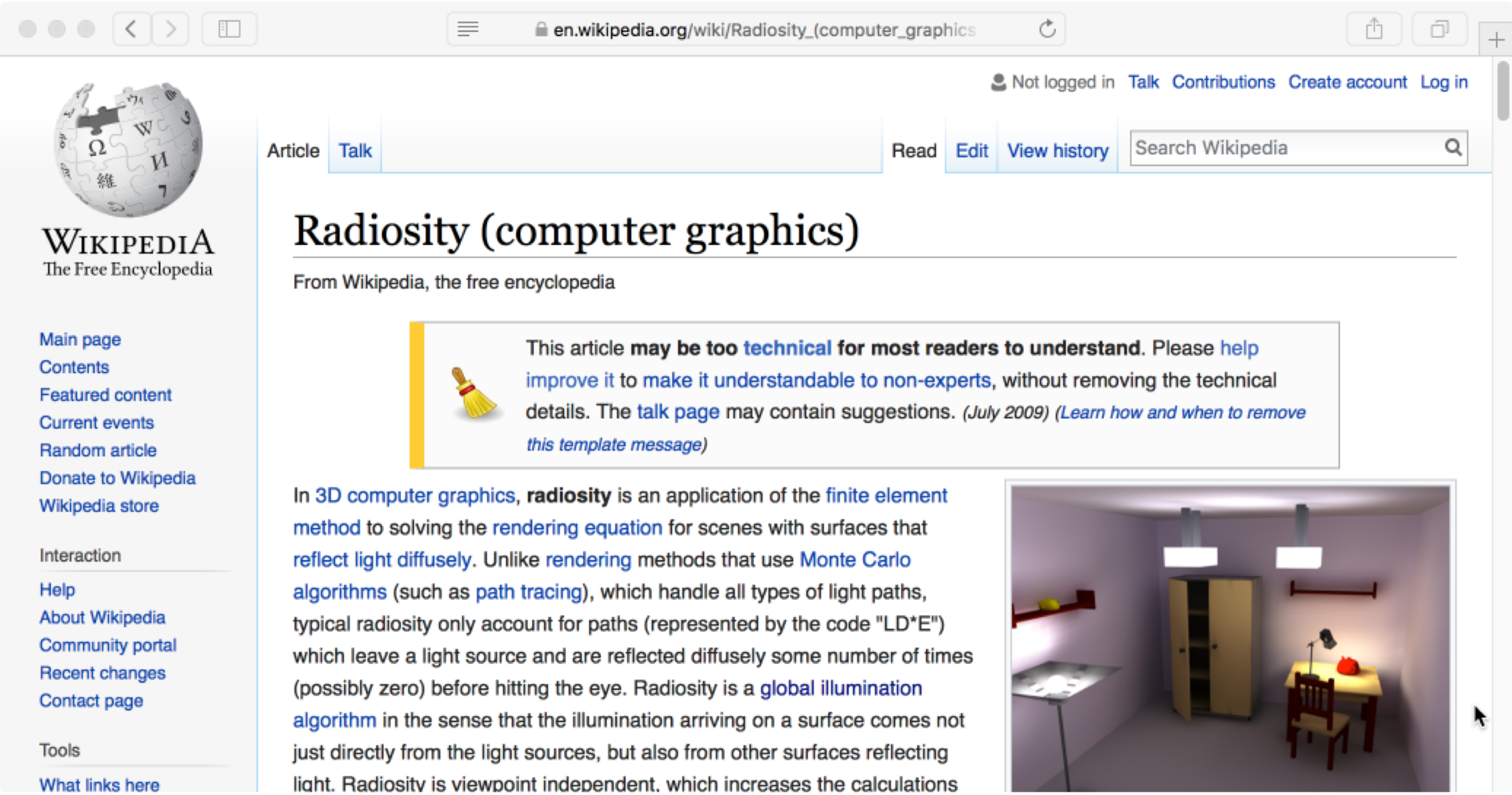
Lecture 02:
Basic Geometry

August 29, 2019

An Aside: Radiosity is Not Ray Tracing



en.wikipedia.org/wiki/Radiosity_(computer_graphics

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here

Radiosity (computer graphics)

From Wikipedia, the free encyclopedia

This article **may be too technical for most readers to understand**. Please help improve it to make it understandable to non-experts, without removing the technical details. The talk page may contain suggestions. *(July 2009) (Learn how and when to remove this template message)*

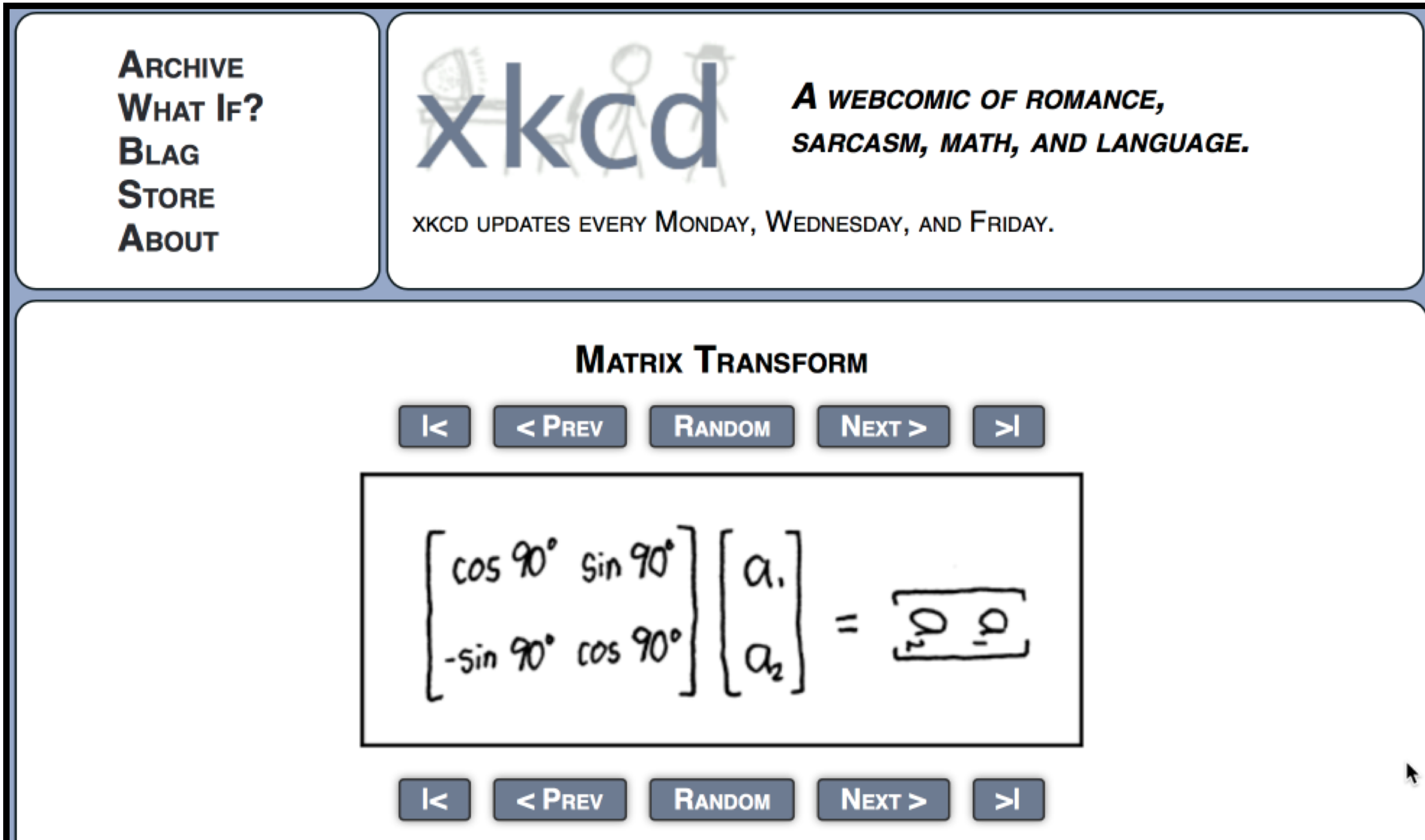
In 3D computer graphics, **radiosity** is an application of the finite element method to solving the rendering equation for scenes with surfaces that reflect light diffusely. Unlike rendering methods that use Monte Carlo algorithms (such as path tracing), which handle all types of light paths, typical radiosity only account for paths (represented by the code "LD*E") which leave a light source and are reflected diffusely some number of times (possibly zero) before hitting the eye. Radiosity is a global illumination algorithm in the sense that the illumination arriving on a surface comes not just directly from the light sources, but also from other surfaces reflecting light. Radiosity is viewpoint independent, which increases the calculations

Know what radiosity computation does. Do not expect to implement nor see underlying equations this semester.

Example Courtesy of Nikolay Radaev

Rendered using 3D Studio Max (+VRay Plugin)

Now – The Journey to 2D Rotation



Vectors, Points & Matrices

- The geometry for graphics rests upon
  - Scalars, Vectors, Points, and Matrices
- And why ? The short answer.
  - Objects are collections of points
  - Light rays are vectors
  - Objects & Light interact in Euclidean spaces
  - Placement in space is done using matrices
- Now for the longer answer…

But let's start with… Scalars

- Scalar - a number.
  - Two Operations -
    - Addition, Multiplication.
  - Axioms
    - Associative
    - Commutative
    - Invertible
  - Invertible implies
    - Subtraction
    - Division

$$\alpha + \beta = \beta + \alpha$$

$$\alpha \cdot \beta = \beta \cdot \alpha$$

$$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$$

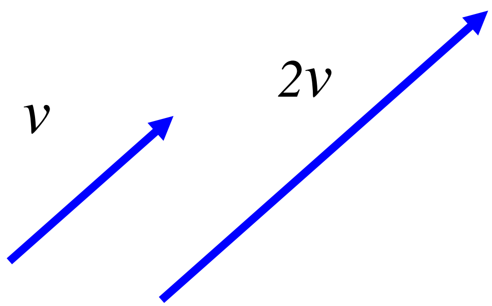
$$\alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma$$

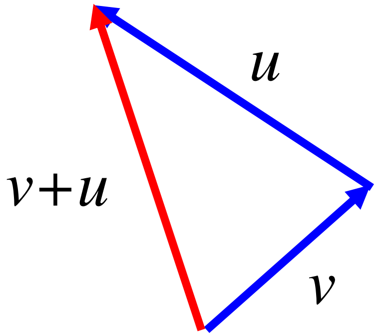
$$\alpha \cdot (\beta + \gamma) = \alpha \cdot \beta + \alpha \cdot \gamma$$

Vectors

- Vector - direction and magnitude
  - Two Operations -
    - Scalar-vector multiplication
    - Vector-vector addition
  - Often expressed as an n-tuple of scalars.

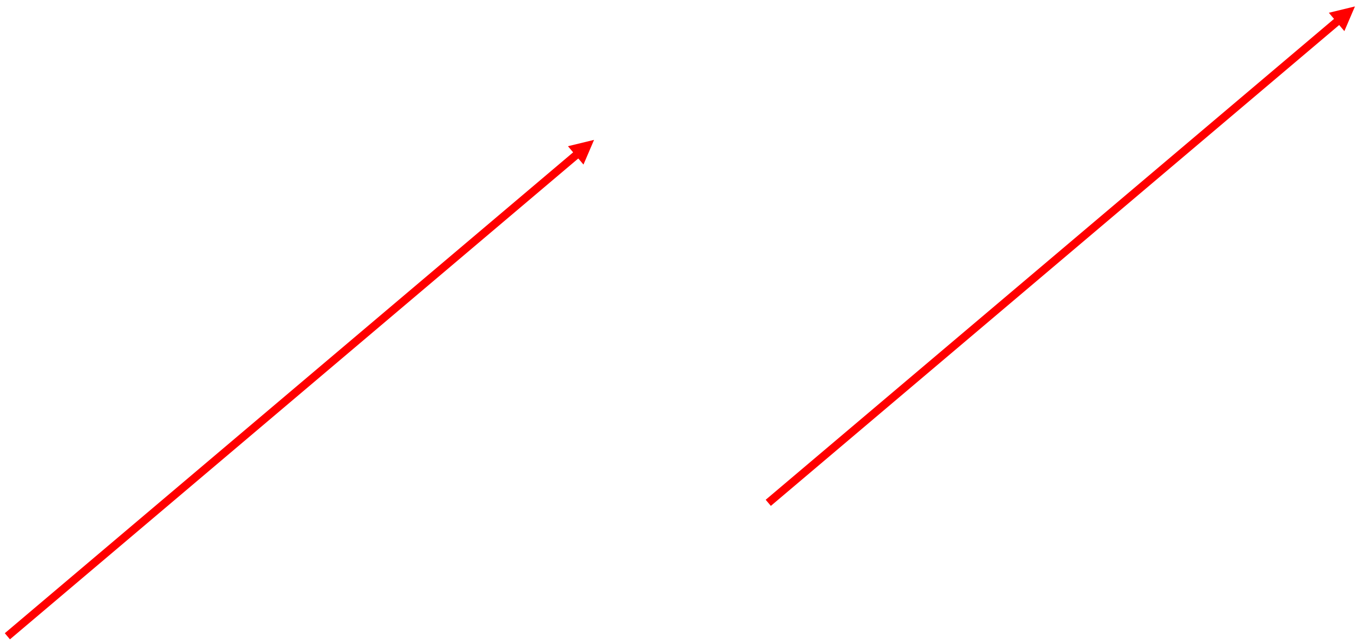
$$v = \left| v_1,\ v_2,\ v_3,\ \ldots\ , v_n \right|$$





Test: Do you Get It?

- Are these two vectors the same?



Vector Spaces

- Combinations of vectors generate new vectors.

$$u = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \alpha_3 \cdot v_3$$

for example ...

$$u = \begin{vmatrix} 3 \\ 4 \\ 1 \end{vmatrix} = 1 \cdot \begin{vmatrix} 1 \\ 0 \\ -1 \end{vmatrix} + 2 \cdot \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} + 4 \cdot \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} \quad \text{or} \quad u = 3 \cdot \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} + 4 \cdot \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} + 1 \cdot \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix}$$

Key Vector Space Concepts

- Span
  - The space of all vectors that can be created by linear combinations of a set of vectors
- Basis Vectors
  - A set of vectors that span a space
  - Generally focus on basis vectors that are
    - Orthogonal to each other (independent axes)
    - Unit length
- What is lacking?
  - Location, distance, angles.

Vectors beg "Where are we?"

Directly over the center of the Earth?

- More seriously, vector spaces lack location
  - Location requires an origin: a reference.
- Vector spaces have no origin.
- Now let us introduce points.
  - A point is **not** the same thing as a vector!
- New operations
  - Point-point subtraction yields a vector.
  - A point plus a vector yields a point.

Point + Vector = Point

– Linear combinations of basis vectors

– … and a specified origin - a point.

$$P = O + \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \alpha_3 \cdot v_3$$

for example ...

$$P = \begin{vmatrix} 7 \\ 4 \\ 3 \end{vmatrix} = \begin{vmatrix} 2 \\ 2 \\ 2 \end{vmatrix} + 5 \cdot \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} + 2 \cdot \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} + 1 \cdot \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix}$$

Typically, we think of the origin as being at [0,0,0], but that somewhat confuses the real meaning of an origin.

With an origin, you always know where you are (relatively).

Tricky Question

- I present you with:

$$a = \begin{vmatrix} 1 \\ 3 \end{vmatrix}$$

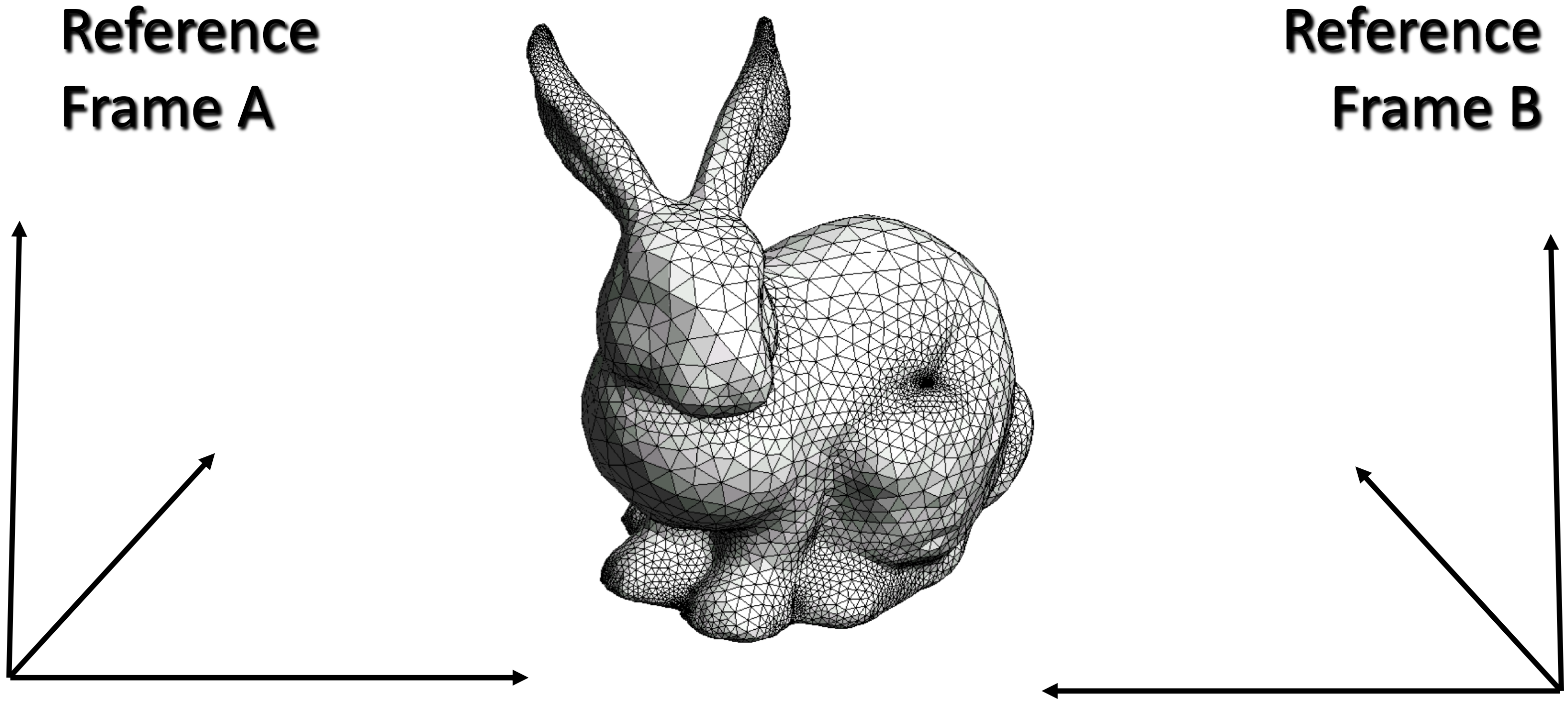
- Is this a vector?
- Is this a point?
- How can you tell?

And a related question …

- Do Points Exist Without Coordinates?
- The answer is – yes!
  - Just ask the Stanford Bunny (see next slide)
- Why does this matter …

In graphics, keeping the intrinsic geometry of objects separate from their coordinate manifestation in a particular frame of reference is essential.

Same Point - By Example



- The Stanford Bunny has intrinsic properties.
  - Independent of reference frame A (or B).
  - Changing reference does **not** change the Bunny.

But don't I need numbers

stanford-bunny.dae

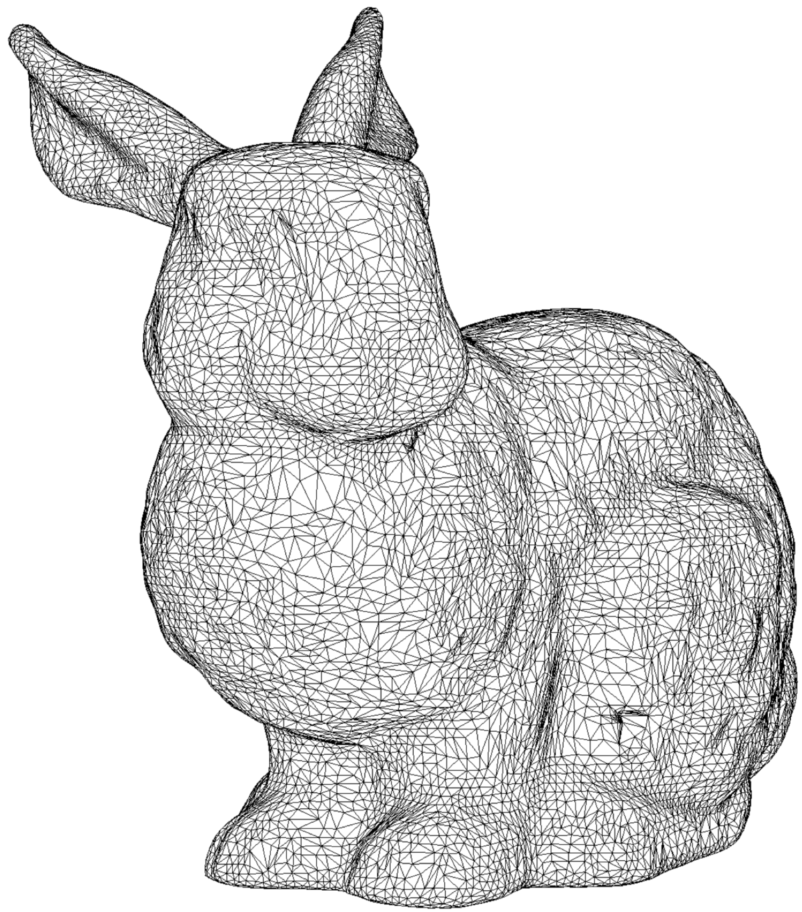
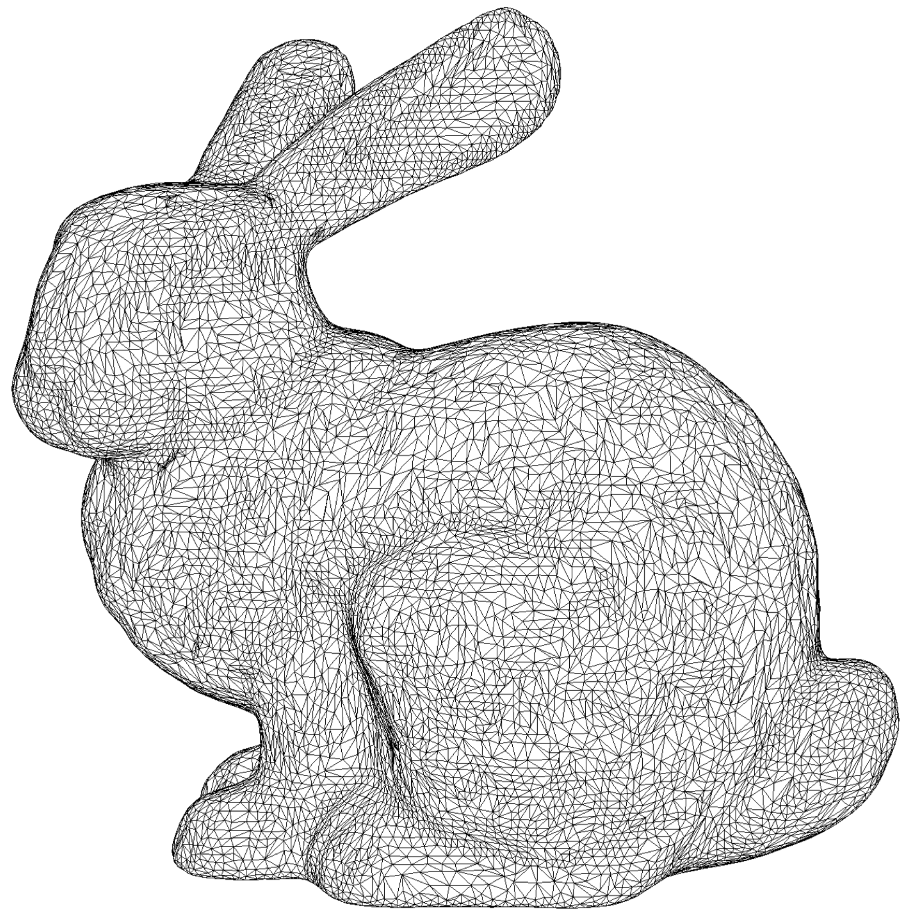

```
  <library_geometries>
    <geometry id="Stanford-Bunny-lib" name="Stanford-BunnyMesh">
      <mesh>
        <source id="Stanford-Bunny-POSITION">
          <float_array id="Stanford-Bunny-POSITION-array" count="45774">
-230.728104 4.686089 63.804157
-227.490784 0.933628 58.946880
-226.531693 0.776154 63.782352
-224.395508 -3.028207 60.970978
-229.676010 4.971947 68.160980
-223.717422 -3.661769 63.714046
-224.162476 -4.185090 57.642754
-225.297577 0.927746 67.325806
-236.843719 10.587538 70.057274
-223.672424 -8.049369 60.428425
-224.319489 -3.761209 54.235916
-226.032654 3.732646 71.860245
-232.944321 12.148590 81.932228
-222.571320 -3.757358 66.912155
-222.646317 -8.040578 65.364716
-224.177475 -8.026236 51.465748
-227.580811 8.416955 81.837120
-222.975342 -0.335431 69.704033
-237.337769 14.674938 79.679108
-223.921432 -12.268403 64.042381
-225.104568 -12.658540 59.631546
-223.789444 3.984940 77.875832
-222.776337 0.893496 73.074867
-232.791306 14.286700 89.040634
```


Somewhere, some how, don't we need to specify triples; the x, y, z coordinates of each 3D vertex.

Intrinsic Vs. Extrinsic

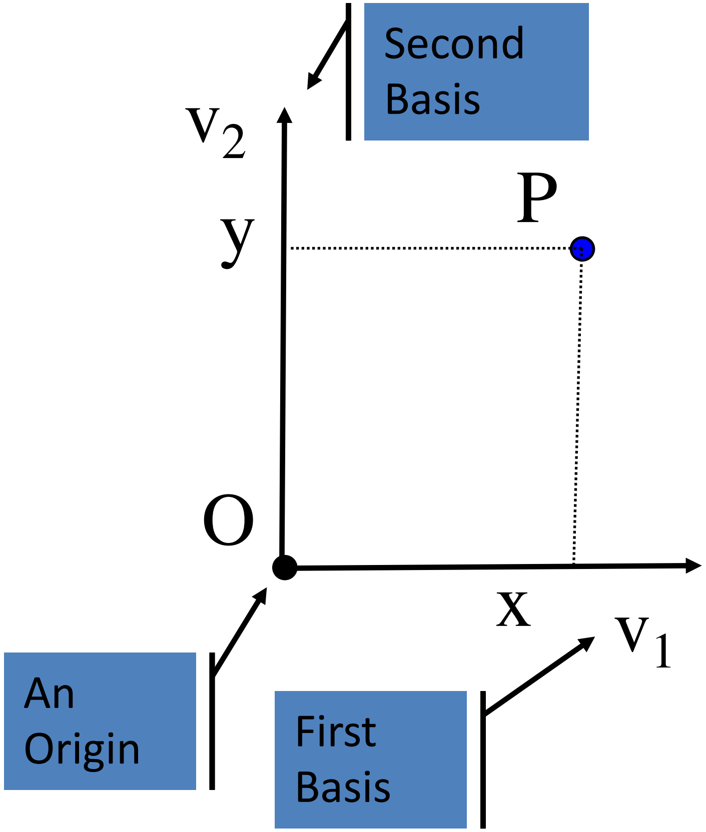
- What matters is the relation of the data to the reference frame.
  - Moving the Bunny toward the reference point is the same as moving the reference point toward the Bunny
  - The same Bunny can be expressed in different reference frames
- "World Coordinates" aren't special
  - As long as all the data is expressed relative to the same reference frame

Same Bunny?

Where is a Point Revisited

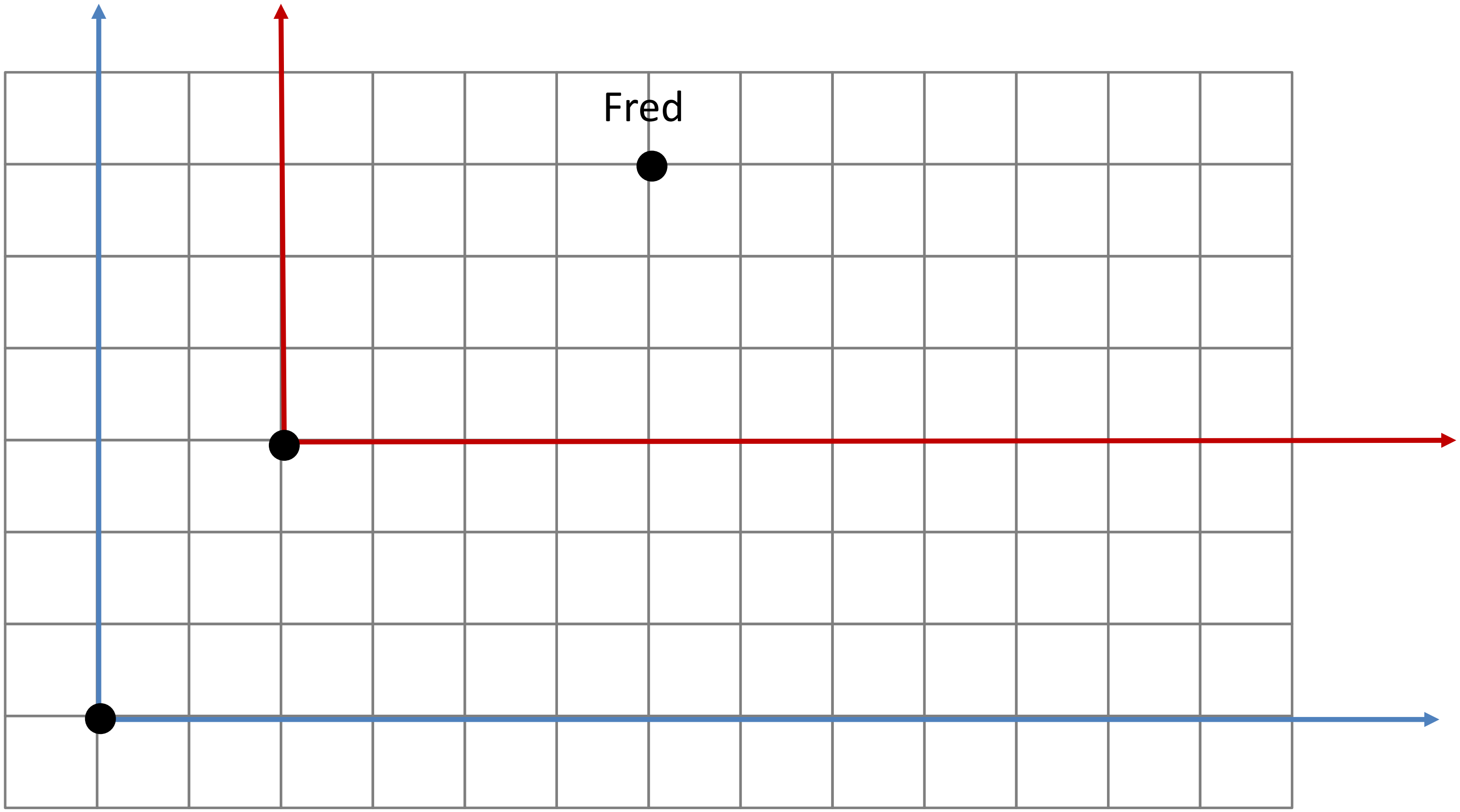
- To specify a point in a Euclidean space.



$$P = O + x v_1 + y v_2$$

$$\begin{vmatrix} p_x \\ p_y \end{vmatrix} = \begin{vmatrix} x_o \\ y_o \end{vmatrix} + x \begin{vmatrix} 1 \\ 0 \end{vmatrix} + y \begin{vmatrix} 0 \\ 1 \end{vmatrix}$$

A Point named Fred



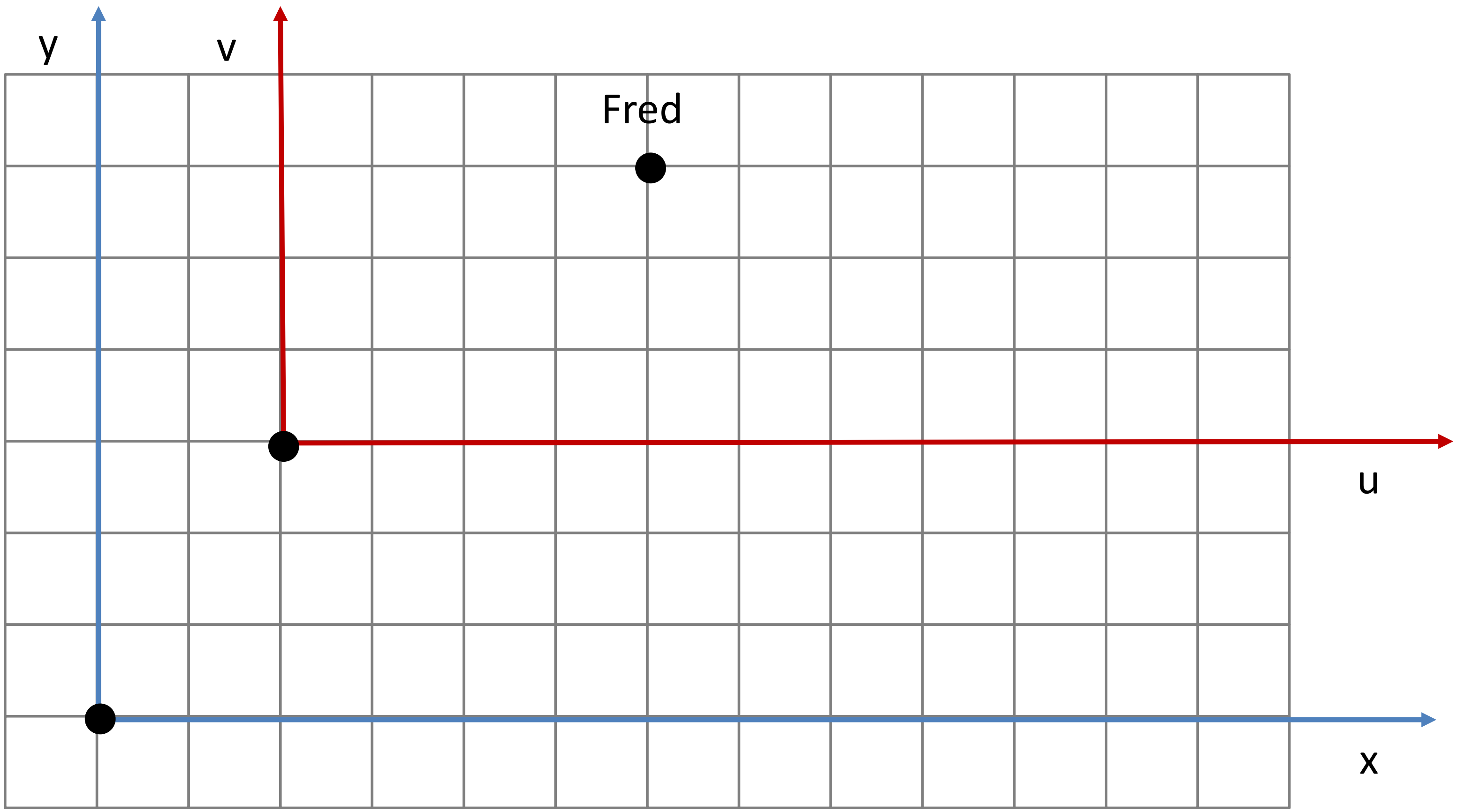
Which is it? $Fred = \begin{vmatrix} 6 \\ 6 \end{vmatrix}$ or $Fred = \begin{vmatrix} 4 \\ 3 \end{vmatrix}$

2D Translation

- Think about the previous example
- Can you decide between
  - Fred was moved down and to the left.
  - Reference frame was moved up and to the right.
- Generally you cannot
- More important

Often in graphics it is equally valid, or even preferable, to think of movement as shifting a reference frame rather than moving an object.

2D Translation - Moving Fred



$\begin{vmatrix} x \\ y \end{vmatrix} = \begin{vmatrix} 6 \\ 6 \end{vmatrix}$ and $\begin{vmatrix} u \\ v \end{vmatrix} = \begin{vmatrix} 4 \\ 3 \end{vmatrix}$ as translation $\begin{vmatrix} 6 \\ 6 \end{vmatrix} = \begin{vmatrix} 4 \\ 3 \end{vmatrix} + \begin{vmatrix} 2 \\ 3 \end{vmatrix}$

Written as we are used to seeing it: $P' = P + T$

Euclidean Space

- Euclidean Space adds a new operation, the *dot product* (inner product).
- You all know the algebraic definition.

$$u \cdot v = \sum_i u_i v_i \qquad |v| = \sqrt{v \cdot v}$$

- Do you know its geometric interpretation?

$$u \cdot v = |u||v|\cos(\theta)$$

- From the dot product - distances and angles

Dot Product as Projection

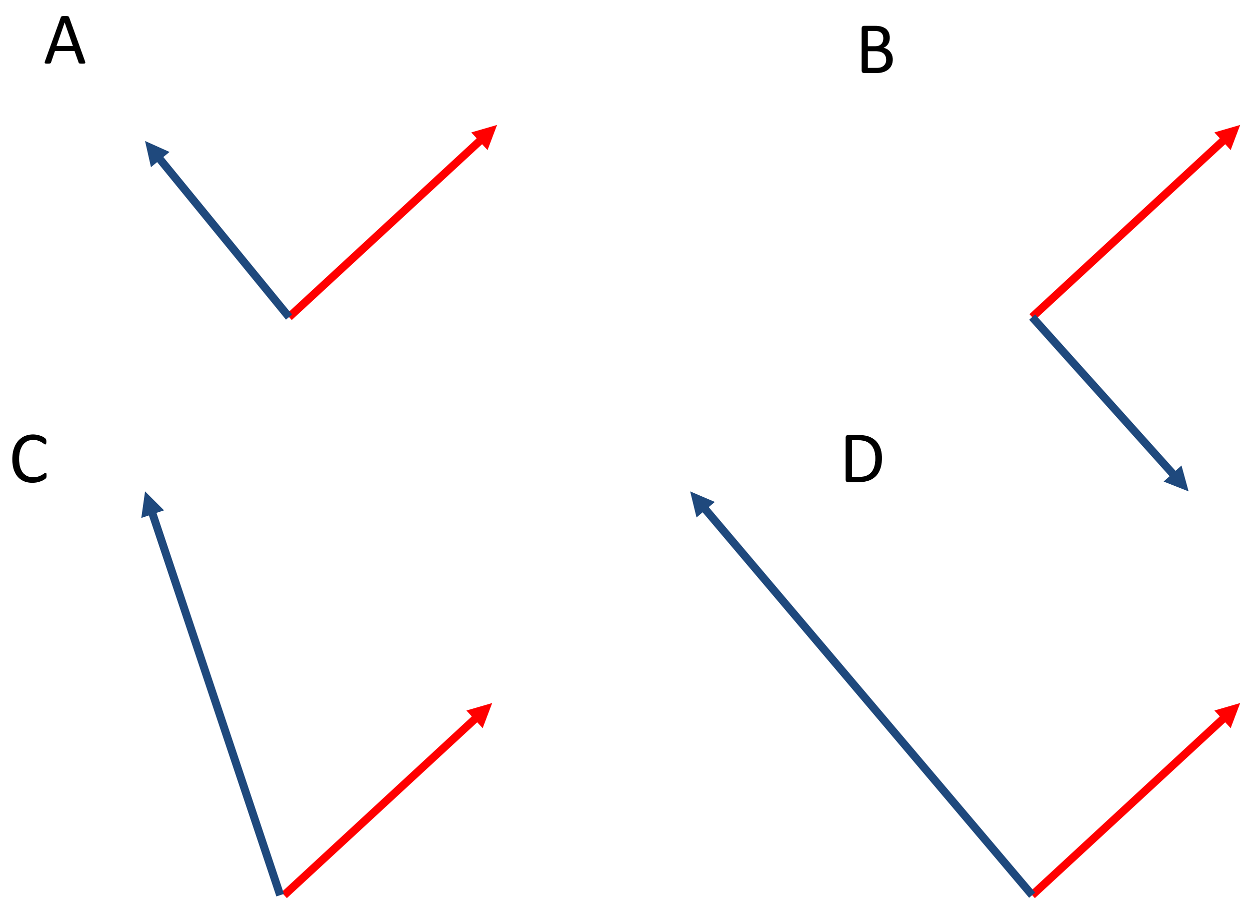
- To start, set origin at zero. Now observe.

$$\begin{vmatrix} p_x \\ p_y \end{vmatrix} = x \begin{vmatrix} 1 \\ 0 \end{vmatrix} + y \begin{vmatrix} 0 \\ 1 \end{vmatrix} \Rightarrow x = \begin{vmatrix} p_x \\ p_y \end{vmatrix} \cdot \begin{vmatrix} 1 \\ 0 \end{vmatrix}, \;\; y = \begin{vmatrix} p_x \\ p_y \end{vmatrix} \cdot \begin{vmatrix} 0 \\ 1 \end{vmatrix}$$

The distance of a point from the origin along a dimension, i.e. along a basis vector, is measured by a dot product between the point and the basis vector

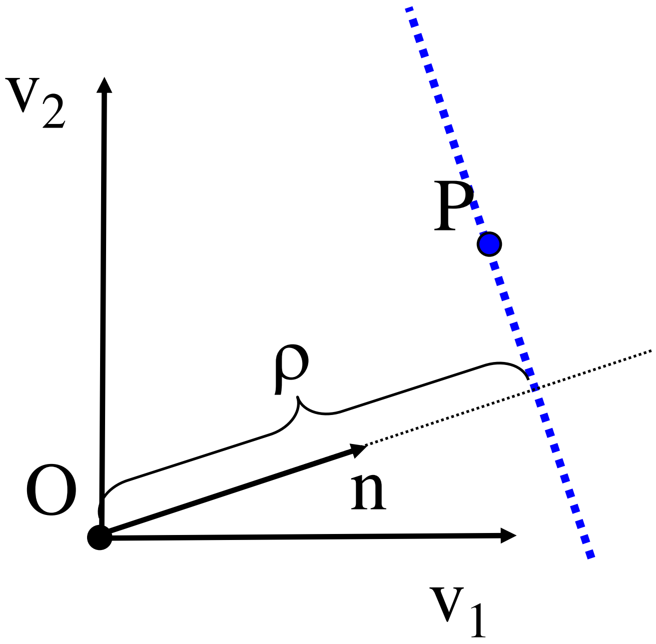
About Orthogonality

$$u \cdot v = |u|\,|v| \cos(\theta) \qquad \cos(90°) = 0$$



Know & Love Dot Products 1

- An easy way to define a line …



$$L \Rightarrow F(x,y) = 0$$

$$n \cdot L - \rho = 0 \qquad n \cdot n = 1$$

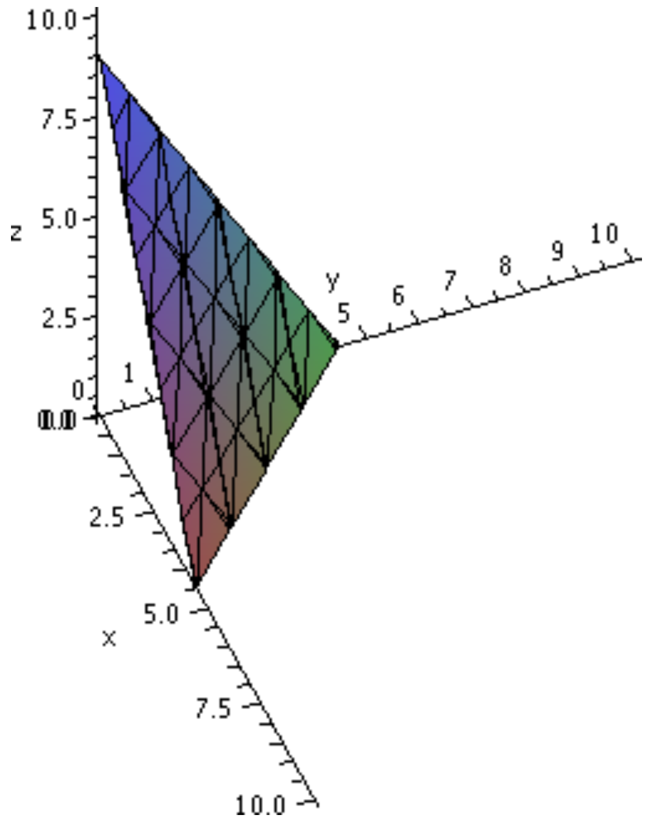
$$\rho = n \cdot P = n_x p_x + n_y p_y$$

$$\begin{vmatrix} n_x \\ n_y \end{vmatrix} \cdot \begin{vmatrix} x \\ y \end{vmatrix} - \rho = 0$$

$$n_x x + n_y y - \rho = 0$$

And in 3D

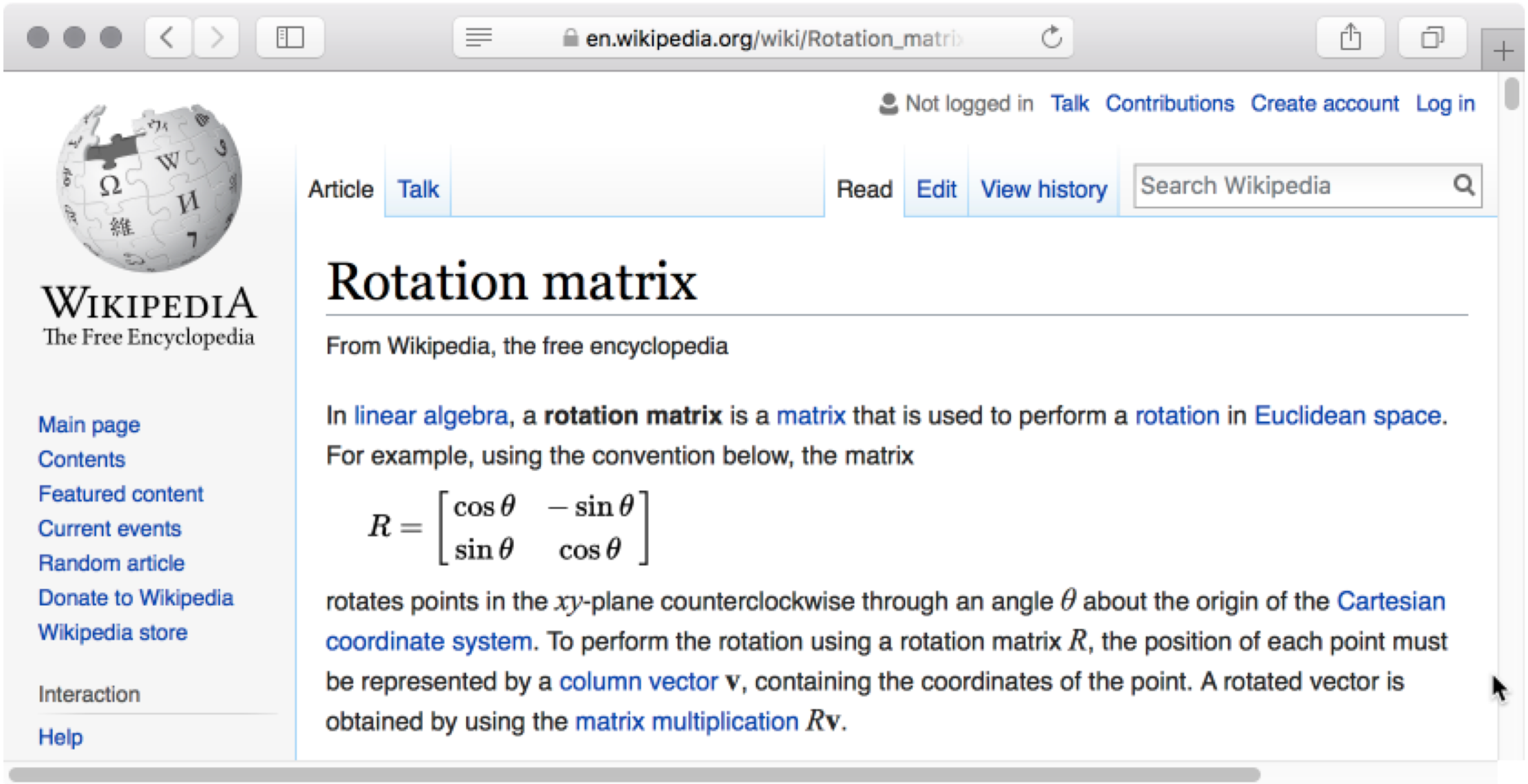
Riddle: *What do you call all points a distance of 3 from the origin measured in a direction defined by a vector n?*



$$\left[P = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, n = \begin{bmatrix} \frac{2}{3} \\ \frac{2}{3} \\ \frac{1}{3} \end{bmatrix} \right]$$

$$F = \frac{2}{3}x + \frac{2}{3}y + \frac{1}{3}z - 3$$

Further Dot Product Motivation


en.wikipedia.org/wiki/Rotation_matrix

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help

Article Talk Read Edit View history Search Wikipedia

Rotation matrix

From Wikipedia, the free encyclopedia

In linear algebra, a **rotation matrix** is a matrix that is used to perform a rotation in Euclidean space. For example, using the convention below, the matrix

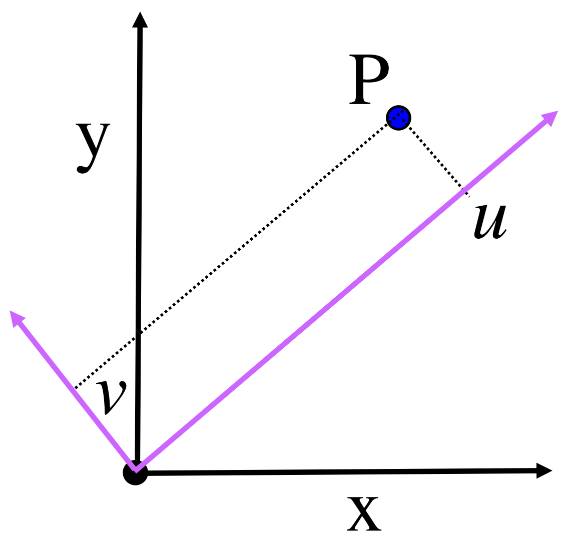
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

rotates points in the xy-plane counterclockwise through an angle θ about the origin of the Cartesian coordinate system. To perform the rotation using a rotation matrix R, the position of each point must be represented by a column vector $\mathbf{v}$, containing the coordinates of the point. A rotated vector is obtained by using the matrix multiplication $R\mathbf{v}$.

Above you see how almost all texts and courses introduction 2D rotation.
This is entirely correct, but there is a more intuitive way to understand rotation.

Know & Love Dot Products 2

- Consider an alternate basis



$$\begin{vmatrix} x \\ y \end{vmatrix} = u \begin{vmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{vmatrix} + v \begin{vmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{vmatrix}$$

$$\begin{vmatrix} u \\ v \end{vmatrix} = \begin{vmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$

Welcome to 2D Rotation

These are the same!

$$\begin{vmatrix} u \\ v \end{vmatrix} = \begin{vmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$

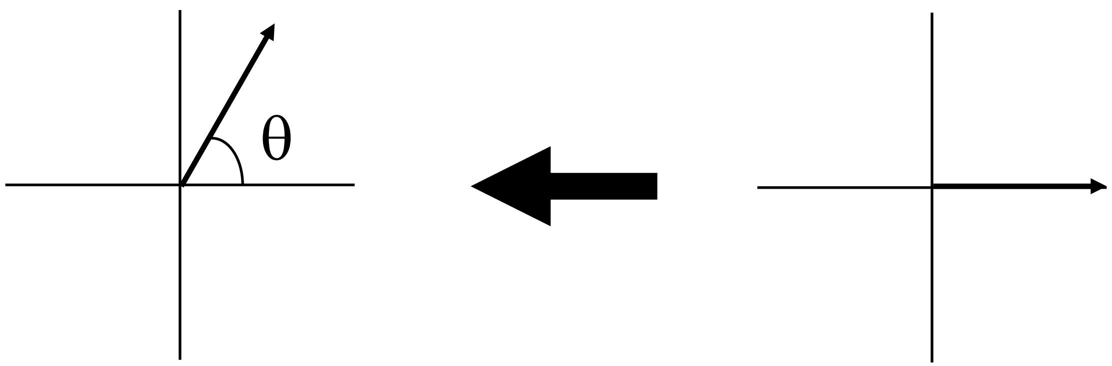
$$\begin{vmatrix} u \\ v \end{vmatrix} = \begin{vmatrix} cos(45^{\circ}) & sin(45^{\circ}) \\ -sin(45^{\circ}) & cos(45^{\circ}) \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$

Rotate by θ

$$M = RP$$

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta)\,x - \sin(\theta)\,y \\ \sin(\theta)\,x + \cos(\theta)\,y \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



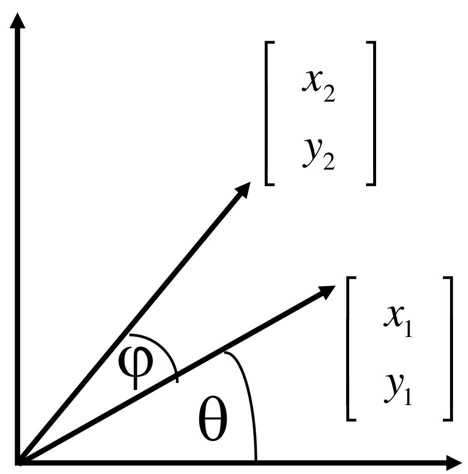
Does this make sense, given the geometry of the dot product?

More Standard Approach

Derivation of Rotation Matrix

$$x_1 = r\cos(\theta) \qquad x_2 = r\cos(\theta+\phi)$$

$$y_1 = r\sin(\theta) \qquad y_2 = r\sin(\theta+\phi)$$



Derivation (cont.)

Magic Trig Identity:

$$\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$$

$$\sin(a+b) = \sin(a)\cos(b) + \sin(b)\cos(a)$$

$$x_2 = r\cos(\theta + \phi)$$

$$x_2 = r\cos(\theta)\cos(\phi) - r\sin(\theta)\sin(\phi)$$

$$x_2 = x_1\cos(\phi) - y_1\sin(\phi)$$

Note: the process for y_2 is the same

The End