Computer Graphics

CGT 520

Computer Graphics Technology Dept.
Purdue University

February 3, 2015


Ceci n'est pas une pipe.


Ceci n'est pas une pipe


Ceci n'est pas une pipe.

Abstract 2D types and operations

- **Scalar** - a one-component value
  - scalar + scalar = scalar
  - scalar * scalar = scalar
  - We will consider the real numbers.
- **Vector** - a quantity with magnitude and direction
  - vector + vector = vector
  - scalar * vector = vector
  - Represented as a 2-element array.
- **Point** - a position in 2D (for now) space
  - point + vector = point
  - point - point = vector
  - Also represented as a 2-element array.

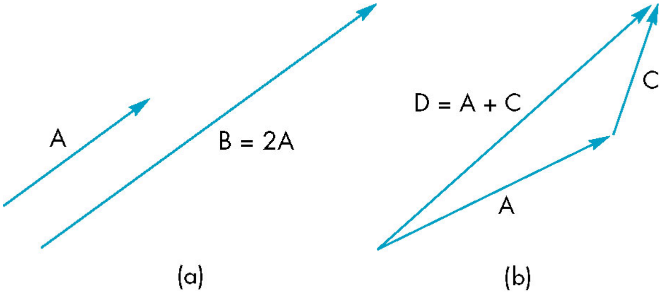
Concrete 2D types

- Scalar : We will consider the real numbers.
- Vector : Represented as a 2-element ordered set of real numbers.
- Point : Also represented as a 2-element ordered set of real numbers.

Specifying **vector** components $\mathbf{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$ implies a choice of basis vectors $\mathbf{v_1}, \mathbf{v_2}$ such that $\mathbf{a} = \alpha_1 \mathbf{v_1} + \alpha_2 \mathbf{v_2}$.

For convenience, we most often choose basis vectors $\mathbf{e_1}, \mathbf{e_2}$ with representations $\mathbf{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\mathbf{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Vector operations



Scalar-vector multiplication : $\alpha\mathbf{v} = \begin{bmatrix} \alpha v_1 \\ \alpha v_2 \end{bmatrix}$

Vector-vector addition : $\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \end{bmatrix}$

Concrete 2D types

Points:

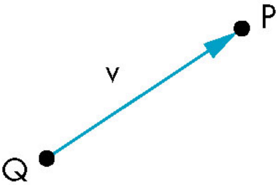
Specifying point coordinates $\mathbf{p} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$ implies a choice of basis vectors $\mathbf{v_1}, \mathbf{v_2}$ and a reference point, $\mathbf{p_0}$, such that $\mathbf{p} = \mathbf{p_0} + \beta_1 \mathbf{v_1} + \beta_2 \mathbf{v_2}$.

For convenience, we most often choose the reference point to have the representation $\mathbf{p_0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Frame:

a reference point,$\mathbf{p_0}$, and basis vectors $\{\mathbf{v_1}, \mathbf{v_2}\}$

Point operations



Point-vector addition: $\mathbf{Q} + \mathbf{v} = \mathbf{P}$
Point-point subtraction: $\mathbf{P} - \mathbf{Q} = \mathbf{v}$

Point operations

Although it is mathematically incorrect to ”scale a point” ($\mathbf{p}$), we can scale the vector $\mathbf{p} - \mathbf{p_0}$.
(Recall the point is defined as $\mathbf{p} = \mathbf{p_0} + \beta_1 \mathbf{v_1} + \beta_2 \mathbf{v_2}$)

So scaling $\mathbf{p} - \mathbf{p_0}$ by α:
$\alpha(\mathbf{p} - \mathbf{p_0}) = \alpha(\beta_1 \mathbf{v_1} + \beta_2 \mathbf{v_2})$

Adding the result to $\mathbf{p_0}$ yields a new point $\mathbf{p}'$ in the same frame as $\mathbf{p}$:
$\mathbf{p}' = \mathbf{p_0} + \alpha(\beta_1 \mathbf{v_1} + \beta_2 \mathbf{v_2})$ which has the representation $\begin{bmatrix} \alpha\beta_1 \\ \alpha\beta_2 \end{bmatrix}$.

- So, in the end, the result is the same as if we had considered $\mathbf{p}$ to be a vector.
- It is still important to recognize the difference between points and vectors.
- Later we will use a representation (homogeneous coordinates) that will allow us to distinguish between points and vectors.

Spaces

Definitions:

Space : A **set** with some **operations** and **properties**.

- Vector space:
  - set of vectors
  - with addition and scalar multiplication operations
  - Properties : Addition is commutative and associative, etc...
- Affine space:
  - set of vectors and points
  - with vector space operators, and the point-vector operations.
  - Properties : Addition is associative, etc...
- Euclidean space:
  - An affine space with the addition of an inner product operation
  - This allows us to measure distance between points and do geometry.

Affine space : Lines

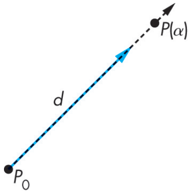
Using the elements and operations of affine space, it is possible to define a line:

$$P(\alpha) = P_0 + \alpha\mathbf{d}$$

These are all valid operation in affine space:

- $\alpha\mathbf{d}$: scalar $\times$ vector $\rightarrow$ vector
- $P_0 + (\alpha\mathbf{d})$: point $+$ vector $\rightarrow$ point
- This is called the **parametric form** of a line.
- Different values of the parameter (α) give different point on the line.

Affine space : Lines



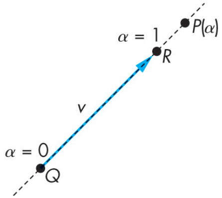
$$P(\alpha) = P_0 + \alpha \mathbf{d}$$

- Line (no endpoints): $\alpha \in (-\infty, \infty)$
- Ray (one endpoint): $\alpha \in [0, \infty)$
- Line segment (two endpoints) : $\alpha \in [\alpha_0, \alpha_1]$

Affine space : Lines

We may also define a line in terms of 2 points on the line

$$P(\alpha) = Q + \alpha(R - Q)$$



When computing points on the line, the equation can be rewritten as

$$P(\alpha) = \alpha R + (1 - \alpha)Q$$

Note that the operations involved are no longer defined in affine space. This is a useful operation to have, so we define it as a valid operation called **affine addition**.

Affine space : Affine addition

Affine addition can be defined for more than 2 points:

$$P = \alpha_1 P_1 + \alpha_2 P_2 + \ldots \alpha_n P_n$$

under the condition that

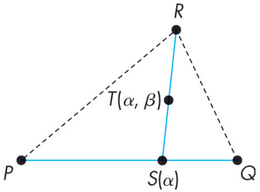
$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1$$

Affine space : Planes

Recall from geometry that 3 **noncollinear** points define a plane. In affine space, a plane can be defined as a direct extension of the parametric line.

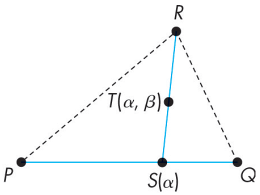
Let P, Q, R be 3 noncollinear points. Consider the line $S(\alpha)$ joining point P and Q.

$$S(\alpha) = \alpha P + (1 - \alpha)Q$$



Now construct a line, T, between any point S on that line and the point R.

Affine space : Planes



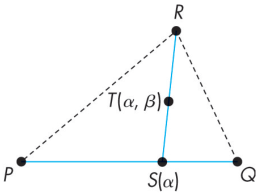
The line T can be written as : $T(\beta) = \beta S + (1 - \beta)R$
where : $S(\alpha) = \alpha P + (1 - \alpha)Q$.

Substituting the equation for S into the equation for T we have

$$T(\alpha, \beta) = \beta(\alpha P + (1 - \alpha)Q) + (1 - \beta)R$$

Plugging in values for α, β gives points on the plane defined by P, Q, R.

Affine space : Planes



$$T(\alpha, \beta) = \beta(\alpha P + (1 - \alpha)Q) + (1 - \beta)R$$

Can you show that this is an affine combination (i.e. the coefficients sum to one)?

Euclidean space : Dot product

Recall the dot product:

$$\mathbf{a} \cdot \mathbf{b} = \sum_i a_i b_i = ||\mathbf{a}||||\mathbf{b}|| \cos\theta$$

Euclidean space adds the notions of length and distance to the affine space. We will define the **length** of a vector, u, to be

$$||u|| = \sqrt{u \cdot u}$$

and the **distance** between points P, Q as $||P - Q||$.

We may **normalize** a vector, **b** to obtain a new vector **a** whose length is 1.

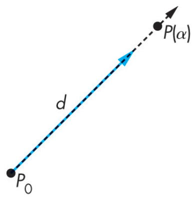
$$\mathbf{a} = \frac{\mathbf{b}}{||\mathbf{b}||}, ||\mathbf{a}|| = 1$$

Euclidean space : Dot product

The dot product can be used to compute the angle between 2 vectors, $\mathbf{u}$, $\mathbf{v}$

$$\cos\theta = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||\ ||\mathbf{v}||}$$

We can use the concepts of Euclidean space to obtain different forms of the line and plane definitions.

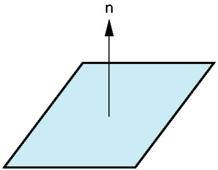


Note that for all points, $P(\alpha)$, on the line defined by P_0 and $\mathbf{d}$ we have

$$(P - P_0) \cdot \mathbf{d}_{\perp} = 0$$

Euclidean space : Dot product

We can specify a plane by specifying one point P_0 on the plane, and a vector, **n**, perpendicular to the plane.



All points, P, on the plane defined by P_0 and **n** we have

$$(P - P_0) \cdot \mathbf{n} = 0$$

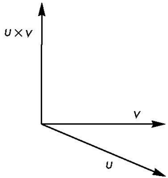
These representations of points and lines are called **implicit**.
They don't allow us to directly generate points on the line/plane, they do allow us to test whether a given point is on the line/plane.

Euclidean space : Cross product

For vectors $\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$ the cross product is defined by

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

Euclidean space : Cross product



If **u** and **v** are not parallel, then the vector $\mathbf{u} \times \mathbf{v}$ is **orthogonal** to **u** and **v**.

The length of the resulting vector is related to the angle between **u** and **v** by

$$|\sin\theta| = \frac{||\mathbf{u} \times \mathbf{v}||}{||\mathbf{u}||\ ||\mathbf{v}||}$$

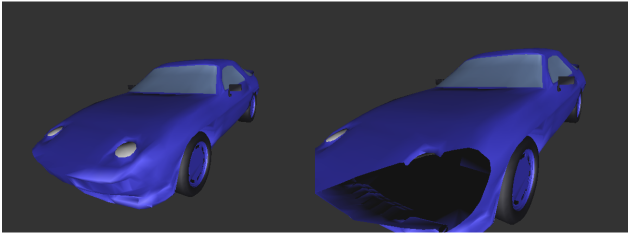
Three-dimensional primitives

We have seen how to represent simple 3D objects : lines and planes. One challenge of computer graphics is representing complex real objects. We will use the following approach:

- Represent objects as **surfaces**, not solids.
- Store the coordinates of points on the surface.
- Approximate the surface using planar polygons.

Represent objects as **surfaces**, not solids.

If the camera does not pass through objects, and if objects never break apart, we will only ever see the surface.

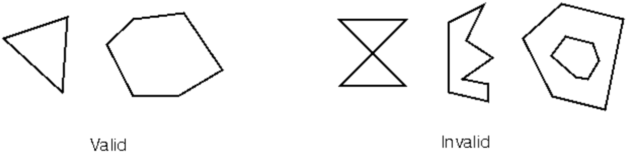


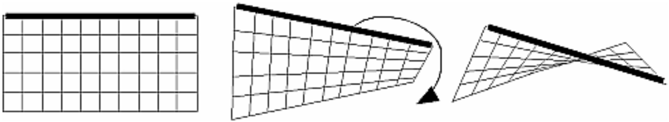
This is only a visual representation, the application programmer must devise techniques to prevent objects from intersecting.

Three-dimensional primitives

Why triangles?

- Always planar.
- Always **simple** (edges do not intersect).
- Always **convex** (line between any 2 points is in triangle).
- It is easy to optimize the hardware for triangles.





Nonplanar skew quadrilateral.