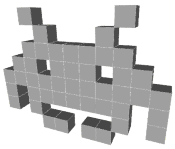
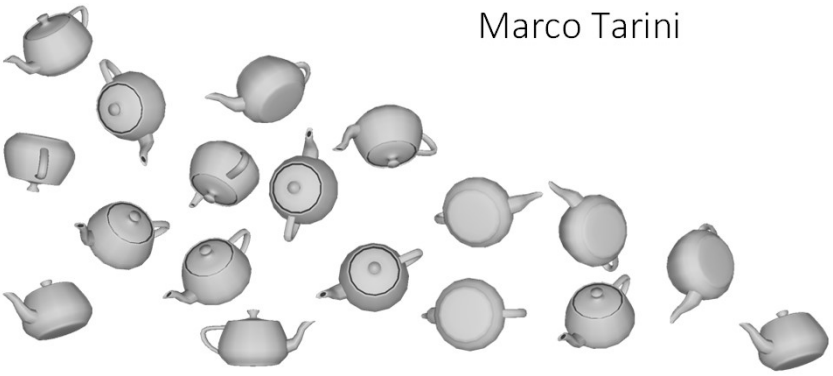
3D VideoGames

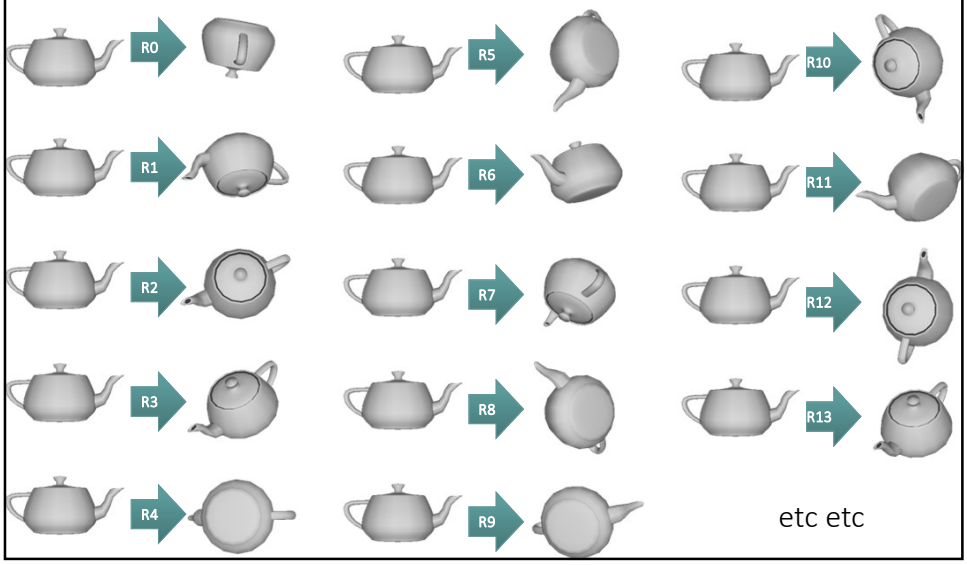
Rotations Representations



Marco Tarini



3D rotations: how many? (...dimension?)



etc etc

3D rotations: how many? (…dimension?)

(clearly they include the identity)

Answer: 3 DOF (degrees of freedom).
i.e., rotations in 3D are a 3 dimensional space.

How to (internally) represent a 3D rotation ?

that is, also, the orientation of an object

- Many possible ways, matching the criteria to different degrees
- Each way is used in some context (realted to gaming)
- Therefore, today we will see them all (including how to switch from one to another)

Compare with: representing *translations* in 3D

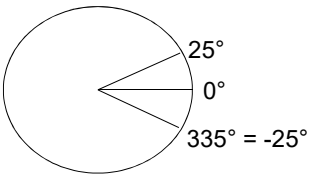
- Trivial:
  displacement vector (3 floats)!
  - perfect under all criteria
    (exercise: verify)

Compare with: representing rotations *in 2D*

- Trivial: one angle (a dimensionless scalar)
  - perfect under all criteria
    (exercise: verify)
  - (only choice: degrees or radiants?)

[0,360°) [0,2·Pi)

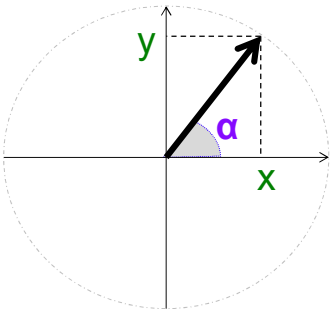
caveat: interpolation!
«pick the shortest route»
mix(25°, 335° , 0.5) = 0
(but, still easy)



Compare with: representing rotations *in 2D*

(a digression)

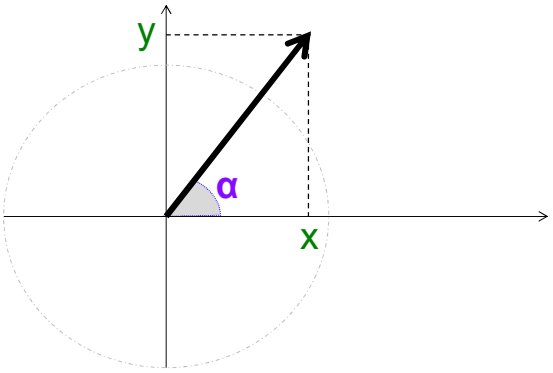
- Compute *angle* → *vector*



Compare with: representing rotations *in 2D*

(a digression)

- Compute *angle* ← *vector*



pro tip: use *atan2* in any language

Way 1: 3x3 matrix (9 floats)

- after all, each rotation is a linear matrix
- (3x3 submatrix of the 4x4 rotation matrix)

$$\begin{matrix} R & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{matrix}$$

- as we know, R is orthonormal, with det = +1

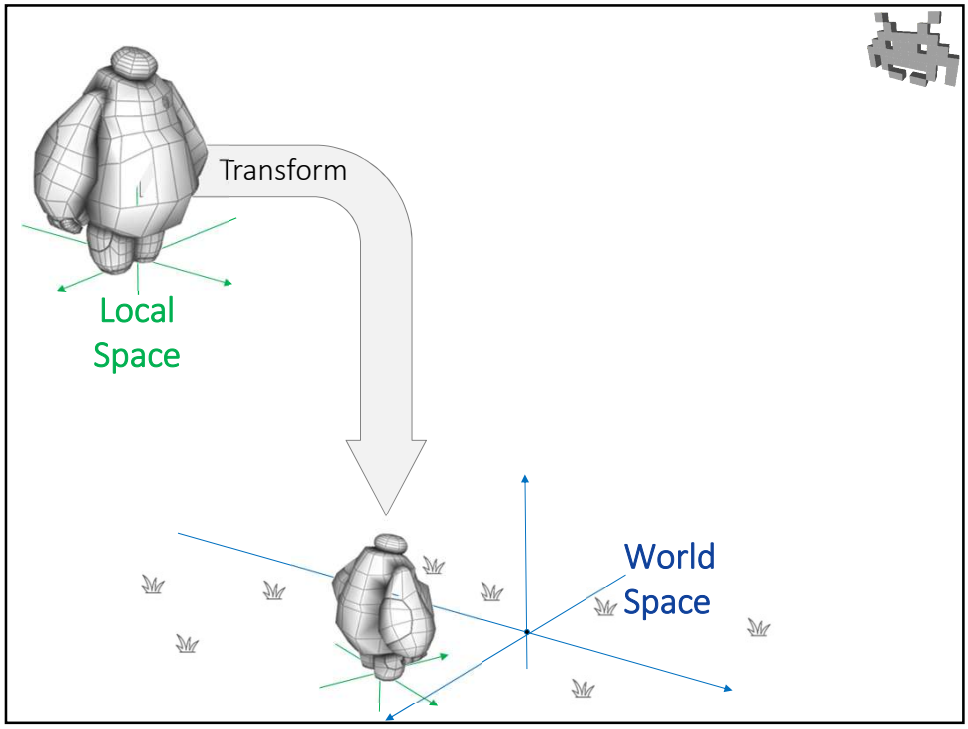
Way 1: 3x3 matrix (9 floats)

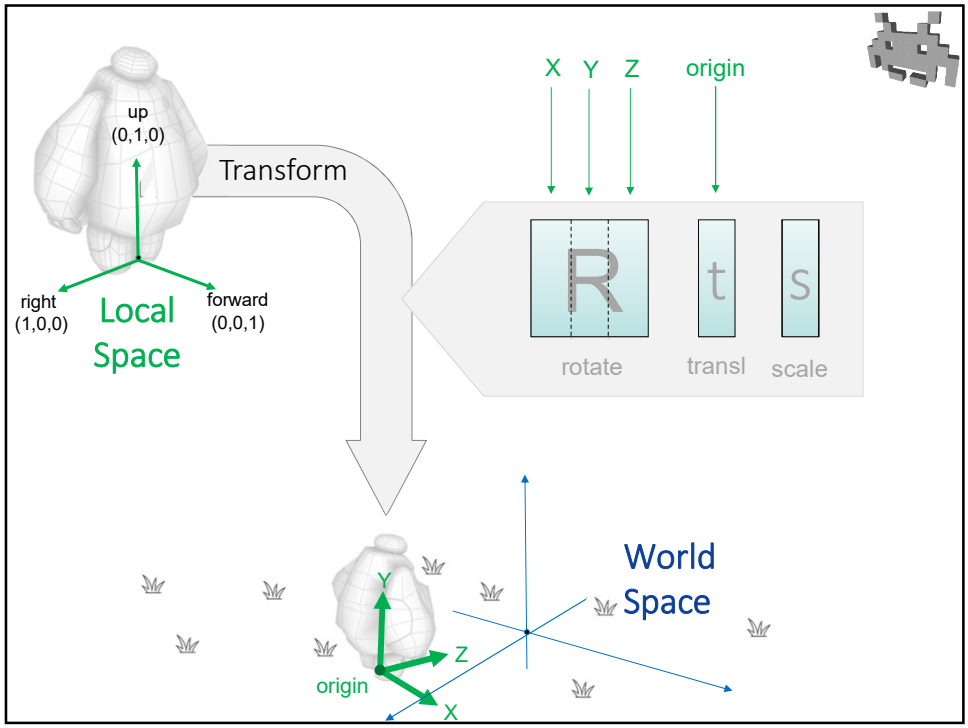
- Wasteful in RAM (9 scalars, versus a minimum of 3)
- Easy to apply (matrix-vector prod: 9 mults)
- Rel. easy to cumulate (matrix-matrix prod: 27 x mult)
- Immediate to invert (just transpose)
- Interpolate: troubles

why?

$$k \; R_0 + (1-k) \; R_1 = M$$

NOT a rotation


Transform
Local
Space
World
Space


up
(0,1,0)
right
(1,0,0)
forward
(0,0,1)
Local
Space
Transform
X Y Z
origin
R
t
S
rotate
transl
scale
World
Space
Y
Z
X
origin

Way 1: 3x3 matrix (9 floats)

- Nice plus:
  its three columns are
  the three versors representing
  the X,Y,Z axis of the locals space
  in global space
  - i.e. the world-space versor
    representing right, upward, forward (in Unity)
    or forward, right, upward (in Unreal engin)
    of the local frame

Recap: representing rotations 1/4

		3x3 Matrix
Space efficient? (in RAM, GPU, storage…)		9 scalars
Efficient / easy to	Apply (to points/vectors)	9 products (3 dot products)
	Invert (produce inverse)	just transpose
	Cumulate (with another rotation)	matrix multiplication (9 dots)
	Interpolate (with another rotation)	
Intuitive? (e.g. to manually set)		?!?
Notes…		Free skew + scale! Free X,Y,Z axis