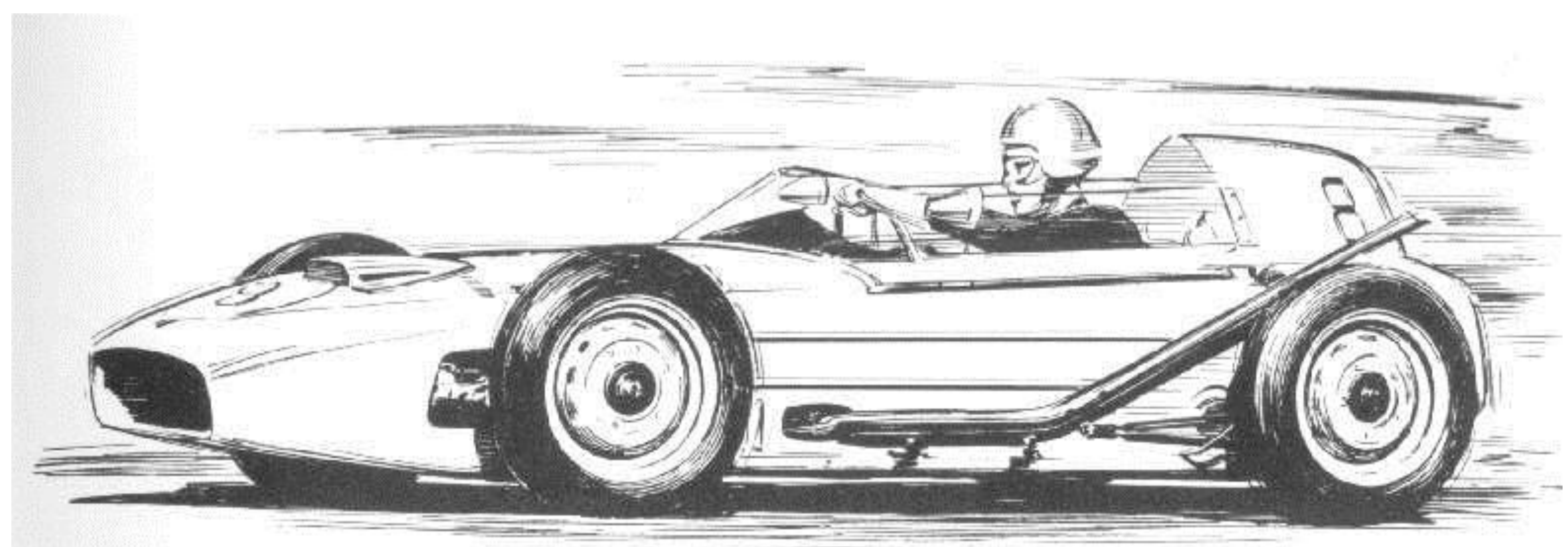
Coordinates and Transformations

MIT ECCS 6.837
Wojciech Matusik

many slides follow Steven Gortler's book

Hierarchical modeling

- **Many coordinate systems:**
  - Camera
  - Static scene
  - car
  - driver
  - arm
  - hand
  - ...



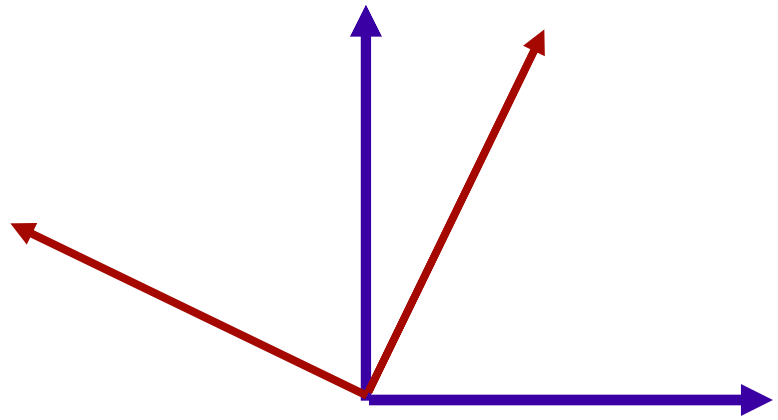
- **Makes it important to understand coordinate systems**

Coordinates

- We are used to represent points with tuples of coordinates such as $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$
- But the tuples are meaningless without a clear coordinate system

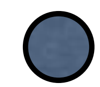
could be this point in the red coordinate system

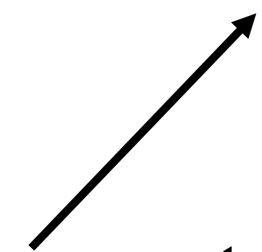
could be this point in the blue coordinate system



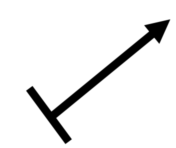
Different objects

- **Points**
  - represent locations





- **Vectors**
  - represent movement, force, displacement from A to B
- **Normals**
  - represent orientation, unit length

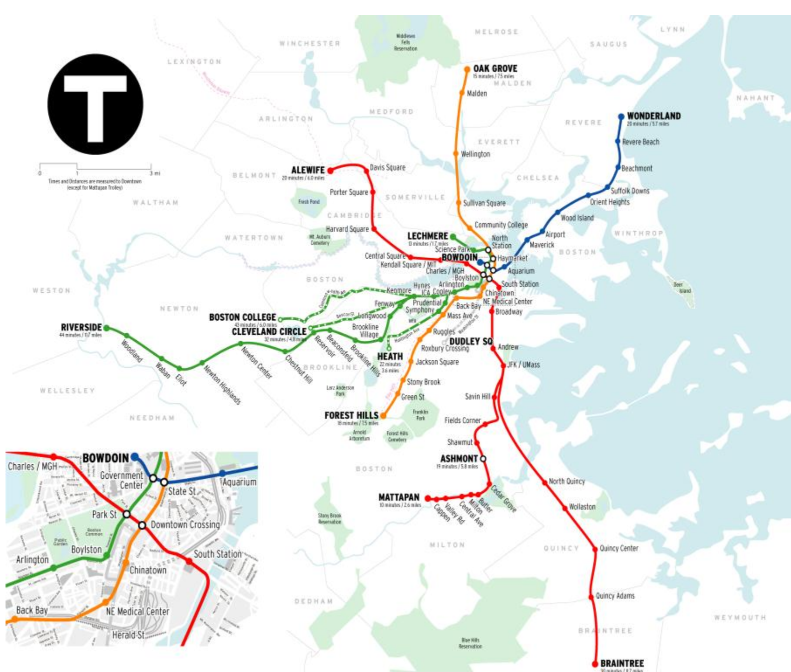


- **Coordinates**
  - numerical representation of the above objects
    in a given coordinate system

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Points & vectors are different

- The 0 vector has a fundamental meaning: no movement, no force
- Why would there be a special 0 point?
- It’s meaningful to add vectors, not points
    - Boston location + NYC location =?

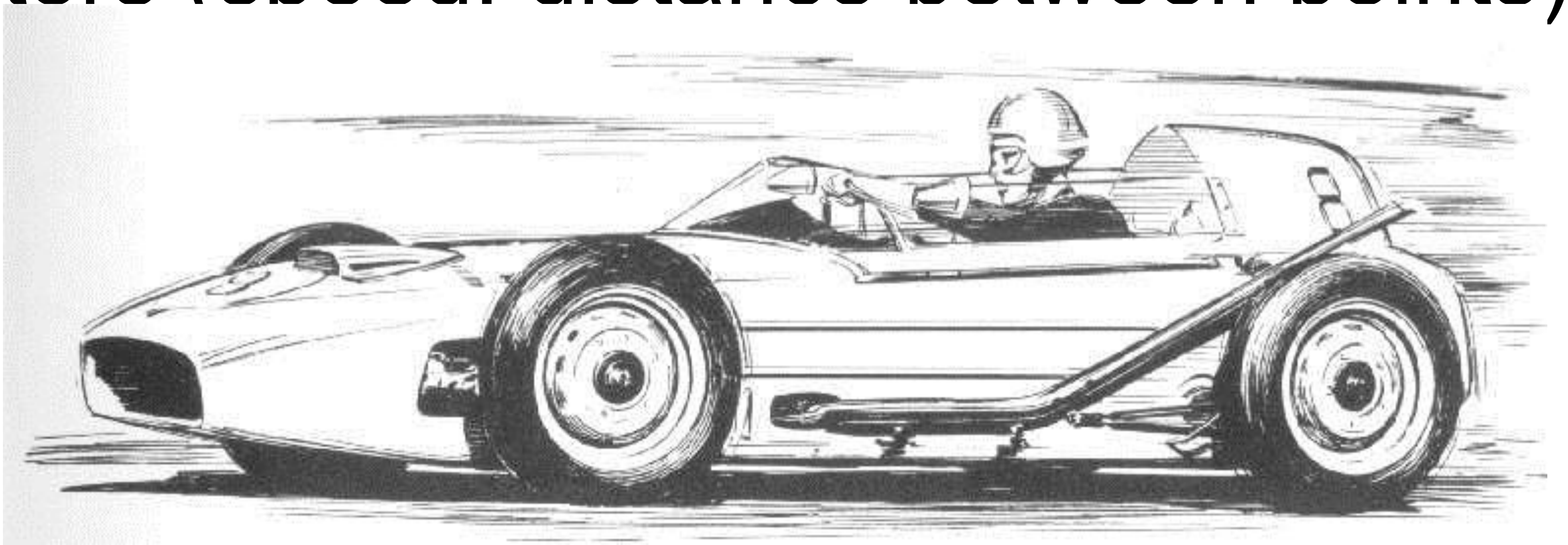


+

=?

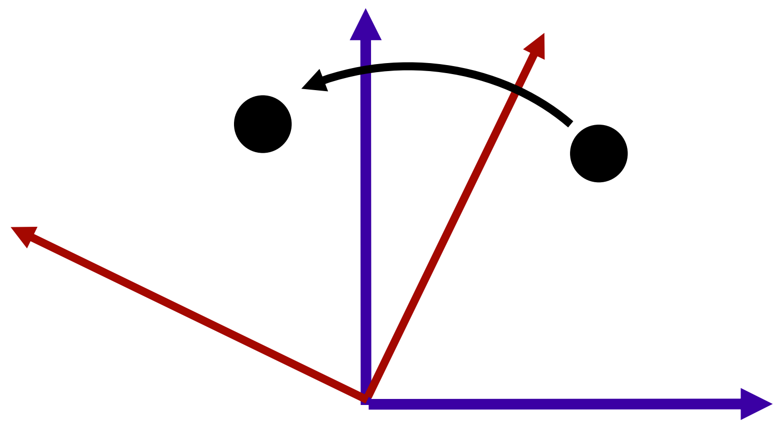
Points & vectors are different

- Moving car
  - points describe location of car elements
  - vectors describe velocity, distance between pairs of points
- If I **translate** the moving car to a different road
  - The points (location) change
  - The vectors (speed, distance between points) don't

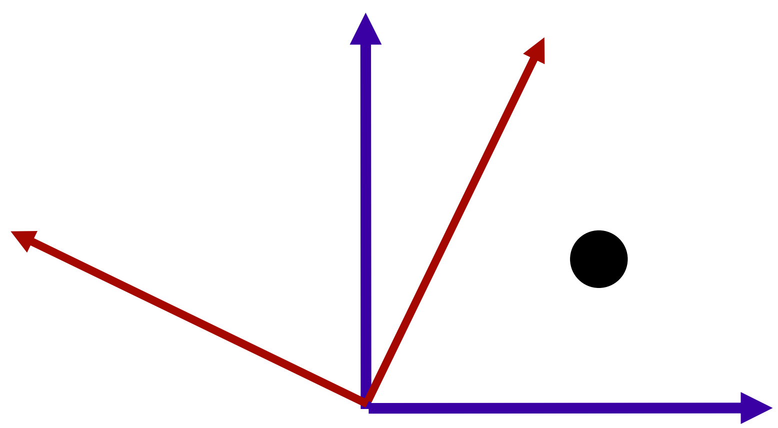


Matrices have two purposes

- (At least for geometry)
- Transform things
  - e.g. rotate the car from facing North to facing East



- Express coordinate system changes
  - e.g. given the driver's location in the coordinate system of the car, express it in the coordinate system of the world



Goals for today

- Make it very explicit what coordinate system is used
- Understand how to change coordinate systems
- Understand how to transform objects
- Understand difference between points, vectors, normals and their coordinates

Questions?

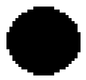
Reference

- This lecture follows the new book by Steven (Shlomo) Gortler from Harvard: Foundations of 3D Computer Graphics

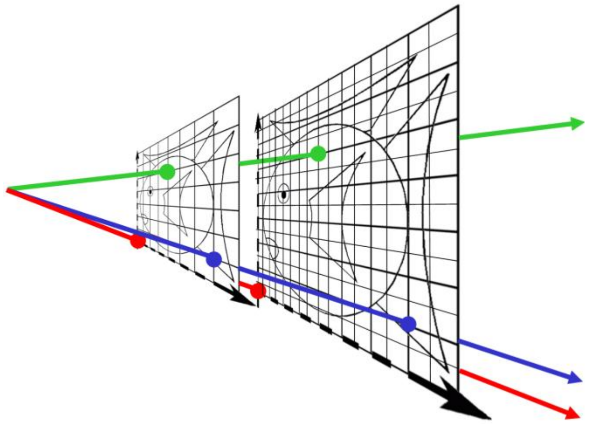
Plan

- Vectors

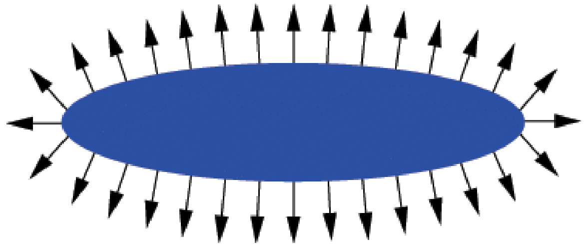
- Points



- Homogeneous coordinates



- Normals (in the next lecture)

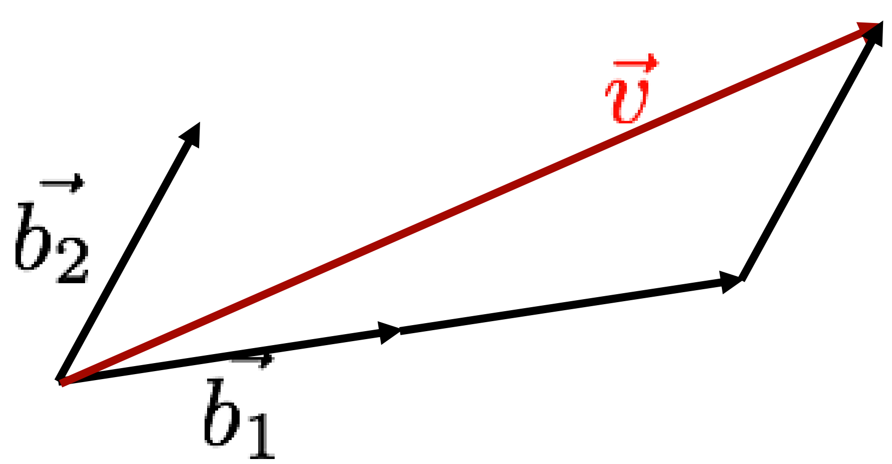


Vectors (linear space)

- Formally, a set of elements equipped with addition and scalar multiplication
  - plus other nice properties
- There is a special element, the zero vector
  - no displacement, no force

Vectors (linear space)

- We can use a ***basis*** to produce all the vectors in the space:
  - Given n basis vectors $\vec{b_i}$ any vector $\vec{v}$ can be written as $$\vec{v} = \sum_i c_i \vec{b_i}$$



here:
$$\vec{v} = 2\vec{b_1} + \vec{b_2}$$

Linear algebra notation

$$\vec{v} = c_1\vec{b_1} + c_2\vec{b_2} + c_3\vec{b_3}$$

- can be written as

$$\begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

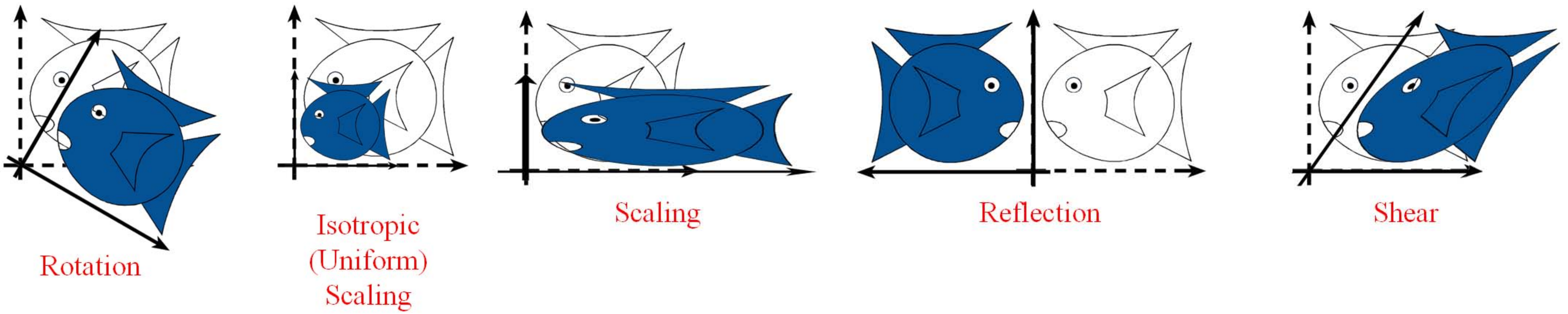
- Nice because it makes the basis (coordinate system) explicit
- Shorthand:

$$\vec{v} = \vec{\mathbf{b}}^t \mathbf{c}$$

- where bold means triplet, t is transpose

Questions?

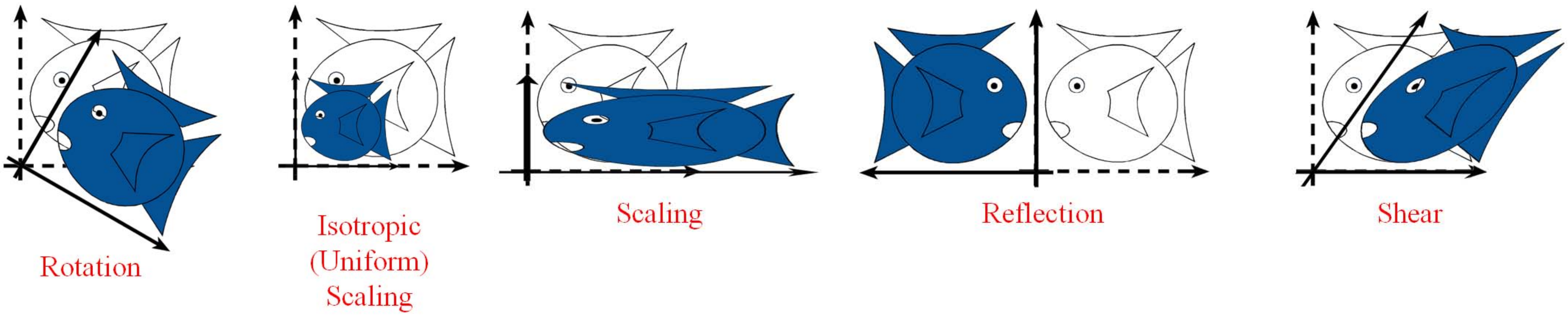
Linear transformation



Courtesy of Prof. Fredo Durand. Used with permission.

- Transformation $\mathcal{L}$ of the vector space

Linear transformation



Courtesy of Prof. Fredo Durand. Used with permission.

- Transformation $\mathcal{L}$ of the vector space so that

$$\mathcal{L}(\vec{v} + \vec{u}) = \mathcal{L}(\vec{v}) + \mathcal{L}(\vec{u})$$

$$\mathcal{L}(\alpha\vec{v}) = \alpha\mathcal{L}(\vec{v})$$

- Note that it implies $\mathcal{L}(\vec{0}) = \vec{0}$
- Notation $\vec{v} \Rightarrow \mathcal{L}(\vec{v})$ for transformations

Matrix notation

- Linearity implies

$$\mathcal{L}(\vec{v}) = \mathcal{L}\left(\sum_i c_i \vec{b_i}\right) = \qquad ?$$

Matrix notation

- Linearity implies

$$\mathcal{L}(\vec{v}) = \mathcal{L}\left(\sum_i c_i \vec{b_i}\right) = \sum_i c_i \mathcal{L}(\vec{b_i})$$

- i.e. we only need to know the basis transformation
- or in algebra notation

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \mathcal{L}(\vec{b}_1) & \mathcal{L}(\vec{b}_2) & \mathcal{L}(\vec{b}_3) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Algebra notation

- The $\mathcal{L}(\vec{b_i})$ are also vectors of the space
- They can be expressed in the basis

...

Algebra notation

- The $\mathcal{L}(\vec{b_i})$ are also vectors of the space
- They can be expressed in the basis for example:

$$\mathcal{L}(\vec{b}_1) = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} \\ M_{2,1} \\ M_{3,1} \end{bmatrix}$$

- which gives us

$$\begin{bmatrix} \mathcal{L}(\vec{b}_1) & \mathcal{L}(\vec{b}_2) & \mathcal{L}(\vec{b}_3) \end{bmatrix} = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$

Recap, matrix notation

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- Given the coordinates ***c*** in basis $\vec{\mathrm{b}}$
  the transformed vector has coordinates Mc in $\vec{\mathrm{b}}$

Why do we care

- We like linear algebra
- It’s always good to get back to an abstraction that we know and for which smarter people have developed a lot of tools
- But we also need to keep track of what basis/coordinate system we use

Questions?

Change of basis

- Critical in computer graphics
  - From world to car to arm to hand coordinate system
  - From Bezier splines to B splines and back

- problem with basis change:
  you never remember which is M or M^{-1}
  it's hard to keep track of where you are

Change of basis

- Assume we have two bases $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$
- And we have the coordinates of $\vec{\mathbf{a}}$ in $\vec{\mathbf{b}}$
- e.g.

$$\vec{a_1} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} \end{bmatrix} \begin{bmatrix} M_{11} \\ M_{21} \\ M_{31} \end{bmatrix}$$

- i.e.

$$\vec{\mathbf{a}}^t = \vec{\mathbf{b}}^t M$$

- which implies $\vec{\mathbf{a}}^t M^{-1} = \vec{\mathbf{b}}^t$

Change of basis

- We have $\vec{\mathbf{a}}^t = \vec{\mathbf{b}}^t M$ & $\vec{\mathbf{a}}^t M^{-1} = \vec{\mathbf{b}}^t$
- Given the coordinate of $\vec{v}$ in $\vec{\mathbf{b}}$: $\vec{v} = \vec{\mathbf{b}}^t \mathbf{c}$
- What are the coordinates in $\vec{\mathbf{a}}$?

Change of basis

- We have $\vec{\mathbf{a}}^t = \vec{\mathbf{b}}^t M$ & $\vec{\mathbf{a}}^t M^{-1} = \vec{\mathbf{b}}^t$
- Given the coordinate of $\vec{v}$ in $\vec{\mathbf{b}}$: $\vec{v} = \vec{\mathbf{b}}^t \mathbf{c}$
- Replace $\vec{\mathbf{b}}$ by its expression in $\vec{\mathbf{a}}$

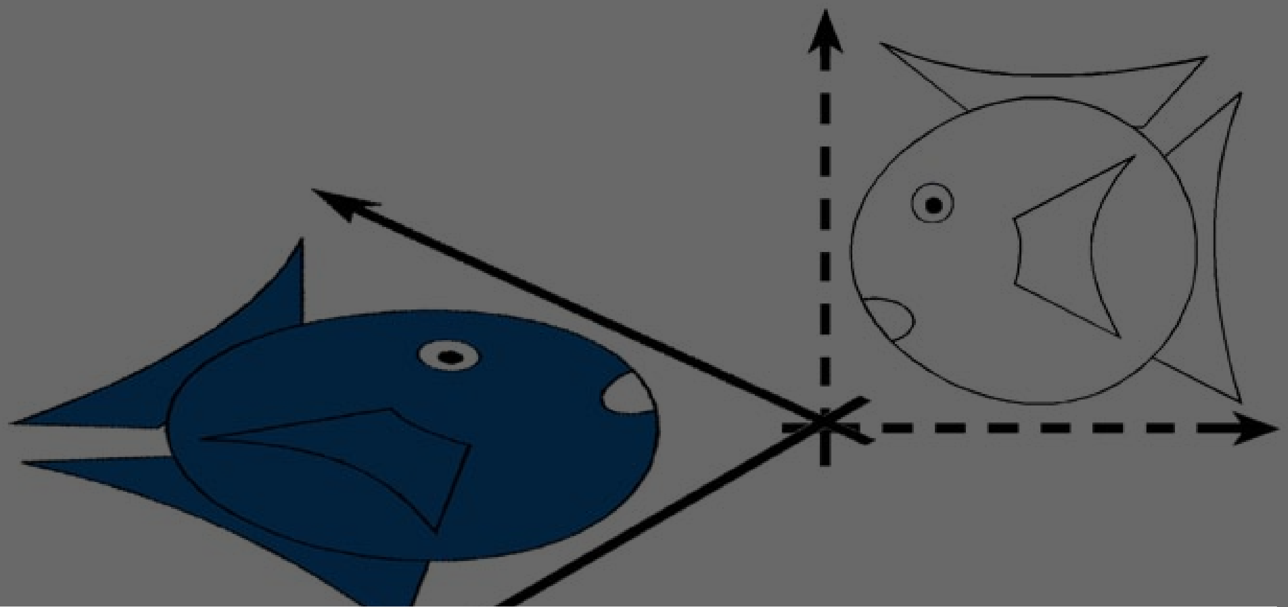
$$\vec{v} = \vec{\mathbf{a}}^t M^{-1} \mathbf{c}$$

- $\vec{v}$ has coordinates $M^{-1}\mathbf{c}$ in $\vec{\mathbf{a}}$
- Note how we keep track of the coordinate system by having the basis on the left

Questions?

Linear Transformations

- $L(p + q) = L(p) + L(q)$
- $L(ap) = a\,L(p)$



Translation is not linear:

$f(p) = p+t$

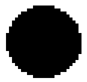
$f(ap) = ap+t \neq a(p+t) = a\,f(p)$

$f(p+q) = p+q+t \neq (p+t)+(q+t) = f(p) + f(q)$

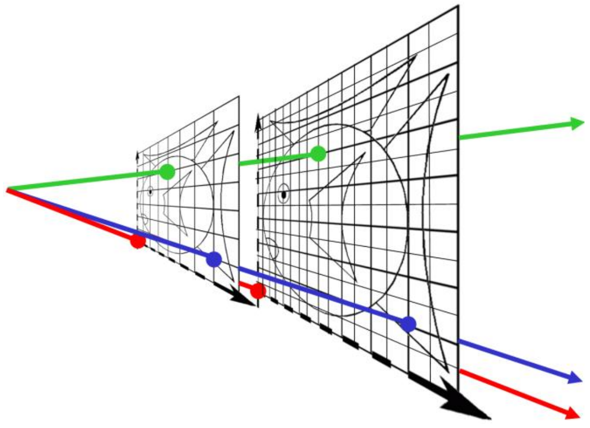
Plan

- Vectors

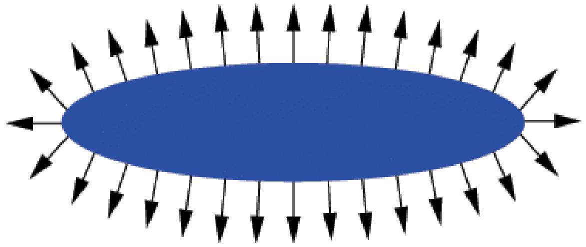
- **Points**



- Homogenous coordinates



- Normals



Points vs. Vectors

- A point is a location
- A vector is a motion between two points
- Adding vectors is meaningful
    - going 3km North + 4km East = going 5km North-East
- Adding points is not meaningful
    - Boston location + New York location = ?
- Multiplying a point by a scalar?
- The zero vector is meaningful (no movement)
- Zero point ?

Affine space

- Points are elements of an affine space
- We denote them with a tilde $\tilde{p}$

- Affine spaces are an extension of vector spaces

Point-vector operations

- Subtracting points gives a vector

$$\tilde{p} - \tilde{q} = \vec{v}$$

- Adding a vector to a point gives a point

$$\tilde{q} + \vec{v} = \tilde{p}$$

Frames

- A frame is an origin $\tilde{o}$ plus a basis $\vec{\mathrm{b}}$
- We can obtain any point in the space by adding a vector to the origin

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b}_i$$

- using the coordinates ***c*** of the vector in $\vec{\mathrm{b}}$

Algebra notation

- We like matrix-vector expressions
- We want to keep track of the frame
- We're going to cheat a little for elegance and decide that 1 times a point is the point

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b}_i \quad = \quad \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} = \vec{\mathbf{f}}^t \mathbf{c}$$

- $\tilde{p}$ is represented in $\vec{\mathbf{f}}$ by 4 coordinate, where the extra dummy coordinate is always 1 (for now)

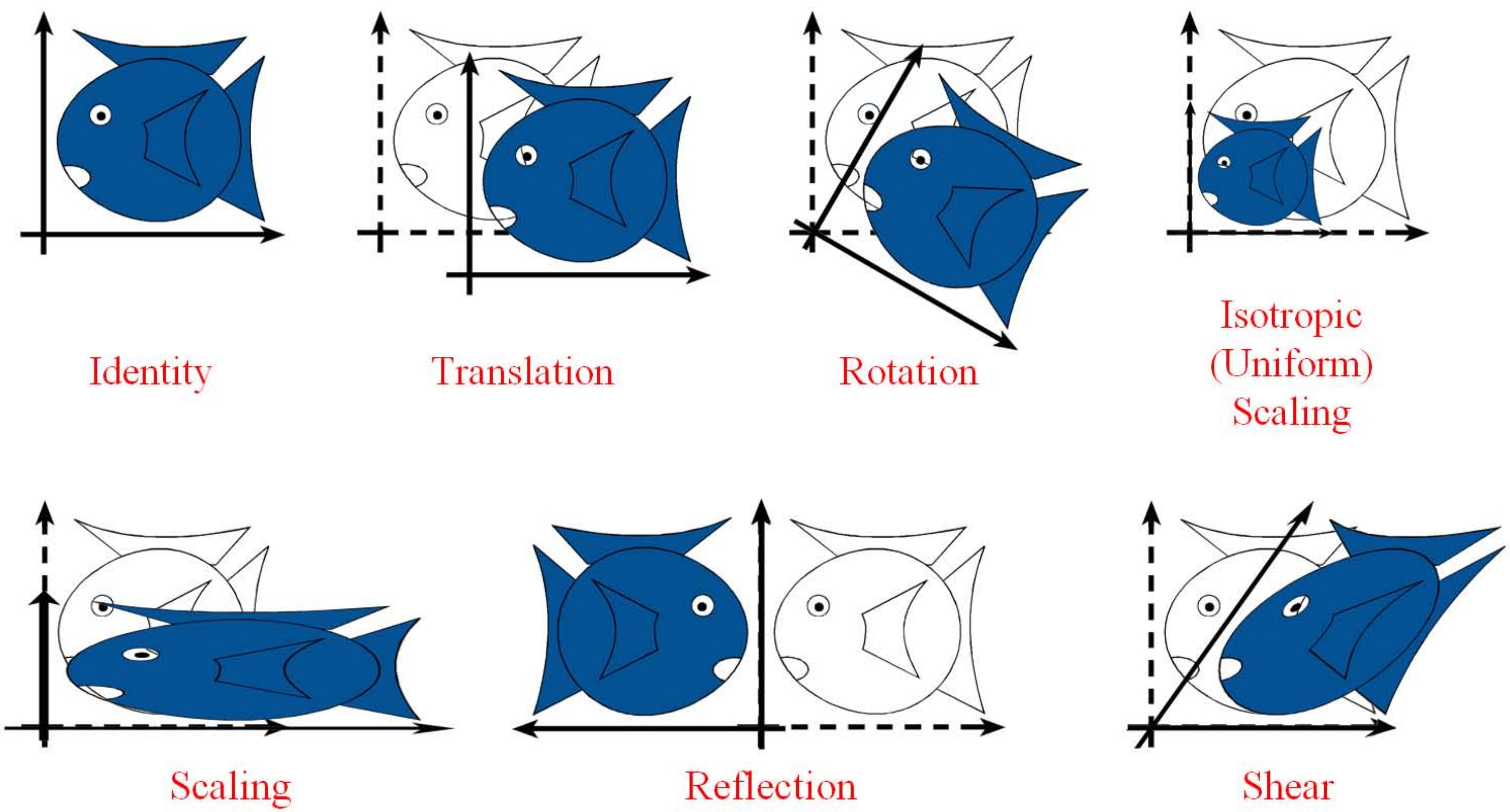
Recap

- Vectors can be expressed in a basis
  - Keep track of basis with left notation $\vec{v} = \vec{\mathbf{b}}^t \mathbf{c}$
  - Change basis $\vec{v} = \vec{\mathbf{a}}^t M^{-1} \mathbf{c}$
- Points can be expressed in a frame (origin+basis)
  - Keep track of frame with left notation
  - adds a dummy 4th coordinate always 1

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b}_i = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} = \vec{\mathbf{f}}^t \mathbf{c}$$

Affine transformations

- Include all linear transformations
  - Applied to the vector basis
- Plus translation



Matrix notation

- We know how to transform the vector basis

$$\left[\begin{array}{ccc} \mathcal{L}(\vec{b}_1) & \mathcal{L}(\vec{b}_2) & \mathcal{L}(\vec{b}_3) \end{array}\right] = \left[\begin{array}{ccc} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{array}\right] \left[\begin{array}{ccc} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{array}\right]$$

- We will soon add translation by a vector $\vec{t}$

$$\tilde{p} \Rightarrow \tilde{p} + \vec{t}$$

Linear component

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

$$\Rightarrow$$

$$\tilde{o} + \sum_i c_i \mathcal{L}(\vec{b_i}) = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & 0 \\ M_{21} & M_{22} & M_{23} & 0 \\ M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

- Note how we leave the fourth component alone

Translation component

$$\tilde{p} \Rightarrow \tilde{p} + \vec{t}$$

- Express translation vector t in the basis

$$\vec{t} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} \end{bmatrix} \begin{bmatrix} M_{14} \\ M_{24} \\ M_{34} \end{bmatrix}$$

Translation

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

$$\Rightarrow$$

$$\tilde{o} + \vec{t} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & M_{14} \\ 0 & 1 & 0 & M_{24} \\ 0 & 0 & 1 & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

Full affine expression

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

$$\Rightarrow$$

$$\tilde{o} + \vec{t} + \sum_i c_i \mathcal{L}(\vec{b_i}) = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

Which tells us both how to get a new frame ftM
or how to get the coordinates Mc after transformation

Questions?

More notation properties

- If the fourth coordinate is zero, we get a vector
- Subtracting two points:

$$\tilde{p} = \vec{f}^t \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} \qquad \tilde{p'} = \vec{f}^t \begin{bmatrix} c'_1 \\ c'_2 \\ c'_3 \\ 1 \end{bmatrix}$$

- Gives us $\tilde{p} - \tilde{p'} = \vec{f}^t \begin{bmatrix} c_1 - c'_1 \\ c_2 - c'_2 \\ c_3 - c'_3 \\ 0 \end{bmatrix}$

  a vector (last coordinate = 0)

More notation properties

- Adding a point

$$\tilde{p} = \vec{f}^t \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

to a vector

$$\vec{v} = \vec{f}^t \begin{bmatrix} c_1' \\ c_2' \\ c_3' \\ 0 \end{bmatrix}$$

- Gives us

$$\tilde{p} + \vec{v} = \vec{f}^t \begin{bmatrix} c_1 + c_1' \\ c_2 + c_2' \\ c_3 + c_3' \\ 1 \end{bmatrix}$$

a point (4th coordinate=1)

More notation properties

- vectors are not affected by the translation part

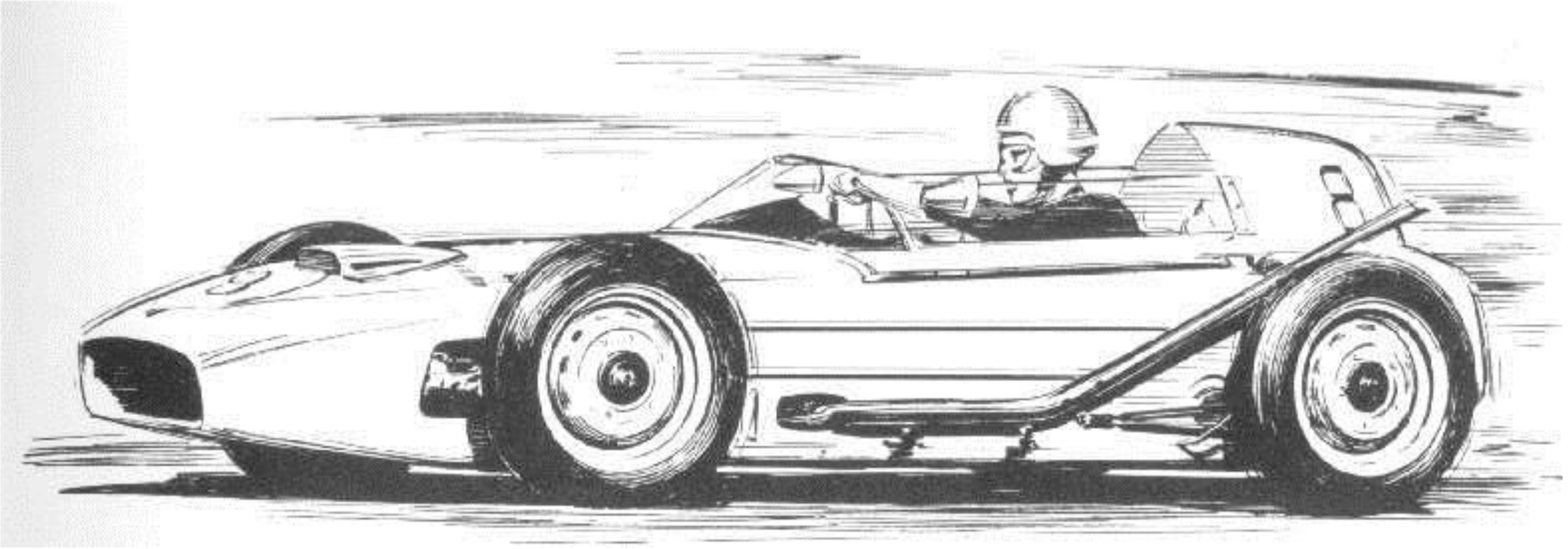
$$\begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 0 \end{bmatrix}$$

- because their 4th coordinate is 0
- If I rotate my moving car in the world, I want its motion to rotate
- If I translate it, motion should be unaffected

Questions?

Frames & hierarchical modeling

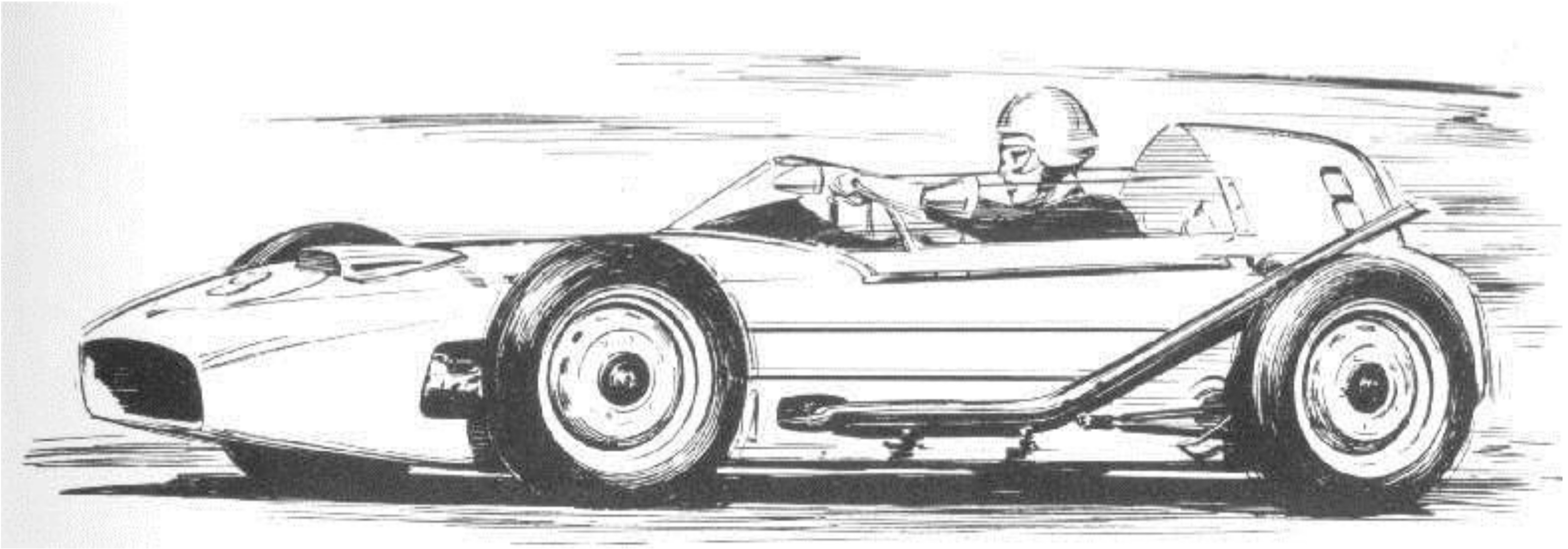
- Many coordinate systems (frames):
  - Camera
  - Static scene
  - car
  - driver
  - arm
  - hand
  - ...



- Need to understand nested transformations

Frames & hierarchical modeling

- Example: what if I rotate the wheel of the moving car:
- frame 1: world
- frame 2: car
- transformation: rotation



Frames & transformations

- Transformation S wrt car frame f

$$\tilde{p} = \vec{\mathbf{f}}^t \mathbf{c} \Rightarrow \vec{\mathbf{f}}^t S \mathbf{c}$$

- how is the world frame a affected by this?
- *we have* $\vec{a}^t = \vec{f}^t A \quad \vec{\mathbf{f}}^t = \vec{\mathbf{a}}^t A^{-1}$
- *which gives* $\vec{a}^t A^{-1} \Rightarrow \vec{a}^t A^{-1} S$

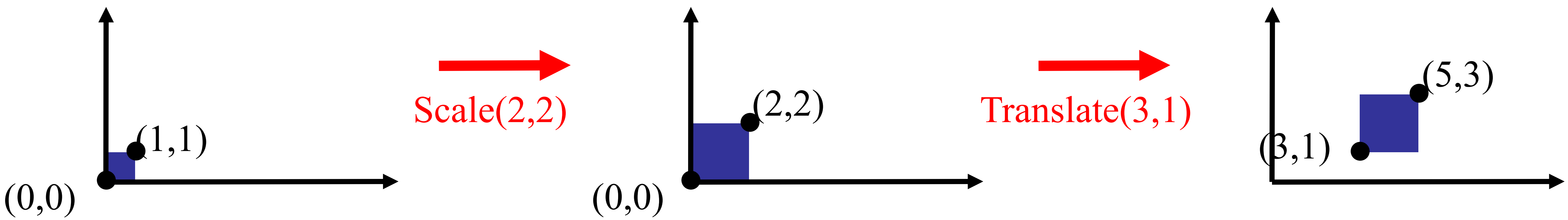
$$\vec{a}^t \Rightarrow \vec{a}^t A^{-1} S A$$

- *i.e. the transformation in a is A-1SA*
- *i.e., from right to left, A takes us from a to f, then we apply S, then we go back to a with A-1*

Questions?

How are transforms combined?

Scale then Translate



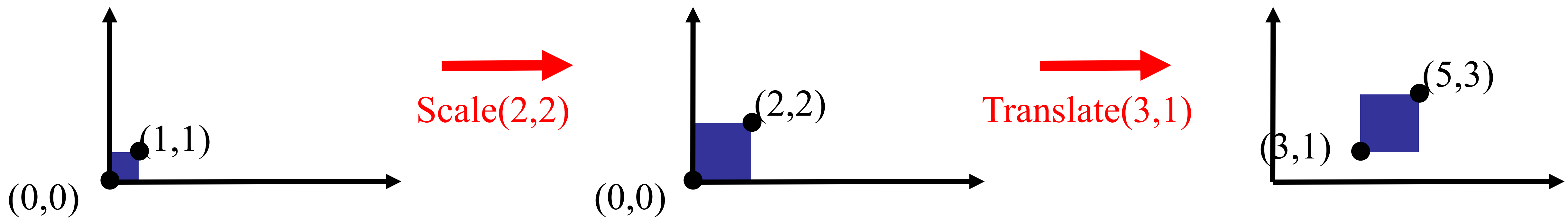
Use matrix multiplication: p' = T (S p) = TS p

$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

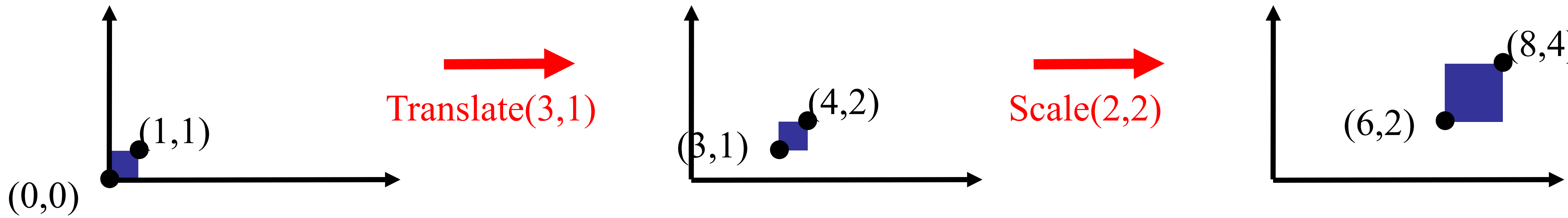
Caution: matrix multiplication is NOT commutative!

Non-commutative Composition

Scale then Translate: $p' = T(Sp) = TSp$



Translate then Scale: $p' = S(Tp) = STp$



Non-commutative Composition

Scale then Translate: $p' = T(S\,p) = TS\,p$

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

Translate then Scale: $p' = S(T\,p) = ST\,p$

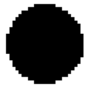
$$ST = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Questions?

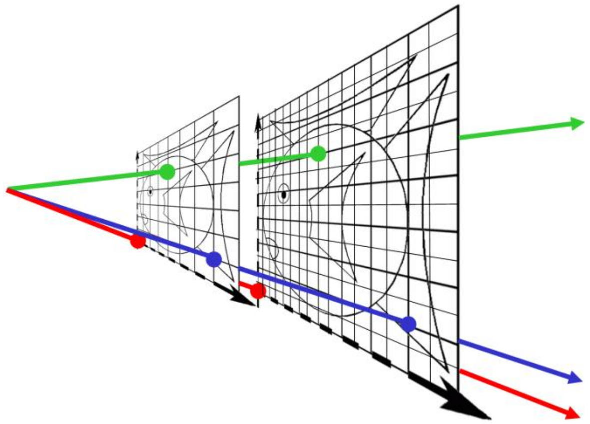
Plan

- Vectors

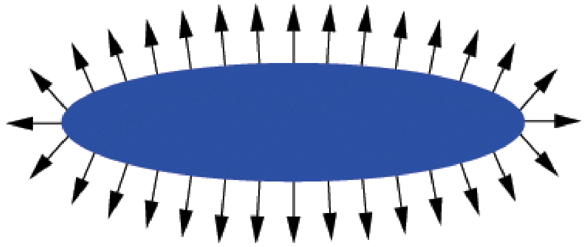
- Points



- **Homogenous coordinates**



- Normals



Forward reference and eye

- The fourth coordinate is useful for perspective projection
- Called homogenous coordinates

Homogeneous Coordinates

- Add an extra dimension (same as frames)
  - in 2D, we use 3-vectors and 3 x 3 matrices
  - In 3D, we use 4-vectors and 4 x 4 matrices
- The extra coordinate is now an **arbitrary** value, w
  - You can think of it as “scale,” or “weight”
  - For all transformations

  except perspective, you can just set *w*=1 and not worry about it

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Projective Equivalence

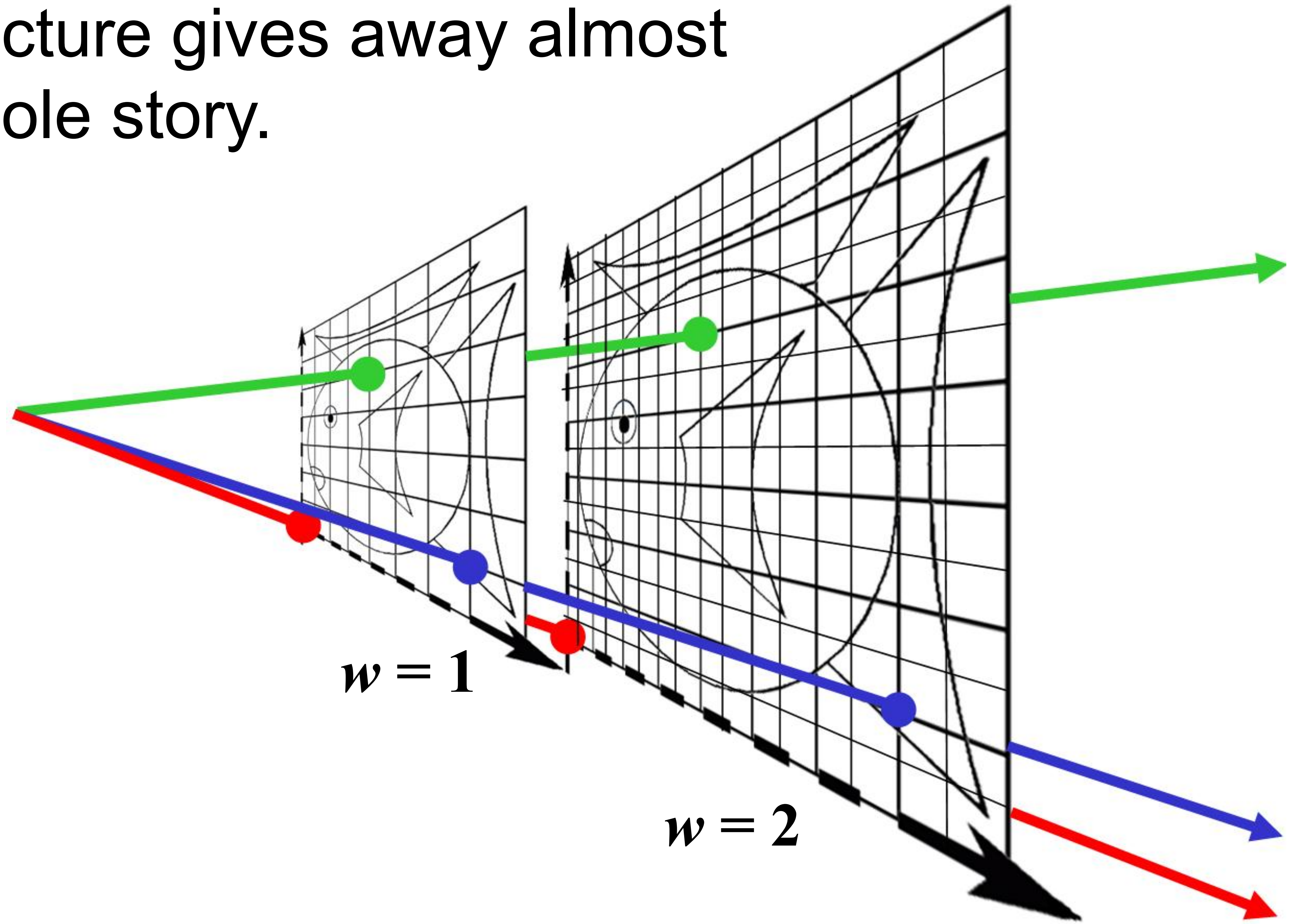
- All non-zero scalar multiples of a point are considered identical
- to get the equivalent Euclidean point, divide by *w*

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az \\ aw \end{bmatrix} \overset{w\ !=0}{=} \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

$$a\ \ != 0$$

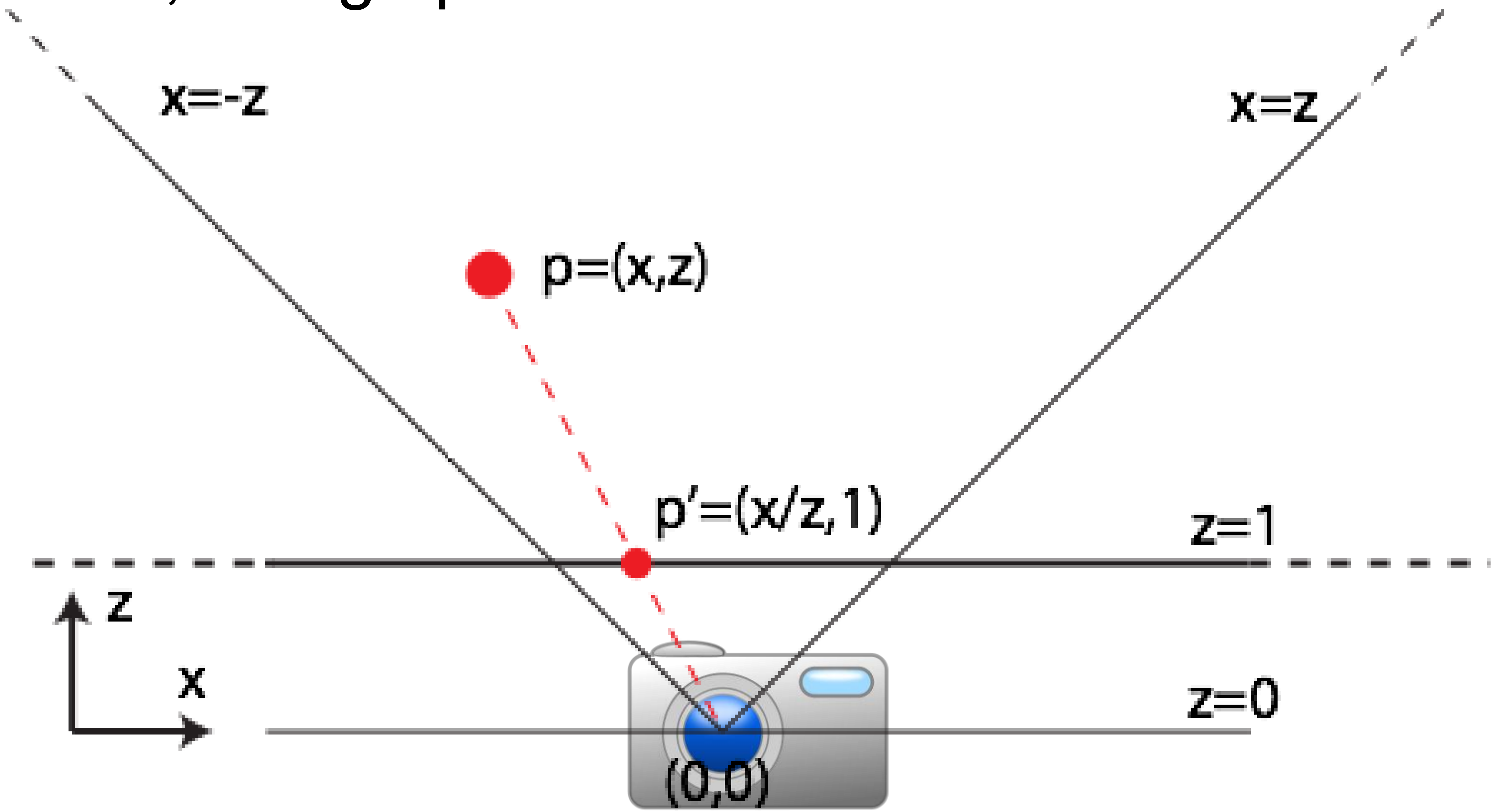
Why bother with extra coord?

- This picture gives away almost the whole story.



Perspective in 2D

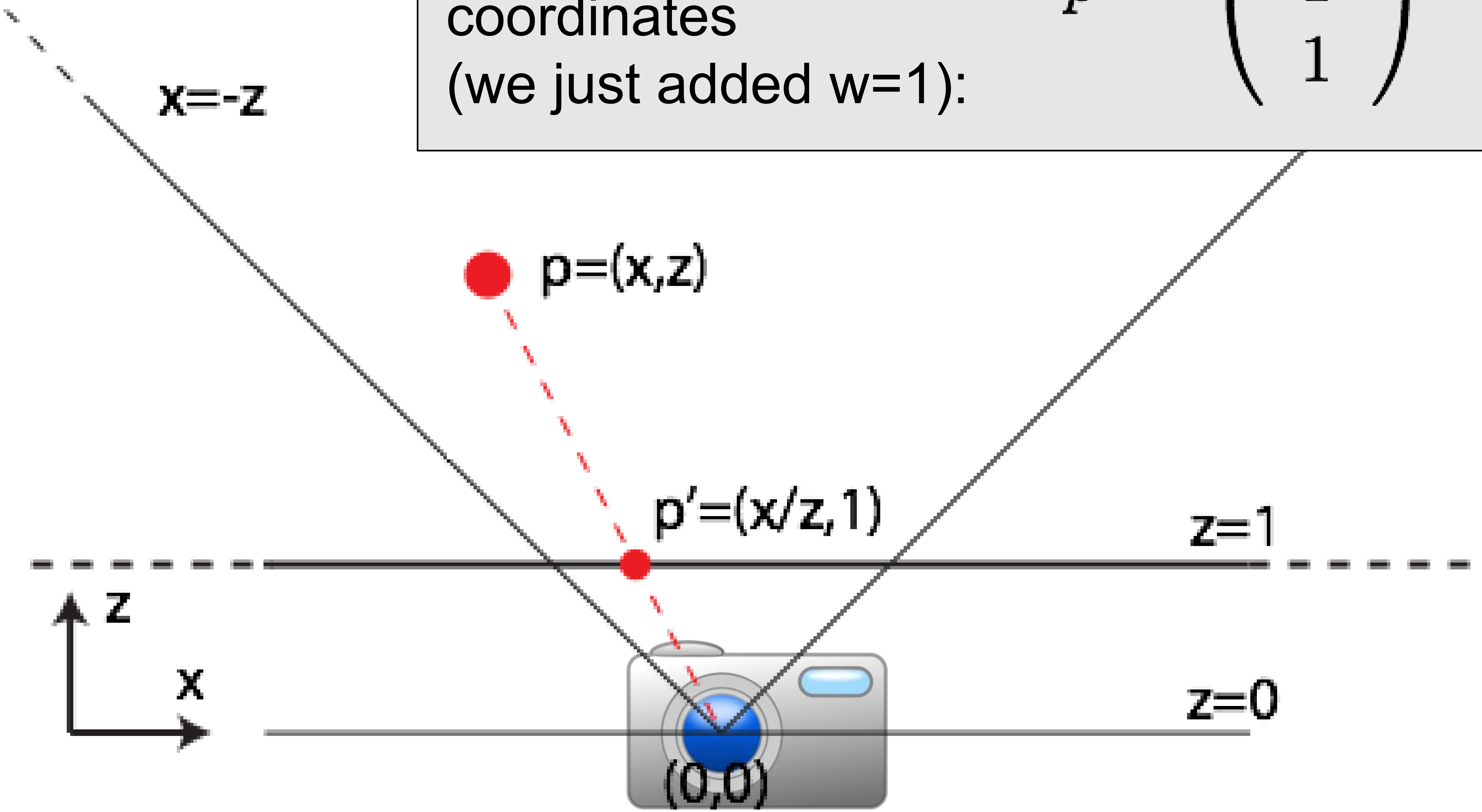
- Camera at origin, looking along *z*, 90 degree f.o.v., "image plane" at *z*=1



Perspective in 2D

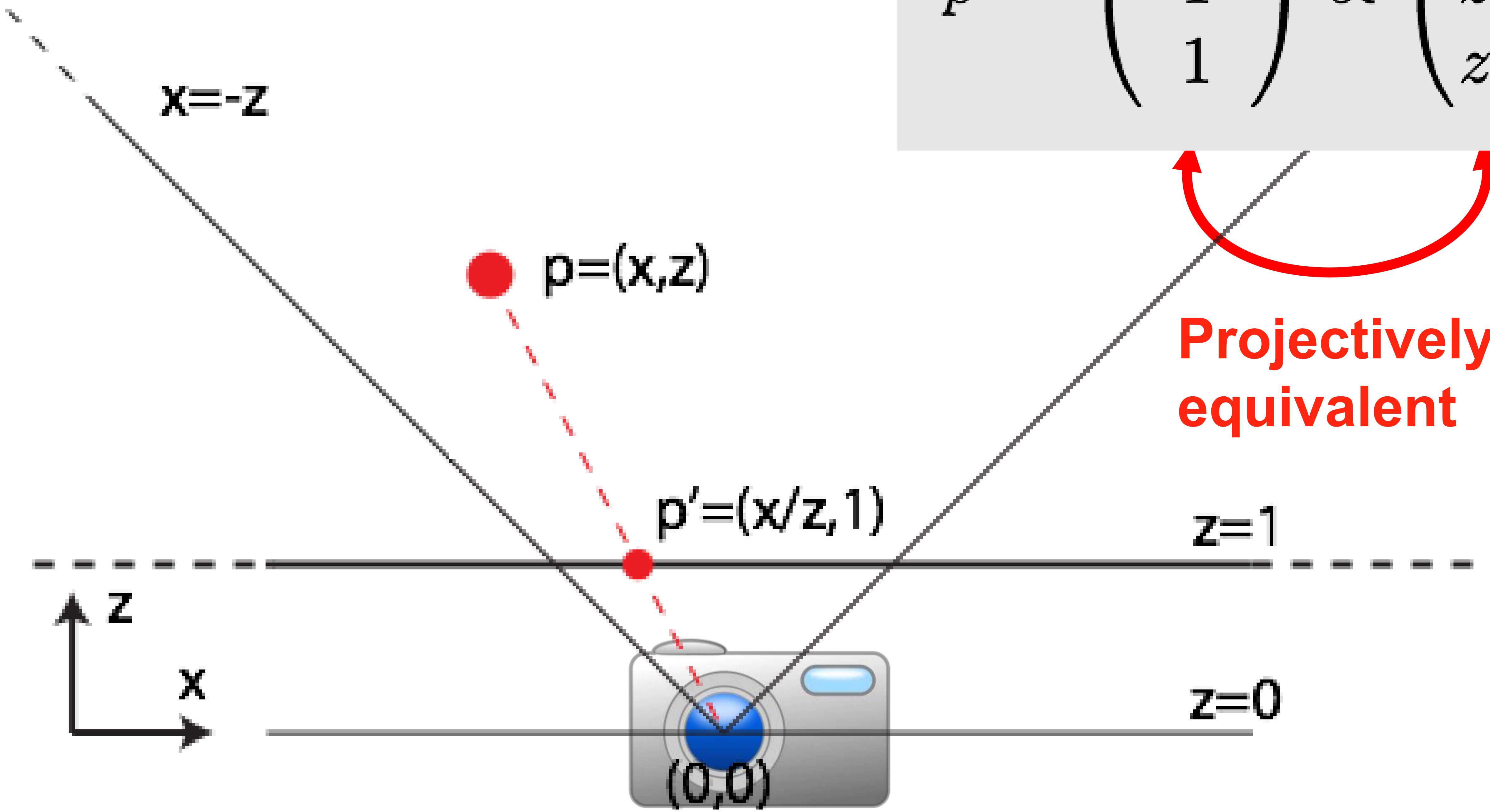
The projected point in homogeneous coordinates (we just added w=1):

$$p' = \begin{pmatrix} x/z \\ 1 \\ 1 \end{pmatrix}$$



Perspective in 2D

$$p' = \begin{pmatrix} x/z \\ 1 \\ 1 \end{pmatrix} \propto \begin{pmatrix} x \\ z \\ z \end{pmatrix}$$

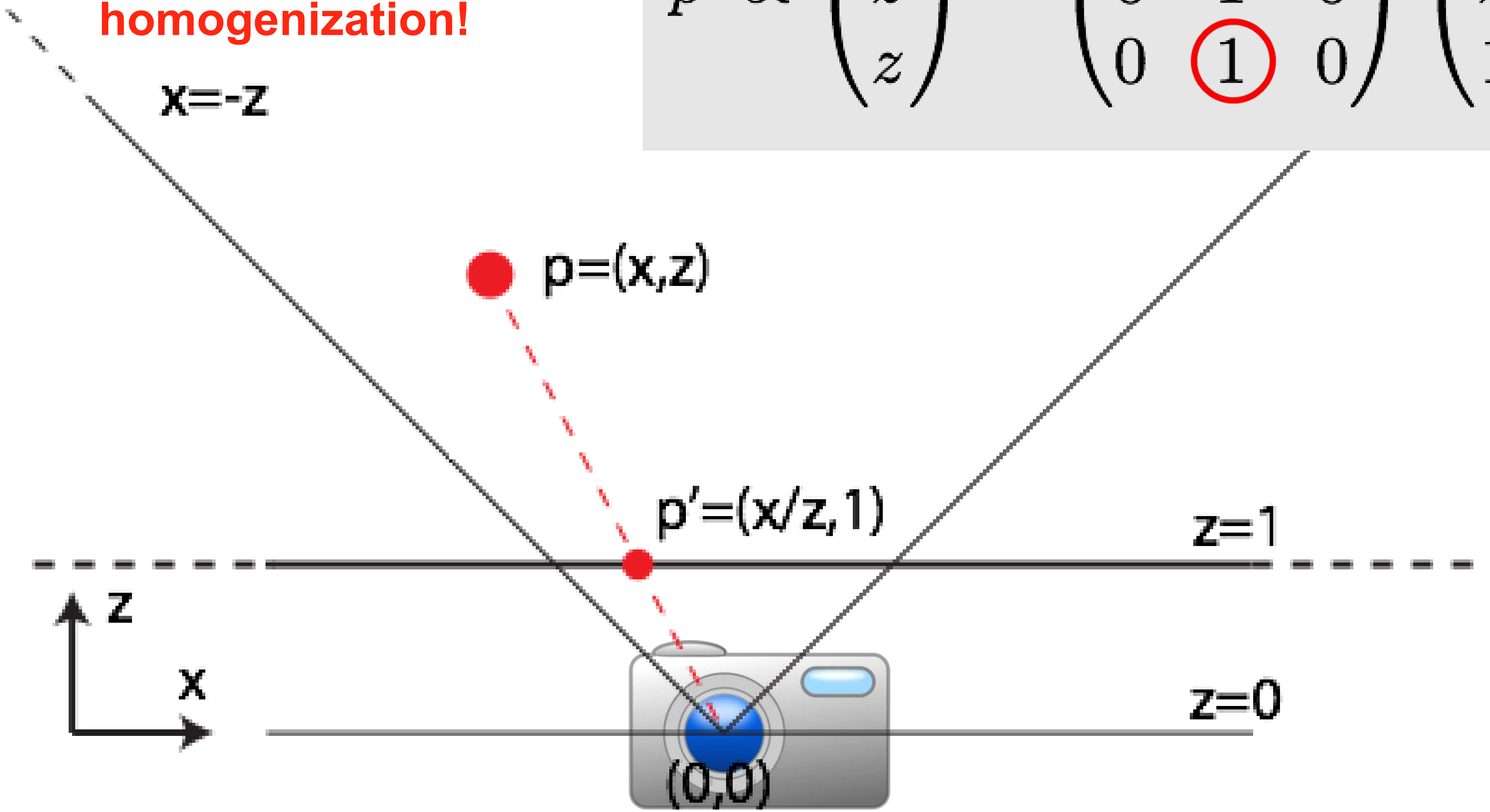


This image is in the public domain.
Source: http://openclipart.org/detail/34051/digicam-by-thesaurus.

Perspective in 2D

We'll just copy z to w, and get the projected point after homogenization!

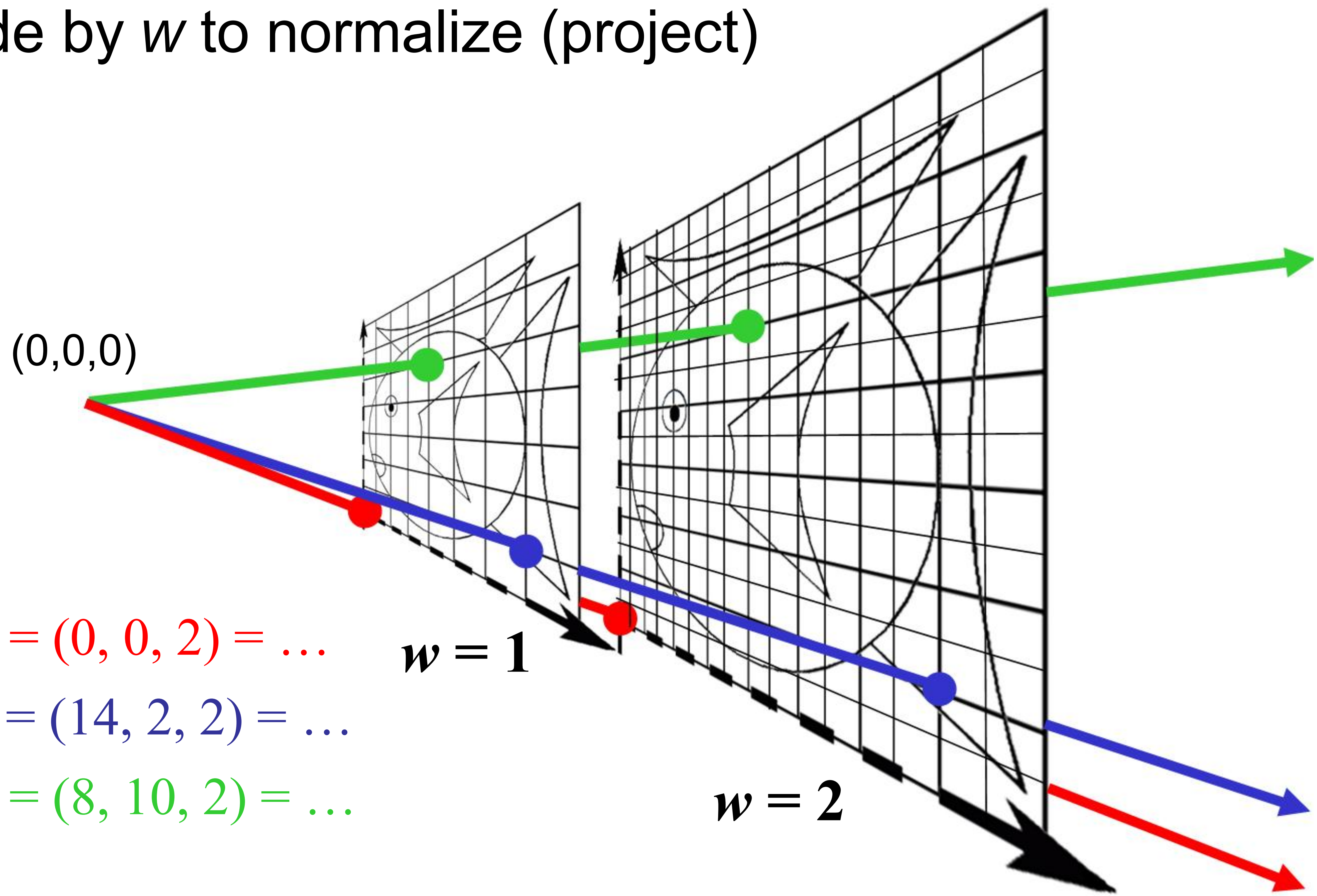
$$p' \propto \begin{pmatrix} x \\ z \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ z \\ 1 \end{pmatrix}$$



This image is in the public domain.
Source: http://openclipart.org/detail/34051/digicam-by-thesaurus.

Homogeneous Visualization

- Divide by w to normalize (project)



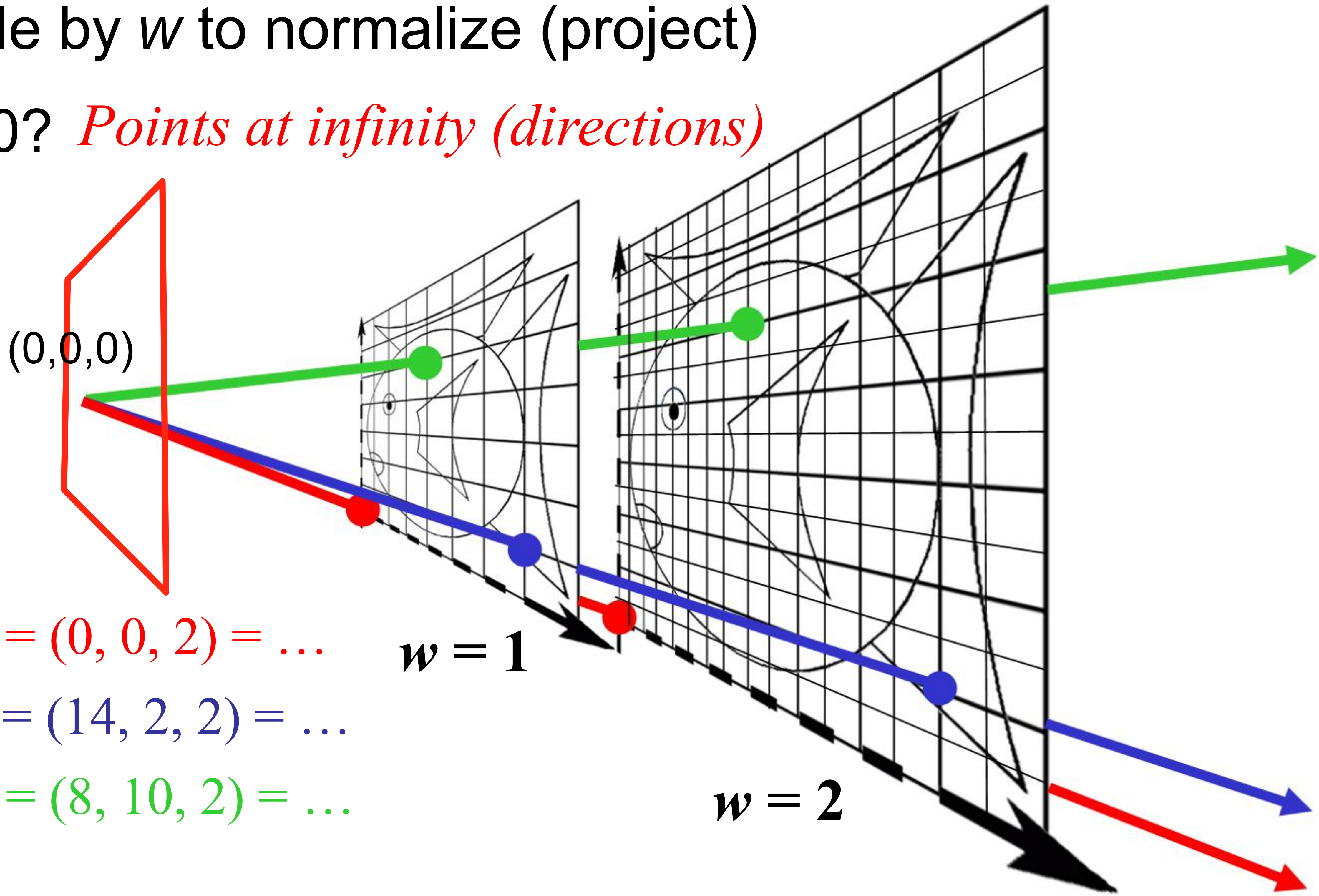
$(0, 0, 1) = (0, 0, 2) = \ldots$

$(7, 1, 1) = (14, 2, 2) = \ldots$

$(4, 5, 1) = (8, 10, 2) = \ldots$

Homogeneous Visualization

- Divide by w to normalize (project)
- $w = 0$? *Points at infinity (directions)*



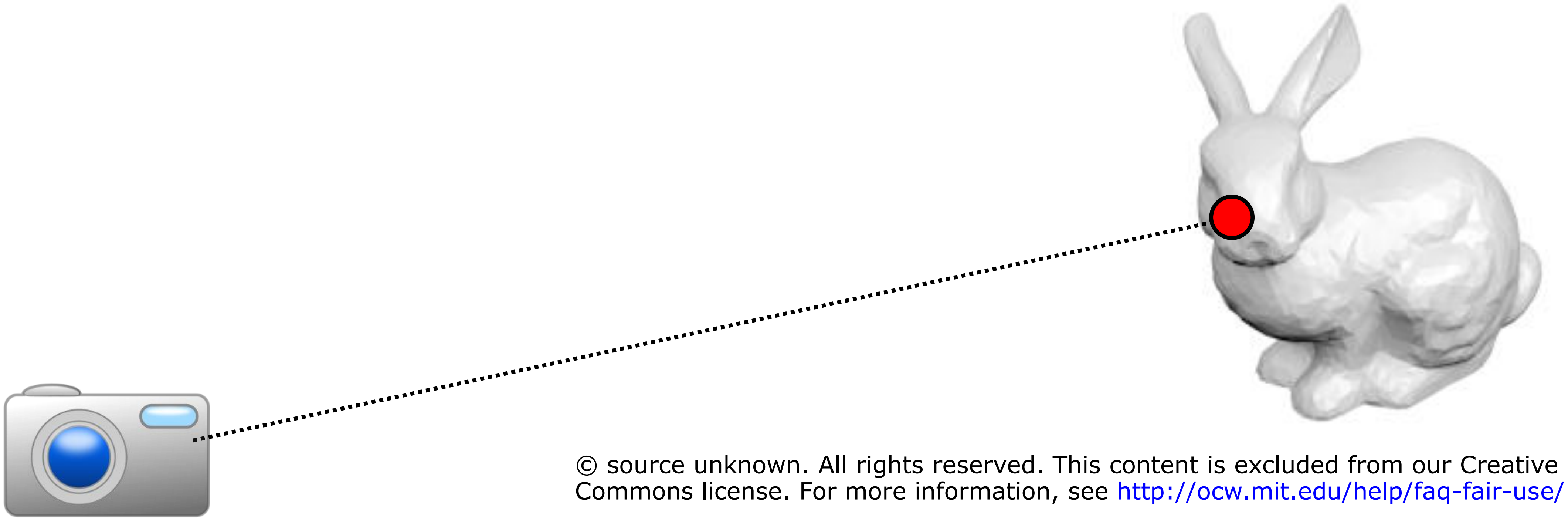
$(0, 0, 1) = (0, 0, 2) = \ldots$

$(7, 1, 1) = (14, 2, 2) = \ldots$

$(4, 5, 1) = (8, 10, 2) = \ldots$

Projective Equivalence – Why?

- For affine transformations,
  adding w=1 in the end proved to be convenient.
- The real showpiece is **perspective.**



Questions?

Eye candy: photo tourism

- Application of homogenous coordinates
- Goal: given N photos of a scene
  - find where they were taken
  - get 3D geometry for points in the scene

(a) (b) (c)

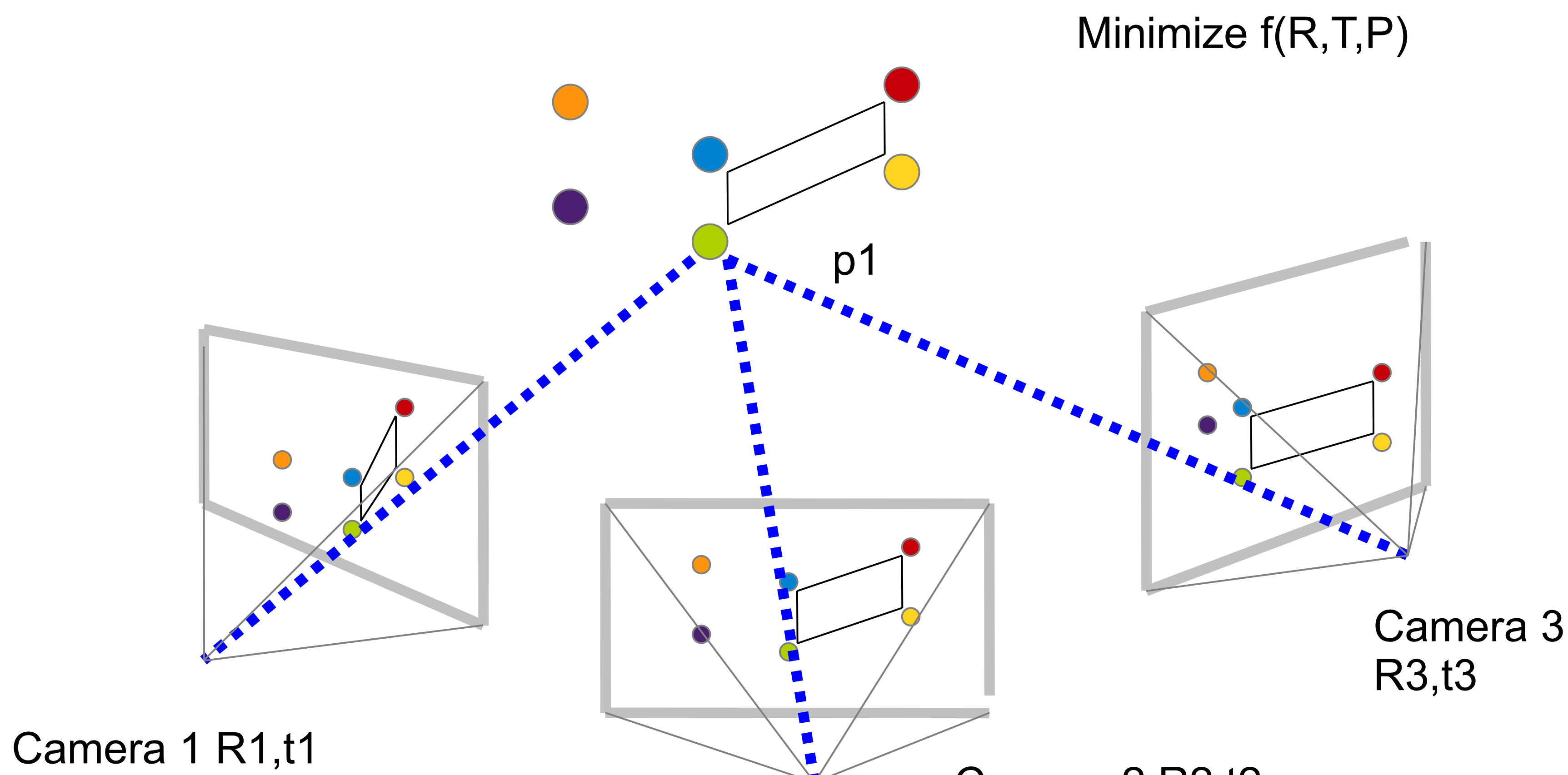
Figure 1: Our system takes unstructured collections of photographs such as those from online image searches (a) and reconstructs 3D points and viewpoints (b) to enable novel ways of browsing the photos (c).

Step 1: point correspondences

- Extract salient points (corners) from images
- Find the same scene point in other images
- To learn how it’s done, take 6.815

Structure from motion

- Given point correspondences
- Unknowns: 3D point location, camera poses
- For each point in each image, write perspective equations



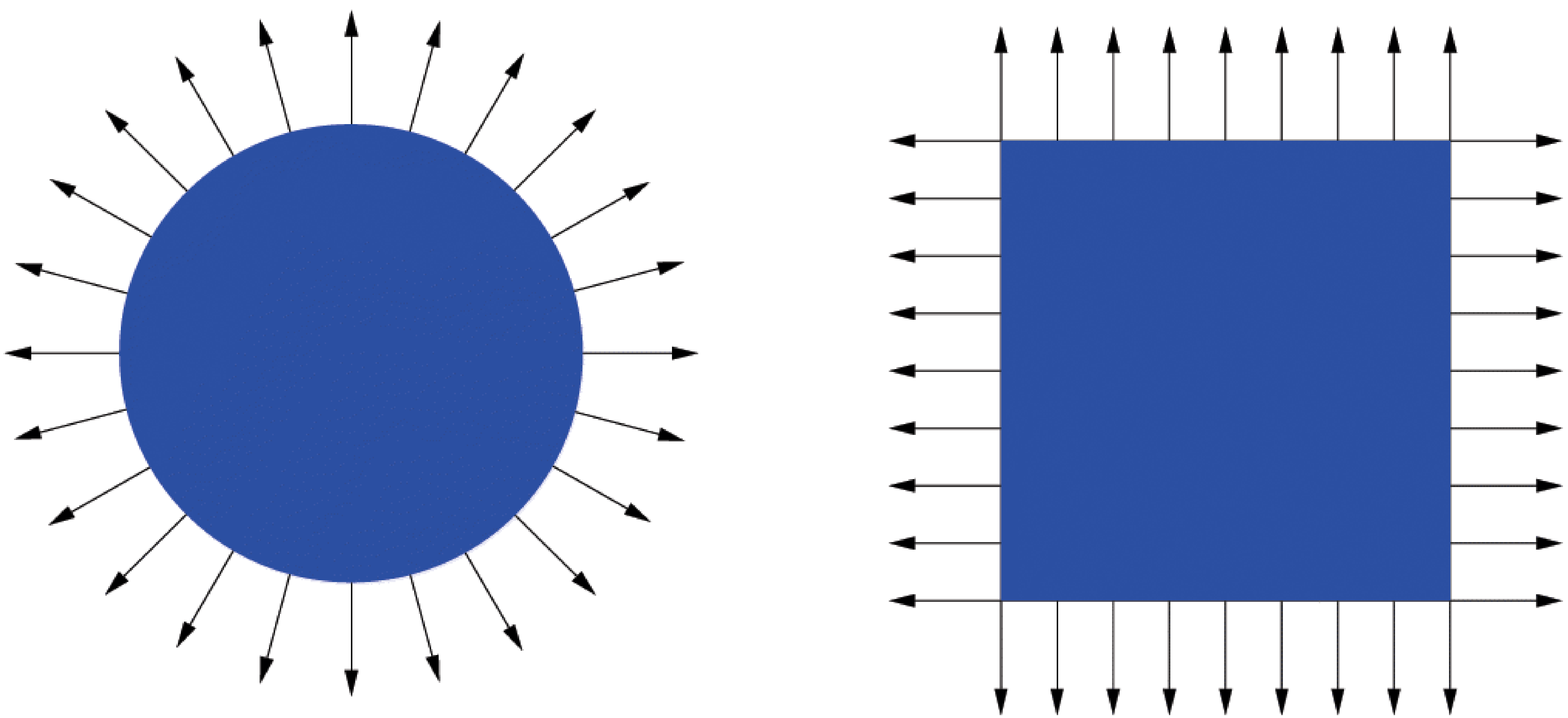
Eye candy: photo tourism

And that's it for today

- The rest on Thursday

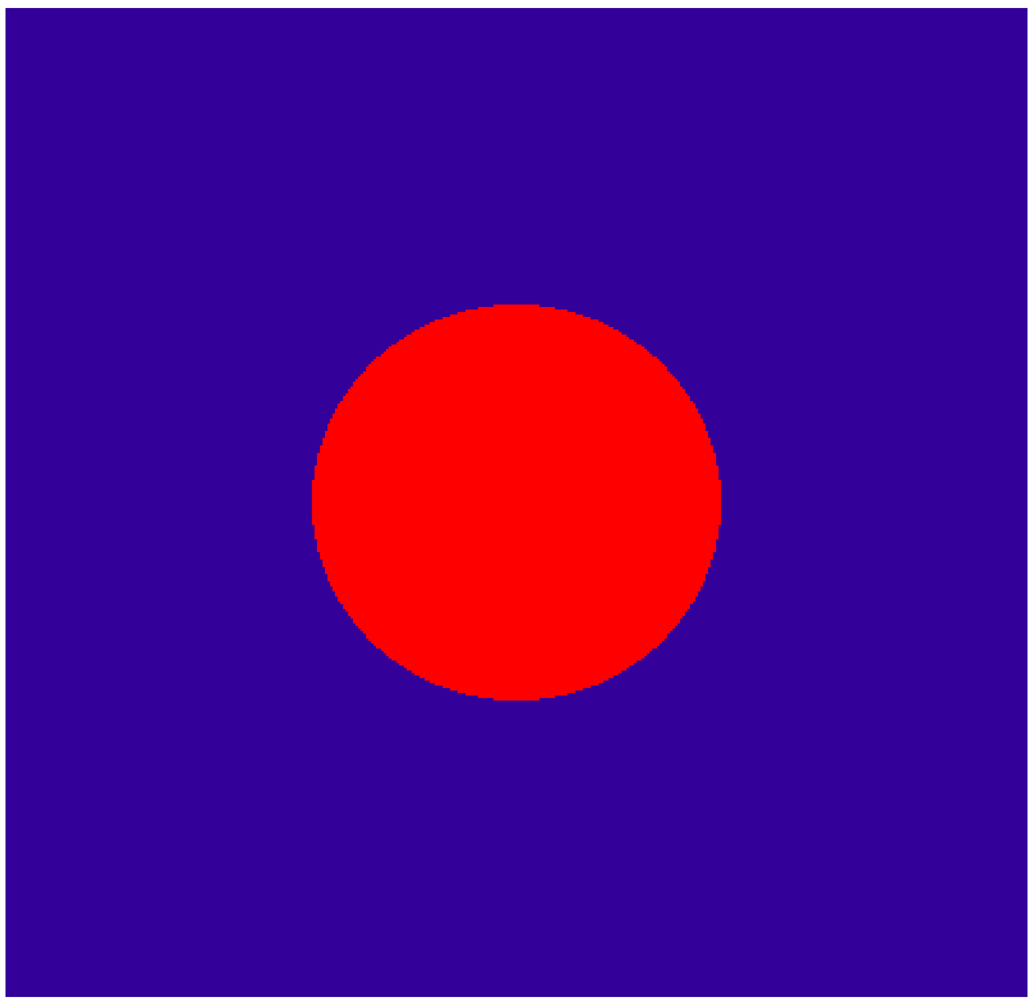
Normal

- Surface Normal: unit vector that is locally perpendicular to the surface

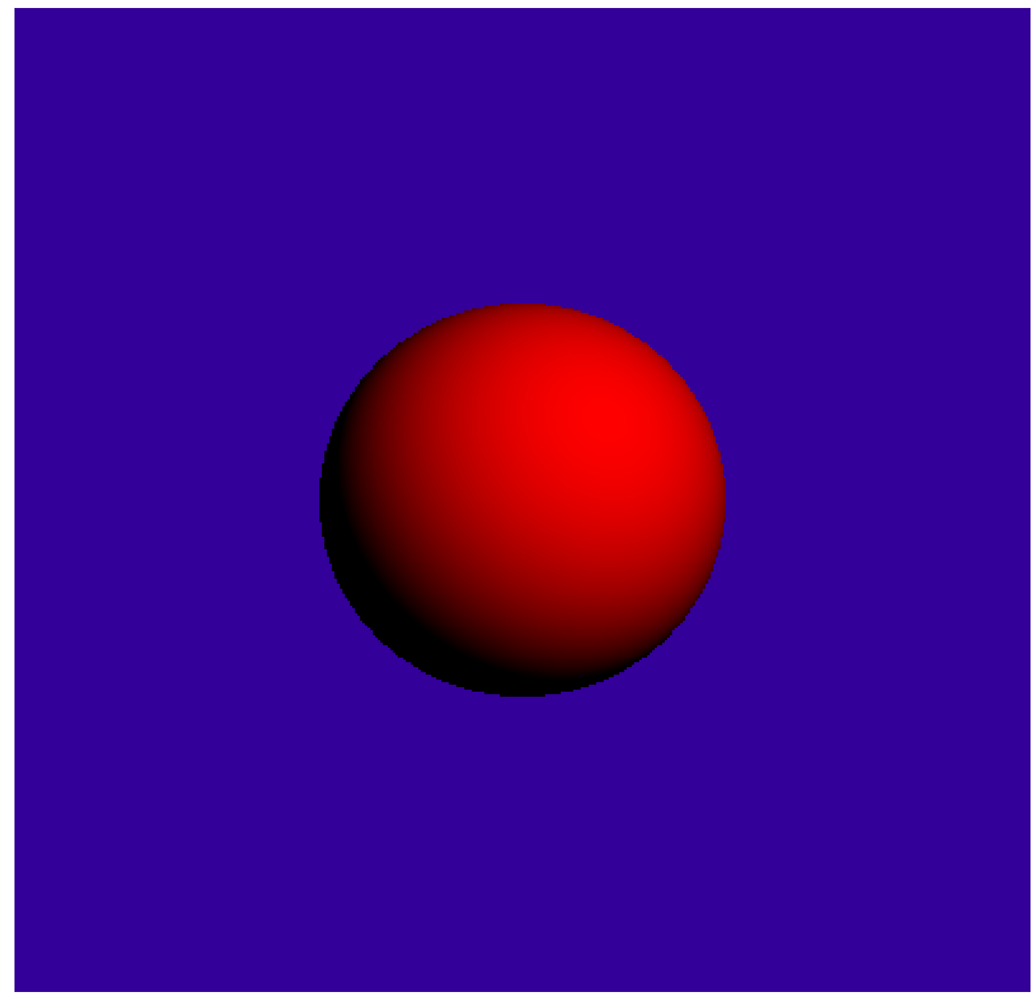


Why is the Normal important?

- It's used for shading — makes things look 3D!

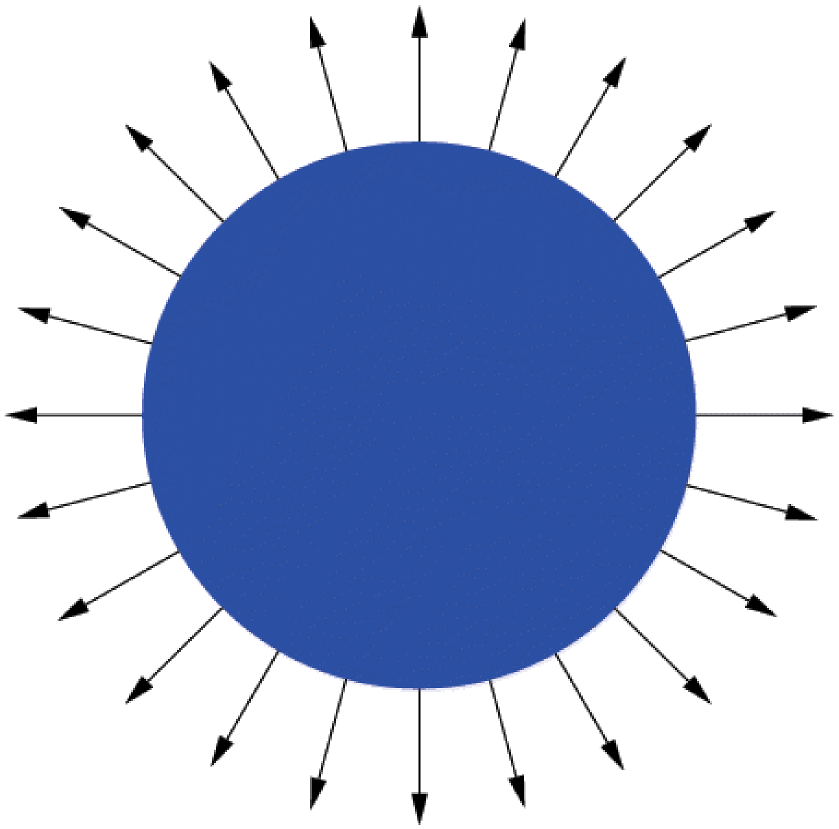


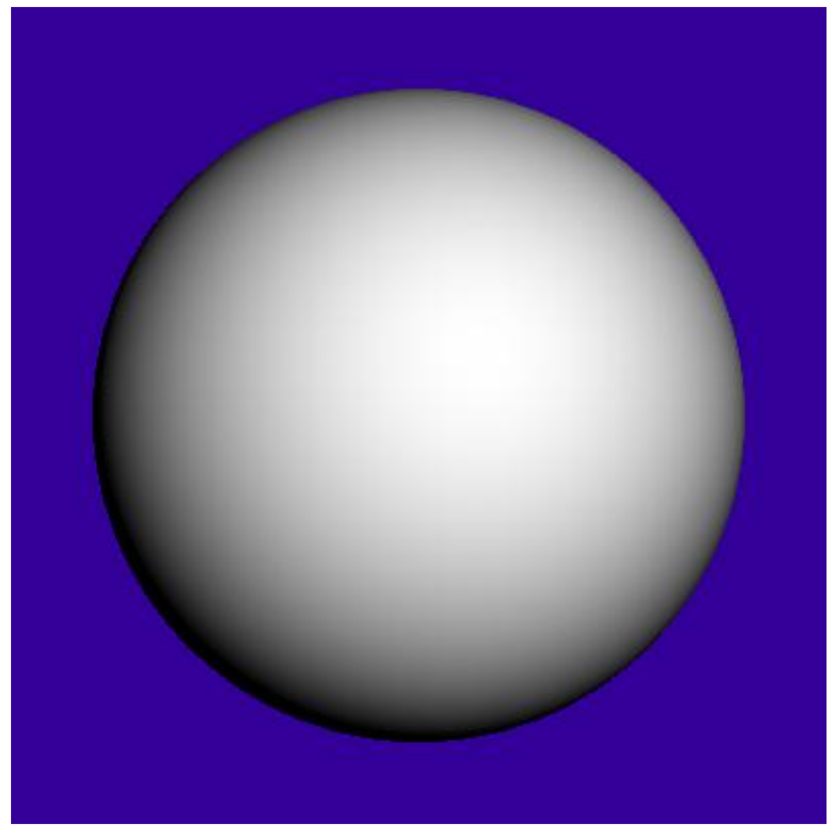
object color only

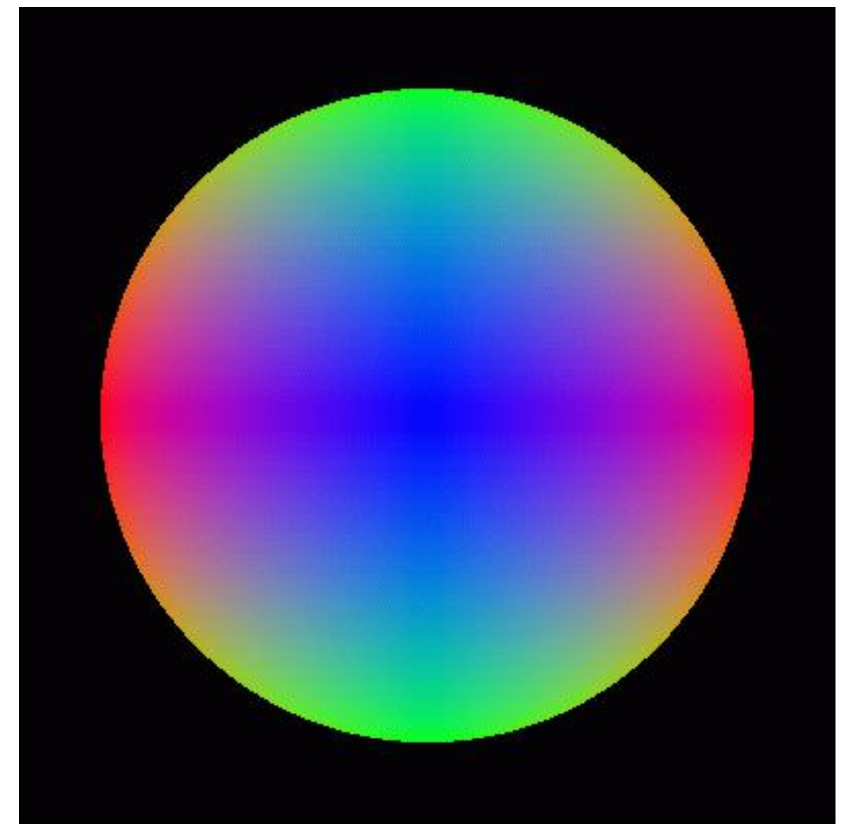


Diffuse Shading

Visualization of Surface Normal





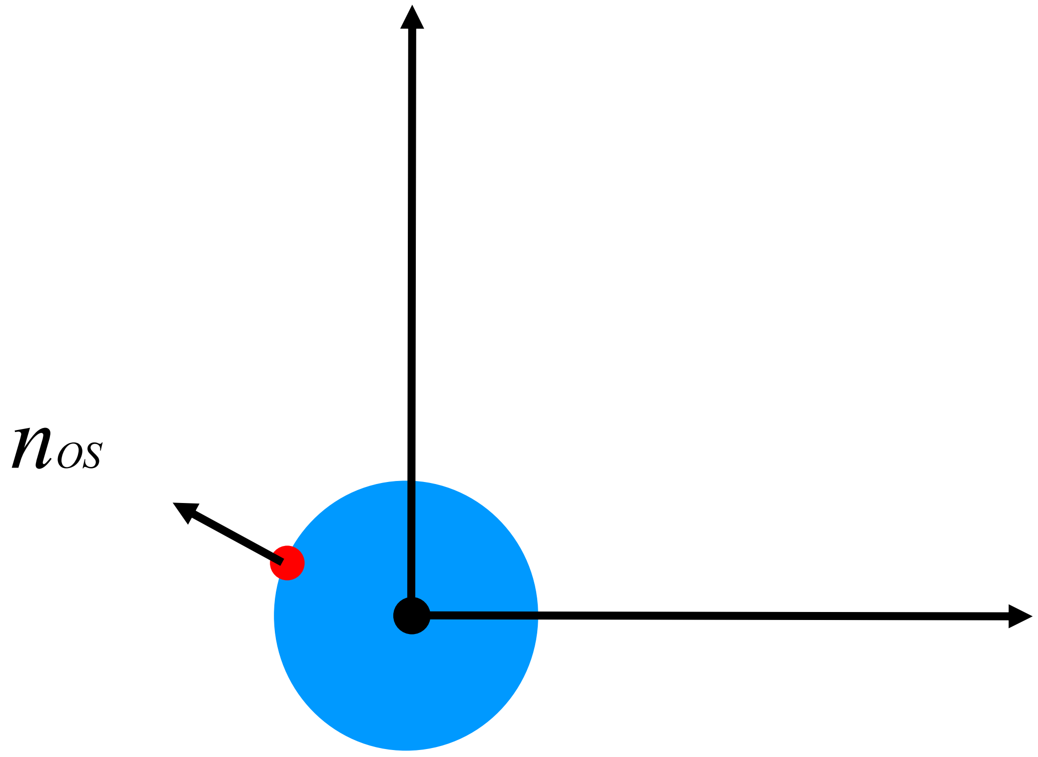


$\pm\, x$ = Red

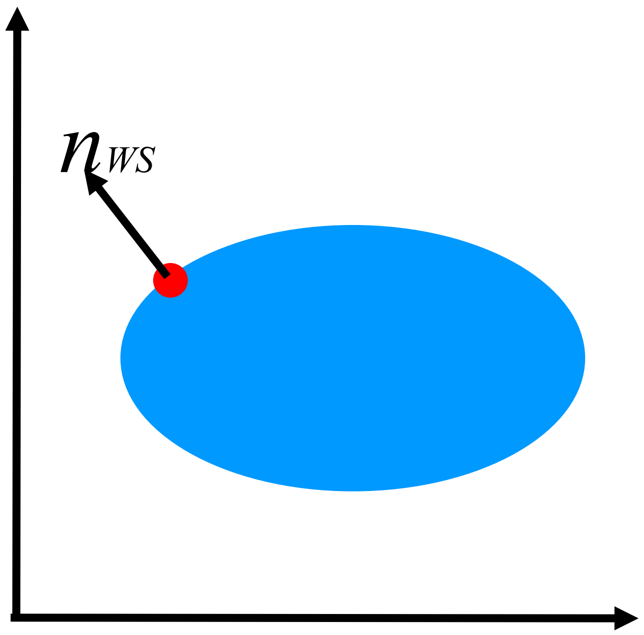
$\pm\, y$ = Green

$\pm\, z$ = Blue

How do we transform normals?



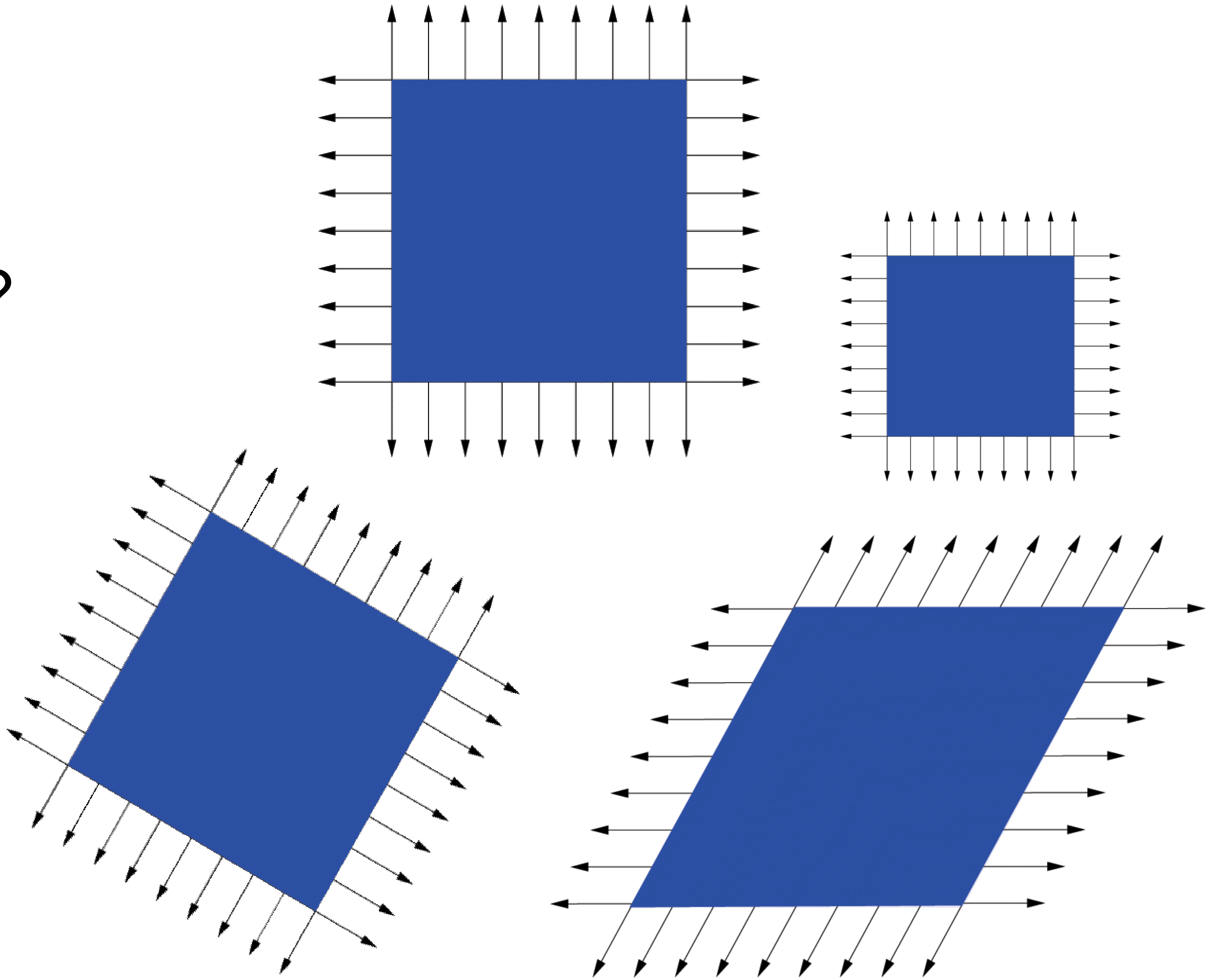
Object Space



World Space

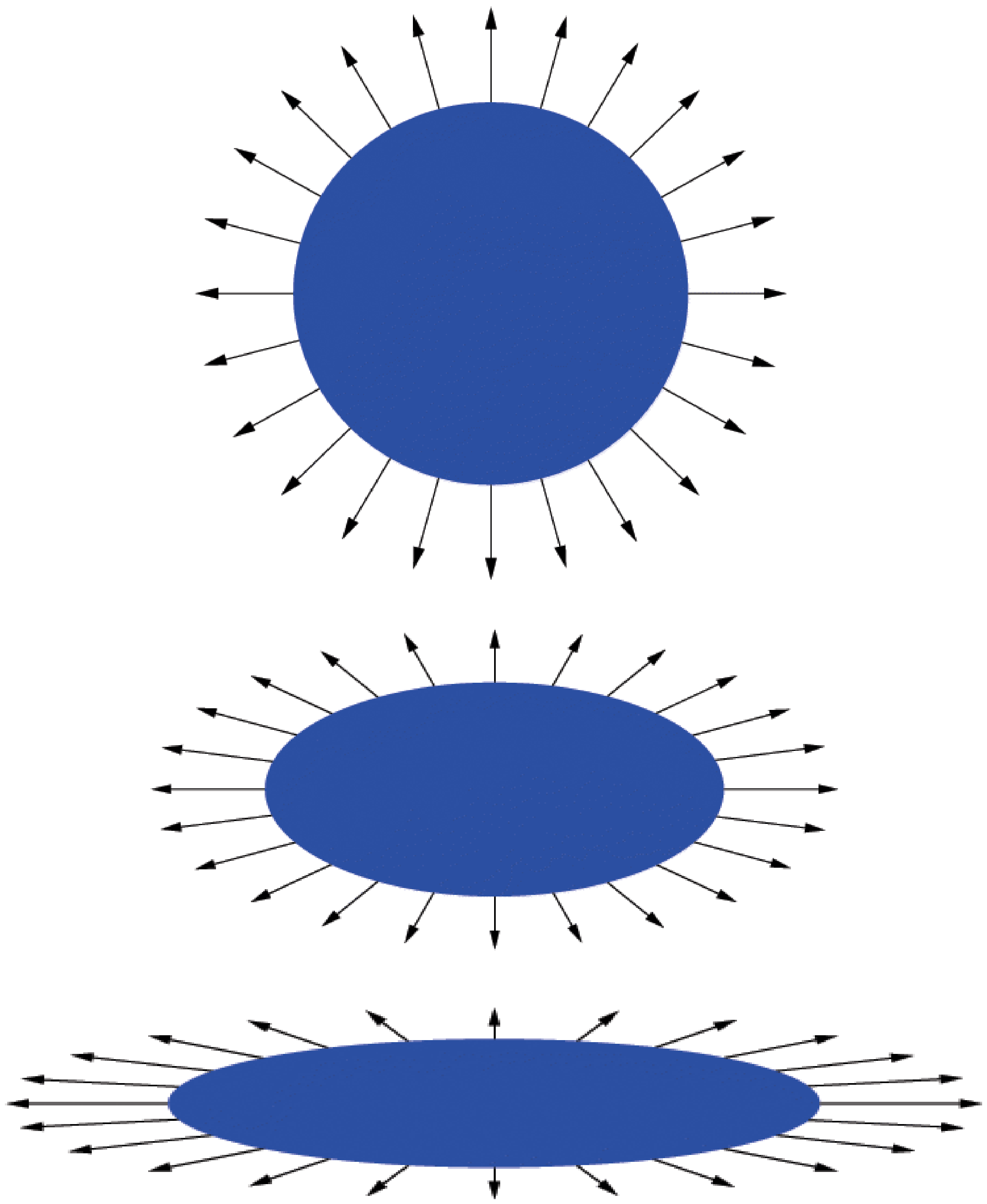
Transform Normal like Object?

- translation?
- rotation?
- isotropic scale?
- scale?
- reflection?
- shear?
- perspective?



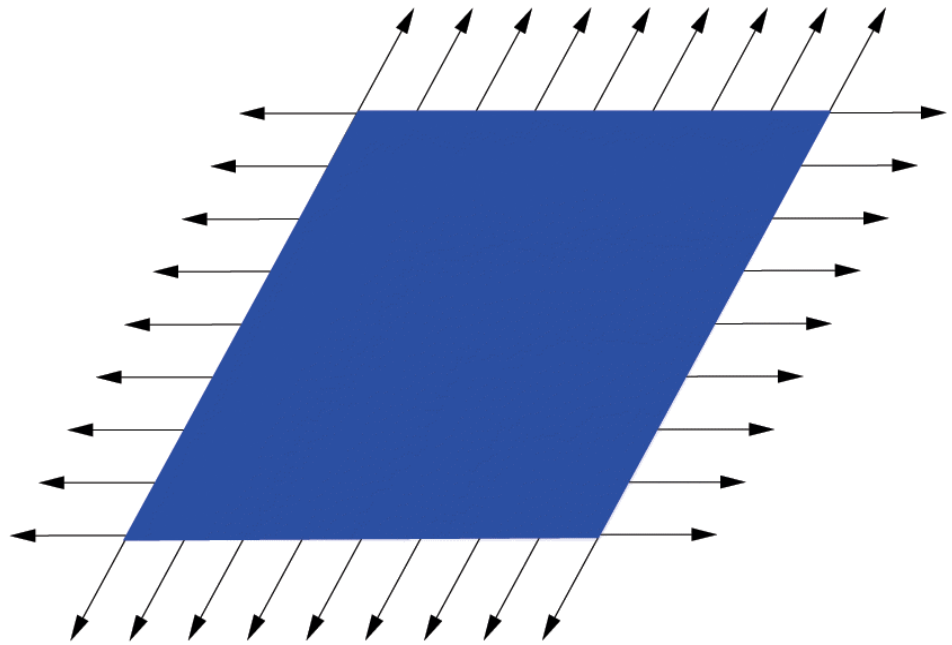
Transform Normal like Object?

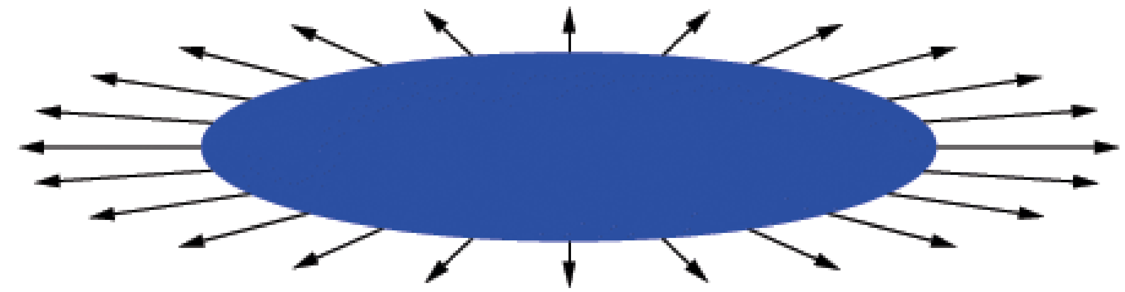
- translation?
- rotation?
- isotropic scale?
- scale?
- reflection?
- shear?
- perspective?



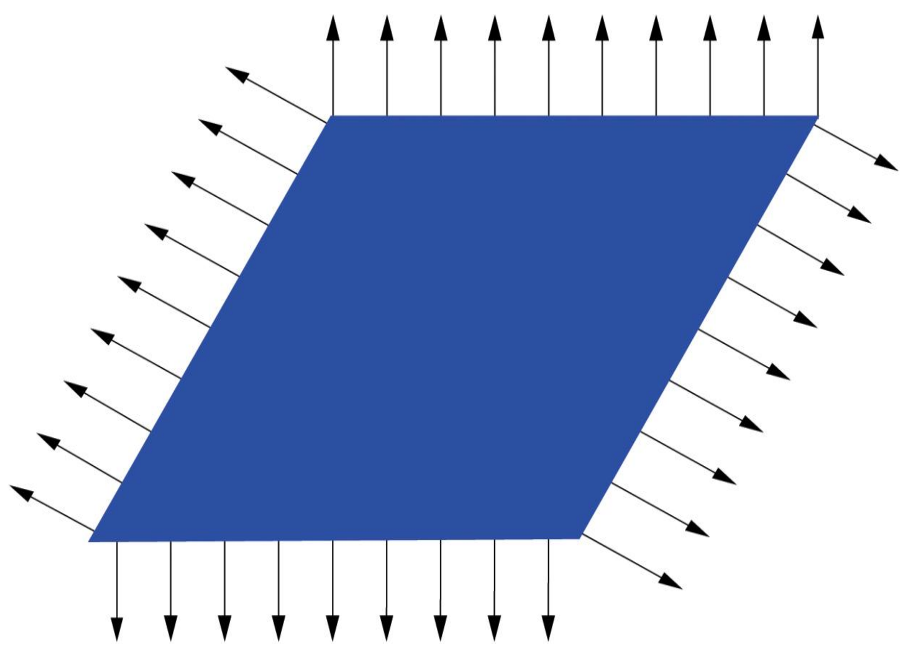
Transformation for shear and scale

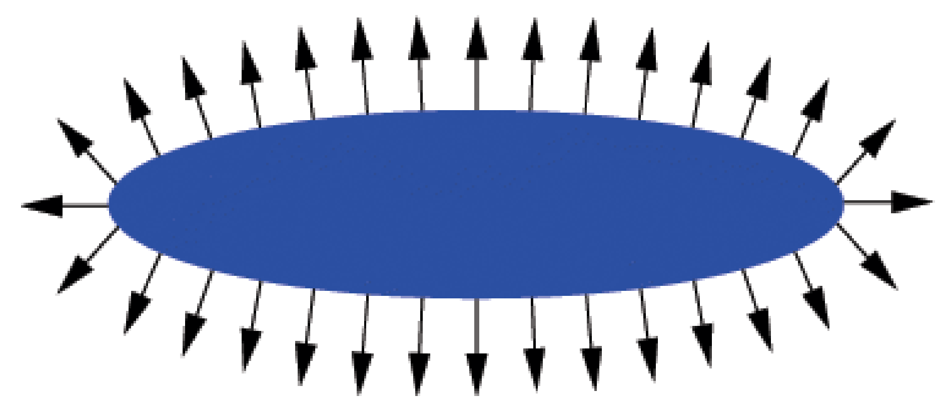
Incorrect
Normal
Transformation



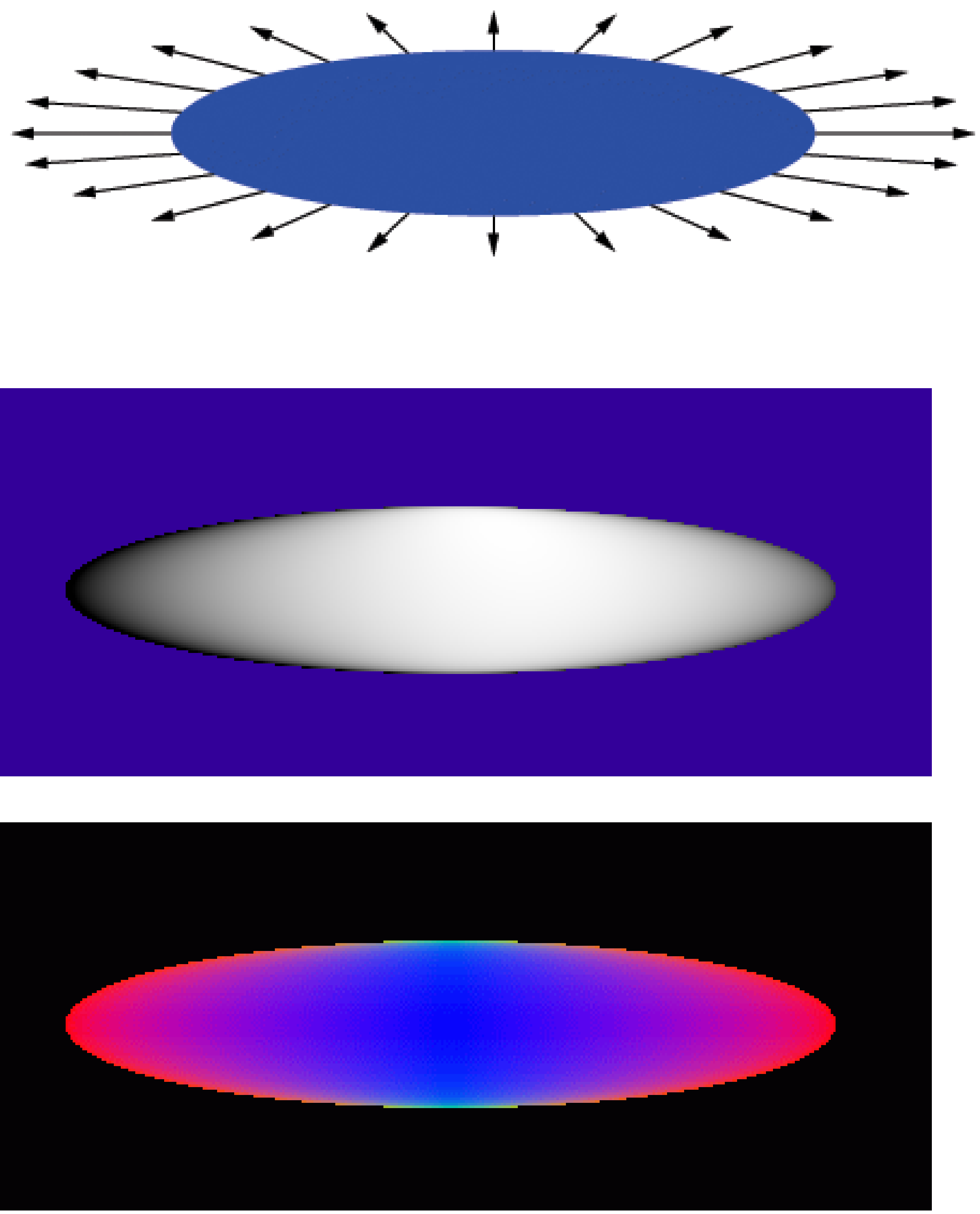


Correct
Normal
Transformation

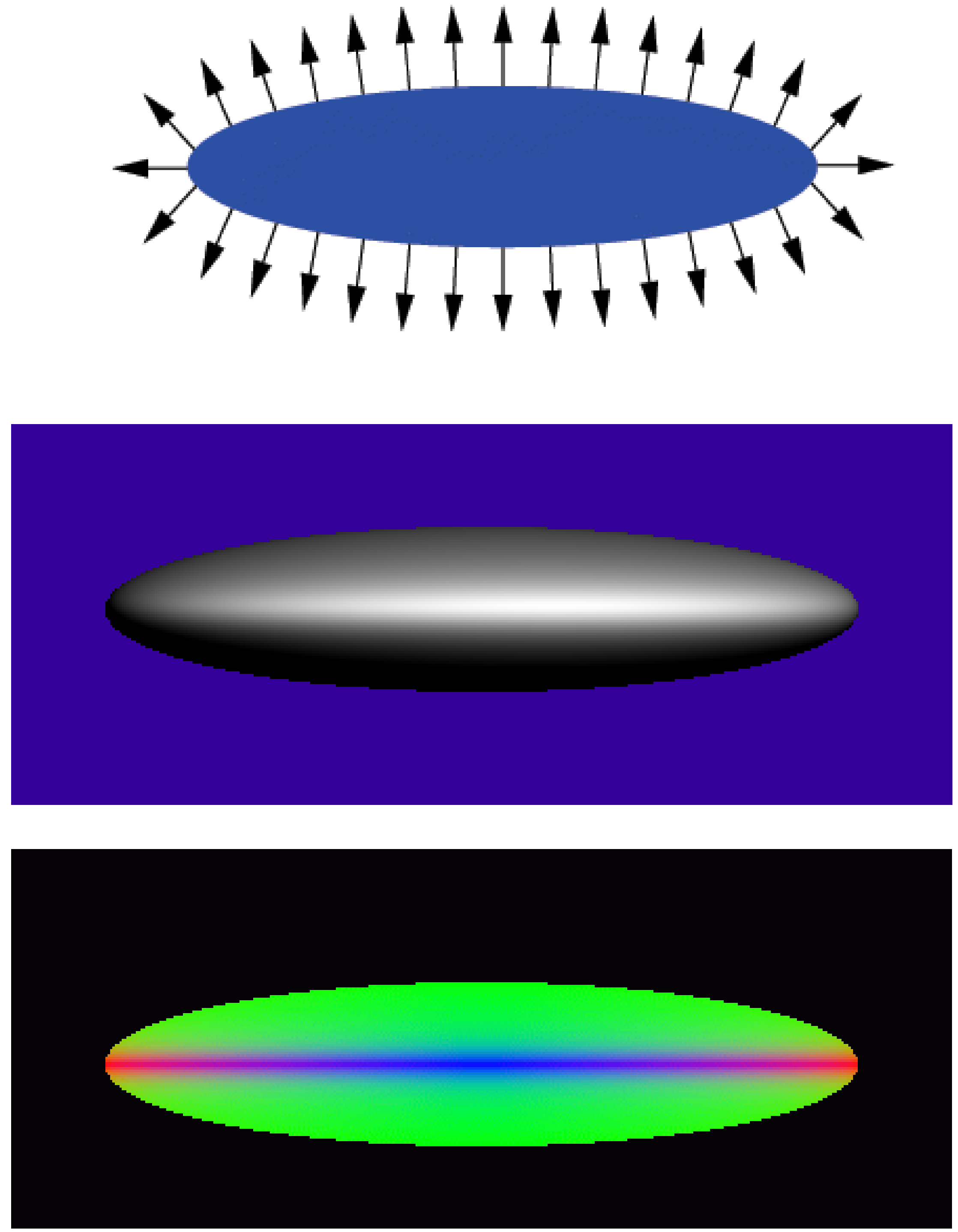




More Normal Visualizations



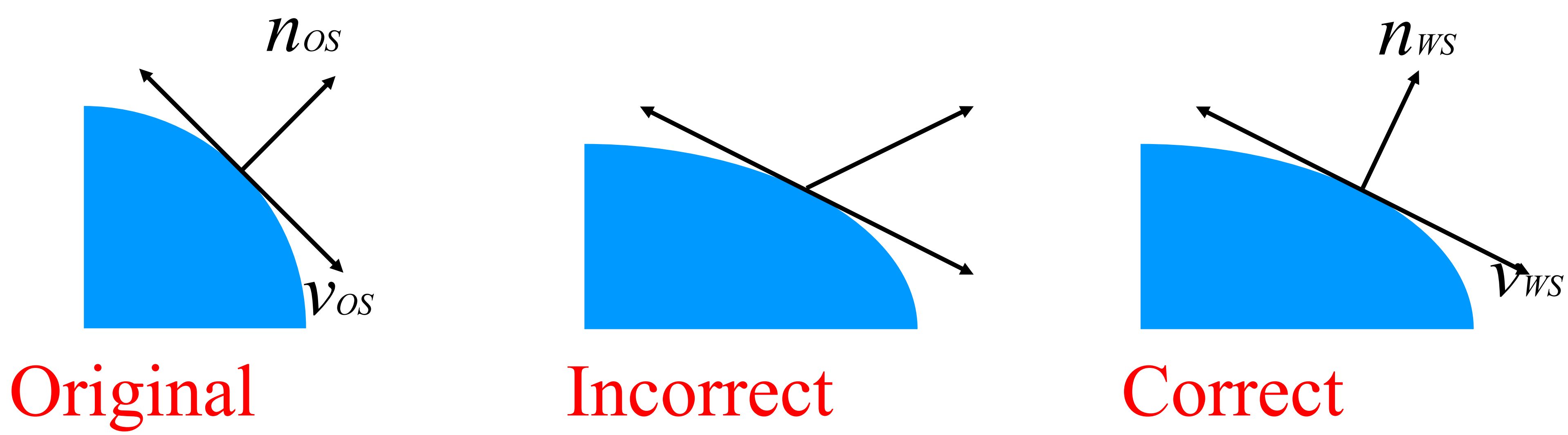
Incorrect Normal Transformation



Correct Normal Transformation

So how do we do it right?

- Think about transforming the *tangent plane* to the normal, not the normal *vector*



Pick any vector v_{OS} in the tangent plane, how is it transformed by matrix **M**?

$$v_{WS} = \mathbf{M}\, v_{OS}$$

Transform tangent vector *v*

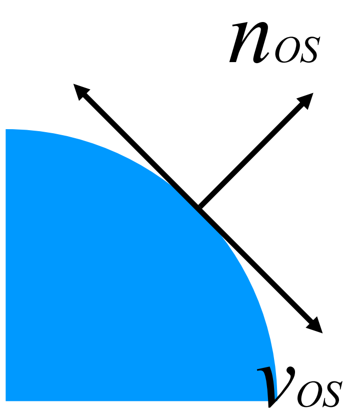
v is perpendicular to normal n:

Dot product $\quad n_{os}^{\mathsf{T}}\, v_{os} = 0$

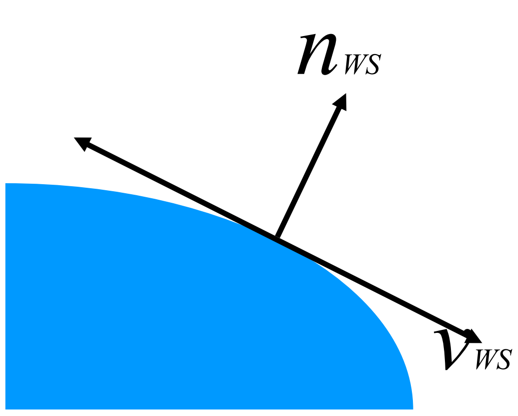
$$n_{os}^{\mathsf{T}}\, (\mathbf{M}^{-1}\ \mathbf{M})\ v_{os} = 0$$

$$(n_{os}^{\mathsf{T}}\ \mathbf{M}^{-1})\ (\mathbf{M}\ \ v_{os}) = 0$$

$$(n_{os}^{\mathsf{T}}\ \mathbf{M}^{-1})\ v_{ws} = 0$$



v_{ws} is perpendicular to normal n_{ws}:



$$n_{ws}^{\mathsf{T}} = n_{os}^{\mathsf{T}}\, (\mathbf{M}^{-1})$$

$$\boxed{n_{ws} = (\mathbf{M}^{-1})^{\mathsf{T}}\, n_{os}}$$

$$n_{ws}^{\mathsf{T}}\, v_{ws} = 0$$

Digression

$$\boxed{n_{WS} = (\mathbf{M}^{-1})^{\mathsf{T}}\, n_{OS}}$$

- The previous proof is not quite rigorous; first you'd need to prove that tangents indeed transform with **M**.
  - Turns out they do, but we'll take it on faith here.
  - If you believe that, then the above formula follows.

Comment

- So the correct way to transform normals is:

$$n_{WS} = (\mathbf{M}^{-1})^{\mathsf{T}}\, n_{OS}$$

Sometimes denoted $\mathrm{M}^{-\mathrm{T}}$

- But why did $n_{WS} = \mathbf{M}\, n_{OS}$ work for similitudes?
- Because for similitude / similarity transforms,

$$(\mathbf{M}^{-1})^{\mathsf{T}} = \lambda\, \mathbf{M}$$

- e.g. for orthonormal basis:

$$\mathbf{M}^{-1} = \mathbf{M}^{\mathsf{T}} \quad \text{i.e.} \quad (\mathbf{M}^{-1})^{\mathsf{T}} = \mathbf{M}$$

Connections

- Not part of class, but cool
  - “Covariant”: transformed by the matrix
    - e.g., tangent
  - “Contravariant”: transformed by the inverse transpose
    - e.g., the normal
    - a normal is a “co-vector”

- Google “differential geometry” to find out more

Questions?

That's All for Today

- Further Reading
  - Buss, Chapter 2

- Other Cool Stuff
  - Algebraic Groups
  - http://phototour.cs.washington.edu/
  - http://phototour.cs.washington.edu/findingpaths/
  - Free-form deformation of solid objects
  - Harmonic coordinates for character articulation