Objectives

- Introduce the elements of geometry
  - Scalars
  - Vectors
  - Points
- Develop mathematical operations among them in a coordinate-free manner
- Define basic primitives
  - Line segments
  - Polygons

Basic Elements

- Geometry is the study of the relationships among objects in an n-dimensional space
  - In computer graphics, we are interested in objects that exist in three dimensions
- Want a minimum set of primitives from which we can build more sophisticated objects
- We will need three basic elements
  - Scalars
  - Vectors
  - Points

Coordinate-Free Geometry

- When we learned simple geometry, most of us started with a Cartesian approach
  - Points were at locations in space \( p=(x,y,z) \)
  - We derived results by algebraic manipulations involving these coordinates
- This approach was nonphysical
  - Physically, points exist regardless of the location of an arbitrary coordinate system
  - Most geometric results are independent of the coordinate system
- Example Euclidean geometry: two triangles are identical if two corresponding sides and the angle between them are identical

Transformations to Change Coordinate Systems

- 4 coordinate systems
  - 1 point \( P \)
  - \( M_{1\to 2} = T(4,2) \)
  - \( M_{2\to 1} = T(2,3) \cdot S(0.5,0.5) \)
  - \( M_{3\to 4} = T(6.7,1.8) \cdot R(45^\circ) \)

Scalars

- Need three basic elements in geometry
  - Scalars, Vectors, Points
- Scalars can be defined as members of sets which can be combined by two operations (addition and multiplication) obeying some fundamental axioms (associativity, commutativity, inverses)
- Examples include the real and complex number systems under the ordinary rules with which we are familiar
- Scalars alone have no geometric properties
Vectors

• Physical definition: a vector is a quantity with two attributes
  - Direction
  - Magnitude
• Examples include
  - Force
  - Velocity
  - Directed line segments
    - Most important example for graphics
    - Can map to other types

Vector Operations

• Every vector has an inverse
  - Same magnitude but points in opposite direction
• Every vector can be multiplied by a scalar
• There is a zero vector
  - Zero magnitude, undefined orientation
• The sum of any two vectors is a vector
  - Use head-to-tail axiom

Linear Vector Spaces

• Mathematical system for manipulating vectors
• Operations
  - Scalar-vector multiplication: \( \mathbf{u} = \alpha \mathbf{v} \)
  - Vector-vector addition: \( \mathbf{w} = \mathbf{u} + \mathbf{v} \)
• Expressions such as
  \[ \mathbf{v} = \mathbf{u} + 2 \mathbf{w} - 3 \mathbf{r} \]
  Make sense in a vector space

Vectors Lack Position

• These vectors are identical
  - Same length and magnitude
• Vectors spaces insufficient for geometry
  - Need points

Points

• Location in space
• Operations allowed between points and vectors
  - Point-point subtraction yields a vector
  - Equivalent to point-vector addition

Affine Spaces

• Point + a vector space
• Operations
  - Vector-vector addition
  - Scalar-vector multiplication
  - Point-vector addition
  - Scalar-scalar operations
• For any point define
  - \( -1 \cdot \mathbf{P} = \mathbf{P} \)
  - \( 0 \cdot \mathbf{P} = \mathbf{0} \) (zero vector)
Lines

- Consider all points of the form \( P(\alpha) = P_0 + \alpha \cdot d \)
- Set of all points that pass through \( P_0 \) in the direction of the vector \( d \)

Parametric Form

- This form is known as the parametric form of the line
  - More robust and general than other forms
  - Extends to curves and surfaces
- Two-dimensional forms
  - Explicit: \( y = mx + h \)
  - Implicit: \( ax + by + c = 0 \)
  - Parametric:
    \[
    x(\alpha) = \alpha x_0 + (1 - \alpha)x_1 \\
    y(\alpha) = \alpha y_0 + (1 - \alpha)y_1
    \]

Rays and Line Segments

- If \( \alpha \geq 0 \), then \( P(\alpha) \) is the ray leaving \( P_0 \) in the direction \( d \)
- If we use two points to define \( v \), then
  \[
  P(\alpha) = Q + \alpha (R - Q) = Q + \alpha v
  \]
  \[
  = aR + (1 - a)Q
  \]
- For \( 0 \leq \alpha \leq 1 \) we get all the points on the line segment joining \( R \) and \( Q \)

Convexity

- An object is convex iff for any two points in the object all points on the line segment between these points are also in the object

Affine Sums

- Consider the “sum” \( P = \alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_n P_n \)
  - If \( \alpha_1 + \alpha_2 + \ldots + \alpha_n = 1 \)
  - in which case we have the affine sum of the points \( P_1, P_2, \ldots, P_n \)
- If, in addition, \( \alpha \geq 0 \), we have the convex hull of \( P_1, P_2, \ldots, P_n \)

Convex Hull

- Smallest convex object containing \( P_1, P_2, \ldots, P_n \)
- Formed by “shrink wrapping” points
Curves and Surfaces

- Curves are one parameter entities of the form \( P(\alpha) \) where the function is nonlinear.
- Surfaces are formed from two-parameter functions \( P(\alpha, \beta) \).
  - Linear functions give planes and polygons.

Planes

- A plane can be defined by a point and two vectors or by three points.

\[ P(\alpha, \beta) = R + \alpha u + \beta v \]

\[ P(\alpha, \beta) = R + \alpha (Q - R) + \beta (P - R) \]

Triangles

- A triangle can be defined by three points.

\[ P(\alpha_1, \alpha_2, \alpha_3) = \alpha_1 P + \alpha_2 Q + \alpha_3 R \]

where

\[ \alpha_1 + \alpha_2 + \alpha_3 = 1 \]

\[ \alpha_i \geq 0 \]

The representation is called the barycentric coordinate representation of \( P \).

Barycentric Coordinates

- Triangle is convex so any point inside can be represented as an affine sum.

Normals

- Every plane has a vector \( n \) normal (perpendicular, orthogonal) to it.
- From point-two vector form \( P(\alpha, \beta) = R + \alpha u + \beta v \), we know we can use the cross product to find \( n = u \times v \) and the equivalent form.

\[ (P(\alpha) - P) \cdot n = 0 \]
Objectives

- Introduce concepts such as dimension and basis
- Introduce coordinate systems for representing vector spaces and frames for representing affine spaces
- Discuss change of frames and bases
- Introduce homogeneous coordinates

Linear Independence

- A set of vectors $v_1, v_2, \ldots, v_n$ is linearly independent if $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = 0$ iff $\alpha_1 = \alpha_2 = \ldots = 0$
- If a set of vectors is linearly independent, we cannot represent one in terms of the others
- If a set of vectors is linearly dependent, at least one can be written in terms of the others

Dimension

- In a vector space, the maximum number of linearly independent vectors is fixed and is called the dimension of the space
- In an $n$-dimensional space, any set of $n$ linearly independent vectors form a basis for the space
- Given a basis $v_1, v_2, \ldots, v_n$, any vector $v$ can be written as $v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
  where the $\{\alpha_i\}$ are unique

Representation

- Until now we have been able to work with geometric entities without using any frame of reference, such as a coordinate system
- Need a frame of reference to relate points and objects to our physical world.
  - For example, where is a point? Can’t answer without a reference system
  - World coordinates
  - Camera coordinates

Coordinate Systems

- Consider a basis $v_1, v_2, \ldots, v_n$
- A vector is written $v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
- The list of scalars $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is the representation of $v$ with respect to the given basis
- We can write the representation as a row or column array of scalars
  $$a = \begin{bmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_n \end{bmatrix}^T$$

Example

- $v = 2v_1 + 3v_2 - 4v_3$
- $a = \begin{bmatrix} 2 & 3 & -4 \end{bmatrix}^T$
- Note that this representation is with respect to a particular basis
- For example, in OpenGL we start by representing vectors using the object basis but later the system needs a representation in terms of the camera or eye basis
Coordinate Systems

• Which is correct?
  \[ \begin{align*}
  \vec{v} & \quad \text{or} \quad \vec{v} \\
  \end{align*} \]

• Both are because vectors have no fixed location

Frames

• A coordinate system is insufficient to represent points
• If we work in an affine space we can add a single point, the origin, to the basis vectors to form a frame

Representation in a Frame

• Frame determined by \((P_0, v_1, v_2, v_3)\)
• Within this frame, every vector can be written as
  \[ \vec{v} = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n \]
• Every point can be written as
  \[ P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n \]

A Single Representation

If we define 0\(\cdot P = 0\) and 1\(\cdot P = P\) then we can write
\[ \begin{align*}
  \vec{v} & = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n \\
  P & = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n \\
\end{align*} \]
Thus we obtain the four-dimensional homogeneous coordinate representation
\[ \begin{align*}
  \vec{v} & = [\alpha_1 \alpha_2 \alpha_3 0]^T \\
  \vec{p} & = [\beta_1 \beta_2 \beta_3 1]^T \\
\end{align*} \]

Confusing Points and Vectors

Consider the point and the vector
\[ \begin{align*}
  P & = P_0 + \beta_1 v_1 + \beta_2 v_2 + \ldots + \beta_n v_n \\
  \vec{v} & = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n \\
\end{align*} \]
They appear to have the similar representations
\[ \begin{align*}
  \vec{p} & = [\beta_1 \beta_2 \beta_3]^T \\
  \vec{v} & = [\alpha_1 \alpha_2 \alpha_3]^T \\
\end{align*} \]
which confuses the point with the vector
A vector has no position

Vector can be placed anywhere
point fixed

Homogeneous Coordinates

The homogeneous coordinates form for a three dimensional point \([x, y, z]^{T}\) is given as
\[ \vec{p} = [x', y', z', w']^{T} = [x x y z w]^{T} \]
We return to a three dimensional point (for \(w=0\)) by
\[ \begin{align*}
  x & = x' / w \\
  y & = y' / w \\
  z & = z' / w \\
\end{align*} \]
If \(w=0\), the representation is that of a vector
Note that homogeneous coordinates replaces points in three dimensions by lines through the origin in four dimensions
For \(w=1\), the representation of a point is \([x, y, z, 1]^{T}\)
Homogeneous Coordinates and Computer Graphics

- Homogeneous coordinates are key to all computer graphics systems
  - All standard transformations (rotation, translation, scaling) can be implemented with matrix multiplications using 4x4 matrices
  - Hardware pipeline works with 4 dimensional representations
  - For orthographic viewing, we can maintain $w=0$ for vectors and $w=1$ for points
  - For perspective we need a perspective division


Change of Coordinate Systems

- Consider two representations of the same vector with respect to two different bases. The representations are

  \[ \mathbf{a} = [\alpha_1, \alpha_2, \alpha_3] \]
  \[ \mathbf{b} = [\beta_1, \beta_2, \beta_3] \]

  where

  \[ \mathbf{v} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 = [\alpha_1, \alpha_2, \alpha_3] \mathbf{v}^T \]
  \[ = \beta_1 \mathbf{u}_1 + \beta_2 \mathbf{u}_2 + \beta_3 \mathbf{u}_3 = [\beta_1, \beta_2, \beta_3] \mathbf{u}^T \]


Representing second basis in terms of first

Each of the basis vectors, $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$, are vectors that can be represented in terms of the first basis

\[ \mathbf{u}_1 = g_{11} \mathbf{v}_1 + g_{12} \mathbf{v}_2 + g_{13} \mathbf{v}_3 \]
\[ \mathbf{u}_2 = g_{21} \mathbf{v}_1 + g_{22} \mathbf{v}_2 + g_{23} \mathbf{v}_3 \]
\[ \mathbf{u}_3 = g_{31} \mathbf{v}_1 + g_{32} \mathbf{v}_2 + g_{33} \mathbf{v}_3 \]

The coefficients define a 3x3 matrix and the bases can be related by

\[ \mathbf{a} = \mathbf{M} \mathbf{b} \]

see text for numerical examples


Change of Frames

- We can apply a similar process in homogeneous coordinates to the representations of both points and vectors

Consider two frames:

\[ (\mathbf{P}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \]  \[ (\mathbf{Q}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3) \]

- Any point or vector can be represented in either frame
- We can represent $\mathbf{Q}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ in terms of $\mathbf{P}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$


Representing One Frame in Terms of the Other

Extending what we did with change of bases

\[ \mathbf{u}_1 = g_{11} \mathbf{v}_1 + g_{12} \mathbf{v}_2 + g_{13} \mathbf{v}_3 \]
\[ \mathbf{u}_2 = g_{21} \mathbf{v}_1 + g_{22} \mathbf{v}_2 + g_{23} \mathbf{v}_3 \]
\[ \mathbf{u}_3 = g_{31} \mathbf{v}_1 + g_{32} \mathbf{v}_2 + g_{33} \mathbf{v}_3 \]
\[ \mathbf{Q}_0 = g_{41} \mathbf{v}_1 + g_{42} \mathbf{v}_2 + g_{43} \mathbf{v}_3 + g_{44} \mathbf{P}_0 \]

defining a 4x4 matrix

\[ \mathbf{M} = \begin{bmatrix}
1 & 12 & 13 & 0 \\
21 & 22 & 23 & 0 \\
31 & 32 & 33 & 0 \\
41 & 42 & 43 & 1
\end{bmatrix} \]

Working with Representations

Within the two frames any point or vector has a representation of the same form

\[ a = [a_1 \ a_2 \ a_3 \ a_4] \] in the first frame
\[ b = [b_1 \ b_2 \ b_3 \ b_4] \] in the second frame

where \( a_4 = b_4 = 1 \) for points and \( a_4 = b_4 = 0 \) for vectors

\[ a = M b \]

The matrix \( M \) is 4 x 4 and specifies an affine transformation in homogeneous coordinates.

The World and Camera Frames

- When we work with representations, we work with n-tuples or arrays of scalars
- Changes in frame are then defined by 4 x 4 matrices
- In OpenGL, the base frame that we start with is the world frame
- Eventually we represent entities in the camera frame by changing the world representation using the model-view matrix
- Initially these frames are the same (\( M = I \))

Moving the Camera

If objects are on both sides of \( z=0 \), we must move camera frame

\[
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]