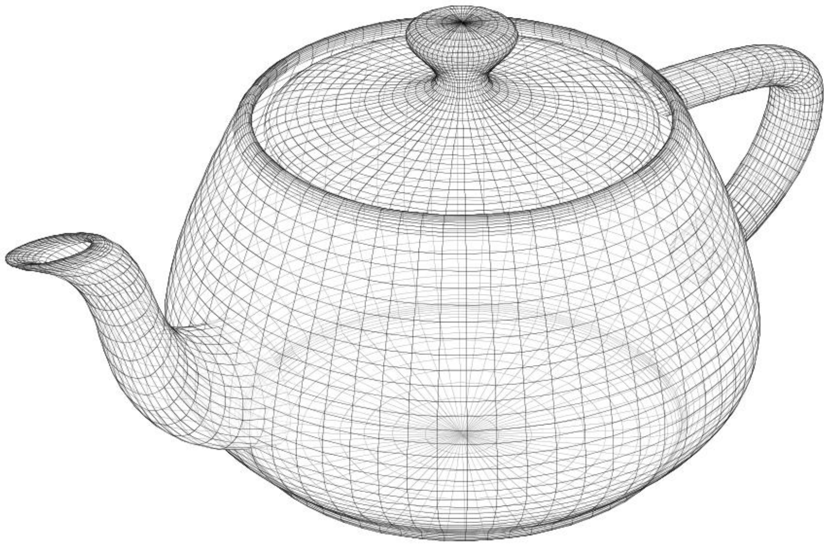


Affine Transformations

Production Computer Graphics

Eric Shaffer

ILLINOIS

Objectives

Understand what affine transformations are

- …and why they are not linear transformations

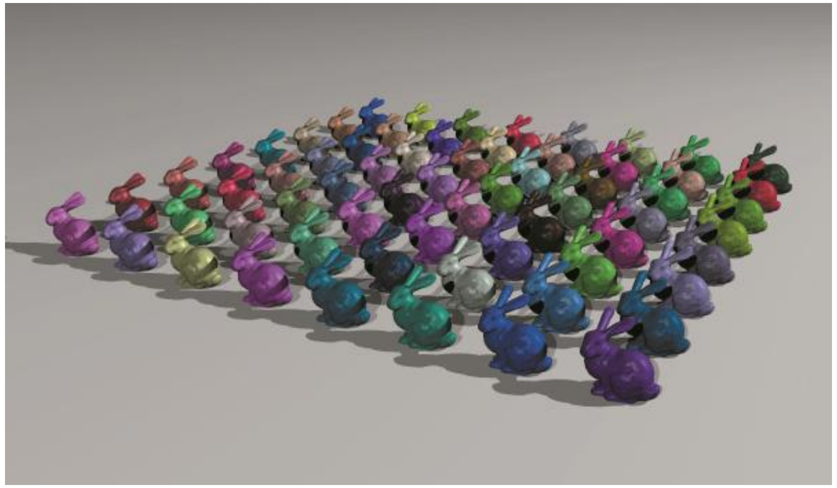
Review how to represent an affine transformation as a matrix

- …and composite transformations
- …and inverse transformations

Review homogeneous coordinates

Understand how to intersect transformed objects

Know how to use instancing



Affine Transformations

We will review:

- Rotation
- Scaling
- Translation
- Reflection

All linear functions are affine functions

Linear functions keep the origin fixed

- **Affine functions need not do so**

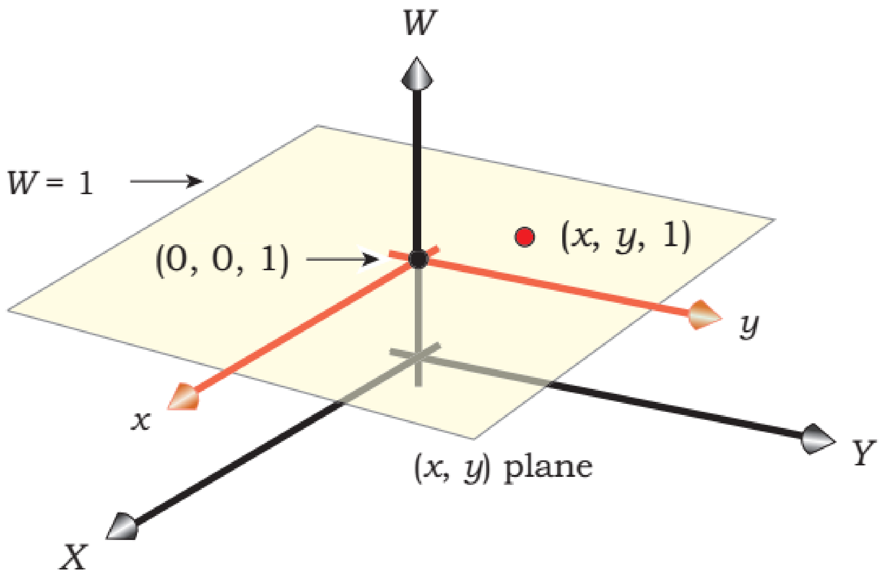
So…to include translation we use affine transformations

- …and we're going to use homogeneous coordinates…

Homogeneous Coordinates

- Cartesian coordinates are often used in Euclidean Geometry
- Homogeneous coordinates are used in Projective Geometry
- Map between a 2D Cartesian point and a homogeneous point
  - w=0 corresponds to a point at infinity
- Mapping generalizes directly to 3D Cartesian points

$$\underset{\text{Homogeneous}}{(x, y, w)} \Leftrightarrow \underset{\text{Cartesian}}{\left(\frac{x}{w}, \frac{y}{w}\right)}$$



The (x, y) plane is embedded at W = 1 in the (X, Y, W) coordinate system.

What does (wX,wY,w) form in projective space?

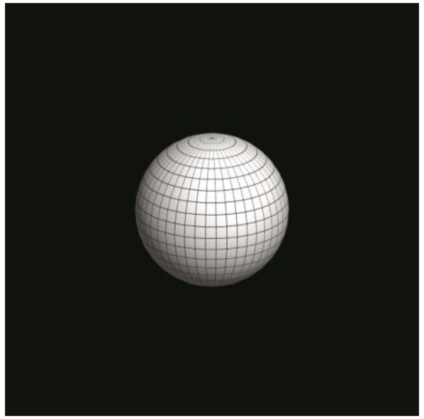
ILLINOIS

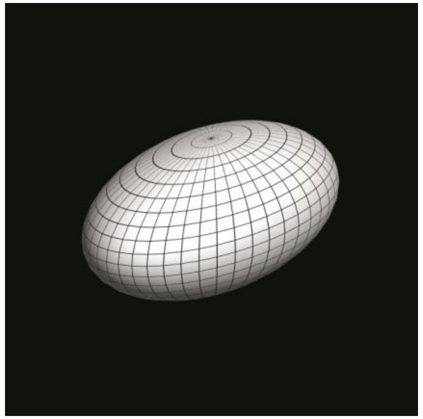
3D Translation

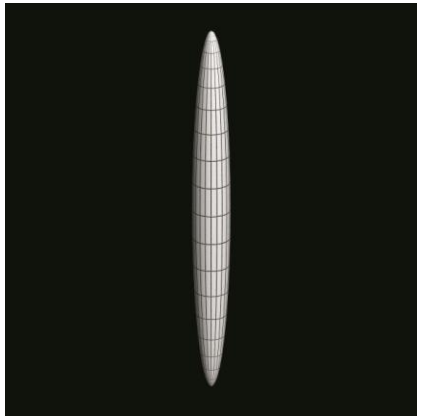
$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

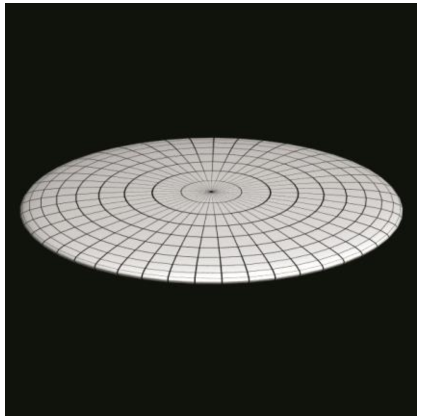
3D Scaling

$$S(a,b,c) = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$









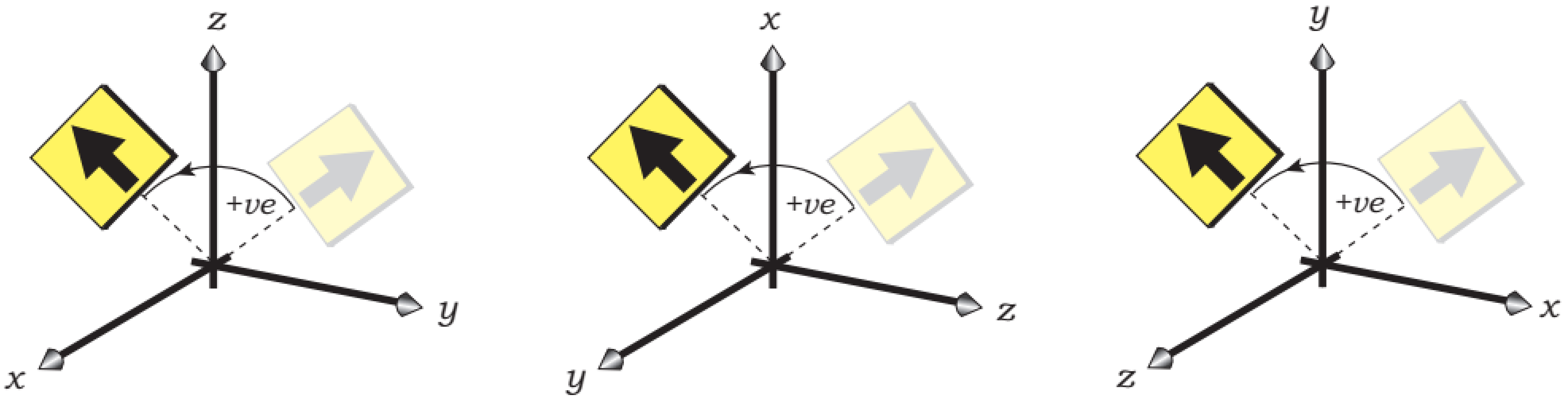
3D Rotation

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pick an axis to rotate around (x, y, or z)

3D Rotation

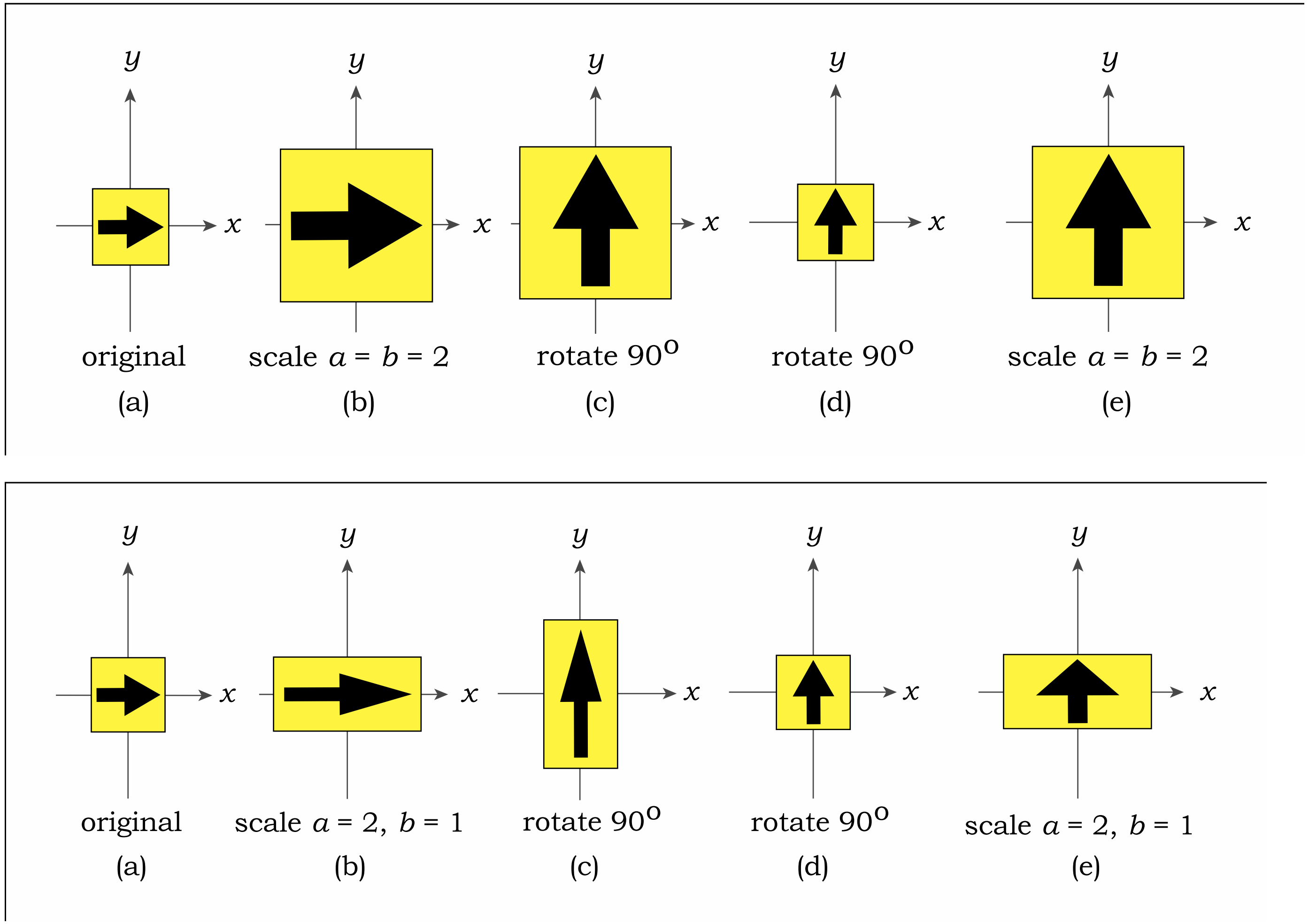


- Pick an axis to rotate around (x, y, or z)
- Looking along the negative axis , the rotation is counter-clockwise

3D Reflection

$$R_x = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

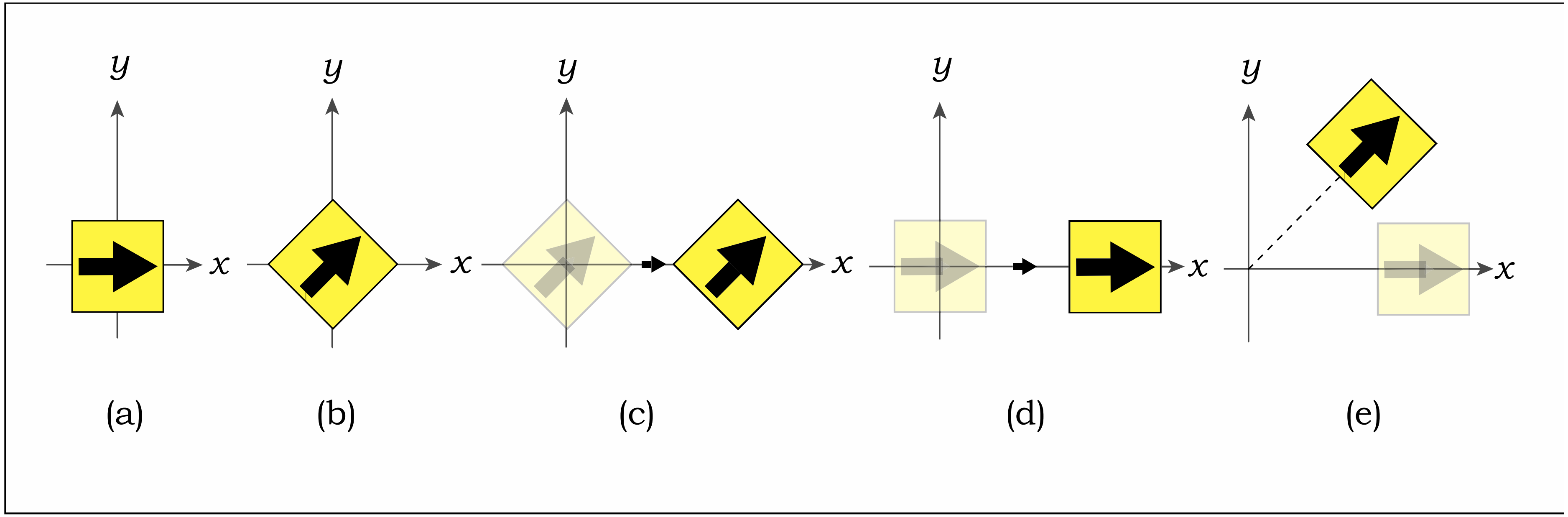
Composing Transformations



Non-uniform scaling and rotation do not commute

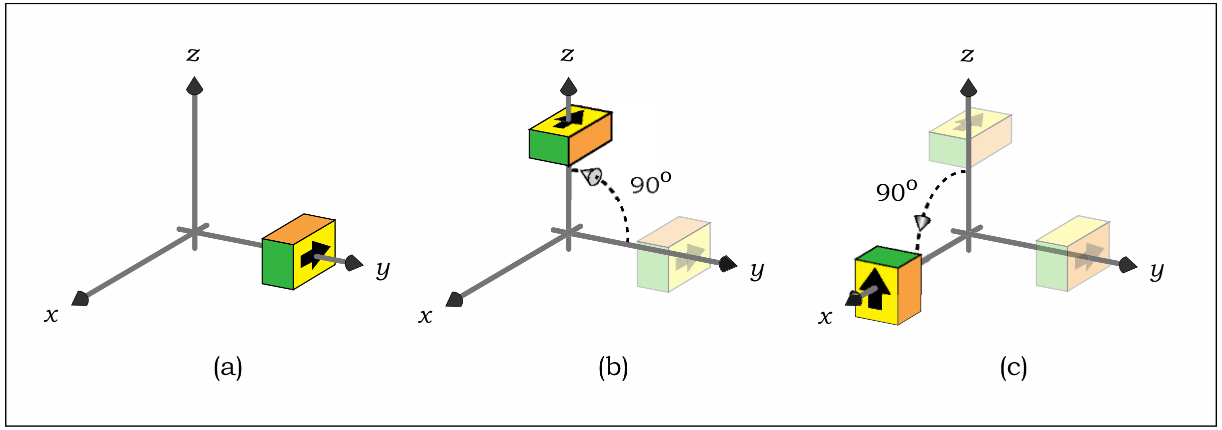
ILLINOIS

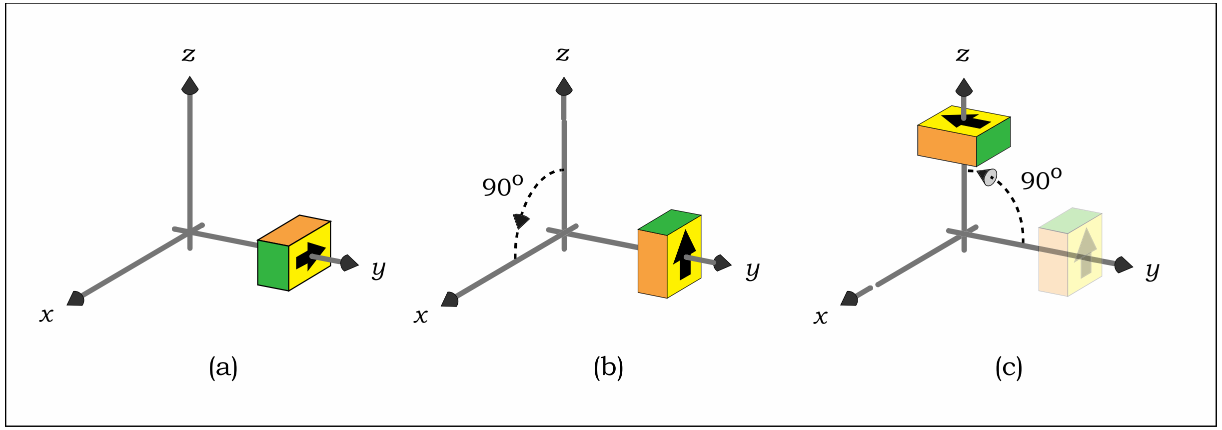
Composing Transformations



Rotation and translation do not commute

Composing Transformations





Rotations do not commute

Inverse Transformations

Generally easy to construct

- For translation and rotation → just negate the terms in the matrix
- For scaling → transform diagonal entries to their reciprocals

If we have $Tv = T_n T_{n-1} \ldots T_1 v$ then $T^{-1} = T_1^{-1} \ldots T^{-1}_{n-1} T^{-1}_{n}$

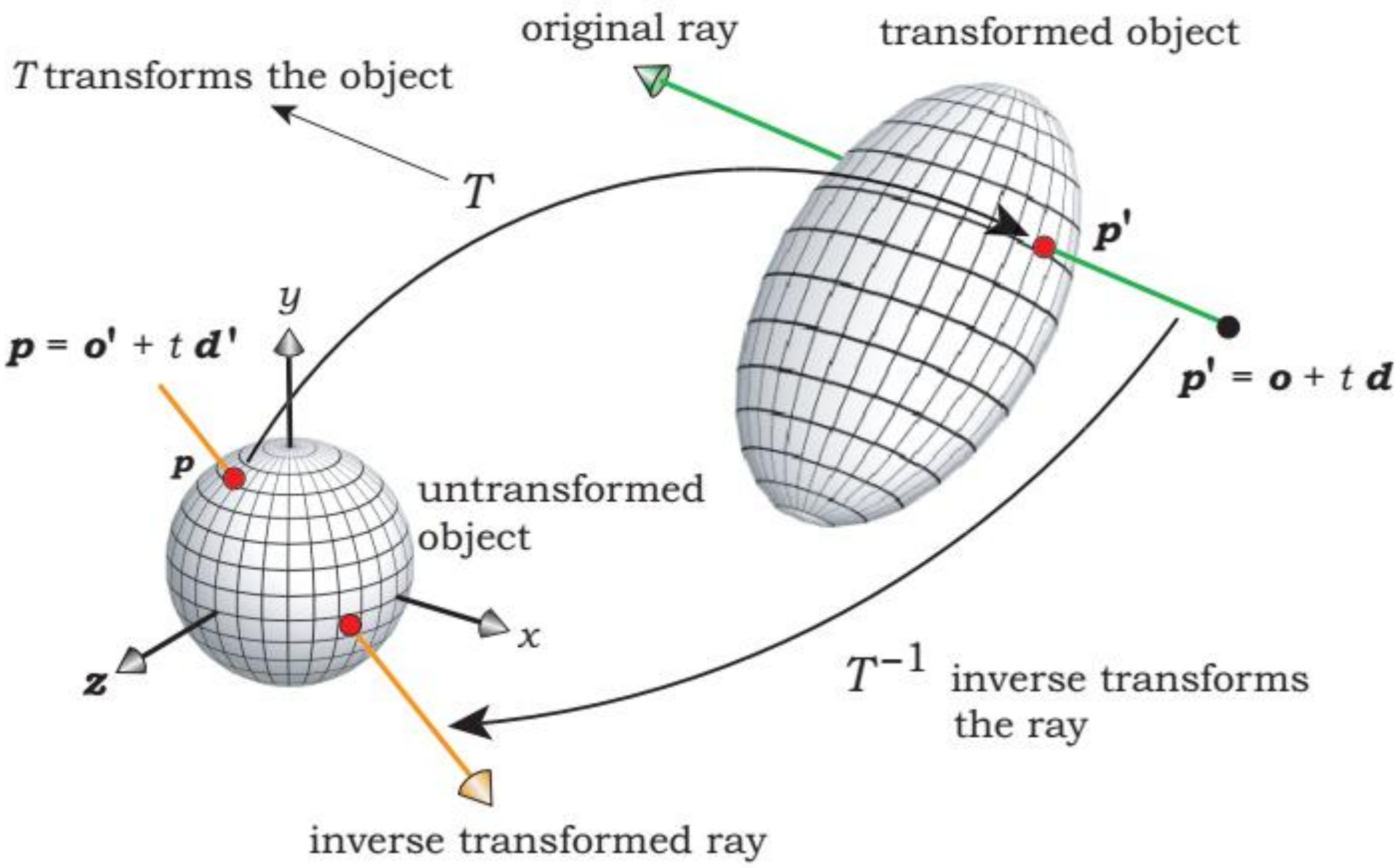
$T^{-1}Tv = v$

ILLINOIS

Intersecting Transformed Objects

- Apply inverse set of transformations to the ray
- Intersect inverse transformed ray with untransformed object
- Compute the normal at the hit point
- Use hit point to compute hit point on transformed object
- Use normal to compute normal on transformed object

Intersecting Transformed Objects



ILLINOIS

Intersecting Transformed Objects

To transform the ray

- Assume matrix T transforms the object
- Assume original ray is ***o* + *td***
- The inverse transformed ray is then $\boldsymbol{T^{-1}o+tT^{-1}d}$

Note that ***d*** is a vector

- It should be unaffected by translation
- Use a homogenous coordinate of 0 in ***d***

$$\begin{bmatrix} d_x \\ d_y \\ d_z \\ 0 \end{bmatrix}$$

Finding the Hit Point

- p = hit point for transformed ray and untransformed object
- p' = hit point for original ray and transformed object
- If the hit point p occurs at t_0 then the hit point p' occurs at t_0
  - Just calculate the point p' using t_0 and the original ray
- Proof:

$$p = o' + t_0 d' = T^{-1}o + t_0 T^{-1}d$$

$$Tp = TT^{-1}o + t_0 TT^{-1}d = o + t_0 d = p'$$

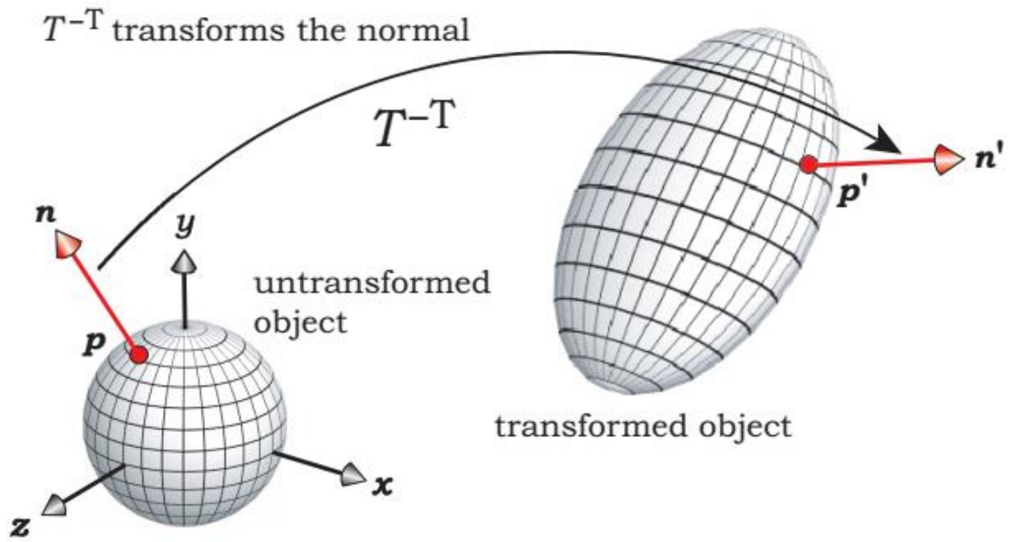
ILLINOIS

Transforming Normals

- Let n be the normal on the untransformed object
  - Found at hit point with inverse transformed ray
- Let n’ be the normal on the transformed object: **$n' = (T^{-1})^T n$**
- If you only use uniform scaling and rotations **$(T^{-1})^T = T$**

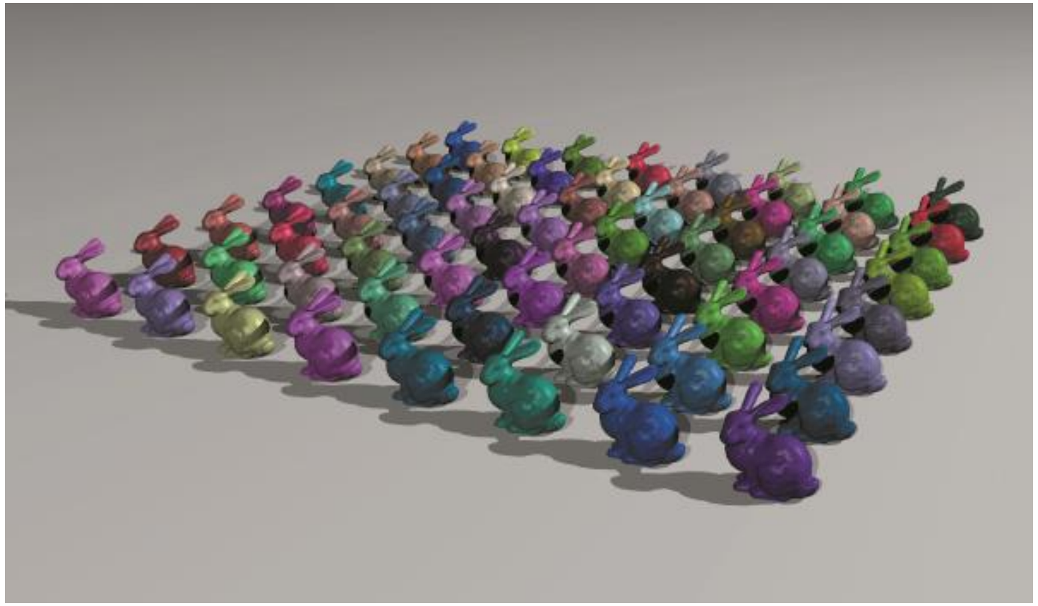
- Use homogeneous coordinate of 0

- Normalize n’ after you find it



Instancing

- Keep a single original geometric model
- Create instance models which consist of
  - Reference to original model
  - Inverse transformation matrix
  - Material



ILLINOIS

Grids and Transformed Objects

- Store references to instances just as you would normal objects
  - For a mesh this would be references to individual triangles
  - …and references to a material and inverse transformation matrix
- How can you compute the bounding box for the instances?
  - Why do you need the bounding box?
  - What information do you need to compute it?

Grids and Transformed Objects