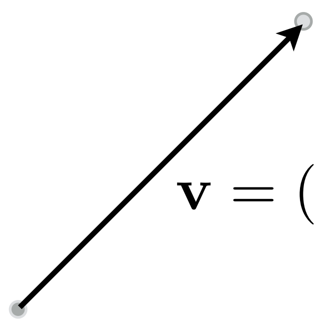
Math Review

Prof. Fabio Pellacini
[original slides by Prof. Steve Marschner]

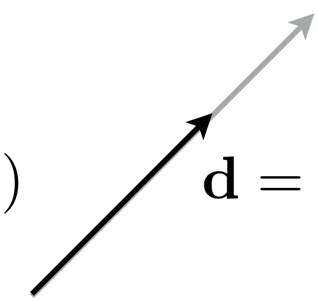
Geometric Quantities

- **Point**: location in space
- **Vector**: orientation with an associated magnitude
  - difference between points
- **Direction**: orientation with unit magnitude
- In cartesian coordinates, representation with 3 numbers

$$\mathbf{p} = (p_x, p_y, p_z)$$



$$\mathbf{v} = (v_x, v_y, v_z)$$



$$\mathbf{d} = (d_x, d_y, d_z)$$

Linear Algebra Vectors

- Most graphics software uses linear algebra vectors for all points, directions and (geometric) vectors
- Indicated with 3 cartesian coordinates

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

- Typical Implementation using simple structs

```
struct vec3f {
  float x, y, z;
}
```

Vector Arithmetic

- Linear algebra operations: sum and scalar product

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_x + v_x \\ u_x + v_y \\ u_z + v_z \end{bmatrix} \qquad s\mathbf{v} = \begin{bmatrix} sv_x \\ sv_y \\ sv_z \end{bmatrix}$$

- Dot product: scalar equal to the product of the vectors' lengths times the cosine between them
  - used to check whether two vectors are orthogonal

$$\mathbf{u} \cdot \mathbf{v} = u_x v_x + u_y v_y + u_z v_z = |\mathbf{u}||\mathbf{v}| \cos\theta$$

- Vector length: square root of the dot product

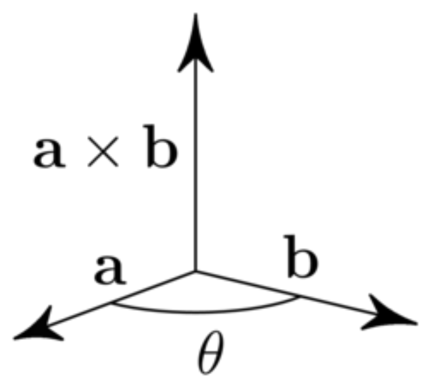
$$l = \sqrt{v_x^2 + v_y^2 + v_z^2} = \sqrt{\mathbf{v} \cdot \mathbf{v}}$$

Vector Arithmetic

- Cross product: vector orthogonal to the operands, whose length is equal to the product of the vectors' lengths times the sine between them
- Used to construct orthonormal vectors

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{bmatrix}$$

$$|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\mathbf{v}| \sin\theta$$
$$(\mathbf{u} \times \mathbf{v}) \cdot \mathbf{u} = 0$$
$$(\mathbf{u} \times \mathbf{v}) \cdot \mathbf{v} = 0$$

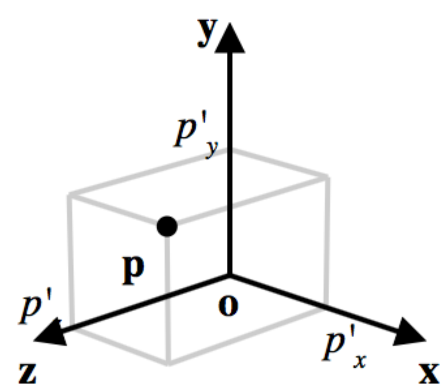


Coordinate Systems

- Points, vectors, directions are defined w.r.t. a coordinate system
- Coordinate systems, or frames, specify an origin and three axis
  - if not defined, we intend the "unit" or "world" frame

$$F = \{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{o}\}$$

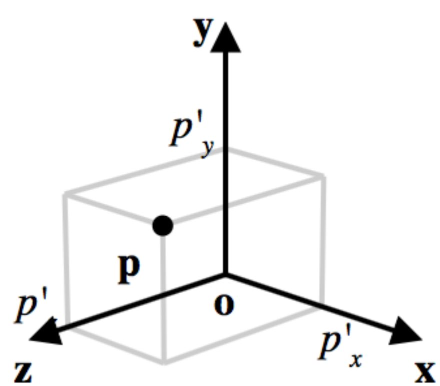
```
struct frame3f {
  vec3f x, y, z, o;
}
```



Coordinate Systems

- Given a frame defined w.r.t world space, we can transform a point coordinates to/from the coordinate frame
- World to local: subtract origin, project onto the frame axes
- Local to world: add origin to the scaled frame axes

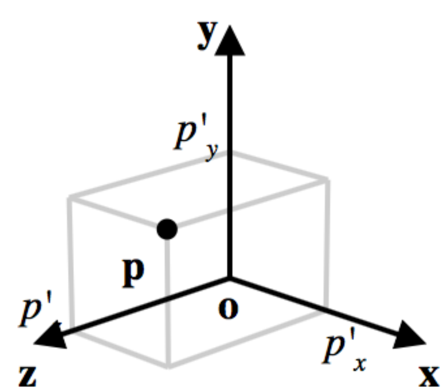
$$\mathbf{p}^l = \begin{bmatrix} (\mathbf{p}-\mathbf{o})\cdot\mathbf{x} \\ (\mathbf{p}-\mathbf{o})\cdot\mathbf{y} \\ (\mathbf{p}-\mathbf{o})\cdot\mathbf{z} \end{bmatrix} \qquad \mathbf{p} = \mathbf{o} + p^l_x\mathbf{x} + p^l_y\mathbf{y} + p^l_z\mathbf{z}$$



Coordinate Systems

- Vector transforms as difference of points
- Boils down to ignoring the origin

$$\mathbf{v}^l = \begin{bmatrix} \mathbf{v} \cdot \mathbf{x} \\ \mathbf{v} \cdot \mathbf{y} \\ \mathbf{v} \cdot \mathbf{z} \end{bmatrix} \qquad \mathbf{v} = v^l_x \mathbf{x} + v^l_y \mathbf{y} + v^l_z \mathbf{z}$$



Matrices

- Consider only 3x3 matrices for now

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

- Write in column notation

$$M = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \end{bmatrix} = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix}$$

Matrices

- Matrix-vector multiply: weighted sum of matrix' columns

$$M\mathbf{v} = v_x\mathbf{x} + v_y\mathbf{y} + v_z\mathbf{z}$$

- Matrix-matrix multiply: columns are matrix-vector multiplies

$$MM' = \left[M\mathbf{x}', M\mathbf{y}', M\mathbf{z}'\right]$$

- Matrix inverse
  - analytic for small matrices, numerical for all others

$$MM^{-1} = M^{-1}M = I$$

Matrices

- Transpose: flip the row and columns

$$M = \left[m_{ij} \right] \rightarrow M^T = \left[M_{ji} \right]$$

- Orthonormal matrices: columns are orthogonal to each other and normalized

$$M_o = \left[\mathbf{c}_i \right] \Leftrightarrow \mathbf{c}_i \cdot \mathbf{c}_j = \begin{cases} 0 & \text{for } i = j \\ 1 & \text{for } i = j \end{cases}$$

- Orthonormal inverse is its transpose

$$M_o^{-1} = M_o^T$$

Matrices and Frames

- Define a matrix whose columns are the coordinate frame axes

$$F = \{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{o}\} \quad M = \begin{bmatrix}\mathbf{x} & \mathbf{y} & \mathbf{z}\end{bmatrix}$$

- Matrix is orthonormal
- Express coordinate transformations with algebra operations

$$\mathbf{p} = M\mathbf{p}^l + \mathbf{o} \qquad \mathbf{v} = M\mathbf{v}^l$$

$$\mathbf{p}^l = M^T(\mathbf{p} - \mathbf{o}) \qquad \mathbf{v}^l = M^T\mathbf{v}$$

Constructing Frames

- Make a vector orthogonal to another by subtracting its projection

$$\mathbf{u}_\perp = \mathbf{u} - (\mathbf{u} \cdot \mathbf{v})\mathbf{v}$$

- Create a frame from **z**’ and **y**’
  - normalize **z’**
  - make **y’** orthogonal to **z** and normalize it
  - compute **x** as cross product of **z** and **y**

$$\mathbf{z} = \frac{\mathbf{z}'}{|\mathbf{z}'|} \quad \mathbf{y} = \frac{\mathbf{y}' - (\mathbf{y}' \cdot \mathbf{z})\mathbf{z}}{|\mathbf{y}' - (\mathbf{y}' \cdot \mathbf{z})\mathbf{z}|} \quad \mathbf{x} = \mathbf{z} \times \mathbf{y}$$

Implicit curves

- Equation to tell whether we are on the curve or surface

$$\{\mathbf{v} \,|\, f(\mathbf{v}) = 0\}$$

- Example: line (orthogonal to u, distance k from 0)

$$\{\mathbf{v} \,|\, \mathbf{v} \cdot \mathbf{u} + k = 0\}$$

(u is a unit vector)

- Example: circle (center p, radius r)

$$\{\mathbf{v} \,|\, (\mathbf{v} - \mathbf{p}) \cdot (\mathbf{v} - \mathbf{p}) - r^2 = 0\}$$

- Always define boundary of region
  - (if f is continuous)

Explicit curves

- Also called parametric
- Equation to map domain into plane

  $$\{f(t) \,|\, t \in D\}$$

- Example: line (containing p, parallel to u)

  $$\{\mathbf{p} + t\mathbf{u} \,|\, t \in \mathbb{R}\}$$

- Example: circle (center b, radius r)

  $$\{\mathbf{p} + r[\cos t \;\; \sin t\,]^T \,|\, t \in [0, 2\pi)\}$$

- Like tracing out the path of a particle over time
- Variable *t* is the "parameter"

Algebra-vs-geometry views

- In the previous slides we often changed between algebraic notation to geometric concepts
- This is very common in graphics
- In general, we tend to use algebraic concepts as much as possible since they are simpler to handle and easier to implement