points vs vectors

- Euclidean vector $\vec{v}$ := motion between points in Euclidean space
  - has the structure of a 3 dimensional linear/vector space.
  - addition and scalar multiplication have meaning
  - zero vector is no motion
  - cannot really translate motion
- Euclidean point $\tilde{p}$ := a position in a Euclidean space
  - addition and scalar mul don’t make sense
  - zero doesn’t make sense

interaction between points and vectors

subtraction of any point pair does make sense, gives us a unique vector

$$\tilde{p} - \tilde{q} = \vec{v}$$

Adding a point and a vector to get a unique point makes sense

$$\tilde{q} + \vec{v} = \tilde{p}$$

if $\vec{v} \neq \vec{0}$ then $\tilde{q} + \vec{v} \neq \tilde{q}$.

new ideas points, frames, Cvecs4, affine transforms, affine matrices

frames

- recall: basis is three vectors

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

- for affine space we will use a frame
  - start with a chosen origin point $\tilde{o}$,
  - add to it a linear combination of vectors using 3 coordinates $c_i$ to get to any desired point $\tilde{p}$.

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b}_i = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} = \vec{f}^t \mathbf{c}$$

4-coordinate vectors

- point is specified with a 4-coordinate vector, we call a Cvec4.
  - four numbers
  - last one is always 1 (for a point).
  - .... or 0. (and we get a vector)

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 0 \end{bmatrix} = \vec{v}$$

- the use of Cvec4s will also come in super handy with camera projections.

affine matrices
• a we will call a matrix an “affine matrix” if it is of the form

\[
\begin{bmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

• we perform an “affine transformation” on a point by placing an affine matrix between a frame and a 4-coordinate vector, just like with linear transforms

\[
\begin{bmatrix}
\vec{b}_1 \\
\vec{b}_2 \\
\vec{b}_3 \\
\vec{o}
\end{bmatrix}
\begin{bmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
c'_1 \\
c'_2 \\
c'_3 \\
1
\end{bmatrix}
= \mathcal{L}(\vec{v})
\]

• for short, \( \vec{f}^T c \rightarrow \vec{f}^T A c \)

• the data types work iff the 4th row is \([0, 0, 0, 1]\)

  – two ways to show.

• similarly, we can apply an affine transform to a frame as \( \vec{f}^T \rightarrow \vec{f}'A \)

inverse

• inverse matrix \( A^{-1} A = I \)

• undoes the affine transform

• it too is an affine transform

building an affine transform from a Linear transform

• suppose i have a linear transform represented by a 3-by-3 matrix

\[
\vec{v} = \begin{bmatrix}
\vec{b}_1 \\
\vec{b}_2 \\
\vec{b}_3
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\vec{b}'_1 \\
\vec{b}'_2 \\
\vec{b}'_3
\end{bmatrix}
\begin{bmatrix}
a & b & c \\
e & f & g \\
i & j & k
\end{bmatrix}
\begin{bmatrix}
c'_1 \\
c'_2 \\
c'_3
\end{bmatrix}
= \mathcal{L}(\vec{v})
\]

• let me put it in the upper left of an affine matrix and apply it to a point

\[
\begin{bmatrix}
\vec{b}_1 \\
\vec{b}_2 \\
\vec{b}_3
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\vec{b}'_1 \\
\vec{b}'_2 \\
\vec{b}'_3 \\
\vec{o}
\end{bmatrix}
\begin{bmatrix}
a & b & 0 & 0 \\
e & f & 0 & 0 \\
i & j & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
c'_1 \\
c'_2 \\
c'_3 \\
1
\end{bmatrix}
= \mathcal{L}(\vec{v}) + \vec{o}
\]
the effect is as if we applied a linear transform on the vectors connecting the origin to the point
so we can use this to, say rotate a point about the origin
see fig
matrix shorthand for an upgraded linear transform

\[ L = \begin{bmatrix} l & 0 \\ 0 & 1 \end{bmatrix} \]

translations

• suppose i transform using a matrix of the form:

\[
\begin{bmatrix}
\vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \vec{d} \\
\end{bmatrix}
\begin{bmatrix}
c_1 \\ c_2 \\ c_3 \\ 1
\end{bmatrix} \rightarrow
\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
c_1 \\ c_2 \\ c_3 \\ 1
\end{bmatrix}
\]

• we see that its effect on the coordinates is

\[ c_1 \rightarrow c_1 + t_x \]
\[ c_2 \rightarrow c_2 + t_y \]
\[ c_3 \rightarrow c_3 + t_z \]

• this is a translation of the point
• For a translation we use the shorthand

\[ T = \begin{bmatrix} i & t \\ 0 & 1 \end{bmatrix} \]

together

• any affine matrix can be factored into lin-trans form

\[
\begin{bmatrix}
a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1
\end{bmatrix}
= \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
a & b & c \\ 0 & f & g \\ 0 & j & k \\ 0 & 0 & 0
\end{bmatrix}
\]

• in shorthand

\[
\begin{bmatrix} l & t \\ 0 & 1 \end{bmatrix}
= \begin{bmatrix} i & t \\ 0 & 1 \end{bmatrix}
\begin{bmatrix} l & 0 \\ 0 & 1 \end{bmatrix}
\]

(1)

\[ A = TL \]  

(2)

• if the linear part is a rotation, then we call this a rigid body matrix which implements a rigid body transform, (RBT).

\[ A = TR \]

• preserves vector dot products, handedness of triples, and distances between points.

affine transform acting on vector

• if fourth coord of \( c \) is zero, this just transforms a vector to a vector.
  – notice that the fourth column is irrelevant for such input
  – translation has no effect on a vector