Geometric Objects 
and Transformations

How to represent basic geometric types, such as points and vectors?

How to convert between various representations? 
(through a linear transformation!)

How to establish a method for dealing with geometric problems independent of coordinate systems?

Homogeneous coordinates

Affine transformations

OpenGL transformation matrices 
(in discussion session)
Mathematical Spaces

Linear Vector Spaces

– Scalars and vectors

– Examples of vector spaces:
  1. geometric vectors
  2. n–tuples of real numbers
  3. scalar–vector multiplication and vector–vector addition.
– The greatest number of linearly independent vectors that we can find in a space gives the dimension of the space
– Bases

Affine Spaces

– Add points → location
– point–to–point subtraction yields a vector
– vector–point multiplication yields a point
– Frames

Euclidean Spaces

– Add measure of size or distance
– Inner product
Fundamental geometric objects:

point – a location in space

vector – a directed line segment

Coordinate systems (frames):

a point can be represented unambiguously with a fixed reference point (the origin)

a vector can be uniquely defined as:

\[ w = \alpha_1 \, v_1 + \alpha_2 \, v_2 + \alpha_3 \, v_3 \]

\[ a^T = [ \alpha_1 \, \alpha_2 \, \alpha_3 ] \]

\[ w = a^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \]

then a point in a coordinate system defined by \( P_0 \) (the origin) and a basis vector (\( v_1, v_2, v_3 \)) can be written as:

\[ P = P_0 + u \, v_1 + v \, v_2 + w \, v_3 \]
Changes of Coordinate Systems

*Object* —> *World* —> *View* —> *Screen*

How the representation of a vector changes when we change the basis vectors?

\[ v = (v_1, v_2, v_3) \quad <----> \quad u = (u_1, u_2, u_3) \]

There exists a relationship between the two bases:

\[ u_1 = \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3 \]
\[ u_2 = \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3 \]
\[ u_3 = \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3 \]

Which is captured by a 3x3 matrix:

\[
M = \begin{bmatrix}
\gamma_{11} & \gamma_{12} & \gamma_{13} \\
\gamma_{21} & \gamma_{22} & \gamma_{23} \\
\gamma_{31} & \gamma_{32} & \gamma_{33}
\end{bmatrix}
\]

transforms a vector in one basis to its representation in the second basis:

\[ u = M \cdot v \quad \text{and} \quad v = M^{-1} u \]
Moving from abstract vectors to working with column matrices of scalars

Consider a vector

\[ w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \]

\[ = a^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad \text{where} \quad a = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \]

while with respect to a different basis

\[ u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \]

\[ w = \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 = b^T u \]

the 2nd basis in terms of the 1st:

\[ w = b^T u \]

\[ = b^T M v = a^T v \]

Hence

\[ a = M^T b \quad \text{and} \quad b = (M^T)^{-1} a \]

Remember the underlying basis!!
Rotation and scaling of a basis:

Translation of a basis cannot be represented with the 3x3 matrix:

Can we "expand" the representation such that we can change frames yet still to use matrices to represent the change?
Homogeneous Coordinates

Usually we represent a point located at \((x, y, z)\) using a frame determined by \(P_0, v_1, v_2, v_3\) as

\[
P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}
\]

for a vector \(w = \delta_1 v_1 + \delta_2 v_2 + \delta_3 v_3\)

\[
w = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix}
\]

How to distinguish between points and vectors?

How can we continue using matrix multiplication in 3 dimensions to represent a change in frames?

Homogeneous Coordinates!!

Use 4–d column matrices to represent both points and vectors in 3 dimensions.
Homogeneous Coordinate Representation

\[ P = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + P_0 \]

\[ w = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \delta_1 v_1 + \delta_2 v_2 + \delta_3 v_3 \]

1. Operating using ordinary matrix algebra
2. All affine transformation can be represented as matrix multiplication
3. Concatenation of successive transformations resulting in more efficient calculations
4. Hardware implemented
So for any two frames defined by 

\((v_1, v_2, v_3, P_0)\) and \((u_1, u_2, u_3, Q_0)\)

A matrix representation of the change of frames:

\[
\begin{bmatrix}
  u_1 \\
  u_2 \\
  u_3 \\
  Q_0
\end{bmatrix} = M \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  P_0
\end{bmatrix}
\]

where \(M = \begin{bmatrix}
  \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\
  \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\
  \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\
  \gamma_{41} & \gamma_{42} & \gamma_{43} & 1
\end{bmatrix}\)

for any two points or two vectors \(a\) and \(b\) in the two frames in homogeneous-coordinate representation:

\[
b^T \begin{bmatrix}
  u_1 \\
  u_2 \\
  u_3 \\
  Q_0
\end{bmatrix} = b^T M \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  P_0
\end{bmatrix} = a^T \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  P_0
\end{bmatrix}
\]

Hence \(a = M^T b\)
**Affine Transformation**

A transformation is a function:

\[ Q = T(P) \]

\[ v = R(u) \]

using homogeneous coordinates, we can define transformation with a single function:

\[ q = f(p) \]

\[ v = f(u) \]

Make \( f() \) a linear function:

such that \( f(\alpha p + \beta q) = \alpha f(p) + \beta f(q) \)

\( f \) is an affine transformation:

1. a combination of linear transformations
2. transform only end points of a line to determine completely a transformed line
3. parallel lines are transformed into parallel lines
A line in point–vector form:
\[ p(\alpha) = p_0 + \alpha v \]

For any affine transformation matrix \( A \):
\[ Ap(\alpha) = Ap_0 + \alpha Ad \]

A line in two–point form:
\[ P(\alpha) = \alpha p_0 + (1-\alpha)P_1 \]

For any affine transformation matrix \( A \):
\[ AP(\alpha) = \alpha Ap_0 + (1-\alpha)AP_1 \]

Affine Transformations preserve parallelism of lines
Rotation, translation, reflection
Scaling
Shearing
Translation

Displace points by a fixed distance in a given direction

\[ P' = P + d \]

Rotation:

\[
\begin{align*}
    x &= \rho \cos\phi \\
    y &= \rho \sin\phi \\

    x' &= \rho \cos(\theta + \phi) \\
    y' &= \rho \sin(\theta + \phi)
\end{align*}
\]

\[
\begin{align*}
    x' &= \rho \cos\phi \cos\phi - \rho \sin\phi \sin\theta \\
    &= x \cos\theta - y \sin\theta
\end{align*}
\]

\[
\begin{align*}
    y' &= \rho \cos\phi \sin\phi + \rho \sin\phi \cos\theta \\
    &= x \sin\theta + y \cos\theta
\end{align*}
\]

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} =
\begin{bmatrix}
    \cos\theta & -\sin\theta \\
    \sin\theta & \cos\theta
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]
Rotation (cont’d)

A fixed point, an angle, and a line or a vector about which to rotate

Scaling

With a fixed point, a vector, and a scaling factor

\[ p' = S \cdot p \]
Transformations in Homogeneous Coordinates

Each affine transformation is represented as a 4x4 matrix

\[
\mathbf{M} = \begin{bmatrix}
\alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\
\alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\
\alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

A point is a column matrix

\[
\mathbf{P} = \begin{bmatrix}
x \\
y \\
z \\
0
\end{bmatrix}
\]
Translation

\[ T = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ p' = T \, p \]

\[ x' = x + \alpha_x \]
\[ y' = y + \alpha_y \]
\[ z' = z + \alpha_z \]

Inverse?

\[ T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -\alpha_x \\ 0 & 1 & 0 & -\alpha_y \\ 0 & 0 & 1 & -\alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Scaling

\[
S = \begin{bmatrix}
\beta_x & 0 & 0 & 0 \\
0 & \beta_y & 0 & 0 \\
0 & 0 & \beta_z & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[p' = S \ p\]

\[x' = \beta_x \ x\]
\[y' = \beta_y \ y\]
\[z' = \beta_z \ z\]

Inverse?

\[
S^{-1} = \begin{bmatrix}
1/\beta_x & 0 & 0 & 0 \\
0 & 1/\beta_y & 0 & 0 \\
0 & 0 & 1/\beta_z & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

What happens if \(\beta_x=\beta_y=\beta_z = -1\) ?
Rotation about a main coordinate axis:

\[
R_x(\theta) = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & -\sin \theta & 0 \\
0 & \sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
R_y(\theta) = \begin{bmatrix}
\cos \theta & 0 & \sin \theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta & 0 & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
R_z(\theta) = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & 0 \\
\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
p' = R_z \, p
\]

\[
x' = x \cos \theta - y \sin \theta \\
y' = x \sin \theta + y \cos \theta \\
z' = z;
\]

Inverse?

\[
R^{-1}(\theta) = R(-\theta) = R^T(\theta)
\]
Concatenation of Transformations

Concatenate sequences of basic transformations to define an arbitrary transformation directly

Matrix product is associative

\[ q = CBAp \]
\[ q = C(B(Ap)) \]

For transforming many points

\[ M = CBA \]
\[ q = M \ p \]
Rotation About a Fixed Point

\[ M = T(p_f) R_z(\theta) T(-p_f) \]

In general

\[ M = T(p_f) R(\theta) T(-p_f) \]
General rotation is easy if $\theta_x, \theta_y, \theta_z$ are known!

**Rotation About an Arbitrary Axis**

To specify a rotation, we need:

1. a fixed point $P_0$, center of rotation
2. a vector about which we rotate
3. an angle of rotation

$$M = T(p_0) \ R(\theta) \ T(-p_0)$$

What is $R(\theta)$?

**Align the axis of rotation with one of the main coordinate axis!!**
\[ R(\theta) = R_x(-\theta_x)R_y(-\theta_y)R_z(\theta)R_y(\theta_y)R_x(\theta_x) \]

\[ M = T(p_0) \ R_x(-\theta_x)R_y(-\theta_y)R_z(\theta)R_y(\theta_y)R_x(\theta_x)T(-p_0) \]
\[ R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 & 0 \\ 0 & \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \]

\[ R_y(\theta_y) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \]

d = \sqrt{\alpha_y^2 + \alpha_z^2}

\[ R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \alpha_z/d & -\alpha_y/d & 0 & 0 \\ 0 & \alpha_y/d & \alpha_z/d & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \]

\[ R_y(\theta_y) = \begin{bmatrix} d & 0 & -\alpha_x & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \alpha_x & 0 & d & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \]