Computer Graphics

Transformations

Fernando Geraci

Subtopics

Basics
- Primitives
- Homogeneous Coordinates
- Transformations

Linear Transformations
- Properties
- Linear Combination
- Scaling
- Rotation

Affine Transformations
- Definition
- Translation Matrix
- Rotation Matrices
- Combining Transformations

Relative Frames
- Points
- Vectors
- Frame Change Matrix

General Form

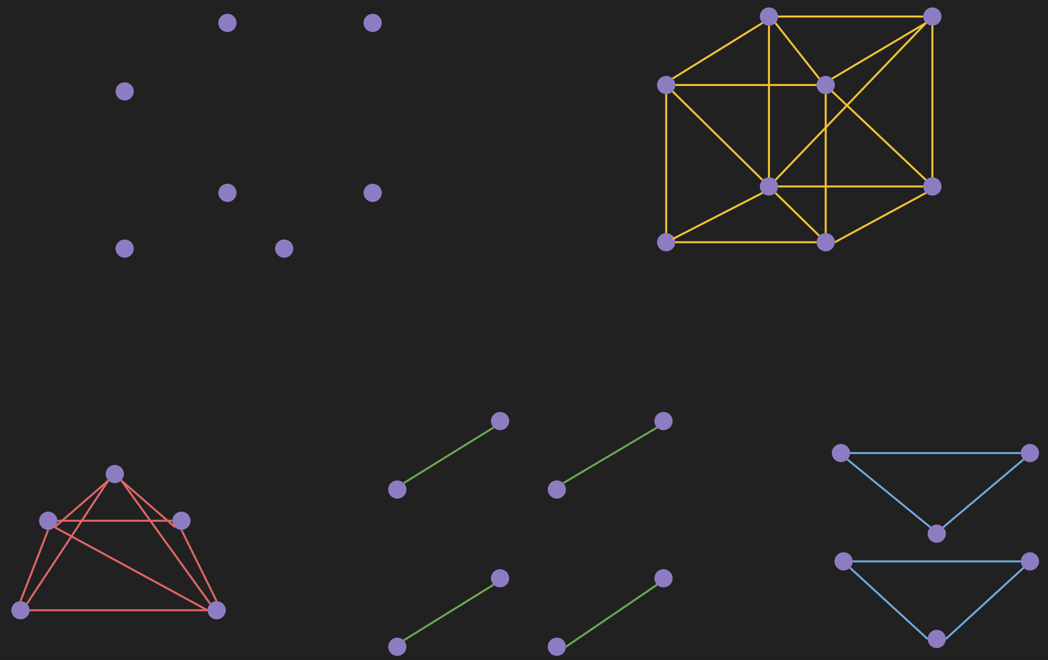
API

Basics - Primitives

We use POINTS - as vectors (x,y,z,w) - to represent geometries.

These can represent:

INPUT:	$< p_1, p_2, \dots, p_k >$
Points List	N points - no lines
Line List	(N / 2) lines
Line Strip	(N - 1) lines
Triangle List	(N / 3) triangles
Triangle Strip	(N - 2) triangles



Basics - Homogeneous Coordinates

Note that we are representing these points as a vector of 4 components $v = (x,y,z,w)$

The w component, denotes whether we are transforming a:

- POINT $(x, y, z, 1)$ - actual location in the coordinates plane
- VECTOR $(x, y, z, 0)$ - only represent magnitude and direction

The transformations can be homogeneously applied to 4x4 matrices just as we do it for 3x3

Note that given two points, p_1 and p_2, if we subtract them, we will obtain a vector v.
Conversely, if we subtract a vector v from p_1 we will obtain a point.

This indicates that properties hold.

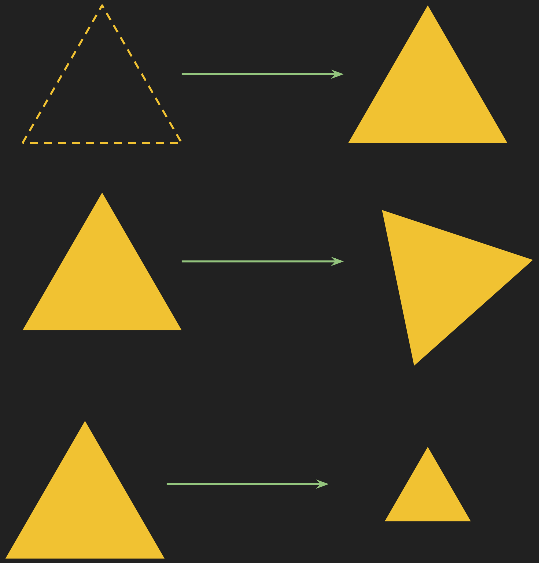
Fernando Geraci

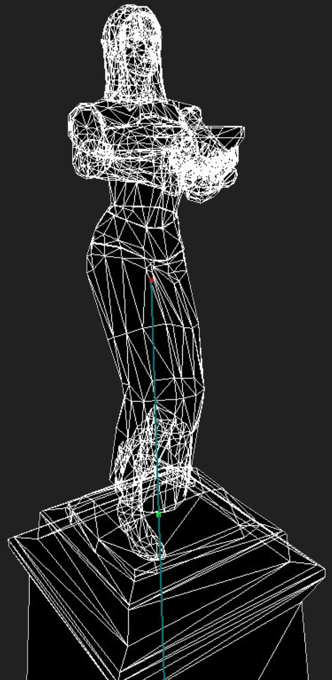
Basics - Transformations

Set of triangles which when combined, form the exterior shell of objects.

Geometric Transformations

- Translation
- Rotation
- Scaling





Linear Transformation - Properties

Given a function $t(u) = u'$, a linear transformation corresponds to such functions on which the following properties hold:

$$t(u + v) = t(u) + t(v)$$
$$t(ku) = kt(u)$$

With some manipulation:

$$\begin{aligned} t(au + bv + cw) &= t(au + (bv + cw)) \\ &= at(u) + t(bv + cw) \\ &= at(u) + bt(v) + ct(w) \end{aligned}$$

Linear Transformation - Linear Combination

Now, let $u = (u_1,u_2,u_3)$, $u' = (u_1',u_2',u_3')$ … what will be get if we do:

$$t(u) = x*t(u) + y*t(u') + z*t(u'')$$

$$= \quad (x \;\; y \;\; z)\begin{pmatrix} u_1 & u_2 & u_3 \\ u'_1 & u'_2 & u'_3 \\ u''_1 & u''_2 & u''_3 \end{pmatrix}$$

We obtain the matrix representation of a linear combination

$$= \quad (\, x\,(u_1 + u_2 + u_3)\,,\, y\,(u_1' + u_2' + u_3')\,,\, z\,(u_1'' + u_2'' + u_3'')\,)$$

Fernando Geraci

Linear Transformation - Scaling

Changing the size of an object.

Given a set of vectors and a scalar value k, we simply scale its components:

$k(v) = (k*v_x, k*v_y, k*v_z)$

Following the linear transformation properties:

$$\left.\begin{aligned} k(u + v) \quad &= (\, k(u_x + v_x)\,,\, k(u_y + v_y),\, k(u_z + v_z)\,) \\ &= (\, ku_x + kv_x\,,\; ku_y + kv_y\,,\, ku_z + kv_z\,) \\ &= (\, ku_x\,,\, ku_y\,,\, ku_z\,) + (\, kv_x\,,\, kv_y\,,\, kv_z\,) \\ &= k(u) + k(v) \end{aligned}\right\}$$

We can derive a matrix representation.

$S * E$, where $E = (\, e_1\,,\, e_2\,,\, e_3\,)$

R^3 Standard Basis

Linear Transformation - Scaling

Scaling Matrices: with inverse:

$$S * (e_1 , e_2 , e_3 ,1) = S * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1/s & 0 & 0 & 0 \\ 0 & 1/s & 0 & 0 \\ 0 & 0 & 1/s & 0 \\ 0 & 0 & 0 & 1/s \end{pmatrix}$$

For example: given two points $p_1 = (1 , 1 , 2)$ and $p_2 = (5, 3, 1)$. We can scale them by a factor of (2 , 2 , 1) the following way:

$$(-1 , 1 , 2, 1) * \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = (-2 , 2 , 4, 1) \quad (5 , 3 , 1 , 1) * \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = (10 , 6 , 2 , 1)$$

Fernando Geraci

Linear Transformation - Rotation

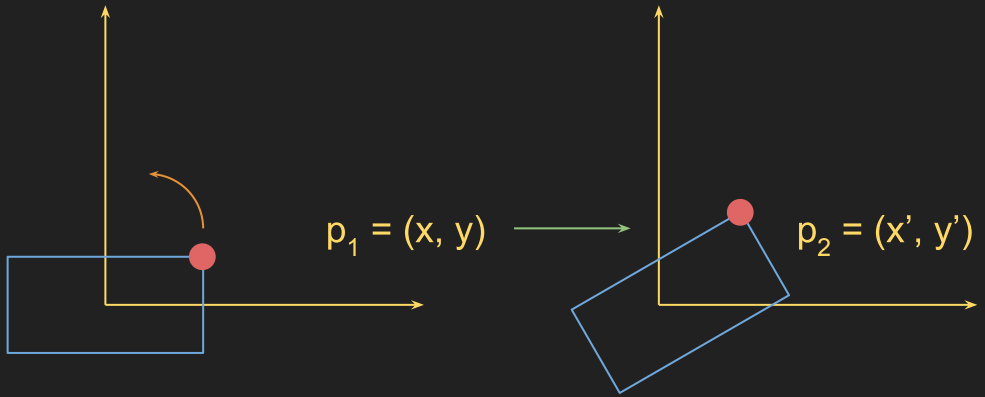
Before jumping into 3D - every rotation happens in 2-axes, while 1 always remains still. Let’s take a look at 2D rotations first:

If

$$x = r\cos(t)$$
$$y = r\sin(t)$$

then

$$x' = r\cos(t + a) = r\cos(t)\cos(a) - r\sin(t)\sin(a) = x\cos(a) - y\sin(a)$$
$$y' = r\sin(t + a) = r\sin(t)\cos(a) + r\cos(t)\sin(a) = y\cos(a) + x\sin(a)$$



With matrices:

counter-clockwise → $(x,y)\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$

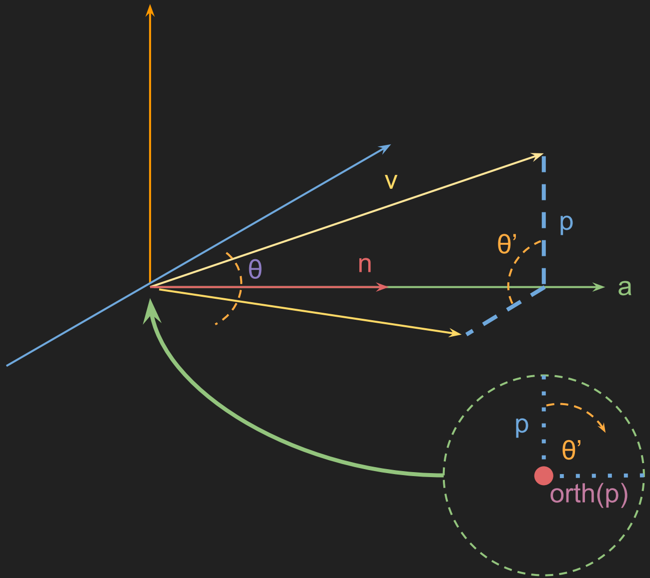
clockwise → $(x,y)^T\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

Fernando Geraci

Linear Transformation - Rotation

We will move a point or vector around an axis (fixed) clockwise.

First of, let $r_a(v,\theta)$ be the rotation of a vector v, around an axis a, θ degrees.



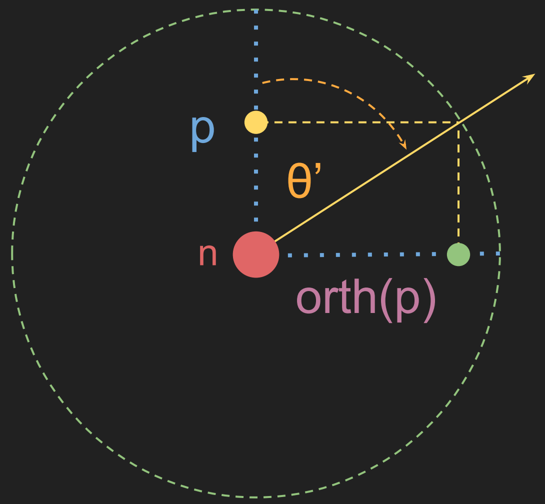
Observe from the picture on the left:

v is the vector we want to rotate around a
But in reality, we will end rotating a vector perpendicular to a in 2D !

1. norm(a) = n such that ||n|| == 1
2. $p = v - dot(v,n)\ n = v - proj_n(v)$
3. orth(p) = (p x n)
4. $r_n(p,\theta)$

Finally: $r_a(v,\theta) = v + r_n(p,\theta)$

Linear Transformation - Rotation



$$r_n(p,\theta) = \cos(\theta)*p + \sin(\theta)*orth(p)$$

Notice that the $||orth(p)|| == ||n \times p|| == ||p||$

Finally if

$$\begin{aligned} R_a(v,\theta) &= proj_n(v) + r_n(p,\theta) \\ &= dot(n,v)\ n + \cos(\theta)\ p + \sin(\theta)\ (n x p) \\ &= dot(n,v)\ n + (1-\cos(\theta))\ dot(n,v)\ n + \sin(\theta)\ (n x p) \end{aligned}$$

Linear Transformation - Rotation

Finally, if we apply the equation to the standard basis, we obtain the general rotation matrix on an arbitrary axis:

c = cos(q)
s = sin(q)
t = (1-c)

$$\begin{pmatrix} c+tx^2 & txy+sz & yxz-sy \\ txy+sz & c+ty^2 & tyz+sx \\ txz+sy & tyz-sx & c+tz^2 \end{pmatrix}$$

We can compute the rotation matrix on an arbitrary axis by replacing x y z and t

Example, let's rotate u, q degrees around the x axis such that $x = e_1$ (1,0,0)

We obtain the rotation matrix of $R_{e1}(q)$ $\longrightarrow$ $R_x(q) = u * \begin{pmatrix} 0 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{pmatrix}$

Fernando Geraci

Linear Transformation - Rotation

Furthermore, for origin rotations, we let n, of R_n ,be (1,0,0) , (0,1,0) and (0,0,1) we obtain:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \qquad R_y = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \qquad R_z = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Note how the row corresponding to the axis we are rotating around, doesn’t change!

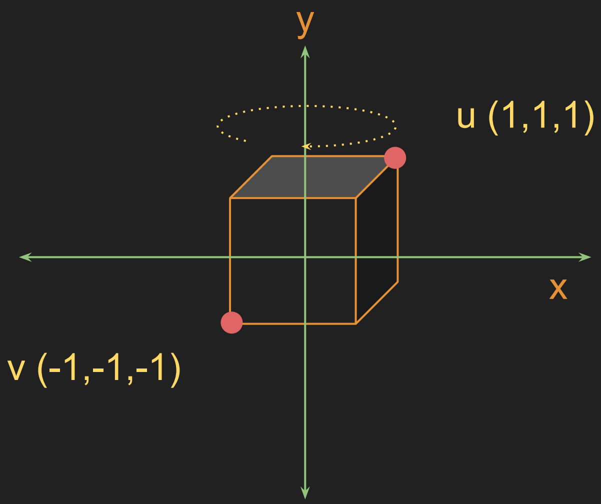
The following is an example of $R_y(n)$ by 45 degrees:

$$R_y\,(n,45) = \begin{pmatrix} \cos45 & 0 & -\sin45 \\ 0 & 1 & 0 \\ \sin45 & 0 & \cos45 \end{pmatrix} \dashrightarrow \begin{pmatrix} \sqrt{2}/2 & 0 & -\sqrt{2}/2 \\ 0 & 1 & 0 \\ \sqrt{2}/2 & 0 & \sqrt{2}/2 \end{pmatrix}$$

Fernando Geraci

Linear Transformation - Rotation

Example:



$$R_y(u,45) = u * \begin{pmatrix} \sqrt{2}/2 & 0 & -\sqrt{2}/2 \\ 0 & 1 & 0 \\ \sqrt{2}/2 & 0 & \sqrt{2}/2 \end{pmatrix} = (2\sqrt{2}/2 , 1 , 0)$$

$$R_y(v,45) = v * \begin{pmatrix} \sqrt{2}/2 & 0 & -\sqrt{2}/2 \\ 0 & 1 & 0 \\ \sqrt{2}/2 & 0 & \sqrt{2}/2 \end{pmatrix} = (-2\sqrt{2}/2 , -1 , 0)$$

Affine Transformations - Definition

Simply a linear transformation AND a translation.

Let t(v) be a linear transformation. Then $t_{affine}(v) = t(v) + b$, hence:

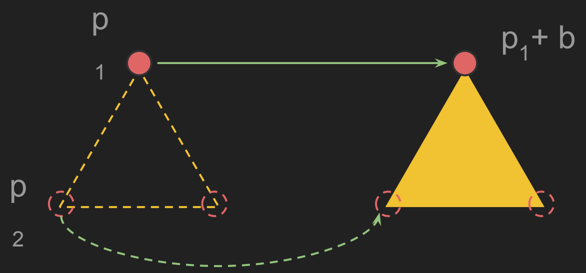
$$t(v) + b = (x,y,z) \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} + (b_x,b_y,b_z) = \underbrace{t(v) + b = (x,y,z,w) \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ b_x & b_y & b_z & 1 \end{pmatrix}}_{\text{Homogeneous Coordinates}}$$

Note that if w = 0, then the translation would not affect a vector. Otherwise, we translate the point

Fernando Geraci

Affine Transformations - Translation

Given a set of point $<p_1,p_2,...,p_k>$ and a vector b, we can displace p_i in direction b:



Any affine translation can be achieved with the following matrix:

With inverse:

$$p_i * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ b_x & b_y & b_z & 1 \end{pmatrix} = ((p_x + b) , (p_y + b) , (p_z + b) , (p_w * 1))$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -b_x & -b_y & -b_z & 1 \end{pmatrix}$$

Fernando Geraci

Affine Transformations - Rotation

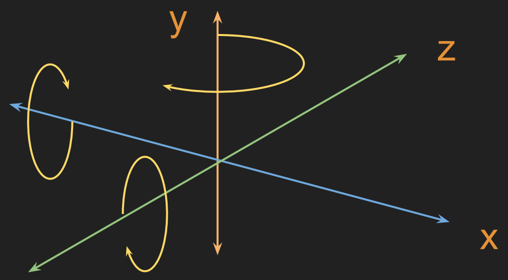
Rotation Matrices using homogeneous coordinates:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_z = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Because R_n is orthonormal, for any rotation matrix R_n, its inverse is $R^{-1} = R^T_n$

For an arbitrary axis, is the same criteria:

$$\begin{pmatrix} c+tx^2 & txy+sz & yxz-sy & 0 \\ txy+sz & c+ty^2 & tyz+sx & 0 \\ txz+sy & tyz-sx & c+tz^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

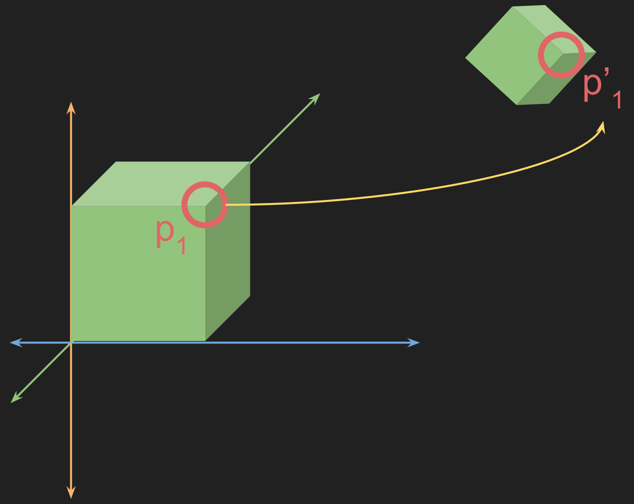


Convention - rotate on:

y	yaw
x	pitch
z	roll

Fernando Geraci

Affine Transformations - Compound Transformations



Here we applied the following transformations:

1) Scale p_i by a ½ factor. S
2) Rotate p_i by 45 degrees R
3) Translate p_i (x,y,z,w) T

These can be expressed:

$$\underbrace{(R_p(S\,(x,y,z,w)))T}_{\text{step-by-step}} = \underbrace{(x,y,z,w)(SRT)}_{\text{compound}}$$

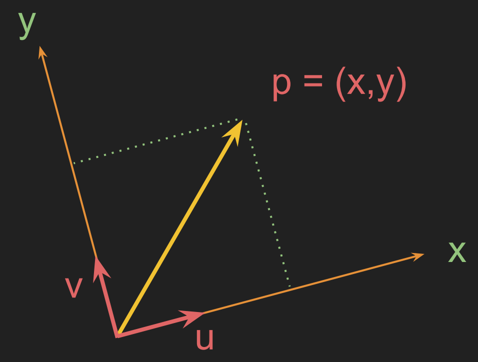
Recall that each transformation takes the form: p_iT , where T is the transformation matrix. Let SRT = C → 2 multiplications

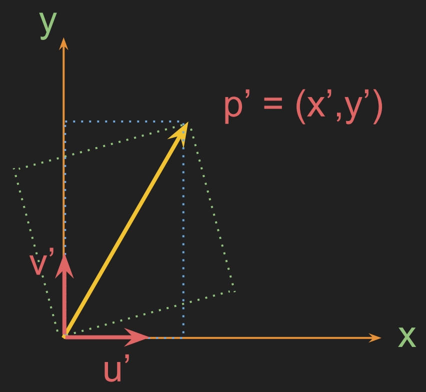
Clearly both methods are mathematically equivalent, due to the associative (not commutative) nature of matrices.
In practice assume there is an object with k points:

1. step-by-step: $(S*k_i)*R_p*T$ → k * 3
2. compound: $k_i*(SRT)$ → k * (C)

Fernando Geraci

Relative Frames - Vectors





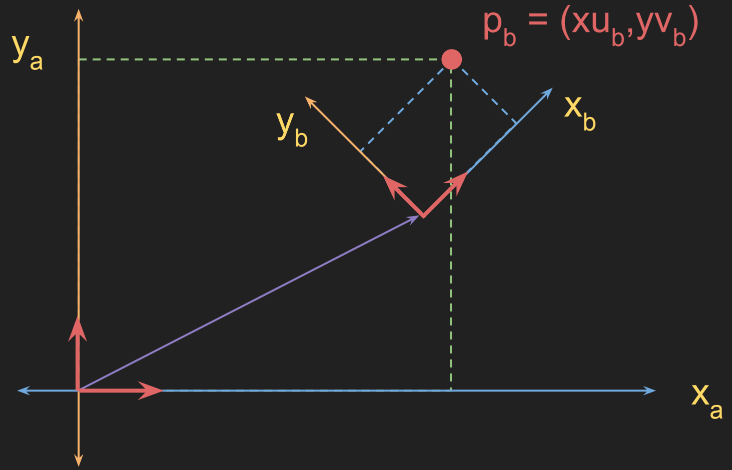
Recall that vector do not have a location in the frame, but rather, they have length and magnitude, which need to translate accordingly to the other frames.

If u and v are unit vectors aiming along each respective axis, then $p_a = (u_a x + v_a y)$ and $p_b = (u_b x + v_b y)$

If $p_a = (ux + vy + qz)$ then $p_b = (u'x + v'y + q'z)$

Relative Frames - Points

Given a point's coordinates, relative to a frame, how can be identify the coordinates of the same element (unchanged) relatives to a different frame?



Intuitively, given a point relative to a frame A, we can calculate its coordinates in another frame B by adding the difference of origins in between frames.

$$p_a(u_a x + v_a x + q_a z) = p_b(u_b x + v_b x + q_b z) + Q_b$$

q (x , y , z) ⟶ origin of the relative frame b

Fernando Geraci

Relative Frames - Frame Change Matrix

Another advantage of using homogeneous coordinates, is that we could handle transformations of points and vectors the same way the following way:

$$p' = xu' + yv' + zs' + wQ$$

If w is 0, then the product is just as if we are transforming the vector, otherwise, it moves the point.

Note how that is the following linear combination:

$$(x , y , z , w) \ * \begin{pmatrix} \longleftarrow & u & \longrightarrow & \\ \longleftarrow & v & \longrightarrow & \\ \longleftarrow & s & \longrightarrow & \\ Q_x & Q_y & Q_z & 1 \end{pmatrix} = (x' , y' , z' , w')$$

Note that for u , v and z, the fourth component is always 0.

Relative Frames - Frame Change Matrix

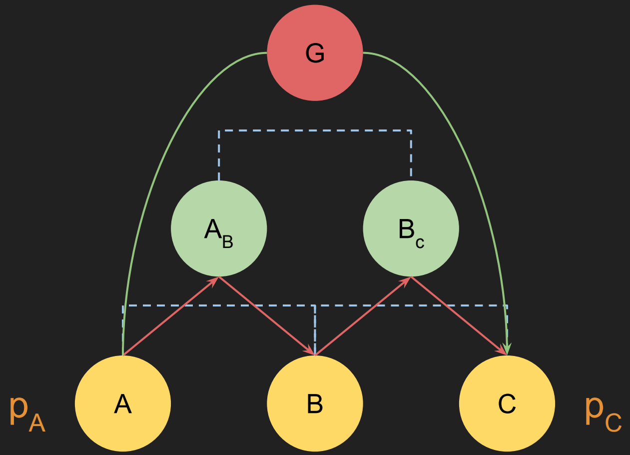
As an example, let us compute the matrix which will express $p_a(1,-2,0)$ and $q_a(1,2,0)$ in p_b terms.
We know that $Q_b(-6,2,0)$, $u_b(1/\sqrt{2},1/\sqrt{2},0)$, $v_b(-1/\sqrt{2},1/\sqrt{2},0)$ and $w_b(0,0,1)$.

$$p_a\,(1,\ -2,\ 0,\ 1)\ *\ \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -6 & 2 & 0 & 1 \end{pmatrix} = p_b\,(-3.8,\ 1.2,\ 0,\ 1)$$

$$q_a\,(1,\ 2,\ 0,\ 0)\ *\ \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -6 & 2 & 0 & 1 \end{pmatrix} = p_b\,(-0.7,\ 2.1,\ 0,\ 0)$$

Fernando Geraci

Relative Frames - Computation



Given a point p_A , assume we want to transform it to a relative point p_C

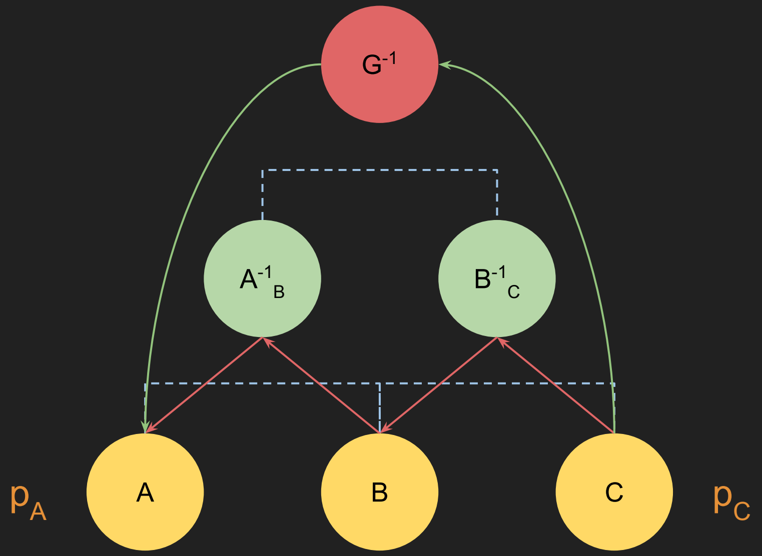
Let A , B , and C be the frames, with A_B and A_C being their transformation matrix, and G being a transformation matrix from A_B to B_C

Step-by-step:	$(p_A * A_B) * A_C \rightarrow p_C$	i * 2	multiplications
Combined:	$(p_A * G) \rightarrow p_C$	i	multiplications

Relative Frames - Two-ways conversions

Recall from the matrices properties, that:

$$p_b = p_a A$$
$$p_b A^{-1} = p_a A A^{-1}$$
$$p_b A^{-1} = p_a I$$
$$p_b A^{-1} = p_a$$



Although the same principle is applied to other operations, we can see how nicely it works for frame transformations.
Do not forget that $G = B^{-1}A^{-1}$

Summary

The following is a summary of the transformations presented thus far:

$$at(x,y,z,w) = xt(i) + yt(j) + zt(k) + wb \longrightarrow (x, y, z, w) * \begin{pmatrix} \longleftarrow & t(i) & \longrightarrow & \\ \longleftarrow & t(j) & \longrightarrow & \\ \longleftarrow & t(k) & \longrightarrow & \\ b_x & b_y & b_z & 1 \end{pmatrix}$$

$$p_b(x',y',z',w') = xu' + yv' + zs' + wQ_b \longrightarrow (x, y, z, w) * \begin{pmatrix} \longleftarrow & u & \longrightarrow & \\ \longleftarrow & v & \longrightarrow & \\ \longleftarrow & s & \longrightarrow & \\ Q_x & Q_y & Q_z & 1 \end{pmatrix}$$

API

DirectXMath provides:

XMMatrixScaling	-	Creates a scaling matrix
XMMatrixScalingFromVector	-	‘’ but using the components of a vector
XMMatrixRotation[X,Y,Z]	-	Clockwise $R_{[x,y,z]}$
XMMatrixRotationAxis	-	Rotates an angle around an axis n
XMMatrixTranslation	-	Creates just a translation matrix
XMMatrixTranslationFromVector	-	“ but using the components of a vector
XMVector3TransformCoord	-	Transforms points w=1
XMVector3TransformNormal	-	Transforms vectors w=0

All these transformations can be done with XMVector4Transform