Math Foundations of CG

Outline

- Abstract spaces: objects and operations
  - Field of real numbers **R**
  - Vector space over **R**
  - Euclidean spaces
  - Affine spaces
    - Affine combinations
    - Convex combinations
    - Frames
    - Affine maps
    - Euclidean spaces
- Read: angel, Appendices B and C, Ch 4.1 43

Geometric ADTs

- Scalars, Points and Vectors are members of mathematical abstract sets
- Abstract spaces for representing and manipulating these sets of objects
- Field-- scalars
- Linear Vector Space – vectors
- Euclidean Space – add concept of distance
- Affine Space – adds the point

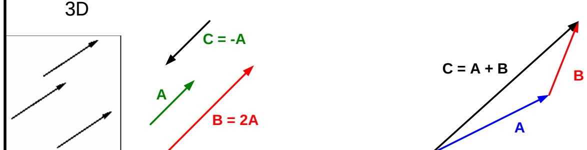
Linear Vector Spaces (defined over scalars)

- *S* is a set of scalars (like the real numbers)
- The set *V* of objects called ***vectors*** , $\{u, v, w, \ldots\}$
  is a ***(linear) vector space*** defined over *S* if there are two operations
  - Vector-vector addition, $u+v, \quad + : V \times V \to V$
  - Scalar-vector multiplication, $\alpha\, u, \quad f : S \times V \to V$

  satisfying the following
  - Axioms
    - Unique additive unit, *the zero vector,* $\mathbf{0}$
      $u + \mathbf{0} = \mathbf{0} + u = u$
    - Every vector u has additive inverse $-u$
      $u + (-u) = (-u) + u = \mathbf{0}$

Vector Spaces (cont.)

- Axioms (cont.)
  - Vector-vector addition is commutative and associative
  - Scalar-vector multiplication is distributive
    $\alpha\,(u+v) = \alpha\, u + \alpha\, v$
    $(\alpha + \beta) u = \alpha\, u + \beta\, u$
- Examples
  - Geometric vectors over **R** , i.e., directed line segments in 3D



Vector Spaces (cont.)

- Examples
  - n-tuples of real numbers (we will use triples usually)
    - A vector is identified with an n-tuple

$$\vec{v} = (v_1, v_2, \ldots, v_n)$$
$$\vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n)$$
$$\alpha\, \vec{u} = (\alpha u_1, \alpha u_2, \ldots, \alpha u_n)$$

Vector Spaces (cont.)

- V is a linear vector space over a field S
  - $u_1,\ldots,u_k, u \in V$,
    u is a *linear combination* of $u_1,\ldots,u_k$, if
    $\exists \alpha_1,\alpha_2,\ldots,\alpha_k \in S, s.t.$
    $u = \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_k u_k$
  - The non-zero vectors $u_1,\ldots,u_k$ are *linearly independent*, if

$\forall \alpha_1,\alpha_2,\ldots,\alpha_k \in S, s.t.$

$$\alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_k u_k = \mathbf{0} \quad \Rightarrow \quad \alpha_1 = \alpha_2 = \ldots = \alpha_k = 0$$

Vector Spaces (cont.)

- V is a linear vector space over a field S
  - The vectors $u_1,\ldots,u_k$ are *linearly dependent*, if one of them can be expressed as a non-trivial linear combination of the rest. (non-trivial means that not all coefficients are 0)
  - Any set of vectors that includes the zero vector is linearly dependent.
  - *Basis* : a maximal linear independent set of vectors, i.e., if one more vector is added to the set it becomes linearly dependent.
  - *Dimension*: number of vectors in the basis

Vector Spaces (cont.)

- V is a n dimensional vector space over a field S, and $B = \{u_1,\ldots,u_n\}$ is a basis:
  - Every vector u is represented uniquely as a linear combination of the basis, i.e., there exist unique scalars
    $\alpha_1,\alpha_2,\ldots,\alpha_n \in S, s.t.$
    $u = \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_n u_n$
  - $\{\alpha_i\}_{i=1}^{n}$ representation (coordinates) of u in the basis B

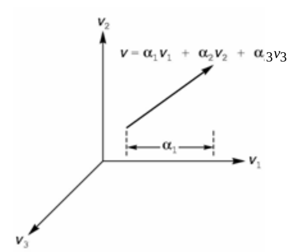
We are concerned with 3D vector space

Represent w as linear combination of three linearly independent vectors, v_1, v_2, v_3

$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$

components basis

$$\mathbf{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \qquad \mathbf{w} = \mathbf{a}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$



Vector Spaces: Changes of Basis

- How do we represent a vector if we change the basis?
- Suppose the {v1,v2,v3} and {u1,u2,u3} are two bases.
- Basis vector in second set can be represented in terms of the first basis
- Given the representation of a vector in one basis, we can change to a representation of the same vector in the other basis by a linear transformation (i.e., matrix multiplication)

Vector Spaces: Change of Basis

- Two basises: **u** and **v**

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}$$

$$\mathbf{u}_1 = \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3$$
$$\mathbf{u}_2 = \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3$$
$$\mathbf{u}_3 = \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3$$

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix} \qquad \mathbf{u} = \mathbf{M}\mathbf{v} \qquad \mathbf{v} = \mathbf{M}^{-1}\mathbf{u}$$

Change of basis is a linear operation.

Vector Spaces: Change of Basis

Notation: for a matrix, a'' denotes the transpose.

Let in basis **v** the vector **w** is represented by a component column matrix **a**, and in **u** , by a component matrix **b**

$$\mathbf{a}=\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix}\quad \mathbf{v}=\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \mathbf{v}_3\end{bmatrix}\qquad \mathbf{b}=\begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix}\quad \mathbf{u}=\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\\ \mathbf{u}_3\end{bmatrix}\qquad \begin{matrix}\mathbf{w}=\mathbf{a}'\mathbf{v}\\ \mathbf{w}=\mathbf{b}'\mathbf{u}\end{matrix}$$

$$\begin{matrix}\mathbf{u}=\mathbf{M}\mathbf{v}\\ \mathbf{v}=\mathbf{M}^{-1}\mathbf{u}\end{matrix}\qquad M=\begin{bmatrix}\gamma_{11}&\gamma_{12}&\gamma_{13}\\ \gamma_{21}&\gamma_{22}&\gamma_{23}\\ \gamma_{31}&\gamma_{32}&\gamma_{33}\end{bmatrix}$$

What is the relation between the two representations **a** and **b**?

$$\mathbf{b}=\mathbf{M}'^{-1}\mathbf{a}$$

Vector Spaces: Change of Basis Example

Given a basis **v**, we want to change to a new basis **u**

$$\underset{old}{\mathbf{v}}=\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \mathbf{v}_3\end{bmatrix}\quad \underset{new}{\mathbf{u}}=\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\\ \mathbf{u}_3\end{bmatrix}\quad \begin{matrix}\mathbf{u}_1=\mathbf{v}_1\\ \mathbf{u}_2=\mathbf{v}_1+\mathbf{v}_2\\ \mathbf{u}_3=\mathbf{v}_1+\mathbf{v}_2+\mathbf{v}_3\end{matrix}\quad M=\begin{bmatrix}1&0&0\\ 1&1&0\\ 1&1&1\end{bmatrix}\quad \begin{matrix}\underset{new}{\mathbf{u}}=\mathbf{M}\underset{old}{\mathbf{v}}\\ \mathbf{v}=\mathbf{M}^{-1}\mathbf{u}\end{matrix}$$

Let **w** has representation **a** in the old basis, **v,**

$$\underset{in\ old}{\mathbf{a}}=\begin{bmatrix}1\\ 2\\ 3\end{bmatrix}$$

Then the representation **b** of **w** in the new basis **u** is $\mathbf{b}=\mathbf{M}'^{-1}\mathbf{a}$

$$\underset{in\ new}{\mathbf{b}}=\left(\begin{bmatrix}1&0&0\\ 1&1&0\\ 1&1&1\end{bmatrix}'\right)^{-1}\begin{bmatrix}1\\ 2\\ 3\end{bmatrix}=\begin{bmatrix}1&-1&0\\ 0&1&-1\\ 0&0&1\end{bmatrix}\begin{bmatrix}1\\ 2\\ 3\end{bmatrix}=\begin{bmatrix}-1\\ -1\\ 3\end{bmatrix}$$

Vector Spaces

- It is common in CG to use vectors to represent
  - Locations (points)
  - Displacements
  - Directions (orientation)
- Keep in mind that points and vectors are different, I.e. have different behavior (methods)
  - Displacement of a point is another point (new location)
  - Displacement of a vector it is the same vector (vectors do not have fixed locations)
- Thus it is not theoretically correct to use vectors to represent both points and displacements although in 3D we do.
- If you use OOP: ADT vector implementation, class vector , another class for point
  - Internally work with 3,4-tuples of real numbers
  - Use matrix algebra in the implementation of methods

Euclidean Space

- We add the notion of a <u>distance and angle</u> to a vector space by means of inner (dot) product.
- E is an Euclidean space, if it is vector space with dot (scalar,inner) product , $u\cdot v,\quad \cdot: E\times E\rightarrow \mathbf{R}$

  i.e., for vectors u and $u\cdot v$ is a real number, such that
  - Axioms

    $u\cdot v=v\cdot u$

    $(\alpha u+\beta v)\cdot w=\alpha(u\cdot w)+\beta(v\cdot w)$

    $u\cdot u>0,\quad u\neq \mathbf{0}$

    $\mathbf{0}\cdot\mathbf{0}=0$

Euclidean Space (cont.)

- The length of a vector

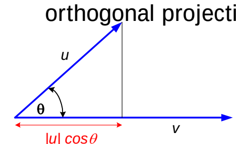
  $|u|=\sqrt{u\cdot u}$
- The norm of a vector

  $\|u\|=|u|^2=u\cdot u$
- Two non zero vectors u and v are orthogonal if $u\cdot v=0$
- The angle between two vectors is given by

  $\cos\theta=\frac{u\cdot v}{|u||v|}$
- Unit vector: a vector of length 1
- Normalizing a vector: $\frac{u}{|u|}$

Euclidean Space (cont.)

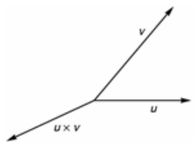
- Orthonormal basis: a basis consisting of unit vectors which are mutually orthogonal
- Projections: $|u|\cos\theta=u\cdot v/|v|$ is the length of orthogonal projection of u onto v



- If v is unit vector, the length of the projection of u on v is $u.v$

3D Euclidean Space

- Cross Product of two vectors u and v is a vector $n = u \times v$, $\times : V \times V \to V$
  - n is orthogonal to v and u,
  - the triple (u,v,n) is right-handed,
  - The length $|u \times v| = |u||v|\sin(\theta)$



Euclidean Spaces (cont)

- Example: $\mathbf{R}^3$ $\quad \mathbf{a},\mathbf{b} \in \mathbf{R}^3$

  $\{i, j, k\}$ orthonormal basis, and $\mathbf{a} = \begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix}$ $\quad \mathbf{b} = \begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix}$
- Dot product:

$$\mathbf{a}\cdot\mathbf{b} = \mathbf{a}'\mathbf{b} = \begin{bmatrix}\alpha_1 & \alpha_2 & \alpha_3\end{bmatrix}\begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix} = \alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_3\beta_3$$

- Cross product:

$$\mathbf{a}\times\mathbf{b} = \begin{bmatrix}\alpha_2\beta_3 - \alpha_3\beta_2\\ \alpha_3\beta_1 - \alpha_1\beta_3\\ \alpha_1\beta_2 - \alpha_2\beta_1\end{bmatrix} = \begin{vmatrix} i & j & k\\ \alpha_1 & \alpha_2 & \alpha_3\\ \beta_1 & \beta_2 & \beta_3\end{vmatrix}$$

Euclidean Space (cont.)

- We can construct orthonormal basis in 3D by using the dot and cross products
  - Given vector *u*,
    - Set $e_1 = \dfrac{u}{|u|}$
    - Calculate e_2 s.t. $e_1 \cdot e_2 = 0, |e_2| = 1$
    - Calculate $e_3 = e_1 \times e_2$
    - The basis (e_1, e_2, e_3) is orthonormal

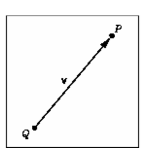
Affine Spaces

- Given a vector space *A*, an **affine space** *A* over the vector space has two types of objects:
  - **points**, *P,Q,...*
  - and **vectors**, *u,v,...*

  and is defined by the following axioms
  - All axioms of the vector space
  - Operations relating points and vectors
    - Point-point subtraction gives unique vector, $P - Q = v$
    - Point-vector addition gives unique point, $Q + v = P$

Affine Spaces (cont)

- Operations relating points and vectors
  - Subtraction of two points yields a vector: $v = P - Q$
  - Point-vector addition yields a point: $P = Q + v$
  - *All* the operations:
    - point-point subtraction,
    - point-vector addition,
    - vector-vector addition,
    - scalar-vector multiplication



Affine Spaces (cont)

- Axioms:
  1. Two points define unique vector, $P - Q = v$
  2. Point and vector define unique point, $Q + v = P$
  3. $Q - P = -(P - Q)$
  4. *head-to-tail axiom:* given points *P,R,* for any other point *Q,*

     $$P - R = (Q - R) + (P - Q)$$
  5. If *O* is an arbitrary point,

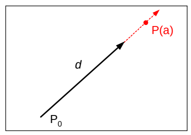
     $$\forall u \in A, \quad \exists!\ P \in A: \quad P - O = u$$

Line: parametric equation

- A line, defied by a point P_0 and a vector d consists of all points P obtained by

  $P(\alpha) = P_0 + \alpha d$

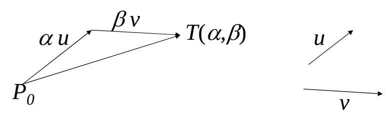
  where α varies over all scalars.
- $P(\alpha)$ is a point for any value of α
- For non negative values, we get a ray emanating from P_0 in the direction of d



Plane: parametric equation

- A plane defined by a point P_0 and two non collinear vectors (non parallel, i.e.,linearly independent) u and v, consists of all points $T(\alpha,\beta)$:

  $T(\alpha,\beta) = P_0 + \alpha u + \beta v$



Affine Spaces (cont)

- All the operations:
  - point-point subtraction,
  - point-vector addition,
  - vector-vector addition,
  - scalar-vector multiplication
- Point-point addition is not defined, but addition-like combinations of points are well-defined.

Affine Combinations of Two Points

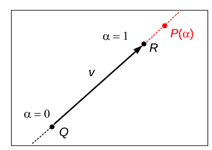
- Given two points Q and R, and two scalars α_1, α_2 where $\alpha_1 + \alpha_2 = 1$ the affine combination of Q and R with coefficients α_1, α_2 is a point P denoted by

$$P = \alpha_1 Q + \alpha_2 R$$

and defined as follows

$$\alpha_1 + \alpha_2 = 1 \Rightarrow \alpha_1 = 1 - \alpha_2$$

$$P = \alpha_1 Q + \alpha_2 R = Q + \alpha_2 (R - Q)$$



- *All affine combinations of two points generate the line through that points.*

Affine Combinations of Three Points

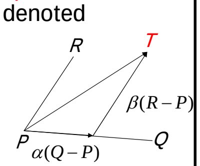
- Given three points P, Q, and R, and three scalars $\alpha_1, \alpha_2, \alpha_3$ where $\alpha_1 + \alpha_2 + \alpha_3 = 1$ the affine combination of the three points with coefficients $\alpha_1, \alpha_2, \alpha_3$ is a point T, denoted

$$T = \alpha_1 P + \alpha_2 Q + \alpha_3 R$$

- The point T is defined by

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

$$\alpha = \alpha_2,\ \beta = \alpha_3$$



- *All affine combinations of three non collinear points generate the plane* through that points.

Affine Combinations of n Points

- Given an affine space *A*, a point *P* is an affine combination of $P_1, P_2, \ldots, P_n$, iff, there exist scalars

  $\exists\, \alpha_1, \alpha_2, \ldots, \alpha_n, \ \sum_{i=1}^{n} \alpha_i = 1$ such that

  $P = P_1 + \alpha_2(P_2 - P_1) + \cdots + \alpha_n(P_n - P_1)$

- The affine combination is denoted by

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

- If the vectors $P_i - P_1, i = 1, \ldots, n,$ are coplanar, what is the set of all affine combinations of the n points?

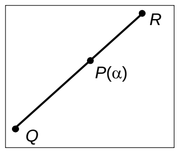
Convexity

- Convex set– a set in which a line segment connecting any two pints of the set is entirely in the set.
- For $0 \le \alpha \le 1$ the affine combinations of points Q and R is the line segment connecting Q and R

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

- This line segment is convex
- The midpoint, $\alpha = 0.5$
- Give the affine combination

representing a point dividing the line Segment in ratio m:n, starting from Q



$$M = Q + (M - Q) \qquad (1)$$

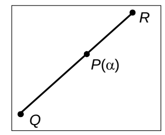
$$M - Q = \frac{m}{m+n}(R - Q) \qquad (2)$$

now substitute (2) in (1):

$$M = Q + \frac{m}{m+n}(R - Q)$$

$$M = (1 - \frac{m}{m+n})Q + \frac{m}{m+n}R$$

$$M = \frac{n}{m+n}Q + \frac{m}{m+n}R$$



Convex (affine) combinations

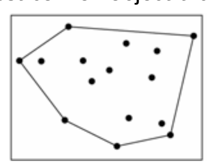
- Convex combinations: affine combinations with positive coefficients,

  $P = \alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_n P_n$
  - $\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1$
  - $\alpha_i \ge 0,\ i = 1,2,\ldots,n$

- Convex hull of a set of points is the set of all convex combination of this points.
  - In particular, for any two points of the set the line segment connecting the points is in the convex hull, thus the convex hull is a convex set.
  - In fact, the convex hull it is the smallest convex set that contains the original points.

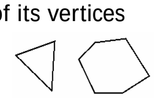
Geometric ADTs: Convexity

- The convex hull could be thought of as the set of points that we form by stretching a tight fitting surface over the given set of points – shrink wrapping the points (all points inside and on the surface)
- It is the smallest convex object that includes the set of points



Convex Polygons

- A convex polygon is completely specified by the set of its vertices



- A convex polygon: the convex hull of the vertices
- Given equilateral triangle give the representation of the center of the mass
- Generate random point inside a triangle

A normal to a plane

- Normal n to a plane : unit vector orthogonal to the plane
- If we are given the parametric equation of the plane

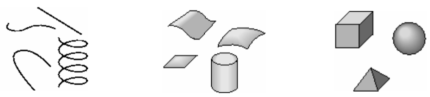
  $T(\alpha,\beta) = P_0 + \alpha u + \beta v,$

  $n = u \times v / | u \times v|$
- Given a polygon, write the outward/front normal
- Given a point P_0 and a vector n , there is unique plane that goes through P_0 and has normal n: it consists of all points P satisfying the normal equation of the plane

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

- Given a plane, defined by point P_0 and a normal n: the plane divides the space into two subspaces (one on the side pointed by the normal, (P-P0)n>0, and the other in the side pointed by –n, (P-P0)n<0.

3D Primitives



Curves　　Surfaces　　Volume Objects

3D Primitives

Objects With Good Characteristics

- Described by their surfaces; thought to be hollow
- Specified through a set of vertices in 3D
- Composed of, or approximated by, flat convex polygons
- For a polygon, when you walk along the edges in order in which the vertices are specified, the right hand rule gives to outward normal.
- Be careful about the order of the vertices when you specify polygons. (in order, counter clockwise when looking from the outside towards the object).

Viewing

- Viewing volume – the volume that is seen by the synthetic camera. Only object inside that volume could possibly be seen in the image.
  glOrtho() specifies rectangular volume aligned with the axes of the camera. The volume is enclosed by front,back, and side clipping planes.
- OpenGL uses a default viewing vollume 2x2x2 cube (otherwise, viewing volume can be set by glOrtho())
- Viewing rectangle/window – the area of the image plane that is seen.
- For gluOrtho2D(), the viewing rectangle is at z=0

Displaying 3D Objects

- Hidden surface removal
  - Painter's algorithm
  - Z-buffer algorithm
- Z buffer (depth buffer), to use in OpenGL
  - must add to display mode
  - must enable
  - must clear before drawing

Displaying 3D Objects In OpenGL

•In main():

```
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
```

•In init():

```
glEnable(GL_DEPTH_TEST);
```

•In display():

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

•Projection: only objects inside the viewing volume will be projected

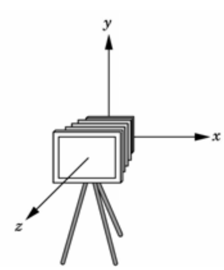
```
glOrtho(GLfloat xmin, GLfloat xmax,
        GLfloat ymin, GLfloat ymax,
        Glfloat zmin, Glfloat zmax);
```

Vertices of object are in viewing coordinates, (x,y,z), s.t.

`xmin<=x<= xmax,  ymin<=y<=ymax,  minz<=z<=zmax`

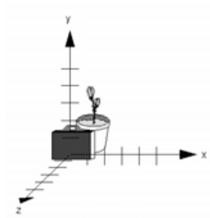
will be projected, the rest are clipped out

Initial Camera Position



- Objects are modeled independently from the location of the camera
- OpenGL places a camera at the origin of the world frame pointing in the negative z direction
- If model view matrix is an identity matrix, then the camera frame and world frame are identical

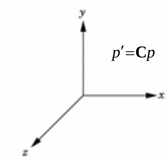
Default Position

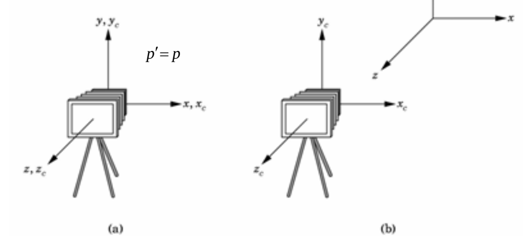


Object and Viewpoint at the Origin

Movement of the Frames

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity( );
glTranslatef( 0.0, 0.0,-d);
```





Two Points of View

- Hold camera frame fixed, move objects in front of the camera: glTranslate, glRotate
- Model objects stationary and move the camera away from the objects, gluLookAt

Affine Spaces (cont): Frames

- Frame: a basis at fixed origin
- Select a point O (origin) and a basis (coordianate vectors) $B = \{u_1, \ldots, u_n\}$
  - Any vector u can be represented as uniquely as a linear combination of the basis vectors

    $$u = \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_n u_n$$
  - Any point P can be represented uniquely as

    $$P = O + \beta_1 u_1 + \beta_2 u_2 + \ldots + \beta_n u_n$$
- Thus, we have affine coordinates for points and for vectors
- Given a frame, points and vectors can be represented uniquely by their affine coordinates

Affine Spaces: Frames (cont)

- If we change frames the coordinates change.
- The change of basis in a vector space is a linear transformation (represented as matrix multiplication)
- The change of frame in an affine space is NOT linear transformation
- We extend the affine coordinates, by adding one more dimension. The new coordinates are called homogeneous coordinates.
- The change of frame in homogeneous coordinates is a linear transformation (i.e represented as matrix multiplication)

Affine coordinates in 3D

Given a frame (P_0, v_1, v_2, v_3),
a vector w and a point P can be represented uniquely by:

$$P = P_0 + \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$
$$w = \eta_1 v_1 + \eta_2 v_2 + \eta_3 v_3$$

The affine coordinate (representations) of the vector and point are

$$P_0 \rightarrow \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \quad w \rightarrow \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{bmatrix}$$

Homogeneous Coordinates

- Use four dimensional column matrices to represent both points and vectors in homogeneous coordinates
- The first three components are the affine coordinates
- To maintain a distinction between points and vectors we use the fourth component: for a vector it is 0 and for a point it is 1

From affine to Homogeneous Coordinates

- Affine coordinate equations and representations:

$$w = \eta_1 v_1 + \eta_2 v_2 + \eta_3 v_3$$
$$P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + P_0$$
$$P \rightarrow \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \quad w \rightarrow \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{bmatrix}$$

affine-coordinate representations

- We agree that

$$1.P_0 = P_0$$
$$0.P_0 = \mathbf{0}, \text{ zero vector}$$

- The homogeneous coordinate equations and representations:

$$w = \eta_1 v_1 + \eta_2 v_2 + \eta_3 v_3 + 0.P_0$$
$$P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + 1.P_0$$
$$P \rightarrow \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix} \quad w \rightarrow \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ 0 \end{bmatrix}$$

homogeneous-coordinate representation of the point and the vector

Homogeneous Coordinates

Frame	Point	Vector
$\{P_0, v_1, v_2, v_3\}$	$p = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}$	$a = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ 0 \end{bmatrix}$

We can write the coordinate equations in matrix form. For example,

$$P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + 1.P_0 \rightarrow \quad P = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

We carry out operations on points and vectors using their homogeneous-coordinate representation and ordinary matrix algebra

Homogeneous Coordinates Change of Frame

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = M \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} \qquad M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

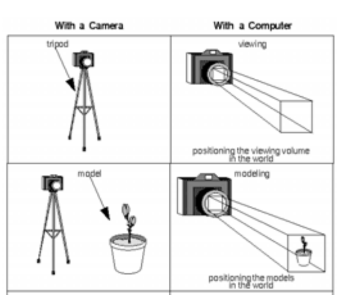
two frames

change of frames matrix representation

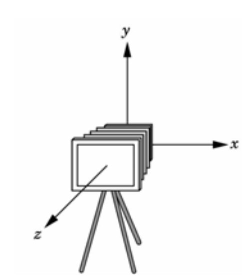
$$b = (M^T)^{-1} a$$

Change of frames is a linear transformation in homogeneous coordinates. All affine transformations can be represented as matrix multiplications in homogeneous coordinates.

Frames In OpenGL



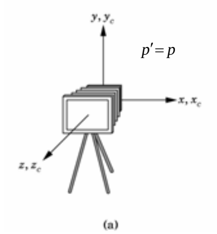
Initial Camera Position

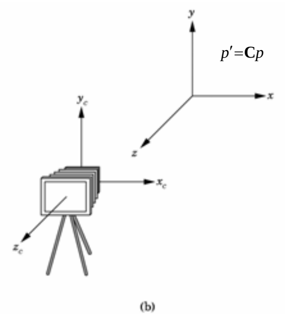


- Objects are modeled independently from the location of the camera
- OpenGL places a camera at the origin of the world frame pointing in the negative z direction
- If model view matrix is an identity matrix, then the camera frame and world frame are identical

Movement of the Frames

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity( );
glTranslatef( 0.0, 0.0,-d);
```





Frames In OpenGL

- We use two frames: the camera frame and the world frame
- We regard the camera frame as fixed
- The model- view matrix positions the world frame relative to the camera frame
- Model- view matrix that translates along z, to separate the two frames, so object could be in camera's field of view:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Frames In OpenGL