Computer Graphics

Vector Tools for Graphics

Time for some math

- We're going to review some of the basic mathematical constructs used in computer graphics
  - Scalars
  - Points
  - Vectors
  - Matrices
  - Other stuff (rays, planes, etc.)
- Why?
  - Most of computer graphics is defined in 3D
    - 2D is only a special case
  - Vector analysis and transformations are crucial to 3D graphics

Scalars

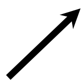
- A <u>scalar</u> is a quantity that does not depend on direction
  - In other words, it's just a regular number
    - *i.e.* 7 is a scalar
    - so is 13.5
    - or -4

Points

- A <u>point</u> is a list of n numbers referring to a location in n-Dimension
- The individual components of a point are often referred to as <u>coordinates</u>
  - *i.e.* (2, 3, 4) is a point in 3-D space
    - This point's x-coordinate is 2, it's y-coordinate is 3, and it's z-coordinate is 4

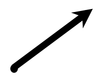
Vectors

- A vector is a list of *n* numbers referring to a direction and magnitude in n-D
- From a data structures perspective, a vector looks exactly the same as a point
  - *i.e.* (2, 3, 4) is a vector in 3-D space
    - Vector does not have a fixed position



Rays

- A ray is just a vector with a starting point
  - Ray = (Point, Vector)
- Let a ray be defined by point **p** and vector **d**
- The parametric form of a ray expresses it as a function as some scalar t, giving the set of all points the ray passes through:
  - $r(t) = \mathbf{p} + t\mathbf{d}, 0 \leq t \leq \infty$



Vectors

- We said that a vector encodes a direction and a magnitude in n-D
  - How does it do this?
- Here are two ways to denote a vector in 2-D:

$$\mathbf{V} = \langle V_x, V_y \rangle$$

$$\mathbf{V} = \begin{bmatrix} V_x \\ V_y \end{bmatrix}$$

Vector Magnitude

- Geometrically, the magnitude of a vector is the Euclidean distance between its start and end points, or more simply, it's length
- Vector magnitude in n-D: $||\mathbf{V}|| = \sqrt{\sum_{i=1}^{n} V_i^2}$
- Vector magnitude in 2-D: $||\mathbf{V}|| = \sqrt{V_x^2 + V_y^2}$

Normalized Vectors

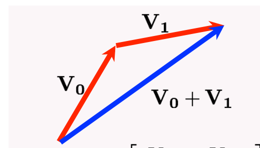
- Most of the time, we want to deal with normalized, or unit, vectors
- This means that the magnitude of the vector is 1: $||\mathbf{V}|| = 1$
- We can normalize a vector by dividing the vector by its magnitude:

$$\hat{V} = \frac{V}{||\mathbf{V}||}$$

Vector Addition

- Vectors are closed under addition
  - Head to tail
  - Vector + Vector = Vector

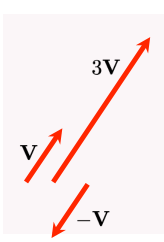
Vector Addition



$$\mathbf{V}_0 + \mathbf{V}_1 = \begin{bmatrix} V_{0,x} + V_{1,x} \\ V_{0,y} + V_{1,y} \end{bmatrix}$$

Vector Scaling

- Vectors are closed under multiplication with a scalar
  - Scalar * Vector = Vector



Vector Scaling

$$a\mathbf{V} = \begin{bmatrix} aV_x \\ aV_y \end{bmatrix}$$

Properties of Vector Addition & Scaling

Addition is Commutative $\quad \mathbf{P} + \mathbf{Q} = \mathbf{Q} + \mathbf{P}$

Addition is Associative $\quad (\mathbf{P} + \mathbf{Q}) + \mathbf{R} = \mathbf{P} + (\mathbf{Q} + \mathbf{R})$

Scaling is Commutative and Associative $\quad (ab)\mathbf{P} = a(b\mathbf{P})$

Scaling and Addition are Distributive

$$a(\mathbf{P} + \mathbf{Q}) = a\mathbf{P} + a\mathbf{Q}$$

$$(a + b)\mathbf{P} = a\mathbf{P} + b\mathbf{P}$$

Points and Vectors

- Can define a vector by 2 points
  - Point - Point = Vector
- Can define a new point by a point and a vector
  - Point + Vector = Point

Vector Multiplication?

- What does it mean to multiply two vectors?
  - Not uniquely defined
- Two product operations are commonly used:
  - Dot (scalar, inner) product
    - Result is a scalar
  - Cross (vector, outer) product
    - Result is a new vector

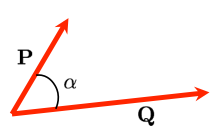
Dot Product Application: Lighting

- P • Q = ||P||• ||Q|| cos(α)
- So what does this mean if P and Q are normalized?
  - Can get cos(α) for just 3 multiplies and 2 adds (in 3D)
    - Very useful in lighting and shading calculations
    - Example: Lambert's cosine law

Dot Product

$$\mathbf{P} \cdot \mathbf{Q} = \sum_{i=1}^{n} P_i Q_i = \begin{bmatrix} P_1 & P_2 & \ldots & P_n \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ \ldots \\ Q_n \end{bmatrix}$$

$$\mathbf{P} \cdot \mathbf{Q} = ||\mathbf{P}||\ ||\mathbf{Q}||\ \cos\alpha$$

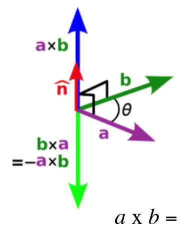


$$\alpha = cos^{-1}\left(\frac{\mathbf{P} \cdot \mathbf{Q}}{||\mathbf{P}||\ ||\mathbf{Q}||}\right)$$

Properties of Vector Dot Products

Commutative	$\mathbf{P}\cdot\mathbf{Q}=\mathbf{Q}\cdot\mathbf{P}$
Associative with Scaling	$(a\mathbf{P})\cdot\mathbf{Q}=a(\mathbf{P}\cdot\mathbf{Q})$
Distributive with Addition	$\mathbf{P}\cdot(\mathbf{Q}+\mathbf{R})=\mathbf{P}\cdot\mathbf{Q}+\mathbf{P}\cdot\mathbf{R}$
	$\mathbf{P}\cdot\mathbf{P}=\lvert\lvert\mathbf{P}\rvert\rvert^2$
	$\lvert\mathbf{P}\cdot\mathbf{Q}\rvert\leq\lvert\lvert\mathbf{P}\rvert\rvert\;\lvert\lvert\mathbf{Q}\rvert\rvert$

Cross Product



$$\mathbf{a}\times\mathbf{b}=\det\begin{bmatrix}\mathbf{i}&\mathbf{j}&\mathbf{k}\\a_1&a_2&a_3\\b_1&b_2&b_3\end{bmatrix}.$$

$$a \times b = i\begin{vmatrix}a_2&a_3\\b_2&b_3\end{vmatrix}-j\begin{vmatrix}a_1&a_3\\b_1&b_3\end{vmatrix}+k\begin{vmatrix}a_1&a_2\\b_2&b_2\end{vmatrix}$$

$$=i(a_2b_3-a_3b_2)-j(a_1b_3-a_3b_1)+k(a_1b_2-a_2b_2)$$

Cross Product Application: Normals

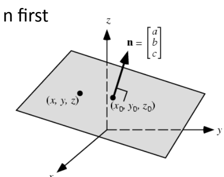
- A normal (or surface normal) is a vector that is perpendicular to a surface at a given point
  - This is often used in lighting calculations
- The cross product of 2 orthogonal vectors on the surface is a vector perpendicular to the surface
  - Can use the cross product to compute the normal

Planes

- How can we define a plane?
  - 3 non-linear points
  - A perpendicular vector and an incident point
    - n • (x-x$_0$) = 0
      - ax + by + cz + d = 0
    - Hessian normal form: Normalize n first
      - n • x = - p

$$n_x=\frac{a}{\sqrt{a^2+b^2+c^2}}$$
$$n_y=\frac{b}{\sqrt{a^2+b^2+c^2}}$$
$$n_z=\frac{c}{\sqrt{a^2+b^2+c^2}}$$

$$p=\frac{d}{\sqrt{a^2+b^2+c^2}}$$



Columns and Rows

- In this class, we will generally assume that a list forms a column vector:

$$(a, b, c, d) \Longrightarrow \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

- The reason for this will become clear when we talk about matrices

Matrices

- Reminder: A matrix is a rectangular array of numbers
  - An m x n matrix has m rows and n columns
- M_{ij} denotes the entry in the i-th row and j-th column of matrix M
  - These are generally thought of as 1-indexed
    - instead of 0-indexed
- Here, M is a 2x5 matrix:

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} \end{bmatrix}$$

Matrix Transposes

- The transpose of an m x n matrix is an n x m matrix
  - Denoted M^T
  - $M^T_{ij} = M_{ji}$

$$\mathbf{M}^T = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} \end{bmatrix}^T = \begin{bmatrix} M_{11} & M_{21} \\ M_{12} & M_{22} \\ M_{13} & M_{23} \\ M_{14} & M_{24} \\ M_{15} & M_{25} \end{bmatrix}$$

Matrix Addition

- Only well defined if the dimensions of the 2 matrices are the same
  - That is, $m_1 = m_2$ and $n_1 = n_2$
  - Here, M and G are both 2 x 5

$$(\mathbf{M} + \mathbf{G})_{ij} = M_{ij} + G_{ij}$$

$$\mathbf{M} + \mathbf{G} = \begin{bmatrix} M_{11}+G_{11} & M_{12}+G_{12} & M_{13}+G_{13} & M_{14}+G_{14} & M_{15}+G_{15} \\ M_{21}+G_{21} & M_{22}+G_{22} & M_{23}+G_{23} & M_{24}+G_{24} & M_{25}+G_{25} \end{bmatrix}$$

Matrix Scaling

– Just like vector scaling
  - Matrix * Scalar = Matrix

$$(a\mathbf{M})_{ij} = aM_{ij}$$

$$a\mathbf{M} = \begin{bmatrix} aM_{11} & aM_{12} & aM_{13} & aM_{14} & aM_{15} \\ aM_{21} & aM_{22} & aM_{23} & aM_{24} & aM_{25} \end{bmatrix}$$

Properties of Matrix Addition and Scaling

Addition is Commutative	$\mathbf{F}+\mathbf{G}=\mathbf{G}+\mathbf{F}$
Addition is Associative	$(\mathbf{F}+\mathbf{G})+\mathbf{H}=\mathbf{F}+(\mathbf{G}+\mathbf{H})$
Scaling is Associative	$a(b\mathbf{F})=(ab)\mathbf{F}$
Scaling and Addition are Distributive	$a(\mathbf{F}+\mathbf{G})=a\mathbf{F}+a\mathbf{G}$
	$(a+b)\mathbf{F}=a\mathbf{F}+b\mathbf{F}$

Matrix Multiplication

– Only well defined if the number of columns of the first matrix and the number of rows of the second matrix are the same
  - Matrix * Matrix = Matrix
  - *i.e.* if F is m x n, and G is n x p, then FG if m x p

– Let's do an example

$$(\mathbf{FG})_{ij} = \sum_{k=1}^{m} F_{ik}G_{kj}$$

The Identity Matrix

- Defined such that the product of any matrix M and the identity matrix I is M
  - **IM** = **MI** = **M**
- The identity matrix is a square matrix with ones on the diagonal and zeros elsewhere

$$(\mathbf{I}_n)_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \qquad \mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Back to Graphics…

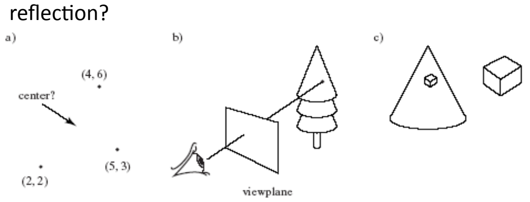
- In computer graphics, we work with objects defined in a three dimensional world
- All objects to be drawn, and the cameras used to draw them, have shape, position, and orientation.
- We must write computer programs that somehow
  - describe these objects
  - describe how light bounces around illuminating them
  - so that the final pixel values on the display can be computed.

Back to Graphics…

- The two fundamental sets of tools that come to our aid in graphics are *vector analysis* (Ch. 4) and *transformations* (Ch. 5).
- We develop methods to describe various geometric objects, and we learn how to convert geometric ideas to numbers.
- This provides a collection of crucial algorithms that we can use in graphics programs.

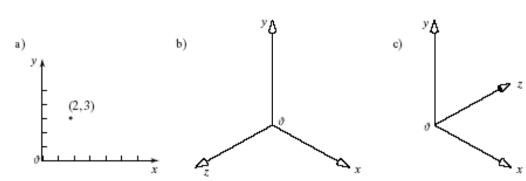
Easy Problems for Vectors

- Where is the center of the circle through the 3 points?
- What image shape appears on the viewplane, and where?
- Where does the reflection of the cube appear on the shiny cone, and what is the exact shape of the reflection?



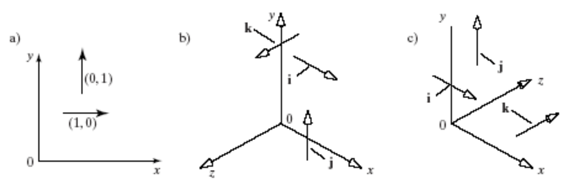
Basics of Points and Vectors

- All points and vectors are defined relative to some coordinate system. Shown below are a 2D coordinate system and a right- and a left-handed 3-D coordinate system.



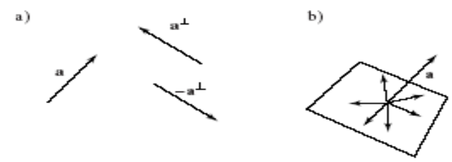
Standard Unit Vectors

- The standard unit vectors in 3D are **i** = (1,0,0), **j** = (0, 1, 0), and **k** = (0, 0, 1). **k** always points in the positive z direction
- In 2D, **i** = (1,0) and **j** = (0, 1).
- The unit vectors are orthogonal.



Finding a 2D "Perp" Vector

- If vector **a** = (a_x, a_y), then the vector perpendicular to **a** in the *counterclockwise* sense is $\mathbf{a}^{\perp}$ = ($-a_y$, a_x), and in the *clockwise* sense it is $-\mathbf{a}^{\perp}$.
- In 3D, any vector in the plane perpendicular to **a** is a "perp" vector.

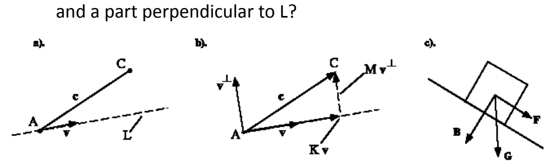


Properties of $\perp$

- $(\mathbf{a} \pm \mathbf{b})^{\perp} = \mathbf{a}^{\perp} \pm \mathbf{b}^{\perp}$
- $(s\mathbf{a})^{\perp} = s(\mathbf{a}^{\perp})$
- $(\mathbf{a}^{\perp})^{\perp} = -\mathbf{a}$
- $\mathbf{a}^{\perp} \cdot \mathbf{b} = -\mathbf{b}^{\perp} \cdot \mathbf{a} = -a_y b_x + a_x b_y$
- $\mathbf{a}^{\perp} \cdot \mathbf{a} = a \cdot \mathbf{a}^{\perp} = 0$
- $|\mathbf{a}^{\perp}| = |\mathbf{a}|$

Orthogonal Projections and Distance from a Line

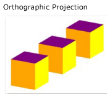
- We are given 2 points A and C and a vector **v**. The following questions arise:
  - How far is C from the line L that passes through A in direction **v**?
  - If we drop a perpendicular onto L from C, where does it hit L?
  - How do we decompose a vector **c** = C – A into a part along L and a part perpendicular to L?



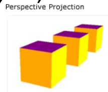
Answering the Questions

- We may write $\mathbf{c} = K\mathbf{v} + M\mathbf{v}^{\perp}$.
- If we take the dot product of each side with $\mathbf{v}$,
  - we obtain $\mathbf{c}\cdot\mathbf{v} = K\mathbf{v}\cdot\mathbf{v} + M\mathbf{v}^{\perp}\cdot v = K|\mathbf{v}|^2$,
  - or $K = \mathbf{c}\cdot\mathbf{v}/|\mathbf{v}|^2$.
- Likewise, taking the dot product with $\mathbf{v}^{\perp}$
  - gives $M = \mathbf{c}\cdot\mathbf{v}^{\perp}/|\mathbf{v}|^2$.
- Answers to the original questions: $M\mathbf{v}^{\perp}$, $K\mathbf{v}$, and both.
- When might this be useful?

Orthographic Projection



Perspective Projection



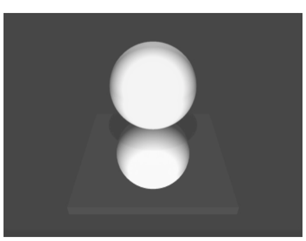
Practice Question

- Find the projection of the vector C=(6, 4) onto v=(1,2)
- How far is the point C=(6, 4) from the line that passes through A=(1, 1) and B=(4, 9)?

Notation used here: Capital letters for points, and lower letters for vectors.

Reflections

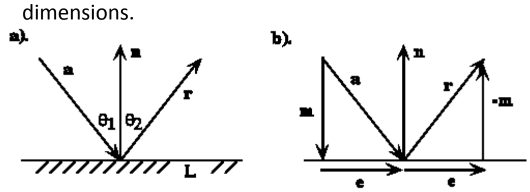
- When a billiard ball hits the wall edge of a table.
- A reflection occurs when light hits a shiny surface (below)



Reflections

- When light reflects from a mirror, the angle of reflection must equal the angle of incidence:

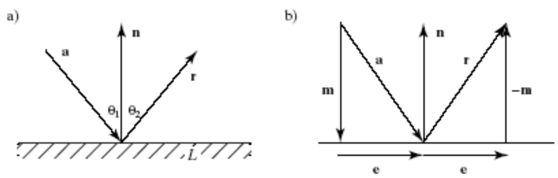
  $\theta_1 = \theta_2$.
- Vectors and projections allow us to compute the new direction r, in either two-dimensions or three dimensions.



Reflection – dot product

- The illustration shows
  - $\mathbf{e} = \mathbf{a} - \mathbf{m}$
  - $\mathbf{r} = \mathbf{e} - \mathbf{m} = \mathbf{a} - 2\mathbf{m}$
  - $\mathbf{m} = [(\mathbf{a}\cdot\mathbf{n})/|\mathbf{n}|^2]\mathbf{n} = (a\cdot \hat{n})\,\hat{n}$

$$r = a - 2\left(\frac{a\cdot n}{|n|^2}\right)n$$
$$= a - 2(a\cdot \hat{n})\cdot \hat{n}$$



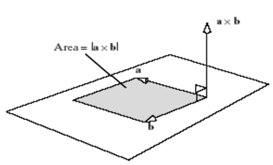
Practice Question

- Given: a=(4, -2) and surface normal n=(0, 3) what is a's reflected light about n?

Vector Cross Product (reminder)

- $\mathbf{a} \times \mathbf{b} = (a_y b_z - a_z b_y)\mathbf{i} + (a_z b_x - a_x b_z)\mathbf{j} + (a_x b_y - a_y b_x)\mathbf{k}$.
- 3D Vectors Only
- The determinant below also gives the result:

$$\mathbf{a}\times\mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

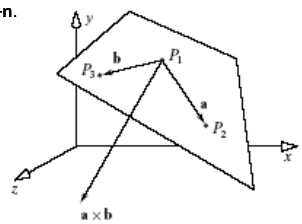


Properties (2)

- **a** · (**a** x **b**) = 0
- **|a x b|** = |**a**||**b**| sin θ, where θ is the smaller angle between **a** and **b**.
- **|a x b|** is also the area of the parallelogram formed by **a** and **b**.
- **|a x b|** = 0 if **a** and **b** point in the same or opposite directions, or if one or both has length 0.

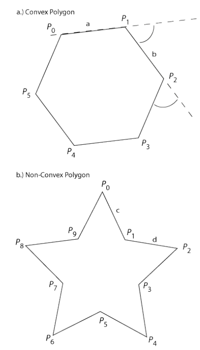
Application: Finding the Normal to a Plane

- Given any 3 non-collinear points P1, P2, and P3 in a plane, we can find a normal to the plane:
  - **a** = P2 – P1, **b** = P3 – P1, **n** = **a** x **b**. The normal on the other side of the plane is –**n**.



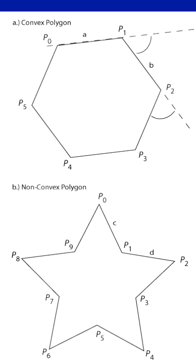
Convexity of Polygons

- Traversing around a <u>convex</u> polygon from one edge to the next
  – either a left turn or a right turn is taken
  – they all must be the same kind of turn
    - all left or all right
- An edge vector points along the edge of the polygon in the direction of travel.



Convexity of Polygons

- Take the cross product of each edge vector with the next forward edge vector.
- If all the cross products point into (or all point out of) the plane, the polygon is convex; otherwise it is not.



Representations of Key Geometric Objects

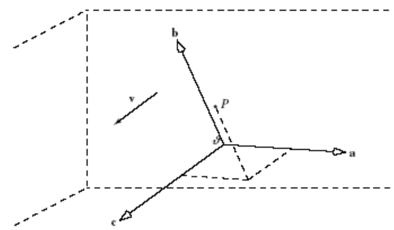
- Lines and planes are essential to graphics, and we must learn how to represent them
  – i.e., how to find an equation or function that distinguishes points on the line or plane from points off the line or plane.
- It turns out that this representation is easiest if we represent vectors and points using 4 coordinates rather than 3.

Coordinate Systems and Frames

- A vector or point has coordinates in an underlying coordinate system.
- In graphics, we may have multiple coordinate systems
  - with origins located anywhere in space.
- We define a coordinate frame as a single point (the origin, O) with 3 mutually perpendicular unit vectors: **a**, **b**, and **c**.

Coordinate Frames

- A vector **v** is represented by (v_1, v_2, v_3) such that $\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c}$.
- A point $P - O = p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$.



Homogeneous Coordinates

- It is useful to represent both points and vectors by the same set of underlying objects, (**a**, **b**, **c**, O).
- A vector has no position, so we represent it as $(\mathbf{a}, \mathbf{b}, \mathbf{c}, O)(v_1, v_2, v_3, 0)^T$.
- A point has an origin (O), so we represent it by $(\mathbf{a}, \mathbf{b}, \mathbf{c}, O)(v_1, v_2, v_3, 1)^T$.

Changing to and from Homogeneous Coordinates

- To: if the object is a vector, add a 0 as the 4th coordinate;
  - if it is a point, add a 1.
- From: simply remove the 4th coordinate.
- OpenGL uses 4D homogeneous coordinates for all its vertices.
  - If you send it a 3-tuple in the form (x, y, z), it converts it immediately to $(x, y, z, 1)$.
  - If you send it a 2D point (x, y), it first appends a 0 for the z-component and then a 1, to form $(x, y, 0, 1)$.
- All computations are done within OpenGL in 4D homogeneous coordinates.

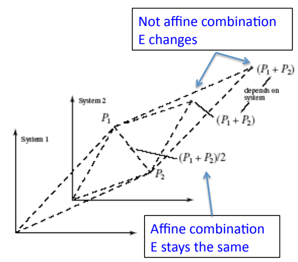
Combinations

- Why? Easy math
- Linear combinations of vectors and points:
  - The difference of 2 points is a vector: the fourth component is 1 – 1 = 0
  - The sum of a point and a vector is a point: the fourth component is 1 + 0 = 1
  - The sum of 2 vectors is a vector: 0 + 0 = 0
  - A vector multiplied by a scalar is still a vector: a x 0 = 0.
  - Linear combinations of vectors are vectors.

Middle Tennessee State University

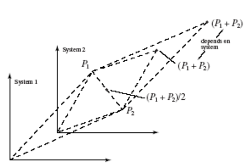
Combinations (2)

- The sum of 2 points: $E=a_1 \cdot P_1 + a_2 \cdot P_2$ is a point only if the points are part of an affine combination, so that $a_1 + a_2 = 1$. The sum is a vector only if $a_1 + a_2 = 0$.
- We require this rule to remedy the problem shown at right:



Combinations (3)

- If we form *any* linear combination of two points, say $E = fP + gR$, when $f + g$ is different from 1, a problem arises if we translate the origin of the coordinate system.
- Suppose the origin is translated by vector **u**, so that P is altered to P + **u** and R is translated to R + **u**.
- If E is a point, it must be translated to $E' = E$ + **u.**
- But we have $E' = fP + gR + (f + g)$**u,** which is *not* E + **u** unless $f + g = 1$.



Point + Vector

- Suppose we add a point A and a vector that has been scaled by a factor t:
  - The result is a point, P = A + t**v**.
- Now suppose **v** = B – A, the difference of 2 points, then: P = tB + (1-t)A,
  - P is an affine combination of two points, A and B
  - P is always on the line connecting A and B
  - The position of P on line AB is proportional to t

Middle Tennessee State University

Linear Interpolation of 2 Points

- P = (1-t)A + tB is a linear interpolation (lerp or tween) of 2 points. This is very useful in graphics in many applications,
  - P_x (t) provides an x value that is fraction t of the way between A_x and B_x. (Likewise P_y, P_z).

```
float Tween (float A, float B, float t)
{
   return    A + (B – A) * t;  // return float
}
```

Middle Tennessee State University

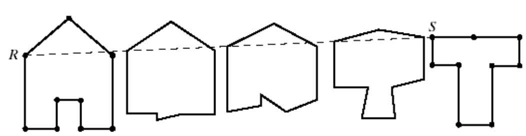
Tweening and Animation

- Tweening takes 2 polylines and interpolates between them (using lerp) to make one turn into another (or vice versa).
- We are finding here the point P(t) that is a fraction t of the way along the straight line (not to be drawn) from point A to point B.
- To start, it is easiest if you use 2 polylines with the same number of lines.

Middle Tennessee State University

Tweening

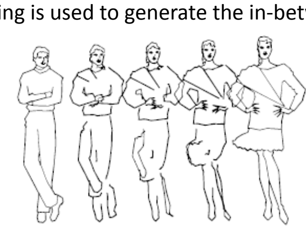
- We use polylines A and B, each with n points numbered 0, 1, …, n-1.
- We form the points P_i (t) = (1-t)A_i + tB_i, for t = 0.0, 0.1, …, 1.0 (or any other set of t in [0, 1]), and draw the polyline for P_i.



Middle Tennessee State University

Use of Tweening in animation

- In films, artists draw only the key frames of an animation sequence (usually the first and last).
  - Tweening is used to generate the in-between frames.



  - Tweening demo

Middle Tennessee State University

Practice Questions

- What is the effect of tweening when all of the points A_i in polyline A are the same? How is polyline B distorted in its appearance in each tween?
- Polyline A is a square with vertices (1, 1), (-1, 1), (-1, -1), (1, -1) and polyline B is a wedge with vertices (4, 3), (5, -2), (4, 0), (3, -2). Sketch the shape P(t) for t=-1, -0.5, 0.5, and 1.5.

Other uses of Tweening

- We want a smooth curve that passes through or near 3 points (A, B, and C). We expand $((1-t) + t)^2$ and write: $P(t) = (1-t)^2A + 2t(1-t)B + t^2C$
  - This is called the Bezier curve for points A, B, and C.
  - It can be extended to 4 points by expanding $((1-t) + t)^3$ and using each term as the coefficient of a point.