Vectors: the “Physical” View

- Object \( \mathbf{u} \) with direction and length/magnitude
- \(-\mathbf{u}\) is vector in opposite direction with same length
- \( \| \mathbf{u} \| \) denotes length of \( \mathbf{u} \)
  - Unit vector has length 1, is typically written with hat: \( \hat{\mathbf{u}} \)
  - Multiplying a vector by a scalar changes its length

Addition of Vectors

- “Parallelogram Rule”

\[
\mathbf{u} \oplus (-\mathbf{v}) \equiv \mathbf{u} - \mathbf{v}
\]

Dot Product

- If \( \theta \) is the angle between \( \mathbf{u} \) and \( \mathbf{v} \), then
  \[
  \mathbf{u} \cdot \mathbf{v} = \| \mathbf{u} \| \| \mathbf{v} \| \cos \theta
  \]
- This is also the length of the orthogonal projection of \( \mathbf{u} \) onto \( \mathbf{v} \), times the length of \( \mathbf{v} \)
- **Note:** \( \mathbf{u} \cdot \mathbf{u} = \| \mathbf{u} \|^2 \)

Orthogonal and Orthonormal Vectors

- **Orthogonal** vectors are at right angles to each other
  - \( \mathbf{u} \cdot \mathbf{v} = 0 \)
- **Orthonormal** vectors are at right angles to each other and each has unit length
  - A set \( \{ \mathbf{u}_i \} \) of orthonormal vectors has
    - \( \mathbf{u}_i \cdot \mathbf{u}_j = 1 \) if \( i = j \)
    - \( \mathbf{u}_i \cdot \mathbf{u}_j = 0 \) if \( i \neq j \)
Cross Product (only 3D)

- If \( \theta \) is the angle between \( \mathbf{u} \) and \( \mathbf{v} \), then
  \[
  \mathbf{u} \times \mathbf{v} = (\|\mathbf{u}\| \|\mathbf{v}\| \sin \theta) \hat{\mathbf{w}}
  \]
- \( \hat{\mathbf{w}} \) is a unit vector orthogonal to both \( \mathbf{u} \) and \( \mathbf{v} \)
- "Right-Hand Rule":
  - Curl fingers of right hand from \( \mathbf{u} \) to \( \mathbf{v} \)
  - Thumb points in direction of \( \hat{\mathbf{w}} \)

\[
\text{Area of parallelogram} = \| \mathbf{u} \times \mathbf{v} \|
\]

Position Vector

- Identifies point in space
- Interpreted as: tip of vector "arrow" when the other end is placed at a fixed point called the origin of the space

Vectors: the “Physical” View

- Directions and positions are different!
- "Legal" Operations:
  - Direction = Scalar * Direction
  - Direction = Direction + Direction
  - Position = Position + Direction
  - Direction = Position - Position
- "Illegal" Operations:
  - Position = Scalar * Position
  - Position = Position + Position
  - Direction = Position + Direction
  - Position = Position - Position

Cartesian/Euclidean \( n \)-space \( \mathbb{R}^n \)

- Vectors represented by real \( n \)-tuples of coordinates \((u_1, u_2, ..., u_n)\)
  - 2D: \((x, y)\)
  - 3D: \((x, y, z)\)
- Represents extent along orthogonal coordinate axes
  - Right-handed system: Curl right hand fingers from \( x \) axis to \( y \) axis, thumb points along \( z \) axis
- Length/magnitude: \( \| \mathbf{u} \| = (u_1^2 + u_2^2 + ... + u_n^2)^{1/2} \)
  - From Pythagoras' Theorem

Cartesian/Euclidean \( n \)-space \( \mathbb{R}^n \)

- Dot Product:
  \[
  \mathbf{u} \cdot \mathbf{v} \equiv (u_1v_1 + u_2v_2 + ... + u_nv_n)
  \]
- Cross Product (remember, 3D only!):
  \[
  \mathbf{u} \times \mathbf{v} \equiv (u_2v_3 - u_3v_2, u_3v_1 - u_1v_3, u_1v_2 - u_2v_1)
  \]
Another Way to Remember the Cross Product in $\mathbb{R}^3$

- Let $u = (x_1, y_1, z_1)$ and $v = (x_2, y_2, z_2)$
- $u \times v \equiv (y_1z_2 - z_1y_2, z_1x_2 - x_1z_2, x_1y_2 - y_1x_2)$

If $u$ and $v$ are 2D instead, this is the area of the parallelogram between them

Vectors: the “Mathematical” View

- Vector space: Set of objects (vectors) closed under
  - Addition of two vectors
  - Multiplication of a vector by a scalar (from field $F$)
- Necessary properties of vector spaces
  - Commutative addition: $x + y = y + x$
  - Associative addition: $(x + y) + z = x + (y + z)$
  - Existence of additive identity $0$: $x + 0 = 0 + x$
    - Can show: $0. x = 0$ (prove!)
  - Existence of additive inverse $-x$: $x + (-x) = 0$
    - Can show: $-x = -1. x$ (prove!)

Properties of vector spaces (contd.)

- Associative scalar multiplication: $r(sx) = (rs)x$
- Distributive scalar sum: $(r+s)x = rx + sx$
- Distributive scalar multiplication: $r(x + y) = rx + ry$
- Scalar multiplication identity: $1.x = x$
  - Note: This says that the identity for multiplication of two scalars is also the identity for multiplication of a vector by a scalar

Linear combination of vectors $\{u_i\}$: $\sum c_i u_i$

- $\{c_i\}$: Scalar coefficients
- $\{u_i\}$ is linearly independent if no $u_i$ can be expressed as a linear combination of the others

Span of $\{u_i\}$: Set of all linear combinations

Basis of vector space: Set of linearly independent vectors whose span is the entire space

Dimension of vector space: Cardinality of basis

Note: All bases have same cardinality

Basis and Dimension

- Linear combination of vectors $\{u_i\}$: $\sum c_i u_i$
  - $\{c\}$: Scalar coefficients
  - $\{u_i\}$ is linearly independent if no $u_i$ can be expressed as a linear combination of the others
- Span of $\{u_i\}$: Set of all linear combinations

Basis of vector space: Set of linearly independent vectors whose span is the entire space

Dimension of vector space: Cardinality of basis

Note: All bases have same cardinality

Basis

- Let a vector space have basis $B = \{u_1, u_2, ..., u_n\}$
- Any vector $u$ can be written as $\sum_{i=1}^n c_i u_i$
  - Or compactly, $(c_1, c_2, ..., c_n)$
  - These are the coordinates of $u$ in basis $B$

Note:

- We’ve generalized our original notion of coordinates: they’re now relative to the selected basis (axes)
- A point has different coordinates in different bases

Back to $\mathbb{R}^n$

- Has a basis $\{\hat{u}_1, \hat{u}_2, ..., \hat{u}_n\}$ along coordinate axes
  - $\hat{u}_1 = (1, 0, ..., 0)$
  - $\hat{u}_2 = (0, 1, ..., 0)$
    - ...
  - $\hat{u}_n = (0, 0, ..., 1)$
  - 2D: $\hat{x}, \hat{y}$
  - 3D: $\hat{x}, \hat{y}, \hat{z}$

Any $n$ linearly independent vectors form a basis

Note: Dot and cross products have same formula in right-handed orthonormal bases (but not in other bases!)
Change of Basis (example in 3D)

- Let \([u, v, n]\) be a basis \(B\) for a vector space.
- Let coordinates of \(u, v, n\) in another basis \(B_0\) be:
  \[u = (u_1, u_2, u_3)\]
  \[v = (v_1, v_2, v_3)\]
  \[n = (n_1, n_2, n_3)\]
- Let coordinates of \(a\) in \(B\) be \((a_1, a_2, a_3)\).
- Let coordinates of \(a\) in \(B_0\) be \((a_1^0, a_2^0, a_3^0)\).

\[
\begin{bmatrix}
a_1^0 \\
a_2^0 \\
a_3^0
\end{bmatrix} =
\begin{bmatrix}
u_1 & v_2 & v_3 \\
n_1 & n_2 & n_3
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3
\end{bmatrix}
\]

(only if bases are orthonormal)

Change of Basis (example in 3D)

- If \(B\) and \(B_0\) are orthonormal, inverse of matrix is its transpose (orthogonal matrix).

\[
\begin{bmatrix}
a_1^0 \\
a_2^0 \\
a_3^0
\end{bmatrix} =
\begin{bmatrix}
u_1 & v_2 & v_3 \\
n_1 & n_2 & n_3
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3
\end{bmatrix}
\]

Metric

- Assign non-negative real number \(d(u, v)\) to every pair of vectors \(u, v\).
- Interpret as distance between \(u\) and \(v\).

- Necessary properties of metric:
  - \(d(u, v) \geq 0\)
  - \(d(u, v) = 0\) if and only if \(u = v\)
  - \(d(u, v) = d(v, u)\)
  - \(d(u, v) \leq d(u, w) + d(w, v)\) (Triangle Inequality)

\[\mathbb{R}^n\] as Metric Space

- Magnitude: \(L_2\) norm: \(\|u\| = (u_1^2 + u_2^2 + \ldots + u_n^2)^{1/2}\)
- Metric \(d(u, v)\) defined as \(\|u - v\|\)
- Other common magnitude functions: \(L_p\) norms
  \[\|u\| = (|u_1|^p + |u_2|^p + \ldots + |u_n|^p)^{1/p}\]
  - \(L_1\) is “Manhattan” norm
  - \(L_{\infty}\) is “max” norm

Inner Product

- Assign scalar \(\langle u, v \rangle \in \mathbb{F}\) to pair of vectors \(u, v\).
  - \(F\) is assumed to be set of real or complex numbers.

- Necessary properties of inner product:
  - \(\langle u, v \rangle = \overline{\langle v, u \rangle}\) (overline denotes complex conjugate)
  - \(\langle su, v \rangle = s\langle u, v \rangle\)
  - \(\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle\)

- The dot product is a valid inner product for \(\mathbb{R}^n\).
Transformations of Vectors
(Thanks to Pat Hanrahan for this section)

- \( \mathbf{u}' = T(\mathbf{u}) \)
- Why?
  - **Modeling:**
    - Create objects in convenient coordinates
    - Multiple instances of prototype shape
    - Kinematics of linked structures
  - **Viewing:**
    - Map between window and device coordinates
    - Virtual camera projections: parallel/perspective
- We'll stick to \( \mathbb{R}^2 \) for now
Composing Transformations

- Rotate, then Translate

- Translate, then Rotate

Order Matters!

- \( T(1, 1) \cdot R(45) \neq R(45) \cdot T(1, 1) \)

World Space and Object Space

- Transformation maps from one to the other
- Construct by composing sequence of basic transforms
  - \textbf{Remember:} Transforms apply \textit{Right-to-Left} in our notation!

World Space and Object Space

- Let’s look at the example on the right
  - Object → World:
    - Rotate by \( \theta \) (ccw), then translate by \( t \)
  - World → Object:
    - Translate by \(-t\), then rotate by \(-\theta\)

Make sure you understand this!

Translation

- \( x' = x + t_x \)
- \( y' = y + t_y \)
Scaling (and Reflection)

\[ x' = s \cdot x \]
\[ y' = s \cdot y \]
(Negative scaling coefficients give reflection)

CCW Rotation By \( \theta \) About Origin

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

Horizontal Shear

\[ x' = x + s \cdot y \]
\[ y' = y \]

Vertical Shear

\[ x' = x \]
\[ y' = s \cdot x + y \]

Types of 2D Transformations

- **Linear Transforms**: \( T(u + v) = T(u) + T(v) \)
  - Scaling
  - Rotation
  - Shear
  - Reflection
- **Affine Transforms**: \( T(u) = L(u) + a \), where \( L \) is linear and \( a \) is a fixed vector
  - Translation
- Other, e.g. perspective projection
- How do we represent these in a common format?

Homogenous Coordinates (2D)

- Point \((x, y) \rightarrow (x, y, 1)\)
- Direction \((x, y) \rightarrow (x, y, 0)\)
- For any scalar \(c\), \((cx, cy, ch) \equiv (x, y, h)\)
- To convert back:
  - If \(h\) is 0: \((x, y, 0) \rightarrow (x, y)\)
  - If \(h\) is non-zero: \((x, y, h) \rightarrow (x/h, y/h)\)
- **Note**:
  - Not 3D vector space, just a new representation for 2D
  - Legal/illegal operations for directions & positions automatically distinguished!
Translation

\[ x' = x + t_x \]
\[ y' = y + t_y \]

\[
\begin{bmatrix}
  x' \\ y' \\ 1
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\ y \\ 1
\end{bmatrix}
\]

Scaling (and Reflection)

\[ x' = s_x x \]
\[ y' = s_y y \]

\[
\begin{bmatrix}
  x' \\ y' \\ 1
\end{bmatrix} = \begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\ y \\ 1
\end{bmatrix}
\]

CCW Rotation By \( \theta \) About Origin

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

\[
\begin{bmatrix}
  x' \\ y' \\ 1
\end{bmatrix} = \begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\ y \\ 1
\end{bmatrix}
\]

Horizontal Shear

\[ x' = x + sy \]
\[ y' = y \]

\[
\begin{bmatrix}
  x' \\ y' \\ 1
\end{bmatrix} = \begin{bmatrix}
  1 & s & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\ y \\ 1
\end{bmatrix}
\]

Vertical Shear

\[ x' = x \]
\[ y' = sx + y \]

\[
\begin{bmatrix}
  x' \\ y' \\ 1
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & 0 \\
  s & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\ y \\ 1
\end{bmatrix}
\]

What about 3D?

- Very similar: \((x, y, z) \rightarrow (x, y, z, h)\)
- Look up the formulæ!
- Rotation is a mess
  - Common method:
    - Map axis of rotation to a coordinate axis (similar to change of basis)
    - Rotate around the coordinate axis
    - Map back
  - Other approaches based on Euler angles and quaternions
**Why Use Matrices?**

- Compute the matrix once
  
  \[
  x' = x \cos \theta - y \sin \theta \\
  y' = x \sin \theta + y \cos \theta
  \]

- Don’t repeatedly evaluate sines and cosines

- Combine sequence of transforms into a single transform

  Store \( M = ABCD \), apply \( M(u) \) instead of \( A(B(C(D(u)))) \)

---

**Hierarchical Modeling**

- Graphics systems maintain a **current transformation matrix** (CTM)
  
  - All geometry is transformed by the CTM
  - CTM defines object space in which geometry is specified
  - Transformation commands are concatenated onto the CTM. The last one added is applied first:
    
    \[
    C = C \times T
    \]

- Graphics systems also maintain a **transformation stack**
  
  - The CTM can be pushed onto the stack
  - The CTM can be restored from the stack

---

**Example: Articulated Robot**

- Translate \( (0, 5, 0) \)
- Torso
- Push Matrix
- Translate \( (0, 5, 0) \)
- Shoulder
- Push Matrix
- Rotate \( Y \text{ neck}_y \)
- Rotate \( X \text{ neck}_x \)
- Head
- Pop Matrix
- Push Matrix
- Translate \( (1.5, 0, 0) \)
- Rotate \( X \text{ l_shoulder}_x \)
- Upper Arm
- Push Matrix
- Translate \( (0, -2, 0) \)
- Rotate \( X \text{ l_elbow}_x \)
- Lower Arm
- Pop Matrix
- Pop Matrix

---

[Diagram of an articulated robot with various joints and labels for body, torso, head, shoulders, arms, hands, hips, left and right legs, and feet.]

---

[Code snippet for the robot, including translation and matrix operations.]