Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA

Grafica al Calcolatore

Riassunto dei principali argomenti

Autore:
Danzi Matteo
Matricola VR388529

Anno Accademico 2016/2017

Indice

1 Introduzione

Cos'è grafica al calcolatore ? Intuitivamente è l'uso di un calcolatore per produrre un'immagine o una sequenza di immagini, non necessariamente realistica o in 3D, non necessariamente interattiva.
Per **computer grafica**, **grafica digitale** o **grafica computerizzata** (in inglese **computer graphics**) si intende:

- Creazione immagini 2d sintetiche e animazioni
- Modellazione 2D, 3D, anche con comportamenti fisici
- Computer Aided Design
- Rendering delle scene, cioè creazione delle immagini simulando la proiezione ottica delle scene sulla camera
- Animazione
- Interfacce grafiche dei computer
- Realtà virtuale
- Enhancement video televisivo
- Visualizzazione scientifica e dell'informazione

1.1 Storia

Nasce con i primi display per calcolatori.
Nel 1960 William Fetter introduce il termine **Computer Graphics** per descrivere la ricerca che stava conducendo alla Boeing. Questa ricerca ha portato alla realizzazione di un modello 3D del corpo umano per progettare la carlinga degli aerei. Nasce quindi l'idea della modellazione 3D che rappresenta una parte rilevante della moderna CG.
Nel 1963 assistiamo alla nascita della **Computer Grafica interattiva**: sistema sketchpad di Ivan Sutherland. In questo caso si tratta della prima **interfaccia grafica** interattiva.
Negli anni '60 inoltre nascono i primi terminali grafici e i primi giochi, si impara quindi a disegnare sullo schermo 2D. Nel 1961 Steve Russell at MIT crea il primo video game, Spacewar.
Negli anni '70 nascono le moderne **interfacce grafiche interattive** dei computer (WIMP - *Window*, *Icon*, *Menu and Pointing device*). La grafica interattiva, in questo caso 2D diventa parte integrante del sistema di interazione uomo-macchina.
Nel 1972 nasce il videogioco Pong (Atari). Anche oggi una delle maggiori applicazioni della grafica interattiva è nel mondo dei **videogiochi**.
Negli anni settanta nascono gli algoritmi per creare immagini da modelli 3D (**rendering**). Nel 1972 Catmull (Univ. Utah) crea la prima animazione di grafica 3D. Si tratta di un modello della sua mano formato da 350 poligoni. Catmull diventerà un cofondatore della *Pixar* (oggi presidente).
Gli algoritmi per creare linee raster, riempire poligoni, proiettare oggetti 3D su telecamere virtuali vengono via via sviluppati negli anni '60-70-80. Questi argomenti sono il cuore della grafica 3D e di questo corso.
Si creano standard e implementazioni di sistemi grafici e si arriva alla situazione attuale:

- 1992 Silicon Graphics crea lo standard **OpenGL**
- 1995 Microsoft rilascia Direct 3D

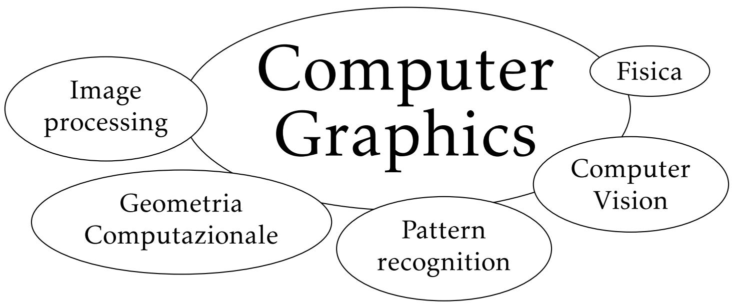
La grafica ha pesantemente condizionato lo sviluppo dell'hardware e l'architettura dei moderni calcolatori (e tablet, smartphone, ...)
Le operazioni grafiche vengono implementate su hardware specifico Inizialmente grafica raster calcolata su CPU, poi (doppio) buffer per mantenere le immagini (doppio perché il calcolo può

essere lento rispetto al refresh dello schermo) Nel 1985 esce il Commodore Amiga, uno dei primi home computer con una GPU (Graphical Processing Unit), nel 1987 il primo PC Ibm con operazioni 2D hardware.
Nel 1995 escono le prime schede video per PC con pipeline grafica 3D (S3 Virge), nel 1999 Nvidia GeForce 256 la prima scheda con transform & lightning engine.

1.2 Applicazioni

- **Modellazione 3D**: prototipazione e stampa 3D, digital manufacturing
- **Grafica non interattiva**: cinema digitale, grafica pubblicitaria, ecc.
- **Grafica interattiva**:
  - *Visualizzazione scientifica*: uso della grafica (2D-3D) per comunicare efficacemente informazione di misure o simulazioni
  - *Visualizzazione dell'informazione*: creazione di modelli "mentali" utili per rappresentare nello spazio dati astratti
  - Realtà virtuale o aumentata e interazione uomo macchina
  - Interfacce naturali per comunicare con i computer o simulare attività reali
  - Simulatori, Videogiochi

Grafica è quindi una *disciplina che studia le tecniche e gli algoritmi per la rappresentazione visuale di informazioni numeriche prodotte o semplicemente elaborate dai computer* (da Scateni e al.).
È quindi legata a molte altre discipline.



1.3 Computer Graphics vs Computer Vision

In senso generale la grafica è il meccanismo opposto dell'**image understanding** o della **computer vision**.
Nel primo caso si passa da immagini a parametri, a interpretazione. Nel secondo si crea un'immagine da un input parametrico. Quindi sono grafica tutti i sistemi informatici che creano e usano immagini sintetiche.

1.4 Visual Computing

Oggi vista la convergenza dei due domini si parla spesso in generale di *visual computing*.
Visual computing is a generic term for all computer science disciplines handling with images and 3D models, i.e. computer graphics, image processing, visualization, computer vision, virtual and augmented reality, video processing, but also includes aspects of pattern recognition, human computer interaction, machine learning and digital libraries. The core challenges are the acquisition, processing, analysis and rendering of visual information (mainly images and video). Application areas include industrial quality control, medical image processing and visualization, surveying, robotics, multimedia systems, virtual heritage, special effects in movies and television, and computer games.

2 Applicazioni

Se lo scopo della grafica al calcolatore è quello di riprodurre un grafico, quel che dovrà fare il nostro software è preparare i valori di output da passare al nostro display.
Il display (output) può essere differente a seconda dell'applicazione. In generale sarà un **display raster** come un monitor, che riproduce una matrice di punti su cui è codificata una terna di valori di colore RGB Sono dominio della grafica le applicazioni che

- Preparano file per la stampa 2D (grafica vettoriale)
- Stampa 3D
- Display innovativi (ad esempio stereo, volumetrici)

Possiamo rappresentare la grafica in un dato digitale in due modi:

1. **Vettoriale** : utilizzando primitive di disegno
2. **Raster**: utilizzando una griglia di valori da riprodurre sul monitor

2.1 Grafica vettoriale

La rappresentazione grafica vettoriale compone le immagini come un ***insieme di primitive di disegno*** come ad esempio *Linee*, *Curve*, *Aree*.
Queste possono essere descritte con **funzioni parametriche** e **coordinate di punti**.
Dove vengono applicate:

- Disegno su display vettoriali.
- Plotter.
- Rappresentazione per la stampa, necessita di conversione a raster di solito effettuata dalla stampante.
- Rappresentazione interna nei calcolatori di forme grafiche che devono essere rappresentate a differente livello di precisione (Es. caratteri di stampa che facciamo grandi o piccoli, ecc.).

Vantaggi:

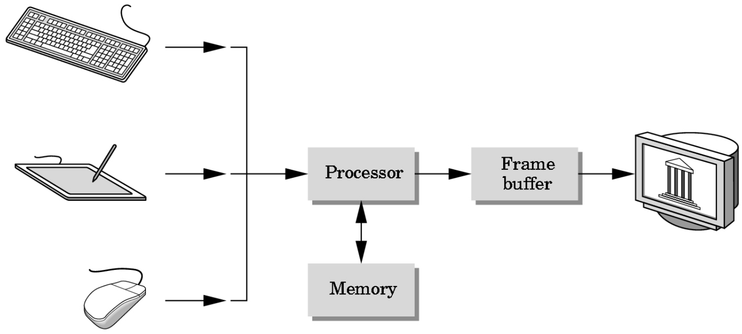
- I dati sono espressi in una forma direttamente comprensibile ad un essere umano (es. lo standard SVG);
- Compattezza di codifica rispetto all'equivalente raster;
- Possibilità di ingrandire l'immagine arbitrariamente, senza che si verifichi una perdita di risoluzione dell'immagine stessa.

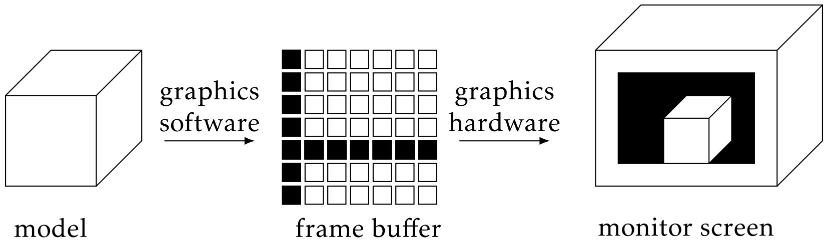
Limiti: per la rappresentazione sulla maggior parte dei display occorre poi convertire a raster.

Figura 1: Differenza di scalatura tra raster(sx) e vettoriale(dx)

2.2 Immagini Raster o bitmap

Le immagini digitali cui siamo abituati con i monitor immagini **raster** e consistono di una matrice di elementi denominati **pixel** dove ogni cella della matrice rappresenta un colore in rgb.
Il nostro software scriverà quindi un *frame buffer*: memoria contenente l'immagine, array di valori per i pixel, che viene modificato direttamente dal programma di grafica video controller il quale legge il frame buffer e costruisce l'immagine sul display.





Le immagini raster sono **matrici** che contengono valori che rappresentano il colore nella casella corrispondente agli indici con valori discretizzati.
Di solito ci sono componenti di colore: i monitor generano il colore con sovrapposizione di luce rossa, verde e blu (Perché?)
Caratteristiche principali (non le uniche):

- **risoluzione** (dimensioni della matrice di pixel)
- **profondità di colore** (bit di memoria per pixel).

8-bit significano 256 colori, mentre 24-bit (o truecolor) rappresentano all'incirca 32 milioni di colori
Nota: è la rappresentazione di uscita tipica di quasi tutti i display odierni, ma non è ovviamente l'unica possibile

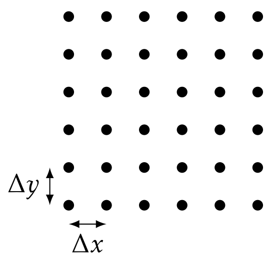
- Es.: display vettoriali: riproducono disegni
- Il formato digitale creato internamente deve ovviamente corrispondere alla capacità del display scelto di riprodurlo

Nel processo di rasterizzazione delle immagini vettoriali la qualità della conversione dipende dalla risoluzione dell'immagine originale (numero di punti o punti per pollice).
Un effetto possibile può essere la scalettatura: effetto dell'**aliasing** delle alte frequenze. Si può ridurre tale effetto sfumando la luminosità e facendo quindi *antialiasing*.

2.3 Aliasing

L'aliasing accade perché l'immagine è il campionamento di un segnale "continuo" che rappresenterebbe il dato misurabile.

$$I(i,j) = I(i\Delta x, j\Delta y) = \iint F(x,y)\delta(x - i\Delta x)\delta(y - i\Delta y)$$



Per il *teorema di Shannon/Nyqvist* del campionamento, non si può ricostruire esattamente il segnale originale se la frequenza del segnale è superiore alla metà della frequenza di campionamento.

$$v_{cx} = \frac{1}{\delta x} \geq 2v_{xmax} \qquad v_{cy} = \frac{1}{\delta y} \geq 2v_{ymax}$$

Dove ci sono discontinuità del colore, ci sono componenti a frequenza alta, si creano artefatti. I filtri che fanno antialiasing attenuano le alte frequenze.

2.4 Caratteristiche Immagini raster

Le principali caratteristiche delle immagini raster sono:

- Risoluzione (numero di righe e colonne matrice)
- Range dinamico: rapporto tra minima differenza misurabile o rappresentabile e range di variabilità del segnale (luminosità). Corrisponde al numero di bit con cui codifichiamo il valore. Tipicamente 8 bit ma si può andare oltre:
  - Immagini HDR
  - Immagini mediche
  - Dato che l'occhio umano non distingue così tante sfumature, lo scopo è di poter creare da esse rendering diversi che possano dare differenti effetti o informazioni

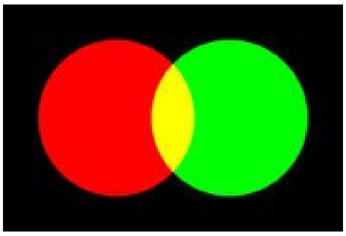
  Il valore codificato dovrebbe corrispondere alla luminosità del punto generata dal monitor o acquisita dal sensore, ma la cosa è un po' più complicata a causa della **non-linearità della percezione umana**.

Insomma quello che c'è nel file non è quello che vediamo. Il rendering ed il dispositivo determinano la visualizzazione. E poi c'è il fattore legato alla percezione umana. Ad esempio: immagini ad alto range dinamico (HDR) (Mantiuk et al 2005)
La percezione umana amplifica le differenze del range dinamico ai bassi livelli. Macchine fotografiche e monitor applicano correzioni (gamma correction)

2.4.1 Colore

Le immagini raster da inviare ai display sono in genere a colori, per simulare il modo in cui vediamo il mondo (a colori). La rappresentazione del colore è generalmente una *terna di valori RGB*. Per la riproduzione corrispondono alle intensità emesse da tre emettitori di luce a tre frequenze determinate (rosso, verde, blu) che danno origine in corrispondenza a un certo colore percepito dall'utente.

Ma cosa significa questo?
Nei monitor si generano i colori nei punti della griglia per sintesi additiva: si mescolano due o più fasci luminosi di diversa.
Nella stampa per sintesi sottrattiva: Due o più inchiostri sovrapposti assorbono diverse frequenze e cambiano la luce diretta all'occhio.
Non tutti i colori possono essere generati in mescolanza additiva o sottrattiva di tre colori primari La scelta di **Rosso Verde Blu** come *primari additivi* e **Giallo, Magenta e Cyan (e nero)** *sottrattivi* cerca di massimizzare i colori rappresentabili



L'uso delle 3 componenti di colore RGB è convenzionale e deriva dalla fisiologia della visione, che mostra che con 3 colori base si possono approssimativamente riprodurre i colori del mondo reale.
I colori visibili però derivano invece da una variazione continua di lunghezza d'onda delle radiazioni elettromagnetiche in un intervallo percettibile di valori circa 370-730 nm.
Percezione del colore: nella retina ci sono 3 tipi di coni, che hanno differenti sensitività a diverse frequenze S,M,L Possiamo fare il matching delle frequenze con la risposta dei recettori.

Il "colore" percepito è dato da 3 grandezze scalari, funzione dello spettro della luce incidente. La corrispondenza non è iniettiva. Spettri diversi possono corrispondere allo stesso colore percepito: metamerismo Conseguenza: Per riprodurre un colore, non è necessario riprodurre lo spettro. È sufficiente che le risposte L, M, S dei coni siano uguali. Può cambiare in funzione dell'illuminazione.
Metamerismo: consiste nella possibilità di ottenere un effetto ottico tale che l'occhio percepisca la stessa sensazione di colore in presenza di luce con distribuzione spettrale diversa dal colore puro in questione. Si tratta di un'illusione ottica basata sulla natura dell'interpretazione del colore da parte dell'occhio umano, è possibile creare la sensazione di un colore "puro", formato, selezionando la sola lunghezza d'onda che genera quella determinata sensazione di colore miscelando a dovere più lunghezze d'onda differenti, un esempio è il bianco di una lampada fluorescente formato da spettri non uniformi, in questo caso la temperatura di colore che si trova sulle confezioni è la temperatura a cui deve essere un corpo nero perché l'occhio umano percepisca la stessa sensazione di colore. Il fenomeno si ha quando colori che appaiono all'occhio identici sotto una certa luce, mostrano tonalità differenti se illuminati con una luce diversa. In sostanza c'è metamerismo quando due colori si equivalgono sotto una fonte di luce, ma risultano differenti ad altre esposizioni.

2.4.2 Legge di Grassman

L'uomo è in grado di fare match di colori mischiando 3 (o più) colori detti primari. Se la luce test T ha un certo colore:

$$T = aP1 + bP2 + cP3$$

il match si verifica lineare.

2.4.3 Funzioni di Matching

Data una terna di colori di base possiamo studiare il matching dei colori sugli osservatori in funzione della frequenza. Componenti colore CIERGB ricavate dall'integrale delle frequenze dello stimolo su tutto il range.

$$L = \int \Phi(\lambda) L(\lambda)\, \mathrm{d}\lambda$$
$$M = \int \Phi(\lambda) M(\lambda)\, \mathrm{d}\lambda$$
$$S = \int \Phi(\lambda) S(\lambda)\, \mathrm{d}\lambda$$

Rappresentazione con matrice:

$$C = \begin{pmatrix} \overline{r}(\lambda_1) & \dots & \overline{r}(\lambda_N) \\ \overline{g}(\lambda_1) & \dots & \overline{g}(\lambda_N) \\ \overline{b}(\lambda_1) & \dots & \overline{b}(\lambda_N) \end{pmatrix}$$

$$\Phi = \begin{pmatrix} \phi(\lambda_1) \\ \vdots \\ \phi(\lambda_N) \end{pmatrix}$$

3 Schema di un'applicazione grafica

Vi è una descrizione di qualche tipo (procedurale o meno) del mondo che deve essere rappresentato. La produzione di tale descrizione (modello) prende il nome di **modellazione**.
Da tale descrizione si ottiene una immagine visualizzabile da un display tale processo è chiamato globalmente **rendering**.
La sequenza di procedure ed algoritmi che implementano il rendering prende il nome di **pipeline grafica**.
Se l'applicazione è interattiva, il disegno dev'essere riprodotto in real time mentre l'utente interagisce con la scena mediante dei dispositivi.

3.1 Modello della scena

Nelle applicazioni 2D può essere un disegno da riprodurre sul display a meno di una trasformazione geometrica e mappatura sui pixel. Nelle applicazioni 3D di cui ci interesseremo sarà invece un vero e proprio modello del "mondo" che vogliamo vedere (e con cui vogliamo interagire) e l'immagine sarà generata simulando il processo di acquisizione di immagini di una telecamera "virtuale".
Dati gli oggetti della scena, quindi dovremo "simulare" la geometria e la fisica della formazione delle immagini (luce, colore)

3.2 Rendering della scena

Il passaggio dalla rappresentazione all'immagine si definisce *rendering*.
Comprende tutti gli algoritmi per creare l'immagine per il display, che supporremo voglia un'immagine raster.
Quindi se partiamo da una **rappresentazione 2D** (grafica vettoriale) il rendering consiste in:

1. Trasformazione delle primitive in rappresentazioni di colore sui pixel (rasterizzazione)
2. Eventuale modifica interattiva del disegno

Se partiamo da una **rappresentazione di scena 3D** il rendering consiste in:

1. Proiezione della scena sul piano immagine della telecamera virtuale
2. Trasformazione della scena proiettata in rappresentazioni di colore sui pixel (rasterizzazione)
3. Eventuale interazione con la scena e conseguenteupdate del rendering

Il rendering comprende molti calcoli da svolgere, la **complessità** dipende dall'applicazione di interesse:

- Applicazioni interattive, real-time:
  - Frame rate alto (>10 fps)
  - Tempo di rendering del singolo frame prefissato

  - Si può/deve sacrificare la qualità per garantire l'interattività

- Applicazioni non interattive (computer animation, grafica pubblicitaria)
  - l'obiettivo primario: massima qualità delle immagini di sintesi
  - Non si hanno vincoli sul tempo di generazione del singolo frame
  - Animazioni calcolate frame by frame da PC cluster, ricomposte successivamente nella successione temporale corretta

Come si implementa la fase di rendering?

- Applicazioni interattive: si avvalgono pesantemente delle moderne schede grafiche (HW dedicato al processing di dati 3D)
- Applicazioni non interattive: fanno uso di ambienti di rendering più sofisticati e flessibili (ad es. RenderMan), spesso eseguiti SW su cluster di PC

4 Modellare lo spazio

Richiamiamo le nozioni basilari di geometria per modellare lo spazio e gli oggetti:

- **Scalari**: unidimensionali, possono rappresentare grandezze fisiche con numeri
- **Punti**: rappresentano una posizione nello spazio
- **Vettori**: rappresentano le direzioni o le distanze tra punti in 2D o 3D

Per definire una posizione nello spazio dobbiamo introdurre un sistema di riferimento con un punto fisso detto origine e una terna di direzioni ortogonali

4.1 Scalari

Gli scalari S costituiscono un corpo (tipicamente useremo IR) con due operazioni, somma e moltiplicazione, che soddisfano le seguenti relazioni:

$$\forall \alpha, \beta, \gamma \in S$$

Commutatività

$$\alpha + \beta = \beta + \alpha$$
$$\alpha\beta = \beta\alpha$$

Associatività

$$\alpha + (\beta + \gamma) = (\beta + \alpha) + \gamma$$
$$(\alpha\beta)\gamma = \alpha(\beta\gamma)$$

Distribuzione

$$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$$

Elementi neutri

$$\exists 0 \in S \, : \, \forall \alpha \in S \quad \alpha + 0 = \alpha$$
$$\exists 1 \in S \, : \, \forall \alpha \in S \quad \alpha 1 = \alpha$$

Elementi inversi

$$\forall \alpha \in S \quad \exists(-\alpha) \in S \, : \, \alpha + 0 = \alpha$$
$$\forall \alpha \in S \quad \exists \alpha^{-1} \in S \, : \, \alpha\alpha^{-1} = 1$$

4.2 Vettori

I vettori costituiscono un **gruppo abeliano** (commutativo) V in cui e definito il **prodotto di un vettore per uno scalare**.

Chiusura

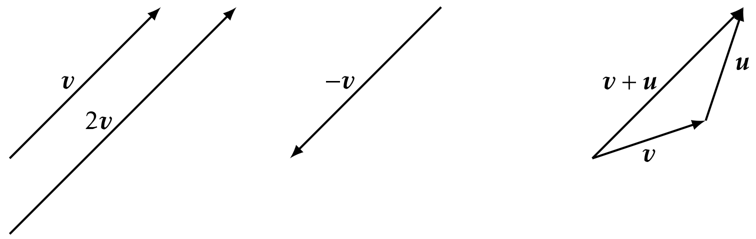
$$\boldsymbol{u} + \boldsymbol{v} \in V \quad \forall \boldsymbol{uv} \in V$$
$$\alpha \boldsymbol{v} \in V \quad \forall \alpha \in S \; \boldsymbol{v} \in V$$

Proprietà Algebriche

$$\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$$
$$\boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w}$$
$$\alpha(\boldsymbol{u} + \boldsymbol{v}) = \alpha\boldsymbol{u} + \alpha\boldsymbol{v}$$
$$(\alpha + \beta)\boldsymbol{u} = \alpha\boldsymbol{u} + \beta\boldsymbol{u}$$

$$\exists 0 \in V \, : \, \forall \boldsymbol{u} \in V \quad \boldsymbol{u} + 0 = \boldsymbol{u}$$
$$\forall \boldsymbol{u} \in V \; \exists(-\boldsymbol{u}) \in V \, : \, \boldsymbol{u} + (-\boldsymbol{u}) = \boldsymbol{0}$$

La definizione è totalmente astratta, ma per semplicità conviene considerare due utili esempi di spazi vettoriali lineari: Geometrico e Algebrico.
Un esempio concreto e dato dai segmenti orientati liberi, ovvero senza un punto di applicazione specificato Il prodotto con uno scalare (numeri reali) cambia la lunghezza del vettore. La somma di due vettori e data dalla regola del parallelogramma.



Un altro esempio è dato dall'insieme delle n-ple ordinate di $\mathbb{R}^n$.

$$\boldsymbol{v} = (\beta_1, \dots, \beta_n) \quad \beta_i \in \mathbb{R} \forall i$$

Il prodotto per uno scalare e la somma di due vettori sono definiti in modo del tutto naturale:

$$(\alpha_1, \dots, \alpha_n) + (\beta_1, \dots, \beta_n) = (\alpha_1 + \beta_1, \dots, \alpha_n + \beta_n)$$
$$\alpha(\beta_1, \dots, \beta_n) = (\alpha\beta_1, \dots, \alpha\beta_n)$$

E' facile vedere qual è l'elemento neutro e qual è l'inverso di un vettore.

4.2.1 Indipendenza lineare

Dati n vettori non nulli, si dicono **linearmente indipendenti** se qualsiasi loro combinazione lineare a coefficienti non tutti nulli è diversa dal vettore nullo.

$$\alpha_1 \boldsymbol{v}_1 + \dots + \alpha_n \boldsymbol{v}_n = \mathbf{0} \Leftrightarrow \alpha_i = 0 \forall i$$

Si dice **dimensione** di uno spazio vettoriale il massimo numero di vettori linearmente indipendenti.
In uno spazio vettoriale a dimensione n, un insieme di n vettori linearmente indipendenti si dice una **base** per lo spazio. Ogni vettore può essere scritto come combinazione lineare dei vettori di una base.

4.2.2 Rappresentazione in componenti

Fissata quindi una **base in uno spazio vettoriale**, ad ogni vettore corrisponde una n-pla di scalari, ovvero i coefficienti dello sviluppo lineare del vettore nei vettori di base; tali scalari sono le componenti del vettore rispetto alla base data.
In genere il corpo è dato dai reali; abbiamo quindi ottenuto la rappresentazione concreta vista prima di uno spazio vettoriale astratto come insieme di n-ple di $\mathbb{R}^n$. Tale rappresentazione dipende dalla base scelta.

4.3 Punti

I vettori non rappresentano punti nello spazio, ma solo spostamenti. Per poter introdurre il concetto di **posizione** si deve passare agli **spazi affini** che sono degli spazi vettoriali a cui si aggiunge il concetto astratto di punto.
I punti sono definiti in senso astratto come nuovi elementi con cui e possibile effettuare solo una operazione: la sottrazione tra punti.

La differenza di due punti è un vettore: $P - Q = \boldsymbol{v}$

Dato un punto Q ed un vettore $\boldsymbol{v}$, esiste un unico punto P tale che $P - Q = \boldsymbol{v}$.

Si definisce quindi una somma tra un punto ed un vettore il cui risultato e un punto: $P = Q + \boldsymbol{v}$
Attenzione: non ho sommato Q da entrambe le parti dell'equazione precedente.
L'interpretazione geometrica è immediata; i punti sono locazioni nello spazio e la differenza di due punti e data dal vettore che li congiunge; è importante non confondere punti e vettori, sono entità geometriche ben distinte.

4.4 Combinazioni affini

Non è definita una somma tra punti e neppure un prodotto di uno scalare per un punto; in generale sono operazioni non lecite, ma c'è una eccezione.
Si prendano tre punti P, Q ed O e si consideri il seguente punto:

$$P' = \alpha(P - O) + \beta(Q - O) + O$$

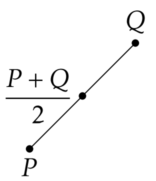
P' non dipende da O, ma solo dai punti P e Q, se e solo se $\alpha + \beta = 1$
In questo caso P' è la **combinazione affine** di P e Q, e si scrive, a volte in modo improprio, come **somma pesata dei punti**.
La combinazione affine di due punti distinti descrive la retta passante per i due punti.
La combinazione affine si estende in modo naturale a n punti.

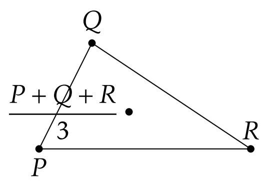
$$P' = \sum_i \alpha_i P_i, \quad \sum_i \alpha_i = 1 \;\; \alpha_i \in \mathbb{R}$$

Un insieme di punti si dice **affinemente indipendente** se *nessun punto è combinazione affine degli altri*.

4.4.1 Combinazione Convessa

La combinazione convessa è una combinazione affine con pesi positivi.
Nel caso della combinazione convessa di due punti, il punto risultante giace sul segmento che congiunge i due punti. Se i pesi sono entrambi pari a 0.5, il punto risultante si trova a metà tra i due.





Nel caso di n punti che formano un poligono convesso, il punto risultante si trova all'interno del poligono. Se tutti i pesi sono uguali a $1/n$, il punto risultante si chiama **centroide** dell'insieme dei punti.

4.4.2 Guscio Convesso

Un insieme $C \in \mathbb{R}^n$ è convesso se per ogni coppia di punti $P_1, P_2 \in C$ si ha che $P' = \alpha(P_1 - P_2) + P_2$ appartiene a C per ogni $\alpha \in [0, 1]$ ovvero tutti i punti sul segmento che unisce P_1 con P_2 appartengono all'insieme C.
Il guscio convesso (*convex hull*) di un insieme di punti è la più piccola regione convessa che contiene tutti i punti dati.

4.5 Prodotto interno

In uno spazio affine non è ancora definito il concetto di distanza o di angolo tra vettori; questi si ottengono passando ad uno spazio euclideo che è uno spazio affine provvisto di un **prodotto interno tra vettori** (***prodotto scalare***) che è definito come:

Dati due vettori $a = [a_i]_1^n$ e $b = [b_i]_1^n$ di $\mathbb{R}^n$

$$\sum_1^n a_i b_i = \begin{bmatrix} a_1 & \dots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = a'b$$

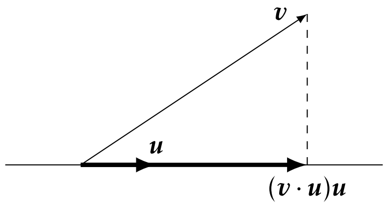
che soddisfa le seguenti relazioni:

$$\begin{aligned} \boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{v} \cdot \boldsymbol{u} \in S \\ (\alpha \boldsymbol{u} + \beta \boldsymbol{v}) \cdot \boldsymbol{w} = \alpha \boldsymbol{u} \cdot \boldsymbol{w} + \beta \boldsymbol{v} \cdot \boldsymbol{w} \\ \boldsymbol{v} \cdot \boldsymbol{v} > 0 \ (\boldsymbol{v} \neq \boldsymbol{0}) \\ \boldsymbol{0} \cdot \boldsymbol{0} = 0 \end{aligned}$$

Se il prodotto interno di due vettori è nullo, diremo che i **due vettori sono ortogonali**. Grazie al prodotto interno e possibile definire la **lunghezza di un vettore** (e quindi la distanza tra due punti) e l'**angolo** tra due vettori.
Norma di un vettore:

$$||\boldsymbol{v}|| = \sqrt{\boldsymbol{v} \cdot \boldsymbol{v}} \quad \cos\theta = \frac{\boldsymbol{v} \cdot \boldsymbol{u}}{||\boldsymbol{v}|| \, ||\boldsymbol{u}||}$$

Il prodotto scalare può essere usato, ad esempio, per trovare la proiezione di un vettore lungo una retta. Sia dato il vettore $\boldsymbol{v}$ e la retta con direzione identificata dal vettore di lunghezza unitaria u; il vettore ottenuto proiettando v lungo la retta sarà della forma $\boldsymbol{v}' = t\boldsymbol{u}$ dove t è un parametro, si può dimostrare che $t = \boldsymbol{v} \cdot \boldsymbol{u}$



4.6 Normalizzazione

Un vettore è normalizzato se la sua lunghezza è 1; dato un vettore qualsiasi lo si può normalizzare moltiplicandolo per il reciproco della sua lunghezza.
*Un vettore normalizzato si dice anche **versore***

Una base è **ortonormale** se e formata da versori a due a due ortogonali:

$$(e_1, \dots, e_n) : ||e_i|| = 1 \ \forall i \quad \text{e} \quad e_i \cdot e_j = 0 \quad \forall i \neq j$$

Data una base ortonormale il prodotto interno tra due vettori si esprime come somma dei prodotti delle componenti (usuale prodotto scalare di vettori)

$$\boldsymbol{v}\boldsymbol{w} = v_1 w_1 + \dots + v_n w_n$$

data una base qualsiasi è sempre possibile derivare da essa una base ortonormale (procedimento di *Gram-Schmidt*)

4.7 Terne

In tre dimensioni una base ortonormale si dice **destrorsa**, se la rotazione attorno ad $\boldsymbol{e}_3$ che porta $\boldsymbol{e}_1$ a coincidere con $\boldsymbol{e}_2$ è antioraria se vista dalla parte positiva di $\boldsymbol{e}_3$. Se tale rotazione è oraria allora la base è **sinistrorsa**.
Si può usare la *prima regola della mano destra*: se si pone il pollice nella direzione di $\boldsymbol{e}_3$, la rotazione che porta $\boldsymbol{e}_1$ in $\boldsymbol{e}_2$ deve seguire il modo naturale con cui si piegano le altre dita.
Oppure la *seconda regola della mano destra* per determinare la destrorsità: se si riesce a porre i tre vettori di base in corrispondenza con pollice, indice e medio della mano destra, tenuti perpendicolari l'uno all'altro, la base e destrorsa. La scelta di un orientamento è del tutto arbitraria, basta essere coerenti. Di norma si usano basi destrorse.

4.8 Sistemi di riferimento (frame)

Il concetto di base si estende a quello di riferimento in uno spazio affine (o euclideo) specificando, oltre alla base, anche un punto O detto origine del riferimento.
Poiché ogni vettore è sviluppabile in una base data ed ogni punto esprimibile come somma di un punto dato e di un vettore, dato un riferimento $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, O)$, i punti ed i vettori dello spazio saranno esprimibili nel seguente modo:

$$\boldsymbol{v} = v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + v_3\boldsymbol{e}_3$$
$$P = p_1\boldsymbol{e}_1 + p_2\boldsymbol{e}_2 + p_3\boldsymbol{e}_3 + O$$

Un riferimento **cartesiano** è dato da un riferimento la cui base di vettori sia ortonormale.
Un riferimento è destrorso se lo è la sua base.

4.9 Coordinate omogenee

Definiamo il prodotto di un punto per 1 e per 0: $P \cdot 1 = P \quad P \cdot 0 = 0$.

In questo modo possiamo definire le coordinate omogenee di un punto e di un vettore rispetto al riferimento $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, O)$.

$$\boldsymbol{v} = (v_1, v_2, v_3, 0)$$
$$P = (p_1, p_2, p_3, 1)$$

La scelta di 0 e 1 come ultima coordinata per vettori e punti è arbitraria, andrebbe bene qualsiasi valore:
Tale scelta però permette il **type checking**: si trattano le 4-ple delle coordinate omogenee come vettori quando si effettua una qualsiasi combinazione lineare di punti e vettori, usando le usuali regole, se l'ultima coordinata del risultato è 0, allora il risultato è un vettore; se è pari a 1 allora il risultato è un punto!
Se non è ne 0 ne 1, allora si è effettuata una operazione non lecita

4.10 Riepilogo

- Gli scalari sono numeri reali
- I vettori identificano direzioni nello spazio
- I punti determinano posizioni nello spazio
- Operazioni ammesse: somma e prodotto tra scalari, prodotto di scalari per vettori, somma di vettori, differenza di punti, somma di un punto con un un vettore, combinazioni affini.
- Il prodotto scalare permette di determinare la lunghezza dei vettori, la distanza tra punti e l'angolo tra due vettori
- Conviene lavorare in una base ortonormale; in questo caso il prodotto scalare tra due vettori e particolarmente semplice
- I tre assi che formano la base si chiamano assi coordinati e si indicano con x, y e z (a volte useremo anche 1, 2 e 3).

4.11 Prodotto vettore

Nel caso particolare delle tre dimensioni è utile introdurre un'ulteriore operazione tra vettori: **il prodotto vettore**.
Si tratta di un caso particolare di prodotto denominato **esterno**; in tre dimensioni particolarmente semplice:

$$\boldsymbol{u} \times \boldsymbol{v} = (u_y v_z - u_y v_y,\ u_z v_x - u_x v_z,\ u_x v_y - u_y v_x)$$

Si dimostra che il prodotto vettore di due vettori $\boldsymbol{u}$ e $\boldsymbol{v}$ è un vettore ortogonale al piano contenente i due vettori e di modulo pari all'area definita da $\boldsymbol{u}$ e $\boldsymbol{v}$. Il verso è scelto in modo tale che $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u} \times \boldsymbol{v})$ formino una terna destrorsa.
Attenzione: il prodotto vettore (a differenza delle proprietà affini dello spazio) dipende dalla scelta del tipo di base, destrorsa o sin.

Esempio: Data una direzione espressa dal vettore unitario $\boldsymbol{v}$, voglio creare un sistema di riferimento ortogonale con l'asse z coincidente con $\boldsymbol{v}$. Come faccio?

- Prendo un qualunque vettore $\boldsymbol{a}$ non parallelo a $\boldsymbol{v}$.
- Prendo la direzione dell'asse x $\boldsymbol{e}_1$ uguale a $\boldsymbol{v} \times \boldsymbol{a}$.
- Prendo la direzione dell'asse y $\boldsymbol{e}_2$ uguale a $\boldsymbol{v} \times \boldsymbol{e}_1$.

4.12 Matrici e trasformazioni

Una matrice è essenzialmente un array bidimensionale di elementi; per i nostri scopi gli elementi saranno sempre degli scalari, tipicamente numeri reali.

Una matrice A puo essere moltiplicata per uno scalare β ottenendo una matrice $C = \beta A$ definita nel seguente modo:

$$c_{ij} = \beta a_{ij} \quad \forall i, j$$

Due matrici A e B si possono sommare se e solo se hanno lo stesso numero di righe e di colonne; in tal caso si ha $C = A + B$ data da:

$$c_{ij} = a_{ij} + b_{ij} \quad \forall i, j$$

Il prodotto tra matrici e definito solo quando il numero di colonne della prima matrice è uguale al numero di righe della seconda. Se A è una matrice $N \times M$ e B e una matrice $M \times K$, allora si ha $C = AB$ (di dimensioni $N \times K$) data da:

$$c_{ij} = \sum_{l=1}^{M} a_{il} b_{lj}$$

Il prodotto tra matrici è **associativo** ($(AB)C = A(BC)$), ma **non commutativo** (in generale $AB \neq BA$)

4.13 Matrice trasposta

Indicata con il simbolo A_T , è la matrice ottenuta scambiando le righe con le colonne di A:

$$a_{ij}^T = a_{ji}$$

Quindi se A è $N \times M$, allora la sua trasposta è $M \times N$.
Per i vettori trasporre equivale a trasformare un vettore riga in un vettore colonna e viceversa D'ora in poi quando parleremo di **trasformazione** di un vettore $\boldsymbol{v}$ con una matrice A intenderemo sempre l'usuale *prodotto di matrici* tra A e il trasposto di $\boldsymbol{v}$ inteso come matrice con una sola colonna, es.:

$$A\boldsymbol{v} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} a_{11}v_1 + a_{12}v_2 \\ a_{21}v_1 + a_{22}v_2 \end{pmatrix}$$

4.14 Determinante

Importante parametro per le matrici quadrate, indicato con il simbolo $detA$ o con il simbolo $|A|$. Si definisce ricorsivamente:

Il determinante di una matrice $2x2$ è definito da:

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

Il determinante di una matrice $N \times N$ è dato dalla formula:

$$detA = \sum_{j=1}^{N} (-1)^{j+k} a_{jk} detA_{jk}$$

dove k e una colonna qualsiasi di A e dove il simbolo A_{jk} indica la matrice $(N-1)\times(N-1)$ ottenuta da A eliminando la riga j e la colonna k. Si può dimostrare che $det(AB) = detA\, detB$
Si può dimostrare che una matrice e invertibile se e solo se il suo determinante è diverso da 0; in tal caso si ha:

$$a_{ij}^{-1} = (-1)^{i+j} \frac{\det A_{ij}}{\det A}$$

4.15 Matrici: trasformazioni e cambiamento di base

Abbiamo visto cosa significa applicare una matrice ad un vettore

- Le matrici quadrate rappresentano quindi delle **applicazioni lineari** di uno spazio vettoriale in sé (formano un gruppo non abeliano).
- Tutte le applicazioni lineari di uno spazio vettoriale in sé sono esprimibili tramite **matrici quadrate**.
- L'applicazione di più di una matrice ad un vettore si effettua sfruttando l'algebra delle matrici; ad esempio applicare prima A, poi B ed infine C equivale ad applicare la matrice CBA.

Abbiamo detto che dato uno spazio vettoriale esistono infinite basi. Nella rappresentazione concreta il **cambiamento da una base ad un'altra** è descritto da una matrice.
In generale dato un vettore (v_1, v_2, v_3), la sua trasformazione in (v_1', v_2', v_3') tramite la matrice M puo essere vista come :

- Una trasformazione identificata da M del vettore fissata la base (**trasformazione attiva**).
- Un cambiamento di base indotto dalla matrice M^{-1} tenendo fisso il vettore (**trasformazione passiva**).

4.16 Cambio di riferimento

L'idea si ripropone negli stessi termini per i sistemi di riferimento.
Dati due riferimenti $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, O)$ e $(\boldsymbol{e}_1', \boldsymbol{e}_2', \boldsymbol{e}_3', O)$ si tratta di trovare una matrice 4×4 che permetta di ottenere le coordinate di un punto rispetto al secondo riferimento date le coordinate dello stesso punto rispetto al primo.

Come nel caso dei cambiamenti di base di un riferimento, se T è la trasformazione attiva che manda il primo riferimento nel secondo (e che manda le coordinate rispetto al secondo nelle coordinate rispetto al primo), allora T_{-1} e la matrice che trasforma le coordinate rispetto al primo riferimento nelle coordinate rispetto al secondo riferimento.

Esempio:
Determinare la rotazione che porta gli assi canonici $\boldsymbol{e}_1 = (1,0,0)$, $\boldsymbol{e}_2 = (0,1,0)$, $\boldsymbol{e}_3 = (0,0,1)$ in una qualunque terna

$$\boldsymbol{e}_1' = (e_{11}', e_{12}', e_{13}'),$$
$$\boldsymbol{e}_2' = (e_{21}', e_{22}', e_{23}'),$$
$$\boldsymbol{e}_3' = (e_{31}', e_{32}', e_{33}')$$

La matrice di rotazione è data da:

$$\begin{pmatrix} \boldsymbol{e}_1\boldsymbol{e}_1' & \boldsymbol{e}_1\boldsymbol{e}_2' & \boldsymbol{e}_1\boldsymbol{e}_3' \\ \boldsymbol{e}_2\boldsymbol{e}_1' & \boldsymbol{e}_2\boldsymbol{e}_2' & \boldsymbol{e}_2\boldsymbol{e}_3' \\ \boldsymbol{e}_3\boldsymbol{e}_1' & \boldsymbol{e}_3\boldsymbol{e}_2' & \boldsymbol{e}_3\boldsymbol{e}_3' \end{pmatrix}$$

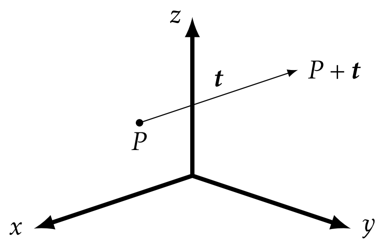
4.17 Traslazione

Una traslazione determinata dal vettore $\boldsymbol{t}$ trasforma il punto P nel punto:

$$P' = P + \boldsymbol{t}$$

In termini di componenti:

$$\begin{aligned} \boldsymbol{t} &= (t_x, t_y, t_z, 0) \\ P &= (p_x, p_y, p_z, 1) \\ P' &= (p_x + t_x,\ p_y + t_y,\ p_z + t_z,\ 1) \end{aligned}$$

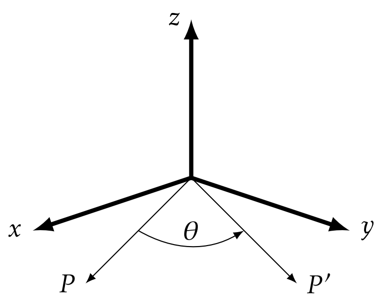


E' facile vedere che la matrice di trasformazione T t per le coordinate omogenee è:

$$T_t = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4.18 Rotazione

Una rotazione di un angolo θ in senso **antiorario** (prima regola della mano destra) **intorno all'asse** z determina la seguente trasformazione di un punto P in P'.



$$\begin{aligned} p'_x &= p_x \cos(\theta) - p_y \sin(\theta) \\ p'_y &= p_x \sin(\theta) + p_y \cos(\theta) \\ p'_z &= p_z \end{aligned}$$

Si può facilmente dimostrare che per rotazioni intorno all'asse x e y si hanno le seguenti espressioni:

$$\begin{aligned} p'_y &= p_y \cos(\theta) - p_z \sin(\theta) \\ p'_z &= p_y \sin(\theta) - p_z \cos(\theta) \\ p'_x &= p_x \end{aligned} \qquad \begin{aligned} p'_z &= p_z \cos(\theta) - p_x \sin(\theta) \\ p'_x &= p_z \sin(\theta) - p_x \cos(\theta) \\ p'_y &= p_y \end{aligned}$$

Matrici che rappresentano le rotazioni rispetto agli assi coordinati:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Osservazioni:
Da notare che un *vettore viene trasformato da una rotazione* (a differenza delle traslazioni che lasciano i vettori inalterati). *Le matrici non commutano.*
Le rotazioni rispetto agli assi cartesiani non commutano; provare a ruotare un oggetto di 90 gradi prima rispetto all'asse x e poi rispetto all'asse y. Ripetete quindi l'operazione prima rispetto all'asse y e poi rispetto all'asse x. Risultato?
Da notare che le rotazioni lasciano inalterati i punti che si trovano sull'asse di rotazione.

Si può dimostrare che $R_x(\theta)^{-1} = R_x(-\theta)$ e similmente per gli altri assi.

Si può dimostrare che le matrici di rotazione date sopra sono **ortogonali**,
ad es. per l'asse x: $R_x(\theta)^{-1} = R_x(-\theta)^T$

La proprietà di ortogonalità è vera per ogni rotazione, non solo per quelle rispetto agli assi coordinati.
Tutte le rotazioni sono esprimibili con matrici.

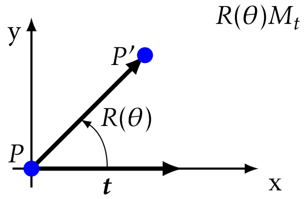
4.19 Composizione di trasformazioni di matrici

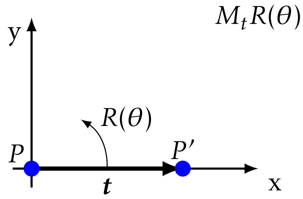
Le trasformazioni espresse come matrici si compongono usando semplicemente l'algebra delle matrici. Date due trasformazioni rappresentate dalle matrici A e B, la composizione di A seguita da B sarà data dalla matrice BA.
Importante: notare l'ordine delle matrici; siccome si applica la matrice risultante a sinistra del vettore delle coordinate omogenee, la trasformazione che viene effettuata per prima va a destra. La composizione di trasformazione si estende immediatamente al caso di piu di due matrici: $T = T_n \dots T_1$

4.20 Non commutatività

Esempio: data una traslazione lungo il vettore $\boldsymbol{t}$ ed una rotazione di un angolo lungo l'asse $\boldsymbol{z}$, si ottiene un risultato diverso effettuando prima la rotazione e poi la traslazione o viceversa.

Per rendersene conto basta guardare come viene trasformato nei due casi un punto che in partenza si trova nell'origine.





4.21 Scalatura

Traslazioni e rotazioni conservano la lunghezza dei vettori e sono un sottogruppo delle trasformazioni affini chiamato trasformazioni isometriche o rigide. Un altro tipo di trasformazione affine che non preserva le distanze è la **scalatura**.
Dato un punto $P = (p_x, p_y, p_z, 1)$ la trasformazione di scala, o scalatura, lo trasforma nel punto $P' = (s_x p_x, s_y p_y, s_z p_z, 1)$ dove i valori (s_x, s_y, s_z) sono i fattori di scala lungo gli assi.
Una scalatura è *omogenea* se $s_x = s_y = s_z = s$

- vettori semplicemente allungati ($s > 1$) o accorciati ($s < 1$).
- Un punto, in una scalatura omogenea, viene invece traslato lungo la retta che passa per l'origine e per il punto stesso.

4.22 Trasformazioni affini

Una generica matrice che lavora in coordinate omogenee rappresenta una trasformazione affine (12 gradi di libertà non solo traslazione rotazione e scala, ma anche shear)

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Se t è il vettore di traslazione e R una matrice di rotazione, la trasformazione in coordinate omogenee è:

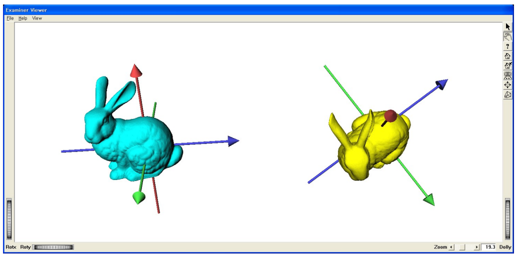
$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

4.23 Rotazioni generiche e orientazione

Dobbiamo considerare rotazioni attorno a qualunque asse. Comunque, non c'è nessuna perdita di generalità nel definirle solo attorno agli assi passanti per l'origine, dato che le altre le posso ricavare traslando l'origine sull'asse, ruotando e ritraslando l'origine all'indietro.

La rotazione rappresenta un cambio di orientazione L'orientazione rappresenta la posa di un oggetto nello spazio La relazione che c'è tra rotazione (movimento) e orientazione (stato) è analoga a quella tra punto e vettore Anche per le operazioni:

- orientazione+rotazione=orientazione
- rotazione+rotazione=rotazione



point : the 3d location of the bunny
vector : translational movement
orientation: the 3d orientation of the bunny
rotation : circular movement

4.24 Teorema della rotazione di Eulero

The general displacement of a rigid body with one point fixed is a rotation about some axis.
Qualsiasi rotazione si può esprimere come rotazione di un angolo rispetto a un asse.
Qualsiasi rotazione lascia invariati un vettore invariato (l'asse).
Una rotazione qualsiasi rispetto ad un asse passante per l'origine puo essere decomposta nel prodotto di tre rotazioni rispetto agli assi coordinati; i tre angoli prendono il nome di **angoli di Eulero**.
La rappresentazione con gli angoli di Eulero non è univoca, a terne diverse può corrispondere la stessa trasformazione.
Una delle rappresentazioni di Eulero impiega gli angoli roll (rollio), pitch (beccheggio) e yaw (imbardata), di derivazione aeronautica.

4.25 Problemi con angoli Eulero

Ci sono alcuni problemi con le rappresentazioni delle rotazioni

- Angoli di Eulero: Rotazioni non univoche:

  $$(z,x,y)[roll,yaw,pitch] = (90,45,45) = (45,0,-45)$$

  mandano entrambi l'asse x in direzione $(1,1,1)$
- Gimbal Lock (blocco del giroscopio)
  - Gimbal è un dispositivo meccanico usato per supportare giroscopi o bussole
  - Ci sono configurazioni problematiche
- Interpolazione di rotazioni: Come calcoliamo il punto medio di una rotazione?

4.26 Rotazione asse angolo

La rotazione generica asse angolo con asse r si può anche rappresentare con la seguente matrice, dove $c = \cos(\alpha)$ e $s = \sin(\alpha)$:

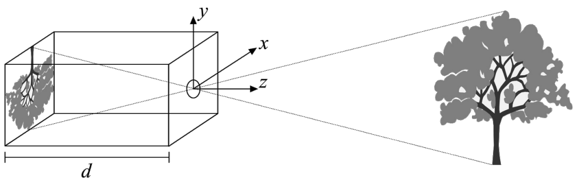
$$R(\alpha,\boldsymbol{r}) = \begin{pmatrix} (1-c)r_1^2+c & (1-c)r_1r_2-sr_3 & (1-c)r_1r_2+sr_2 & 0 \\ (1-c)r_1r_2+sr_3 & (1-c)r_2^2+c & (1-c)r_2r_2-sr_1 & 0 \\ (1-c)r_1r_3-sr_2 & (1-c)r_2r_3+sr_1 & (1-c)r_3^2+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4.27 Matrici e proiezioni

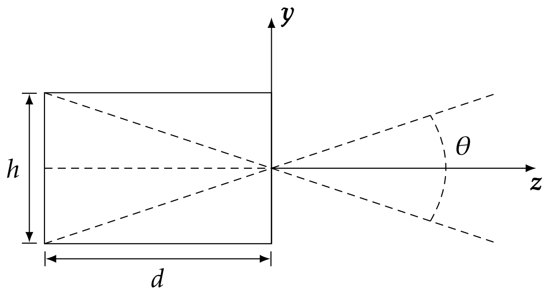
- Abbiamo il nostro mondo dove creare la scena inserendo i modelli: spazio Euclideo.
- Sappiamo trasformare i punti dello spazio traslando, ruotando e scalando.
- Per simulare la formazione delle immagini ci serve un ultimo strumento geometrico: la modellazione della proiezione degli oggetti sul piano immagine.
- Questo si fa con la proiezione prospettica o, in casi semplificati, con la proiezione ortografica o parallela
- Anche queste si possono modellare con matrici, solo che dovranno trasformare uno spazio 3D in uno 2D (espressi in coordinate omogenee). Quindi sono matrici.

4.28 La macchina fotografica virtuale

La metafora utilizzata per descrivere le relazioni scena/osservatore è quella della macchina fotografica virtuale (synthetic camera).
Il modello semplice usato anche in Computer Vision è la **telecamera pinhole**.



La macchina fotografica virtuale è costituita da un parallelepipedo in cui la faccia anteriore presenta un foro di dimensioni infinitesime (pinhole camera) e sulla faccia posteriore si formano le immagini.



Immagini nitide, nessun problema di luminosità, l'angolo di vista può essere modificato variando il rapporto tra la distanza focale (d) e la dimensione del piano immagine.
Per convenzione (e maggiore semplicità) si assume l'esistenza di un piano immagine tra la scena ed il centro di proiezione
Ne risulta il modello matematico della proiezione prospettica.

4.29 Proiezione prospettica

La relazione che lega i punti 3D ai punti sul piano in questa ipotesi è data dalla proiezione prospettica. Con semplici ragionamenti sui triangoli simili si ha che la proiezione di un punto $P = (P_x, P_y, P_z)$ è data da:

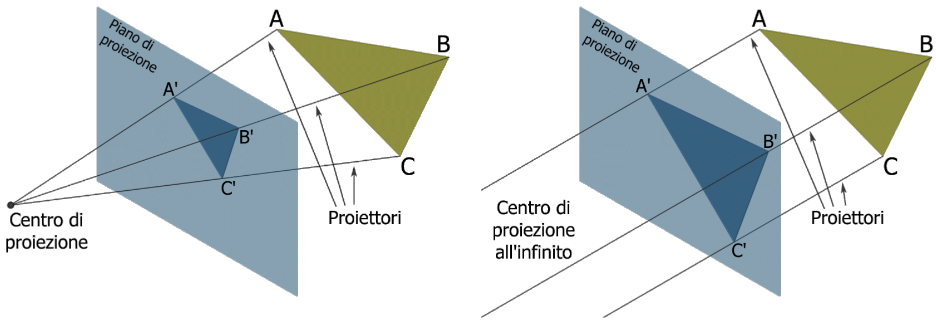
- $P' = (-\frac{P_x d}{P_z}, -\frac{P_y d}{P_z}, 1)$ per il piano immagine dietro.
- $P' = (\frac{P_x d}{P_z}, \frac{P_y d}{P_z}, 1)$, per il piano immagine davanti.

Possiamo scrivere la proiezione in forma matriciale:

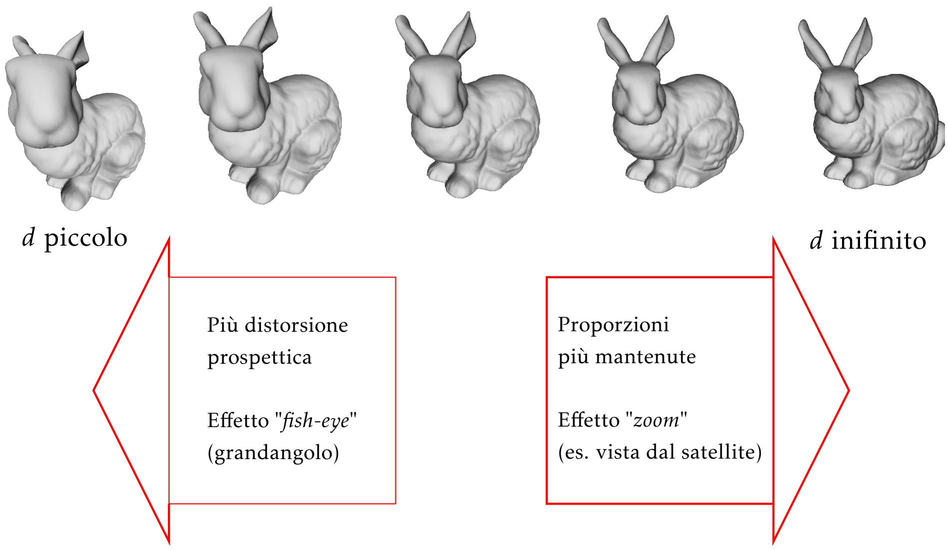
$$P' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

Da un punto di vista geometrico, la proiezione è definita per mezzo di un insieme di rette di proiezione (i proiettori) aventi origine comune in un centro di proiezione, passanti per tutti i punti dell'oggetto da proiettare ed intersecanti un piano di proiezione.
La proiezione di un segmento è a sua volta un segmento. Non è quindi necessario calcolare i proiettori di tutti i punti di una scena, ma solo quelli relativi ai vertici delle primitive che la descrivono.
Le proiezioni geometriche piane si classificano in:

- Proiezioni **prospettiche** (distanza finita tra il centro ed il piano di proiezione)
- Proiezioni **parallele** (distanza infinita tra il centro ed il piano di proiezione)



Al variare della distanza focale d si ha che:



Per motivi che capiremo, in grafica si usa in realtà rappresentare la proiezione prospettica con una trasformazione che mappa comunque sullo spazio 3D, quindi una matrice 4x4.
Per passare alla rappresentazione 2D basta poi eliminare la z (che è sempre uguale a d)

$$\boldsymbol{P'} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

In realtà la coordinata omogenea 2D che ricavo dall'applicazione della matrice è

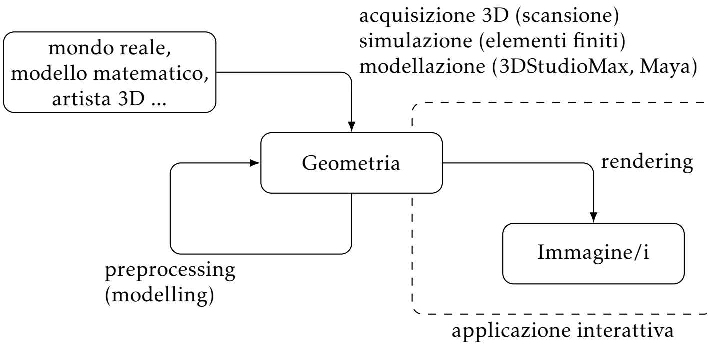
$$P' = (P_x, P_y, P_z/d)$$

L'operazione che trasforma in

$$P' = (\frac{P_x d}{P_z}, \frac{P_y d}{P_z}, 1)$$

per avere la forma standard dei punti è la cosiddetta divisione.
Prima della divisione i tre valori possono essere usati per rappresentare l'equivalenza dei diversi punti rispetto alla proiezione.
Definiamo ora possibili strutture dati per modellare gli oggetti nello spazio.
Poi vedremo come modellare anche la formazione delle immagini attraverso il "rendering".

5 Modellare gli oggetti nello spazio



5.1 Geometria Analitica

Premessa: prima di vedere come si modellano gli oggetti, ricordiamo come si definiscono nello spazio Euclideo 3D le figure geometriche importanti dal punto di vista della modellazione grafica e del rendering.
Rette: sono identificabili da un punto qualsiasi Q che giaccia sulla retta e da una direzione data da un versore u. È facile vedere che sono il luogo dei punti dati da $P = Q + t\boldsymbol{u} \quad t \in \mathbb{R}$
In termini di componenti si vede facilmente che vale la seguente equazione

$$\frac{x - x_Q}{u_x} = \frac{y - y_Q}{u_y} = \frac{z - z_Q}{u_z}$$

Se si vuole specificare una retta dati due punti R e Q, basta usare le formule date qui sopra tenendo conto che il versore che identica la retta e dato da: $\boldsymbol{u} = \dfrac{(R - Q)}{|R - Q|}$
Semiretta: basta aggiungere il vincolo $t \geq 0$
Segmenti: dati i punti iniziale e finale P e Q possiamo scriverli come $P = Q + t(R - Q) \quad t \in [0;1]$
Sfere: dato centro O e raggio r, i punti della superficie sferica sono dati dall'equazione $P = O + r\boldsymbol{u}$ con $\boldsymbol{u}$ versore generico. Si dimostra facilmente che, in termini delle coordinate, la superficie sferica è data dall'equazione

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$$

Piani: dati 3 punti non allineati P, Q ed R il luogo dei punti che descrive il piano che li comprende è la combinazione affine

$$S = \alpha P + \beta Q + \gamma R \quad \alpha, \beta, \gamma \in \mathbb{R} \quad \alpha + \beta + \gamma = 1$$

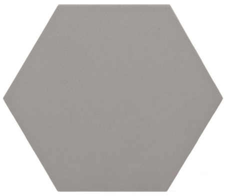
Alternativamente si puo definire un piano a partire da un punto Q che vi appartiene e da un vettore $\boldsymbol{u}$ che ne identica la normale come il luogo dei punti P tali che $(P-Q)\boldsymbol{u} = 0$. In termini di coordinate abbiamo

$$(x-x')u_x + (y-y')u_y + (z-z')u_z = 0$$

Per passare dalla prima alla seconda rappresentazione basta prendere come punto Q e come vettore $\boldsymbol{u} = (P-Q) \times (R-Q)$
Semispazi: il piano di cui sopra identica due semispazi, uno positivo ed uno negativo:

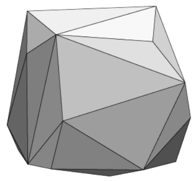
$$(P-Q)\boldsymbol{u} > 0$$
$$(P-Q)\boldsymbol{u} < 0$$

5.2 Poligoni



Un poligono P è un insieme finito di segmenti (spigoli) di $\mathbb{R}^2$, in cui ogni estremo (vertice) e comune a esattamente due segmenti, che si dicono adiacenti.
Un poligono è detto semplice se ogni coppia di spigoli non adiacenti ha intersezione vuota.
Teorema di Jordan: Un poligono semplice P divide il piano in due regioni o facce, una limitata (detta interno di P) ed una illimitata (detta esterno di P).
Per convenzione, un poligono viene rappresentato dalla sequenza dei suoi vertici $P_1 \dots P_n$ ordinati in modo che l'interno del poligono giaccia alla sinistra della retta orientata da P_i a P_{i+1} , ovvero i vertici sono ordinati in senso antiorario.

5.3 Poliedri



In $\mathbb{R}^3$ un poliedro semplice e denito da un insieme finito di poligoni (facce) tali che ciascuno spigolo di una faccia e condiviso da esattamente un'altra faccia e le facce non si intersecano che negli spigoli.

5.4 Rappresentazione degli oggetti

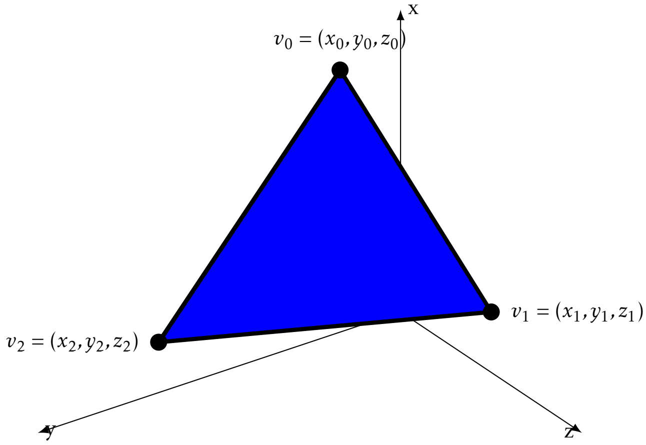
Gli oggetti che si vogliono rappresentare in una applicazione grafica hanno di solito caratteristiche particolari

- Sono finiti
- Sono chiusi (non sempre)
- Sono continui

Le rappresentazioni di oggetti (regioni dello spazio, in generale) si suddividono in

- basate sul **contorno** (boundary): descrivono una regione in termini della superficie che a delimita (boundary representation, o b-rep).
- basate sullo **spazio occupato** (o volumetriche).

La rappresentazione più comune è quella di **maglie(mesh) di triangoli**:

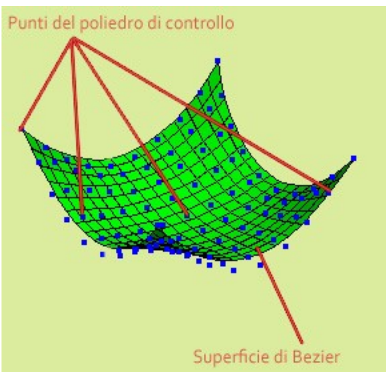


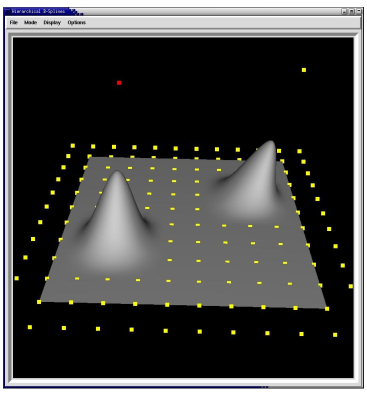
Esistono però delle alternative:

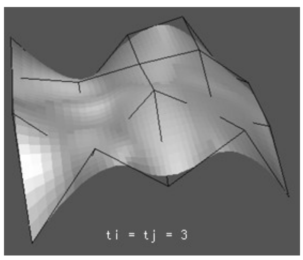
- **Boundary**: Superfici parametriche (lisce, non hanno problemi di "tessellazione" cioè visibilità degli spigoli tra le facce)
- **Volumetriche**:
  - Rappresentazione "voxellizata" (a cubetti)
  - Geometria costruttiva solida
- **Image based rendering**:
  - Non si modella effettivamente la scena, ma si memorizzano campionamenti della luce, renedendo però possibile una visualizzazione da più punti di vista, interattiva
  - Light fields

Il vantaggio principale nell'uso di superfici per modellare un oggetto sarebbe l'assenza del problema della tessellazione visibile (cioè approssimo una superficie lisca coi triangoli, ma vedo poi i triangoli evidenti, effetto ridotto di solito con trucchi opportuni nel rendering)
L'uso di superfici parametriche risulta pesante per applicazioni in tempo reale; per lo più le superfici parametriche utilizzate in fase di modellazione o per rendering non interattivo.
Negli ultimi tempi le cose sono cambiate ed oggi cominciano ad apparire applicazioni di grafica avanzata che usano superfici curve anche in tempo reale.

Esempio: **curve/superfici di Bezier**: Dati N punti di controllo la curva la curva passa per il primo e l'ultimo e approssima gli altri con una funzione da essi dipendente. Con una griglia si generano superfici.







5.4.1 Geometria costruttiva solida (CSG)

Altra rappresentazione particolarmente adatta per il modeling (diffusa nel settore CAD), ma poco efficiente per il rendering è quella della **Geometria costruttiva solida (CSG)**.
Si tratta, essenzialmente, di costruire degli oggetti geometrici complessi a partire da modelli base con operazioni booleane
Operazioni:

- **Unione**: l'unione $A \cup B$ è l'insieme dei punti che appartengono ad almeno uno dei due solidi (or non esclusivo).
- **Differenza**: la differenza $A \setminus B$ è l'insieme dei punti che appartengono ad A, ma non a B.
- **Intersezione**: l'intersezione $A \cap B$ è l'insieme dei punti che appartengono ad A ed a B (and).
- Le operazioni CSG possono essere descritte tramite un albero (gerarchia). Ciascun nodo di un albero che non sia una foglia contiene una delle tre operazioni elementari $\cup$, $\cap$ o $\setminus$. Ciascuna foglia contiene una primitiva.

5.4.2 Acquisizione dal vero: point clouds

Un modo per generare modelli 3D per mondi virtuali è acquisire dal vero.
La scansione 3D, oggi tecnologia matura con diverse tecnologie Laser, luce strutturata, visione computazionale.
Gli scanner (esattamente come le macchine fotografiche in 2D) di fatto non acquisiscono un modello del mondo, ma campionano la geometria (con eventuali attributi, es. colore) in una serie di punti discreti: si parla di **nuvole di punti (point clouds)**

5.4.3 Partizionamento spaziale (voxel)

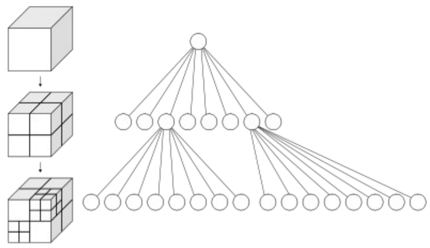
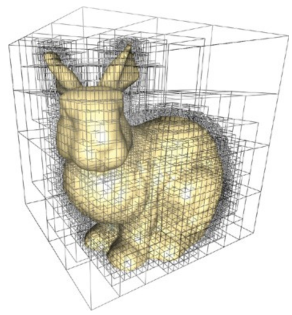
Lo spazio viene suddiviso in celle adiacenti dette in 3D voxel (equivalente dei pixel delle immagini): una cella è "piena" se ha intersezione non vuota con la regione, è detta vuota in caso contrario.
Oppure contiene un valore di densità (tipico dei dati diagnostici es. TAC)
Una rappresentazione di una scena complessa ad alta risoluzione richiederebbe l'impiego di un numero enorme numero di voxel, per cui questa rappresentazione è in genere limitata a singoli oggetti.
Ma le cose stanno cambiando grazie a progressi nell'HW e nel SW.
Rendering di modelli voxel-based:

- Algoritmi ad hoc (tecniche direct volume rendering)
- conversione da voxel a rappresentazione per superfici (triangle- based)

Da una rappresentazione volumetrica voxelizzata si può passare efficientemente a una rappresentazione poligonale della superficie mediante l'algoritmo detto marching cubes.

5.4.4 Rappresentazioni compatte(Octree)

Se ho solo i valori pieno/vuoto, posso rappresentare in modo compatto il volume con una struttura **octree**.
Si parte con un cubo contenente la regione e si suddivide ricorsivamente. Ci si ferma ogni volta che un ottante contiene tutte celle piene o tutte celle vuote.
Più economica rispetto alla enumerazione delle singole celle, poiché grandi aree uniformi (piene o vuote) vengono rappresentate con una sola foglia (anche se nel caso peggiore il numero delle foglie è pari a quello delle celle).

Le strutture di partizionamento spaziale sono anche utili per "contenere" le geometrie poligonali: come vedremo consente di rendere più semplice la ricerca di intersezioni tra oggetti e con i raggi ottici.

5.5 Maglie poligonali (mesh)

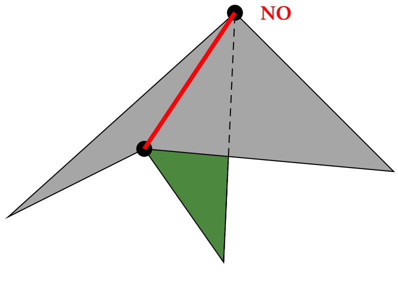
Nella grafica 3D interattiva si usa quasi sempre la modellazione basata su approssimazione poligonale degli oggetti (del loro contorno: è una *boundary representation*).
Si tratta di approssimare una superficie 2D con un insieme di poligoni convessi opportunamente connessi gli uni agli altri.
Nella pipeline di rendering si lavora in genere con i soli **triangoli**: tutte le altre rappresentazioni eventualmente usate nel programma sono convertite prima del rendering in triangoli Possiamo usare la geometria per definire rigorosamente le proprietà dei modelli triangolati.

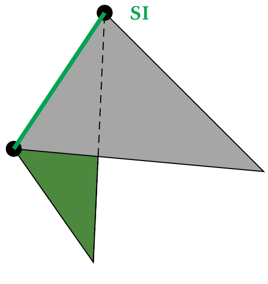
Una **varietà** k-dimensionale X è un sottoinsieme di $\mathbb{R}^d$ in cui ogni punto ha un intorno omeomorfo alla sfera aperta di $\mathbb{R}^k$.
In generale le superifici degli oggetti solidi (sfere, poliedri, ecc.) sono varietà bidimensionali.
Omeomorfismo: applicazione biiettiva, continua, con inversa continua. Intuizione: trasformazione senza "strappi". In una varietà k-dimensionale con bordo ogni punto ha un intorno omeomorfo alla sfera aperta o alla semisfera aperta di $\mathbb{R}^k$.
Il bordo di X è l'insieme dei punti che hanno un intorno omeomorfo alla semisfera aperta. Una varietà è sempre una varietà con bordo, eventualmente vuoto.
Il bordo, se non è vuoto, è a sua volta una varietà $k-1$ dimensionale senza bordo.

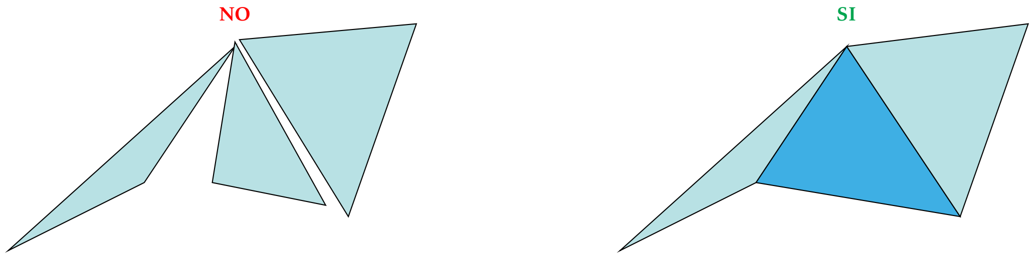
Una **maglia (mesh) triangolare** è un'insieme di triangoli le cui intersezioni siano esclusivamente vertici e lati (spigoli) dei triangoli e che sia anche una varietà bidimensionale con bordo. Due triangoli che condividono un lato si dicono adiacenti I triangoli della maglia si chiamano anche facce.
La condizione di essere varietà si traduce nei seguenti vincoli sulla struttura:

- uno spigolo appartiene al massimo a due triangoli (quelli eventuali che appartengono ad uno solo formano il bordo della maglia)
- se due triangoli incidono sullo stesso vertice allora devono essere raggiungibili l'uno dall'altro attraverso un percorso tra triangoli adiacenti ovvero devono formare un ventaglio o un ombrello.

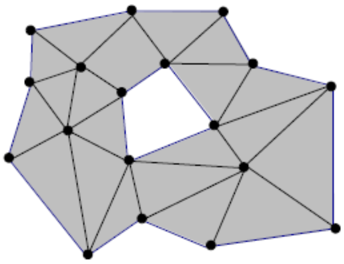
Si usa il termine condizione 2-manifold (varietà)

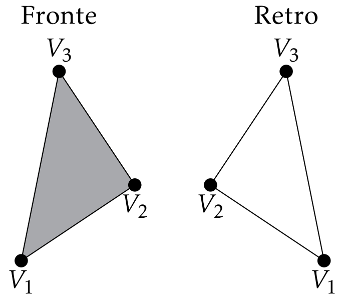






Il bordo della maglia consiste di uno o più anelli (sequenza chiusa di spigoli) o loop.
Se non esistono spigoli di bordo la maglia è chiusa (come quelle che rappresentano la superficie di una sfera).
L'**orientazione** di una faccia è data dall'ordine ciclico (orario o antiorario) dei suoi vertici incidenti. L'orientazione determina il fronte ed il retro della faccia. La convenzione (usata anche da OpenGL) è che la faccia mostra il fronte.





L'orientazione di due facce adiacenti è compatibile se i due vertici del loro spigolo in comune sono in ordine inverso. Vuol dire che l'orientazione non cambia attraversando lo spigolo in comune.
La maglia si dice **orientabile** se esiste una scelta dell'orientazione delle facce che rende compatibili tutte le coppie di facce adiacenti.

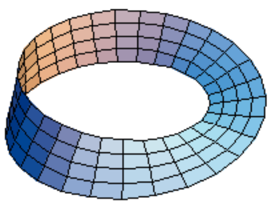


Figura 2: Non tutte le mesh 2-manifold sono orientabili (es. anello di Moebius)

Abbiamo definito la maglia triangolare. In maniera analoga si può estendere la definizione a maglie poligonali generiche
Maglie poligonali generiche: i poligoni possono avere qualsiasi numero di spigoli e non è detto che ci sia un solo tipo di poligono. Sono raramente utilizzate in grafica al calcolatore
Quadrangolari (quad meshes): gli elementi poligonali sono tutti quadrilateri. Sono alle volte usate, per esempio se si vuole fare il rendering di un terreno descritto da un array di altezze.
In una maglia quadrangolare bisogna imporre un vincolo aggiuntivo di planarità per ogni quadrilatero che la compone.
OpenGL consente di descrivere maglie poligonali generiche, ma per disegnarle li suddivide usualmente in triangoli.

5.5.1 Equazione di Eulero

Se V è il numero di vertici, L il numero di spigoli ed F il numero di facce della maglia poligonale orientabile chiusa di genere G, allora vale la Formula di Eulero $V - L + F = 2 - 2G$ Una superficie ha genere G se può essere tagliata lungo G linee semplici chiuse senza disconnetterla (intuitivamente, ma non rigorosamente "numero di buchi").

l genere di una superficie determina la sua topologia; per una sfera, per esempio, $G = 0$, mentre per un toro (una ciambella) $G = 1$.
Più in generale, per una maglia poligonale orientabile (e varietà bidimensionale) vale la formula $V - L + F = 2(S - G) - B$
S numero di componenti connesse, B è il numero di anelli di bordo.

5.6 Mesh di triangoli

Nella pratica sono il tipo di modello dominante, usato nella gran parte delle applicazioni interattive dato che il rendering è ottimizzato in hardware.
Generate da modellazione CAD, acquisizione con scanner, ricostruzione da immagini (Computer Vision).
Il numero di poligoni determina il dettaglio, ma il costo in memoria può essere notevole.

5.7 Costruzione della mesh triangolare

Conversione da altri formati:

- Poligoni $\longrightarrow$ Triangoli
- Superf. Quadriche $\longrightarrow$ Triangoli
- Campi di altezze o Punti $\longrightarrow$ Triangoli

Per rappresentare gli oggetti con le proprietà fisiche relativa a colore e riflessione della luce, devo abbinare alla geometria dei valori di proprietà (*attributi*)
Posso definirli:

- per vertice: esplicito un attributo per ogni vertice
- per faccia: esplicito un attributo per ogni faccia
- per wedge (vertice di faccia): esplicito tre attributi per ogni faccia

Attributi più comuni: Colore, Normali (versori perpendicolari), coordinate texture.
Non è sempre semplice modellare le entità da rappresentare con triangoli. Ad esempio: Nuvole, Fiamme, Capelli, pelliccia etc.

Quando si devono disegnare due triangoli con uno spigolo in comune, questo viene disegnato due volte. Questo introduce un certo grado di ridondanza che può incidere sulle prestazioni.
Si preferisce quindi raggruppare i triangoli di una maglia in opportuni gruppi che possono essere elaborati in maniere più efficiente. Si possono ad esempio utilizzare:

- **Fan di triangoli**: è un gruppo di triangoli che hanno in comune un vertice. Il primo viene specificato completamente, per i successivi basta dare il nuovo vertice. Efficiente, ma i triangoli che incidono su un vertice sono in genere pochi.
- **Strip di triangoli**: gruppo di triangoli che posseggono a due a due uno spigolo in comune. Di nuovo il primo triangolo viene specificato normalmente, per i successivi basta specificare il nuovo vertice. Meno efficiente, ma le strip in genere contengono più triangoli delle fan.

Esistono algoritmi per creare queste rappresentazioni dalle mesh.

5.8 Mesh e rendering

Per determinare l'effetto di una qualsiasi trasformazione affine su un oggetto (traslazione, rotazione, scalatura, composizioni varie di queste), basta applicare la trasformazione ai vertici (che sono punti); le informazioni connettive date dagli spigoli non cambiano in questo tipo di trasformazioni.
Questo rende piuttosto semplice il rendering di oggetti descritti in termini di maglie poligonali.

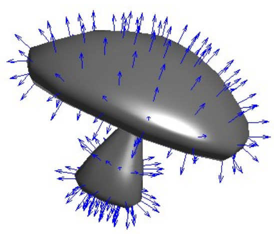
Qualsiasi trasformazione viene eseguita sui vertici, cioè si tratta di applicare trasformazioni affini su punti.
L'affermazione precedente è vera anche per la proiezione; per vedere come si proietta la forma di una maglia su un piano (l'immagine), basta seguire la proiezione dei vertici.

5.9 Memorizzazione

Alla creazione dei modelli vengono in genere generati nodi, connettività e attributi. Gli algoritmi di processing e rendering devono accedere in vario modo a tale informazione. Esistono diversi modi di rappresentare questa informazione.
Nei programmi applicativi sono quindi necessarie delle procedure per convertire una rappresentazione in un'altra.
Progettare ed implementare tali procedure è un ottimo modo per capire nel dettaglio le varie rappresentazioni utilizzate per descrivere maglie poligonali.
Nei disegni e negli esempi ci concentreremo sul caso di maglia triangolare, ma il discorso è valido in generale per tutti i poligoni convessi.

5.10 Elementi base

Vertici: sono gli elementi 0 dimensionali e sono identificabili con punti nello spazio 3D (essenzialmente tre coordinate); alle volte può essere utile associare ai vertici altre caratteristiche oltre alla posizione (tipo il colore).
Spigoli: sono elementi 1 dimensionali e rappresentano un segmento che unisce due vertici. Di solito non contengono altre informazioni.
Facce: sono i poligoni bidimensionali, formati da un certo numero di spigoli e di vertici (dimostrare che sono in numero uguale). I vertici o gli spigoli si usano per identificare la faccia; possono contenere altre informazioni (tipo il colore).
Normali: è fondamentale sapere quale è l'esterno della superficie e quale l'interno, e qual è l'orientazione locale della superficie; a tal scopo si associa spesso ad una maglia poligonale anche l'informazione sulla normale uscente.
La normale $\boldsymbol{n}$ ad una faccia è data dal prodotto vettore di due suoi spigoli consecutivi non collineari:
Attenzione al verso: la normale è uscente dal fronte della faccia
Per un triangolo (V_1, V_2, V_3) si ha: $\boldsymbol{n} = (V_3 - V_2) \times (V_1 - V_2)$.



Attenzione (vedremo in lab): ci servono le normali per calcolare il modello di Phong. Le possiamo calcolare per i modelli, cui poi sono applicate le trasformazioni geometriche e di proiezione.
Ma se applichiamo trasformazioni generiche ai modelli queste trasformazioni non preservano necessariamente l'ortogonalità di normale e superficie.
Per trasformare senza questo problema le normali possiamo trasformarle con una matrice diversa.

Consideriamo un vettore tangente $\boldsymbol{t}$ alla superficie:

$$\boldsymbol{n}\boldsymbol{t}^T = 0$$
$$\boldsymbol{n}M^{-1}M\boldsymbol{t} = 0$$
$$(M^{-1})^T\boldsymbol{n}M\boldsymbol{t} = 0$$

Quindi la normale trasformato da $(M^{-1})^T\boldsymbol{n}$ è perpendicolare alla superficie trasformata. Cioè per evitare problemi si trasformeranno le normali con l'inversa trasposta della matrice di trasformazione del modello.

5.11 Struttura della mesh

I vertici danno informazioni di tipo posizionale, gli spigoli informazioni di tipo connettivo (non c'è informazione spaziale).
Gli spigoli connettono i vertici, permettendo di introdurre un concetto di "vicinanza" tra vertici e dando le informazioni di tipo topologico (ovvero definiscono un grafo).
Le facce sono determinate una volta dati i vertici e gli spigoli, quindi non introducono nulla.
Al più possono avere associati attributi, anche se è raro.

Ci sono vari modi di conservare le informazioni sui modelli. Questi possono differire per

- Memoria occupata
- Complessità di implementazione di operazioni di ricerca (es. ricerche di adiacenza
  - Quali sono i vertici vicini a uno dato?
  - Quali sono gli spigoli che contengono un vertice dato?
  - Quali sono le facce che contengono uno spigolo?
  - ...

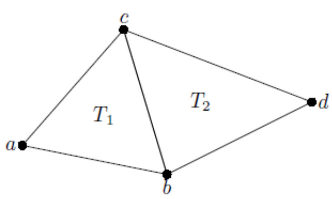
Possibilità di controllo sulla qualità delle superfici (es. manifoldness)

5.12 Lista di triangoli e indexed

- Immediata: specificare tutte le facce della maglia come terne di triplette di coordinate cartesiane
- Spreco di memoria: duplico le coordinate. Meglio usare una struttura indicizzata, con la lista dei vertici e la lista delle facce con i puntatori ai vertici (indici)
- Normalmente usate in openGL per il rendering
- Non ottimali per le ricerche

```
typedef struct{
    float v1[3];
    float v2[3];
    float v3[3];
} faccia;
```

```
typedef struct {
    float x,y,z;
} vertice;
typedef struct {
    vertice* v1,v2,v3;
} faccia;
```



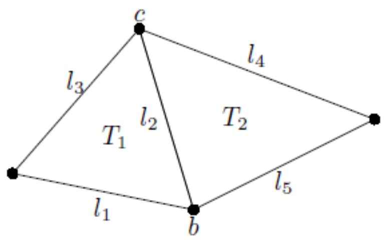
5.13 Winged edge (Baugmart 1975)

Si aggiungono dei puntatori allo spigolo per rendere più semplice l'analisi delle incidenze.
L'elemento base è lo spigolo (edge) con le sue due facce incidenti (wings):

- Lo spigolo l_2 contiene un puntatore ai due vertici su cui incide (b; c), alle due facce su cui incide (T_1, T_2) ed ai due spigoli uscenti da ciascun vertice
- Un vertice contiene un puntatore ad uno degli spigoli che incide su di esso, più le coordinate (ed altro)
- La faccia contiene un puntatore ad uno degli spigoli che vi incide (ed altro).

La struttura assume che ogni spigolo non di bordo abbia due facce incidenti (manifold).

```
typedef struct{
    we_vertice* v_ini, v_fin;
    we_spigolo* vi_sin, vi_dstr;
    we_spigolo* vf_sin, vf_dstr;
    we_faccia* f_sin, f_dstr;
} we_spigolo;
typedef struct {
float x, y, z;
    we_spigolo* spigolo;
} we_vertice;
typedef struct {
    we_spigolo* spigolo;
} we_faccia;
```



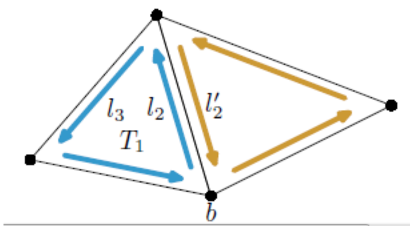
5.14 Half edge

Ogni spigolo viene diviso in due spigoli orientati in modo opposto:

- Ciascun mezzo spigolo contiene un puntatore al vertice iniziale, alla faccia a cui "appartiene", al mezzo spigolo successivo (seguendo l'ordinamento) ed al mezzo spigolo gemello
- Ogni vertice, oltre alle coordinate (e attributi) contiene un puntatore ad uno qualsiasi dei mezzi spigoli che esce da tale vertice
- Ogni faccia contiene uno dei suoi mezzi spigoli (oltre ad altre caratteristiche quali, ad esempio, la normale)

Efficiente per le ricerche ed elegante.

```
typedef struct {
    he_vertice* origine;
    he_spigolo* gemello;
    he_faccia* faccia;
    he_spigolo* successivo;
} he_spigolo;
typedef struct {
    float x, y, z;
    he_spigolo* spigolo;
} he_vertice;
typedef struct {
    he_spigolo* spigolo;
} he_faccia;
```



Note:
La stessa applicazione grafica può far uso di più di una struttura dati.
La rappresentazione con la lista di vertici essendo semplice e leggera è tipicamente usata come formato per i file contenenti la geometria di oggetti.
Le applicazioni grafiche in genere caricano tali file ed usano l'informazione contenuta in essi per riempire una struttura dati più utile ai fini algoritmici (per esempio la half-edge).

6 Rendering

Interazione luce-materia.
La luce raggiunge la superficie. Viene riflessa, ma come? Molti effetti.
Parte assorbita, parte riemessa, magari in punto diverso visto che può essere riflessa sotto la superficie, anche con ritardo di tempo Modifica lunghezza d'onda.
Troppo complicato, ma alcuni effetti per essere simulati richiederebbero questo. Fosforescenza, fluorescenza, ecc.

In grafica di solito usiamo la **BRDF** (Bidirectional Reflectance Distribution Function): modelliamo riflessione come se dipendente da soli 4 parametri. Direzione luce incidente-riflessa, è comunque complicatissimo. Inoltre assumeremo di poter pensare alla luce come composizione di 3 fasci monocromatici a frequenze fisse (R,G,B) o di un solo valore di luminosità (ovviamente falso).
La radiazione luminosa è caratterizzata da

- distribuzione spettrale, che ne determina il colore
- energia che ne determina l'intensità o luminosità per ora usiamo in modo informale questi termini.

Fotometria: la misura dell'energia trasportata dalle onde elettromagnetiche della gamma ottica (spettro visibile). la fotometria si occupa dell'azione della luce visibile sull'occhio umano.

6.1 Radiometria

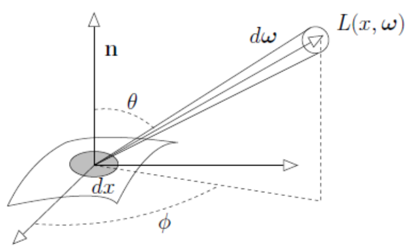
Radiometria: si occupa di radiazioni estese sull'intero intervallo delle possibili lunghezze d'onda e non considera gli effetti sull'osservatore. Essendo interessati ad un modello oggettivo, definiremo le grandezza radiometriche.
Assumendo che non ci sia interazione tra le diverse lunghezze d'onda, si può misurare l'energia indipendentemente per un certo numero di lunghezze d'onda campione che servono a rappresentare l'intera distribuzione spettrale.
Di solito se ne usano 3, per motivi legati al sistema visivo umano, corrispondenti al rosso, verde e blu (RGB).
Tutte le grandezze che definiremo sono implicitamente spettrali, ovvero riferite ad una singola lunghezza d'onda.

Le quantità che misuriamo sono:

- Il **flusso radiante** Φ è la velocità alla quale l'energia luminosa viene emessa (o assorbita) da una superficie, ha le dimensioni di una potenza (energia per unità di tempo) e si misura in Watt [W].
- **Irradianza** $E(x)$ il rapporto tra il flusso ricevuto da un elemento infinitesimo di superficie in x e la sua area dx:
$$E(x) = \frac{d\Phi}{dx}$$
- **Radiosità** $B(x)$ il rapporto tra il flusso emesso da un elemento infinitesimo di superficie in x e la sua area dx:
$$B(x) = \frac{d\Phi}{dx}$$
  Irradianza e radiosità sono la stessa grandezza (una densità superficiale di flusso) e si misurano in $\left[\mathrm{W/m^2}\right]$. La differenza è che l'**irradianza è energia ricevuta**, la **radiosità è energia emessa**. In entrambi i casi, l'energia ricevuta/emessa si considera da/verso tutte le direzioni.

- L'**intensità radiante** è il flusso radiante emesso in un angolo solido infinitesimo $d\omega$ lungo una particolare direzione ω :
$$I(\omega) = \frac{d\Phi}{d\omega}$$
Si usa soprattutto per descrivere sorgenti luminose puntiformi.

- L'**angolo solido sotteso** da un oggetto rispetto un punto P è pari all'area della proiezione dell'oggetto su una sfera unitaria centrata in P.
- L'**angolo solido sotteso** da un elemento infinitesimo di superficie dx centrato in x ed orientato con normale $\boldsymbol{n}$, rispetto ad un punto y distante r vale: $d\omega = dx \cos\theta / r_2$ dove θ è l'angolo formato dalla normale $\boldsymbol{n}$ con la congiungente y ed x.
Il termine $dx \cos\theta$ rappresenta l'area proiettata di dx lungo la congiungente y ed x. Se poniamo in y una sorgente di luce puntiforme con intensità radiante I, allora l'irradianza nel punto x vale:
$$E(x) = \frac{d\Phi}{dx} = \frac{I\cos\theta}{r_2}$$
- **Radianza** $L(x,\boldsymbol{\omega})$ nel punto x in una direzione ω è la densità superficiale della intensità radiante in x lungo la direzione $\boldsymbol{\omega}$, considerando l'area della superficie proiettata:
$$L(x,\boldsymbol{\omega}) = \frac{dI(\omega)}{dx(\boldsymbol{\omega}\cdot\boldsymbol{n})}$$
L'area proiettata della superficie infinitesima dx è l'area della proiezione di dx (la cui normale è $\boldsymbol{n}$) sul piano perpendicolare a $\boldsymbol{\omega}$, e vale dunque $dx(\boldsymbol{\omega}\cdot\boldsymbol{n})$.
La direzione $\boldsymbol{\omega}$ è data da due angoli: l'elevazione θ (rispetto alla normale alla superficie $\boldsymbol{n}$) e l'azimuth ϕ(rispetto ad una direzione fissata sulla superficie.)
Possiamo dunque scrivere: $\boldsymbol{\omega}\cdot\boldsymbol{n} = \cos\theta$.
La radianza $L(x,\boldsymbol{\omega})$ è la densità di flusso nel punto x in una direzione $\boldsymbol{\omega}$, misurata rispetto ad una superficie infinitesima perpendicolare a $\boldsymbol{\omega}$.
La radianza è pari al il flusso radiante per unità di angolo solido per unità di area proiettata lungo la direzione di propagazione, e si misura in $\left[\mathrm{W}/(\mathrm{m}^2\cdot\mathrm{sr})\right]$.



6.1.1 Radianza lungo un raggio

Dati due punti x e y (nel vuoto) la radianza che lascia x verso y è uguale a quella che raggiunge y dalla direzione di x: non si attenua con la distanza.
Il modello che si usa in grafica è quello di raggi luminosi che trasportano una certa quantità di energia luminosa. Nel ray casting, dunque, i raggi luminosi trasportano radianza, ed i pixel registrano il valore della radianza (idealmente).
Quando informalmente si parla di "intensità" del pixel, si fa riferimento alla radianza.
In realtà nelle immagini digitali, l'intensità è mappata in un range dinamico limitato (tipicamente 8 bit) e non linearmente per compensare la non linearità della percezione umana.

Sia Ω la semisfera delle direzioni attorno alla normale in x.
Dall'equazione scritta prima si ha:
$$L(x,\boldsymbol{\omega})(\boldsymbol{\omega}\cdot\boldsymbol{n}) = \frac{d^2\Phi}{d\omega dx}$$

ed integrando:

$$\int_\Omega L(x,\omega)(\omega \cdot \boldsymbol{n})d\omega = \int_\Omega \frac{d^2\Phi}{d\omega dx}d\omega = \frac{d\Phi}{dx} = B(x)$$

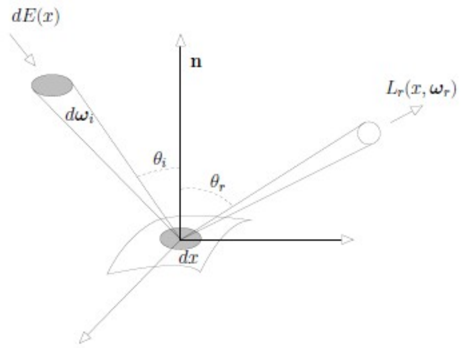
Similmente, per l'energia incidente, l'irradianza vale:

$$E(x) = \int_\Omega L(x,\omega)(\omega \cdot \boldsymbol{n})d\omega$$

Nota: la radianza è definita per unità di area proiettata, mentre la irradianza/radiosità sono definite per unità di area (effettiva).

6.2 BRDF

Ritroviamo quindi la Bidirectional Reflectance Distribution Function (BRDF) che caratterizza il materiale di cui è composta la superficie. La BRDF $\rho(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_r})$, è il rapporto tra la radianza riflessa da x lungo la direzione $\boldsymbol{\omega_r}$ e l'irradianza della luce incidente nel punto x da un angolo solido infinitesimale $d\boldsymbol{\omega_i}$ centrato in $\boldsymbol{\omega_i}$:



4 gradi di libertà (due angoli). Sarebbero 5 se considerassimo la dipendenza dalla lunghezza d'onda della luce. E' già una drastica semplificazione. Supponiamo che localmente sia costante (stesso materiale in una regione). Non modella alcuni effetti dei materiali reali:

- Dipendenza dalle lunghezze d'onda.
- Fluorescenza: riemissione di energia a frequenze diverse da quelle ricevute, tipicamente visibile da stimolo ultravioletto.
- Fosforescenza: riemissione anche dopo la cessazione dello stimolo Scattering sotto la superficie.
- Occorrerebbe, come detto all'inizio, introdurre funzioni con più gradi di libertà.

Si usa la irradianza e non la radianza per misurare la densità di flusso incidente perché quest'ultima non tiene conto della reale orientazione della superficie.
Si vede facilmente che l'irradianza è legata alla radianza della luce incidente $L_i(x, \boldsymbol{\omega_i})$ da:

$$dE(x) = L_i(x, \boldsymbol{\omega_i})(\boldsymbol{\omega_i} \cdot \boldsymbol{n})d\boldsymbol{\omega_i}$$

dove $P_0(\boldsymbol{\omega_i} \cdot \boldsymbol{n}) = \cos\Theta_i$
Dunque la BRDF si scrive :

$$\rho(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_r}) = \frac{L_r(x, \boldsymbol{\omega_r})}{L_i(x, \boldsymbol{\omega_i})(\boldsymbol{\omega_i} \cdot \boldsymbol{n})d\boldsymbol{\omega_i}}$$

Siccome la radianza è definita per unità di area proiettata, moltiplicando per $(\boldsymbol{\omega_i} \cdot \boldsymbol{n})$ la si converte in una misura per unità di area (non proiettata).
In altri termini, si tiene conto del fatto che la radianza è misurata rispetto ad un'area infinitesima orientata diversamente da quella che in effetti viene illuminata, mentre noi vogliamo esprimere

l'effettivo flusso incidente.
Se consideriamo i contributi di irradianza da tutte le direzioni di incidenza, la radianza totale riflessa nella direzione ω_r, e vale:

$$L_r(x,\omega_r) = \int_{\omega_i \in \Omega} \rho(x,\boldsymbol{\omega_i},\boldsymbol{\omega_r}) L_i(x,\boldsymbol{\omega_i})(\boldsymbol{\omega_i} \cdot \boldsymbol{n}) d\boldsymbol{\omega_i}$$

6.2.1 Diffusione pura

Una superficie Lambertiana (o diffusore perfetto) ha una BRDF costante: $\rho(x,\boldsymbol{\omega_i},\boldsymbol{\omega_r}) = \rho(x)$. La radianza (riflessa) di tale superficie non dipende dalla direzione. Inoltre:

$$L_r(x,\boldsymbol{\omega_r}) = \rho(x) \int_{\Omega} L_i(x,\boldsymbol{\omega_i})(\boldsymbol{\omega_i} \cdot \boldsymbol{n}) d\boldsymbol{\omega_i} = \rho(x)E(x) = L(x)$$

$$B(x) = \int_{\Omega} L(x,\boldsymbol{\omega})(\boldsymbol{\omega} \cdot \boldsymbol{n}) d\boldsymbol{\omega} = L(x) \int_{\Omega} \cos\theta d\boldsymbol{\omega} = \rho(x)E(x)\pi$$

$\rho_d(x) = \pi\rho(x)$ prende il nome di **albedo**.

- L'albedo è la frazione di irradianza $E(x)$ che viene riflessa come radiosità $B(x)$. Il resto dell'energia viene assorbito
- Diffusore perfetto: superficie ruvida (es. gesso, coccio) che ripartisce la radianza entrante uniformemente su tutte le direzioni

La specifica esatta della BRDF per superfici reali è estramemente difficile da ottenere.
Nella grafica al calcolatore si usano approssimazioni della BRDF. Le due più semplici e più usate modellano due comportamenti ideali dei materiali: una è appunto la diffusione, l'altra è la riflessione speculare.
Un riflettore speculare si comporta come uno specchio perfetto, che riflette il raggio incidente lungo una direzione che forma con la normale lo stesso angolo formato dalla direzione di incidenza.