3D geometry

Georgia Tech College of Computing
Interactive & Intelligent Computing

- Points, vectors, operators
- Intersections
- Poses, changes of coordinates systems
- Homogeneous transforms
- Screws

Updated September 27, 2011

Introduction

- **Points** are associated with locations of space
- **Vectors** represent displacements between points or directions
- Points, vectors, and **operators** that combine them are the common tools for solving many geometric problems that arise in Geometric Modeling, Computer Graphics, Animation, Visualization, and Computational Geometry.
- The three most fundamental operators on vectors are the **dot** product, the **cross** product, and the **mixed** product (sometimes called the triple product).
- Although a **point** and a **vector** may be represented by its three coordinates, they are of **different** type and should not be mixed
- **Avoid**, whenever possible, using **coordinates** when formulating geometric constructions. Instead use vectors and points.

What you are expected to learn here

- Properties of dot, cross, and mixed products
- How to write simple tests for
  - 4-points and 2-lines co-planarity
  - Intersection of two coplanar edges
  - Parallelism of two edges or of two lines
  - Clockwise orientation of a triangle from a viewpoint
  - Positive orientation of a tetrahedron
  - Edge/triangle and ray/triangle intersection
- How to compute
  - Center of mass of a triangle
  - Volume of a tetrahedron
  - “Shadow” (orthogonal projection) of a vector on a plane
  - Line/plane intersection
  - Plane/plane intersection
  - Plane/plane/plane intersection
- How to change coordinate systems
- How to compute, combine, and apply transformations

Terminology

- “Normal to” = **“Orthogonal** to” = “forms a 90^o angle with”
- **“Norm** of a vector” = “length of the vector”
- “Triangle **normal**” = “vector normal to plane containing triangle ”
- “Are **coplanar”** = “there is a plain containing them”
- “Are **collinear**” = “there is a line containing them”

Vectors

- Vector space (analytic definition from Descartes in the 1600s)
  - $\underline{U}+\underline{V}=\underline{V}+\underline{U}$ (vector **addition**)
  - $\underline{U}+(\underline{V}+\underline{W})=(\underline{U}+\underline{V})+\underline{W}$
  - $\underline{0}$ is the zero (or **null**) vector if $\forall\underline{V}$, $\underline{0}+\underline{V}=\underline{V}$
  - $-\underline{V}$ is the **inverse** of $\underline{V}$, so that the vector **subtraction** $\underline{V}-\underline{V}=\underline{0}$
  - $s(\underline{U}+\underline{V})=s\underline{U}+s\underline{V}$ (**scaling**, scalar multiplication)
  - $\underline{U}/s$ is the scalar division (same as multiplication by s^{-1})
  - $(a+b)\underline{V}=a\underline{V}+b\underline{V}$
- **Vectors** are used to represent **displacement** between points
- Each vector $\underline{V}$ has a **norm** (length) denoted $||\underline{V}||$
  - $\underline{V} / ||\underline{V}||$ is the unit vector (length 1) of $\underline{V}$. We denote it $\underline{V}_u$
  - If $||\underline{V}||==0$, then $\underline{V}$ is the null vector $\underline{0}$
- **Unit vectors** are used to represent
  - **Basis vectors** of a coordinate system
  - **Directions** of tangents or normals in definitions of lines or planes

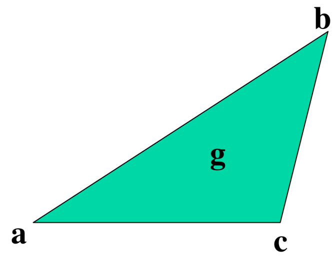
Points

- **Affine** space
  - Defined together with a vector space and the point difference mapping
  - V=B–A (vector V is the displacement from point A to point B)
  - Notation: AB stands for the vector B–A
  - Point B may be defined as A+AB
  - Pick A as origin: mapping between a point B and the vector AB.
  - AB+BC=AC, since (C–B) + (B–A) = C–A
  - We may extend the vector operations to points, but usually only when the result is **independent of the choice of the origin**!
    - A+B is not allowed (the result depends on the origin)
    - (A+B)/2 is OK
- Points are often used to represent
  - The vertices of a triangle or polyhedron
  - The origin of a coordinate system
  - Points in the definition of lines or plane

Center of mass of a triangle

- What is the center of mass g of triangle area (a,b,c)?

$\mathbf{g} = (\mathbf{a}+\mathbf{b}+\mathbf{c})/3$

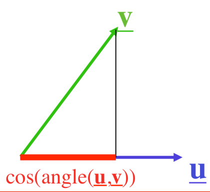


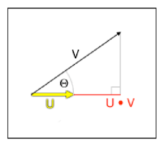
Dot Product

U•V denotes the **dot** product (also called the **inner** product)

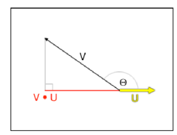
U•V = ||U||·||V||·cos(angle(U,V))

- U•V is a **scalar**.
- U•V==0 ⇒ (U==0 or V==0 or (U and V are orthogonal))
- U•V is positive if the angle between U and V is less than 90^o
- Note that U•V = V•U, because: cos(a)=cos(–a).
- u•v = cos(angle(u,v))   # unit vectors: ||u|| ≡ ||v|| ≡ 1
  - the dot product of two unit vectors is the cosine of their angle
- V•u = measues the the orthogonal projection of V onto the direction of u
- ||U||= sqrt(U•U) = length of U = norm of U





V•**U** = **U**•**V** > 0 here



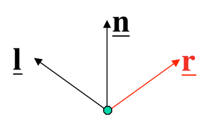
V•**U** = **U**•**V** < 0 here

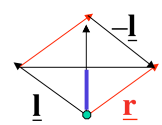
Computing the **reflection** vector

- Given the unit normal n to a mirror surface and the unit direction l towards the light, compute the direction r of the reflected light.

By symmetry, **l**+**r** is parallel to **n** and has a norm that is twice the length of the projection **n**•**l** of **l** upon **n**. Hence:

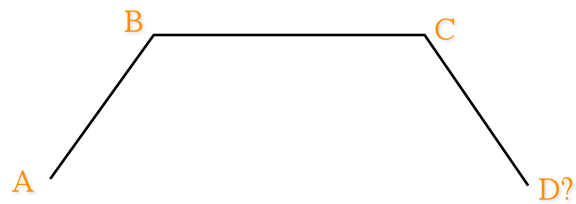
r = 2(**n**•**l**)**n**–**l**



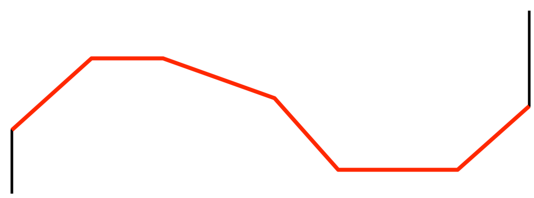


Extending a polygon

- How to construct D as a combination of A, B, and C



- Useful for extending polygons



Computing a "shadow" vector (projection)

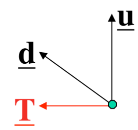
- Given the unit up-vector $\underline{u}$ and the unit direction $\underline{d}$, compute the shadow $\underline{T}$ of $\underline{d}$ onto the floor (orthogonal to $\underline{u}$).

$\underline{\mathbf{T}}$ and $\underline{\mathbf{u}}$ are orthogonal.

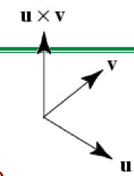
$\underline{\mathbf{d}}$ is the vector sum $\underline{\mathbf{T}}+(\underline{\mathbf{d}}\bullet\underline{\mathbf{u}})\underline{\mathbf{u}}$.

Hence:

$$\underline{\mathbf{T}} = \underline{\mathbf{d}} - (\underline{\mathbf{d}}\bullet\underline{\mathbf{u}})\underline{\mathbf{u}}$$



Cross product



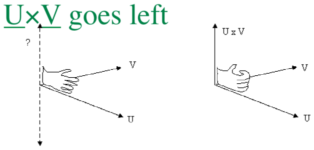
U×V denotes the cross product (it is a **vector**)

U×V is either 0 or a vector orthogonal to both U and V

- When U==0 or V==0 or U//V (**parallel**) then U×V= 0
- Otherwise, U×V is orthogonal to both U and V

The direction of U×V is defined by the thumb of the right hand

- Curling the fingers from U to V
  - Or standing parallel to U and looking at V, U×V goes left



||U×V|| ≡ ||U||·||V||·**sin**(angle(U,V))

- **sin**(angle(u,v))^2 ≡ 1–(u•v)^2 # unit vectors
- (u×v== 0) ⇔ u//v (parallel)
- U×V ≡ –V×U
- Useful identity: U×(V×W) ≡ (U•W)V – (U•V)W

When are **two edges parallel** in 3D?

- **Edge**(a,b) **is parallel to Edge**(c,d) **if** $\mathbf{ab} \times \mathbf{cd} == \underline{0}$

Mixed product

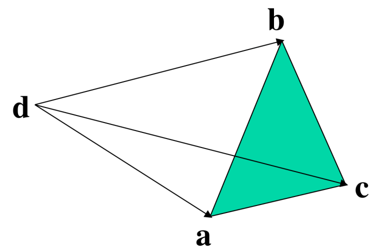
$\underline{U}\bullet(\underline{V}\times\underline{W})$ is called a **mixed product**

- $\underline{U}\bullet(\underline{V}\times\underline{W})$ is a scalar
- $\underline{U}\bullet(\underline{V}\times\underline{W})$==0 when one of the vectors is null or all 3 are coplanar
- $\underline{U}\bullet(\underline{V}\times\underline{W})$ is the **determinant** | $\underline{U}$ $\underline{V}$ $\underline{W}$ |
- $\underline{U}\bullet(\underline{V}\times\underline{W}) \equiv \underline{V}\bullet(\underline{W}\times\underline{U}) \equiv - \underline{U}\bullet(\underline{W}\times\underline{V})$ # cyclic permutation

Testing whether a triangle is **front-facing**

- When does the triangle a, b, c appear clockwise from d?

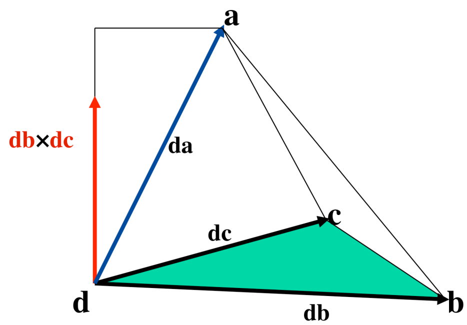
when $\mathbf{da}\bullet(\mathbf{db}\times\mathbf{dc}) > 0$

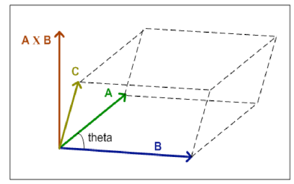


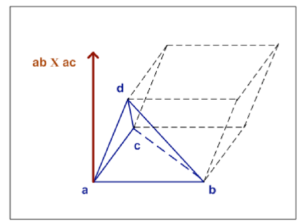
Volume of a tetrahedron

Volume of a tetrahedron with vertices a, b, c and d is

$$v \equiv |\, \mathbf{da} \bullet (\mathbf{db} \times \mathbf{dc}) \,| \,/\, 6$$







z, *s*, and v functions

zero: *z*(a,b,c,d) ≡ da•(db×dc)**==0** ***# tests co-planarity of 4 points***

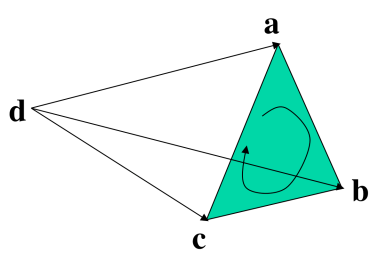
- Returns Boolean ***TRUE*** **when** a,b,c,d **are** coplanar

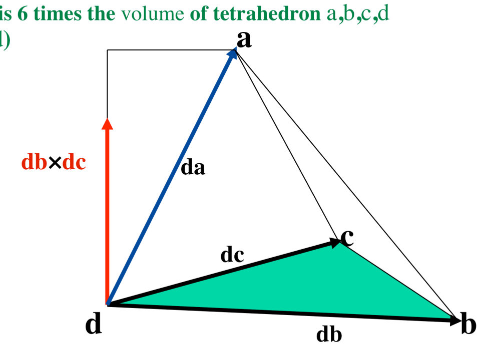
sign: ***s***(a,b,c,d) ≡ da•(db×dc)**>0** ***# test orientation or side***

- Returns Boolean ***TRUE*** **when** a,b,c **appear clockwise from** d
- **Used to test whether d in on the "good" side of plane through** a,b,c

value*:** **v**(a,b,c,d) ≡ da•(db×dc) ***# compute volume

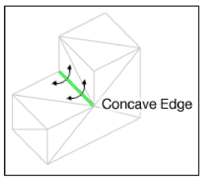
- Returns scalar **whose absolute value is 6 times the** volume **of tetrahedron** a,b,c,d
  - **v**(a,b,c,d) ≡ **v**(d,a,b,c) ≡ –**v**(b,a,c,d)

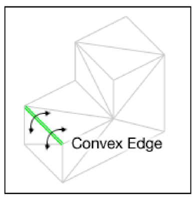


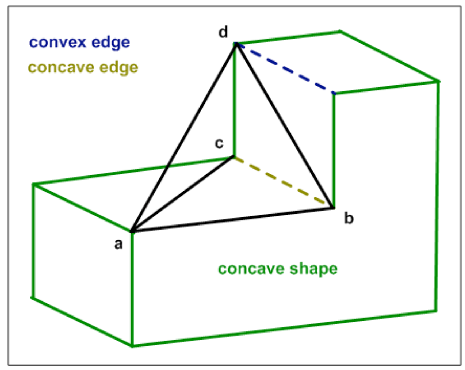


Testing whether an edge is **concave**

- How to test whether the edge (c,b) shared by triangles (a,b,c) and (c,b,d) is concave?



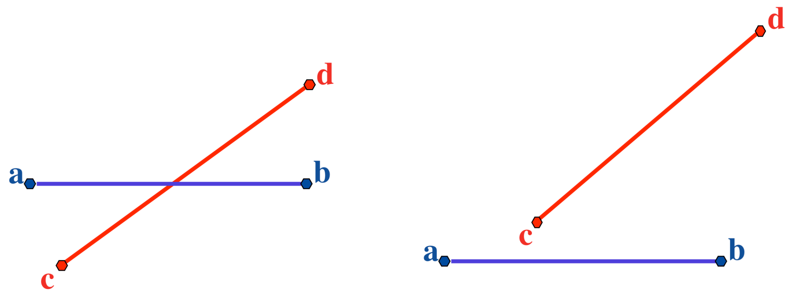




When do two edges **intersect** in 2D?

- Write a geometric expression that returns *true* when two coplanar edges (a,b) and (c,d) intersect

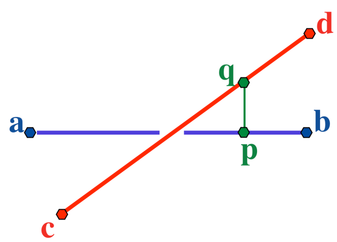
$0 \geq (\mathbf{ab}\times\mathbf{ac})\bullet(\mathbf{ab}\times\mathbf{ad})$ and $0 \geq (\mathbf{cd}\times\mathbf{ca})\bullet(\mathbf{cd}\times\mathbf{cb})$

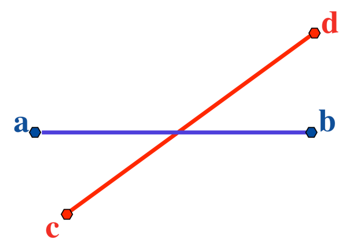


When do two edges **intersect** in **3D**?

Consider two edges: edge(a,b) and edge(c,d)

- When are they co-planar? When z**(a,b,c,d)**
- When do they intersect?
  When they are co-planar and
  0 ≥ **(ab×ac)•(ab×ad)** and 0 ≥ **(cd×ca)•(cd×cb)**

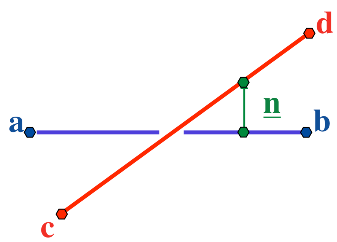




Common normal to two edges in 3D?

Consider two edges: edge(a,b) and edge(c,d)

What is their common normal n? $\underline{\mathbf{n}}=(\mathbf{ab}\times\mathbf{cd})_{\mathbf{u}}$



Use it to compute the shortest distance between two lines, or even between two edges

When is a point inside a tetrahedron?

When does point p lie inside tetrahedron a, b, c, d ?

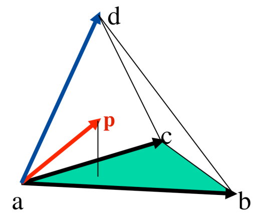
- Assume *z*(a,b,c,d) is *FALSE* (not coplanar)

When

s(**p**,**b**,**c**,**d**),
s(**a**,**p**,**c**,**d**),
s(**a**,**b**,**p**,**d**), and
s(**a**,**b**,**c**,**p**)

are identical.

(i.e. all return TRUE or all return FALSE)



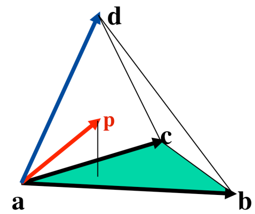
- When is p on the boundary of the tetrahedron?
  - Do as an exercise for practice.

A faster point-in-tetrahedron test ?

- Suggested by Nguyen Truong
  - Write ap = sab+tac+uad
  - Solve for s, t, u (linear system of 3 equations)
  - Requires 17 multiplications, 3 divisions, and 11 additions
  - Check that s, t, and u are positive and that s+u+t<1
- A variation:

  Compute s, t, u, w
  - w = v(a,b,c,d)
  - s = v(a,p,c,d)
  - u = v(a,b,p,d)
  - t = v(a,b,c,p)

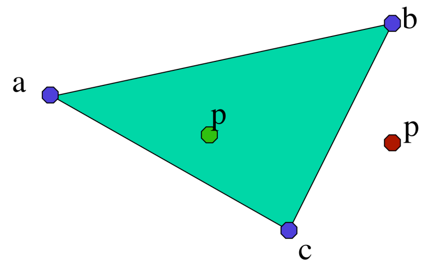
  Check that
  - w, s, t, u have the same sign and that s+u+t<w



When is a 3D point inside a triangle

- When does point p lie inside triangle with vertices a, b, c ?

When z(p,a,b,c) **and** **(ab×ap)•(bc×bp)**≥0 and **(bc×bp)•(ca×cp)**≥0

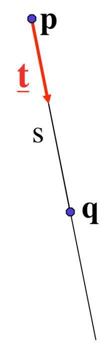


Parametric representation of a **line**

- Let Line(p,t) be the line through point p with tangent vector t
- Its parametric form associates a scalar s with a point q(s) on the line
  - s defines the distance from p to q(s)

    $$q(s) = p + st$$

  - Note that the direction of t gives an **orientation** to the line (direction where s is positive)
- We can represent a line by p and t

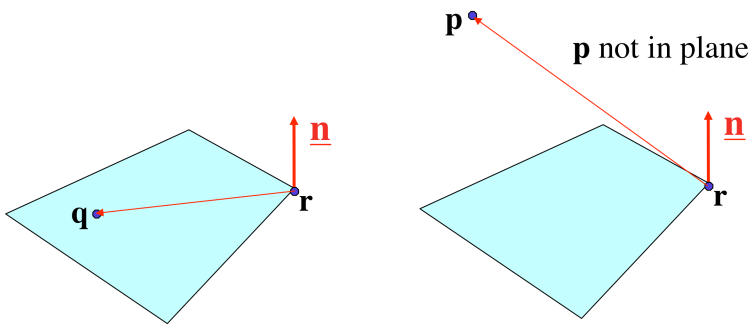


Implicit representation of a **plane**

- Let Plane(r,<u>n</u>) be the plane through point r with normal <u>n</u>
- A point q lies on Plane(r,<u>n</u>) when rq•<u>n</u>=0

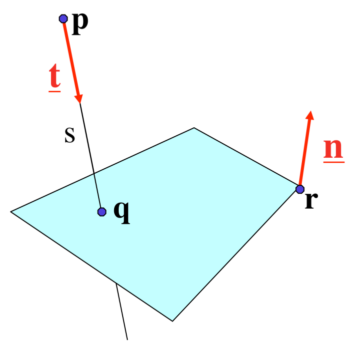
  rq = q – r

  using –<u>n</u> changes the orientation of the plane



Line/plane intersection

Compute the intersection q between Line(p,t) and Plane(r,n)



Replacing **q** by **p**+s**t** in **(q–r)•n**=0 yields **(p–r**+s**t)•n**=0

Solving for s yields

rp•n+s**t•n**=0 and

s = – **rp•n / t•n**

s = **pr•n / t•n**

Hence **q = p** + (**pr•n**)**t** / (**t•n**)

When are two lines coplanar in 3D?

- L(p,$\underline{t}$) and L(q,$\underline{u}$) are coplanar

When $\underline{\mathbf{t}} \times \underline{\mathbf{u}} == \underline{0}$ OR $\underline{\mathbf{pq}} \bullet (\underline{\mathbf{t}} \times \underline{\mathbf{u}}) == 0$

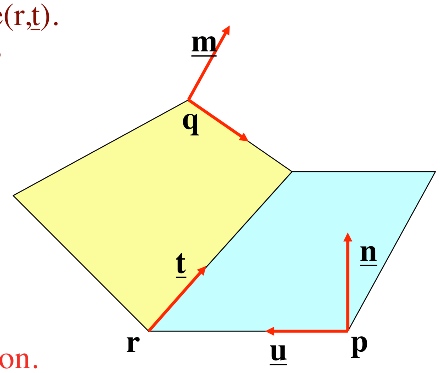
What is the **intersection** of **two planes**

- Consider two planes Plane(p,n) and Plane(q,m)
- Assume that n and m are not parallel (i.e. $\underline{n}\times\underline{m} \neq 0$)
- Their intersection is a line Line(r,t).
- How can one compute r and t ?

$\underline{\mathbf{t}} := (\underline{\mathbf{n}}\times\underline{\mathbf{m}})_u$

let $\underline{\mathbf{u}} := \underline{\mathbf{n}}\times\underline{\mathbf{t}}$

$\mathbf{r} := \mathbf{p}+(\mathbf{pq}\bullet\underline{\mathbf{m}})\underline{\mathbf{u}}/(\underline{\mathbf{u}}\bullet\underline{\mathbf{m}})$



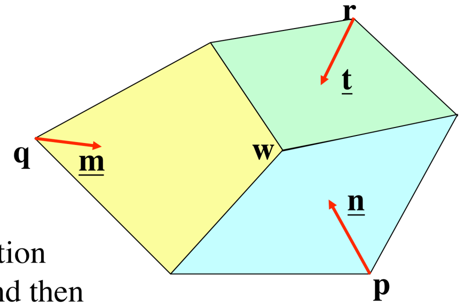
If correct, then provide the derivation.

Otherwise, provided the correct answer.

What is the **intersection** of **three planes**

- Consider planes Plane(p,m), Plane(q,n), and Plane(r,m).
- How do you compute their intersection w ?

Write **w**=**p**+a**n**+b**m**+c**t** then solve the linear system {**pw**•**n**=0, **qw**•**m**=0, **rw**•**t**=0} for a, b, and c



or compute the **line** of intersection between **two** of these planes and then **intersect** it with the **third** one.

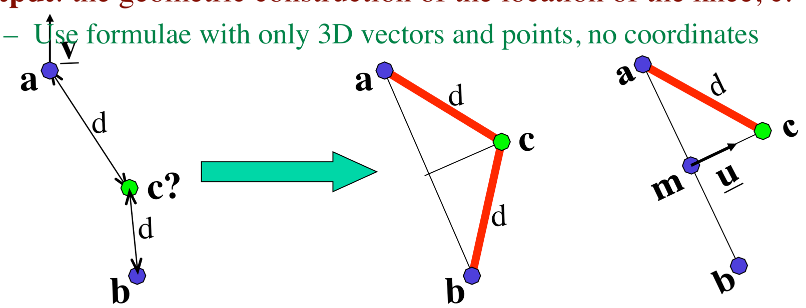
Practice problem

Input: Two points in 3D, a and b, length d, and up-direction v.

Objective: Design a single-articulation leg that has its hip at a, its toe at b, and has two limbs of equal length, d. Its knee is at point c. Furthermore, we want the leg to lie in a vertical plane and want the knee to be up (not down).

Output: the geometric construction of the location of the knee, c.

- Use formulae with only 3D vectors and points, no coordinates



Implementation

- **Points and vectors are each represented by their 3 coordinates**
  - p=(p_x,p_y,p_z) **and** $\underline{v}$=(v_x,v_y,v_z)
- $\underline{U} \bullet \underline{V} = U_x V_x + U_y V_y + U_z V_z$
- $\underline{U} \times \underline{V} = (U_y V_z - U_z V_y) - (U_x V_z - U_z V_x) + (U_x V_y - U_y V_x)$

- Implement**:**
  - Points and vectors
  - Dot, cross, and mixed products
  - z, s, and v functions from mixed products
  - Edge/triangle intersection
  - Lines and Planes
  - Line/Plane, Plane/Plane, and Plane/Plane/Plane intersections
    - Return exception for singular cases (parallelism…)

What do **coordinates** mean?

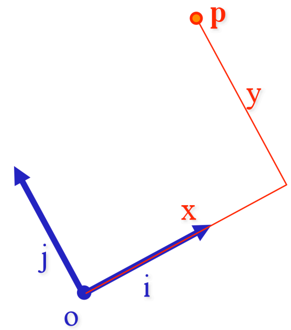
- Let C be a **coordinate system** with origin o and basis (i, j)
  - i and j are unit length vectors orthogonal to each other
- Let **point** p have **coordinates** (x,y) in C
- Define p in terms of x, y, o, i, j:

p = o + x**i** + y**j**

start at o

make x steps in the **i** direction

make y steps in the **j** direction



How do we **represent** points and vectors?

Using an ordered set of coordinates in an agreed-upon, *yet not specified*, **global coordinate system**

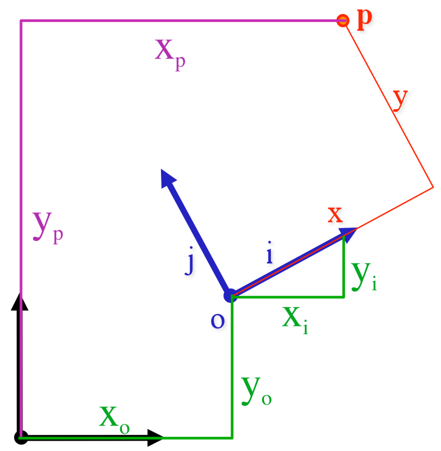
$\mathbf{o} = (x_o, y_o)$

$\mathbf{i} = (x_i, y_i)$

$\mathbf{j} = (x_j, y_j)$

$\mathbf{p} = (x_p, y_p)$

$\mathbf{p} = x\mathbf{i} + y\mathbf{j} + \mathbf{o}$



Operations on points and vectors

- Subtracting two points yields a vector (displacement)

  $(x_p , y_p) - (x_o , y_o) = (x_p - x_o , \; p_o - y_o)$
- Adding a vector to a point moves the point

  $(x_o , y_o) + (x_j , y_j) = (x_o + x_j , \; y_o + y_j)$
- Adding two vectors adds the corresponding coordinates

  $(x_i , y_i) + (x_j , y_j) = (x_i + x_j , \; y_i + y_j)$
- Multiplying a vector by a scalar multiplies its length

  $s(x_i , y_i) = (sx_i , sy_i)$
- The **dot product** of two vectors is the sum of the products of corresponding terms

  $(x_i , y_i) \bullet (x_j , y_j) = (x_i x_j + y_i y_j)$

Transformations

- Homogeneous (projective space):
    - 4x4 matrix transform, then normalization: (x/w, y/w, z/w)
- Affine:
    - 3x3 matrix transform, then translation: (x+a, y+b, z+c)
  - Similarity:
    - Combination of rotation, translation, and uniform scale
  - Rigid
    - Combinations of rotation and translation
  - Screw
    - Combinations of rotation and translation along same axis
- Linear:
    - 3x3 matrix transform: Combination of rotation and non-uniform scale
  - Spiral
    - Combination of rotation and non-uniform scal

What is a **matrix**?

- A scalar or ordered set of matrices
  - scalars, vectors, m×n tables are matrices

For example (i , j , o) is a 2×3 matrix often presented as a table (list of vectors and point)

$$\mathbf{(i\ ,j\ ,o\ )} = \begin{pmatrix} \mathbf{x_i} & \mathbf{x_j} & \mathbf{x_o} \\ \mathbf{y_i} & \mathbf{y_j} & \mathbf{y_o} \end{pmatrix}$$

What is a **matrix-vector product**?

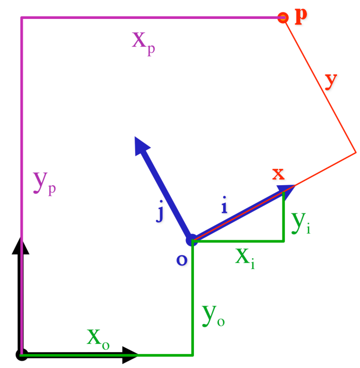
Generalization of the dot product:

Sum of products of corresponding terms

$(\mathbf{a}\ ,\mathbf{b}\ ,\mathbf{c}\ )\bullet(\ x\ ,y\ ,z\ ) = x\mathbf{a}+y\mathbf{b}+z\mathbf{c}$

What is the **matrix form** of (x_p, y_p) ?

$\mathbf{p} = x\mathbf{i}+y\mathbf{j}+\mathbf{o} = (\mathbf{i}\ \mathbf{j}, \mathbf{o}) \bullet (x,y,1)$



$$
\begin{aligned}
(x_p, y_p) &= x(x_i, y_i) + y(x_j, y_j) + (x_o, y_o) \\
&= (xx_i, xy_i) + (yx_j, yy_j) + (x_o, y_o) \\
&= (xx_i+yx_j+x_o,\ xy_i+yy_j+y_o)
\end{aligned}
$$

$$
\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_i & x_j & x_o \\ y_i & y_j & y_o \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
$$

What is the **homogeneous** representation?

- Add a third coordinate in 2D (4th in 3D):

  1 for points, 0 for vectors

$$\mathbf{o} = (x_o, y_o, 1)$$

$$\mathbf{i} = (x_i, y_i, 0)$$

$$\mathbf{j} = (x_j, y_j, 0)$$

$$\mathbf{p} = (x_p, y_p, 1)$$

Homogeneous **matrix form** of (x_p,y_p) ?

$p = xi + yj + o = (i , j , o) \bullet (x , y , 1)$

$(x_p , y_p , 1) = x(x_i , y_i , 0) + y(x_j , y_j , 0) + 1(x_o , y_o , 1)$

$= (xx_i+yx_j+x_o , xy_i+yy_j+ y_o , 1)$

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{pmatrix} x_i & x_j & x_o \\ y_i & y_j & y_o \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Performing the multiplication

- Multiply columns and add

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_i & x_j & x_o \\ y_i & y_j & y_o \\ 0 & 0 & 1 \end{bmatrix}$$

Bottom row is not used here. Will be used for perspective.

Compute (x,y) = apply inverse transform

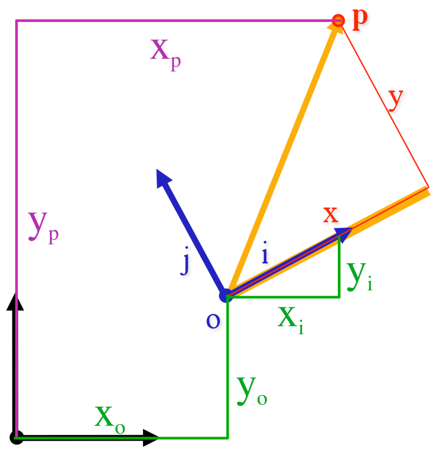
$o = (x_o, y_o), i = (x_i, y_i), j = (x_j, y_j), p = (x_p, y_p)$

$op = p - o$

$x = op \bullet i$

$y = op \bullet j$

v•i is the normal projection of vector v on the unit vector i



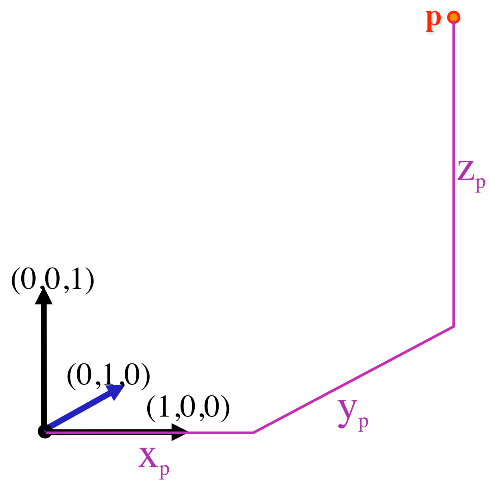
How to extend this to **3D** ?

$\mathbf{o} = (x_o, y_o, z_o, 1)$

$\mathbf{i} = (x_i, y_i, z_i, 0)$

$\mathbf{j} = (x_j, y_j, z_j, 0)$

$\mathbf{p} = (x_p, y_p, z_p, 1)$



Homogeneous Coordinates

- A 3D point is represented by a 4 element vector
- Determined by dividing the fourth component into the first 3

- The fourth component is 1 for points and 0 for vectors

$$\left(\frac{x}{w}, \frac{y}{w}, \frac{y}{w}\right) = [x, y, z, w]$$

$$(x, y, z) = [x, y, z, 1]$$

General transformation matrix

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & m \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

4x4 matrix formulation

$$S_{(a,b,c)} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{(d,e,f)} = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & e \\ 0 & 0 & 1 & f \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{(\underline{\mathbf{u}},\underline{\mathbf{v}},\underline{\mathbf{w}})} = \begin{bmatrix} \underline{u}_x & \underline{v}_x & \underline{w}_x & 0 \\ \underline{u}_y & \underline{v}_y & \underline{w}_y & 0 \\ \underline{u}_z & \underline{v}_z & \underline{w}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z(a) = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \underline{u}_x & \underline{v}_x & \underline{w}_x & d \\ \underline{u}_y & \underline{v}_y & \underline{w}_y & e \\ \underline{u}_z & \underline{v}_z & \underline{w}_z & f \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M \bullet P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u_x x + v_x y + w_x z + d \\ u_y x + v_y y + w_y z + e \\ u_z x + v_z y + w_z z + f \\ 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_x \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scale in x, y, z

$$\begin{bmatrix} S_x.x \\ S_y.y \\ S_y.z \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotations around the x-axis

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotations around the y-axis

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotations around the z-axis

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

General rotation

- $\|\underline{u}\|=\|\underline{v}\|=1$
- $\underline{u}\bullet\underline{v} = 0$
- $\underline{w}=\underline{u}\times\underline{v}$
- **R** = { $\underline{u}$ $\underline{v}$ $\underline{w}$ }, top-left 3x3 part of 4x4 matrix

$$\begin{bmatrix} u_x & v_x & w_x & \mathbf{0} \\ u_y & v_y & w_y & \mathbf{0} \\ u_z & v_z & w_z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

- **Rotation to map $\underline{z}$ to $\underline{w}$**
  - $\underline{U}$:=$\underline{y}\times\underline{w}$ or $\underline{x}\times\underline{w}$
  - $\underline{u}$:=U($\underline{U}$)
  - $\underline{v}$:=$\underline{w}$x$\underline{u}$
  - $\mathbf{R}_{\mathbf{ZtoW}}$={ $\underline{u}$ $\underline{v}$ $\underline{w}$ }
- **Rotation to map $\underline{w}$ to $\underline{z}$ axis**
  - ◣$\mathbf{R}_{\mathbf{ZtoW}}$
- **R' = ◣R, inverse = transpose**

$$\underbrace{\begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}}_{R} \xrightarrow{\text{inverse}} \underbrace{\begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix}}_{R'}$$

Compounding Transformations

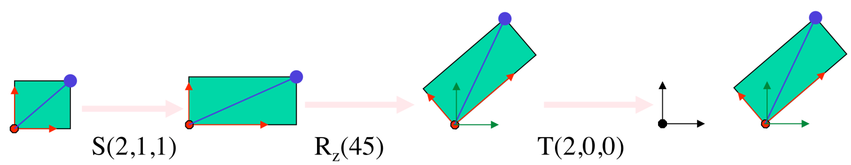
$$
\begin{aligned}
\widehat{P} &= M_1 M_2 M_3 M_4 M_5 M_6 P \\
M &= M_1 M_2 M_3 M_4 M_5 M_6 \\
\widehat{P} &= MP
\end{aligned}
$$

- Associative $(AB)C = A(BC)$
- NOT Commutative $AB \neq BA$

Cascading transformations

Two mental models:

1) Each transform is expressed in current frame and changes it
2) Transforms expressed in global frame & applied in reverse order

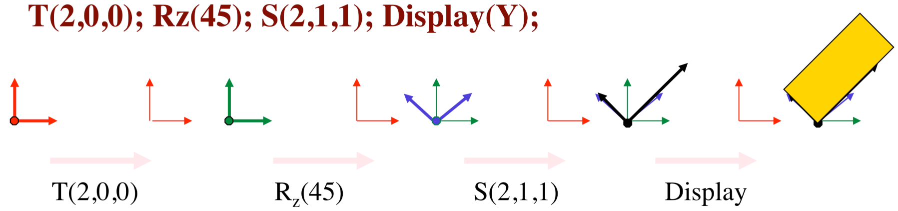


Execute API calls in reverse order**:**

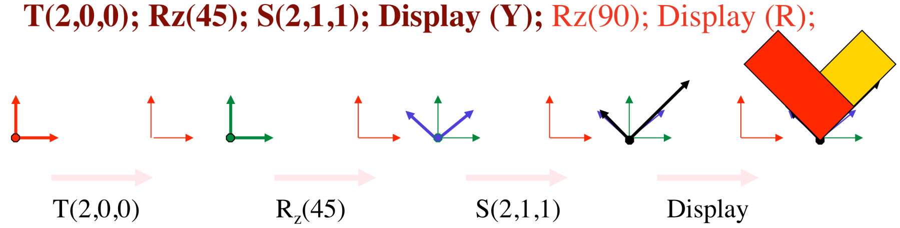
– **T(2,0,0); Rz(45); S(2,1,1); Display**

View operations as changing the canvas

T(2,0,0); Rz(45); S(2,1,1); Display(Y);

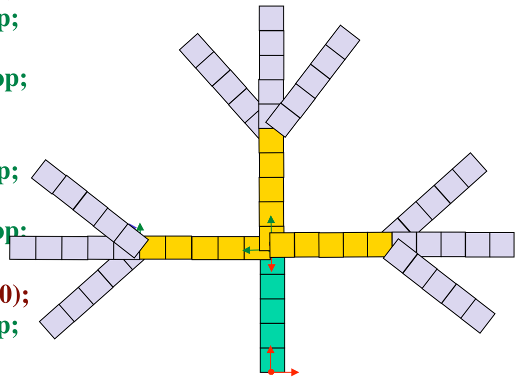


T(2,0,0); Rz(45); S(2,1,1); Display (Y); Rz(90); Display (R);



Using the stack

- **Show: {T(-0.5,0,0); S(1,5,1); ShadeCube(1,1,1)}**
- **Show; T(0,5,0);**
- **Push; R_z(90);Show; T(0,5,0);**
  - **Push; R_z(45); Show; Pop;**
  - **Show;**
  - **Push; R_z(-45); Show; Pop;**
  - **Pop;**
- **Push; Show; T(0,5,0);**
  - **Push; R_z(45); Show; Pop;**
  - **Show;**
  - **Push; R_z(-45); Show; Pop;**
  - **Pop;**
- **Push; R_z(-90); Show; T(0,5,0);**
  - **Push; R_z(45); Show; Pop;**
  - **Show;**
  - **Push; R_z(-45); Show; Pop;**
  - **Pop;**



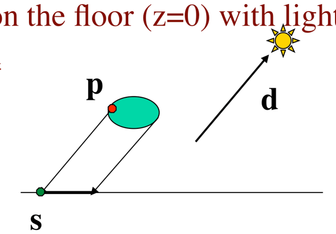
Displaying a tree from a single cone

- Tree(rec)
  - IF rec=0 THEN RETURN;
  - Display cone of length L along the z-axis
  - Push; T(0,0,3L/5); $R_X(42)$; $R_Z(63)$;S(0.72); Tree(rec-1); Pop;
  - Push; T(0,0,4L/6); $R_Z(67)$; $R_X(32)$; $R_Z(131)$;S(0.91); Tree(rec-1); Pop;
  - Push; T(0,0,5L/7); $R_Z(-17)$; $R_X(26)$; $R_Z(-33)$;S(0.79); Tree(rec-1); Pop;

Oblique floor shadow

- Compute the shadow **s** of point **p** on the floor (z=0) with light coming from infinity in direction **d**

$\mathbf{s}=\mathbf{p}-p_z\underline{\mathbf{d}}/d_z$



Write the shadow projection as a 4x4 matrix

1	0	$-d_x/d_z$	0
0	1	$-d_y/d_z$	0
0	0	0	0
0	0	0	1

No tilt viewing coordinate system

- Given
  - a viewing direction z̲ and
  - an upward vertical direction u̲,

  compute two vectors x̲ and y̲, so that

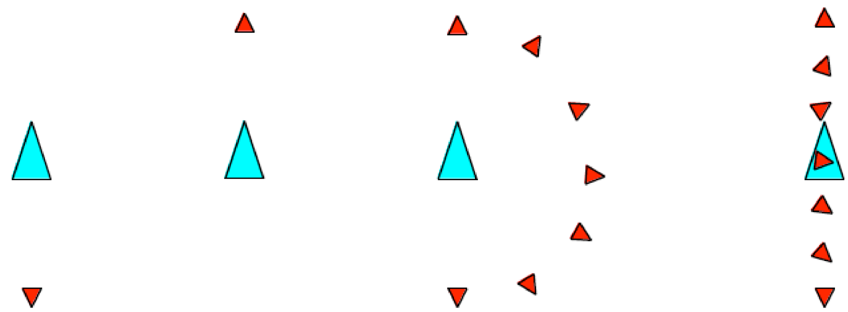
  (x,y,z) form an orthonormal basis in which

  the projection of u onto the screen spanned by (x,y) is parallel to y (vertical appears vertical: no tilt)

$$\underline{\mathbf{x}} := (\underline{\mathbf{u}} - (\underline{\mathbf{u}} \bullet \underline{\mathbf{z}})\underline{\mathbf{z}})_{\mathbf{u}}$$

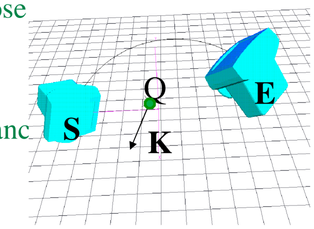
$$\underline{\mathbf{y}} := -\underline{\mathbf{x}} \times \underline{\mathbf{z}}$$

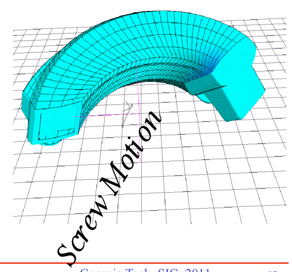
Motion independent of CS



The universal screw-motion

- Screw motions are great!
  - Uniquely defined by start pose S and end pose
  - Independent of coordinate system
  - Subsumes pure rotations and translations
  - Minimizes rotation angle & translation distanc
  - Natural motions for many application
- Simple to apply for any value of t in [0,1]
  - Rotation by angle ***tb*** around axis Axis(**Q**,**K**)
  - Translation by distance ***td*** along Axis(**Q**,**K**)
  - Each point moves along a **helix**
- Simple to compute from poses S and E
  - Axis: point **Q** and direction **K**
  - Angle **b**
  - Distance **d**

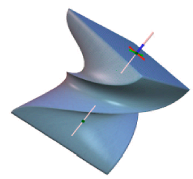




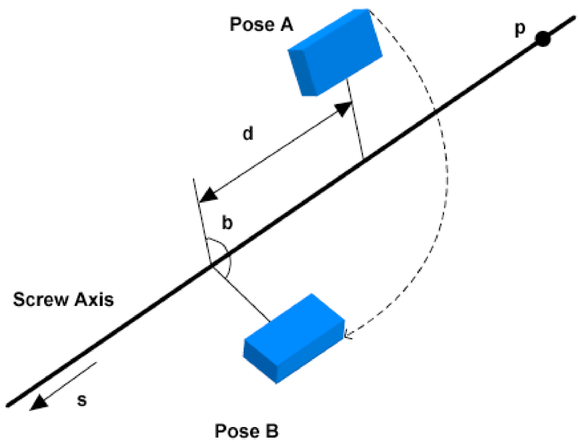
Screw history

(Ceccarelli [2000] Detailed study of screw motion history)

- Archimede (287–212 BC) designed helicoidal screw for water pumps
- Leonardo da Vinci (1452–1519) description of helicoidal motion
- Dal Monte (1545–1607) and Galileo (1564–1642) mechanical studies on helicoidal geometry
- Giulio Mozzi (1763) screw axis as the "spontaneous axis of rotation"
- L.B. Francoeur (1807) theorem of helicoidal motion
- Gaetano Giorgini (1830) analytical demonstration of the existence of the "axis of motion" (locus of minimum displacement points)
- Ball (1900) "*Theory of screws*"
- Rodrigues (1940) helicoidal motion as general motion
- ….
- Zefrant and Kumar (CAD 1998) Interpolating motions



Screw Parameters



Computing the screw parameters

From initial and final poses:

M(0) and M(1)

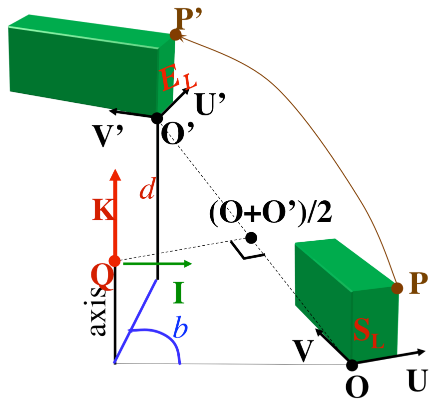
K:=(U'–U)×(V'–V);

K:=K / ||K||;

$b := 2 \sin^{-1}(||U'–U|| / (2\ ||K\times U||))$;

d:=K•OO';

Q:=(O+O')/2 + (K×OO') / (2tan(b/2));

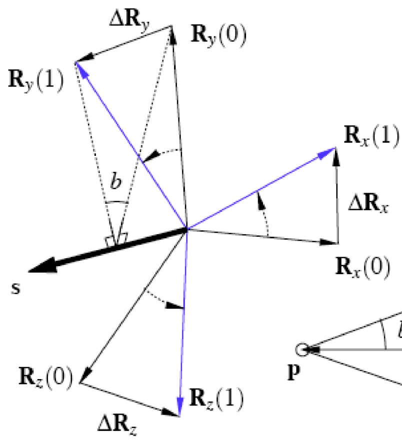


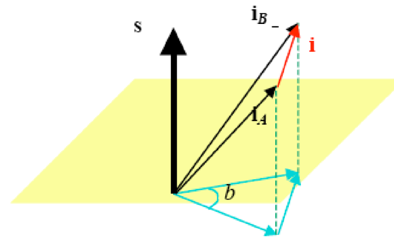
To apply a t-fraction of the screw:

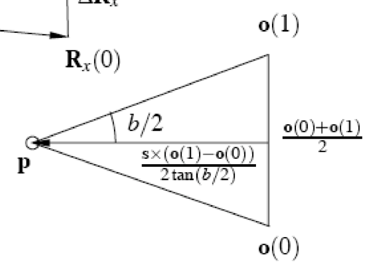
- Translate by –Q;
- Rotate around K by tb;
- Translate by Q;
- Translate by (0,0,td);

Notation on Previous slide : s=K and p=Q

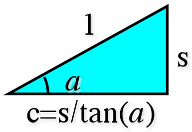
Construction details

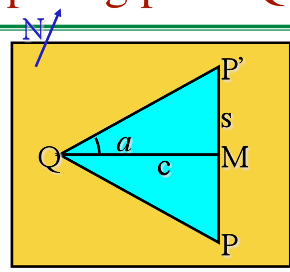


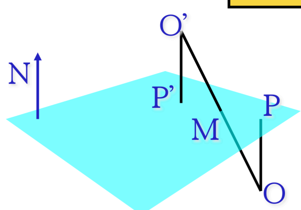




Computing point Q on screw axis







Given P, P', and a, compute Q the center of rotation by angle $b=2a$ that brings P to P'

$M=(P+P')/2$

$T=N\times PP'/||PP'||$

$c=||PP'||/(2\tan(a))$

$A=M+cT$

$\quad =(P+P')/2+N\times PP'/(2\tan(a))$

$P=O+hN$, $P'=O'-hN$ (projections)

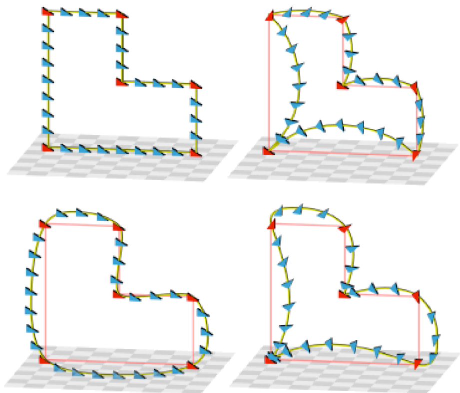
$P+P'=O+O'$, $PP'=OO'-2hN$

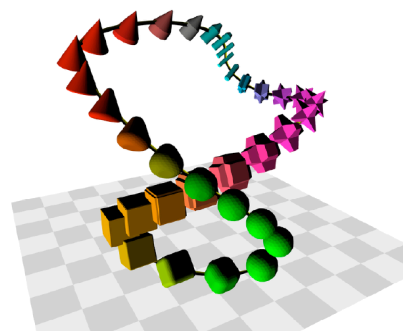
$N\times PP'= N\times OO'-2hN\times N = N\times OO'$

$A = (O+O')/2 + N\times OO'/(2\tan(b/2))$

Smoothing piecewise-screw motions

"ScrewBender: Polyscrew Subdivision for Smoothing Interpolating Motions" A. Powell and J. Rossignac

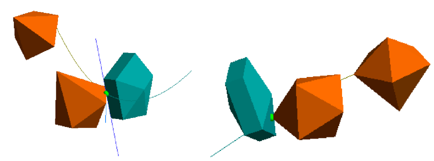


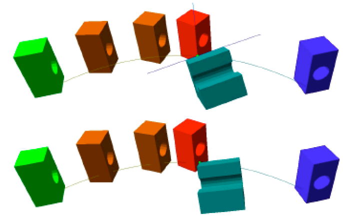


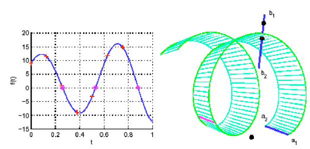
Collision during screw motions

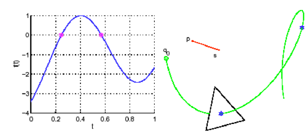
"Collision Prediction for Polyhedra under Screw Motions", BM Kim and J. Rossignac, ACM Symposium on Solid Modeling and Applications**, 2003.**

- **Helix is** $V(t) = r\cos(tb)i + r\sin(tb)j + tdk$ **in screw coordinates**
- **The screw intersects plane** $d + V(t) \cdot n = 0$ **for values of** t **satisfying**





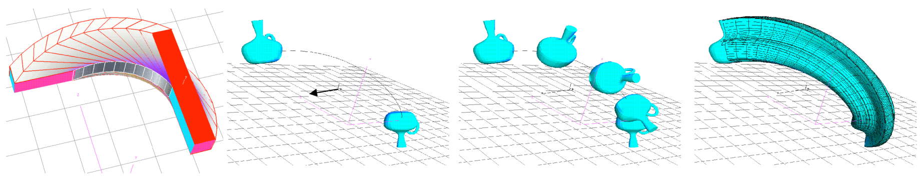




Volume swept during screw motion

Computing and visualizing pose-interpolating 3D motions

Jarek Rossignac and Jay J. Kim, CAD, 33(4)279:291, April 2001.



SweepTrimmer: Boundaries of regions swept by sculptured solids during a pose-interpolating screw motion

Jarek Rossignac and Jay J. Kim