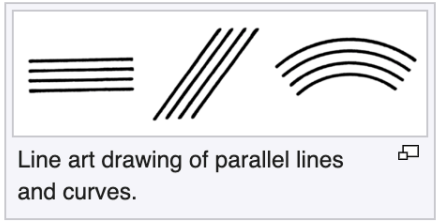
- Most common computer graphics, art, visual system
- Further objects are smaller
- Parallel lines not parallel; converge to single point



Perspective Projection

- Euclid was wrong??!!

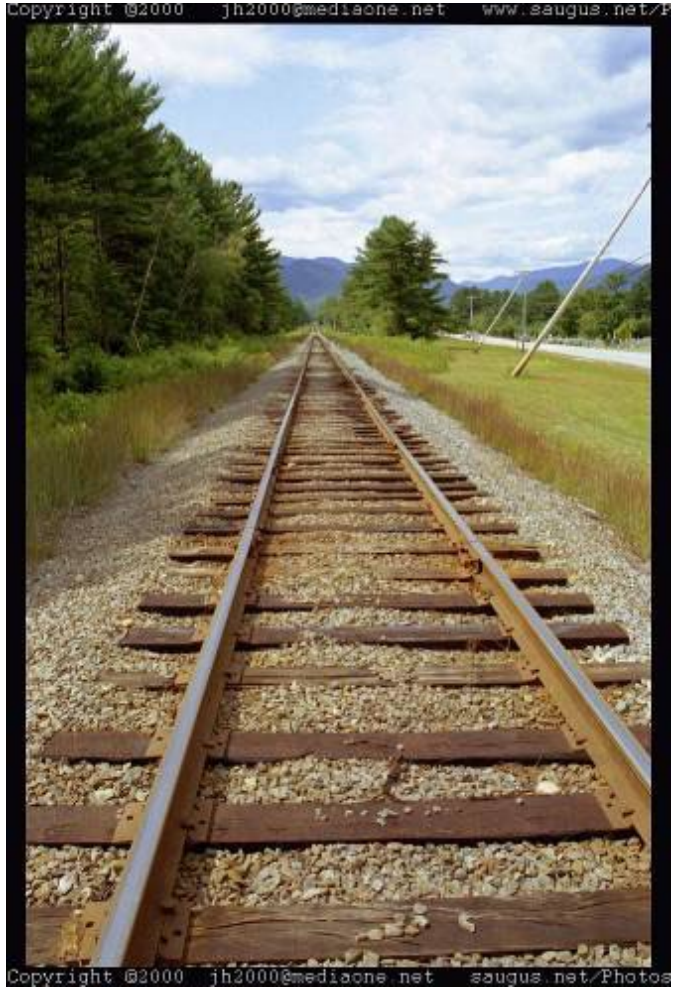
In geometry, **parallel** lines are lines in a plane which do not meet; that is, two lines in a plane that do not intersect or touch each other at any point are said to be parallel. By extension, a line and a plane, or two planes, in three-dimensional Euclidean space that do not share a point are said to be parallel. However, two lines in three-dimensional space which do not meet must be in a common plane to be considered parallel; otherwise they are called skew lines. Parallel planes are planes in the same three-dimensional space that never meet.



Line art drawing of parallel lines and curves.

Parallel lines are the subject of Euclid's parallel postulate.[1] Parallelism is primarily a property of affine geometries and Euclidean geometry is a special instance of this type of geometry. In some other geometries, such as hyperbolic geometry, lines can have analogous properties that are referred to as parallelism.

https://en.wikipedia.org/wiki/Parallel_(geometry)



Final Project Q&A

- Clarification/concerns on the proposal: we will reach out to individuals if any. Otherwise, go ahead with your plan

- “How many figures/buttons do I need?”
  - Algorithm Projects (i.e., reproducing research articles): whatever the paper does to solve a problem
  - Applications Projects: as an “application”/system, they are **commonly** interactive. Also think about using all the techniques we covered. Consider the interaction to bridge spatial/temporal/network data, etc.
  - Capstone/Research projects: dependss

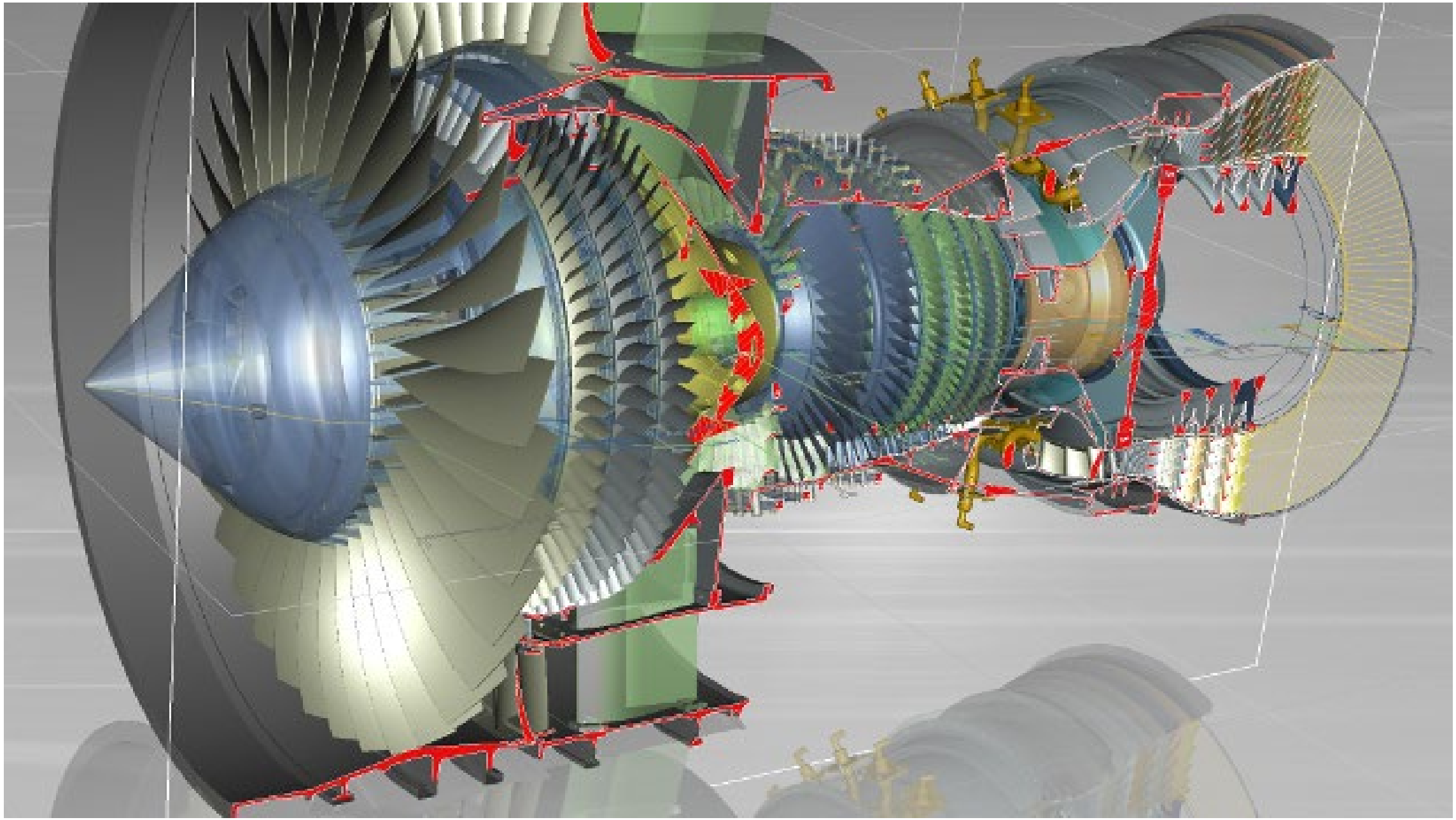
3D Visualization

- 3D Data (Scientific Visualization)
- Non-3D Data

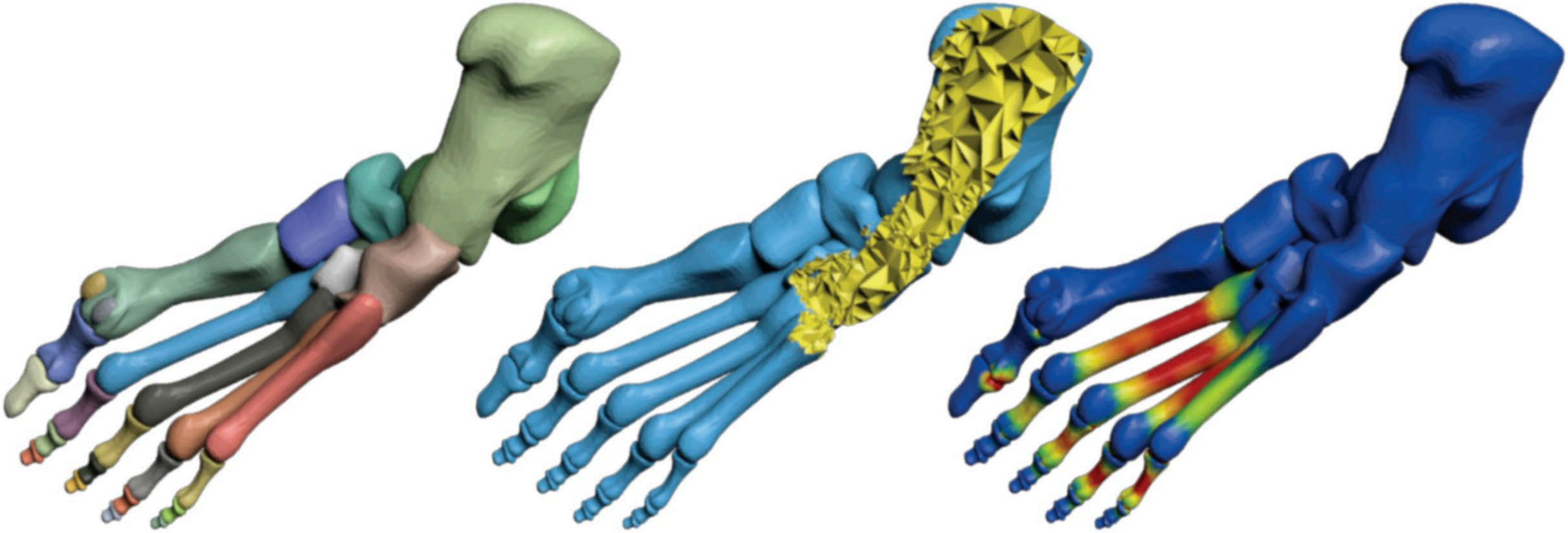
3D Visualization for 3D Data

3D Data

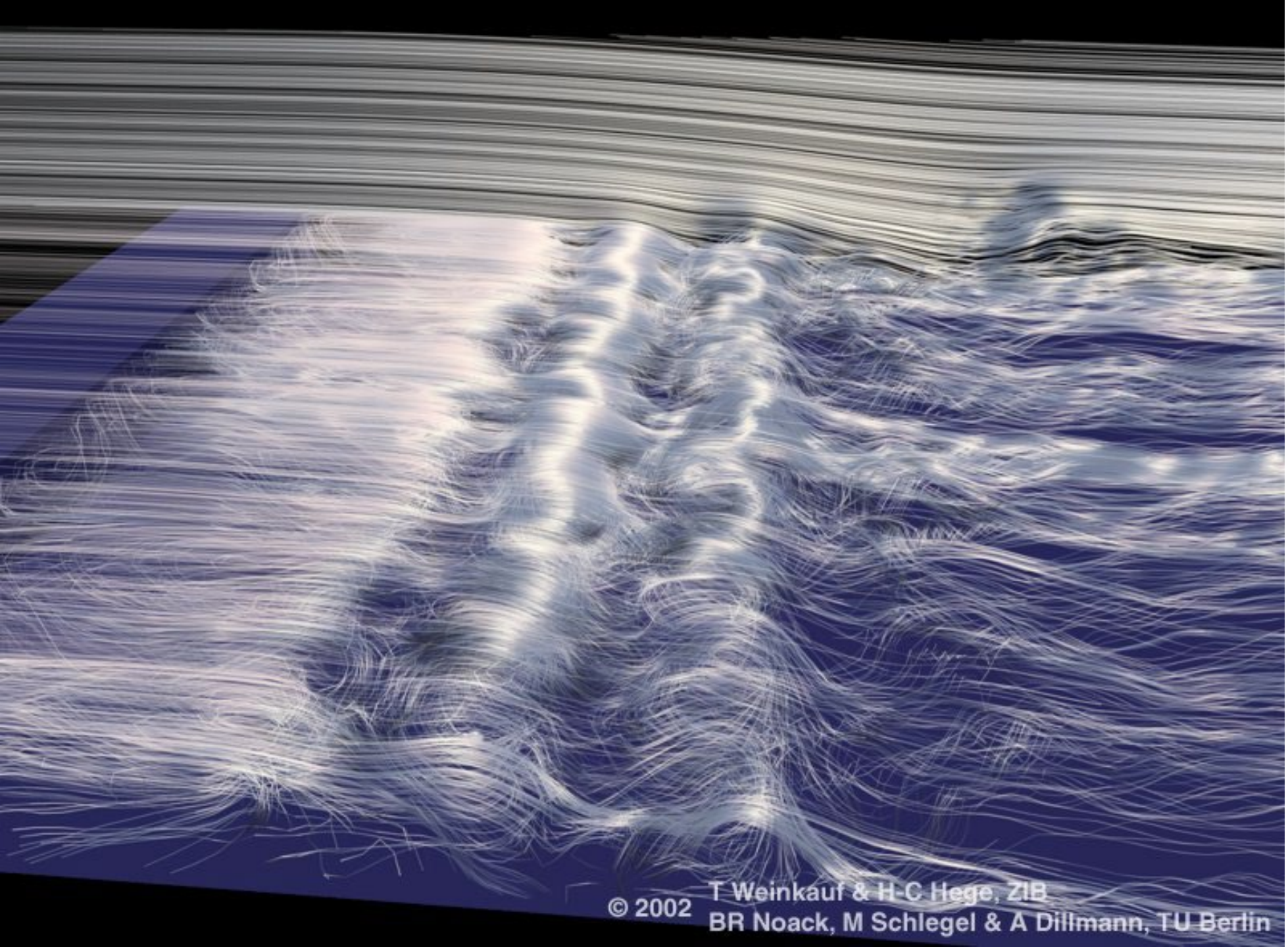
- Data that are intrinsically 3D-related.
  - Every observation is associated with position (x, y, z).
- Two sub-categories:
  - 3D geometry is the data.
  - Data is associated with 3D geometry.



Source: https://www.plm.automation.siemens.com/global/en/products/collaboration/digital-mockup.html

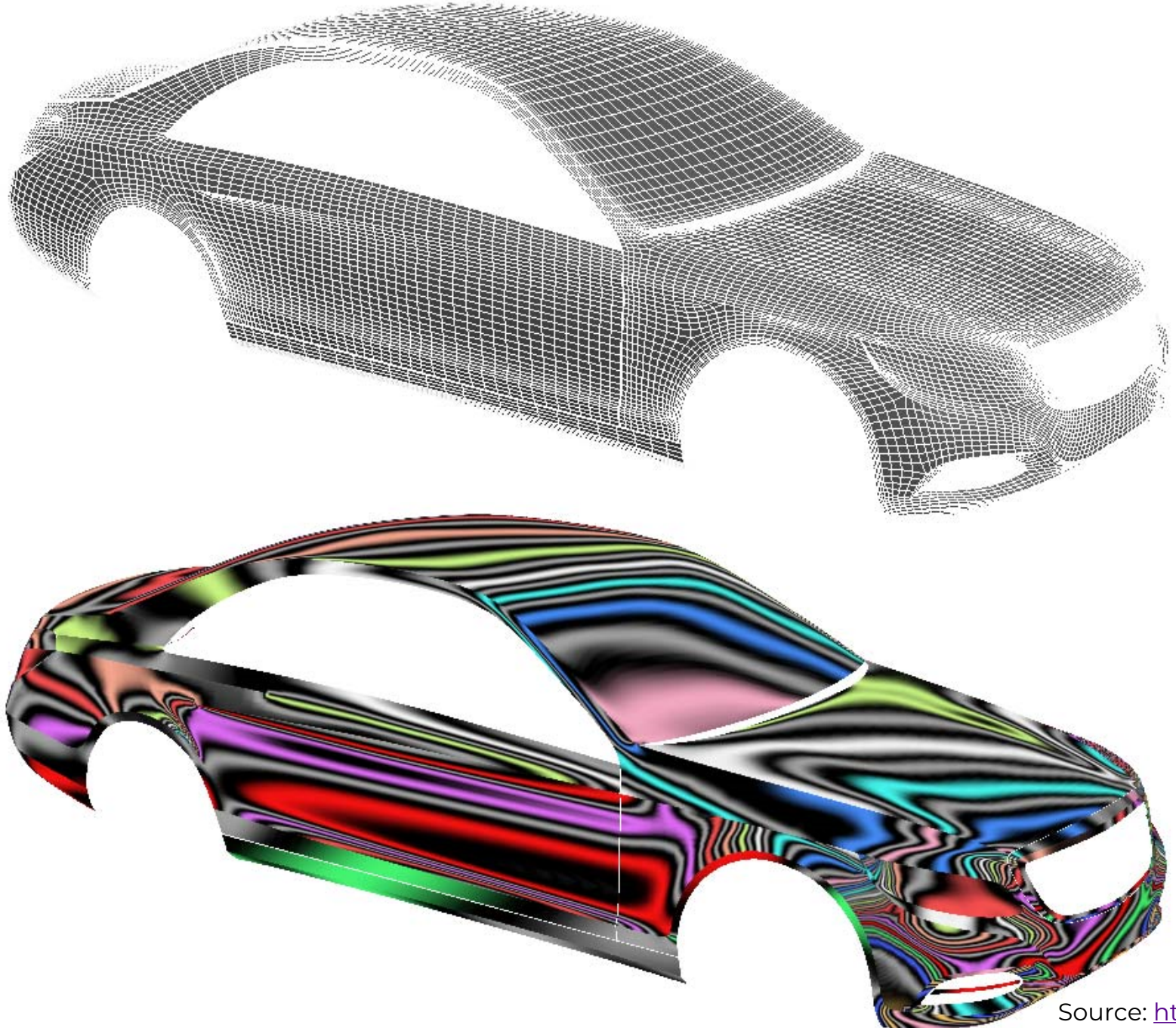


Source: http://www.cs.columbia.edu/cg/mesh-arrangements/mesh-arrangements-for-solid-geometry-siggraph-2016-zhou-et-al.pdf

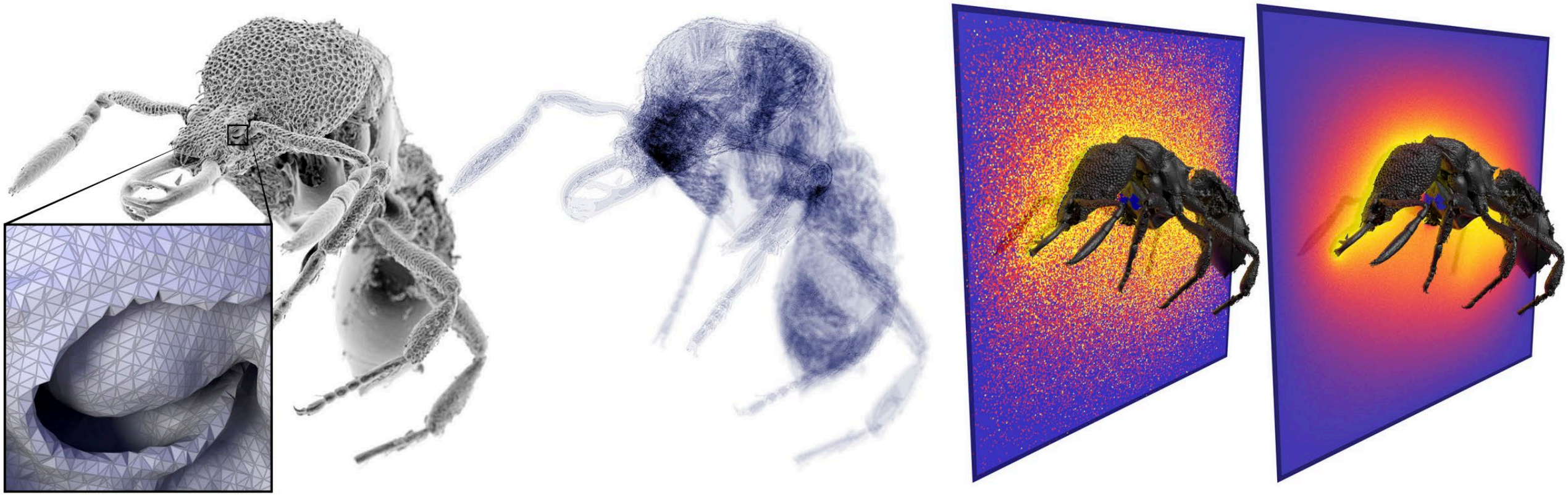


Stream Line Visualization of the Flow around a Backward-Facing Step

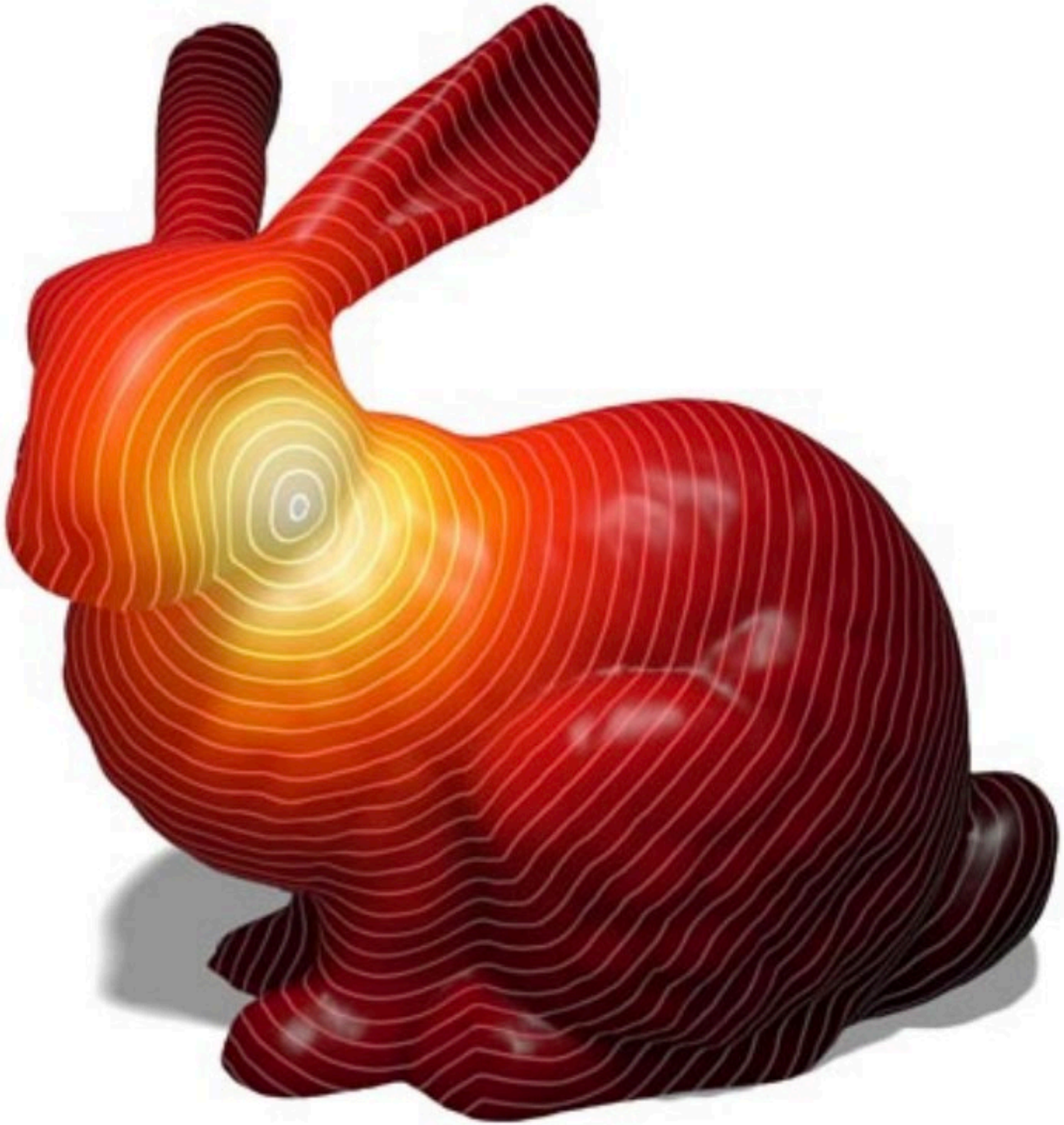
Source: https://www.csc.kth.se/~weinkauf/notes/step.html



Source: https://www.ebalstudios.com/blog/modeling-cars-polygons

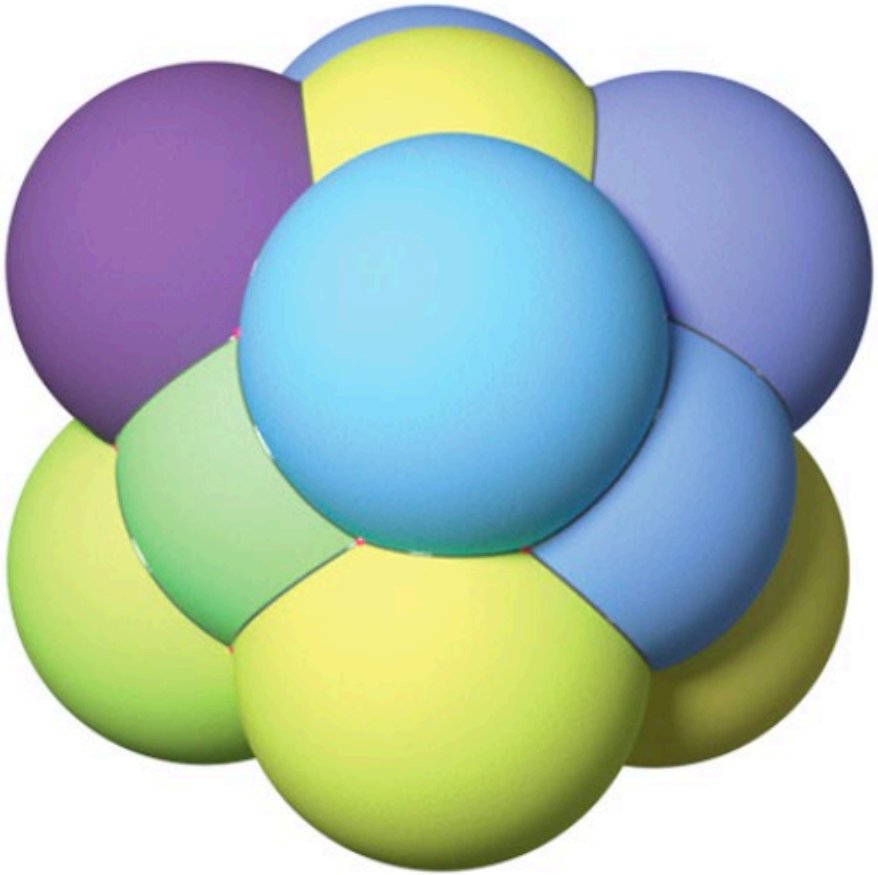


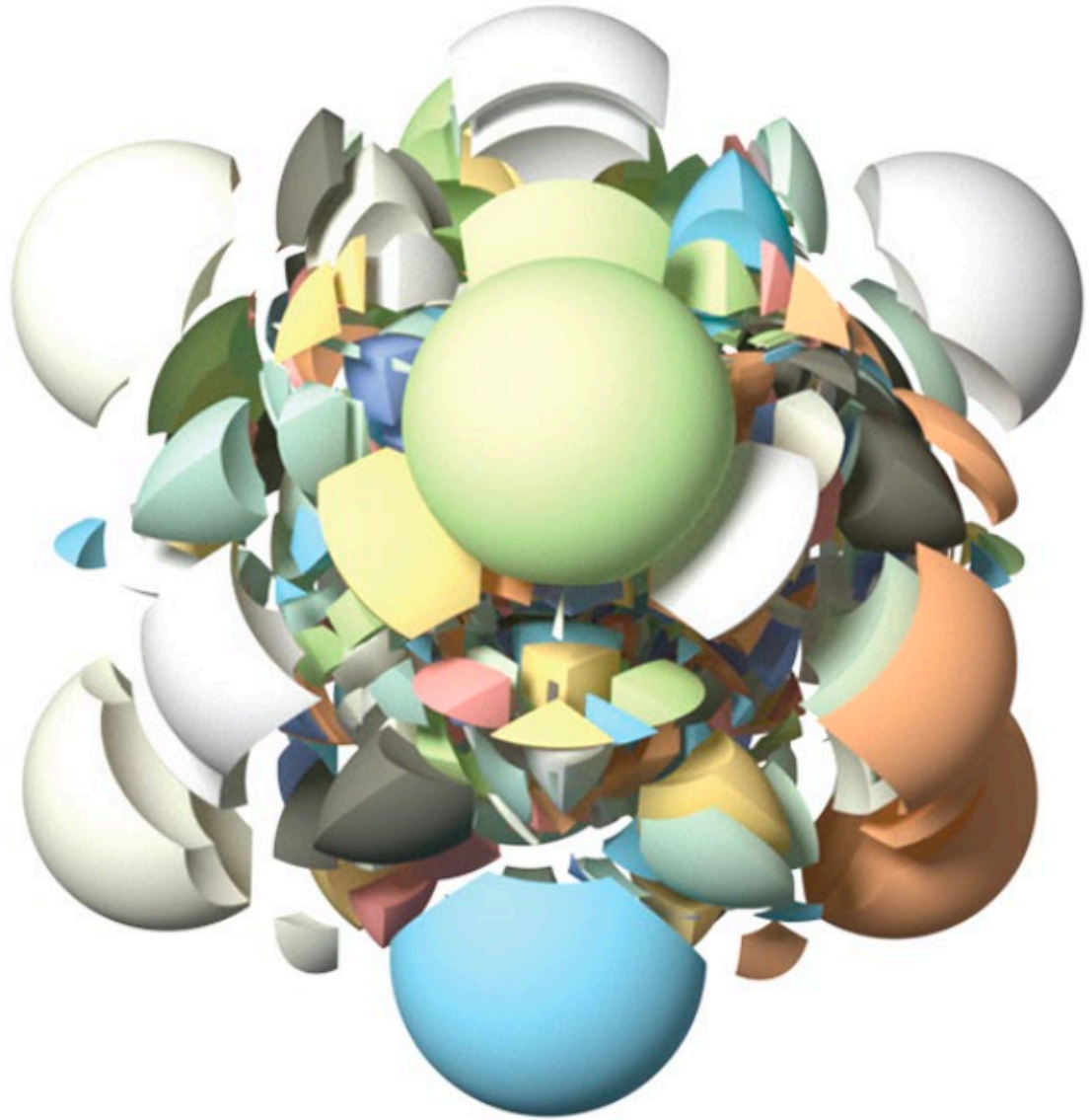
Source: https://www.cs.cmu.edu/~kmcrane/Projects/MonteCarloGeometryProcessing/paper.pdf



Source: https://arxiv.org/pdf/1204.6216.pdf

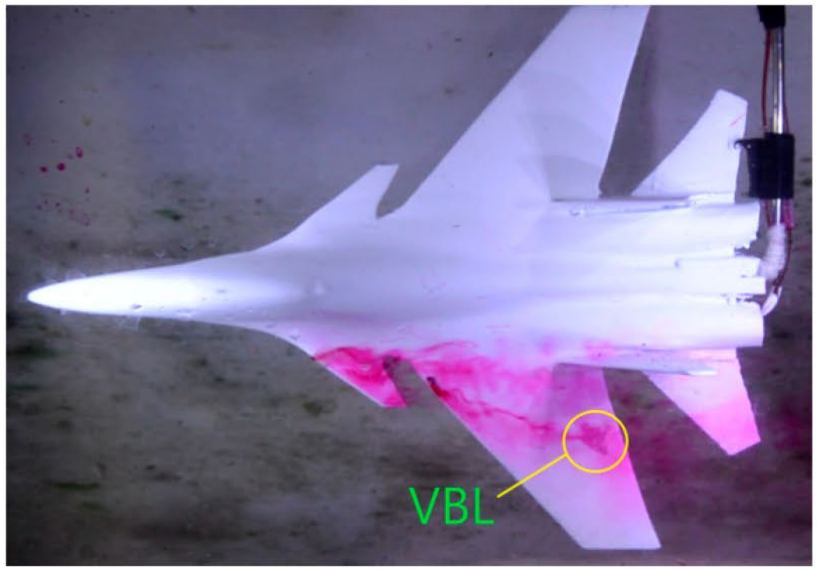
Source: https://github.com/PyMesh/PyMesh



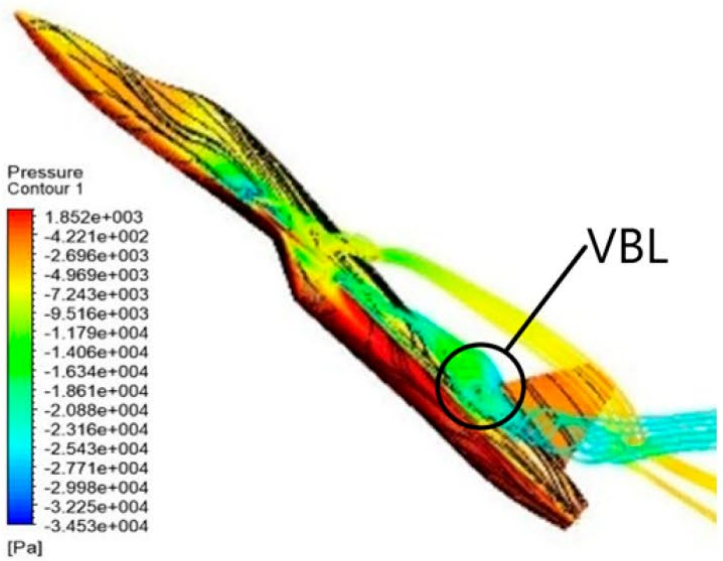


A NASA study on wingtip vortices

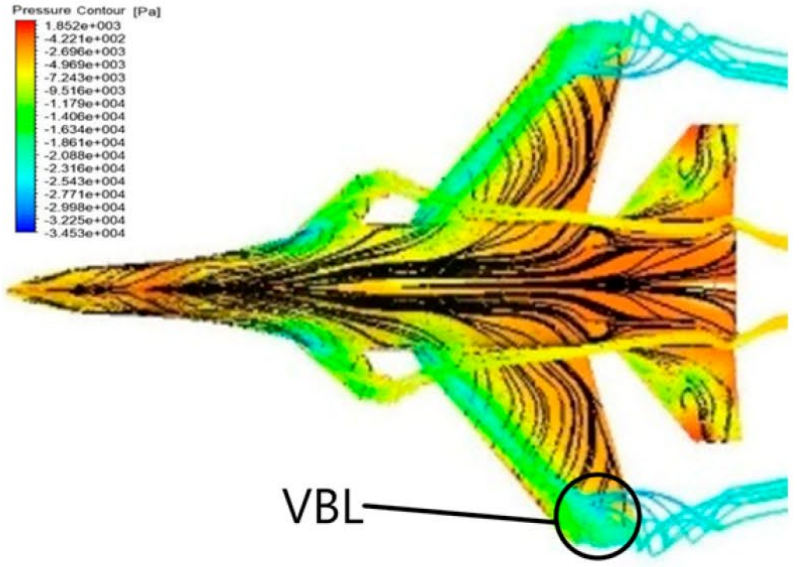
Sourece: https://en.wikipedia.org/wiki/Wingtip_vortices



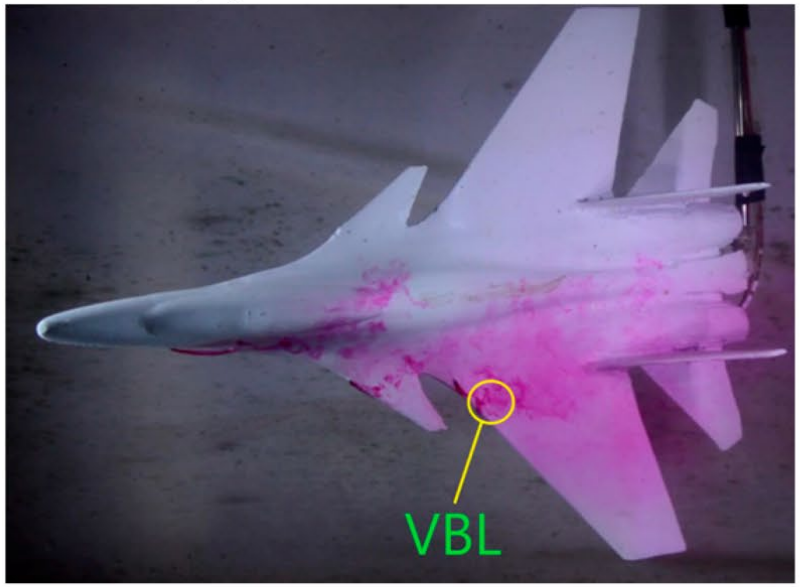
(a) WaTu AoA 30°



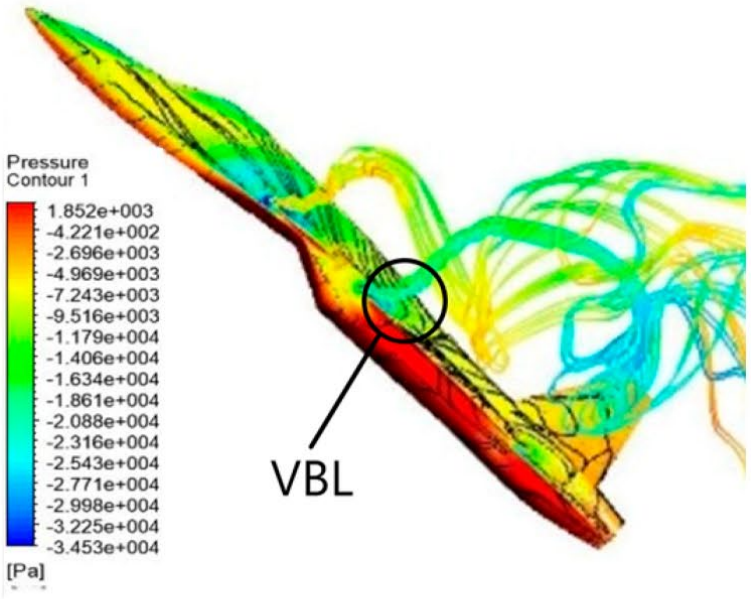
(b) CFD AoA 30°; side view



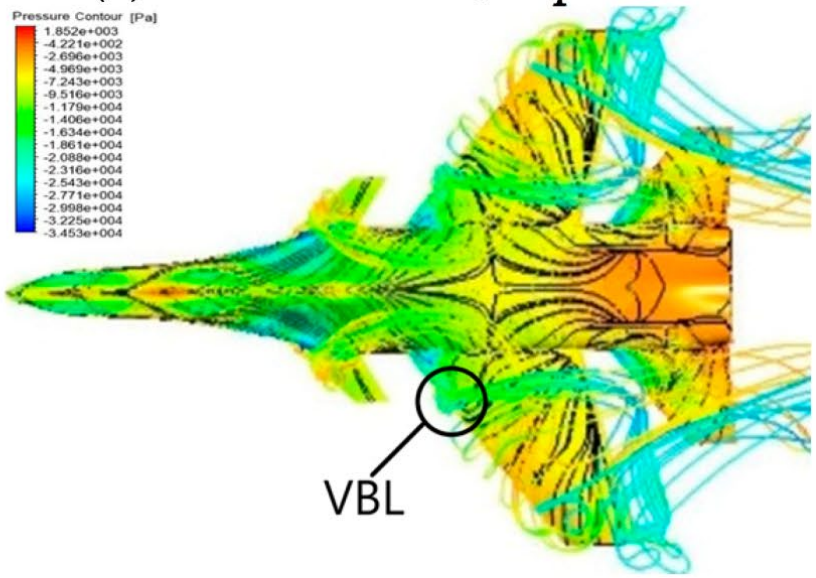
(c) CFD AoA 30°; top view



(d) WaTu AoA 50°



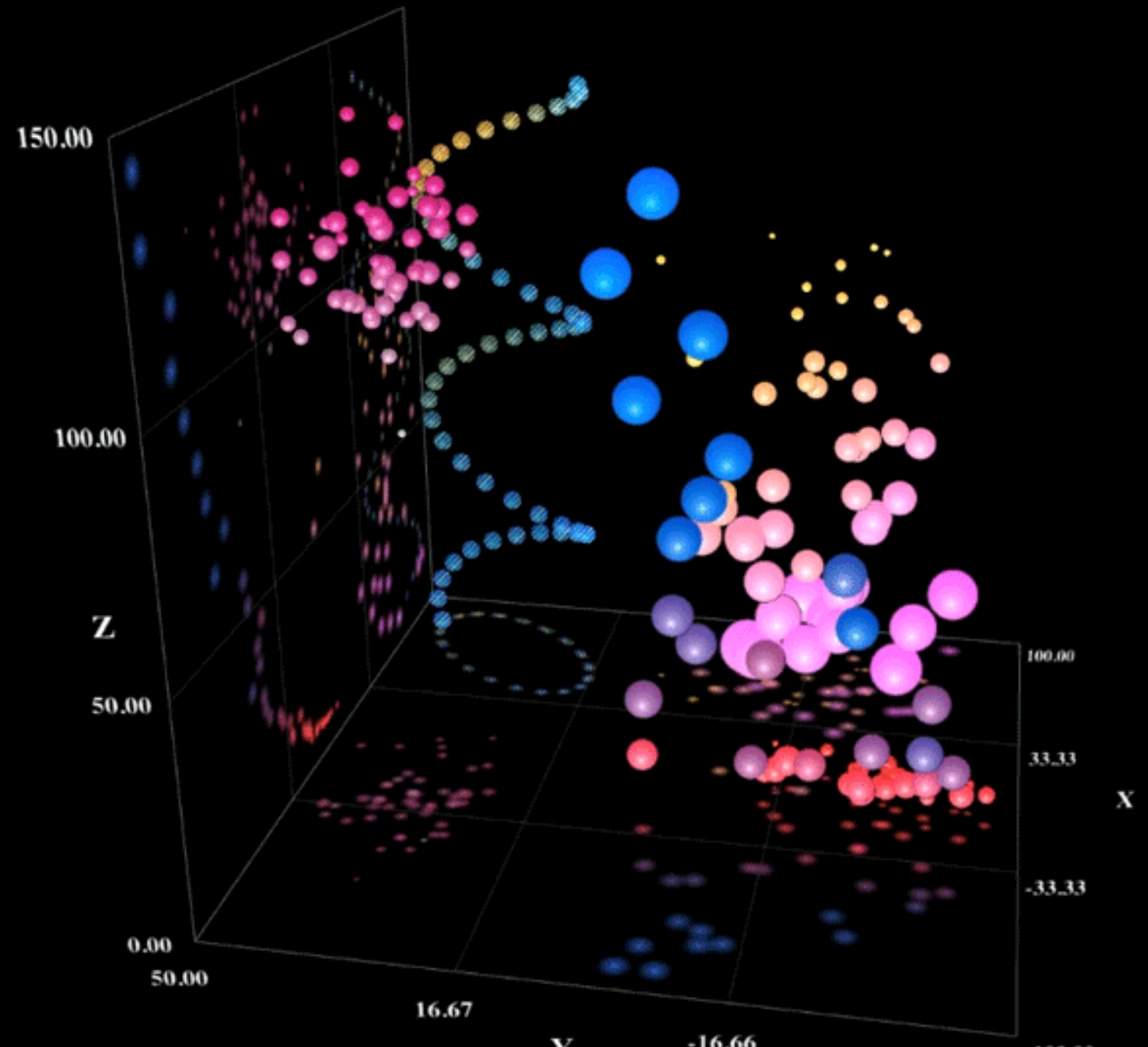
(e) CFD AoA 50°; side view



(f) CFD AoA 50°; top view

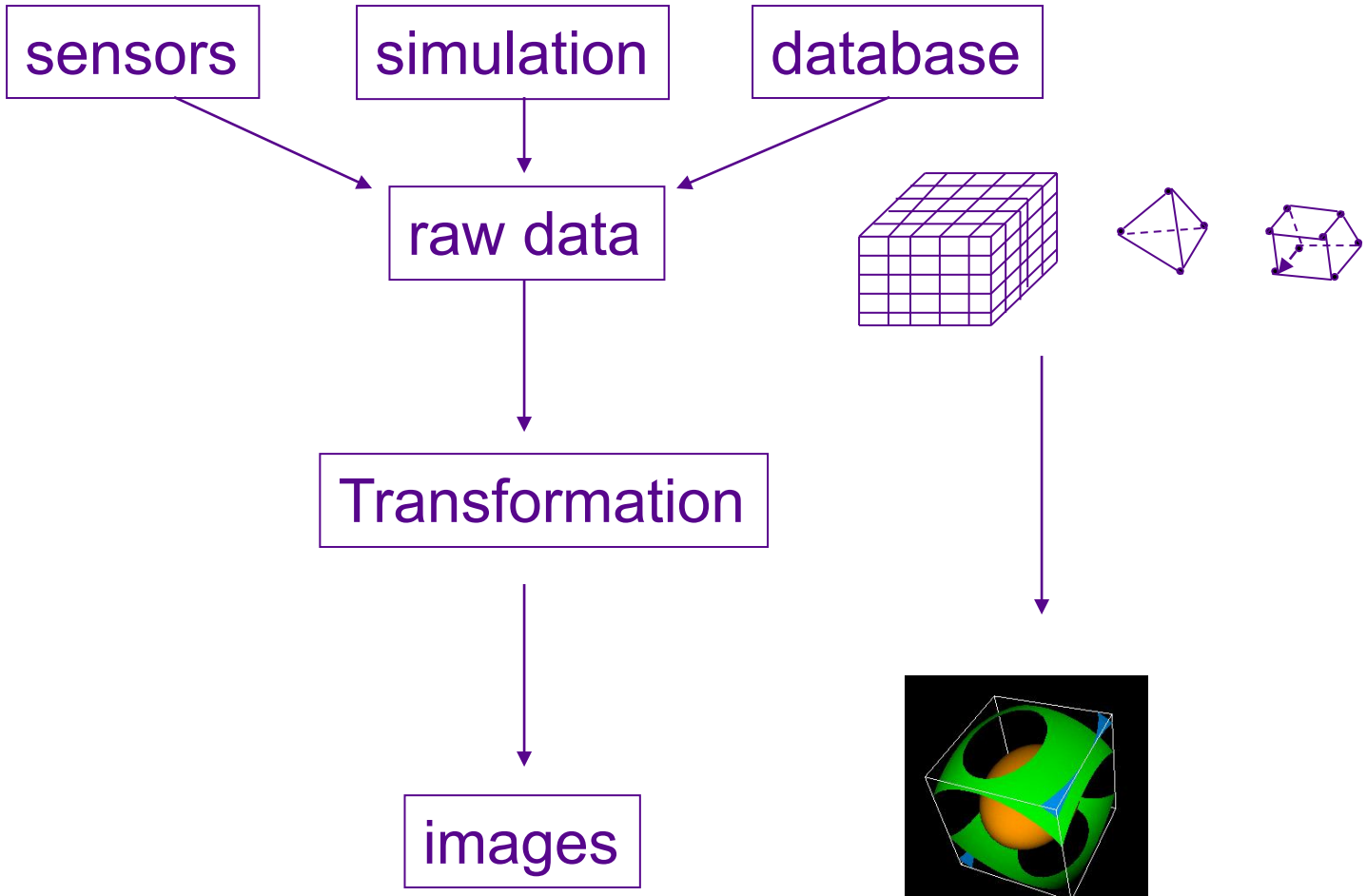
Source: https://www.mdpi.com/2226-4310/6/2/12/htm


Lombard St
Wick St

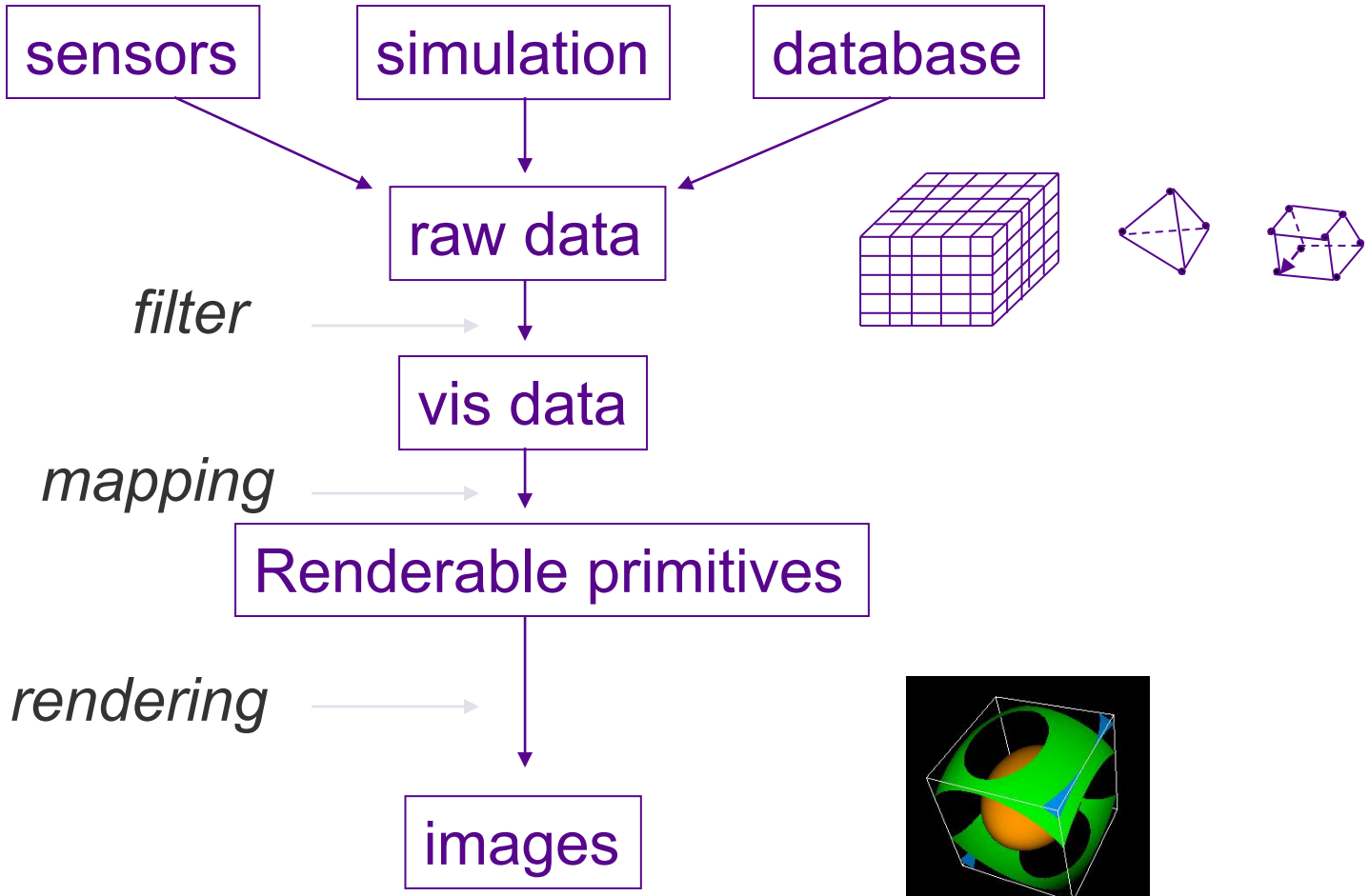

150.00
100.00
Z
50.00
0.00
50.00
16.67
-16.66
100.00
33.33
-33.33
X

3D Visualization Pipeline

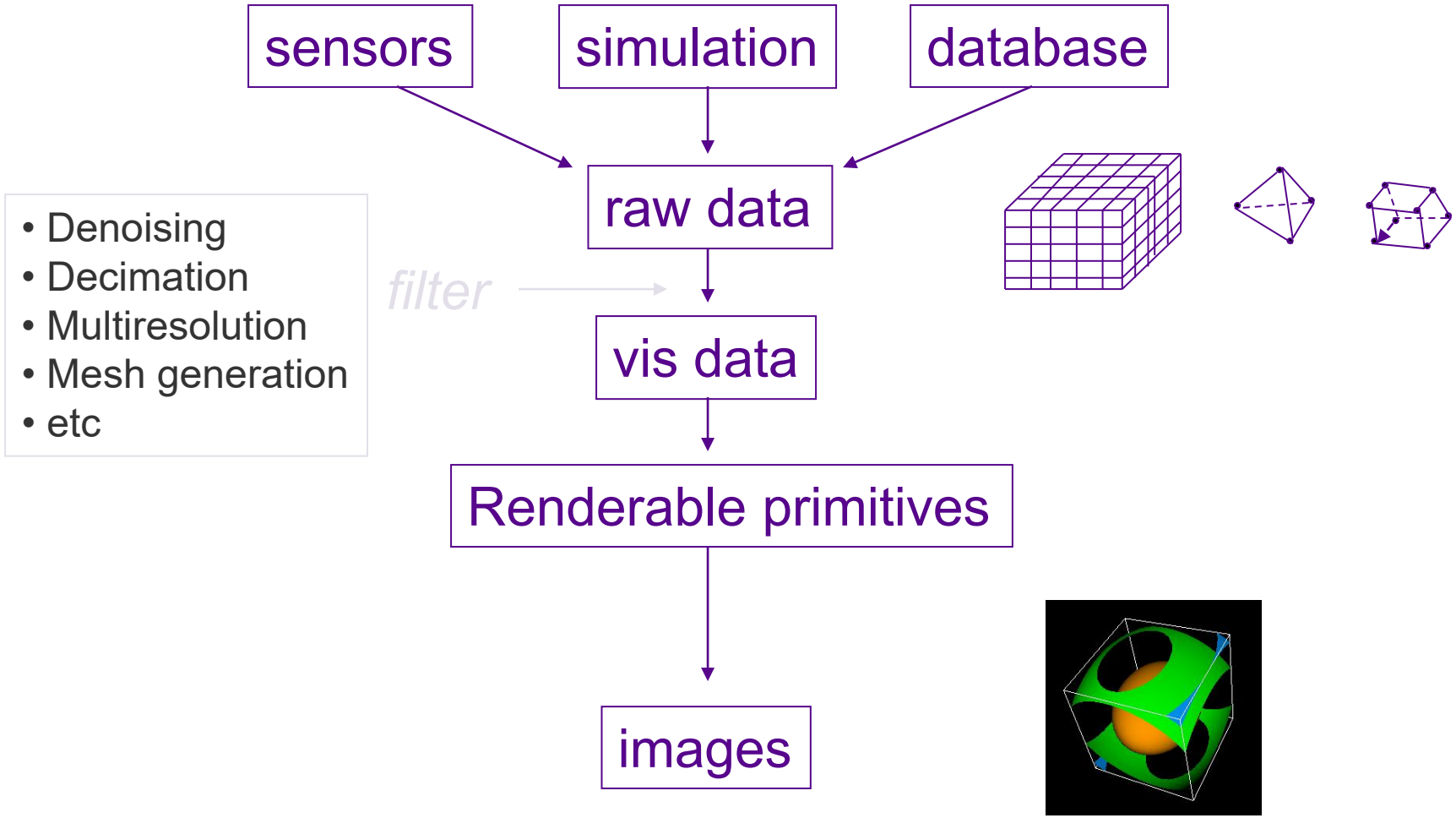
Visualization Process



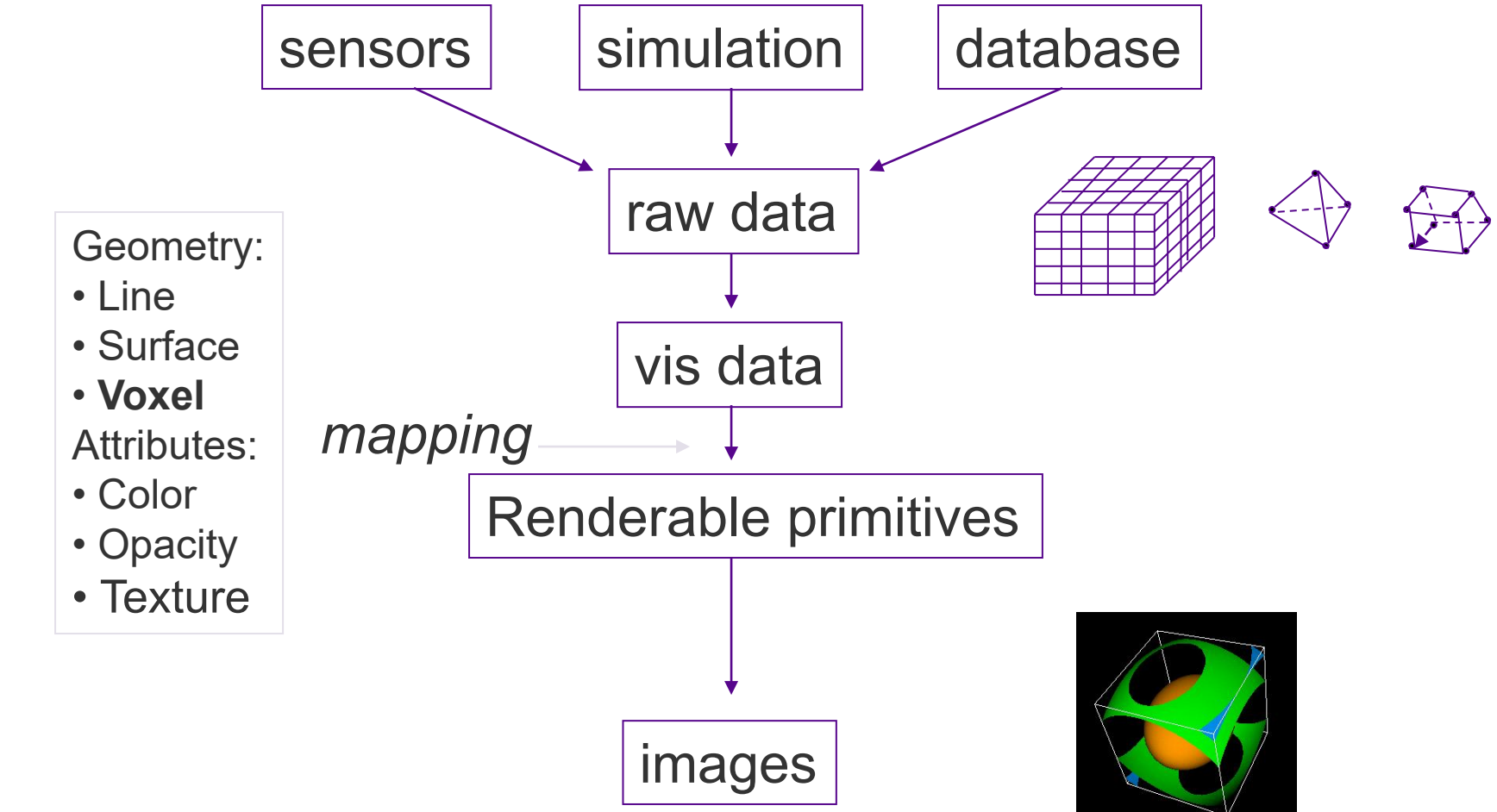
Visualization Pipeline



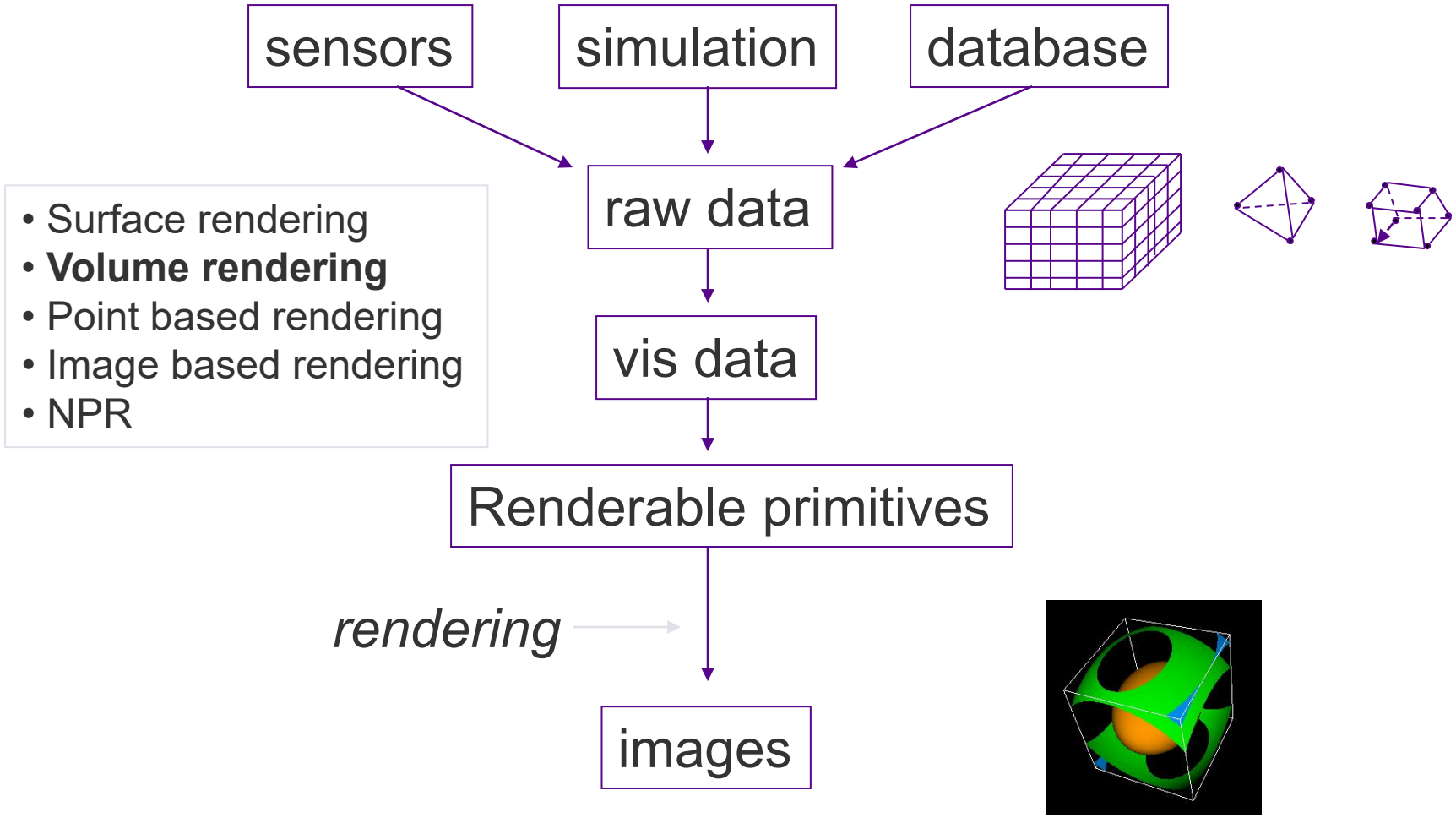
Visualization Pipeline



Visualization Pipeline

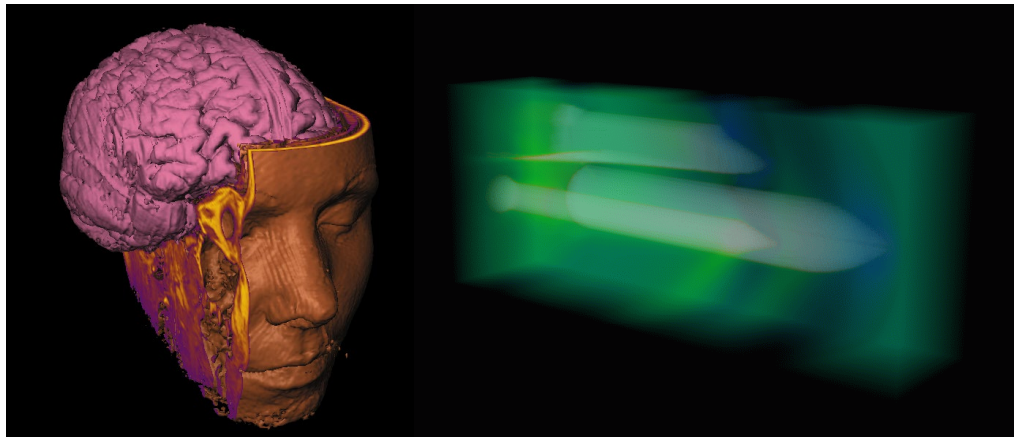


Visualization Pipeline

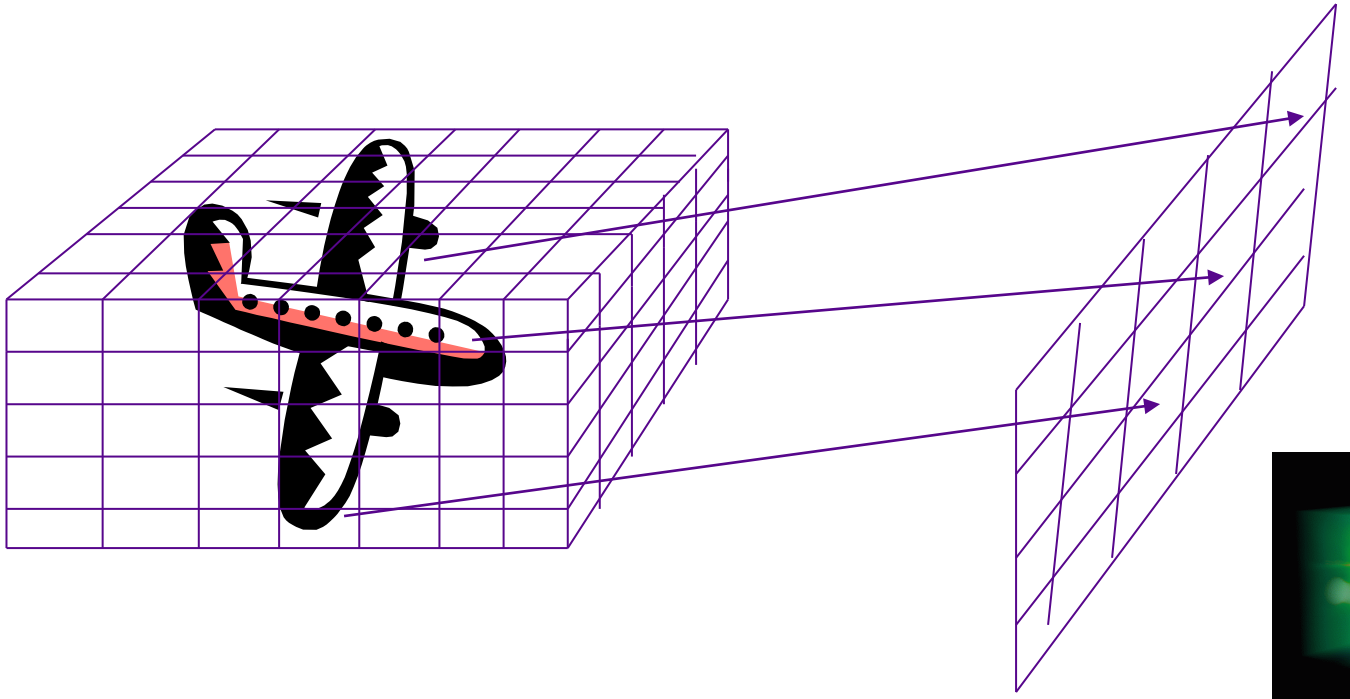


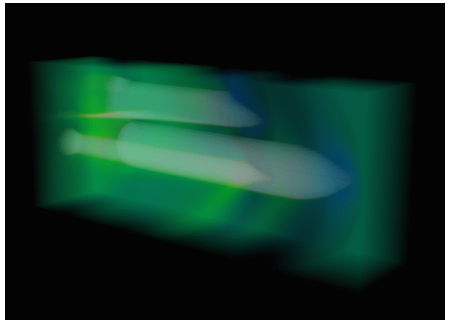
Volume Rendering

- Goal: visualize three-dimensional functions
  - Measurements (medical imaging)
  - Numerical simulation output
  - Analytic functions

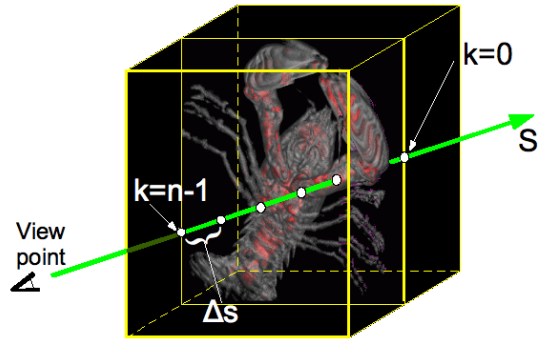


Volume Rendering



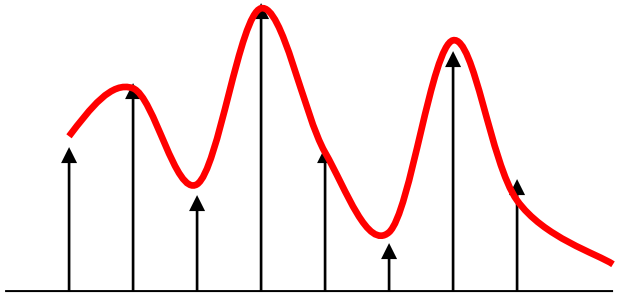


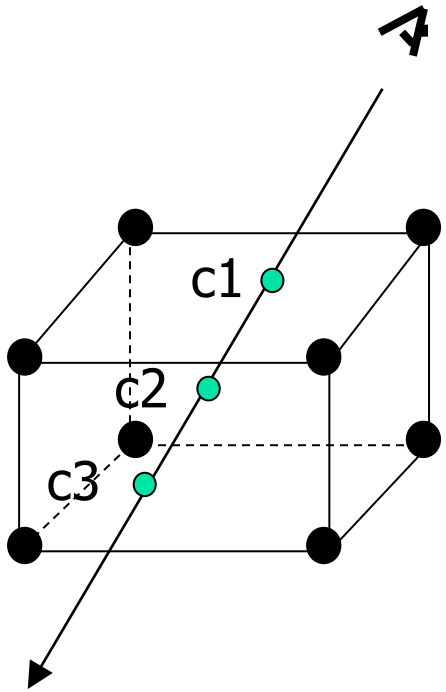
Volume Data Visualization



Ray Casting

Recover the original continuous function from discrete samples





View Synthesis from Volume Rendering: Neural Radiance Field

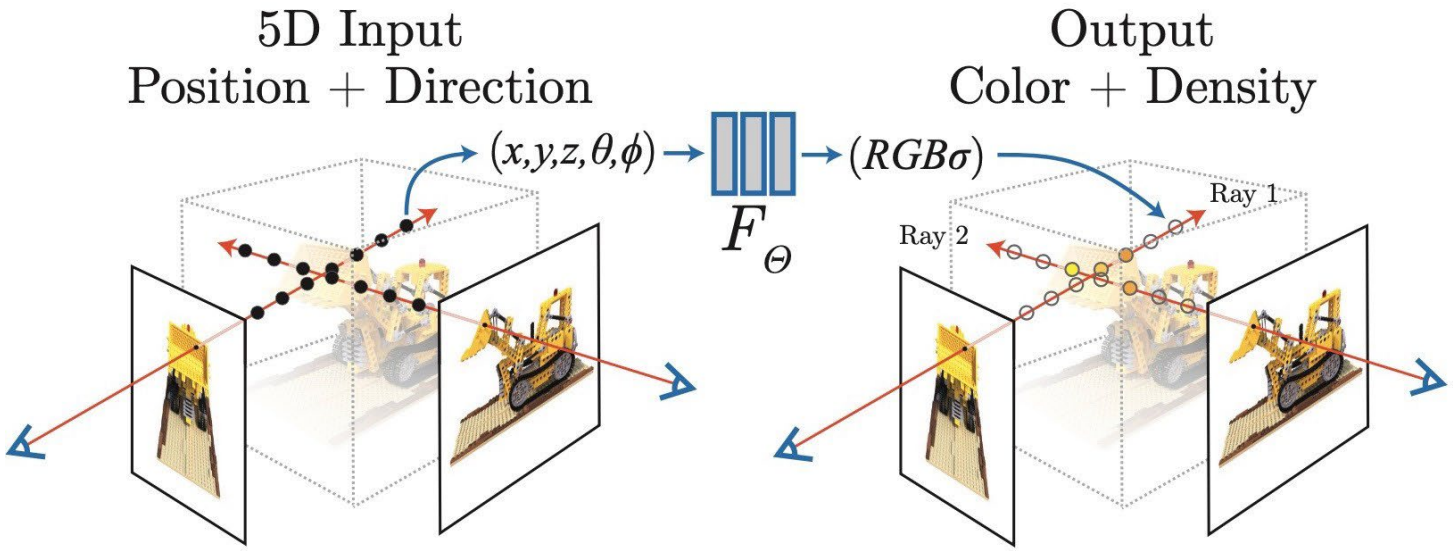
- Given a dataset containing RGB images of a static scene, their corresponding camera poses, and intrinsic parameters,
- Predict the color and volume density for every viewing location and direction

Inputs:

- x, y, z Target position of the camera
- θ, ϕ Target orientation of the camera

Outputs:

- c = (R, G, B) Color
- σ Volume density



NeRF: Volume Rendering

- Generating a view from NeRF requires rendering all rays that pass through each pixel of the desired virtual camera

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Expected color of a camera ray

Predicted Volume Density

Predicted Color

Probability that nothing has blocked the ray up to this point

- Numerically estimated using quadrature and stratified sampling

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$$

The relative contribution of this segment

Predicted Color

Probability that nothing has blocked the ray up to this point

- Differentiable: allows optimization using gradient descent

NeRF: Hierarchical Sampling

- Problem: It is inefficient to integrate over empty and occluded spaces in a scene
- Solution: Allocate samples proportionally to their expected effect on the final rendering.

- Evaluate a “coarse” network on a set of N_c locations along a ray to produce a PDF along the ray

- Evaluate a “fine” network on N_c and a second a set of N_f locations sampled from the PDF

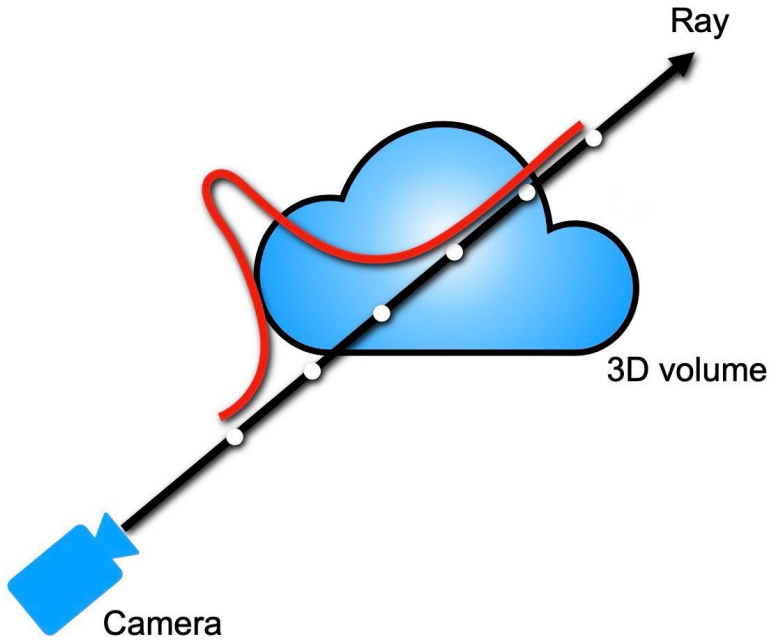


Image: Jonathan Barron

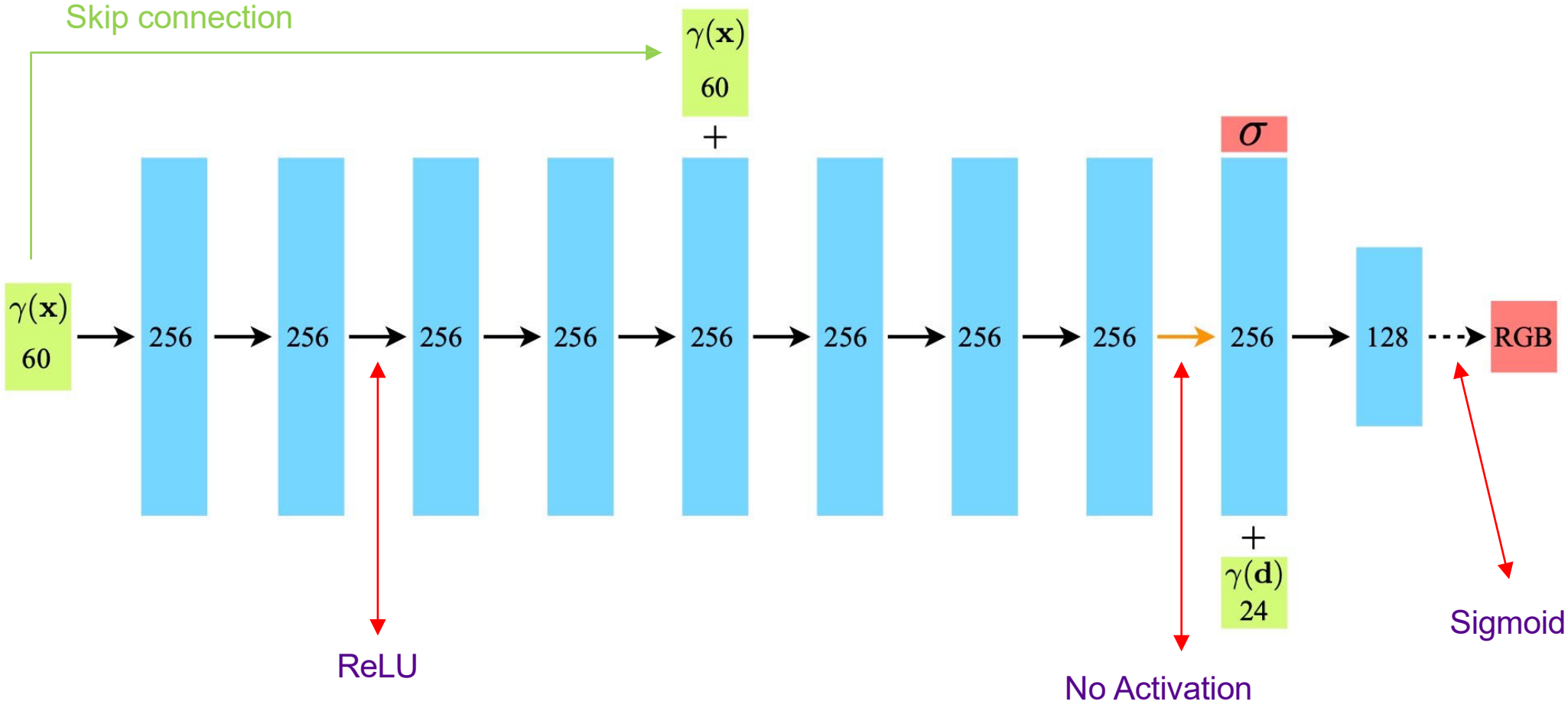
NeRF: Positional Encoding

- Map individual components of position and direction vectors to a higher dimensional space

$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right)$$

- Similar concept as positional encoding in Transformer Architectures
- Empirically improves the preservation of high-frequency geometry and texture
- Surprising result, explored further in a follow-up work by the same authors
  - “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains” (Tancik, Srinivasan, Mildenhall, et al., 2020)

Network Architecture



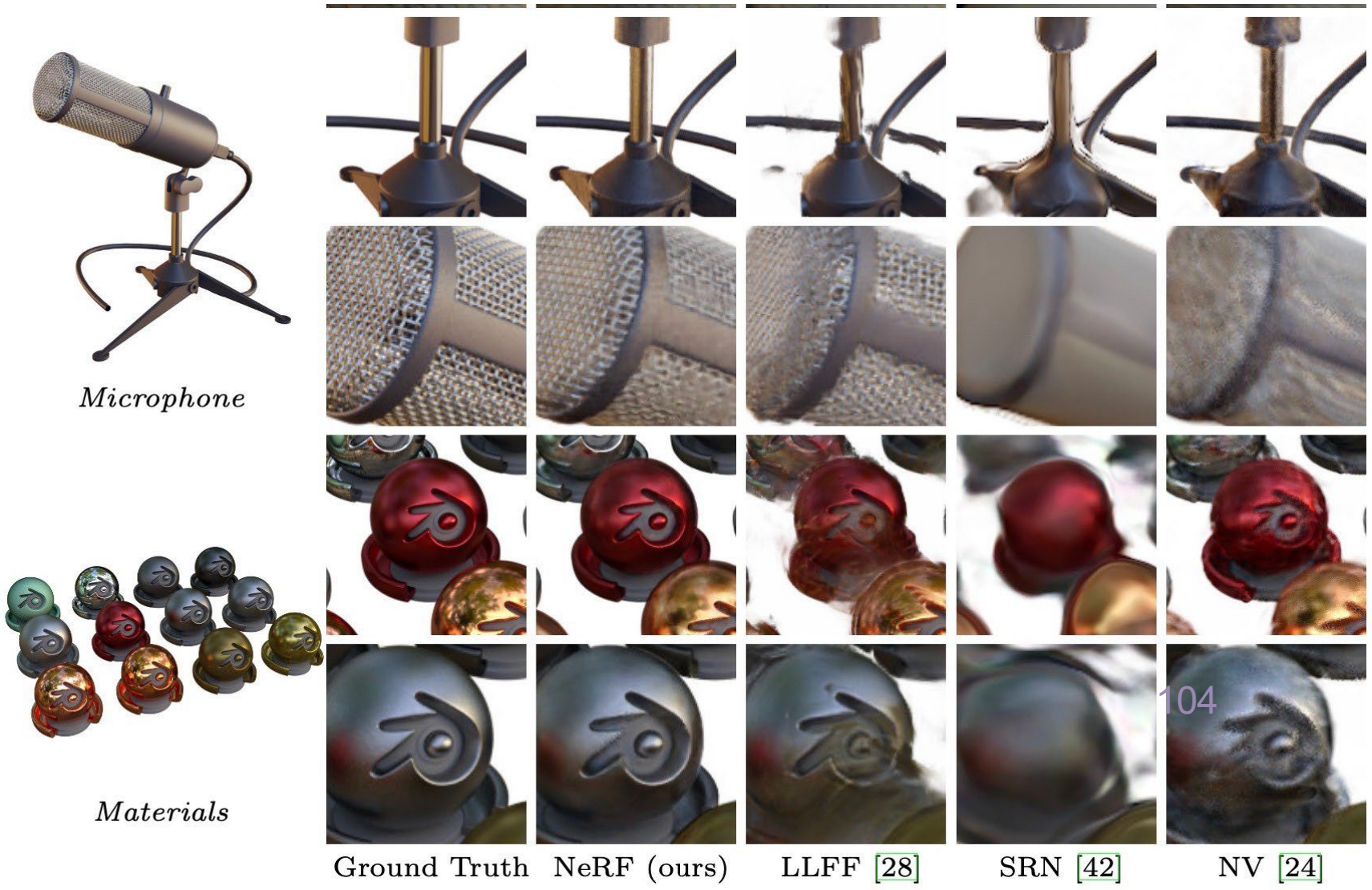
Training Summary

- Sample a batch of camera rays from the dataset (bs=4096)
- Use hierarchical sampling to query coarse and fine points
- Use the volume rendering equation to calculate the color of the ray
- Compute the squared error between rendered and true pixel colors

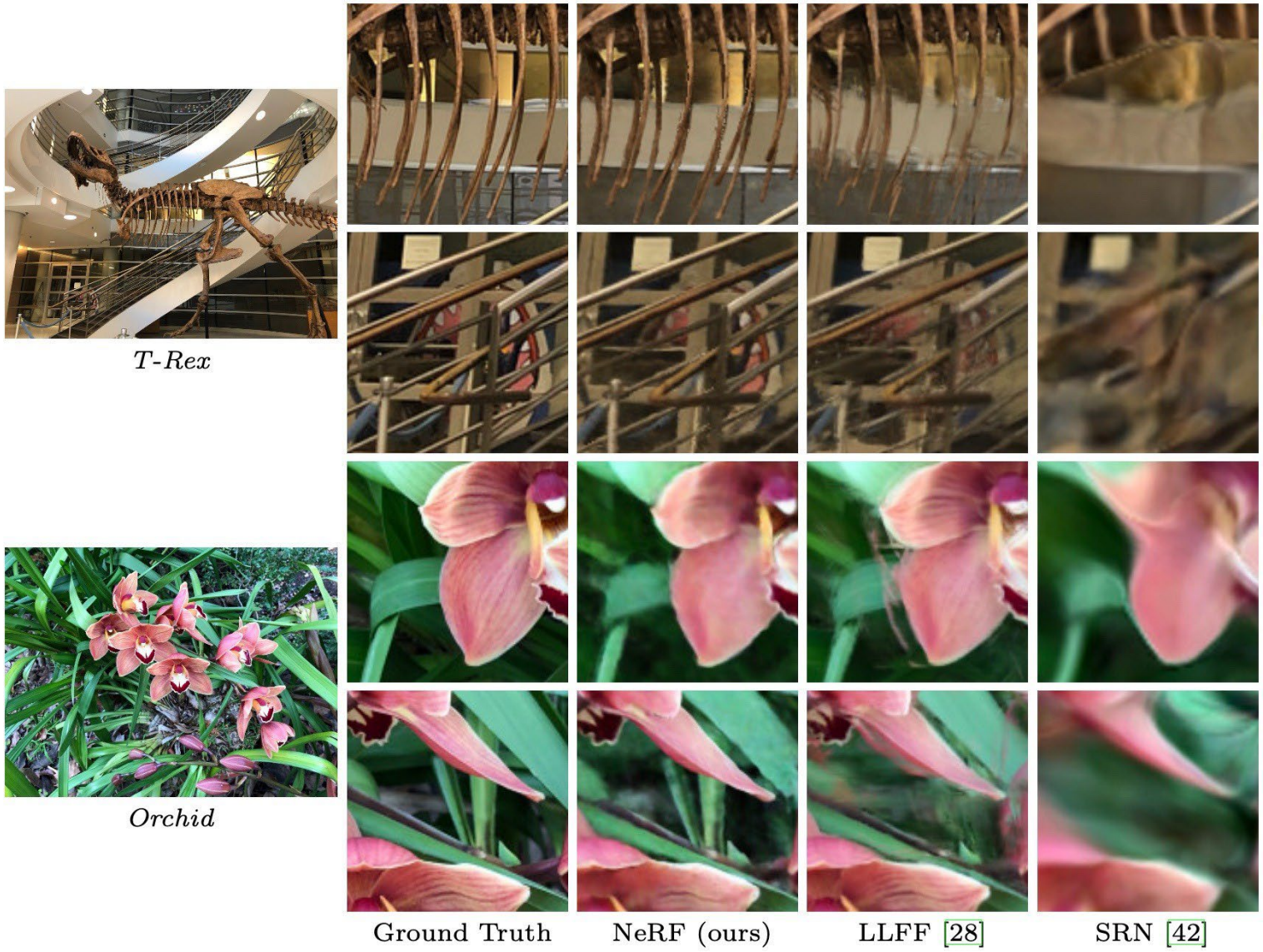
$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

- Optimize network parameters using Adam

Qualitative Results: Realistic Synthetic Objects



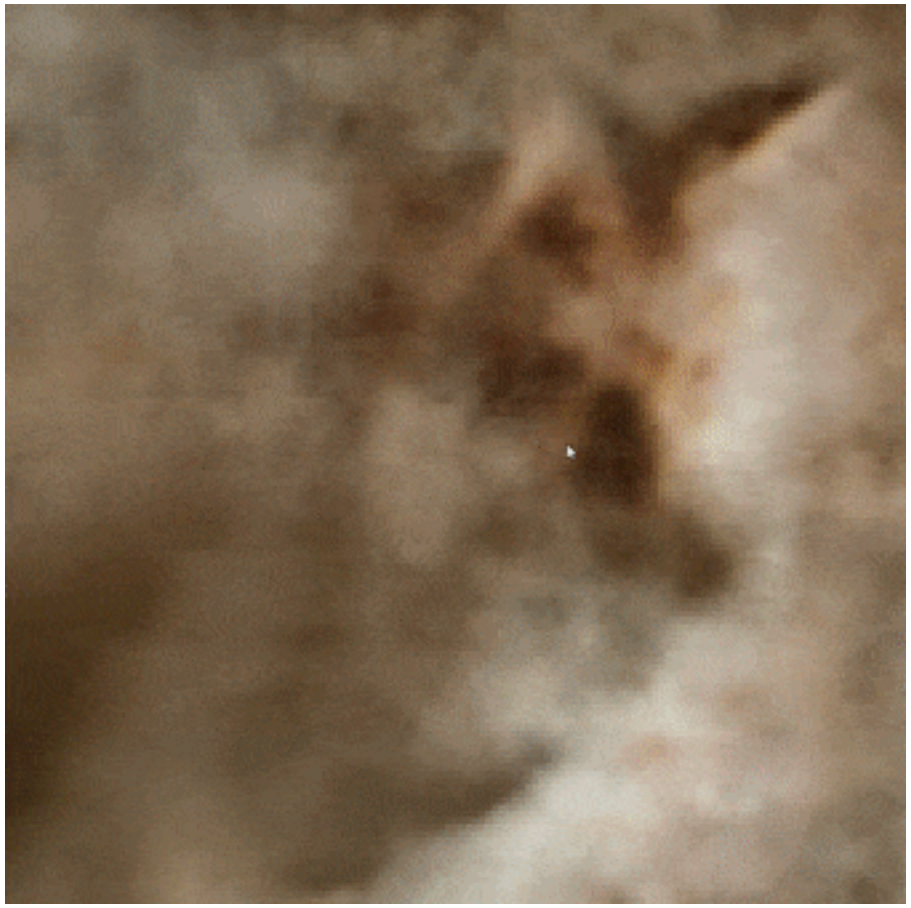
Qualitative Results: Real-world scenes

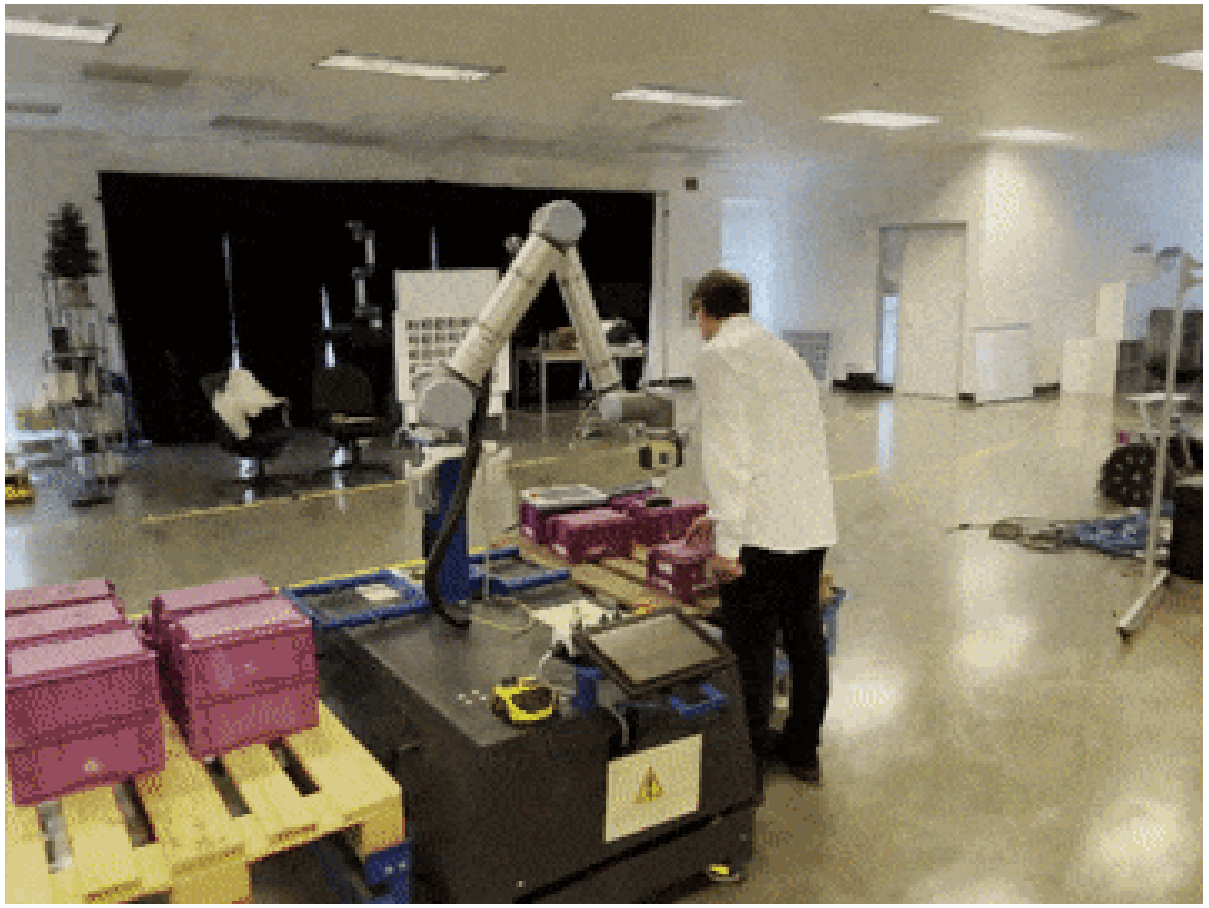


Animated Results

Video: https://www.matthewtancik.com/nerf

Instant Neural Graphics Primitives





https://github.com/NVlabs/instant-ngp

Applications

NeX: Real-time View Synthesis with Neural Basis Expansion

Readings

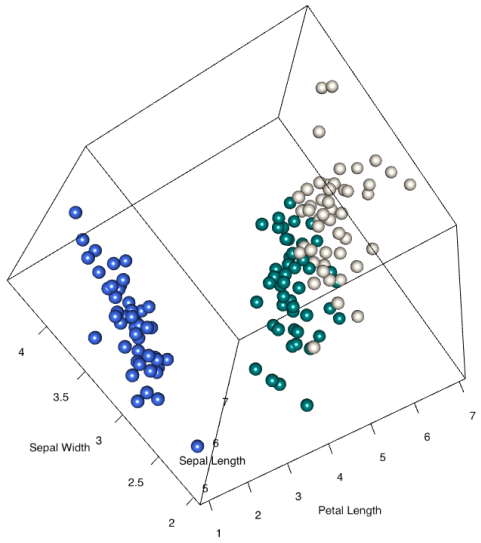
- Neural Volumes (NV), 2019
  - Deep 3D convolutional network architecture
  - Predicts a fixed-size discretized voxel grid
  - Limitation: discrete voxel grids do not scale well and lose fine detail at high resolutions
  - Limitation: requires a bounded volume and knowledge of the background
- Scene Representation Networks (SRN), 2019
  - Uses a recurrent neural network to model a rendering function
  - Limited to simple shapes with low geometric complexity
- Local Light Field Fusion (LLFF), 2019
  - 3D convolutional network architecture
  - Predicts multiplane images and fuses them to create new views
  - Fast to train (<10 minutes) at the cost of large storage requirements (~GB for each scene)

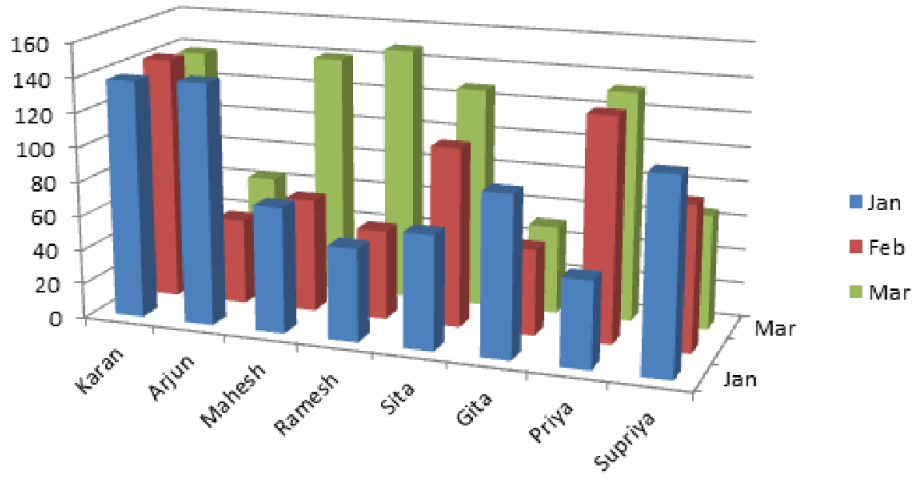
3D Visualization for non-3D Data

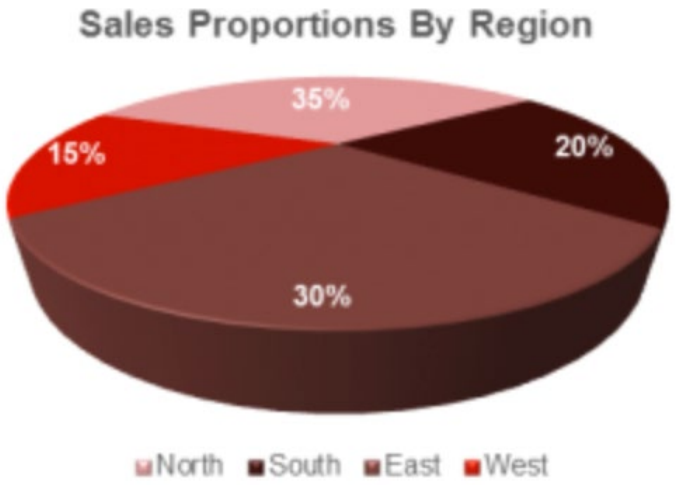

To 3D or not to 3D?

When not to 3D?

- 2D visualization is sufficient.
- 3rd dimension encode no information.
- Size/lengths are foreshortened.
- Occlusion.







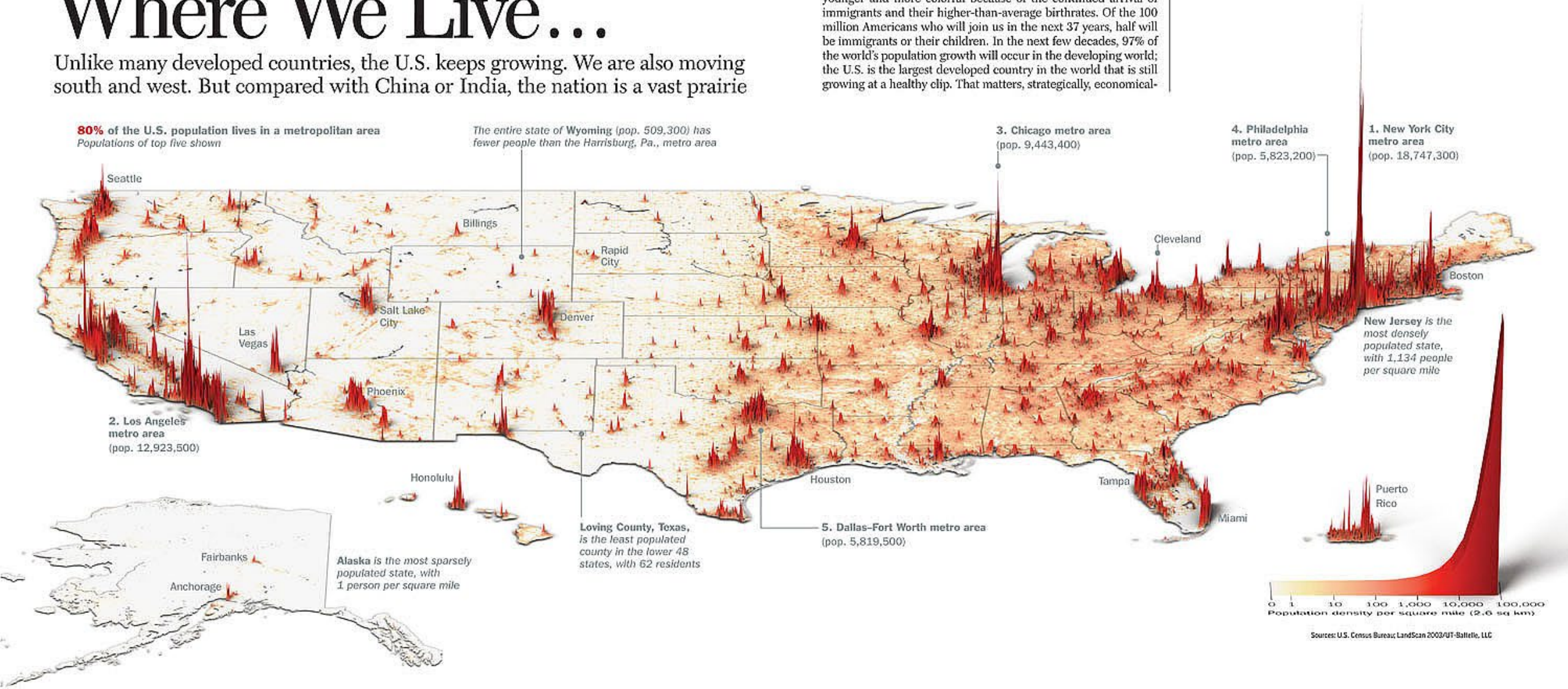
Non-3D Data

- Data that are not intrinsically 3D-related.
  - No variables that are directly related to x, y and z.
  - E.g. stock price, house price, votes, population, etc.

Where We Live…

Unlike many developed countries, the U.S. keeps growing. We are also moving south and west. But compared with China or India, the nation is a vast prairie

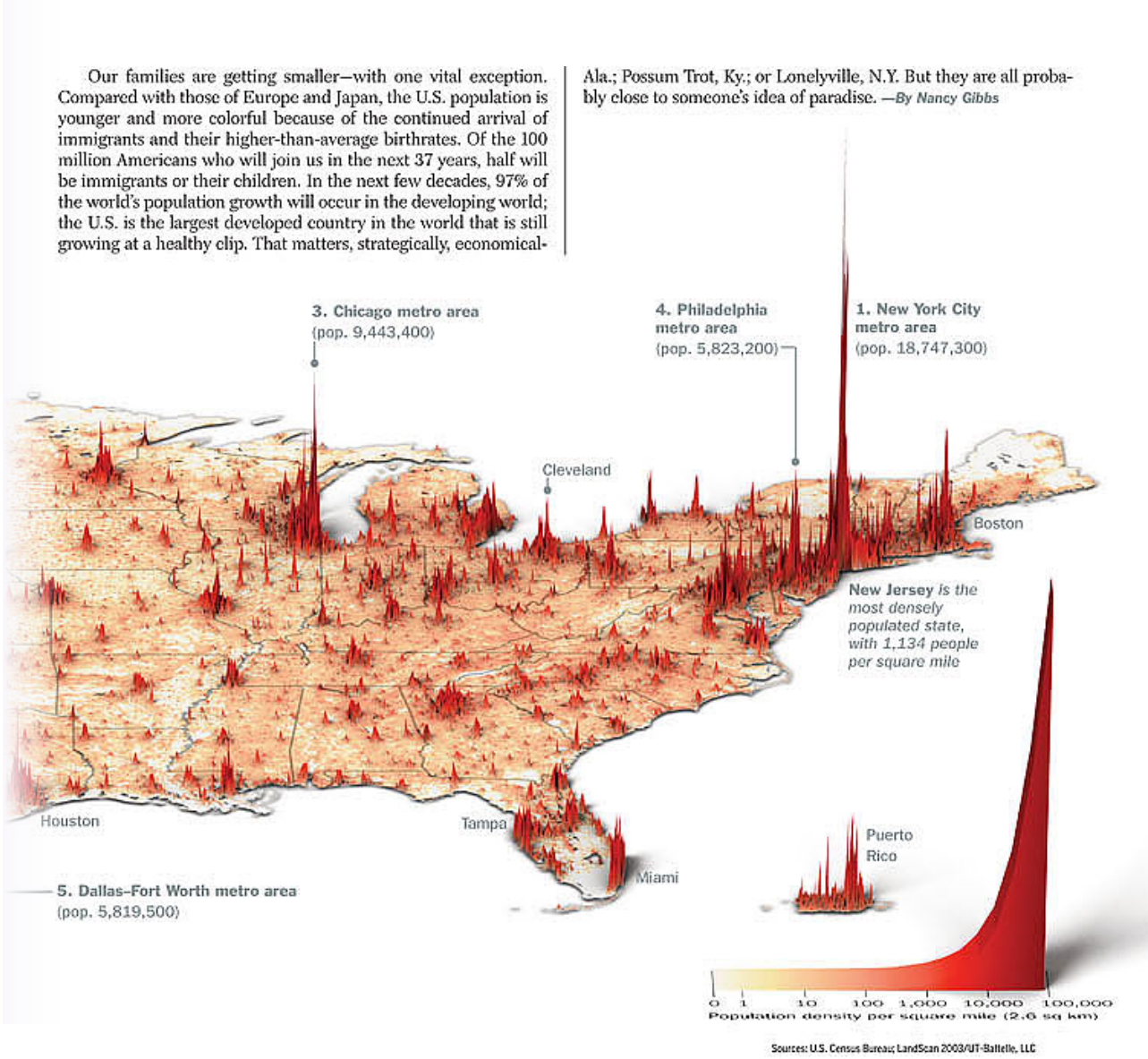
Our families are getting smaller—with one vital exception. Compared with those of Europe and Japan, the U.S. population is younger and more colorful because of the continued arrival of immigrants and their higher-than-average birthrates. Of the 100 million Americans who will join us in the next 37 years, half will be immigrants or their children. In the next few decades, 97% of the world's population growth will occur in the developing world; the U.S. is the largest developed country in the world that is still growing at a healthy clip. That matters, strategically, economically. Ala.; Possum Trot, Ky.; or Lonelyville, N.Y. But they are all probably close to someone's idea of paradise. —*By Nancy Gibbs*



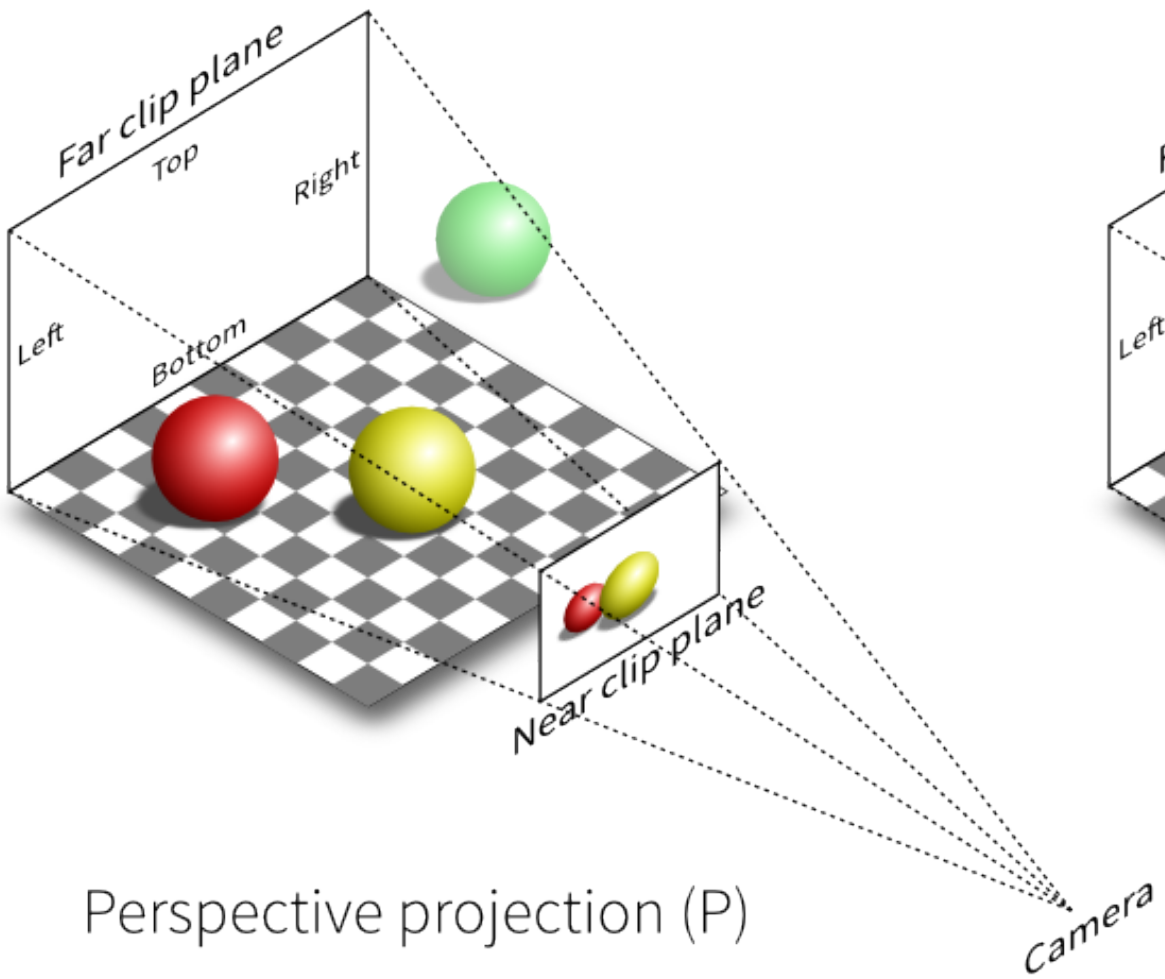
Source: https://johnstonarchitects.files.wordpress.com/2012/04/pop_lg.jpg

What works?

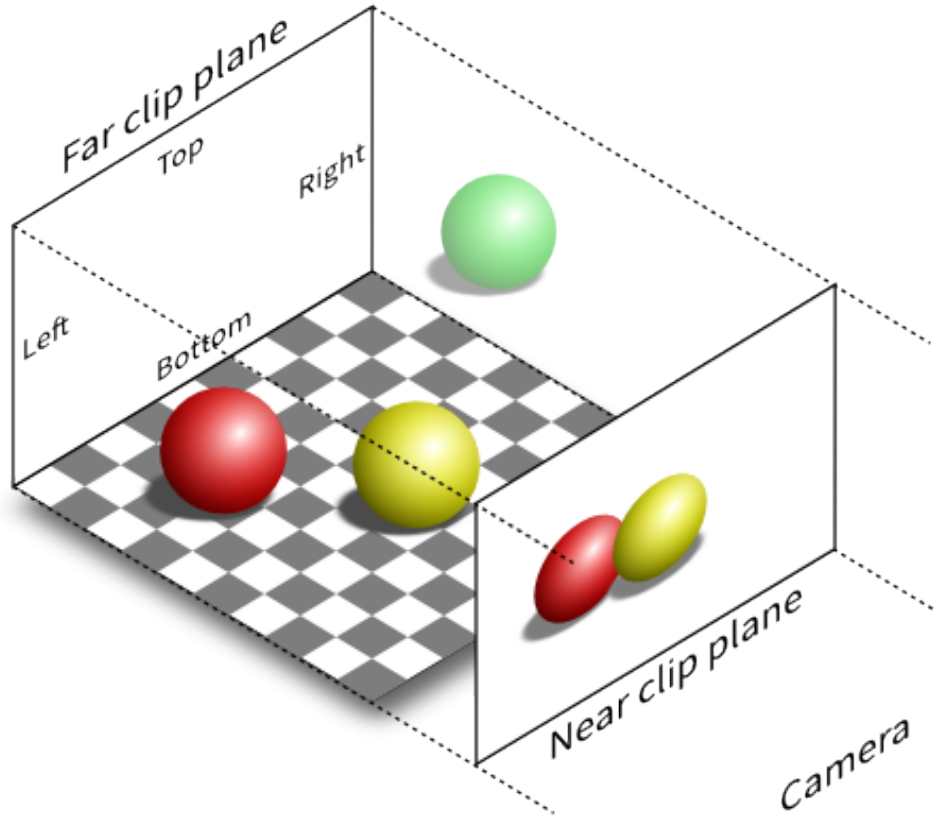
- Orthographic projection.
- Extensive labeling.
- Great rendering.
- Occlusion is not severe due to the usage of log scale.



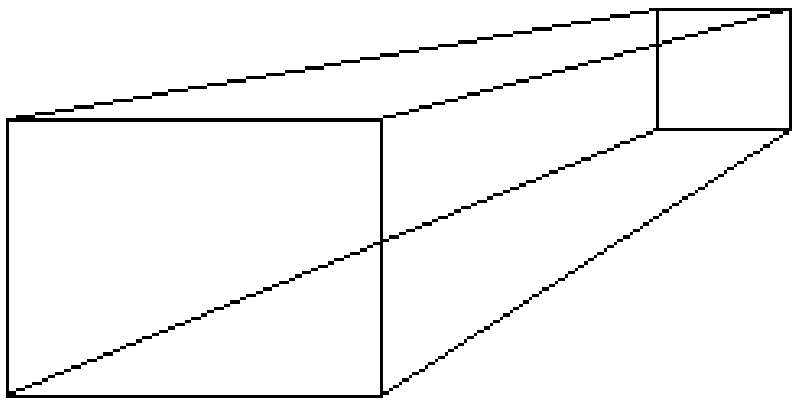
Source: https://johnstonarchitects.files.wordpress.com/2012/04/pop_lg.jpg



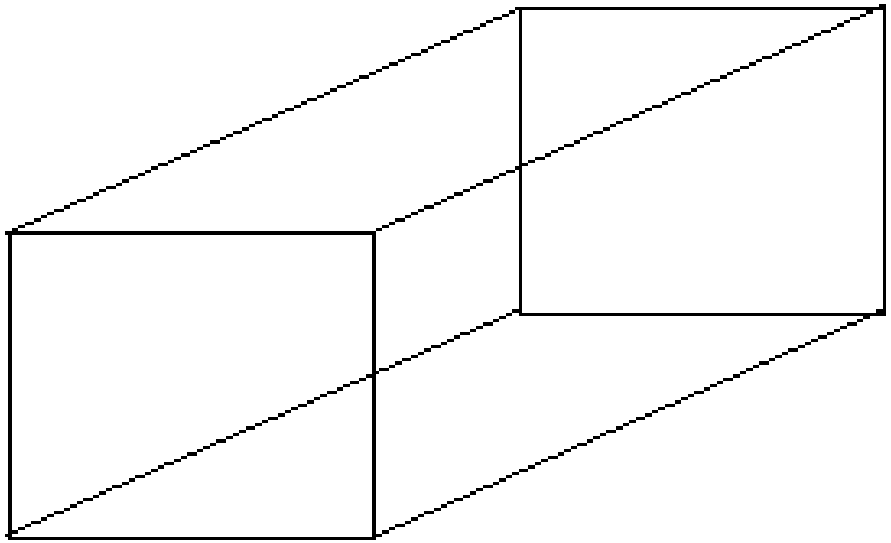
Perspective projection (P)



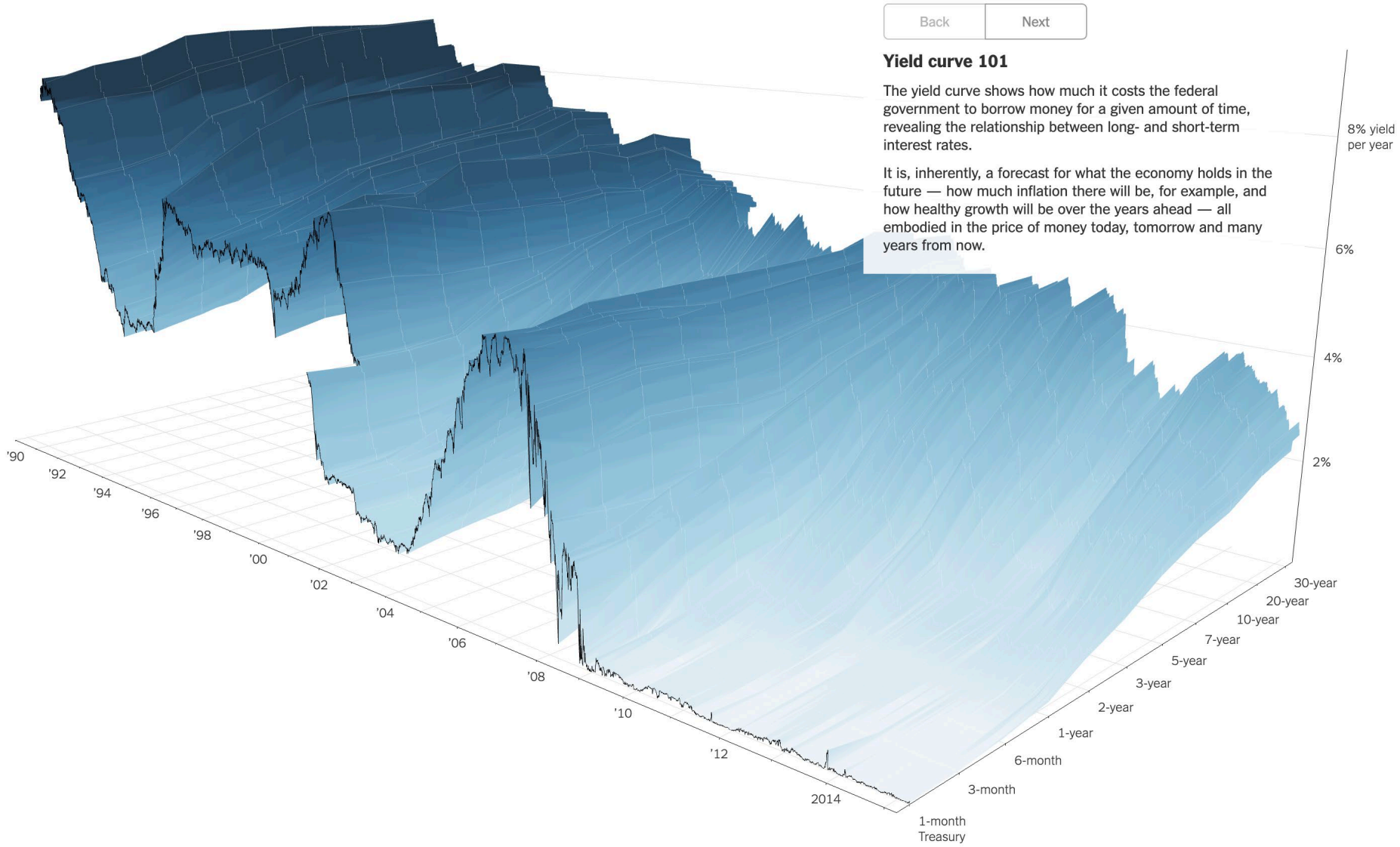
Orthographic projection (O)



Perspective projection



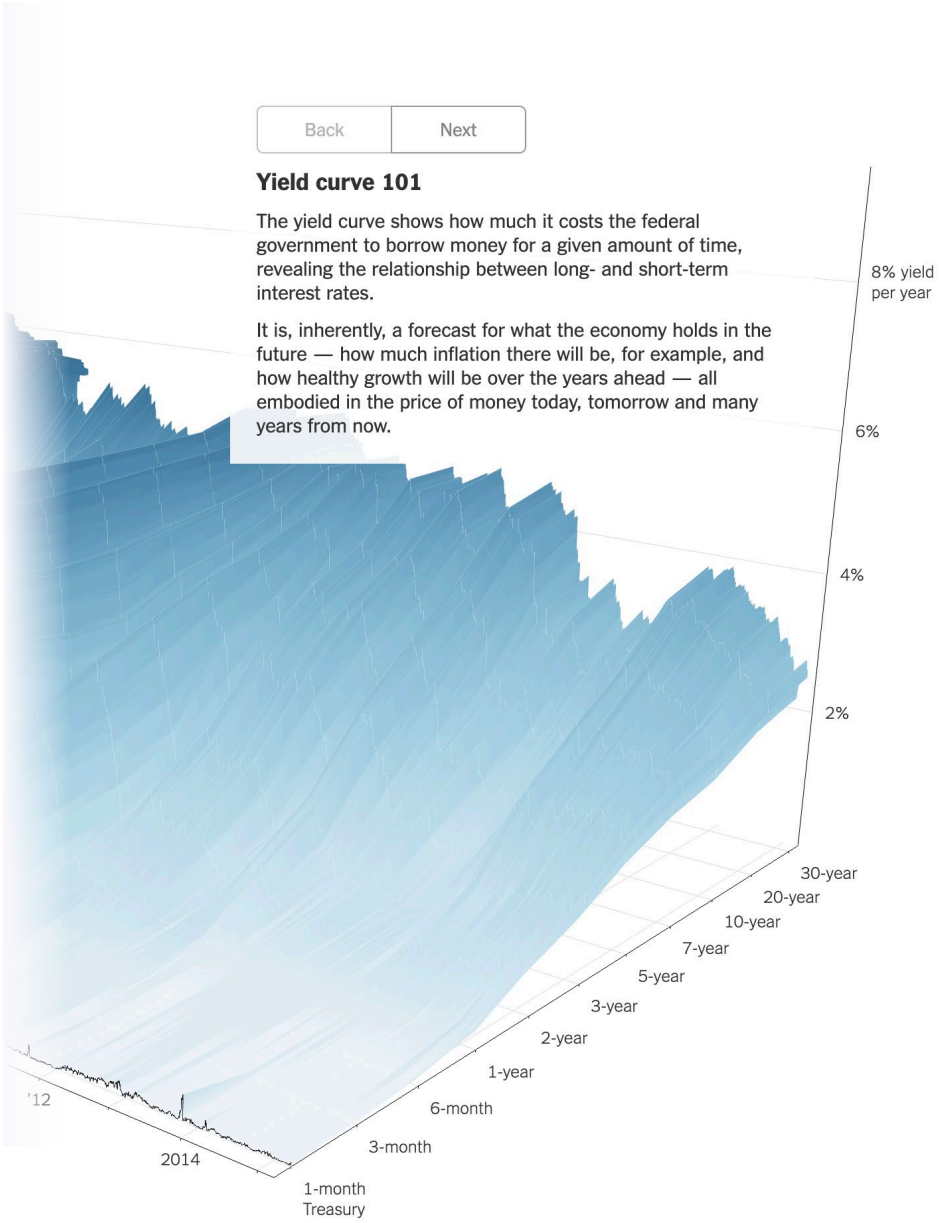
Orthographic projection



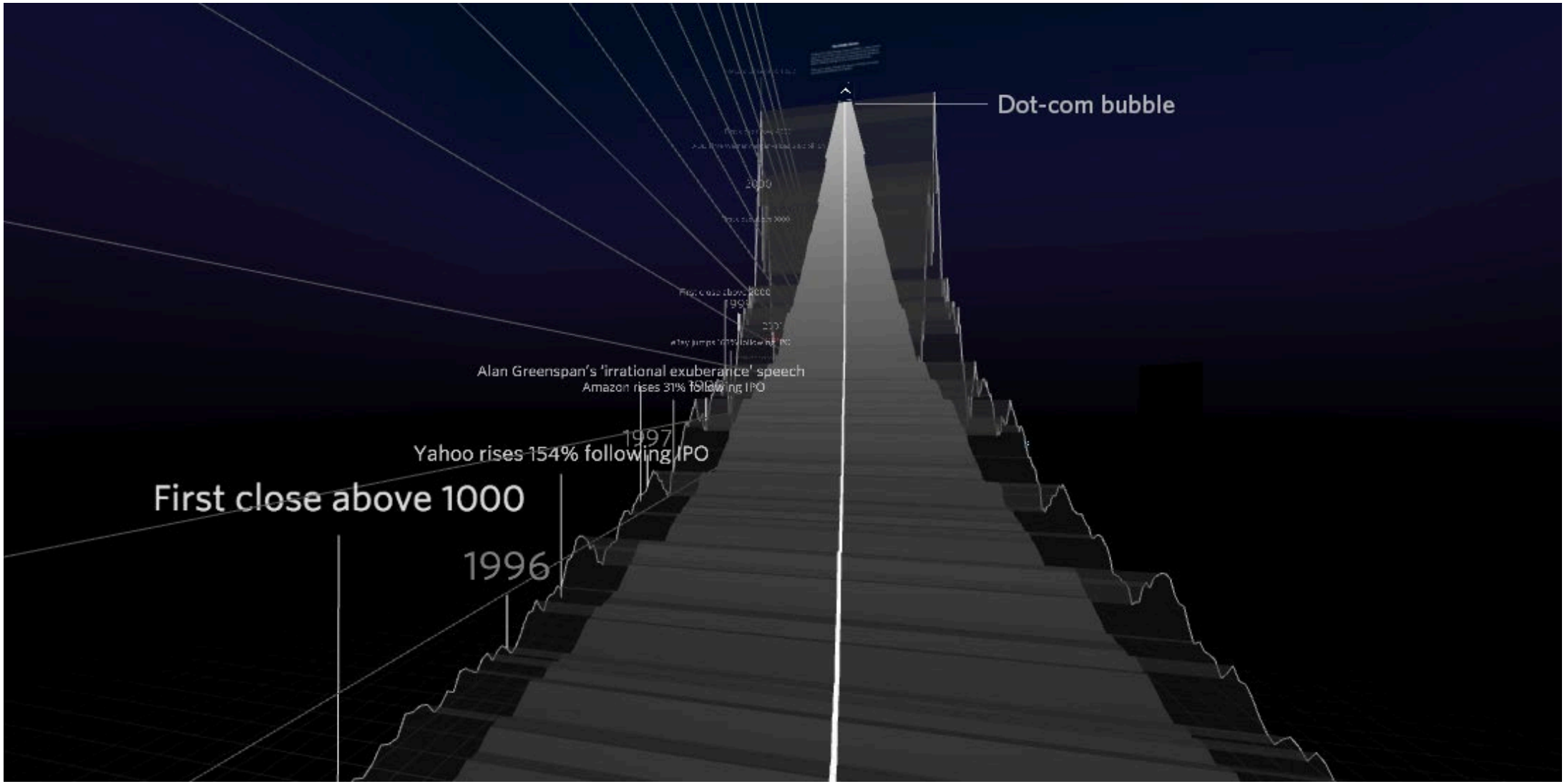
Source: https://www.nytimes.com/interactive/2015/03/19/upshot/3d-yield-curve-economic-growth.html

What Works?

- Interactivity.
- Multiple selected views.
- In-chart explanation.
- Grid lines.



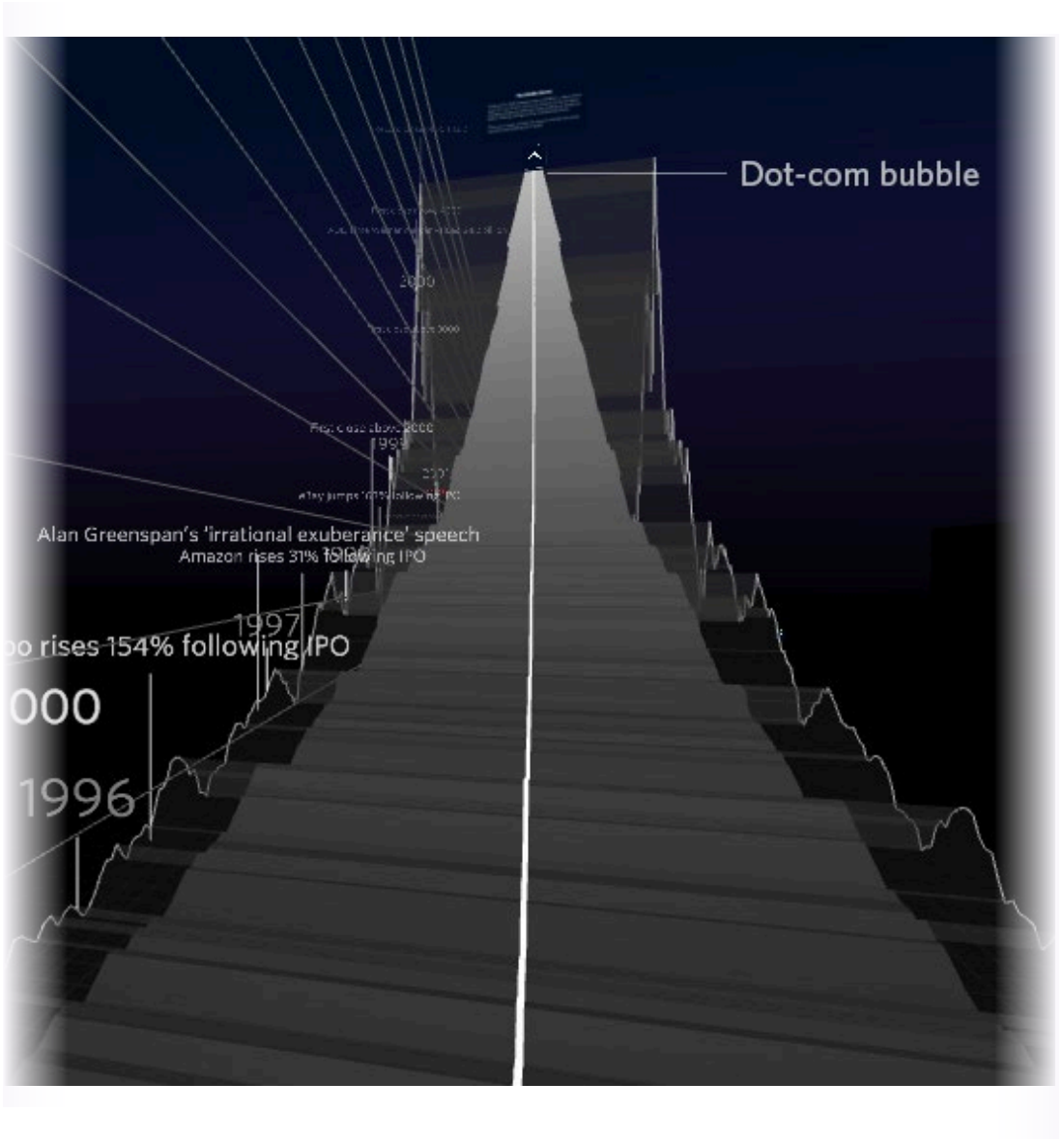
Source: https://www.nytimes.com/interactive/2015/03/19/upshot/3d-yield-curve-economic-growth.html



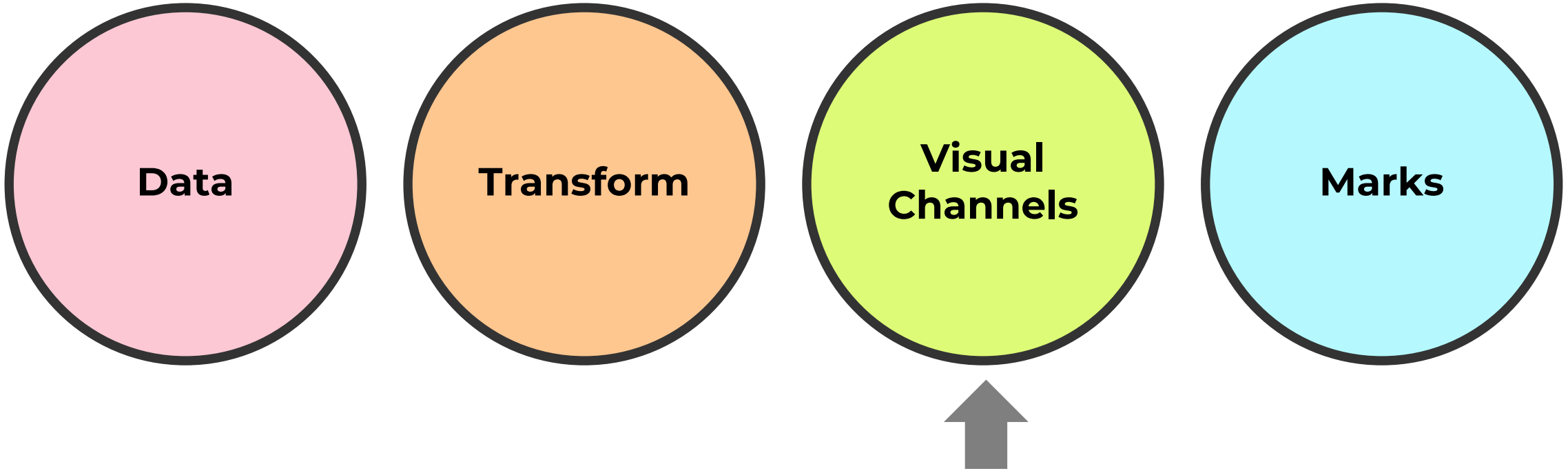
Source: http://graphics.wsj.com/3d-nasdaq/

What Works?

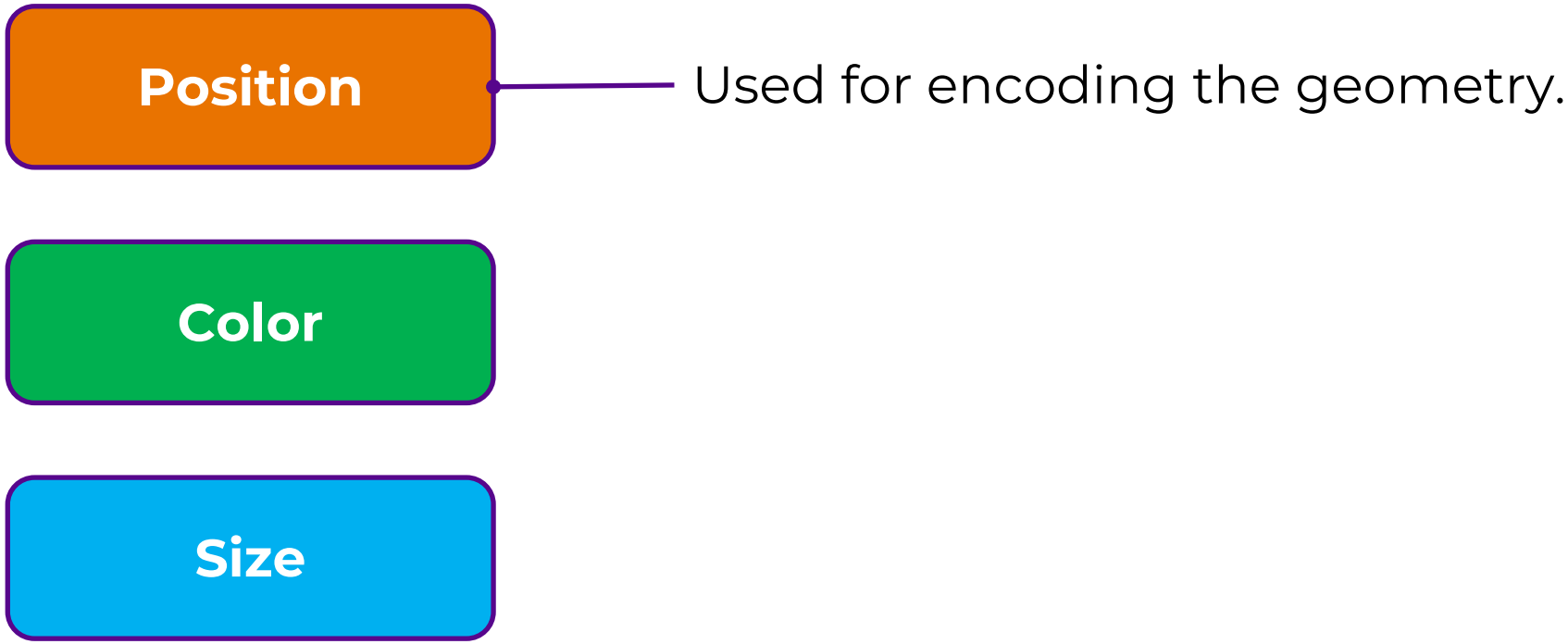
- Novel viewing medium (VR).
- Draw from familiar experience.
- Well annotated.



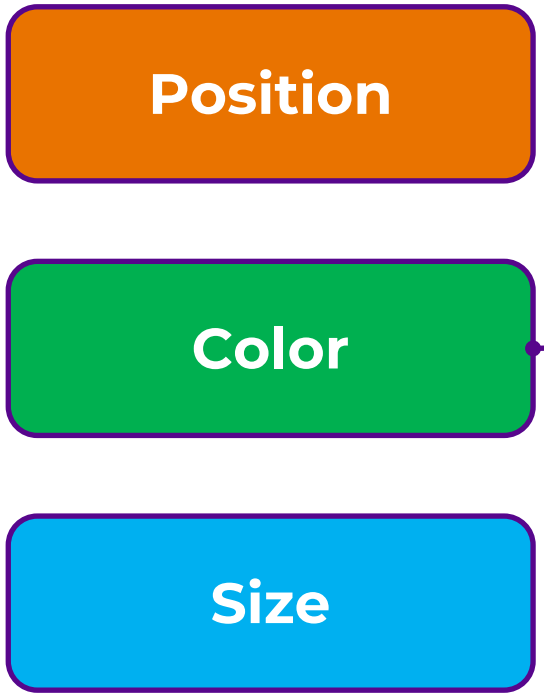
3D Visualization Grammar



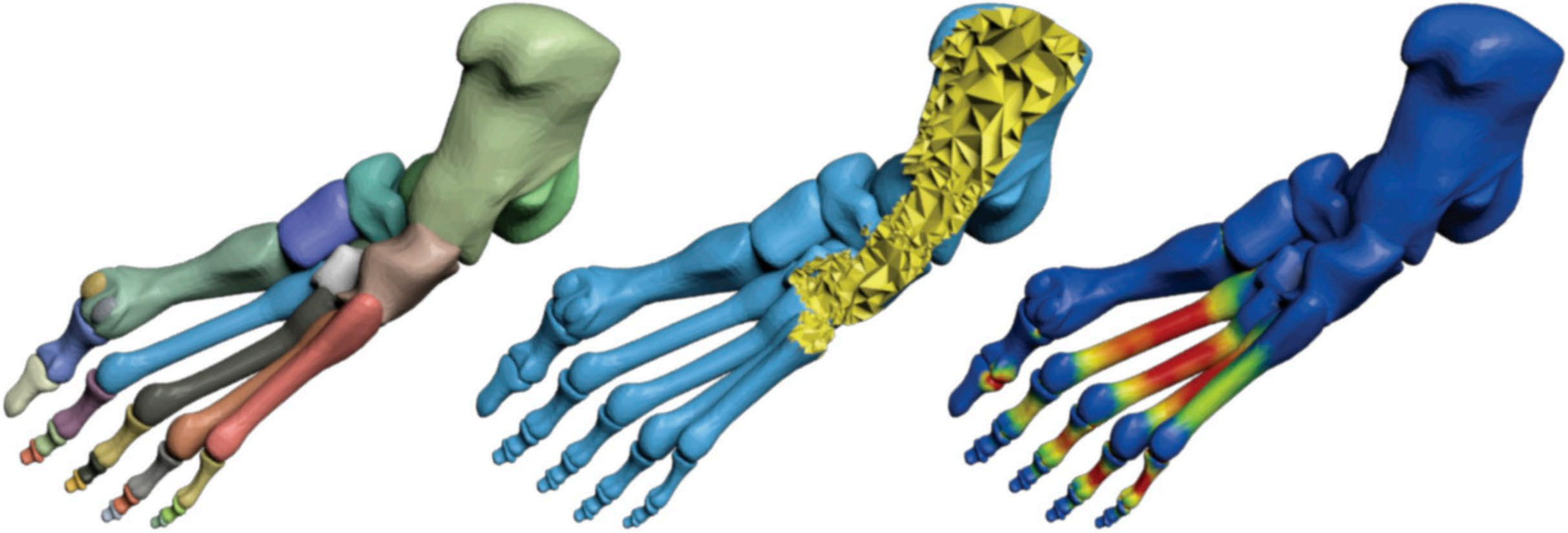
Visual Channels



Visual Channels

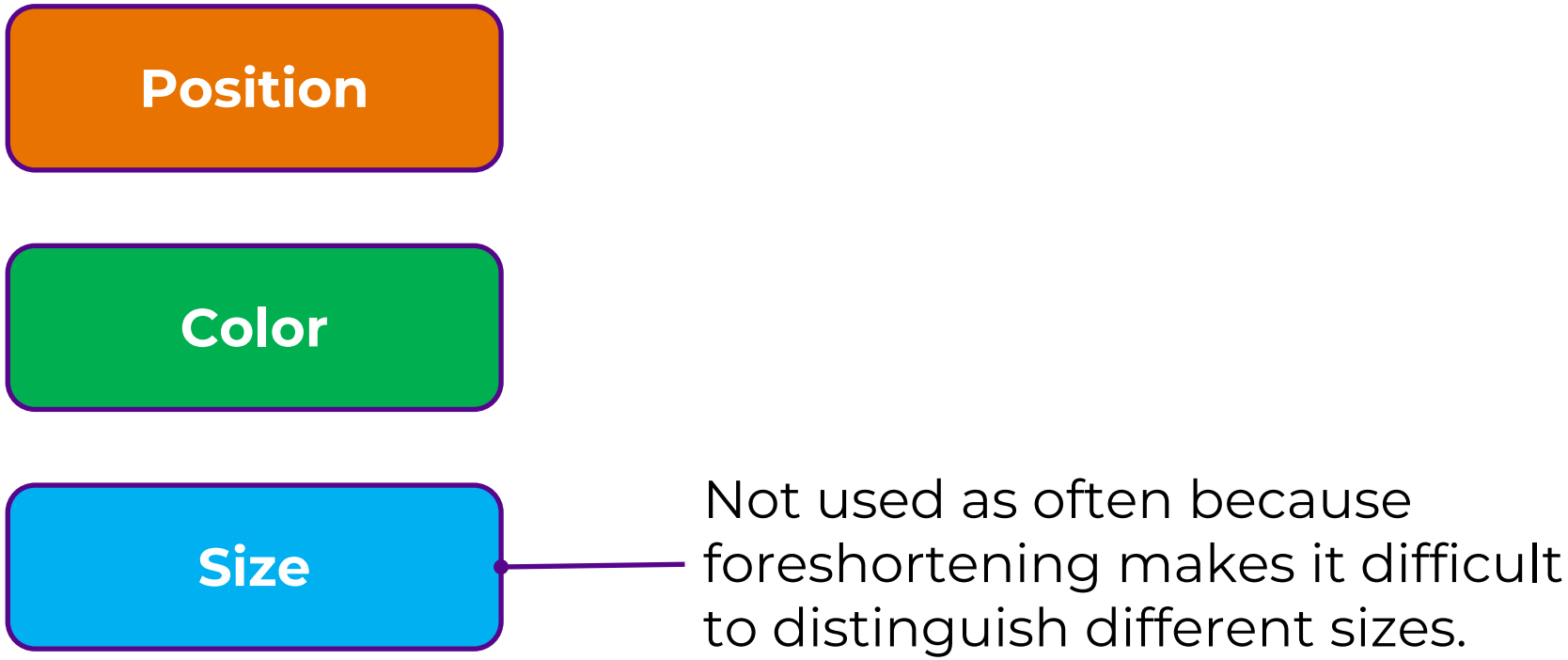


- Encode quantitative/categorical data.
- Lights/shadows may change the color.
- Our eyes/brains are trained to recognize colors under different lighting conditions.



Source: http://www.cs.columbia.edu/cg/mesh-arrangements/mesh-arrangements-for-solid-geometry-siggraph-2016-zhou-et-al.pdf

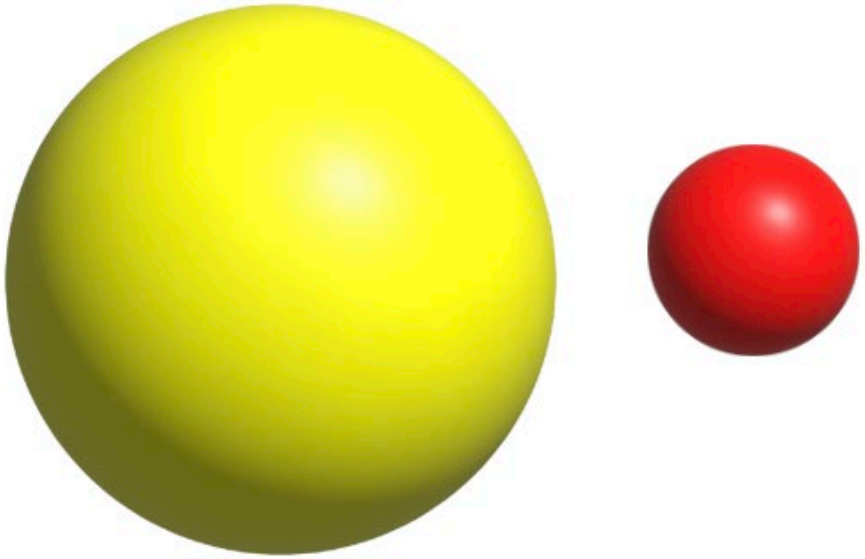
Visual Channels

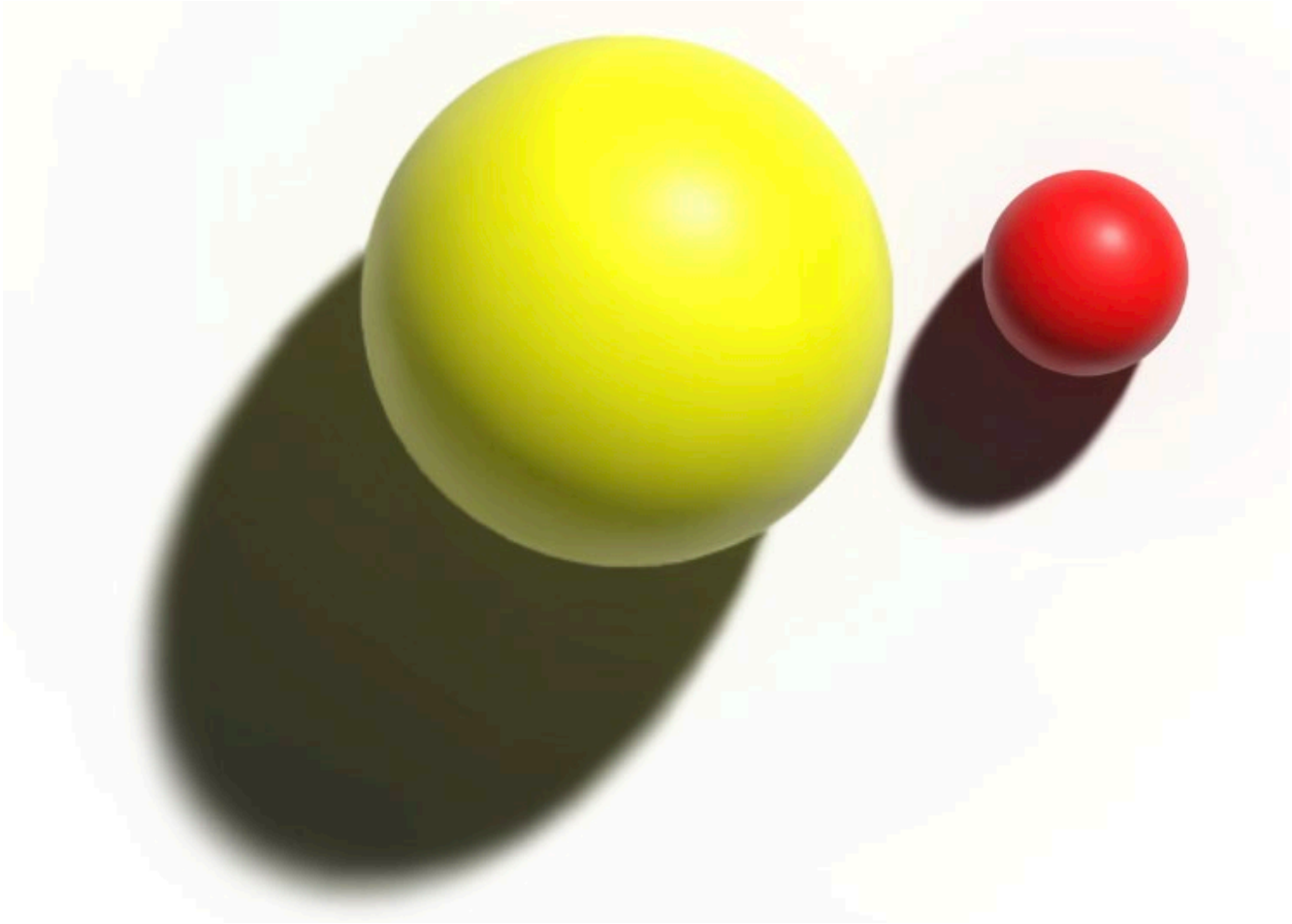


Is the red ball **smaller** than the yellow ball?

Or…

Is the red ball **further away** than the yellow ball?

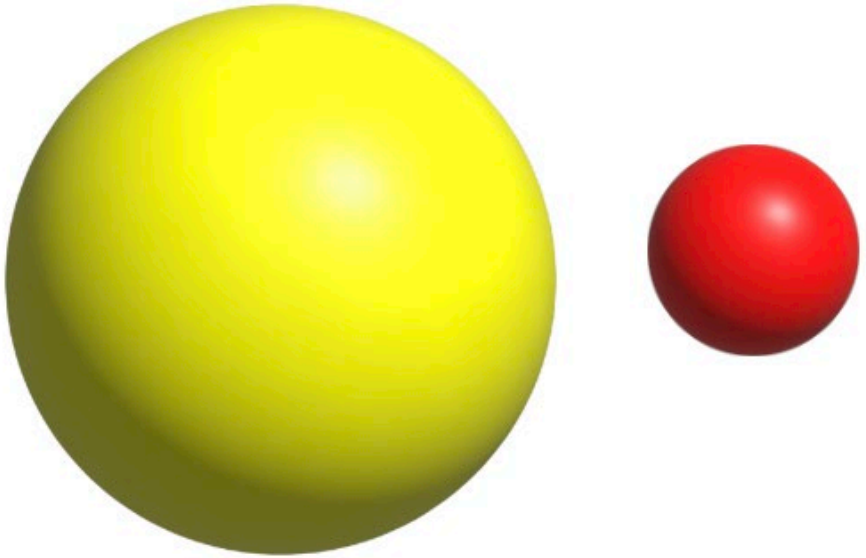


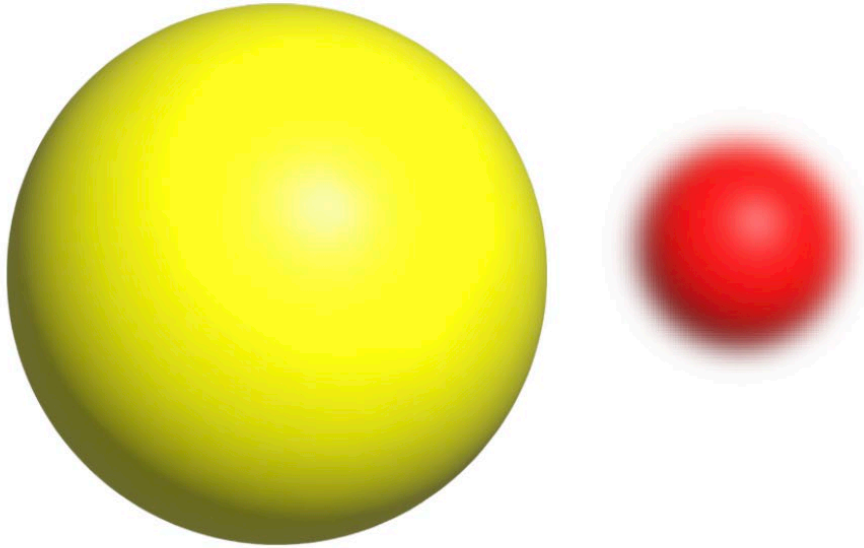


Is the red ball **smaller** than the yellow ball?

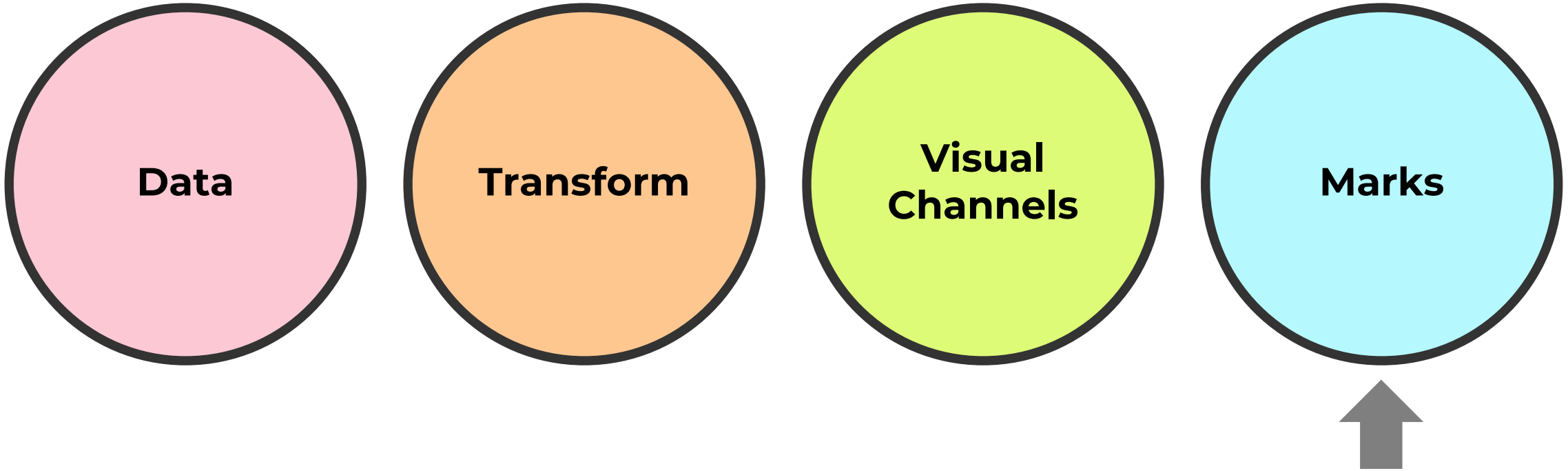
Or...

Is the red ball **further away** than the yellow ball?





3D Visualization Grammar



Marks

- Points
- Curves/lines
- Triangles/quads
- Tetrahedra/Hexahedra