CS 428: Fall 2009

Introduction to Computer Graphics

Geometric Transformations

Topic overview

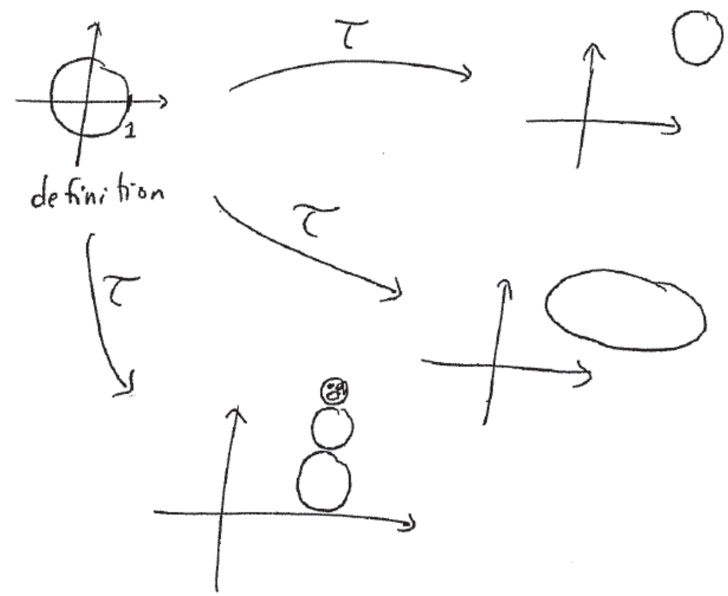
- **Image formation and OpenGL (last week)**
  - **Modeling the image formation process**
  - **OpenGL primitives, OpenGL state machine**
- Transformations and viewing
- Polygons and polygon meshes
  - Programmable pipelines
- Modeling and animation
- Rendering

Topic overview

- Image formation and OpenGL
- **Transformations and viewing (next weeks)**
  - **Linear algebra review, Homogeneous coordinates**
  - **Geometric + projective transformations**
  - **Viewing, Viewports, Clipping**
- Polygons and polygon meshes
  - Programmable pipelines
- Modeling and animation
- Rendering

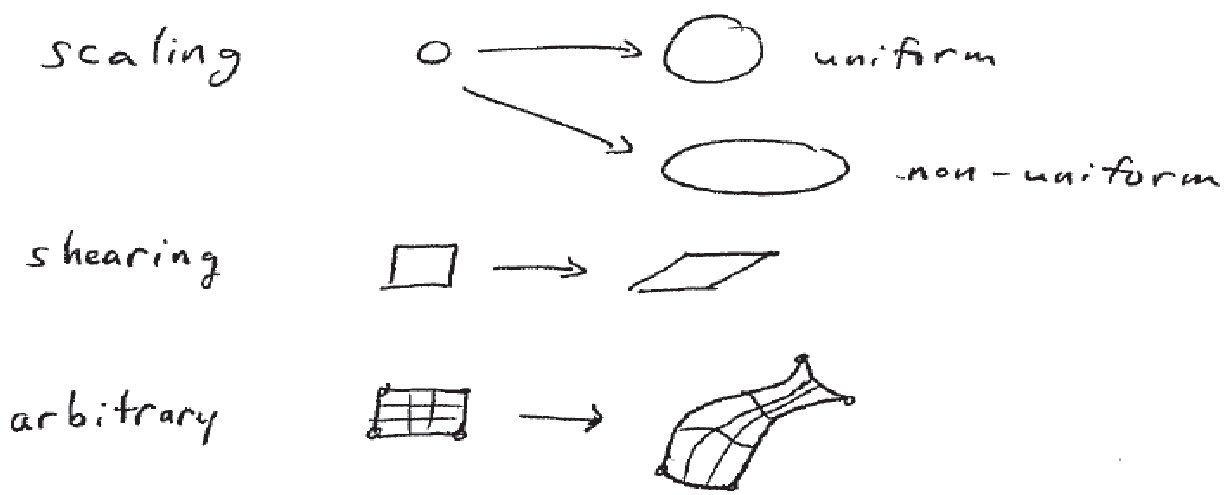
Transformations in CG

- Specify placement of objects in the **world**
  - relative to the configuration in which they are defined
- Allow for reuse of objects in different places, sizes
- Specify the camera position
- Specify the camera model (projection)



Transformations in CG

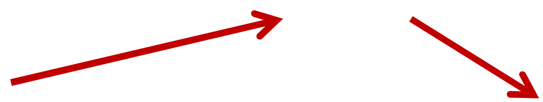
- The “where” is specified by **translations and rotations** (= rigid body motions)
- Shape changes include



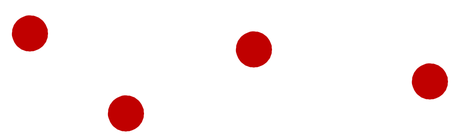
- For now we will only use linear deformations
  - Linear algebra!

Representations in CG

- Computations should not depend on coordinate system (such as midpoint/origin)
- Need careful accounting of points and vectors
  - Both $\in \mathfrak{R}^3$ (3 tuples of floating point values)
- **Vectors**



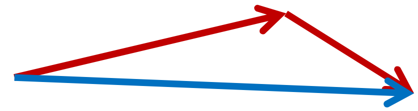
  - Displacements, velocities, directions, trajectories, surface normals, etc.
- **Points**



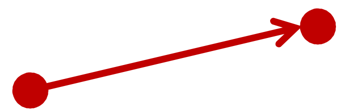
  - Locations!

Vector/point operations

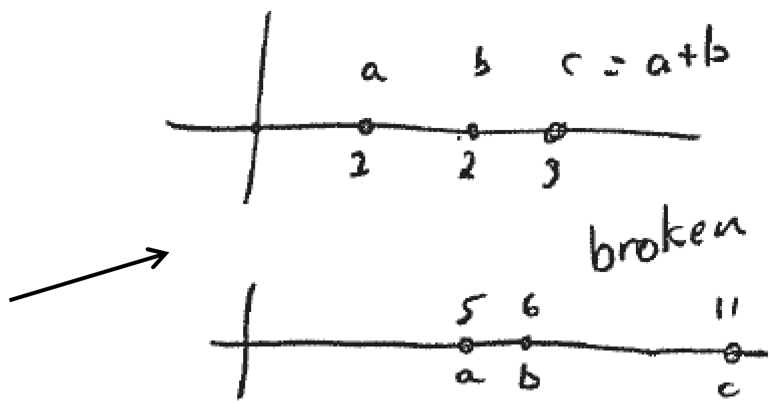
- Vector + vector = vector



- Point + vector = point

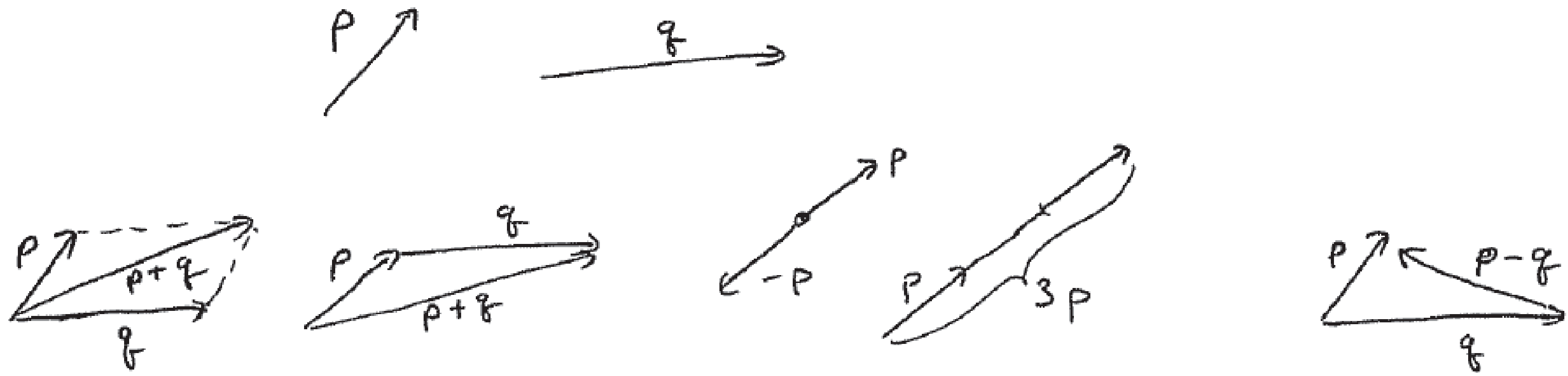


- Point + point = invalid!
  - Street address analogy



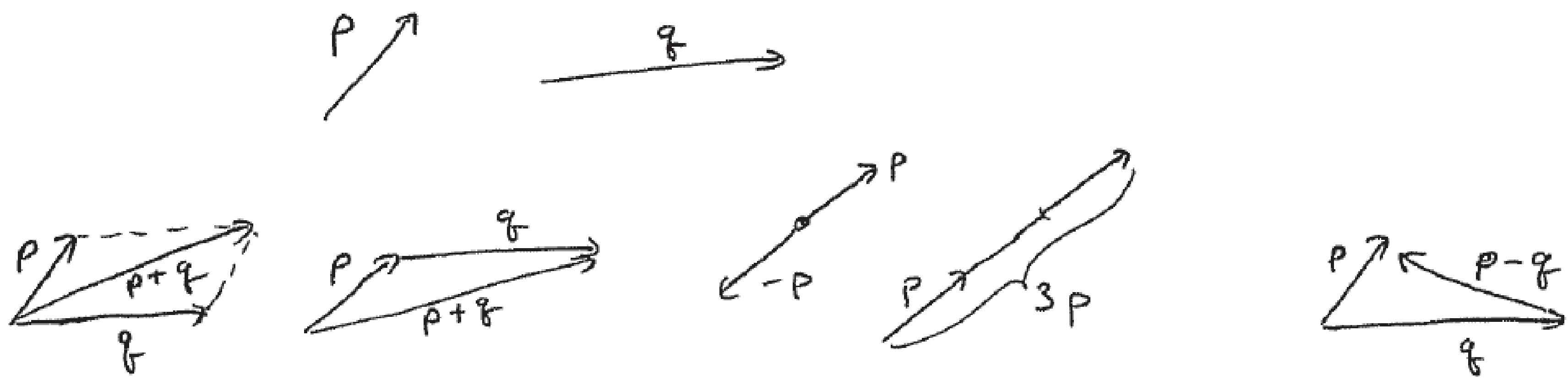
- Point – point = vector
  - Works!

Vector review

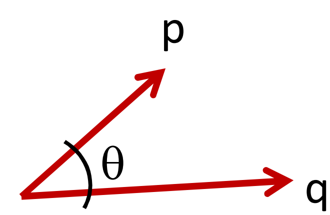


- $[p + q]_i = p_i + q_i$ addition
- $[s\,p]_i = s \cdot p_i$ scalar multiplication
- $|| \, p \, || = \mathrm{sqrt}[\, (p_i)^2 \,]$ length

Vector review



- $p \cdot q = \Sigma\, p_i \cdot q_i$ dot product $\|\mathbf{p}\| \cdot \|\mathbf{q}\| \cdot \cos\theta$



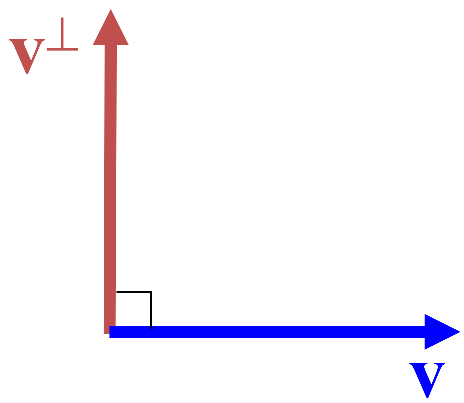
- Normalization $\hat{\mathbf{p}} = \dfrac{\mathbf{p}}{\|\mathbf{p}\|}$

Perpendicular vectors

$$< \mathbf{v}, \mathbf{w} >= 0$$

$$\mathbf{v} = (x_v, y_v) \Rightarrow \mathbf{v}^{\perp} = \pm(-y_v, x_v)$$

In 2D only!



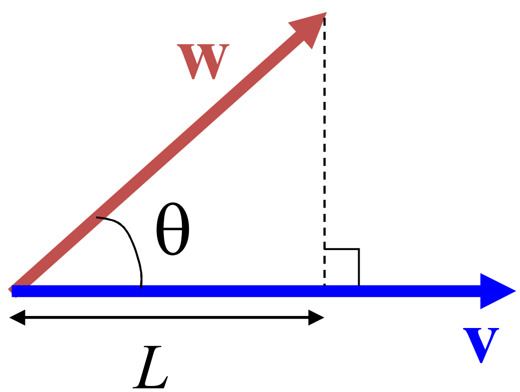
Linear combination + Basis

- Linear combination
  - $\lambda_1 \cdot v_1 + \lambda_2 \cdot v_2 + \ldots + \lambda_n \cdot v_n$ with $\lambda_i \in R$
- Linear independence of vectors $v_1, \ldots, v_n$
  - $\lambda_1 \cdot v_1 + \ldots + \lambda_n \cdot v_n = 0$ only when $\lambda_i = \ldots = \lambda_n = 0$
- **Basis** of n-dimensions is a set of n linearly independent vectors
  - Every vector in R^n has a unique set of λ's to represent it → Cartesian coordinates

Inner (dot) product

- Defined for vectors:

$$< \mathbf{v},\ \mathbf{w} > = \| \mathbf{v} \| \cdot \| \mathbf{w} \| \cdot \cos\theta$$



$$\cos\theta = \frac{L}{\|\mathbf{w}\|}$$

$$L = \frac{< \mathbf{v}, \mathbf{w} >}{\| \mathbf{v} \|}$$

Projection of **w** onto **v**

Distance between point and line

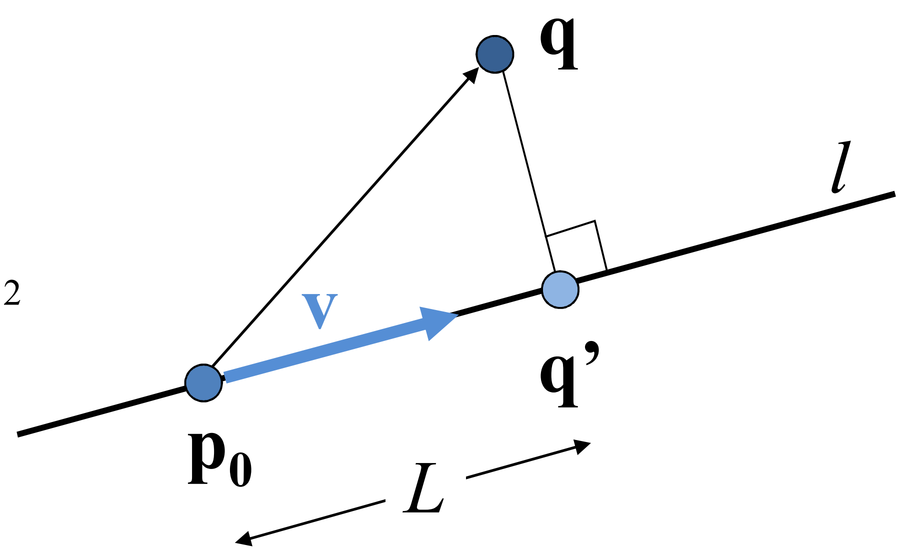
Pythagoras :

(1) $L^2 + \text{dist}(\mathbf{q}, \mathbf{q}')^2 = \| \mathbf{q} - \mathbf{p_0} \|^2$

(2) $L = \dfrac{< \mathbf{q} - \mathbf{p_0}, \mathbf{v} >}{\| \mathbf{v} \|}$

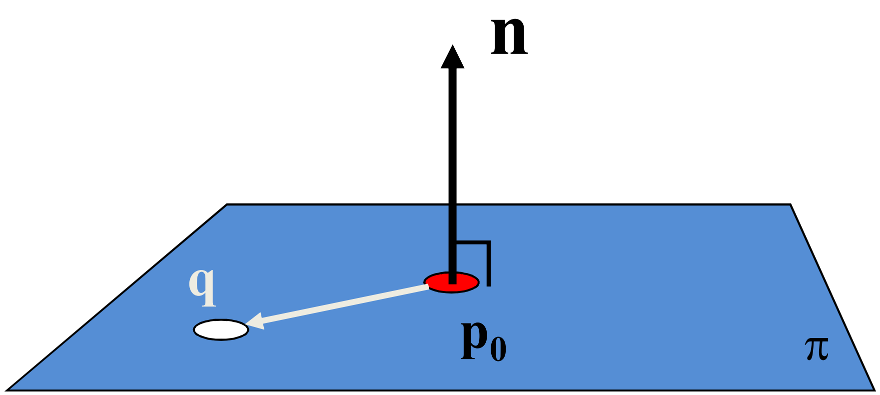
$\Rightarrow \quad \text{dist}(\mathbf{q}, \mathbf{q}')^2 = \| \mathbf{q} - \mathbf{p_0} \|^2 - L^2 =$

$= \| \mathbf{q} - \mathbf{p_0} \|^2 - \dfrac{< \mathbf{q} - \mathbf{p_0}, \mathbf{v} >^2}{\| \mathbf{v} \|^2}.$



Representation of a plane in 3D space

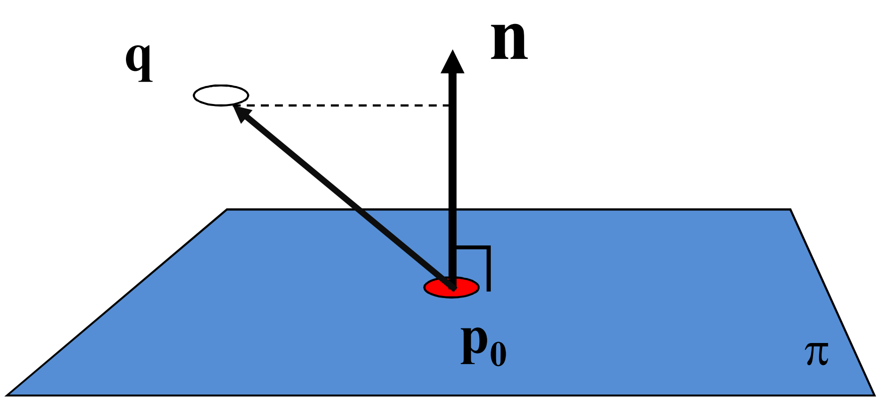
- A plane π is defined by a normal $\mathbf{n}$ and one point in the plane $\mathbf{p_0}$.
- A point $\mathbf{q} \in$ plane $\Leftrightarrow\ < \mathbf{q} - \mathbf{p_0}\, ,\ \mathbf{n} > = 0$
- The normal $\mathbf{n}$ is perpendicular to all vectors in the plane



Distance between point and plane

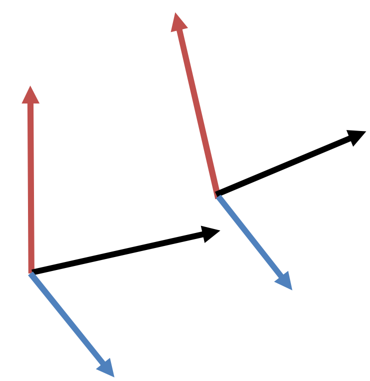
- Geometric way:
  - Project ($\mathbf{q}$ - $\mathbf{p}_0$) onto $\mathbf{n}$!

$$dist = \frac{|< \mathbf{q} - \mathbf{p}_0, \mathbf{n} >|}{\|\mathbf{n}\|}$$



Coordinates

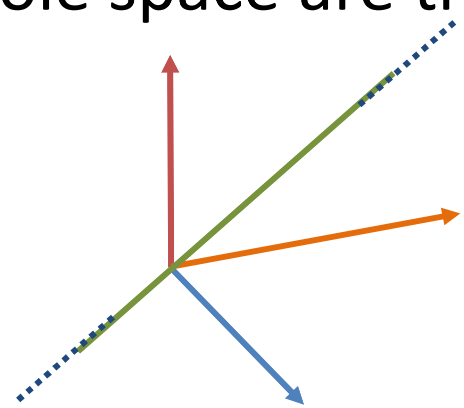
- Connect drawing plane/space with R^2 or R^3
- Coordinate origin and axes are problem specific
  - Example: orthogonal coordinates in the lower corner of this room
- Affine spaces have
  - No fixed origin
  - No fixed axes
  - (which is not the case in linear spaces)



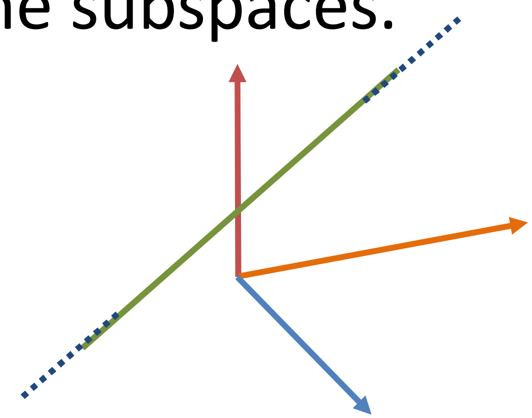
Coordinates

Affine space

- "An affine space is a vector space that's forgotten its origin" – John Baez
  - In *R^3*, the origin, lines and planes through the origin and the whole space are linear
  - points, lines and planes in general as well as the whole space are the affine subspaces.



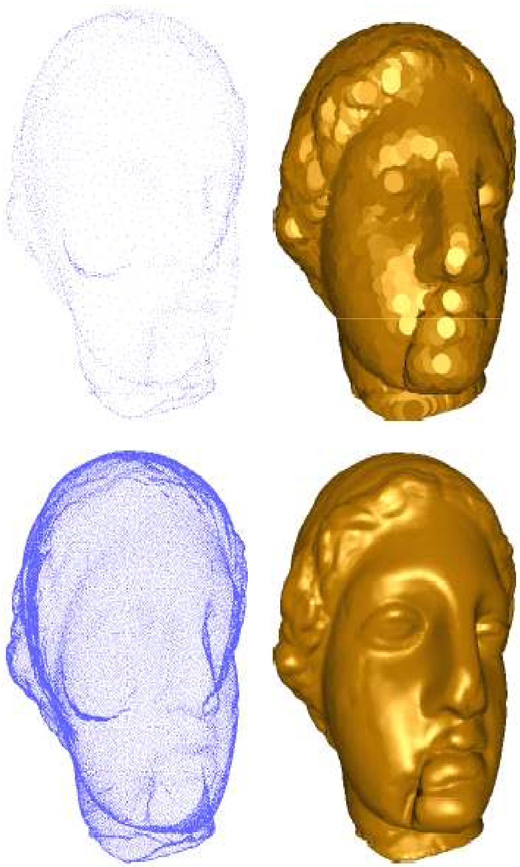
Linear subspace



Affine subspace

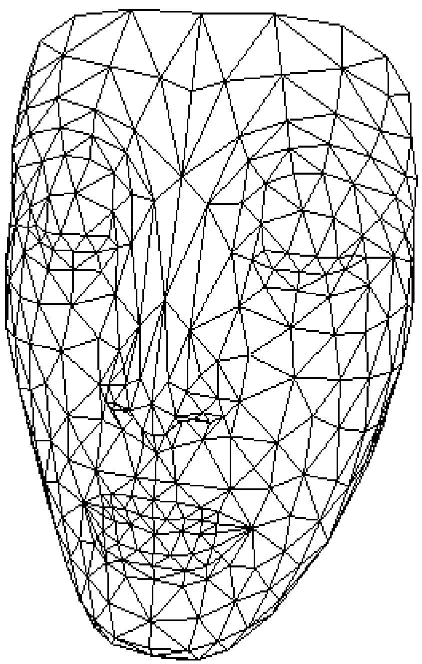
Primitives

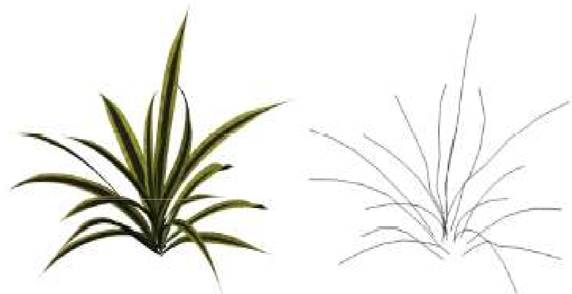
Points



Primitives

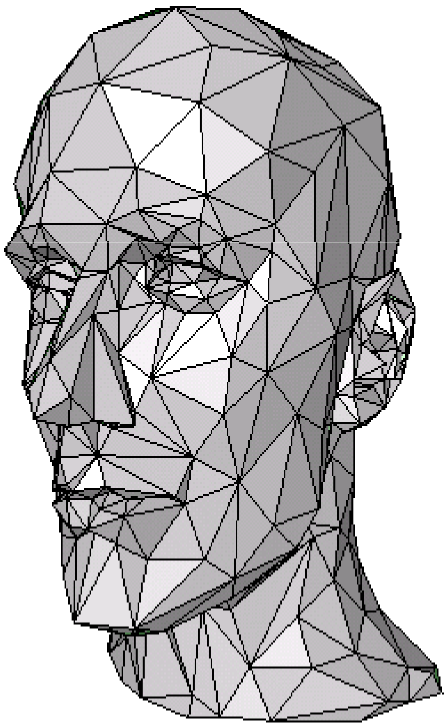
Lines





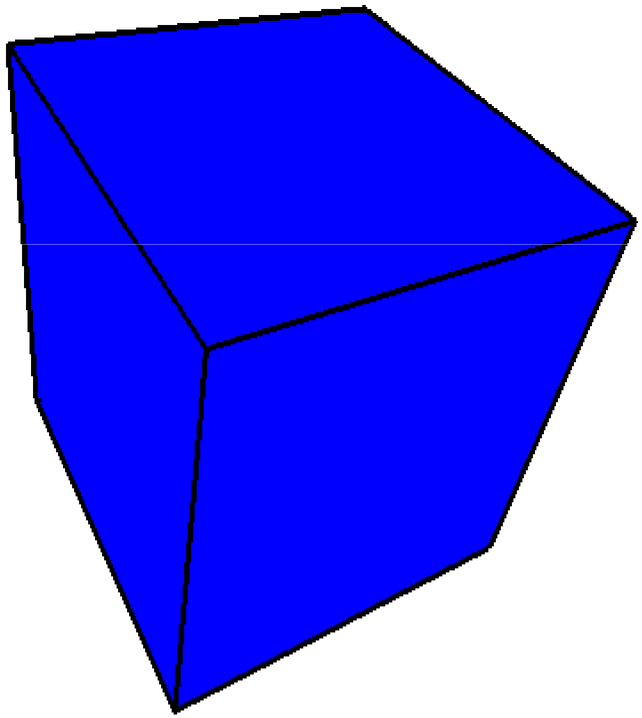
Primitives

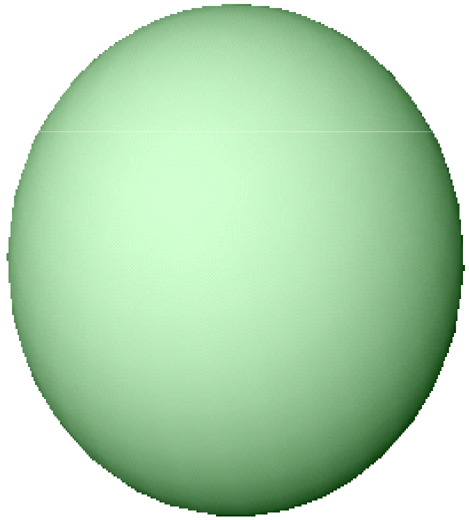
Triangles



Primitives

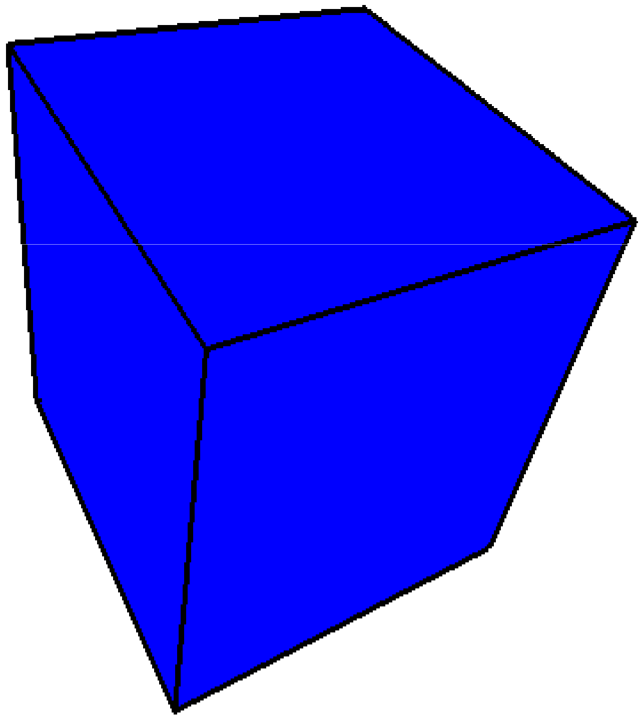
Shapes

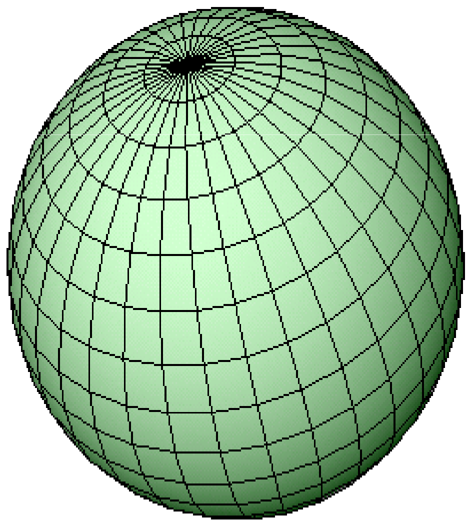




Primitives

Shapes … are tessellated





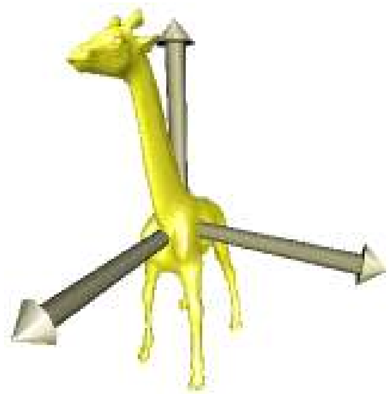
Primitives

Positioning

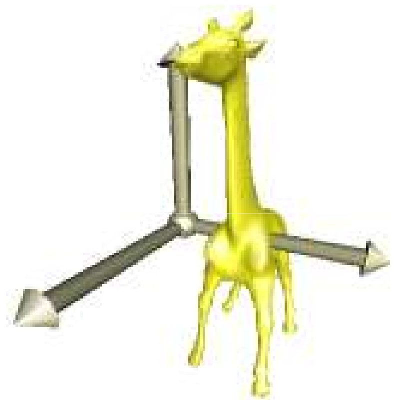
- Absolute coordinates?

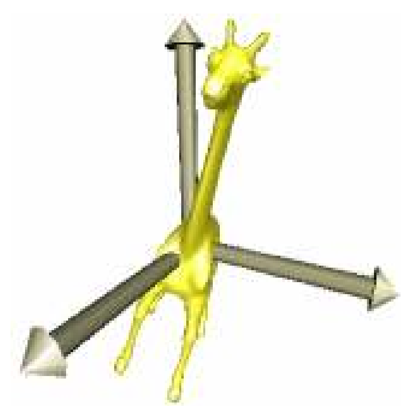
Primitives

Positioning



- Transformation + relative coordinates
  - Translation
  - Rotation
  - Scaling
  - Shearing
- Affine maps / Transformations!





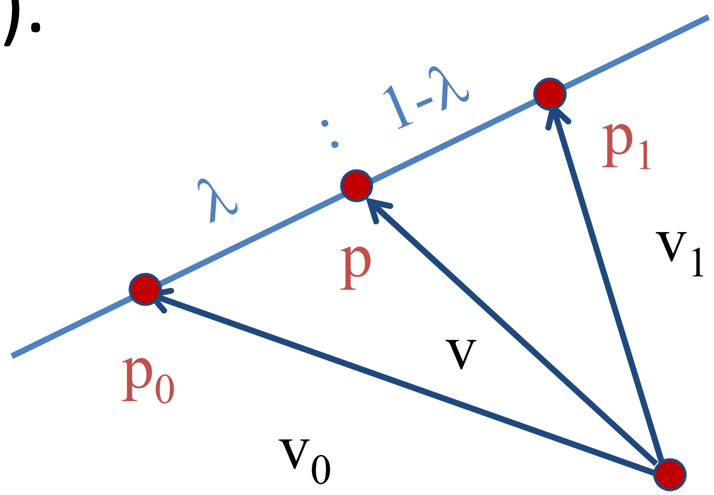
Affine maps

Affine combinations

- The set

$$\left\{ v \in V \mid v = \sum_{i=0}^{n} \lambda_i \cdot v_i, \quad \sum_{i=0}^{n} \lambda_i = 1 \right\}$$

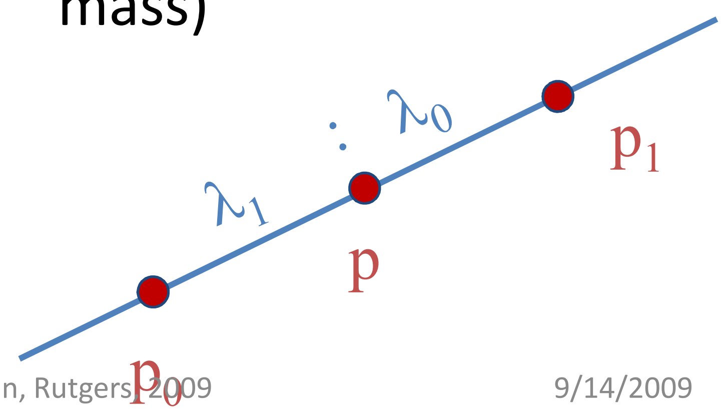
is an affine combination of vectors $\mathbf{v}_i$ (or of points $\mathbf{p}_i$).



Affine maps

Barycentric coordinates

- Given and affine space A with coordinate system $\mathrm{B}=\{\mathbf{p}_0,\ldots\mathbf{p}_n\}$
- For a point $p=\sum_{i=0}^{n}\lambda_i \cdot p_1$ with $\sum_{i=0}^{n}\lambda_i = 1$
  the λ_i are known as **barycentric coordinates**
- Physical interpretation:
  - Points $\mathbf{p}_i$ have mass $\lambda_i \rightarrow \mathbf{p}$ ist the centroid (= center of mass)



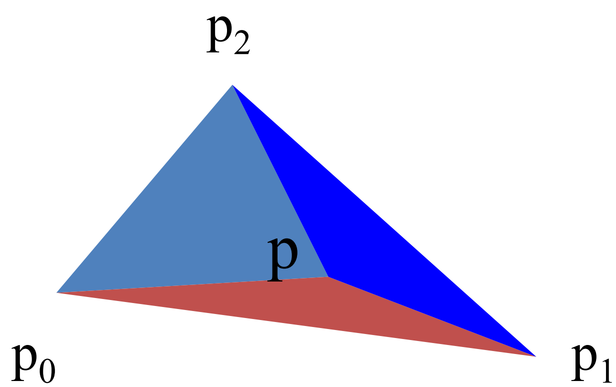
$$\lambda_0 + \lambda_1 = 1$$

$$\lambda_0 = \frac{\|p_1 - p\|}{\|p_1 - p_0\|}$$

$$\lambda_1 = \frac{\|p - p_0\|}{\|p_1 - p_0\|}$$

Affine maps

Barycentric coordinates



$$p = \lambda_0 \cdot p_0 + \lambda_1 \cdot p_1 + \lambda_2 \cdot p_2$$

$$\lambda_0 = \frac{A(\Delta(p, p_1, p_2))}{A(\Delta(p_0, p_1, p_2))}$$

$$\lambda_1 = \frac{A(\Delta(p, p_0, p_2))}{A(\Delta(p_0, p_1, p_2))}$$

$$\lambda_2 = \frac{A(\Delta(p, p_0, p_1))}{A(\Delta(p_0, p_1, p_2))}$$

$$A(\Delta(p_0, p_1, p_2)) = \frac{1}{2}\left|(p_1 - p_0) \times (p_2 - p_0)\right|$$

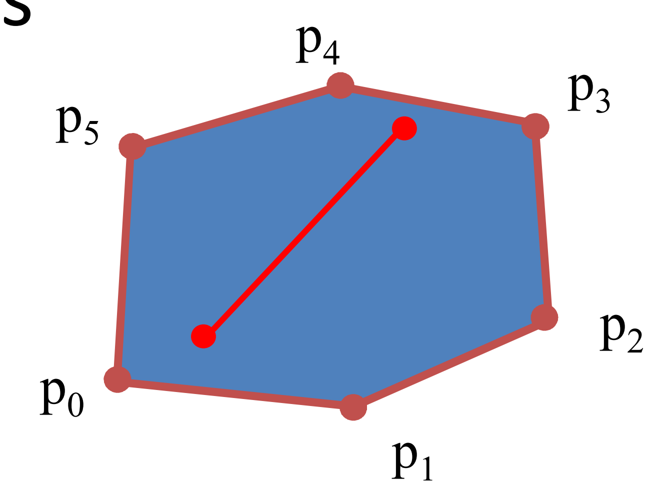
Affine maps

Convex hull

- The set

$$co\{p_0,\ldots,p_n\}=\left\{p \mid p=\sum_{i=0}^{n}\lambda_i \cdot p_i, \sum_{i=0}^{n}\lambda_i = 1, \qquad \lambda_i \geq 0, i=0,\ldots,n\right\}$$

  is the convex hull co{$p_0,\ldots,p_n$} of points $p_0,\ldots,p_n$
- The convex hull contains all convex combinations of the points
  - Convex combinations = affine combinations /w barycentric coordinates greater/equal to zero



Affine maps

...as linear maps

- A map $\Phi: \mathrm{R}^n \rightarrow \mathrm{R}^m$ is affine
  - when Φ can be represented as $\Phi(\mathbf{v})=\mathrm{A}(\mathbf{v})+\mathbf{b}$ where A is a linear map and $\mathbf{b} \in \mathrm{R}^m$
- Affine maps have a linear part (multiplication) and a translation (additive)

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

Linear transformation

Translation

Affine transformations

- Preserve parallel lines
  - lines → lines, planes → planes
- Might not preserve length and angles
  - But do preserve relative length along lines
- If they do preserve length and angles then the transformation is an **isometry**

- **Affine = linear + translation**

Affine maps

...as linear maps

- Leads to the use of **projective geometry**
- 2D points **and** vectors represented as **(x, y, w)** → homogeneous coordinates

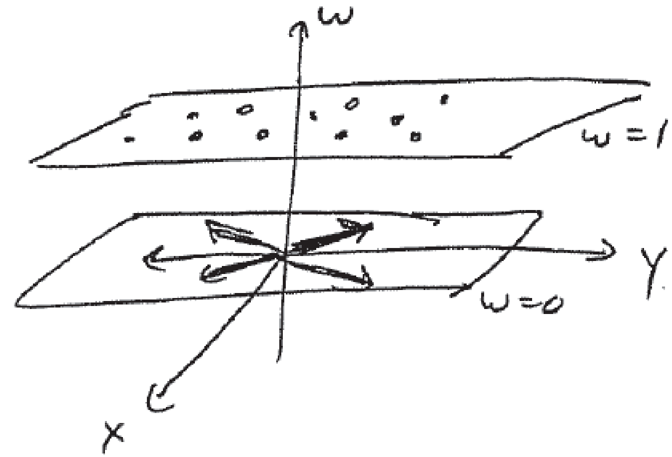
w = 1	point
w = 0	vector

- Point (0, 0, 0) not allowed, so domain $R^3 - \{(0, 0, 0)\}$
- If $w \in (0,1]$ then (x, y, w) → (x/w, y/w, 1)

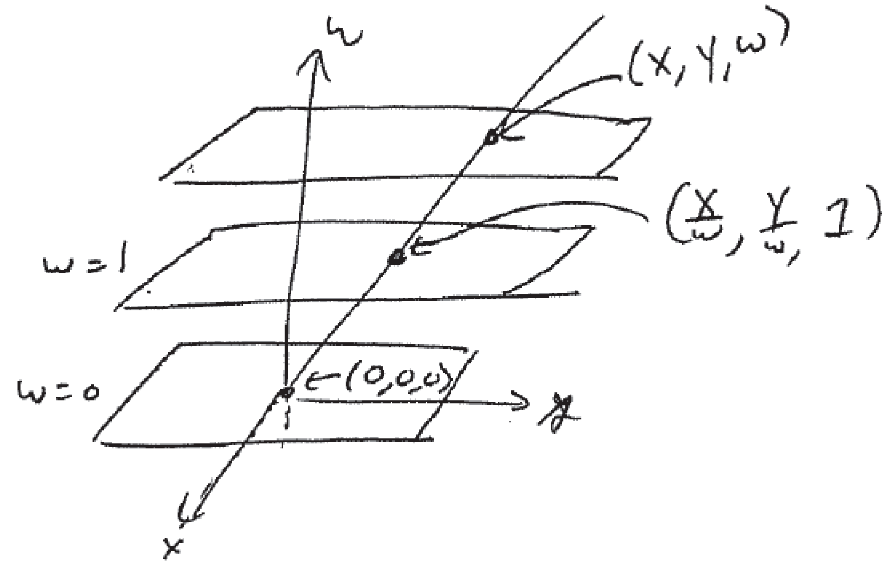
↖ **A point**

What is w ?

2D case!



- A kind of a **type**
- Points + "points at infinity"
  - Points at infinity are not affected by translation
- Infinite # of points correspond to (x, y, 1) → $\{(tx, ty, t) \mid t \neq 0\}$
  - Line through origin – {origin}

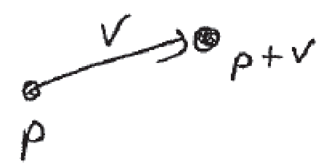


Homogeneous coordinates

- Works nicely for points and vectors

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ 1 \end{bmatrix}$$

(point) + (vector) = (point)



$$\frac{1}{2}\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + q_x \\ p_y + q_y \\ 1 \end{bmatrix}$$

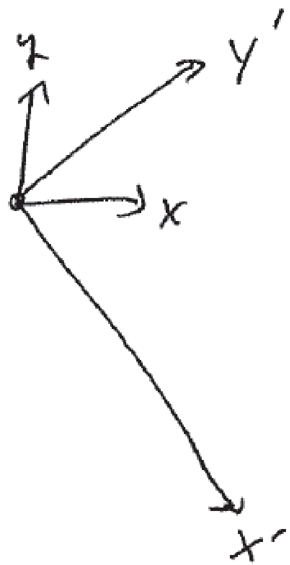
(affine c. of pts) (point)

in 3D: (x, y, z, w)

- Adding and scaling works too
- More in “projections”, where $w \in [0,1]$

Linear transformation

- Purely linear transformation



$$x' = ax + cy$$
$$y' = bx + dy$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & c \\ b & d \end{bmatrix}}_{\text{matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

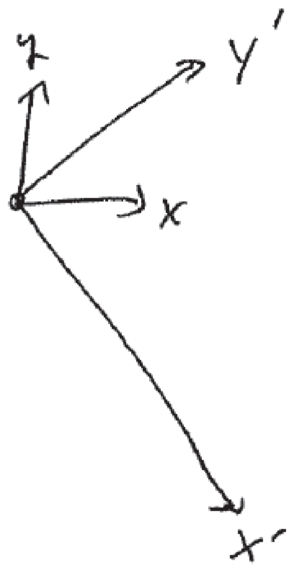
– homogeneous version –

or $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

- Origin does not move
- New coordinate axes are lin. comb. of old ones

Linear transformation

- Purely linear transformation



$$x' = ax + cy$$
$$y' = bx + dy$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & c \\ b & d \end{bmatrix}}_{\text{matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

or

– homogeneous version –

$$\begin{bmatrix} x' \\ y' \\ 0 \end{bmatrix} = \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

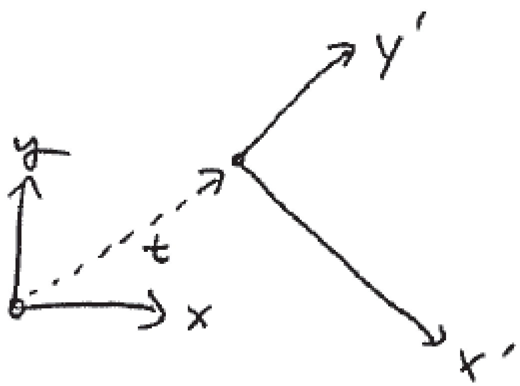
if $x + y$ are $\hat{i}, \hat{j}$ $\left(\begin{array}{l} x = (1, 0, 0) \\ y = (0, 1, 0) \end{array} \right)$

$x' = \begin{bmatrix} a \\ b \end{bmatrix}$ $y' = \begin{bmatrix} c \\ d \end{bmatrix}$ (columns of matrix)

Affine transformation

as a linear transformation + translation in n dimensions

- Origin moves → translation



$t = (e, f, o)$

still, for vectors,

$x' = ax + cy$

$y' = bx + dy$

but points:

$p_x' = ap_x + cp_y + e$

$p_y' = bp_x + dp_y + f$

Affine transformation

as a linear transformation in n+1 dimensions

- Origin moves → translation

$$\begin{bmatrix} p_x' \\ p_y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$

points

good for points and vectors!

$$\begin{bmatrix} x' \\ y' \\ 0 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

vectors

What is so great about this?

- Easy to implement
- Checks for errors in the implementation
  - Can always check the w coordinate to make sure that points and vectors remain unchanged
- **Unified representation** for linear + translation
  - Can compose many transformations into a single matrix through concatenation

$$\mathbf{M = M_{rot} \cdot M_{scale} \cdot M_{translate} \cdot \ldots}$$