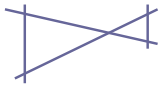
A Review of Some Math Needed for Computer Graphics

Mathematical Foundations

- I'll give a brief, informal review of some of the mathematical tools of CG
  - Geometry (2D, 3D)
  - Trigonometry
  - Vector and affine spaces
    - Points, vectors, and coordinates
  - Dot and cross products
  - Linear transforms and matrices
- Bear with me…

2D Geometry

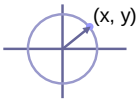
- Know your high-school geometry:
  - Total angle around a circle is 360° or 2 π radians
  - When two lines cross:
    - Opposite angles are equivalent
    - Angles along line sum to 180°



  - Similar triangles:
    - All corresponding angles are equivalent
    - Corresponding pairs of sides have the same length ratio and are separated by equivalent angles
    - Any corresponding pairs of sides have same length ratio

Trigonometry

- Sine: "opposite over hypotenuse"
- Cosine: "adjacent over hypotenuse"
- Tangent: "opposite over adjacent"
- Unit circle definitions:
  - sin (Θ) = y
  - cos (Θ) = x
  - tan (Θ) = y/x
  - Etc…



3D Geometry

- To model, animate, and render 3D scenes, we must specify:
  - Location
  - Displacement from arbitrary locations
  - Orientation
- We'll look at two types of spaces:
  - *Vector spaces*
  - *Affine spaces*
- We will often be sloppy about the distinction

Geometric Preliminaries

- **Affine Geometry**
  - Scalars + Points + Vectors and their operations
- **Euclidian Geometry**
  - Affine Geometry lacks angles, distance
  - New op: Inner/Dot product, which gives
    - Length, distance, normalization
    - Angle, Orthogonality, Orthogonal projection
- **Projective Geometry**

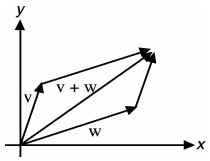
Vector Spaces

- Two types of elements:
  - Scalars (real numbers): $\alpha, \beta, \gamma, \delta, \ldots$
  - Vectors (n-tuples): **u**, **v**, **w**, …
- Supports two operations:
  - Addition operation **u + v**, with:
    - Identity **0** **v** + **0** = **v**
    - Inverse **-** **v** + (-**v**) = **0**
  - Scalar multiplication:
    - Distributive rule: $\alpha(\mathbf{u} + \mathbf{v}) = \alpha(\mathbf{u}) + \alpha(\mathbf{v})$

      $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$

Vector Preliminaries

- Vector: an *n*-tuple of real numbers
- Vector Operations
  - Vector addition: $u + v = w$
    - Commutative, associative, identity element (0)
  - Scalar multiplication: cv



- Note: Vectors and Points are different
  - Can not add points
  - Can find the vector between two points

Vector Spaces

- A *linear combination* of vectors results in a new vector:

  $\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n$
- If the only set of scalars such that

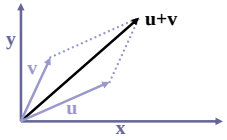
  $\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n = \mathbf{0}$

  is $\alpha_1 = \alpha_2 = \ldots = \alpha_3 = \mathbf{0}$

  then we say the vectors are *linearly independent*
- The *dimension* of a space is the greatest number of linearly independent vectors possible in a vector set
- For a vector space of dimension *n*, any set of *n* linearly independent vectors form a *basis*

Vector Spaces: A Familiar Example

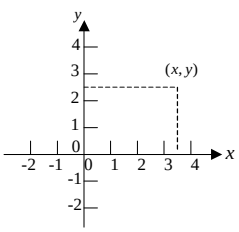
- Our common notion of vectors in a 2D plane is a vector space:
  - Vectors are directions ("arrows") rooted at the origin
  - Scalar multiplication "streches" the arrow, changing its length (*magnitude*) but not its direction
  - Addition uses the "trapezoid rule":



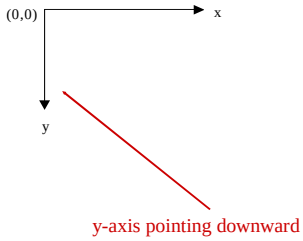
Vector Spaces: Basis Vectors

- Given a basis for a vector space:
  - Each vector in the space is a *unique* linear combination of the basis vectors
  - The *coordinates* of a vector are the scalars from this linear combination
  - Best-known example: Cartesian coordinates
    - Draw example on the board
  - Note that a given vector **v** will have different coordinates for different bases

2D Coordinate System



Java 2D Coordinate System

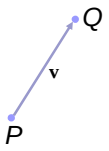


Vectors And Points

- We commonly use vectors to represent:
  - Direction (i.e., orientation)
  - Points in space (i.e., location)
  - Displacements from point to point
- But we want points and directions to behave differently
  - Ex: To *translate* something means to move it without changing its orientation
  - Translation of a point = different point
  - Translation of a direction = same direction

Affine Spaces

- To be more rigorous, we need an explicit notion of *position*
- *Affine spaces* add a third element to vector spaces: *points* (*P, Q, R, …*)
- Points support these operations
  - Point-point subtraction: $Q - P = \mathbf{v}$
    - Result is a vector pointing from *P* to *Q*
  - Vector-point addition: $P + \mathbf{v} = Q$
    - Result is a new point
    - $P + \mathbf{0} = P$
  - Note that the addition of two points is not defined

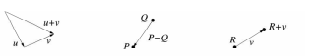


Affine Spaces

- Points, like vectors, can be expressed in coordinates
  - The definition uses an affine combination
  - Net effect is same: expressing a point in terms of a basis
- Thus the common practice of representing points as vectors with coordinates
- Be careful to avoid nonsensical operations
  - Point + point
  - Scalar * point

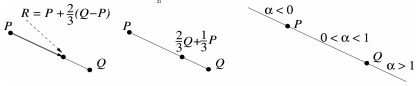
Affine Geometry

- *Affine Operations*:



$vector \leftarrow scalar \cdot vector, \quad vector \leftarrow vector / scalar$	scalar-vector multiplication
$vector \leftarrow vector + vector, \quad vector \leftarrow vector - vector$	vector-vector addition
$vector \leftarrow point - point$	point-point difference
$point \leftarrow point + vector, \quad point \leftarrow point - vector$	point-vector addition

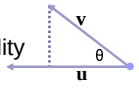
- *Affine Combinations*: $\alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_n P_n$, where $P_1, P_2, \ldots, P_n$ are points and $\Sigma_i \alpha_i = 1$
  Example: $R = (1-\alpha)P + \alpha Q$



Dot Product

- The *dot product* or, more generally, *inner product* of two vectors is a scalar:

  $\mathbf{v}_1 \bullet \mathbf{v}_2 = \mathbf{x}_1\mathbf{x}_2 + \mathbf{y}_1\mathbf{y}_2 + \mathbf{z}_1\mathbf{z}_2$ (in 3D)
- Useful for many purposes
  - Computing the length of a vector: *length*(**v**) = sqrt(**v** • **v**)
  - *Normalizing* a vector, making it unit-length
  - Computing the angle between two vectors:
    u • **v** = |**u**| |**v**| cos(θ)
  - Checking two vectors for orthogonality
  - *Projecting* one vector onto another



Linear Combinations & Dot Products

- A *linear combination* of the vectors $v_1, v_2, \ldots v_n$ is any vector of the form $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$ where α_i is a real number (i.e. a scalar)
- *Dot Product*: $u \cdot v = \sum_{k=1}^{n} u_k v_k$ a real value $u_1 v_1 + u_2 v_2 + \ldots + u_n v_n$ written as $u \bullet v$

Fun with Dot Products

- *Euclidian Distance* from *(x,y)* to *(0,0)* $\sqrt{x^2 + y^2}$ in general: $\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$ which is just: $\sqrt{\vec{x} \bullet \vec{x}}$
- This is also the length of vector $\underline{v}$: $||\underline{v}||$ or $|\underline{v}|$
- *Normalization* of a vector: $\hat{v} = \frac{\vec{v}}{|\vec{v}|}$.
- *Orthogonal* vectors: $\vec{u} \cdot \vec{v} = 0$

Cross Product

- The *cross product* or *vector product* of two vectors is a vector:

$$\mathbf{v}_1 \times \mathbf{v}_2 = \begin{bmatrix} y_1 z_2 - y_2 z_1 \\ -(x_1 z_2 - x_2 z_1) \\ x_1 y_2 - x_2 y_1 \end{bmatrix}$$

- Cross product of two vectors is orthogonal to both
- Right-hand rule dictates direction of cross product
- Cross product is handy for finding surface orientation
  - Lighting
  - Visibility