



Computer Graphics 2023

Lezioni

Martedì	16-19
Giovedì	11-13

Lezioni

Il sistema grafico

Sistema grafico

Raster devices

Standard Display resolution

Luce e colori

Prisma di Newton

Sistema Visivo Umano

Teoria tricromatica di Young

Modelli di colore

Modello RGB

Modello CMY

Spazio di colori: HSI (Hue, Saturation, Intensity)

Immagini raster ed immagini vettoriali

Grafica raster

Grafica vettoriale

Scalable Vector Graphics SVG

Raster Scan Display System

Scheda video

RAM

BIOS

DAC

Il collegamento Scheda-Monitor

Display

Refresh rate e frame rate

Scansione progressiva e interlacciata

Frame buffer

Double buffer

Color Frame buffer

Metodi per codificare immagini a colori nel frame buffer

Rendering Graphics pipeline

Real-Time Graphics Pipeline

Geometry Stage

Rasterization
Fixed Function Pipeline
 Offline rendering
 Soluzione
Programmable Pipeline
Vertex Shader
Geometria per la Computer Graphics
 Sistemi di coordinate utilizzato nella grafica
 Spazi
 Scalare
 Punto
 Vettore
 Spazio vettoriale/lineare
 Operazioni
 Definizione di combinazione lineare
 Definizione di indipendenza lineare
 Dimensione dello spazio
 Definizione di base
 Base canonica
 Definizione del prodotto scalare canonico
 Proprietà
 Ortogonalità di due vettori
 Norma di un vettore
 Normalizzazione di un vettore
 Prodotto scalare (prodotto interno o dot product)
 Proiezione ortogonale e Interpretazione geometrica
 Prodotto scalare con vettori generici
 Vettori perpendicolari
 Cross Product
 Proprietà del prodotto vettoriale
 Spazi affini
 Mappare i punti in vettori
 Coordinate baricentriche di un punto nello spazio affine
 Combinazione convessa
 Coordinate baricentriche di un punto appartenente ad un triangolo
 Convessità
 Guscio Convesso
 Distanza tra due punti
 Distanza tra un punto ed una linea
 Equazione in forma implicita di una linea 2D
 Intersezione linea-segmento
 Rappresentazione di un piano nello spazio 3D
 Distanza tra un punto Q ed un piano π
 Rappresentazione implicita dei piano in 3D
 Sistemi di riferimento

Riferimento o Frame
Coordinate omogenee
Conclusione
Cambio di sistema di riferimento in 2D
Esempio 2
Cambio di sistema di riferimento in 3D
Trasformazioni geometriche in 2D
Trasformazioni affini
Trasformazione di Traslazione
Trasformazione di scalatura
Trasformazione di rotazione
Le coordinate omogenee
Trasformazioni e coordinate omogenee
Altre trasformazioni: riflessione
Deformazione (shear)
Composizione di Trasformazioni
Non commutatività
Invertibilità delle Trasformazioni
Trasformazioni geometriche in 3D
Traslazione
Scalatura 3D
Rotazione 3D
Curve interpolanti Hermite
Curve parametriche
Lunghezza di una curva
Continuità geometrica e continuità di una curva
Curve interpolanti di Hermite
Curve Interpolanti
Curve Interpolanti di Hermite
Dimostrazione
Rendering Pipeline: Geometric Stage
WCS
VCS
Geometric Stage
Trasformazioni di Modellazione
Pinhole camera
Semplificazioni
Trasformazione di vista
Definire la vista: camera look at
Costruzione della matrice di Trasformazione di Vista
Projection transform
Clipping
Proiezioni Prospettiche
Proiezioni Parallele
Proiezioni ortografiche assonometriche

Proiezione Isometrica

Proiezioni oblique

Proiezione geometrica di una figura piana da 3D → 2D

Volume di vista, proiezione prospettica

Volume di vista, proiezione parallela ortografica

Normalizzazione

Normalizzazione del volume di vista nelle proiezioni ortogonali

Matematica delle proiezioni

Pipeline di rendering

Scan Conversion

Rasterization

Fragment processing

Visibility/Display

Rasterization

Algoritmo DDA, Digital Differential Analyzer

Algoritmo di Bresenham o Mid-Point

Curve di Bezier

Caso generale

Vantaggi

Caso generale polinomio di grado n definito su un intervallo

Proprietà dei polinomi di Bernstein di grado n

Proprietà dell'inviluppo convesso, convex Hull

Altre proprietà

Valutazione di un polinomio della base di Bernstein

Valutazione curva di Bezier

Interpolazione lineare, Lerp

Interpretazione geometrica dell'algoritmo di de Casteljau

Modelli di illuminazione e shading

Modelli di shading

Lightening vs Shading

Curve Spline

Curve Spline polinomiali di ordine m a nodi multipli

Definizione

Spline polinomiali a nodi multipli

Base dello spazio delle Spline B-spline normalizzate

Funzioni base B-spline, nodi semplici

Formule di Cox per la valutazione di un B-spline

Partizione nodale estesa

Effetto dei nodi multipli sulle funzioni B-spline

Proprietà delle funzioni B-spline con nodi multipli normalizzati

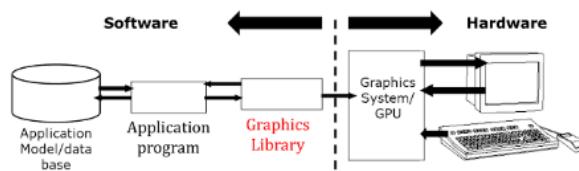
Il sistema grafico

Sistema grafico

Un sistema grafico è costituito da dispositivi **hardware** e programmi **software** che interagiscono per produrre immagini grafiche.

I **dispositivi hardware** rappresentano la potenzialità del sistema grafico: le risorse che rendono possibile produrre immagini.

Il **software** serve ad organizzare le risorse hardware: utilizza l'hardware per creare e manipolare immagini grafiche.



Il programma applicativo che si interfaccia all'hardware grafico mediante funzioni che appartengono ad una libreria grafica (OpenGL, DirectX, etc) che prendono il nome di **Application Programmer's Interface API**. I drivers della scheda video sono software che permettono al sistema operativo di comunicare con la scheda video: interpretano l'output delle API e lo convertono in una forma che è riconoscibile per la scheda video.

Lo sviluppo della computer graphics è parallelo a quello della tecnologia hardware, quindi per poter conoscere e sfruttare fino in fondo le sue potenzialità è necessario avere una **visione chiara di come si è evoluto nel tempo l'hardware grafico** e delle conseguenze di quest'evoluzione nella pipeline grafica.

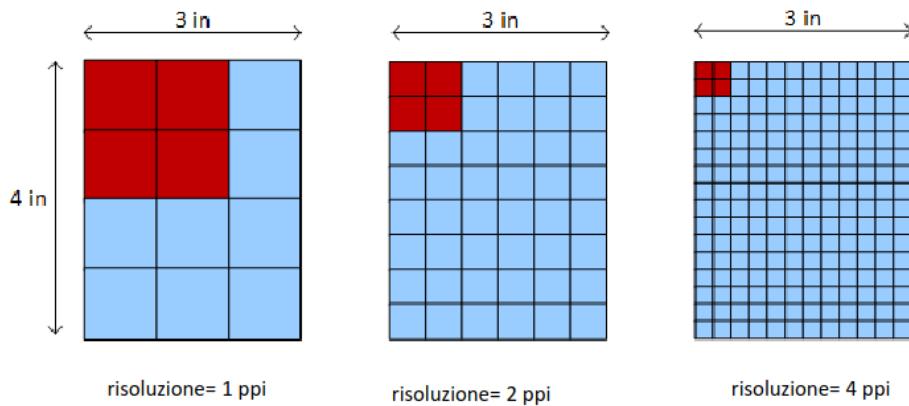
Raster devices

Un **raster devices** è composto da una matrice rettangolare di campioni o **pixel**. La **risoluzione** di un dispositivo raster misura la **densità dei pixel**: è il numero di pixel su unità di distanza o area.

- **Video:** pixel per inch **ppi**, oppure pixel per centimetro **ppc**
- **Stampanti/Scanner:** dot per inch **dpi**

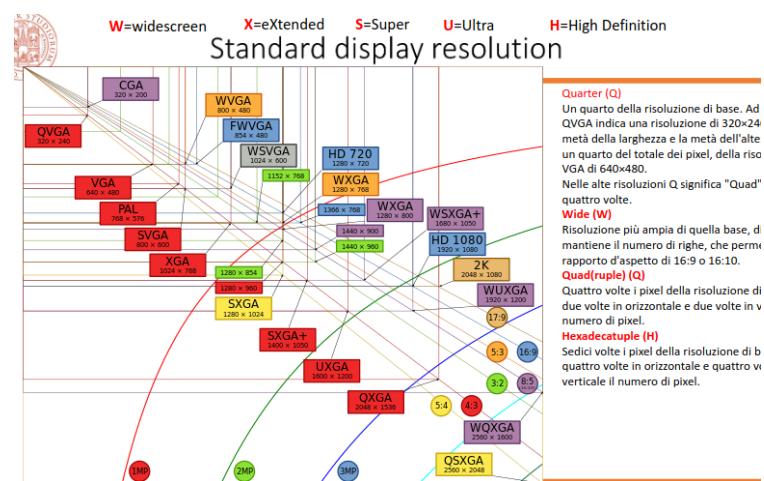
La **risoluzione display** è il numero di pixel per ogni dimensione. La **dimensione fisica** di un dispositivo raster: larghezza x altezza.

$$\text{Dimensione fisica} = \text{Risoluzione display}/\text{Risoluzione raster}$$



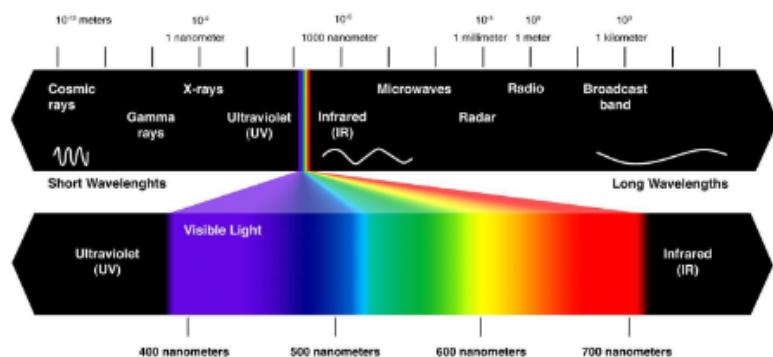
I dispositivi hanno la stessa dimensione fisica, MA risoluzioni diverse

Standard Display resolution



Luce e colori

Il sistema visivo umano percepisce una parte molto limitata delle radiazioni elettromagnetiche, quelle con lunghezze d'onda comprese tra 400 nanometri e 800 nanometri.



La sensazione fisiologica che il sistema visivo umano produce quando l'occhio viene colpito dalla luce riflessa, emessa o trasmessa da un oggetto, viene definita come **colore**.

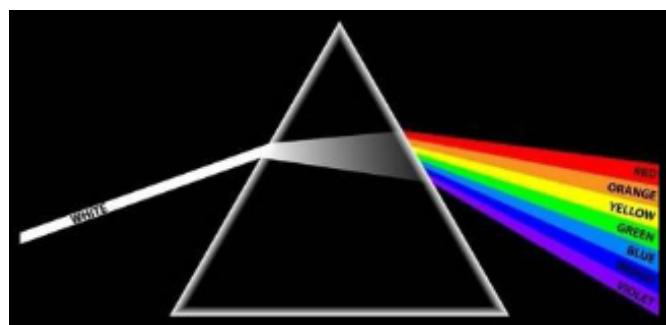


- **Lunghezza d'onda**, distanza tra due massimi o due minimi dell'onda
- **Aampiezza**, la distanza tra un massimo e il piano mediano che interseca l'onda
- **Frequenza**, numero di oscillazioni nell'unità di tempo

L'ampiezza dell'onda influisce sull'**intensità luminosa** dello stimolo elaborato dal cervello, mentre la **lunghezza** dell'onda influenza la **tonalità del colore percepito**

Prisma di Newton

La luce bianca è composta da tutti i colori dello spettro.



La luce bianca entra nel prisma e lo attraversa, scomponendosi nei sette colori fondamentali

Gli oggetti posseggono delle particelle, chiamate **pigmenti colorati**, che sono responsabili dell'assorbimento selettivo della luce.

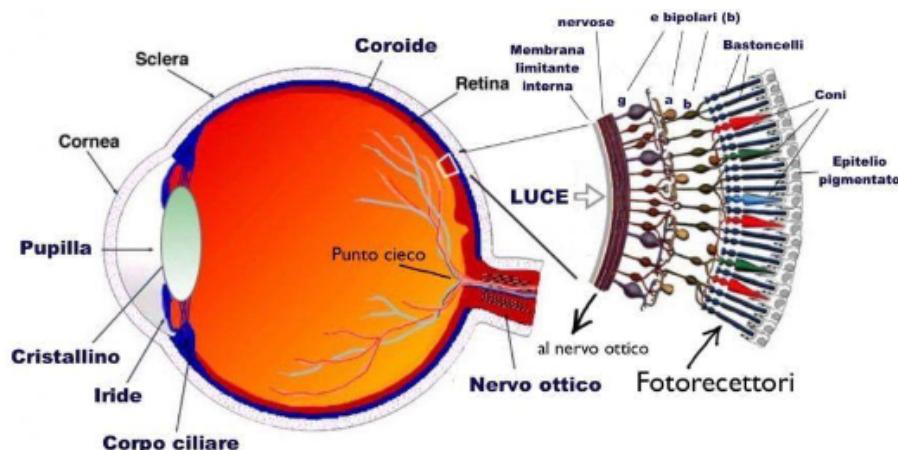
Gli oggetti appaiono di un certo colore perché assorbono, o sottraggono, tutti i colori della luce visibile tranne il colore che viene riflesso all'occhio.

- Una mela rossa assorbe tutte le lunghezze dell'onda della luce eccetto il rosso, che riflette
- Una palla blu assorbe tutte le lunghezze dell'onda della luce eccetto il blu, che riflette
- Una filtro giallo assorbe tutte le lunghezze dell'onda della luce eccetto il giallo, che trasmette

Sistema Visivo Umano

Nella retina vi sono due tipi di cellule sensibili alla luce:

- **coni**: sono sensibili alla lunghezza d'onda, percezione del colore
- **bastoncelli**: hanno lo scopo di adattarsi ai cambiamenti di intensità di luce ad esempio alla luce crepuscolare e notturna



Teoria tricromatica di Young

Thomas Young ritenne impossibile per l'esiguità della superficie della retina, **che in ogni punto potessero esistere infiniti fotorecettori diversi quanti i colori discriminati dall'occhio**.

Teorizzò quindi l'esistenza di **tre soli tipi di recettori**, associati ai tre colori primari pittorici: **giallo, magenta e ciano**.

Modelli di colore

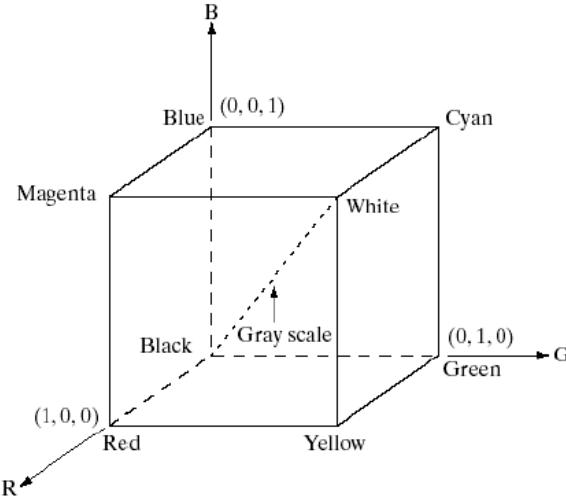
Un modello di colore è un modello matematico astratto che descrive il modo in cui i colori possono essere rappresentati come tuple di numeri, tipicamente come tre o quattro valori di componenti di colore.

I principali modelli del colore di un'immagine digitalizzata sono:

- **Modello RGB**, sintesi adattiva, il colore risultante è ottenuto sommando i contributi luminosi dei tre colori primari
- **Modello CMY (cyan, magenta, yellow)**, sintesi sottrattiva, il colore risultante è ottenuto rimuovendo dalla luce bianca i contributi dei 3 colori primari. Gli oggetti appaiono colorati in un certo modo soltanto perché le sostanze che li costituiscono assorbono dalla radiazione incidente l'energia associata a certe lunghezze d'onda.
 - $(CMY) = (1, 1, 1) - (RGB)$

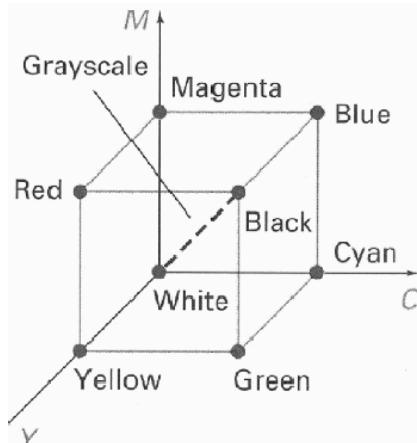
Modello RGB

Nella rappresentazione 3D del modello colorimetrico possiamo associare ad ogni asse un colore primario; la rappresentazione del modello RGB è il cubo unitario.



I grigi sono sulla diagonale del cubo unitario, rappresentato con tutti e 3 i valori uguali (100, 100, 100)

Modello CMY



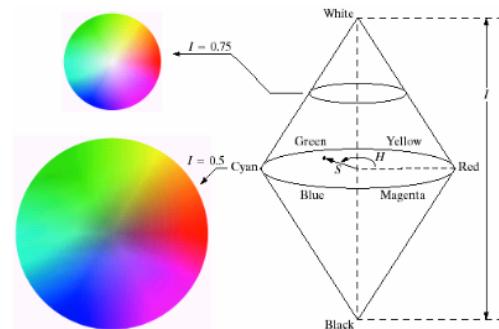
Spazio di colori: HSI (Hue, Saturation, Intensity)

Viene introdotto lo spazio **HSI** perché è abbastanza innaturale per un essere umano descrivere un colore attraverso le sue **componenti RGB**. Permette di descrivere un colore attraverso i concetti più familiari di tinta (hue), saturazione (saturation) e luminosità (intensity).

- **Hue**, tonalità, colore dominante così come viene percepito dall'osservatore
- **Saturation**, purezza del colore o **quantità di luce bianca miscelata** con una certa tinta

- **Intensity**, luminanza, **luminosità** o oscurità di un particolare colore, varia tra 0 e 1 (nero e bianco)

Si immagini di mettere il cubo RGB in equilibrio sul vertice $(0, 0, 0)$ il vertice $(1, 1, 1)$ in alto e quindi mantenendo verticale la linea dei grigi. Il cubo viene ad assomigliare così ad un doppio cono, con la base a mezz'altezza e i due vertici in basso e in alto.



La **tonalità H** viene misurata da un angolo intorno all'asse verticale, con il rosso a 0 gradi, il verde a 120 e il blu a 240.

La **saturazione S** vale zero sull'asse del modello: una tinta non satura ha un colore che tende ad un grigio chiaro, una tinta satura ha un colore vivo e forte.

L'altezza del modello rappresenta l'**intensità I** con lo zero che rappresenta il nero e uno il bianco.

Immagini raster ed immagini vettoriali

Nella grafica raster, l'immagine è una griglia rettangolare di pixel colorati. Nella memoria del computer vengono conservati i singoli pixel dell'immagine bitmapped. Un file **bitmap** contiene i valori dei pixel (.bmp, .gif).

Nella grafica ad oggetti, detta **grafica vettoriale**, un'immagine consiste di oggetti grafici ognuno dei quali è definito, nella memoria del computer, da un'equazione matematica. Il file contenente un'immagine vettoriale contiene comandi e dati (.dxf, .svg).

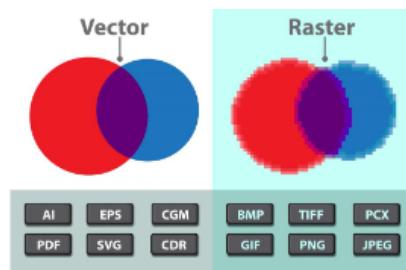
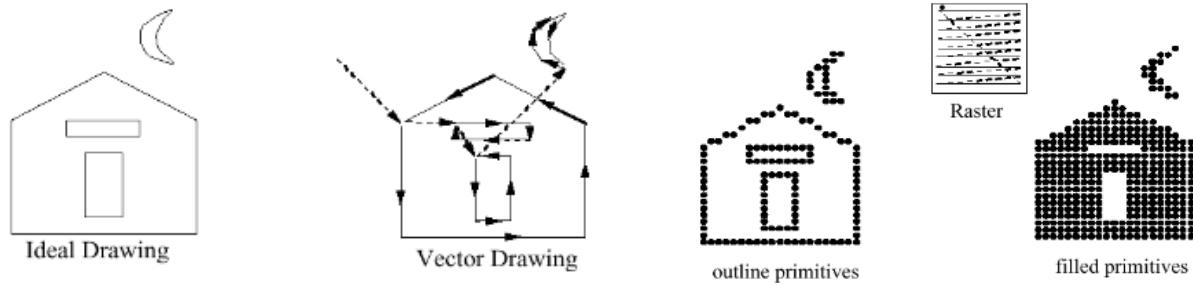


Immagine vettoriale

Immagine raster



Poiché ogni oggetto nella grafica vettoriale è rappresentato da un'**equazione matematica**, per riprodurre l'immagine su un dispositivo raster questa **va trasformata in pixel**, operazione che si dice **rasterizzazione**.

Per le **immagini in bitmap** ovviamente non esiste il concetto di rasterizzazione, in quanto sono già per definizione una **griglia di pixel colorati**.

Grafica raster

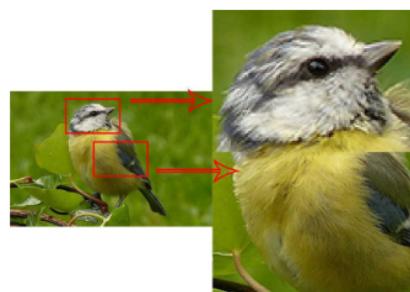
La **grafica a punti** (bitmap o raster) trova applicazione soprattutto nell'elaborazione di immagini fotografiche e nelle illustrazioni pittoriche. La loro natura a mosaico costituisce il punto di forza e anche di debolezza delle immagini bitmap.

Vantaggi

Lavorando con i singoli pixel si possono ottenere effetti simili a quelli della pittura grafica e tradizionale. I programmi di fotoritocco funzionano con immagini raster ed i ritocchi sono possibili punto per punto.

Svantaggi

L'immagine si può ingrandire solo ingrandendo la dimensione del pixel, che può diventare visibile, fino a creare effetti sgradevoli



Per **elaborare** una parte dell'immagine occorre letteralmente selezionare alcuni pixel e spostarli, indipendentemente da quello che rappresentano. La zona in cui erano rimane vuota, e i pixel vanno a sovrascrivere permanentemente quelli che si trovano nella posizione di arrivo.

Nel **ruotare** un'immagine bitmap, i pixel vengono risistemati in modo che l'immagine appaia ruotata; ma, a meno che la rotazione non sia stata di 90° o multipli, la rotazione è solo

un'approssimazione dell'immagine originale.

Nello stampare un'immagine bitmap, la stampante riproduce all'immagine punto per punto esattamente come i punti sono. Ciò indipendentemente dalla risoluzione della stampante.

Grafica vettoriale

La grafica a oggetti è un'evoluzione della grafica a punti. I programmi che consentono di creare immagini ad **oggetti** non memorizzano il disegno come insieme di punti, ma con **formule matematiche che descrivono i singoli oggetti**. Ogni oggetto del disegno viene memorizzato in un database interno di oggetti grafici descritti matematicamente.

Grazie a questa tecnica gli oggetti si possono **ingrandire, rimpicciolire, ruotare, ridimensionare, colorare senza nessuna perdita di qualità**.

Gli oggetti possono essere trattati in modo indipendente, come se ognuno fosse tracciato su un foglio trasparente; pertanto è anche possibile mettere gli oggetti uno sull'altro.

In fase di stampa, invece di indicare alla stampante **dove vanno stampati i singoli pixel**, alla stampante arriva la descrizione matematica e la stampante stampa l'immagine alla sua migliore risoluzione.

La qualità d'illustrazione ad oggetti è **device independent**: l'output uscirà alla miglior risoluzione della stampante. Generalmente poi la descrizione di un'immagine ad oggetti occupa meno spazio della descrizione di un'immagine a punti.

Svantaggi

Uno dei piccoli svantaggi consiste nella **natura intrinsecamente discreta della rappresentazione tramite pixel**: le primitive grafiche, devono essere **convertite nelle matrici di pixel** che meglio le rappresentano, per poter essere visualizzate su un display di tipo raster. Questa operazione è chiamata **scan conversion**.

Scalable Vector Graphics SVG

SVG è uno standard definito in un file di testo XML per la rappresentazione di immagini vettoriali, non solo per l'uso destinato al web. È un linguaggio per descrivere grafici 2D in XML.

```
<svg width="100" height="100">
    <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="y
</svg>
```

Raster Scan Display System

I sistemi grafici a **display raster, Raster Scan Display System**, consistono essenzialmente di tre componenti:

- **Scheda grafica**, al cui interno si trova la **GPU o processore grafico** e una speciale memoria RAM detta **Frame Buffer**
- **Monitor**, connesso via cavo alla scheda grafica
- **Device driver**, mediante il quale il sistema operativo controlla la scheda video

La scheda video è responsabile del rendering dell'immagine sullo schermo del computer: traduce i dati binari prodotti dalla CPU in immagini sullo schermo.

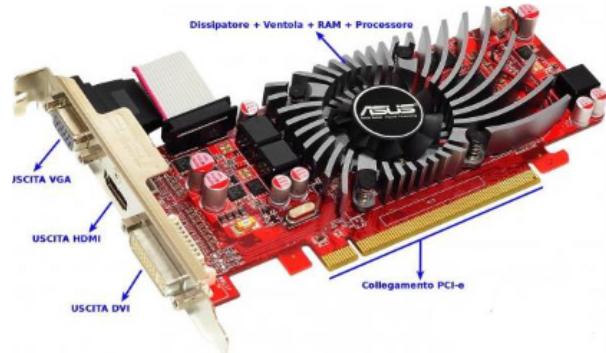
La CPU lavorando insieme ad applicazioni software invia informazioni sull'immagine alla scheda video. La scheda video decide come utilizzare i pixel sullo schermo per creare l'immagine.

Il computer deve eseguire questo processo circa sessanta volte al secondo.

Scheda video

Una scheda video è costituita da un circuito stampato su cui sono montati i componenti:

- GPU
- Memoria video
- BIOS video
- Connessione monitor
- Connessione scheda madre per dati e alimentazione



Unità di elaborazione grafica, o GPU, è il cuore della scheda video moderna, quello che permette di **elaborare** le immagini eseguendo rapidamente gran parte dei complessi calcoli matematici e geometrici richiesti e di **coordinare** l'invio dell'immagine al monitor nel momento giusto.

In genere la GPU è praticamente invisibile in quanto, data l'elevata velocità di elaborazione e la conseguente attitudine a produrre molto calore, è nascosta sotto un dissipatore.

RAM

La RAM è un componente in grado di **memorizzare le informazioni elaborate dalla GPU ed eventualmente immagini completate nell'attesa di inviarle al monitor**. Ciascuna locazione di memoria della RAM si occupa di memorizzare i dati relativi a ciascun pixel, inclusi il colore e la sua posizione sullo schermo.

Si parla di RAM in **modalità frame-buffer** quando una parte di essa è dedicata a memorizzare le immagini complete finché queste non debbano essere visualizzate.

BIOS

Il BIOS è un chip che **memorizza le impostazioni della scheda video e**, al momento del suo avvio, **esegue tutti i test di diagnostica sulla memoria, sull'input e sull'output** della scheda stessa.

DAC

Il DAC è un componente in grado di convertire il segnale digitale generato dalla GPU in un segnale analogico leggibile ed utilizzabile dai monitor VGA. Viene utilizzato soltanto per la "sincronizzazione" della palette in caso di collegamento DVI/HDMI, in tal caso il monitor è in grado di leggere il segnale digitale e non c'è bisogno di conversione.

Il DAC invia le immagini al monitor tramite un cavo di collegamento disponibile ad oggi in diversi formati.

Il collegamento Scheda-Monitor

- **VGA**, si tratta di uno standard compatibile con i monitor a tubo catodico (CRT) ma usato ancora oggi per compatibilità in una gran quantità di monitor LCD. Essendo uno standard vecchio è molto soggetto a problemi come la distorsione delle immagini e la bassa risoluzione
- **DVI**, introdotto grazie ai monitor LCD, risolve molti problemi dei cavi VGA
- **HDMI**, si tratta di uno standard più recente in grado di supportare risoluzioni ad alta definizione e che cerca ancora oggi di imporsi a tutti gli altri standard

Display

Il display utilizzato con il PC è un **dispositivo raster**: lo schermo consiste di una matrice rettangolare di pixel. Ogni pixel del monitor può assumere un colore tra quelli disponibili. L'**intensità del colore** di ogni pixel viene letta dalla memoria frame buffer.

- tubo a raggi catodico
- cristalli liquidi (LCD)
- al plasma (PDP, Plasma Display Peripheral)

Refresh rate e frame rate

Il **refresh rate** è il numero di frame che vengono visualizzati sullo schermo al secondo (include il disegno ripetuto di frame identici).

Il **frame rate**, anche noto come frames per seconds **FPS**, misura quante immagini differenti al secondo sono visualizzate.

Scansione progressiva e interlacciata

I display LCD sono nativamente a **scansione progressiva**; avviene attraverso la scansione di tutti i frame immediatamente, una linea alla volta dall'alto verso il basso e da sinistra verso destra.

Nella **scansione interlacciata** la scansione avviene sulla divisione di un frame in due semiquadri: un semiquadro contiene tutte le linee dispari dell'immagine, l'altro contiene tutte le linee pari. Il video interlacciato visualizza tutte le linee di scansione pari e dispari come campi separati. Le linee di scansione **pari** vengono disegnate sullo schermo, successivamente vengono disegnate le linee di scansione **dispari** e si ottiene così un fotogramma video.

Tearing: un singolo fotogramma visualizzato contiene informazioni provenienti da due o più fotogrammi.

Questo problema si ha con la scansione interlacciata per il fatto che visualizza solo metà immagine alla volta



Frame buffer

Il **frame buffer** è una porzione della **memoria RAM** presente nella **scheda video**, dove vengono memorizzate le immagini prima di essere visualizzate sul display.

Il frame buffer è una **collezione di diversi buffer** dedicati. Il principale tra questi è il **color frame buffer**. Esso contiene le componenti del colore per ogni pixel.

Double buffer

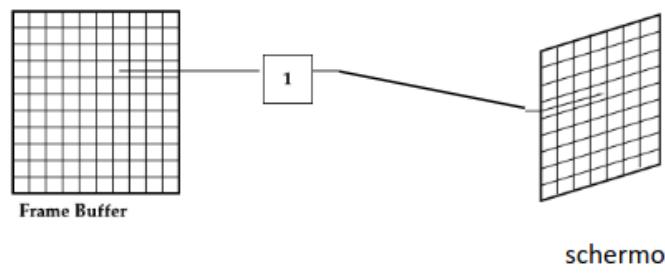
Il **double buffer** è comunemente usato per velocizzare le animazioni. Se l'immagine finale mostrata a video è contenuta nel **front buffer**, nel **back buffer** è contenuta la scena che sta per essere mostrata. I due buffer vengono poi scambiati dal processore grafico, dopo lo scambio i due ruoli di back e front buffer si scambiano, e così succede dopo ogni frame visualizzato.

Color Frame buffer

Nel **color frame buffer** ogni posizione di memoria indica il **colore di un pixel sul display**. Poiché il **colore dei pixel** di un monitor si forma in **sintesi additiva** a partire da tre primari R, G e B, ogni posizione sarà divisa in tre zone per i valori R, G, B; se per ogni colore primario sono riservati otto bit, ogni pixel richiede 24 bit (3 byte).

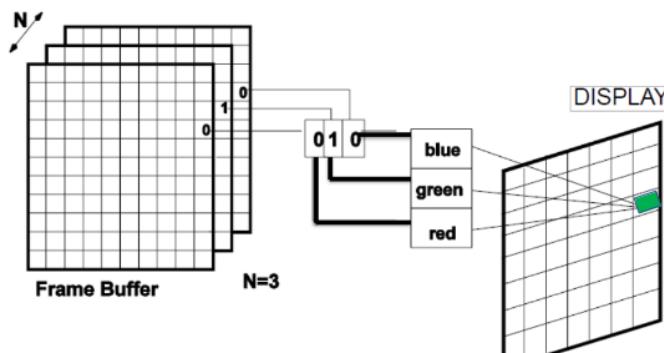
Metodi per codificare immagini a colori nel frame buffer

Video monocromatico: il color buffer ha 1 bit per locazione, quindi basta 1 bit per pixel e quindi il color buffer ha un solo piano di bit



Il color buffer per un monitor a colori può essere gestito in **modalità true color o pseudocolor**.

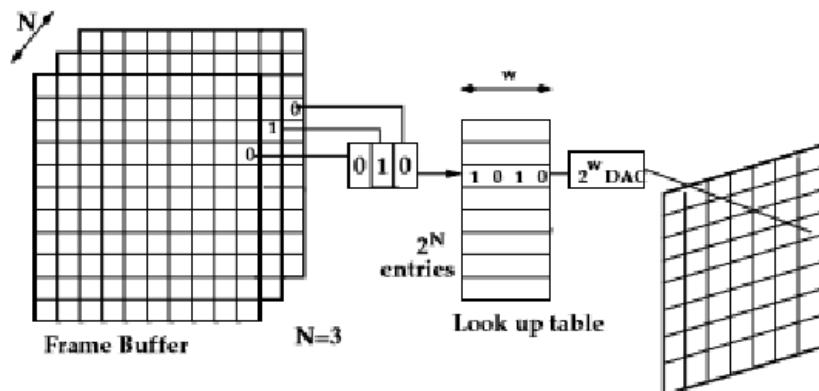
- **Modalità True Color**, i valori memorizzati nel color frame buffer rappresentano i valori effettivi delle componenti colore da visualizzare.



Tipicamente si fa uso di **24 piani di bit** per le tre componenti colore e un **quarto byte per il canale alfa**, che gestisce la trasparenza. Se un pixel ha canale alfa pari a 0, il pixel dovrà essere considerato **trasparente** e quindi non visualizzato.

- **Modalità pseudocolor**, i valori memorizzati nel frame buffer sono trattati come **indirizzi ad una tabella di colori**, chiamata **look up table o colormap, LUT**, che contiene 2^N elementi ciascuno dei quali composto da w bit.

La colormap memorizza 2^w intensità di colore differenti, ma solo 2^N colori differenti possono essere disponibili su Frame Buffer.



Il valore di ogni posizione nel frame buffer viene trattato come un numero di indice nella mappa dei colori; un **vantaggio** di questa modalità è la possibilità di **risparmiare spazio di archiviazione**.

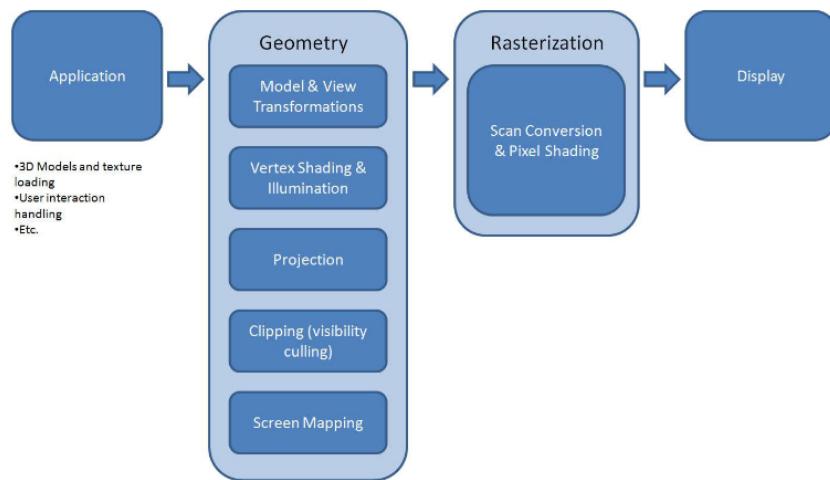
Rendering Graphics pipeline

La GPU ha architettura a pipeline poiché progettata per soddisfare la **graphics rendering pipeline**. In input alla pipeline vi è un insieme di primitive geometriche 3D, camera virtuale, risorse luminose, texture ed altro.

I vari stadi della pipeline possono essere globalmente accorpati in tre principali stadi:

- **application**
- **geometry**
- **rasterizer**

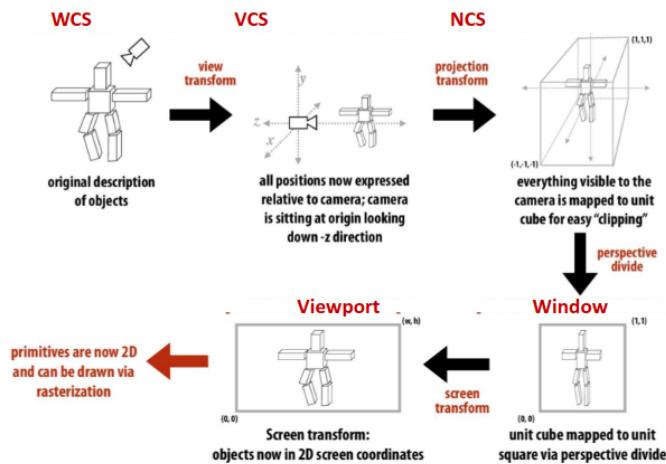
Real-Time Graphics Pipeline



Geometry Stage

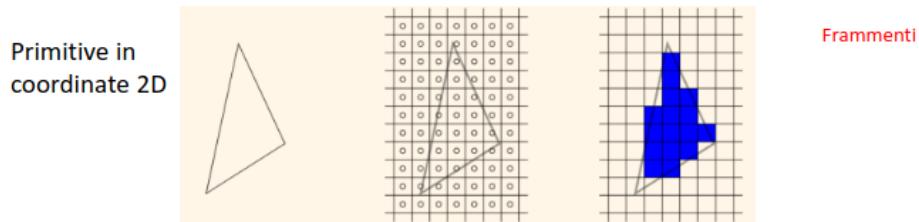
1. **Model & View Transformation**, la GPU deve prima trasformare tutti gli oggetti in un sistema di coordinate comune: dal sistema di riferimento dell'oggetto, al sistema di riferimento del mondo, al sistema di riferimento della camera virtuale
2. **Vertex Shading & Illumination**, una volta che ogni triangolo è in un sistema globale di coordinate, la GPU può calcolare il suo colore in base alle luci nella scena
3. **Projection**, proiezione del volume di vista che contiene la scena in un volume canonico di x, y, z che variano tra -1 e 1
4. **Clipping (visibility culling)**, clipping ⇒ eliminare tutto ciò che è esterno al volume di vista canonico

5. **Screen Mapping**, trasformazione in coordinate dello spazio dello schermo: le primitive sono adesso in 2D e si può passare all processi di **rasterizzazione**



Rasterization

Scan Conversion & Pixel Shading, il processo di **scan conversion** consiste nel determinare i pixel dello schermo interni ad ogni triangolo visibile sullo schermo. Genera i frammenti

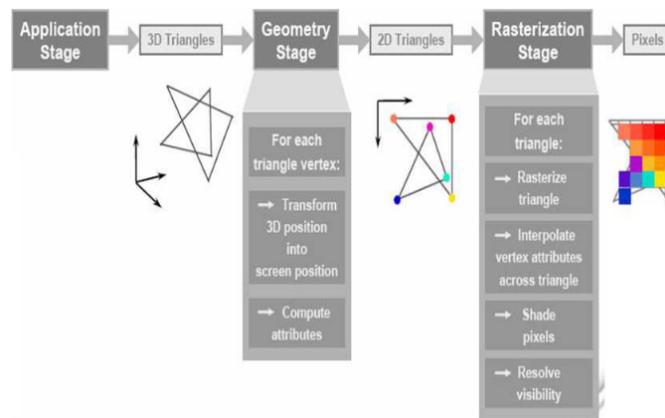


Rasterization, detto a parole povere, scrive la matrice dei pixel che sono coperti dal triangolo

- **Fragment processing**, i frammenti hanno una posizione, *pixel location*, e altri attributi, come colore e coordinate di texture che vengono determinati interpolando i valori sui vertici.

Il colore effettivo di ogni pixel può essere preso direttamente dai calcoli di illuminazione, ma per un maggiore realismo, è possibile mappare delle texture sui triangoli

- **Z-buffer e test di visibilità, Blending**, nella maggior parte delle scene, alcuni oggetti oscurano altri oggetti. Tutte le moderne GPU forniscono un **buffer di profondità**, una regione di memoria che memorizza la distanza da ogni pixel dal visualizzatore. Oppure hanno un livello di trasparenza ed il loro colore si mescola al colore degli oggetti che stanno dietro di essi, *blending*.

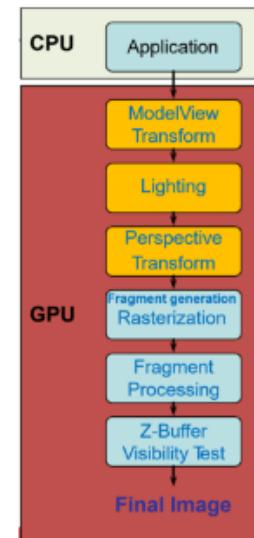


Fixed Function Pipeline

In una **fixed-function pipeline**, ovvero in una pipeline non programmabile, il programmatore interagisce con la pipeline via interfaccia software standard, per esempio OpenGL o DirectX.

Una fixed-function pipeline della 3D API implementa gli algoritmi di rendering direttamente nella scheda grafica, memorizza i dati dei vertici nella onboard video memory, evitando il passaggio dei dati attraverso il system bus.

Con l'evoluzione dell'hardware grafico, si è arrivati alla progettazione e realizzazione di **pipeline programmabili**, in cui c'è la possibilità di poter programmare il comportamento di alcune sue fasi.



La realizzazione delle pipeline in hardware, da un lato ha **permesso di migliorare notevolmente le performance delle applicazioni**, e le interfacce API hanno semplificato lo sviluppo di applicazioni, fornendo ai programmati un livello di attrazione rispetto alle specifiche del sistema, garantendo una maggiore portabilità. Da un lato però, la standardizzazione degli algoritmi di rendering non permetteva la flessibilità negli effetti di resa grafica, limitandone qualità e realismo.

Così facendo i programmati persero un notevole **controllo sull'immagine prodotta**.

Offline rendering

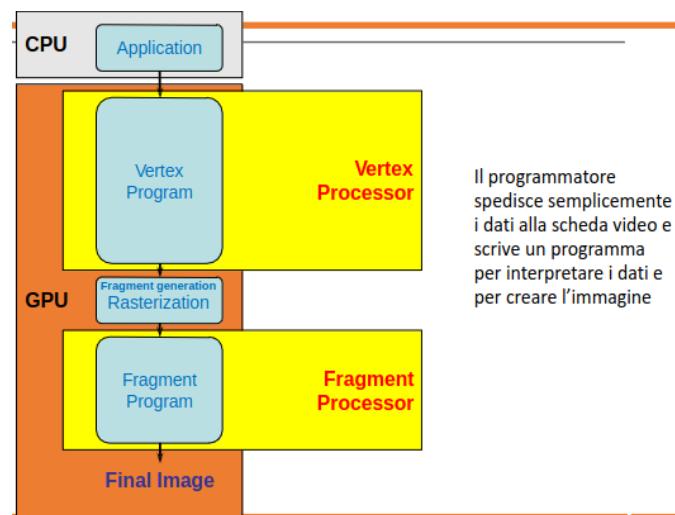
L'offline rendering, a differenza del **real-time rendering** si avvaleva di standard general purpose CPU che costruivano frame by frame le animazioni impiegando giorni, settimane. Il **vantaggio** di usare general purpose CPU era la flessibilità del programmatore di poter usare la CPU per ogni effetto che si poteva immaginare.

L'offline rendering peccava di **scarsa velocità**, ma riusciva ad ottenere rendering di **alta qualità e realismo**. La flessibilità e generalità di offline rendering systems sono le caratteristiche chiave che erano state abbandonate nelle precedenti generazioni di hardware grafico 3D.

Soluzione

Per poter realizzare real-time rendering con effetti paragonabili a quelli ottenibili già qualche anno prima tramite tecniche di offline rendering, era necessario **realizzare GPU in cui le diverse fasi della pipeline fossero programmabili**

Programmable Pipeline



Vertex Shader

Un **Vertex Shader** è un programma che prende in ingresso una serie di informazioni su un vertice della scena, e da in uscita la **posizione di questo nel sistema di coordinate di vista**, e optionalmente, **il colore, il vettore normale alla superficie e le coordinate texture**.

Questi programmi aprono la strada per una vasta gamma di effetti, che prima dovevano essere elaborati nel livello applicativo della pipeline grafica come **deformazioni della geometria o distorsioni che simulano effetti ottici** come la rifrazione della luce attraverso diversi materiali.

Vertex Shader

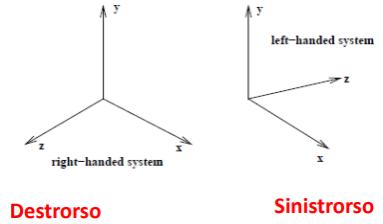
Prendono come input gli output del programma vertex shader e le texture map. Producono un colore finale e una trasparenza come output. Sono spesso chiamati shader di frammenti.

Geometria per la Computer Graphics

Sistemi di coordinate utilizzato nella grafica

In computer graphics si utilizza la convenzione dell'asse z uscente dallo schermo, l'asse z è perpendicolare allo schermo.

- **Destrоро**, se la rotazione attorno ad **y** che porta a **z** coincide con **x**, è antiorientata se vista dalla parte positiva di **y**
- **Sinistrоро**, se la rotazione è oraria



In generale ci baseremo sul sistema destrorso per la modellazione e sinistrorso per il rendering.

Spazi

In particolare ci sono due tipi di spazi:

- **spazi vettoriali lineari**, che contengono due tipi diversi di oggetti, gli scalari e i vettori, a cui si aggiunge il concetto di *prodotto interno*.
- **spazi affini**, che sono spazi vettoriali a cui si aggiunge il concetto di *punto*

Lo spazio Euclideo è un particolare esempio di spazio affine reale.

Scalare	Punto	Vettore
Specifica quantità	Entità il cui unico attributo è la sua posizione rispetto ad un sistema di riferimento	Entità i cui attributi sono lunghezza e direzione. Non ha una posizione nello spazio.

Spazio vettoriale/lineare

Sia R^n l'insieme dei vettori ad n componenti reali, gli elementi di R^n sono i vettori e gli scalari.

L'insieme R^n in cui si definiscono le operazioni di addizione tra due vettori e di moltiplicazione di un vettore per uno scalare è munito di una struttura di **spazio vettoriale sul campo R**.

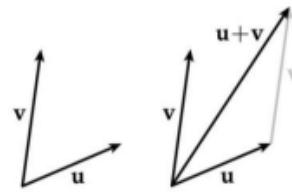
$$u \in R^n \quad v \in R^n \quad u + v = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{bmatrix} \quad \lambda u = \begin{bmatrix} \lambda u_1 \\ \lambda u_2 \\ \vdots \\ \lambda u_n \end{bmatrix}, \quad \lambda \in R$$

Le entità sono i **vettori** e gli **scalari**.

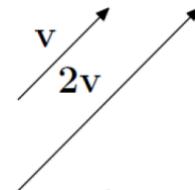
Operazioni

- Somma di due vettori, è data dalla regola del parallelogramma ed è commutativa:

$$u + v = v + u$$



- Moltiplicazione di uno scalare per vettore, cambia la lunghezza del vettore



Definizione di combinazione lineare

Dati k vettori, $v_1, v_2, \dots, v_k \in R + n$ e k scalari $\lambda_1, \lambda_2, \dots, \lambda_k \in R$, la quantità:

$$\lambda_1 v_1 + \lambda_2 v_2 + \dots + \lambda_k v_k$$

si dice **combinazione lineare** di dei vettori v_1, v_2, \dots, v_k con coefficienti $\lambda_1, \lambda_2, \dots, \lambda_k$

Definizione di indipendenza lineare

I vettori, v_1, v_2, \dots, v_k si dicono linearmente indipendenti se una loro combinazione lineare

$$\lambda_1 v_1 + \lambda_2 v_2 + \dots + \lambda_k v_k = 0$$

se e solo se $\lambda_i = 0$ $i = 1, 2, \dots, k$ cioè se nessuno di essi può essere ottenuto come combinazione lineare degli altri.

Dimensione dello spazio

In uno spazio vettoriale V , il numero massimo di vettori linearmente indipendenti è fissato ed è chiamato **dimensione** dello spazio.

In uno spazio a n dimensioni, un qualunque insieme di n vettori linearmente indipendenti forma **una base** per lo spazio.

Definizione di base

Data una base v_1, v_2, \dots, v_n , un qualunque vettore $v \in V$, può essere scritto come combinazione lineare

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n$$

dove gli α_i sono unici

Base canonica

È una base formata dai vettori:

$$e_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix} \leftarrow i \quad i = 1, \dots, n$$

Quando scriviamo un vettore in genere lo pensiamo rappresentato nella base canonica:

$$v = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \Leftrightarrow x_1 \cdot \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + x_2 \cdot \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \dots + x_n \cdot \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Definizione del prodotto scalare canonico

Siano $x, y \in R^n$, il loro prodotto scalare canonico è definito come:

$$x \cdot y := x_1 \cdot y_1 + x_2 \cdot y_2 + \dots + x_n \cdot y_n = \sum_{i=1}^n x_i \cdot y_i = x^T \cdot y$$

Per indicare il prodotto scalare canonico tra due vettori si può utilizzare anche la notazione equivalente $\langle x, y \rangle$

Proprietà

In generale, un prodotto scalare sullo spazio vettoriale R^n è un'applicazione da $R^n \times R^n$ a R che gode delle seguenti proprietà:

- $x \cdot y = y \cdot x \quad \forall x, y \in R^n$
- $x \cdot (\lambda_1 y_1 + \lambda_2 y_2) = \lambda_1 x \cdot y_1 + \lambda_2 x \cdot y_2 \quad \forall x, y_1, y_2 \in R^n, \quad \forall \lambda_1, \lambda_2 \in R$
- $x \cdot x > 0$ se $x \neq 0$

Ortogonalità di due vettori

Due vettori $x, y \in R^n$ si dicono **ortogonali** rispetto al prodotto scalare canonico se il loro prodotto scalare è nullo cioè:

$$\langle x, y \rangle = 0$$

I vettori $x = \begin{bmatrix} 3 \\ 0 \\ -1 \\ 1 \end{bmatrix}$ $y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix}$ sono ortogonali rispetto al prodotto scalare canonico,
infatti: $x \cdot y = 3 + 0 - 3 + 0 = 0$

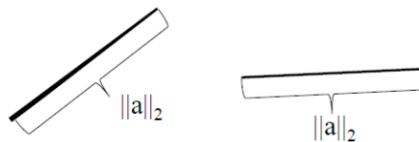
Norma di un vettore

Per misurare la lunghezza di un vettore si utilizza la **norma 2 o la norma euclidea**

Norma di un vettore

$$\|a\|_2 = \sqrt{\langle a, a \rangle} = \sqrt{a^T a} = \sqrt{\sum_{i=1}^n (a_i)^2} = \sqrt{a_1^2 + a_2^2 + \dots + a_n^2}$$

La norma euclidea fornisce una nozione di lunghezza che rimane **inalterata per rotazioni/traslazioni e riflessioni del vettore**



Un vettore con lunghezza 1 è detto **vettore unitario**.

Normalizzazione di un vettore

Dato un vettore qualsiasi lo si può **normalizzare per renderlo di lunghezza unitaria**, moltiplicandolo per il reciproco della sua lunghezza

$$\frac{a}{\|a\|_2}$$

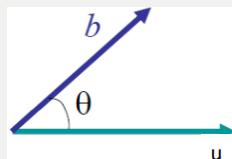
Un vettore normalizzato si dice anche **versore** e viene usato per individuare una direzione ed un verso; ogni vettore si può ottenere come prodotto della sua lunghezza per il versore che ha la stessa direzione e lo stesso verso.

$$a = \frac{a}{\|a\|_2} \cdot \|a\|_2$$

Prodotto scalare (prodotto interno o dot product)

Angolo tra due vettori

$$\begin{aligned} u \cdot b &= \|u\| \|b\| \cdot \cos \theta \\ \cos \theta &= \frac{u \cdot b}{\|u\| \|b\|} \\ \theta &= \arccos \left(\frac{u \cdot b}{\|u\| \|b\|} \right) \end{aligned}$$



Proiezione ortogonale e Interpretazione geometrica

Se $\|u\| \neq 1$, calcoliamo la lunghezza $\|a\|$ della proiezione di un vettore b sul vettore u .

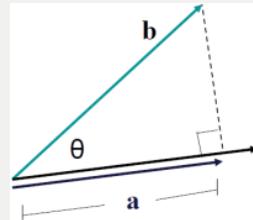
$\|a\|$: lunghezza della proiezione ortogonale di b su u

$$\|a\| = \|b\| \cos \theta$$

Sostituendo la precedente nella formula del prodotto scalare si ha:

$$u \cdot b = \|u\| \|b\| \cos \theta = \|u\| \|a\|$$

Il prodotto scalare tra due vettori u e b si può interpretare come il prodotto tra lunghezza del vettore u e la lunghezza della proiezione ortogonale del vettore b su esso.



Si può dedurre inoltre che:

$$\|a\| = \frac{\langle u, b \rangle}{\|u\|}$$

Se $\|u\| = 1$, si ha $\|a\| = \langle u, b \rangle$

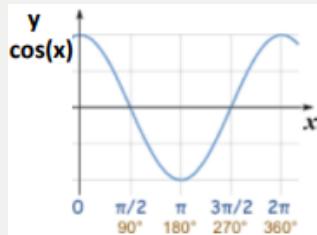
Prodotto scalare con vettori generici

Il prodotto scalare è un valore scalare che dà informazioni sulla relazione tra i due vettori.

$$a \cdot b = \|a\| \|b\| \cos \theta$$

In particolare il suo segno dà informazioni sull'angolo che formano i vettori, limitando il dominio a $[0, 2\pi]$:

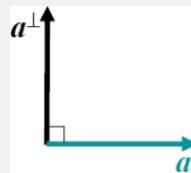
- se $a \cdot b > 0 \rightarrow 0 \leq \theta < \pi/2$ oppure $3\pi/2 < \theta < \pi$
- se $a \cdot b < 0 \rightarrow \pi/2 < \theta < 3\pi$
- se $a \cdot b = 0 \rightarrow \theta = \pi/2$ oppure $\theta = 3\pi/2$ oppure uno dei due vettori è $(0, 0, 0)$



Vettori perpendicolari

Dato un vettore $a = (a_x, a_y)$, il vettore ad esso **perpendicolare** è dato da:

$$a^\perp = \pm (-a_y, a_x)$$

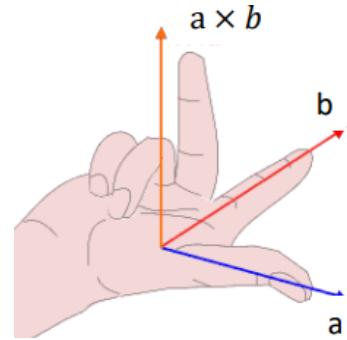


Cross Product

Il **prodotto esterno o vettoriale**, $a \times b$ è un vettore perpendicolare ad entrambi a e b ed ha la direzione definita dalla regola della mano destra.

Tale regola prevede di utilizzare le dita della mano destra e disporle come segue:

- il pollice deve seguire il verso del primo vettore del prodotto vettoriale
- l'indice segue il percorso del secondo vettore del prodotto vettoriale
- il medio andrà disposto perpendicolarmente al palmo della mano e definisce il verso del prodotto vettoriale



Proprietà del prodotto vettoriale

- $\|a \times b\| = \|a\| \|b\| \sin \theta$
- $\|a \times b\|$ è l'area del parallelogramma individuato dai due vettori
- $\|a \times b\| = 0$ se a e b sono paralleli

Siano e_1, e_2, e_3 i vettori della base canonica di R^3 . Per la base canonica (e_1, e_2, e_3) useremo la notazione più classica in meccanica $i = e_1, j = e_2, k = e_3$

$$a = a_x i + a_y j + a_z k \quad b = b_x i + b_y j + b_z k$$

Allora è facile verificare che il prodotto vettoriale $a \times b$ è il determinante della matrice:

$$A = \begin{bmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{bmatrix}$$

DIM

$$\begin{aligned} a \times b &= \det(A) = i(a_y b_z - a_z b_y) - j(a_x b_z - a_z b_x) + k(a_x b_y - a_y b_x) \\ &= [a_y b_z - a_z b_y, -a_x b_z + a_z b_x, a_x b_y - a_y b_x] = a \times b \end{aligned}$$

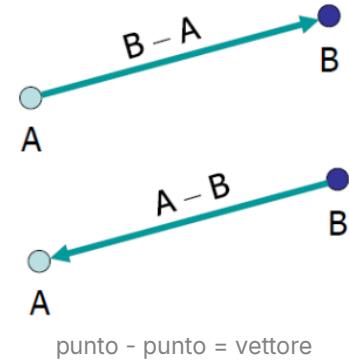
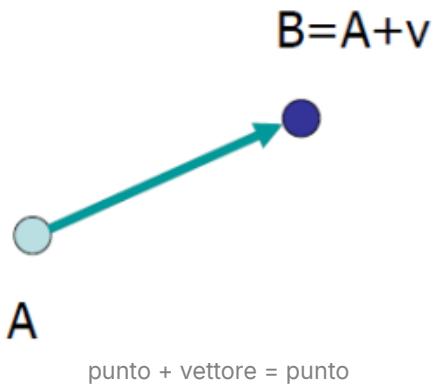
Spazi affini

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**.

Gli **spazi affini** sono un'estensione dello spazio lineare che include un'ulteriore tipo di entità: i **punti**.

Oltre a somma e moltiplicazione tra scalari, somma tra vettori, e moltiplicazione scalare-vettore, si introducono due nuove operazioni:

- **differenza punto-punto**, che definisce un vettore $v = P - Q$
- **somma punto+vettore**, $P + v$, che definisce un nuovo punto: $Q = P + v$



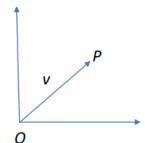
L'interpolazione geometrica è immediata; i **punti** sono **locazioni nello spazio** e la **differenza** di due punti è data dal **vettore che li congiunge**.

Mappare i punti in vettori

Se abbiamo un sistema di coordinate con origine O , possiamo definire una corrispondenza tra punti e vettori:

$$P \rightarrow v = P - O$$

$$v \rightarrow P = O + v$$



Coordinate baricentriche di un punto nello spazio affine

Una **combinazione affine** è una combinazione lineare di punti con coefficienti che hanno somma 1.

$$P = \alpha_0 P_0 + \alpha_1 P_1 + \cdots + \alpha_N P_N \quad \text{con } \alpha_0 + \alpha_1 + \cdots + \alpha_N = 1$$

I coefficienti $\alpha_0, \alpha_1, \dots, \alpha_N$ sono le **coordinate baricentriche**, affini, di P nello spazio affine.

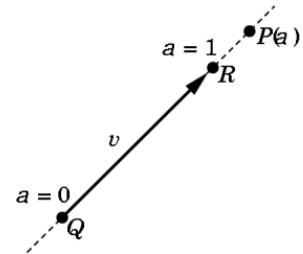
La combinazione affine di due punti distinti descrive la retta passante per i due punti.

Siano Q e R due punti nello spazio affine reale e sia v il vettore da essi individuato:

$$v = R - Q$$

Considerando la loro combinazione affine, $\alpha + \beta = 1$:

$$\begin{aligned} P &= \alpha R + \beta Q, \quad \beta = 1 - \alpha \\ P(\alpha) &= \alpha R + (1 - \alpha)Q \\ P(\alpha) &= Q + \alpha(R - Q) \\ P &= Q + \alpha v \end{aligned}$$

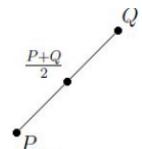


Combinazione convessa

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.

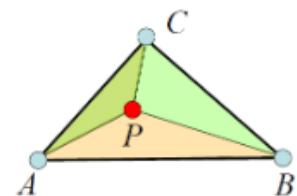
Se i pesi sono entrambi pari a 0.5, il punto risultante si trova a metà tra i due.

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.



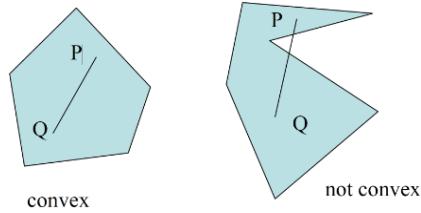
Coordinate baricentriche di un punto appartenente ad un triangolo

$$\begin{aligned} P &= \alpha_0 A + \alpha_1 B + \alpha_2 C \\ \alpha_0 + \alpha_1 + \alpha_2 &= 1, \alpha_i > 0, i = 0, 1, 2 \\ \alpha_0 &= \frac{\text{area}(PBC)}{\text{area}(ABC)} \quad \alpha_1 = \frac{\text{area}(PCA)}{\text{area}(ABC)} \quad \alpha_2 = \frac{\text{area}(PAB)}{\text{area}(ABC)} \end{aligned}$$



Convessità

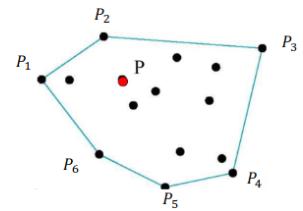
Un oggetto è **convesso** se e solo se comunque presi due punti nell'oggetto tutti i punti sul segmento di linea tra questi punti sono anche nell'oggetto.



Guscio Convesso

Dato un insieme di punti P_1, P_2, \dots, P_n ; l'insieme di tutti i punti P che possono essere rappresentati come combinazioni convesse è detto **inviluppo convesso** dell'insieme.

Il guscio convesso, **convex hull**, di un insieme di punti è la più piccola regione convessa che contiene i punti dati.



Distanza tra due punti

Distanza tra due punti nel piano cartesiano

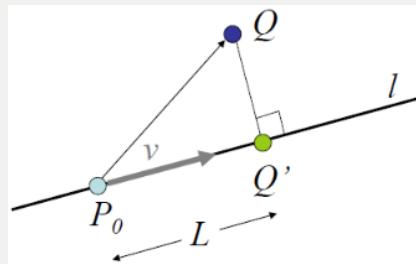
Distanza tra un punto ed una linea

Cercare un punto Q' , tale che $(Q - Q') \perp v$

$$dist(Q, l) = \|Q - Q'\|$$

Pitagora:

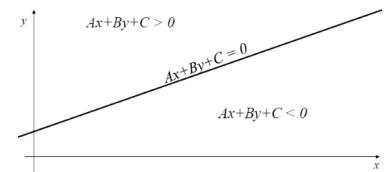
$$\begin{aligned} L^2 + dist(Q, Q')^2 &= \|Q - P_0\|^2 \\ L &= \frac{\langle Q - P_0, v \rangle}{\|v\|} \\ \Rightarrow dist(Q, Q')^2 &= \|Q - P_0\|^2 - L^2 = \|Q - P_0\|^2 - \frac{\langle Q - P_0, v \rangle^2}{\|v\|^2} \end{aligned}$$



Proiezione ortogonale di $Q - P_0$ su v

Equazione in forma implicita di una linea 2D

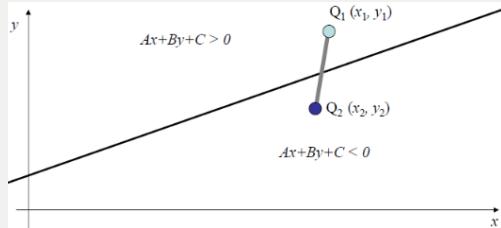
$$Ax + By + C = 0, \quad A, B, C \in R, \quad A, B \neq 0$$



Intersezione linea-segmento

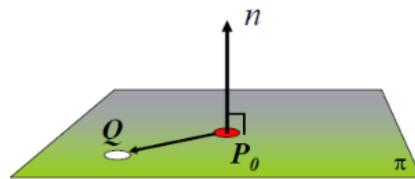
Il segmento Q_1, Q_2 interseca la linea se e solo se:

$$(Ax_1 + By_1 + C)(Ax_2 + By_2 + C) \leq 0$$



Rappresentazione di un piano nello spazio 3D

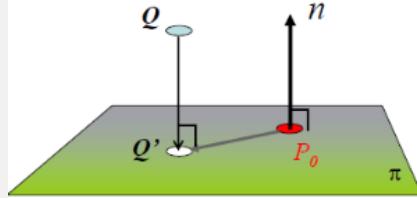
Un piano π è definito da una normale n ed un punto sul piano (P_0). Un punto Q appartiene al piano **se e solo se** $\langle Q - P_0, n \rangle \geq 0$. La normale n è normale a tutti i vettori nel piano.



Distanza tra un punto Q ed un piano π

Proiettare il punto Q sul piano nella direzione della normale.

$$dist(Q, \pi) = \|Q' - Q\|$$



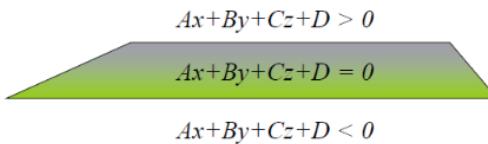
$$\begin{aligned} \|Q' - Q\|n &\Rightarrow Q' - Q = sn, s \in R \Rightarrow Q' = Q + sn \\ &\quad \langle Q' - P_0, n \rangle = 0 \\ &\quad \langle Q - P_0 + sn, n \rangle = 0 \\ &\quad \langle Q - P_0, n \rangle + s \langle n, n \rangle = 0 \\ &\quad s = -\frac{\langle P_0 - Q, n \rangle}{\|n\|^2} \end{aligned}$$

$$dist^2(Q, \pi) = \|Q' - Q\|^2 = s^2 \|n\|^2 = \frac{\langle Q - P_0, n \rangle^2}{\|n\|^2}$$

Rappresentazione implicita dei piano in 3D

(x, y, z) sono le coordinate di un punto sul piano; (A, B, C) sono le coordinate di un vettore normale al piano.

$$Ax + By + Cz + D = 0, \quad A, B, C, D \in R, \quad ABC \neq 0$$



Sistemi di riferimento

Una base con dimensione pari a 3 non basta per definire la posizione di un punto, ma occorre anche un punto di riferimento chiamato **l'origine del sistema di riferimento**.

Riferimento o Frame

Il concetto di **base** si estende a quello di **riferimento (frame)** in uno spazio affine specificando, oltre alla base, anche un punto P_0 detto **origine di riferimento**.

Dato un riferimento $F = (e_1, e_2, e_3, P_0)$, i punti ed i vettori dello spazio saranno esprimibili nel seguente modo:

$$\begin{aligned}P &= P_0 + v = P_0 + v_1 e_1 + v_2 e_2 + v_3 e_3 \\v &= v_1 e_1 + v_2 e_2 + v_3 e_3\end{aligned}$$

Gli scalari (v_1, v_2, v_3) sono le coordinate del punto P nel sistema di riferimento $F = (e_1, e_2, e_3, P_0)$.

Coordinate omogenee

Rappresentare sia i vettori che i punti usando tra scalari è ambiguo.

Un **vettore** è rappresentato mediante la terna (v_1, v_2, v_3) come:

$$v = v_1 e_1 + v_2 e_2 + v_3 e_3$$

Un **punto** P di coordinate (v_1, v_2, v_3) nel frame $F = (e_1, e_2, e_3, P_0)$ è rappresentato come:

$$P = v_1 e_1 + v_2 e_2 + v_3 e_3 + P_0$$

Vogliamo trovare un sistema di coordinate che porti ad una rappresentazione univoca per punti e vettori; facciamo la seguente assunzione: $1 * P = P$ e $0 * P = 0$.

Sotto questa ipotesi, un **vettore** è allora dato da:

$$v = v_1 e_1 + v_2 e_2 + v_3 e_3 + 0 \cdot P_0$$

Un **punto** è invece dato da:

$$P = v_1 e_1 + v_2 e_2 + v_3 e_3 + 1 \cdot P_0$$

Conclusione

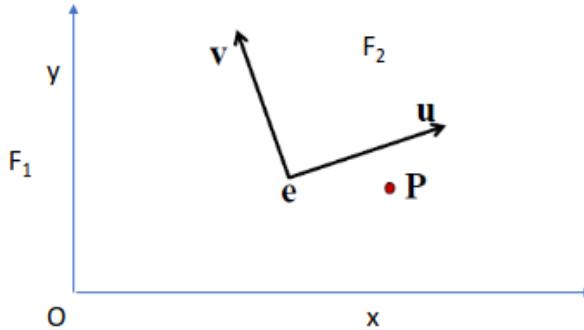
Quindi in uno spazio affine i punti sono rappresentati come **vettori riga** con ultima componente 1 e i vettori sono rappresentati come **vettori riga** con ultima componente 0.

$$P = [v_1 \quad v_2 \quad v_3 \quad 1] \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ p_0 \end{bmatrix} \quad v = [v_1 \quad v_2 \quad v_3 \quad 0] \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ p_0 \end{bmatrix}$$

Cambio di sistema di riferimento in 2D

Cambio delle coordinate di un vettore/punto da un sistema ad un altro.

Siano $F_1 = (x, y, O)$ ed $F_2 = (u, v, e)$ due frame di uno stesso spazio



Supponiamo di conoscere le coordinate del punto P nel frame F_2 e vogliamo vedere quali sono le sue coordinate nel frame $F_1 = (x, y, O)$

Esprimiamo il vettore e l'origine di F_2 in termini di vettori e origine di F_1 :

$$\begin{aligned} u &= x_u x + y_u y + 0 \cdot O \\ v &= x_v x + y_v y + 0 \cdot O \\ e &= x_e x + y_e y + 1 \cdot O \end{aligned}$$

In forma matriciale diventa

$$\begin{bmatrix} u \\ v \\ e \end{bmatrix} = \begin{bmatrix} x_u & y_u & 0 \\ x_v & y_v & 0 \\ x_e & y_e & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ O \end{bmatrix}$$

La matrice $M = \begin{bmatrix} x_u & y_u & 0 \\ x_v & y_v & 0 \\ x_e & y_e & 1 \end{bmatrix}$ rappresenta il cambiamento del sistema di riferimento da F_1 ad F_2 .

Il punto P nel frame $F_1 = (x, y, O)$ avrà coordinate omogenee $a^T = (x_p, y_p, 1)^T$ e si esprimerà come: $P = x_p x + y_p y + 1 \cdot O$.

Il punto P nel frame $F_2 = (u, v, e)$ avrà coordinate omogenee $b^T = (u_p, v_p, 1)^T$ e si esprimerà come: $P = u_p u + v_p v + 1 \cdot e$

P nel frame $F_1 = (x, y, O)$, in termini matriciali, si esprime come:

$$P = a^T \begin{bmatrix} x \\ y \\ O \end{bmatrix}$$

P nel frame $F_2 = (u, v, e)$, in termini matriciali, si esprime come:

$$P = b^T \begin{bmatrix} u \\ v \\ e \end{bmatrix}$$

$\Rightarrow \blacksquare$

$$\text{Ma } \begin{bmatrix} u \\ v \\ e \end{bmatrix} = M \begin{bmatrix} x \\ y \\ O \end{bmatrix}, \text{ quindi segue } a^T \begin{bmatrix} x \\ y \\ O \end{bmatrix} = b^T M \begin{bmatrix} x \\ y \\ O \end{bmatrix}$$

Segue che $a^T = b^T M$ e quindi $a = M^T b$ e $b = (M^T)^{-1} a$

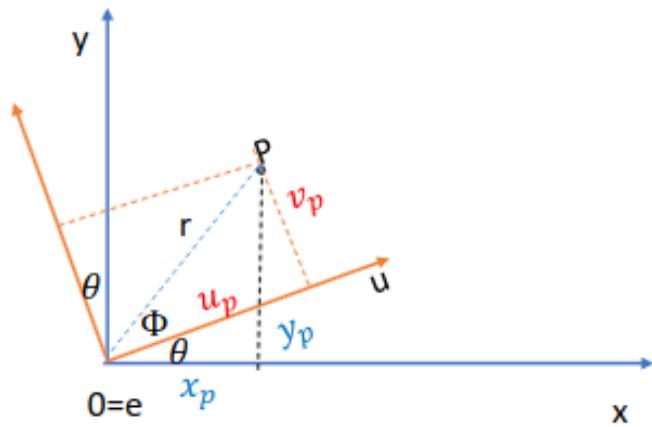
$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_e \\ y_u & y_v & y_e \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_e \\ y_u & y_v & y_e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix}$$

[https://virtuale.unibo.it/pluginfile.php/1840492/mod_resource/content/0/Geometria_per\(CG\).pdf](https://virtuale.unibo.it/pluginfile.php/1840492/mod_resource/content/0/Geometria_per(CG).pdf)

esempio pag 53

Esempio 2



$$F_1 = (x, y, O)$$

$$F_2 = (u, v, e)$$

punto P in $F_2 = (u, v, e)$ ha coordinate omogenee

$$b^T = (u_p, v_p, 1)^T$$

coordinate di P in

$$F_1 = (x, y, O)$$

$$a^T = (x_p, y_p, 1)^T ???$$

Esprimo P in coordinate polari nel sistema di riferimento $F_1 = (x, y, O)$

$$u_p = r \cos \phi \quad y_p = r \sin \phi$$

r: lunghezza del vettore che congiunge P con l'origine

ϕ : angolo che il vettore PO forma con l'asse x

Esprimiamo P in coordinate polari nel sistema di riferimento $F_2 = (u, v, e)$

$$\begin{aligned}x_p &= r \cos(\phi + \theta) = r \cos \phi \cos \theta - r \sin \phi \sin \theta \\y_p &= r \sin(\phi + \theta) = r \sin \phi \cos \theta + r \cos \phi \sin \theta\end{aligned}$$

Poiché vale $u_p = r \cos \phi$, $y_p = r \sin \phi$, le due formule precedenti diventano:

$$\begin{aligned}x_p &= u_p \cos \theta - y_p \sin \theta \\y_p &= v_p \cos \theta + u_p \sin \theta\end{aligned}$$

ed in termini matriciali si possono scrivere come:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

Cambio di sistema di riferimento in 3D

Siano $F_1 = (x, y, z, O)$ ed $F_2 = (u, v, w, e)$ due frame di uno stesso spazio, il punto P nel frame F_1 avrà coordinate omogenee $\rightarrow a = (x_p, y_p, z_p, 1)^T$ e si esprime come:

$$P = x_p x + y_p y + z_p z + 1 \cdot O$$

Il punto P nel frame F_2 avrà coordinate omogenee $b = (u_p, v_p, w_p, 1)^T$ e si esprime come:

$$P = u_p u + v_p v + w_p w + 1 \cdot O$$

La relazione che lega le coordinate nel punto P nel frame F_1 alle coordinate nel frame F_2 è la seguente:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_w & x_e \\ y_u & y_v & y_w & y_e \\ z_u & z_v & z_w & z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix}$$

Coordinate del vettore u rispetto al sistema di riferimento F_1

Trasformazioni geometriche in 2D

Ogni trasformazione geometrica complessa può essere decomposta in una concatenazione di trasformazioni geometriche elementari quali traslazione, scala e rotazione.

Queste **trasformazioni** modificano le coordinate di un oggetto per ottenerne un altro simile, ma differente per posizione, orientamento e dimensione. Modificando la geometria, MA non la

topologia degli oggetti.

Le trasformazioni vengono utilizzate per:

- posizionare gli oggetti nella scena, modeling
- cambiare la forma degli oggetti
- creare copie di multiple degli oggetti
- animazioni

Trasformazioni affini

Le trasformazioni geometriche sono lo strumento che consente di manipolare **punti e vettori** all'interno del mondo dell'applicazione grafica.

Le trasformazioni geometriche sono **funzioni** che mappano un punto in un altro punto.

La trasformazione di una primitiva geometrica si riduce alla trasformazione dei punti caratteristici, *vertici*, che la identificano nel rispetto della connettività originale.

Una **trasformazione geometrica affine** è una trasformazione **lineare**

$$f(aP + bQ) = af(P) + bf(Q)$$

L'importanza della proprietà di linearità sta nel fatto che, note le trasformazioni dei vertici, si possono ottenere le trasformazioni di combinazioni lineari dei vertici combinando linearmente le trasformazioni dei vertici.

Esse preservano:

- **collinearità**: i punti di una linea giacciono ancora su di una linea dopo la trasformazione
- **rapport tra le distanze**: il punto medio di un segmento rimane il punto medio di un segmento anche dopo la trasformazione

Note

Ogni oggetto a partire dal proprio sistema di riferimento, *object space*, viene trasformato opportunamente in un sistema di riferimento comune, *world space*, per andare a far parte dell'oggetto finale.

Trasformazione di Traslazione

Traslare una primitiva geometrica nel piano significa muovere ogni suo punto $P(x, y)$ di d_x unità lungo l'asse x e di d_y unità lungo l'asse y fino a raggiungere la nuova posizione $P'(x', y')$ dove:

$$x' = x + d_x, \quad y' = y + d_y$$

In notazione matriciale:

$$P' = P + T \quad \text{dove:} \\ P = \begin{bmatrix} x \\ y \end{bmatrix}; \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}; \quad T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Con T vettore di traslazione

Trasformazione di scalatura

Scelto un punto C , punto fisso di riferimento, scalare una primitiva geometrica significa riposizionare rispetto a C tutti i suoi punti in accordo ai fattori di scala s_x , lungo l'asse x , e s_y , lungo l'asse y .

Se il punto fisso è l'origine O degli assi,, la trasformazione di P in P' si ottiene con:

$$x' = s_x \cdot x, \quad y' = s_y \cdot y$$

In notazione matriciale:

$$P' = S \cdot P \quad \text{dove:} \\ P = \begin{bmatrix} x \\ y \end{bmatrix}; \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}; \quad S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

S pre-moltiplica P in quanto P è definito come vettore colonna

I fattori di scala inferiori a 1 avvicinano l'oggetto al punto fisso di riferimento, origine; invece i fattori di scala maggiori di 1 lo allontanano.

Se $s_x \neq s_y$ le proporzioni dell'oggetto non sono mantenute e si parla di **scalatura non uniforme**, invece se $s_x = s_y$ le proporzioni sono mantenute e si parla di **scalatura uniforme**.

Trasformazione di rotazione

Scelto un punto C , pivot di riferimento ed un verso di rotazione (orario, antiorario), ruotare una primitiva geometrica attorno a C significa muovere tutti i suoi punti nel verso assegnato in maniera che si conservi la **distanza** da C .

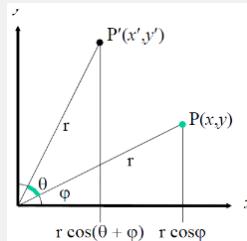
Una rotazione di θ attorno all'origine O degli assi è definita come:

$$x' = x \cdot \cos \theta - y \cdot \sin \theta, \quad y' = x \cdot \sin \theta + y \cdot \cos \theta$$

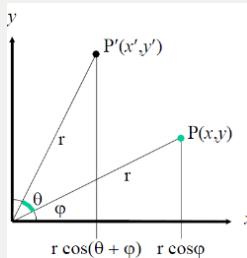
La relazione tra P' e P si ricava trigonometricamente; le coordinate di P possono essere espresse in coordinate polari:

$$x = r \cdot \cos \theta; \quad y = r \cdot \sin \theta$$

$$\begin{aligned} x' &= r \cdot \cos(\theta + \phi) \\ &= r \cdot \cos \theta \cdot \cos \phi - r \cdot \sin \theta \sin \phi \\ &= r \cdot \cos(\theta) \cdot \frac{x}{r} - r \cdot \sin(\theta) \cdot \frac{y}{r} \\ &= x \cdot \cos \theta - y \cdot \sin \theta \end{aligned}$$



$$\begin{aligned} y' &= r \cdot \sin(\theta + \phi) \\ &= r \cdot \sin \theta \cdot \cos \phi - r \cdot \cos \theta \sin \phi \\ &= r \cdot \sin(\theta) \cdot \frac{x}{r} - r \cdot \cos(\theta) \cdot \frac{y}{r} \\ &= x \cdot \sin \theta - y \cdot \cos \theta \end{aligned}$$



In notazione matriciale abbiamo:

$$P = \begin{bmatrix} x \\ y \end{bmatrix}; \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}; \quad R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

dove:

Note

Gli angoli sono considerati positivi quando misurati in senso **antiorario**, per le rotazioni di angoli negativi, senso orario, si ricorre alle identità:

$$\cos(-\theta) = \cos \theta; \quad \sin(-\theta) = -\sin(\theta)$$

Le coordinate omogenee

Le trasformazioni geometriche possono essere applicate in sequenza; la presenza di una somma di un punto con vettore:

$$P' = p + T$$

e di moltiplicazioni, scalatura e rotazione:

$$\begin{aligned} P' &= S \cdot P \\ P' &= R \cdot P \end{aligned}$$

rende disomogenea la concatenazione di trasformazioni: $P' = S(R(P + T)) + T$

Trasformazioni e coordinate omogenee

Nella notazione in coordinate omogenee possiamo riscrivere le trasformazioni geometriche di base come:

- *Trasformazione di traslazione*

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Concatenazione di due traslazioni

$$\begin{bmatrix} 1 & 0 & d_{x1} \\ 0 & 1 & d_{y1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & d_{x2} \\ 0 & 1 & d_{y2} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_{x1} + d_{x2} \\ 0 & 1 & d_{y1} + d_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

- *Trasformazione di scalatura*

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- *Trasformazione di rotazione*

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Altre trasformazioni: riflessione

- Riflessione rispetto all'asse x

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Riflessione rispetto all'asse y

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Riflessione rispetto all'origine degli assi

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Deformazione (shear)

- Riflessione rispetto all'asse x

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Riflessione rispetto all'asse y

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Riflessione rispetto ad entrambi gli assi

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

[https://virtuale.unibo.it/pluginfile.php/1840492/mod_resource/content/0/Geometria_per\(CG\).pdf](https://virtuale.unibo.it/pluginfile.php/1840492/mod_resource/content/0/Geometria_per(CG).pdf)

esempio pag 91

Composizione di Trasformazioni

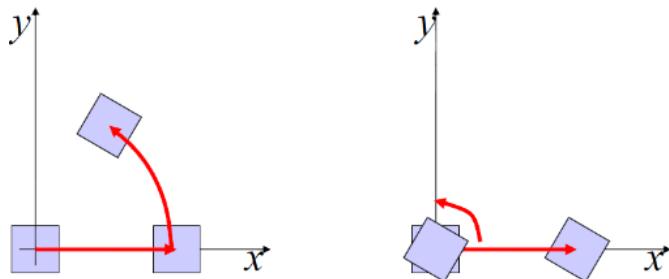
La rappresentazione in coordinate omogenee permette la concatenazione di trasformazioni; l'ordine di concatenazione è importante perché le trasformazioni geometriche sono associative ma non sono (in generale) commutative.

La corretta sequenza delle trasformazioni T_1, T_2, T_3 e T_4 si ottiene componendo T come:

$$T = T_4 \cdot T_3 \cdot T_2 \cdot T_1$$

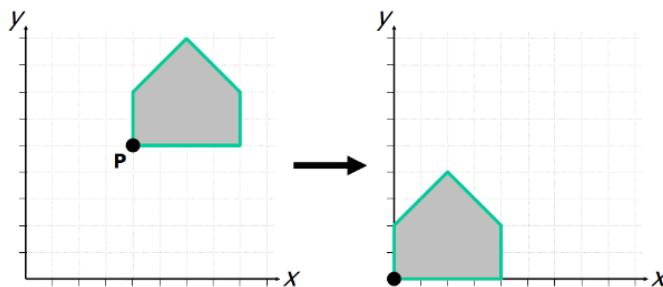
Non commutatività

La traslazione eseguita da rotazione attorno all'origine (sinistra) e rotazione intorno all'origine seguita da traslazione (destra).

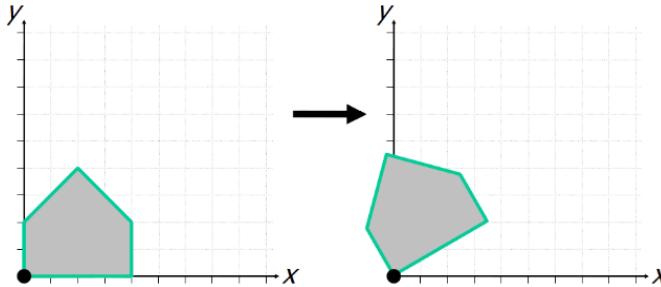


Rotazione oraria di un angolo θ attorno ad un punto P generico:

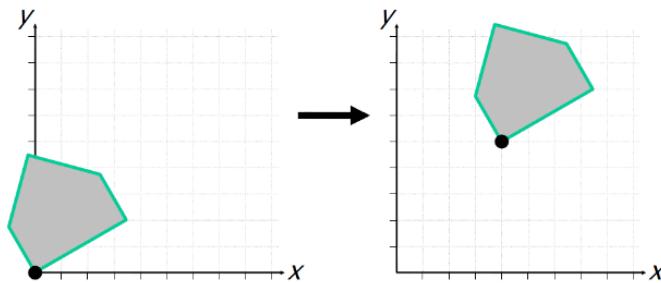
- traslazione che muove P nell'origine degli assi



- rotazione attorno all'origine



- traslazione opposta alla precedente che riporta P nella sua posizione originale



$$R_g = \begin{bmatrix} 1 & 0 & P_x \\ 0 & 1 & P_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -P_x \\ 0 & 1 & -P_y \\ 0 & 0 & 1 \end{bmatrix}$$

Trasformazione di scalatura attorno ad un punto P generico:

- trasformazione che muove P nell'origine degli assi
- trasformazione attorno all'origine
- traslazione opposta alla precedente che riporta P nella sua posizione originale

$$R_g = \begin{bmatrix} 1 & 0 & P_x \\ 0 & 1 & P_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -P_x \\ 0 & 1 & -P_y \\ 0 & 0 & 1 \end{bmatrix}$$

Invertibilità delle Trasformazioni

Sia M una trasformazione tra quelle che abbiamo visto. Poiché M non è singolare, allora esiste la matrice inversa $\$M^{-1}$ tale che: $MM^{-1} = I$, dove I rappresenta la **matrice identità**.

Allora applicare ad un punto

P' , trasformato di P mediante M , la matrice inversa della matrice di trasformazione M equivale a riottenere P .

Se

$P' = MP$, allora moltiplicando a destra e sinistra per M^{-1} otteniamo $P = M^{-1}P'$.

Fortunatamente per le trasformazioni viste le matrici inverse sono particolarmente semplici da calcolare: l'inversa della matrice di rotazione di R di un angolo ϑ è data dalla matrice di rotazione di un angolo $-\vartheta$: $R(-\vartheta) = R^T(\vartheta) = R^{-1}(\vartheta)$. (R infatti è una matrice ortogonale)

L'inversa della trasformazione di traslazione T di un vettore t è $T^{-1}(t) = T(-t)$

L'inversa della trasformazione di scala S è $S^{-1} = S(\frac{1}{s_x}, \frac{1}{s_y})$

Trasformazioni geometriche in 3D

Se tutte le trasformazioni nel piano possono essere rappresentate, in coordinate omogenee, da matrici 3×3 , le trasformazioni nello spazio possono essere rappresentate da matrici 4×4 .

Nello spazio un punto in coordinate omogenee è rappresentato da una quadrupla:

$$(X, Y, Z, W)$$

Traslazione

La matrice di traslazione in 3D è una semplice estensione della matrice 2D

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scalatura 3D

Anche la matrice di scalatura è una semplice estensione della matrice 2D

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotazione 3D

La matrice che descrive la rotazione generica dello spazio è molto più complessa. Ci viene in aiuto la proprietà che qualunque rotazione 3D si può ottenere come composizione di 3 rotazioni attorno agli assi coordinati.

Rotazione intorno all'asse x

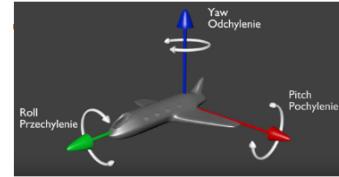
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotazione intorno all'asse y

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotazione intorno all'asse z

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Rollio (Roll)= rotazione intorno all'asse Z
Beccheggio (Pitch)= rotazione intorno all'asse X
Imbardata (Yaw) = rotazione intorno all'asse Y

[https://virtuale.unibo.it/pluginfile.php/1840492/mod_resource/content/0/Geometria_per\(CG\).pdf](https://virtuale.unibo.it/pluginfile.php/1840492/mod_resource/content/0/Geometria_per(CG).pdf)

Formule di Rodrigues per la rotazione di un angolo θ intorno ad un asse generico ω

Curve interpolanti Hermite

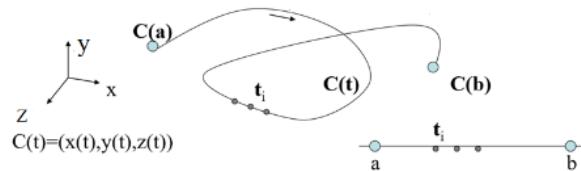
Curve parametriche

Una curva parametrizzata in R^3 è un'applicazione

$$\begin{aligned} C(t) : I = [a, b] \subseteq R &\rightarrow R^3 \\ t &\rightarrow (x(t), y(t), z(t)) \end{aligned}$$

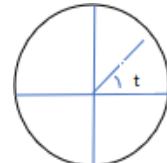
dove $x(t)$, $y(t)$ e $z(t)$ sono funzioni continue dal **parametro** $t \in [a, b] \subseteq R$, dette **componenti parametriche** della curva.

Al variare di t le coordinate $(x(t), y(t), z(t))$ individuano un punto che si sposta sulla curva



Circonferenza di centro l'origine e raggio r

$$C(t) = \begin{cases} x(t) = r \cos(t) \\ y(t) = r \sin(t) \quad t \in [0, 2\pi] \end{cases}$$



Equazione parametrica del segmento che congiunge due punti

$$P(t) = P_0 + t(P_1 - P_2)$$

$$P(t) = \begin{cases} x = x_0 + t(x_1 - x_0) \\ y = y_0 + t(y_1 - y_0) & t \in [0, 1] \end{cases}$$

$P_1 = (x_1, y_1)$
 $P_0 = (x_0, y_0)$

L'intervallo $I = [a, b]$ definisce un insieme di punti in cui si sceglie di visualizzare la curva anche se essa vive per valori al di fuori da questo intervallo: $(-\infty, +\infty)$.

L'immagine in R^3 tramite C dell'intervallo $I = [a, b] \subseteq R$, $C(I)$, prende il nome di **supporto**, **sostegno** o **traiettoria della curva**.

Una curva si dice **regolare** se è **differenziabile** per ogni valore di $t \in I$ e se **la norma del vettore derivata non è nulla in alcun punto dell'intervallo I** .

Lunghezza di una curva

Sia $C(t) : [a, b] \rightarrow R^3$ una curva regolare, consideriamo una suddivisione dell'intervallo $[a, b]$, mediante un insieme di punti $t_1 = a < t_2 < \dots < t_{n-1} < t_n = b$



Per ogni valore del parametro t_i , $i = 1, \dots, N$ consideriamo i il punto $C(t_i)$, $i = 1, \dots, N$ sulla curva.

Consideriamo la poligonale P_N che collega i vertici $C(t_i)$, $i = 1, \dots, N$ sulla curva.

La lunghezza di questa poligonale è data dalla **somma delle lunghezze dei suoi lati**:

$$L(P_N) = \sum_{i=1}^{N-1} \|C(t_{i+1}) - C(t_i)\|$$

Al tendere all'infinito del numero dei punti delle suddivisioni di $[a, b]$ così ottenute, le lunghezze delle poligonali associate formano una successione monotona non decrescente che converge all'integrale che rappresenta quindi la lunghezza della traiettoria

$$L(C) = \lim_{N \rightarrow \infty} L(P_N) = \sum_{i=1}^{N-1} \|C(t_{i+1}) - C(t_i)\| = \int_a^b \|C'(t)\| dt$$

Siano $x'(t)$, $y'(t)$ e $z'(t)$ continue in $[a, b]$ le derivate prime delle componenti parametriche della curva $C(t) = (x(t), y(t), z(t))$, allora la lunghezza di C tra $C(a)$ e $C(b)$ è definita da:

$$L = \int_a^b \|C'(t)\| dt = \int_a^b \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2} dt$$

Continuità geometrica e continuità di una curva

Se due segmenti di curva si uniscono ad un estremo, si dice che tra i due segmenti di curva c'è un **raccordo** C^0 che assicura l'assenza di salti. Se due tratti di curva si uniscono in un punto P_0 e detti v_1 e v_2 i vettori velocità in P_0 del primo e secondo tratto di curva rispettivamente, si definisce:

- **continuità parametrica** C^1 , le direzioni e i moduli dei vettori tangenti v_1 e v_2 dei due segmenti curvi nel punto di contatto P_0 sono uguali
- **continuità geometrica** G^1 , le direzioni dei vettori tangenti v_1 e v_2 dei due segmenti curvi nel punto di contatto sono uguali, i moduli possono essere diversi



- **continuità parametrica** C^n , le derivate fino a quella di ordine n dei due segmenti di curva C_1 e C_2 nel punto di contatto P_0 sono uguali

$$\frac{d^n C_1(t)}{dt^n} \Big|_{P_0} = \frac{d^n C_2(t)}{dt^n} \Big|_{P_0}$$

Curvatura

Ciò che vogliamo definire e calcolare è la misura matematica di quanto una curva devii dall'essere retta nell'intorno di un punto. Poiché la curvatura di una curva in generale cambia da punto a punto, si calcola come la variazione del versore tangente rispetto ad una variazione nella posizione.

Sia una curva regolare di classe C^k , $k \geq 2$, definiamo il versore della tangente:

$$T(t) = \frac{C'(t)}{\|C'(t)\|}$$

Sia C una curva regolare di classe C^k , $k \geq 2$, la **curvatura** è la funzione:

$$k(t) : I \rightarrow R^+$$

di classe C^{k-2} , che calcola la variazione del versore tangente rispetto ad una variazione nella posizione $k(t) = \frac{\|T'(t)\|}{\|C'(t)\|}$

Sia $C(t)$ una circonferenza di raggio R

$$C(t) = \begin{bmatrix} R \cos(t) \\ R \sin(t) \end{bmatrix}, C'(t) = \begin{bmatrix} -R \sin(t) \\ R \cos(t) \end{bmatrix}, \\ \|C'(t)\|_2 = \sqrt{R^2(\sin^2(t) + \cos^2(t))} = R$$

Calcoliamo il *versore della normale*:

$$T(t) = \frac{C'(t)}{\|C'(t)\|_2} = \frac{\begin{bmatrix} -R \sin(t) \\ R \cos(t) \end{bmatrix}}{\|C'(t)\|_2} = \begin{bmatrix} -\sin(t) \\ \cos(t) \end{bmatrix}$$

Curvatura

$$k(t) = \frac{\|T'(t)\|}{\|C'(t)\|} = \frac{\left\| \begin{bmatrix} -\cos(t) \\ -\sin(t) \end{bmatrix} \right\|}{R} = \frac{\sqrt{\cos^2 t + \sin^2 t}}{R} = \frac{1}{R}$$

Curve interpolanti di Hermite

Curve Interpolanti

Dati i punti del piano P_i , $i = 0, 1, \dots, N$, dove possiamo pensare che le coordinate del punto P_i possano essere considerate come valutazione di due funzioni parametriche $x(t)$ e $y(t)$ nel valore del parametro t_i

$$P_i \equiv \begin{pmatrix} x_i = x(t_i) \\ y_i = y(t_i) \end{pmatrix}$$

Vogliamo costruire la curva interpolante dei punti P_i , $i = 0, 1, \dots, N$

$$C(t_i) = p_i \equiv \begin{pmatrix} x_i = x(t_i) \\ y_i = y(t_i) \end{pmatrix}$$

Questo equivale a risolvere due problemi di interpolazione, uno per la funzione parametrica x e l'altro per la funzione parametrica y . Scegliamo una base di funzioni φ per lo spazio in cui vogliamo costruire le funzioni interpolanti.

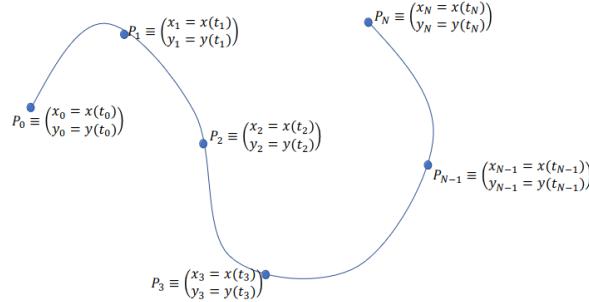
Interpoliamo quindi le coppie (t_i, x_i) , $i = 0, 1, \dots, N$

$$X_N(t) = \sum_{j=0}^N \alpha_j \varphi_j(t) \text{ tale che } X_N(t) = x_i$$

Interpoliamo quindi le coppie (t_i, y_i) , $i = 0, 1, \dots, N$

$$Y_N(t) = \sum_{j=0}^N \beta_j \varphi_j(t) \text{ tale che } Y_N(t) = y_i$$

$$C(t) = \begin{cases} X_N(t) \\ Y_N(t) \end{cases} \implies C(t_i) = p_i = \begin{pmatrix} x_i = x(t_i) \\ y_i = y(t_i) \end{pmatrix}$$

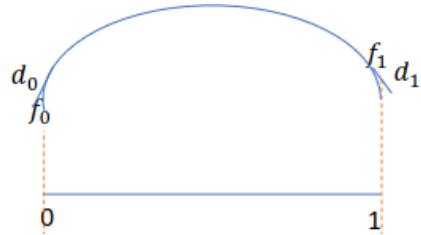


Curve Interpolanti di Hermite

Dati $N + 1$ punti da interpolare $p_i = \begin{pmatrix} x_i = x(t_i) \\ y_i = y(t_i) \end{pmatrix}$ $i = 0, 1, \dots, N$

costruire N segmenti di curve costituite a partire da polinomi cubici, tali che nei punti di giunzione abbiano lo stesso valore e la stessa derivata prima

Assegnati agli estremi dell'intervallo $[0, 1]$ i valori f_0 ed f_1 della funzione ed i valori d_0 ed d_1 della derivata prima



il polinomio cubico $P_3(t) = a_0 + a_1t + a_2t^2 + a_3t^3 = [1 \ t \ t^2 \ t^3] \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$ che interpola questi dati, cioè:

$$\begin{aligned} P_3(0) &= f_0 & P_3(1) &= f_1 \\ P'_3(0) &= d_0 & P'_3(1) &= d_1 \end{aligned}$$

Si calcola imponendo le condizioni di interpolazione:

$$\begin{aligned} P_3(0) &= a_0 = f_0 \\ P_3(1) &= a_0 + a_1 + a_2 + a_3 = f_1 \\ &\quad a_1 = d_0, a_2 = 2d_0, a_3 = 3d_0 \end{aligned} \qquad P'_3(t) = a_1 + 2a_2t + 3a_3t^2$$

Hermite ha dimostrato che il polinomio $P_3(t) = a_0 + a_1t + a_2t^2 + a_3t^3$ che interpola questi dati, cioè:

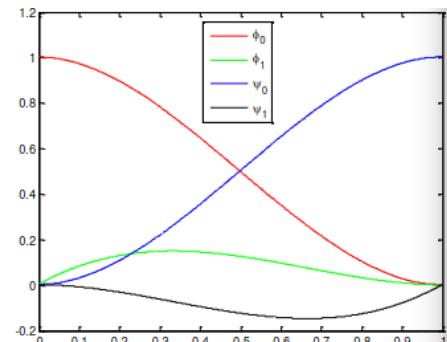
$$\begin{aligned} P_3(0) &= f_0 & P_3(1) &= f_1 \\ P'_3(0) &= d_0 & P'_3(1) &= d_1 \end{aligned}$$

si può esprimere come:

$$P_3(t) = f_0 \cdot \phi_0(t) + d_0 \cdot \phi_1(t) + f_1 \psi_0(t) + d_1 \psi_1(t)$$

dove

$$\begin{aligned} \phi_0(t) &= 2 \cdot t^3 - 3 \cdot t^2 + 1 \\ \phi_1(t) &= t^3 - 2 \cdot t^2 + t \\ \psi_0(t) &= -2 \cdot t^3 + 3 \cdot t^2 \\ \psi_1(t) &= t^3 - t^2 \\ 0 \leq t &\leq 1 \end{aligned}$$



Dimostrazione

da finire

Rendering Pipeline: Geometric Stage

Definita una camera virtuale e una scena tridimensionale, la **pipeline di rendering** costituisce una serie di **trasformazioni** che proiettano la **scena tridimensionale** in un'immagine in **una finestra contenuta nello schermo bidimensionale**.

Questo stadio della rendering pipeline può essere vista come una **pipeline di trasformazioni geometriche o trasformazioni di sistema di coordinate**.

Un modello geometrico è infatti trasformato mediante **trasformazioni di modellazione, vista, proiezione e viewport**, ovvero trasformazioni tra vari sistemi di riferimento.

WCS

Inizialmente il modello geometrico viene creato nello **spazio locale del modello**. Viene quindi posizionato ed orientato in scena nel **World Coordinate System**, WCS, mediante **trasformazioni di modellazione** sui vertici del modello.

Il **WCS** è unico e dopo che tutti i modelli geometrici necessari per creare una scena sono stati posizionati tutti i modelli sono rappresentati in questo spazio.

VCS

Poiché solo i modelli "visualizzati" dalla camera virtuale sono resi, tutti i modelli sono quindi trasformati mediante una *trasformazione di vista* nel sistema di riferimento della camera o camera frame, **View Coordinate System** (VCS).

Dal centro del sistema della camera solo una porzione di volume, detto **volume di vista**, contiene la scena che è visibile all'osservatore. Quindi il sistema di rendering elabora la **trasformazione di proiezione** per trasformare il volume di vista contenente la scena visibile dall'osservatore, in un cubo di lato 2 e diagonale di estremi $(-1, -1, -1)$ e $(1, 1, 1)$; le **proiezioni** sono in generale ortografica o prospettica, entrambe possono essere costruite con matrici 4×4 . **Queste proiezioni trasformano un volume di vista in un altro volume**.

La vera e propria proiezione dal volume di vista al piano di vista viene infine effettuata **mediante una proiezione ortogonale su una faccia del cubo** che conserva quindi le coordinate

x ed y , memorizzando opportunamente la coordinata z , che rappresenta la profondità dei vertici, in una speciale memoria del frame buffer, detta **Z-buffer**.

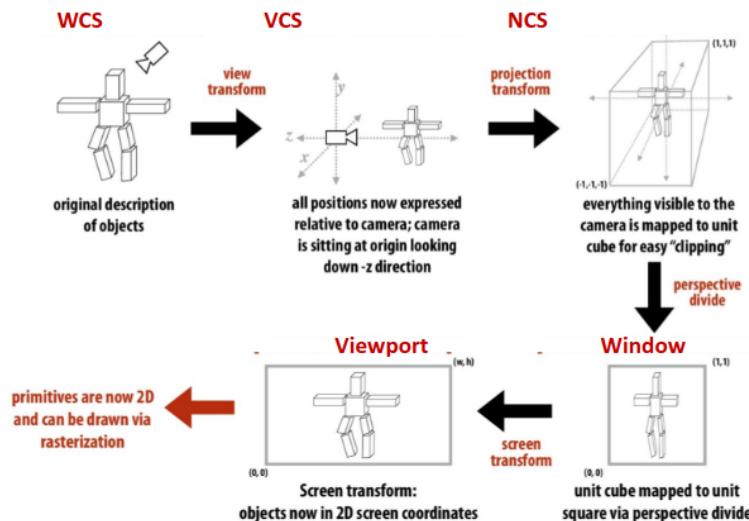
Questo speciale passaggio intermedio da **volume di vista 3D a cubo** serve per facilitare le **operazioni di clipping**. In breve le primitive risiedono interamente all'interno del volume di vista verranno disegnate sullo schermo, quelle completamente esterne verranno scartate, mentre quelle parzialmente contenute nel volume di vista verranno eliminate.

Solo le primitive interne al volume di vista sono passate all'ultima trasformazione del *geometry stage*, la **trasformazione di viewport**.

Quest'ultima converte le coordinate (x, y) reali di ogni vertice, in coordinate schermo espresse in pixel.

Geometric Stage

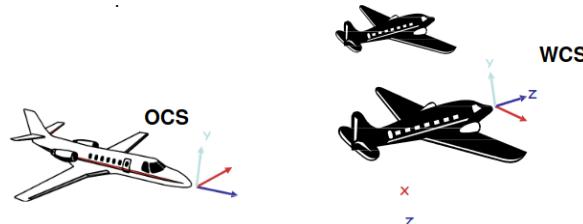
1. Composizione degli oggetti in scena in un sistema di riferimento destroverso WCS
2. WCS \Rightarrow VCS, trasformazione delle coordinate della scena da WCS al sistema di coordinate della camera VCS sinistrorso
3. Volume di vista: porzione dello spazio WCS che risulta visibile all'osservatore; Clipping: tutto quello che si trova esterno al volume risulta tagliato e quindi non visibile
4. Trasformazione spazio immagine: dal volume di vista al cubo; infine una **proiezione ortogonale proietta 3D \Rightarrow 2D su un piano immagine 2D**
5. Trasformazione di *viewport window* \Rightarrow *viewport* trasforma l'immagine contenuta nella window in coordinate *pixel viewport*



Trasformazioni di Modellazione

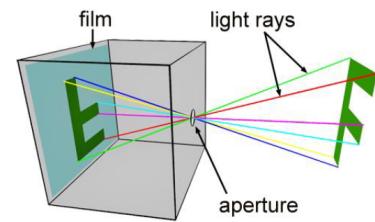
Ogni oggetto è definito in un suo sistema di coordinate, *Object Coordinate System*. Le **trasformazioni di modellazione** permettono di **muovere**, **orientare** e **trasformare** modelli all'interno di un sistema di riferimento comune, sistema di coordinate del mondo, WCS.

Si moltiplica ogni vertice per una matrice di trasformazione affine 4×4 chiamata T_m



Pinhole camera

Una **pinhole** camera è il modello più semplice di camera virtuale; una scatola con un piccolissimo foro su una parete, detto *pinhole*, che permette alla luce di passare. La luce di una scena passa attraverso l'apertura e proietta un'immagine capovolta sul lato opposto della scatola, dove è posizionata una pellicola fotosensibile.



La dimensione delle immagini dipende dalla distanza tra l'oggetto e il foro.

Semplificazioni

Una semplificazione che si fa è quella di **spostare il piano su cui avviene la proiezione dell'altro lato rispetto al pinhole**, così da evitare che l'immagine sia generata capovolta. Il pinhole diventa quindi l'occhio dell'osservatore, mentre la pellicola diventa la finestra attraverso cui passano i raggi che chiameremo *image plane*.

Trasformazione di vista

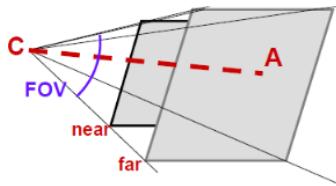
Per costruire una trasformazione di vista è necessario **posizionare la camera nel WCS e orientarla opportunamente**, questo permette di generare la **matrice del cambiamento di sistema di riferimento**. Questa matrice di trasformazione sarà utilizzata per **trasformare ogni vertice degli oggetti in scena dal WCS al VCS**.

- **Specifica la vista 3D**, impostazione della fotocamera sintetica orientata e posizionata in WCS
- **Costruzione della Trasformazione di Vista** a partire dal posizionamento ed orientamento della telecamera, creazione di una matrice di trasformazione T_v
- **Applicazione della trasformazione** ad ogni vertice dell'oggetto

Definire la vista: camera look at

Per definire la vista della telecamera bisogna sapere quattro informazioni sul modello di fotocamera sintetica per costruire la trasformazione della vista:

- Punto C , posizione della telecamera in WCS
- Punto A , il vettore A specifica in quale direzione sta puntando la telecamera
- *field of view*, FOV \Rightarrow indica l'angolo di visibilità verticale nella scena espresso in radianti, maggiore è questo valore, maggiore sarà la porzione di scena visualizzata; è possibile regolare lo zoom con questo parametro
- *depth of field*, distanza tra l'oggetto più vicino e quello più lontano che appaiono perfettamente a fuoco



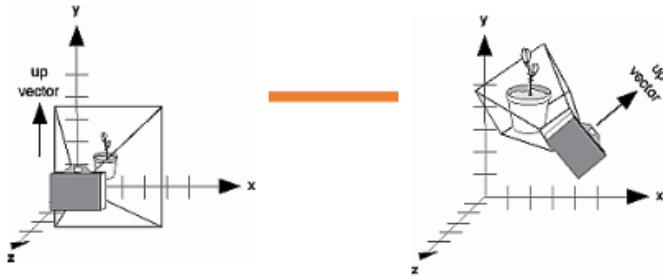
Definiamo un sistema di riferimento, VCS, associato all'osservatore di origine C , che stabilisce la posizione della camera $F(C, u, v, w)$:

- $C \Rightarrow$ posizione della telecamera
- asse $w \Rightarrow$ la direzione unitaria verso cui punta la fotocamera per convenzione, la telecamera guarda in direzione $-w$

$$w = \frac{C - A}{\|C - A\|}$$

- **View Up Vector**, VUP, determina il modo in cui la fotocamera viene ruotata attorno alla direzione di vista
- asse $u \Rightarrow$ asse unitario che punta alla destra dell'osservatore, è perpendicolare sia all'asse w che al vettore up VUP

$$u = \frac{VUP \times w}{\|VUP \times w\|}$$



Nel caso in cui VUP || all'asse y , il prodotto vettoriale si semplifica come: $[0, 1, 0] \times w = [w_z, 0, -w_x]$

- asse $v \Rightarrow$ è ortogonale sia all'asse w che all'asse y , quindi: $v = w \times u$

Poiché w e u sono normalizzati, anche v risulta normalizzato

$$|v| = |u||v| \cos(90^\circ) = 1$$

Costruzione della matrice di Trasformazione di Vista

Si esprime il sistema di riferimento della camera VCS (C, u, v, w) in termini del sistema WCS (O, x, y, z):

$$\begin{aligned} u &= u_x x + u_y y + u_z z + 0 \cdot O \\ v &= v_x x + v_y y + v_z z + 0 \cdot O \\ w &= w_x x + w_y y + w_z z + 0 \cdot O \\ C &= C_x x + C_y y + C_z z + 1 \cdot O \end{aligned}$$

La matrice trasposta M^T = $\begin{bmatrix} u_x & v_x & w_x & C_x \\ u_y & v_y & w_y & C_y \\ u_z & v_z & w_z & C_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$ **mappa le coordinate di un punto del sistema VCS nelle coordinate del sistema WCS.**

Se un punto nel frame WCS ha coordinate omogenee $P_w = (x_p, y_p, z_p, 1)$ e nel frame VCS ha coordinate omogenee $P_v = (u_p, v_p, w_p, 1)$ allora:

$$P_w = MP_v$$

La matrice M^T ci fornisce il cambio di coordinate da frame VCS a sistema WCS, a noi serve la matrice inversa $(M^T)^{-1}$ da WCS a VCS che chiameremo T_v :

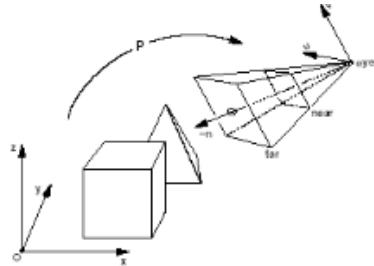
$$P_v = (M^T)^{-1} P_w = T_v P_w$$

che determina la rappresentazione di un punto P_v in coordinate omogenee in VCS data la sua rappresentazione in WCS.

$$\begin{aligned}
 \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} &= \begin{bmatrix} x_u & x_v & x_w & x_c \\ y_u & y_v & y_w & y_c \\ z_u & z_v & z_w & z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix} \\
 P_{xyz} &= \begin{bmatrix} u & v & w & C \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{uvw} \\
 P_{xyz} &= M P_{uvw}
 \end{aligned}
 \quad
 \begin{aligned}
 \begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix} &= \begin{bmatrix} x_u & x_v & x_w & x_c \\ y_u & y_v & y_w & y_c \\ z_u & z_v & z_w & z_c \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \\
 P_{uvw} &= \begin{bmatrix} u & v & w & C \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} P_{xyz} \\
 P_{uvw} &= M^{-1} P_{xyz}
 \end{aligned}$$

Projection transform

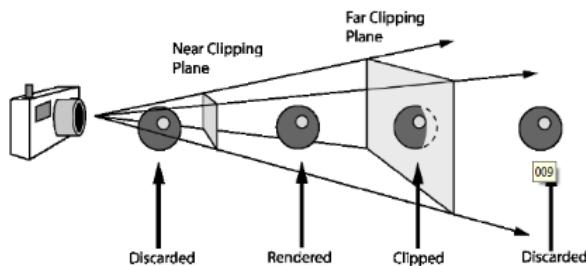
- INPUT: vertici in 3D in **VCS**
- OUTPUT: coordinate di schermo 2D dei vertici visibili nello **Screen space**



Clipping

Il volume di spazio tra i piani di clipping anteriore e posteriore definisce cosa può vedere la telecamera.

- La posizione dei piani è definita dalla distanza lungo la direzione di vista
- Gli oggetti che appaiono al di fuori del volume della vista non vengono disegnati
- Gli oggetti che intersecano il volume della vista vengono tagliati

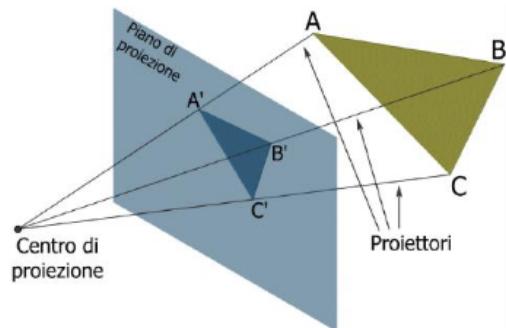




Proiezioni

Si dice **proiezione** una trasformazione geometrica con il dominio in uno spazio di dimensione n ed il codominio in uno spazio di dimensione $n - 1$ o inferiore. A noi interessano solo le proiezioni da 3 a 2 dimensioni

La proiezione di un oggetto 3D è definita da un insieme di **rette di proiezione**, dette **proiettori**, aventi origine comune da un centro di proiezione che passano per tutti i punti dell'oggetto e **intersecano un piano di proiezione per formare la proiezione vera e propria**.



Questo tipo di proiezioni, caratterizzate dal fatto che:

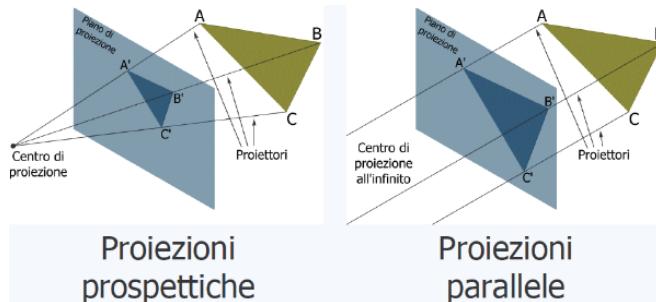
- i proiettori sono rette
- la proiezione è su di un piano

si chiamano **proiezioni geometriche piane**.

Esistono due classi in cui si possono suddividere le proiezioni geometriche piane:

- **prospettive**
- **parallele**

La differenza tra le due classi è data dalla distanza tra il centro di proiezione ed il piano di proiezione: se tale distanza è finita la proiezione è prospettica, se è infinita è parallela.



La proiezione prospettica è la più **realistica**, in quanto riesce a riprodurre il modo con cui nella realtà vediamo gli oggetti: **oggetti più grandi sono più vicini all'osservatore, oggetti più piccoli sono più lontani.**

Le proiezioni parallele fanno sì che **linee parallele nel modello tridimensionale rimangono tali nella proiezione**. Sono utilizzate nel disegno tecnico perché si ha la necessità di compiere misurazioni sul risultato della proiezione

Proiezioni Prospettiche

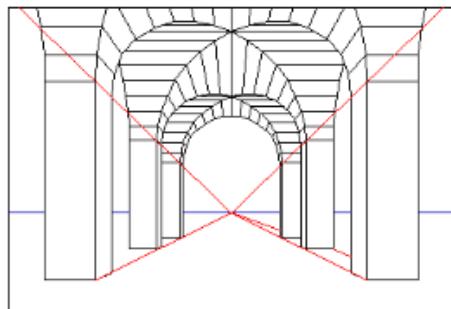
Sono caratterizzate da **vanishing point**; la proiezione di ogni insieme di linee parallele non parallele al piano di proiezione converge in un punto detto **vanishing point, punto di convergenza**.

Il numero di questi punti è infinito, come il numero delle possibili direzioni di fasci di rette parallele.

Se l'insieme di linee parallele è a sua volta **parallelo ad uno degli assi coordinati** il punto di convergenza si chiama **axis vanishing point**. Di questi punti è possibile averne al massimo tre.

Le proiezioni si classificano in base al numero di vanishing point principali, che è il numero di assi del sistema di coordinate che intersecano il piano di proiezione.

Se il piano di vista è parallelo ad **un piano coordinato** la prospettiva si dice ad 1 punto di fuga.



1 Punti di fuga

Se invece è parallelo ad un asse coordinato presenta **2 punti di fuga**

Proiezioni Parallelle

Si classificano in base alla relazione che c'è tra la direzione di proiezione e la normale al piano di proiezione.



Ortografica

direzione di proiezione coincide con la normale al piano di proiezione si parla di *proiezione ortografiche*.

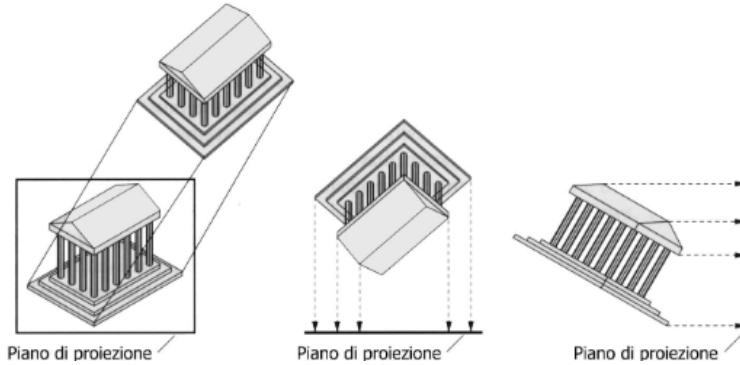
Obliqua



Nel caso contrario si parla di *proiezione obliqua*.

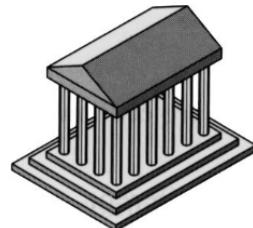
Proiezioni ortografiche assonometriche

Sono ancora proiezioni **ortografiche**, ma non essendo la **direzione di proiezione allineata con un asse principale**, si mostrano facce diverse di un oggetto, assomigliando in questo caso alle proiezioni prospettiche.



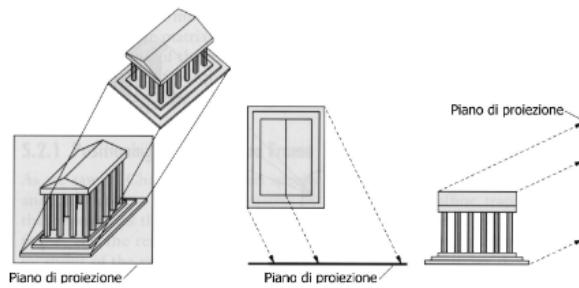
Proiezione Isometrica

La direzione di proiezione è identificata da una delle bisettrici degli ottanti dello spazio cartesiano.



Proiezioni oblique

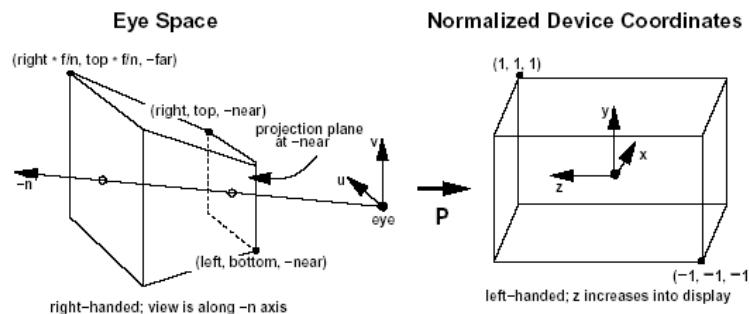
Sono il tipo più generale di proiezioni parallele, caratterizzate dal fatto che i **proiettori possono formare un angolo qualsiasi con il piano di proiezione**.



- **Cavaliera**, in cui la direzione di proiezione forma un angolo di 45° con il piano di proiezione e quindi linee ortogonali al piano conservano la loro lunghezza.
- **Cabinet**, la direzione di proiezione forma un angolo di $\arctan(2) = 63.4^\circ$ e quindi linee perpendicolari al piano hanno lunghezza pari alla metà di quella reale.

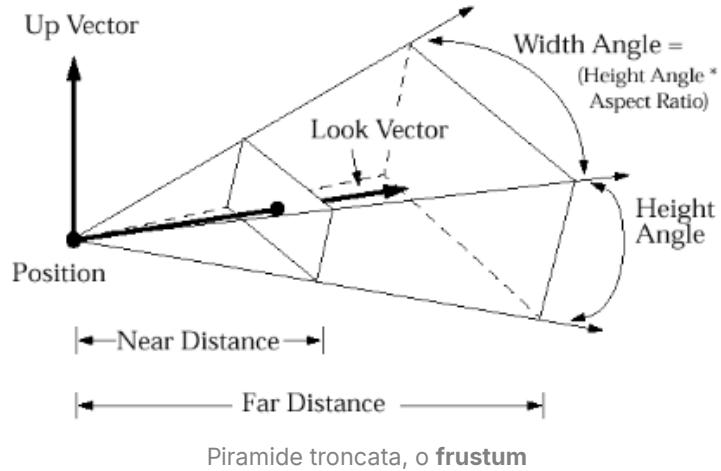
Proiezione geometrica di una figura piana da 3D → 2D

1. Determinare il volume di spazio tra i piano di clipping anteriore e posteriore, che definisce lo spazio limitato che la telecamera può “vedere”, *volume di vista*. Il tipo di proiezione definisce **la forma del volume della vista**.
2. Proiettare il volume di vista nel sistema di coordinate **normalizzate**, cuboide con il centro nell’origine definito in $[-1, 1] \times [-1, 1] \times [-1, 1]$.

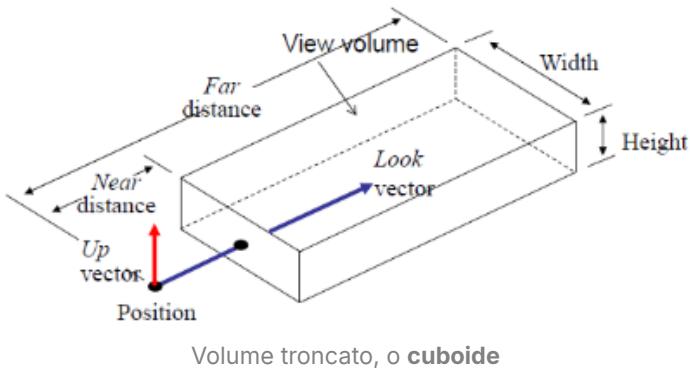


Il clipping rispetto ad un cubo con assi paralleli agli assi coordinate è più semplice da realizzare.

Volume di vista, proiezione prospettica



Volume di vista, proiezione parallela ortografica



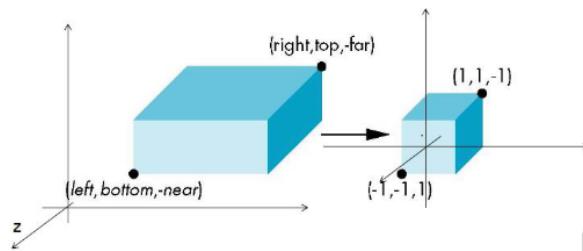
Normalizzazione

Consiste nel proiettare il volume di vista nel sistema di coordinate normalizzato. Trasforma il volume di vista in un volume di vista parallelo.

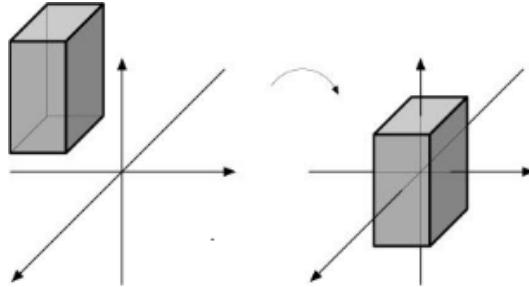
La normalizzazione consente una **singola pipeline sia per la visualizzazione prospettica che per la visualizzazione ortografica**.

Si rimane in quattro coordinate omogenee dimensionali il più a lungo possibile per conservare le informazioni tridimensionali necessarie per la rimozione delle superfici nascoste.

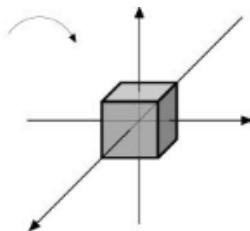
Normalizzazione del volume di vista nelle proiezioni ortogonali



- Spostare il centro nell'origine,
 $T = (-(left + right)/2, -(bottom + top)/2, (near + far)/2)$



- Scalare per avere lati di lunghezza 2,
 $S = (2/(right - left), 2/(top - bottom), 2/(near - far))$



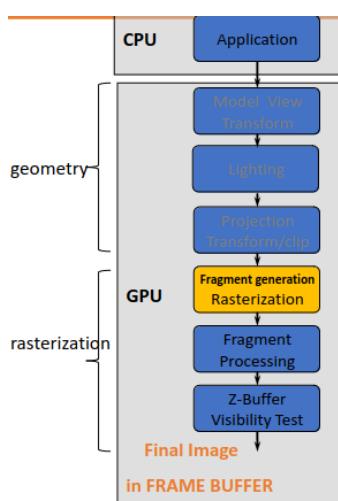
Matematica delle proiezioni

da finire

Pipeline di rendering

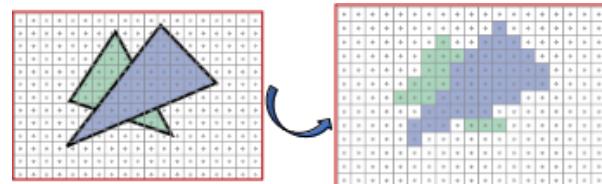
Scan Conversion

Rasterization

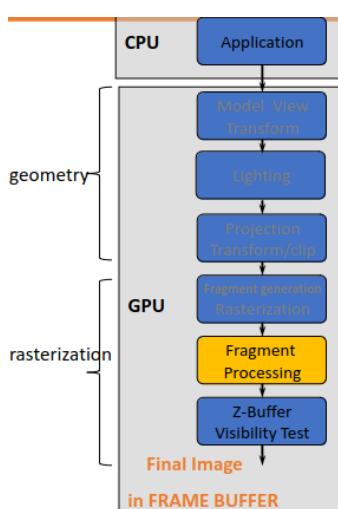


Determina quali pixel sono interni alla primitiva specificata da un insieme di vertici, producendo un insieme di frammenti.

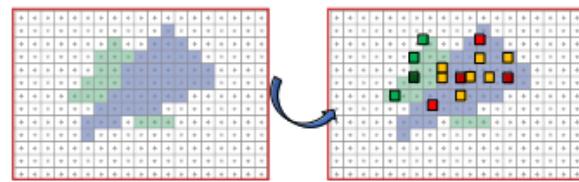
I **Frammenti** hanno una posizione, *pixel location*, e altri attributi, come colore e coordinate di texture che vengono determinati interpolando i valori sui vertici



Fragment processing

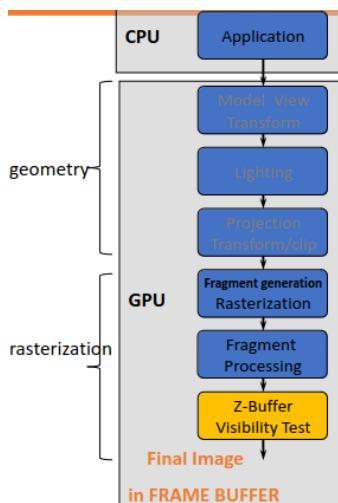


I colori dei pixel vengono determinati successivamente usando colori, texture ed altre proprietà dei vertici, *texture mapping*.



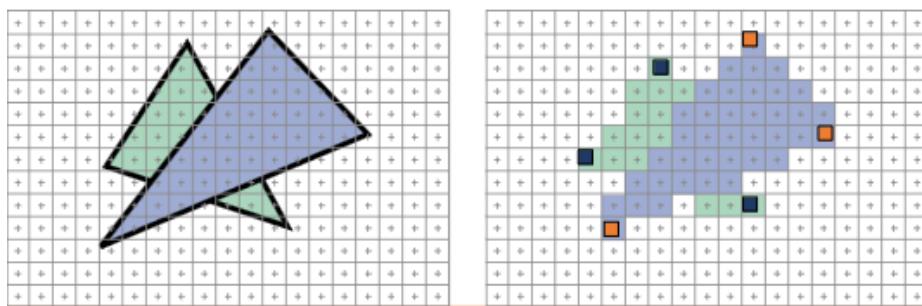
Visibility/Display

Disegna l'oggetto più vicino, *depth buffer*



Rasterization

Sono algoritmi per la generazione efficiente dei frammenti occupati dalle primitive geometriche, enumerando i frammenti occupati dalla primitiva.



Con il termine **rasterizzazione** si intende propriamente il processo di **discretizzazione** che consente di **trasformare una primitiva geometrica definita in uno spazio continuo 2D** nella sua **rappresentazione discreta**, composta da un **insieme di pixel di un dispositivo di output**.



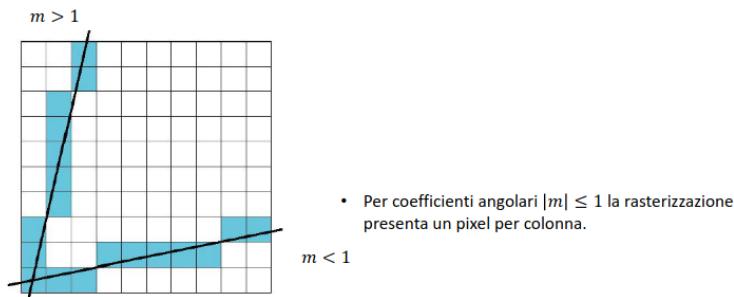
Drawing

Tracciare un segmento di retta all'interno di una griglia regolare di pixel che hanno distanza unitaria in ascissa e in ordinata, *coordinate di schermo*

Casi speciali

- Linea orizzontale, disegna il pixel P e incrementa il valore della coordinata x di 1 per ottenere il pixel successivo
- Linea verticale, disegna il pixel P e incrementa il valore della coordinata y di 1 per ottenere il pixel successivo
- Linea diagonale, disegna il pixel P e incrementa entrambe le coordinate x e y di 1 per ottenere il pixel successivo

Per coefficienti angolari $|m| > 1$
la rasterizzazione presenta un pixel per riga.



Algoritmo DDA, Digital Differential Analyzer

$$y = mx + q, \quad |m| \leq 1 \rightarrow \text{il prossimo pixel si accende per } x+1$$

$$y_{new} = m(x + 1) + q = mx + q + m \equiv y + m$$

$$|m| \geq 1 \rightarrow \text{il prossimo pixel si accende per } y+1$$

$$x_{new} = \frac{1}{m}(y + 1 - q) = \frac{1}{m}(y - q) + \frac{1}{m} \equiv x + \frac{1}{m}$$

!!!!I segni \geq e \leq sono invertiti!!!!

```
Calcola m=(y1-y0)/(x1-x0)
Se |m|<=1 allora
    Poni y=y0
    Ripeti per x=x0; x<=x1; x++
        Accendi il punto (x, round(y))
        y+=m
```

```

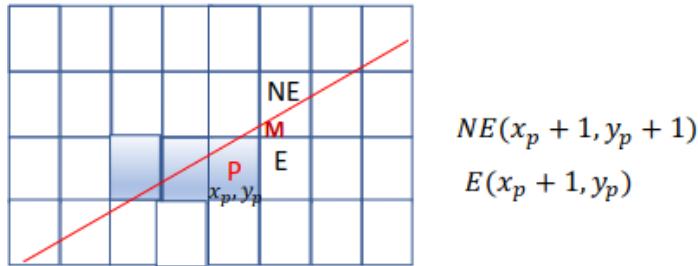
Altrimenti
Poni x=x0
Ripeti per y=y0; y<=y1; y++
    Accendi il punto (round(x), y)
    x+=(1/m)

```

Algoritmo di Bresenham o Mid-Point

L'algoritmo di Bresenham risolve il problema dell'errore introdotto dall'uso di aritmetica floating point nell'algoritmo DDA.

L'algoritmo di Bresenham fa uso solamente di operazioni in aritmetica intera, ed è ancora un algoritmo di tipo differenziale: fa uso delle informazioni calcolate per individuare il pixel al passo i per individuare il pixel al passo $i + 1$.



L'idea di base di questo approccio è quella di individuare, passo dopo passo, il pixel più vicino all'effettivo punto di intersezione Q tra la retta da disegnare e la retta $x = x_p + 1$ corrispondente al successivo punto da accendere.

I candidati, ad ogni passo sono solo i due pixel $E(x_p + 1, y_p)$ ovvero $NE(x_p + 1, y_p + 1)$

Per far questo si valuterà il passaggio della retta, espressa in forma implicita $F(x, y) = ax + by + c = 0$, per il punto medio $M(x_p + 1, y_p + \frac{1}{2})$.

L'implementazione è incrementale.

Si consideri l'equazione della retta in forma esplicita:

$$y = mx + q = (y_1 - y_0)/(x_1 - x_0)x + q = (dy/dx)x + q$$

allora la forma implicita si ottiene come:

$$F(x, y) = dy \cdot x - dx \cdot y + dx \cdot q = ax + by + c = 0$$

Si definisce una **variabile di decisione** d

$$d = F(x_M, y_M) = F\left(x_p + 1, y_p + \frac{1}{2}\right) = a(x_p + 1) + b\left(y_p + \frac{1}{2}\right) + c$$

Per $d > 0$ si accenderà NE , mentre se $d \leq 0$ si accenderà E .

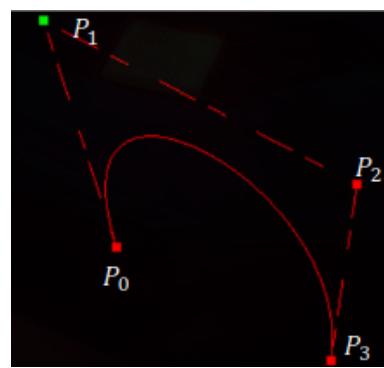
L'algoritmo consiste nel calcolare incrementalmente la sola variabile di decisione d

da finire fino a 28

Curve di Bezier

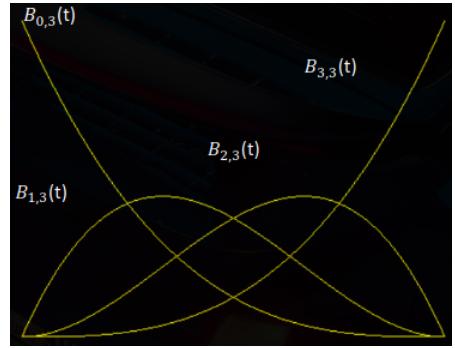
Il concetto base delle curve di Bezier è sostituire i valori D_1 e D_2 , che nelle curve di Hermite vengono interpolati, con altri due valori puntuali, che non vengono interpolati ma solo approssimati e la cui posizione influenza la forma della curva.

I dati di un segmento di curva cubica di Bezier sono quindi quattro valori puntuali che chiameremo P_0, P_1, P_2, P_3 di cui P_0 e P_3 vengono interpolati, mentre gli altri due vengono solo approssimati. La loro posizione **determina il vettore tangente all'inizio ed alla fine del segmento di curva**.



Bezier ha dimostrato che sono i polinomi di Bernstein (di grado 3) cioè:

$$\begin{aligned}
 B_{0,3}(t) &= (1-t)^3 \\
 B_{1,3}(t) &= 3t(1-t)^2 \\
 B_{2,3}(t) &= 3t^2(1-t) \\
 B_{3,3}(t) &= t^3
 \end{aligned}$$



Il segmento di curva cubica di Bezier che intercala P_0 e P_3 e che in P_0 e P_3 è tangente ai segmenti $P_1 - P_0$ e $P_2 - P_3$ rispettivamente è dato da:

$$C(t) = \sum_{i=0}^3 P_i B_{1,3}(t) \quad t \in [0, 1]$$

I punti P_i sono detti **punti di controllo della curva**, ed individuano il poligono di controllo.

Caso generale

Nel caso generale le curve di Bezier di grado n abbiamo a che fare con $n+1$ punti, di cui il primo e l'ultimo sono interpolati, il secondo e il penultimo danno la direzione della derivata negli estremi, mentre gli altri punti vengono "approssimati" e servono a modellare la forma della curva.



Polinomio di base

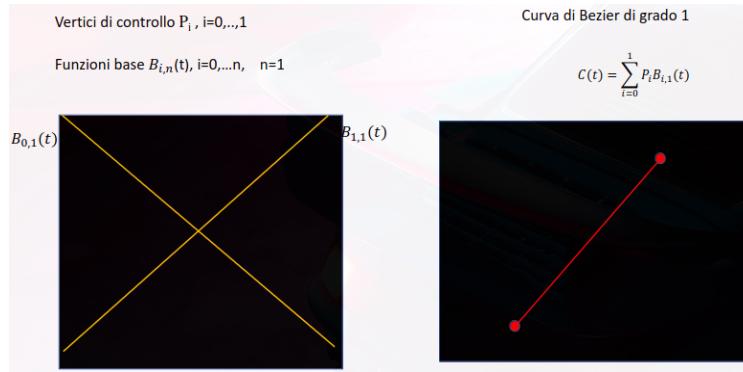
I polinomi di base di Bernstein di grado n definiti nell'intervallo $[0, 1]$ sono dati da:

$$B_{1,n}(t) = \binom{n}{i} (1-t)^{n-i} t^i \quad i = 0, \dots, n$$

dove

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

è chiamato **coefficiente binomiale**



Vantaggi

Uno dei vantaggi principali è il fatto di lavorare interattivamente ottenendo una modellazione della curva in modo molto più intuitivo. In un certo senso si può dire che la forma della curva approssima quella del poligono che si ottiene congiungendo con dei segmenti i punti di controllo. Questo poligono è detto **poligono di controllo**.

I polinomi di Bernstein sono particolarmente utili a rappresentare un polinomio definito in un intervallo $[a, b]$, in quanto sono propriamente definiti relativamente a quell'intervallo.

Caso generale polinomio di grado n definito su un intervallo

In generale un polinomio di grado n sull'intervallo $[a, b]$ può essere rappresentato come:

$$P_n(x) = \sum_{i=0}^n c_i B_{i,n}(x) \quad x \in [a, b]$$

dove i $B_{i,n}(x)$, $i = 0, \dots, n$ sono i polinomi di Bernstein di grado n definiti su $[a, b]$ dalla relazione:

$$B_{i,n}(x) = \binom{n}{i} \frac{(b-x)^{n-i}(x-a)^i}{(b-a)^n} \quad i = 0, \dots, n$$

e c_0, c_1, \dots, c_n sono i coefficienti che identificano il polinomio espresso nella base di Bernstein.

$$t = \frac{x-a}{b-a}$$

In $[0, 1]$ si ha

$$B_{i,n}(t) = \binom{i}{n} (1-t)^{n-i} t^i I \quad i = 0, \dots, n$$

$$P_n(t) = \sum_{i=0}^n c_i B_{i,n}(t) \quad t \in [0, 1]$$

Proprietà dei polinomi di Bernstein di grado n

1. I polinomi di Bernstein di grado n definiti in $[a, b]$ sono legati dai polinomi di Bernstein di grado $n - 1$ definiti su $[a, b]$ dalla seguente formula:

$$B_{i,n}(x) = \left(\frac{x-a}{b-a} B_{i-1,n-1}(x) + \frac{b-x}{b-a} B_{i,n-1}(x) \right)$$

con condizione iniziale che $B_{0,0}(x) = 1$ e $B_{i,n}(x) = 0 \forall i \notin [0, n]$.

Se $[a, b] = [0, 1]$

$$B_{i,n}(t) = t \cdot B_{i-1,n-1}(t) + (1-t) \cdot B_{i,n-1}(t) \quad t \in [0, 1]$$

Entrambe le formule presentano solo somme di quantità positive, quindi sono numericamente stabili. Ciascun polinomio di grado n è la combinazione convessa di polinomi di grado $n - 1$: combinazione di quantità positive mediante coefficienti positivi, la cui somma vale 1.

2. Non negatività

$$B_{i,n}(x) \geq 0 \quad \forall x \in [0, 1]$$

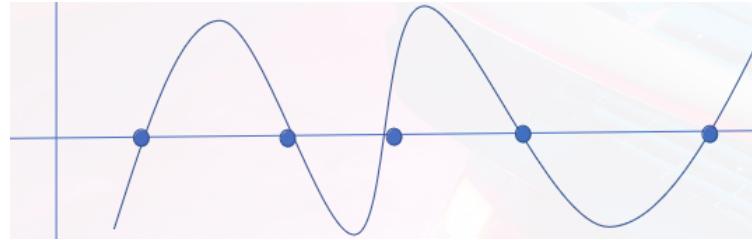
3. I polinomi di Bernstein fomrano una **partizione dell'unità**:

$$\sum_{i=0}^n B_{i,n}(t) = 1$$

4. Il polinomio $p(x) = \sum_{i=0}^n c_i B_{i,n}(x)$ è una **combinazione lineare convessa dei valori dei suoi coefficienti**, e quindi:

$$\min_{i=0, \dots, n} \{c_i\} \leq p(x) \leq \max_{i=0, \dots, n} \{c_i\}$$

5. Il numero di **variazioni di segno di un polinomio** espresso nella base di Bernstein è **minore o uguale al numero di variazioni di segno dei suoi coefficienti**. Questo ci consente di avere per il polinomio un limite superiore per il numero dei suoi zeri semplici in $[a, b]$.

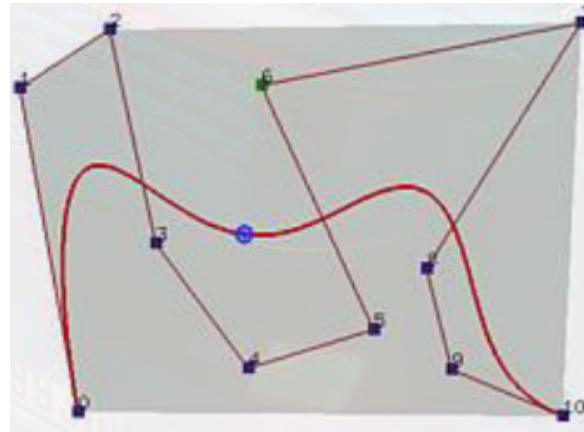


Proprietà dell'inviluppo convesso, convex Hull

Definizione

Assegnati i punti c_0, c_1, \dots, c_n si definisce inviluppo convesso, **convex hull**, dell'insieme dei punti $\{c_i\}$ la più piccola regione convessa che contiene i punti dati.

Il grafico del polinomio espresso nella base di Bernstein è contenuto nell'inviluppo convesso dell'insieme dei vertici del suo poligono di controllo.



Altre proprietà

- **Precisione lineare**, quando tutti i punti di controllo giacciono su una linea, la curva di Bézier è il segmento che interporla i punti.
- **Approssimazione di forma**, il numero di intersezioni di una qualsiasi retta con il poligono di controllo è maggiore o uguale del numero di intersezioni della stessa retta con il polinomio



Dati i vertici di controllo $P_0 \equiv \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, P_1 \equiv \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \dots, P_n \equiv \begin{bmatrix} x_n \\ y_n \end{bmatrix}$, la curva di Bézier di grado n si esprime come:

$$C(t) = \begin{bmatrix} x(t) = \sum_{i=0}^n x_i B_{i,n}(t) \\ y(t) = \sum_{i=0}^n y_i B_{i,n}(t) \end{bmatrix} = \sum_{i=0}^n P_i B_{i,n}(t)$$

Valutazione di un polinomio della base di Bernstein

È ben noto che la valutazione in un punto di un polinomio di grado n espresso nella base monomiale si può effettuare in modo efficiente facendo uso dello schema di **Horner**, che ha una complessità computazionale dell'ordine di $O(n)$ ed è numericamente stabile.

L'algoritmo di **De Casteljau** rappresenta l'analogo per un polinomio espresso nella base di Bernstein. Esso consiste nell'applicare ripetutamente le formule ricorrenti viste per i polinomi della base di Bernstein.

La valutazione in un punto t di un polinomio espresso nella base di Bernstein consiste nel valutare:

$$P_n(t) = \sum_{i=0}^n c_i B_{i,n}(t) \quad \text{in un punto } t \text{ assegnato}$$

Procedimento dimostrazione

Si ottiene:

$$P(t) = \sum_{i=0}^{n-1} c_i^{[j]} B_{i,n-j}(t)$$

con: $c_i^{[j]} = t \cdot c_{i+1}^{[j-1]} + (1-t) \cdot c_i^{[j-1]}$ $i = 0, \dots, n-j$, $j = 1, \dots, n$

Complessità computazionale: $2 \sum_{j=1}^n n - j + 1 = 2 \sum_{j=1}^n j = 2 \cdot O\left(\frac{n^2}{2}\right)$

Valutazione curva di Bezier

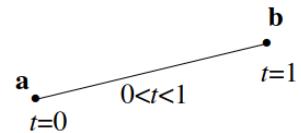
Per ogni valore del parametro t , applicare l'algoritmo di De Casteljau ad ognuna delle sue componenti parametriche

$$C(t) = \sum_{i=0}^n P_i B_{i,n}(t) = \begin{cases} \sum_{i=0}^n x_i B_{i,n}(t) \\ \sum_{i=0}^n y_i B_{i,n}(t) \end{cases}$$

Interpolazione lineare, Lerp

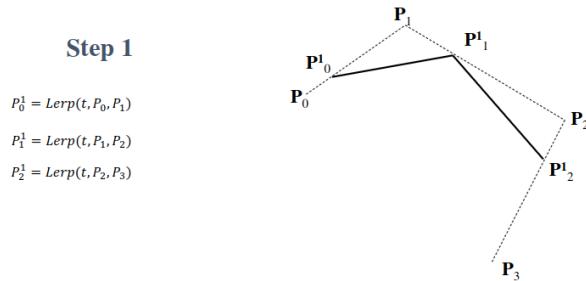
L'interpolazione lineare, **Lerp**, calcola un valore compreso tra due valori, l'interpolante lineare tra due punti a e b con parametro t è data da:

$$Lerp(t, a, b) = (1 - t)a + tb$$



Interpretazione geometrica dell'algoritmo di de Casteljau

Dato un valore di t , si valuta $C(t)$; si considera una curva di Bezier cubica.



Modelli di illuminazione e shading

I **modelli di illuminazione** descrivono i fattori che determinano il colore di una superficie in un determinato punto, tramite le interazioni tra le luci e le superfici, tenendo conto delle proprietà delle superfici e della natura della radiazione luminosa incidente.

L'uso di un modello di illuminazione è necessario per ottenere una rappresentazione realistica delle superfici tridimensionali.

Modelli di shading

I **modelli di shading** determinano come viene applicato il modello di illuminazione e quali argomenti prende in input per determinare il colore di un punto sulla superficie.

Alcuni modelli di shading richiamano il modello di illuminazione per ogni pixel nell'immagine, altri invece richiamano il modello di illuminazione solo per alcuni pixel dell'immagine e colorano i rimanenti pixel per interpolazione.

Lightening vs Shading

Lightening, operazione per *vertex*, è il processo di calcolo dell'intensità luminosa in un particolare punto 3D, solitamente su una superficie, in funzione delle caratteristiche della luce e del materiale

Shading, operazione per *fragment*, assegna il colore al pixel specificando come la luce viene usata per colorare il pixel

Curve Spline

Curve Spline polinomiali di ordine m a nodi multipli

I difetti delle curve di Bezier consistono nella mancanza di controllo locale della curva e nel fatto che il grado dei polinomi di base di Bernstein è leggermente legato al numero di vertici di controllo.

Definizione

Dati i vertici di controllo P_i , $i = 1, \dots, N$, una curva **spline** si definisce come :

$$c(t) = \sum_{i=1}^N P_i N_{i,m}(t) \quad t \in [a, b]$$

dove le funzioni base $N_{i,m}(t)$:

- hanno ordine **m** molto inferiore rispetto al numero di vertici di controllo
- è possibile controllare localmente la forma delle curve, in quanto le funzioni base **B** spline sono a supporto compatto, non sono diverse da zero su tutto l'intervallo $[a, b]$, ma hanno un supporto locale che dipende dal loro ordine

Spline polinomiali a nodi multipli

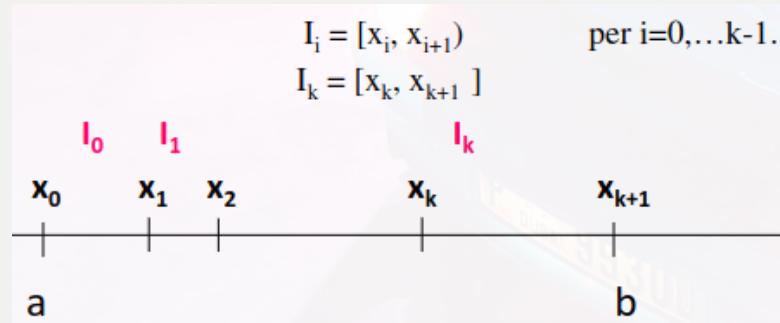


Definizione (Condizioni di raccordo)

Sia $[a, b]$ un intervallo chiuso e limitato, sia Δ una partizione di $[a, b]$ così definita:

$$\Delta = \{a = x_0 < x_1 < \dots < x_k < x_{k+1} = b\}$$

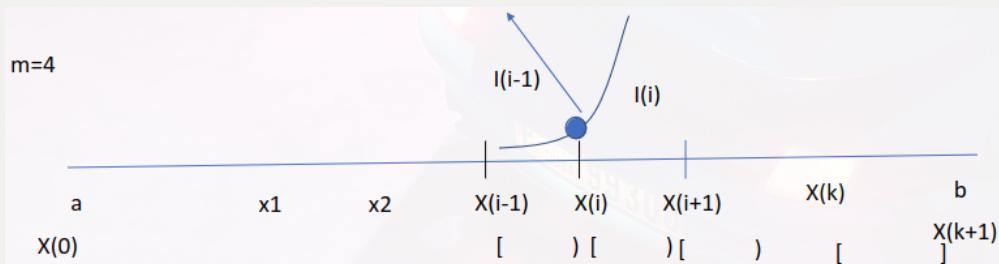
che induce una partizione di $[a, b]$ in $k + 1$ sotto intervalli:



Sia $m < k$ un numero intero positivo e sia $M = (m_1, m_2, \dots, m_k)$ un vettore di interi positivi tali che $1 \leq m_i \leq m, \forall i = 1, \dots, k$.

Si definisce **funzione spline polinomiale** di ordine m con nodi x_1, \dots, x_k di molteplicità m_1, \dots, m_k , una funzione $s(x)$ che in ciascun sottointervallo $I_i, i = 0, \dots, k$ coincide con un polinomio $s_i(x)$ di ordine m e che nei nodi $x_i, i = 0, \dots, k$ soddisfi le condizioni di continuità:

$$\frac{d^j s_{i-1}(x_i)}{dx^j} = \frac{d^j s_i(x_i)}{dx^j} \quad i = 1, \dots, k; j = 0, m - m_i - 1$$



È possibile variare il tipo di raccordo in un nodo x_i a seconda del valore di m_i .

In base a questo valore si può decidere sino a quale derivata effettuare il raccordo: maggiore sarà la molteplicità, minori saranno le condizioni di raccordo sui nodi.



Teorema

Lo spazio delle spline polinomiali di ordine m a nodi multipli con nodi x_1, \dots, x_k di molteplicità $M = (m_1, \dots, m_k)$, che verrà indicato con $S_m(\Delta, M)$, ha dimensioni pari a $m + K$ dove $K = \sum_{i=1}^k m_i$.

DIM

Per ciascuno dei $k + 1$ sotto intervalli il polinomio di ordine m ha m gradi di libertà, quindi si hanno $(k + 1)m$ gradi di libertà; a queste si devono sottrarre le $m - m_1$ condizioni di raccordo imposte su ogni nodo x_i , $i = 1, \dots, k$. Si ha quindi:

$$\begin{aligned} & m(k+1) - \sum_{i=1}^k (m - m_1) \\ &= mk + m - \sum_{i=1}^k m_1 \\ &= mk + m - mk + K \\ &= m + K \end{aligned}$$

Base dello spazio delle Spline B-spline normalizzate

Una buona base per lo spazio delle spline $S_m(\Delta, M)$ è la base delle B-spline normalizzate

Funzioni base B-spline, nodi semplici

Assegnata una successione di nodi $\dots < x_1 < x_2 < \dots$ semplici si definisce **B-spline** normalizzata di ordine m relativa al nodo x_1 , $N_{i,m}(x)$, $x \in [a, b]$, una funzione che gode delle seguenti proprietà:

- $N_{i,m}(x) = 0$ per $x < x_i, x > x_{i+m}$ (**supporto compatto**)
- $\int_{x_i}^{x_{i+m}} N_{i,m}(x) dx = \left(\frac{x_{i+m}-x_i}{m}\right)$ (**condizione di normalizzazione**)

In ciascun intervallo I_j , $j = 1, \dots, i + m - 1$ coincide con un polinomio di ordine m che si raccorda opportunamente con i vicini, in modo tale che $N_{i,m}(x) \in C^{m-2}[a, b]$.

Formule di Cox per la valutazione di un B-spline

Per calcolare le funzioni base, si utilizzano le formule di Cox, che consentono di calcolare $N_{i,m}(x)$ mediante combinazione di due funzioni base di ordine $m - 1$.

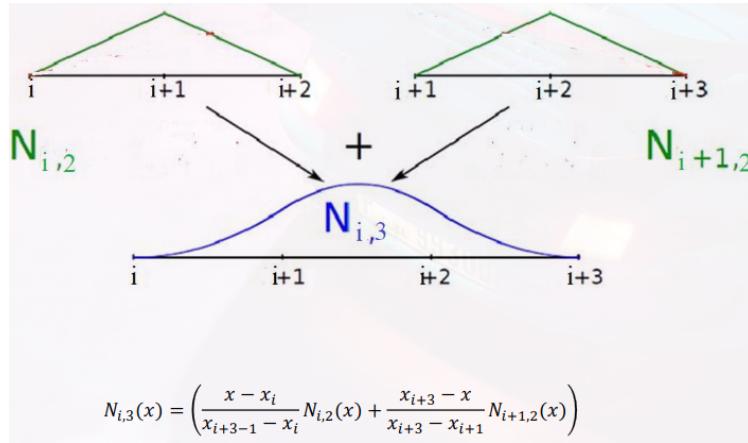
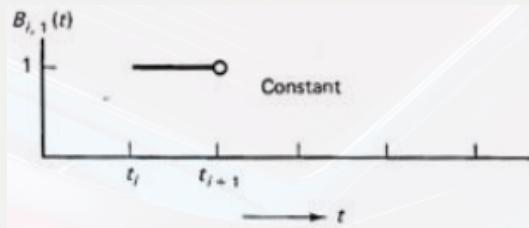


per $h = 1$ si ha:

$$N_{i,1}(x) = \begin{cases} 1 & x_i \leq x \leq x_{i+1} \\ 0 & \text{altrimenti} \end{cases}$$

per $h = 2$ si ha:

$$N_{i,h}(x) = \left(\frac{x - x_i}{x_{i+h-1} - x_i} N_{i,h-1}(x) + \frac{x_{i+h} - x}{x_{i+h} - x_{i+1}} N_{i+1,h-1}(x) \right)$$



Partizione nodale estesa

Data la partizione nodale $\Delta = \{a = x_0 < x_1 < \dots < x_k < x_{k+1} = b\}$ ed il vettore di molteplicità $M = (m_1, \dots, m_k)$, si costruisce la partizione nodale estesa $\Delta^* = \{t_i\}_{i=1}^{2m+k}$ con $K = \sum_{i=1}^k m_i$ nel seguente modo:

- $t_1 \leq t_2 \leq \dots \leq t_{2m+K}$ i nuovi nodi sono semplici ma possono coincidere
- $t_m \equiv a, \quad t_{m+K+1} \equiv b$
- $t_{m+1} \leq t_{m+2} \leq \dots \leq t_{m+K}$ sono scelti in modo tale che siano coincidenti con:

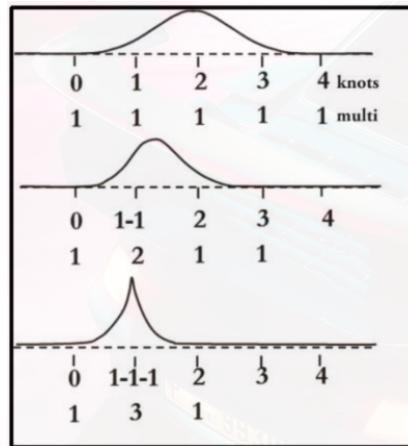
$$\underbrace{(x_1 \equiv x_1 \equiv \cdots \equiv x_1)}_{m_1 \text{ volte}} < \underbrace{(x_2 \equiv x_2 \equiv \cdots \equiv x_2)}_{m_2 \text{ volte}} < \cdots < \underbrace{(x_k \equiv x_k \equiv \cdots \equiv x_k)}_{m_k \text{ volte}}$$

- $t_i \leq a \quad i = 1, \dots, m-1$ (possono coincidere tutti con a)

$t_i \geq b \quad i = m+K+2, \dots, 2, +K$ (possono coincidere tutti con b)

I punti $x_i, i = 1, \dots, k$, vengono chiamati anche **break points**.

Effetto dei nodi multipli sulle funzioni B-spline



Proprietà delle funzioni B-spline con nodi multipli normalizzati

1. Formano una base per lo spazio $S_m(\Delta, M)$

Ciascuna $s(x) \in S_m(\Delta, M)$ può essere espressa come:

$$s(x) = \sum_{i=1}^{m+k} c_i N_{i,m}(x) \quad x \in [a, b]$$

2. Hanno supporto compatto:

$$N_{i,m}(x) \text{ per } x < x_i, x > x_{i+m}$$

3. Sono non negative sul loro supporto. $N_{i,m}(x) \geq 0, x \in [t_i, t_{i+m}]$

4. Costituisco una partizione dell'unità, cioè: $\sum_{i=1}^{m+K} N_{i,m}(x) = 1$

È presente il controllo locale.