

transformations

- we will often want to perform rigid transformations to these objects to position them
 - translate
 - rotate
- or move them in an animation
 - time varying tform
- position or move virtual camera
- we also may use non-rigid tforms to specify shape
 - squash a sphere into an ellipsoid.

SO....

- so we must understand how to manipulate 3d coordinates
- we must pay attention to order of tforms
- we must pay attention to the role of the coordinate system w.r.t. which we perform a tform
- we will look at linear and affine transformations
- at end of the day, our code will have vertices with 3d coords and we will use 4 by 4 matrices to describe properly manipulate them
- but to figure out what to code, we need to first do some thinking/paper-pencil work.

Geometric data types

- we describe a point using a coordinate vector

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

- specifies position wrt an agreed upon coordinate system
 - agreed origin
 - three agreed directions
 - if we change agreed upon c.s., we must change the coordinate vector
- so a point is specified with a coordinate system and a coordinate vector

4 geometric data types

- point: \tilde{p}
 - represents place
- vector: \vec{v}
 - represents motion between points
- coordinate vector: \mathbf{c}
- coordinate system $\vec{\mathbf{s}}^t$
 - “basis” for vectors
 - “frame” is for points

vectors vs coordinate vectors

- a vector is a geometric entity (motion between points)
- a coordinate vector is a set of numbers used to specify a vector given an agreed coordinate system

vector space

- a vector space V : some set of elements \vec{v}
- needs an addition operation
- needs scalar multiplication
- some other rules
 - addition is associative and commutative
 - scalar mul must distribute across vector add

$$\alpha(\vec{v} + \vec{w}) = \alpha\vec{v} + \alpha\vec{w}$$

examples of vector spaces

- the set V may be lots of different things
 - motion between points !!!!
 - polynomial expressions
 - farm animals
 - triplets of numbers

coordinate system: basis

- a basis is a (minimal) set of vectors that we can use to get to all of the vectors using our ops.
 - linearly independent
- dimension is number of basis elements needed
- for us it will be 3
- basis can be used to address all of the vectors uniquely
 - using coordinates

$$\vec{v} = \sum_i c_i \vec{b}_i$$

shorthand

- write this as

$$\vec{v} = \sum_i c_i \vec{b}_i = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- even shorter

$$\vec{v} = \vec{\mathbf{b}}^t \mathbf{c}$$

linear transformation

- a linear tfrom \mathcal{L} maps from V to V
- satisfies 2 rules

$$\begin{aligned} \mathcal{L}(\vec{v} + \vec{u}) &= \mathcal{L}(\vec{v}) + \mathcal{L}(\vec{u}) \\ \mathcal{L}(\alpha\vec{v}) &= \alpha\mathcal{L}(\vec{v}) \end{aligned}$$

linear tforms and matrices

- linear transformation can be exactly specified by telling us its effect on the basis vectors.
- linear transforms can be expressed with matrix multiplication
- Linearity implies

$$\vec{v} \Rightarrow \mathcal{L}(\vec{v}) = \mathcal{L}\left(\sum_i c_i \vec{b}_i\right) = \sum_i c_i \mathcal{L}(\vec{b}_i)$$

- in our shorthand this is

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \mathcal{L}(\vec{b}_1) & \mathcal{L}(\vec{b}_2) & \mathcal{L}(\vec{b}_3) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- each $\mathcal{L}(\vec{b}_i)$ can ultimately be written as some linear combination of the original basis vectors using numbers $M_{i,j}$

$$\begin{bmatrix} \mathcal{L}(\vec{b}_1) & \mathcal{L}(\vec{b}_2) & \mathcal{L}(\vec{b}_3) \end{bmatrix} = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$

so

- a linear mapping operating on a vector can be expressed as

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

well defined ops

- vector to vector

$$\vec{b}^t \mathbf{c} \Rightarrow \vec{b}^t M \mathbf{c}$$

- basis to basis

$$\vec{b}^t \Rightarrow \vec{b}^t M$$

- coordinate vector to coordinate vector (this is the one we will see in code, but not until then).

$$\mathbf{c} \Rightarrow M \mathbf{c}$$

matrices for change of basis

- we just saw as an intermediate result an expression of the form

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \vec{a}_3 \end{bmatrix} = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$

- or in shorthand

$$\vec{\mathbf{a}}^t = \vec{\mathbf{b}}^t M$$

- this is not a transformation.
- we have used a matrix to express one named basis with respect to another.
- this will be useful too.
- we can also use this to have different expressions for the same vector

$$\vec{v} = \vec{\mathbf{a}}^t \mathbf{c} = \vec{\mathbf{b}}^t M \mathbf{c}$$

2D rotations

- rotate by θ degrees counter clockwise about the origin

$$\begin{aligned} & \begin{bmatrix} \vec{b}_1 & \vec{b}_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ \Rightarrow & \begin{bmatrix} \vec{b}_1 & \vec{b}_2 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \end{aligned}$$

3d handedness

- in 2d forward rotation is counterclockwise
- right handed coordinate system for 3D
- in 3d direction rotation is right handed

3d rotations

- rotate a point by θ degrees around the z axis of the basis

$$\begin{aligned} & \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ \Rightarrow & \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \end{aligned}$$

- where $c \equiv \cos \theta$, and $s \equiv \sin \theta$.
- fixes points on z axis
- for points in $z = k$ plane, it is like a 2D rotation
- basis is important (z direction)

more 3d rotations

- around x axis

$$\begin{aligned} & \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\ \Rightarrow & \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \end{aligned}$$

- forward rotation around the y axis

$$\begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix}$$

arbitrary rotation

- can get any rotation by applying one x,y,z
- can get any rotation by applying one x,y,x
 - called eulter angles
 - visualize with set of gimballs
- one can specify rotation with unit vector axis $[k_x, k_y, k_z]^t$ and θ using matrix

$$\begin{bmatrix} k_x^2 v + c & k_x k_y v - k_z s & k_x k_z v + k_y s \\ k_y k_x v + k_z s & k_y^2 v + c & k_y k_z v - k_x s \\ k_z k_x v - k_y s & k_z k_y v + k_x s & k_z^2 v + c \end{bmatrix}$$

- where $v \equiv 1 - c$

other linear transforms

- uniform scales

$$\begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}$$

- non-uniform scales

$$\begin{bmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{bmatrix}$$

- shears

$$\begin{bmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- not so common