

# **Sistemi di realtà virtuale**

# Indice

---

<b>1. INTRODUZIONE.....</b>	<b>7</b>
PANORAMICA .....	7
CENNI STORICI .....	9
TIPI DI SISTEMI RV .....	10
CARATTERISTICHE TECNICHE DEI SISTEMI DI REALTÀ VIRTUALE .....	12
GESTIONE DEGLI INPUT E DEL TRACCIAMENTO DELL'UTENTE .....	13
VR SOFTWARE.....	14
RAPPRESENTAZIONE E INTERAZIONE DELL'UTENTE .....	15
<i>Rappresentazione</i> .....	15
<i>Interazione dell'utente</i> .....	15
CAMPPI DI APPLICAZIONE.....	16
TREND ATTUALI .....	17
<b>2. VOLUMI DI CONTENIMENTO (BOUNDING BOX, BV).....</b>	<b>19</b>
VOLUMI DI CONTENIMENTO.....	19
<i>AABB (Axis Aligned Bounding Box, parallelepipedo con i lati paralleli agli assi)</i> .....	20
Creazione di un AABB .....	20
<i>OBB (Oriented Bounding Box, parallelepipedo generico)</i> .....	20
Creazione di un OBB .....	20
<i>k-DOP (Discrete Oriented Polytope)</i> .....	21
Creazione di un <i>k</i> -DOP .....	21
<i>Sfera</i> .....	21
Creazione di una sfera .....	21
GERARCHIE DI BOUNDING VOLUMES (BHV) .....	22
<b>3. BASI GEOMETRICHE .....</b>	<b>23</b>
RETTA .....	23
DEFINIZIONE DI UNA RETTA MEDIANTE VETTORI .....	24
CIRCONFERENZA.....	24
SUPERFICIE SFERICA.....	24
PIANO .....	24
LE TRASFORMAZIONI BIDIMENSIONALI.....	25
<i>Traslazione</i> .....	25
<i>Rotazione</i> .....	25
<i>Cambiamento di scala</i> .....	26
COORDINATE OMogenee .....	27
<i>Traslazione in coordinate omogenee</i> .....	27
<i>Rotazione in coordinate omogenee</i> .....	27
<i>Cambiamento di scala in coordinate omogenee</i> .....	27
VETTORI .....	28
<i>Prodotto scalare</i> .....	28
<i>Prodotto vettoriale</i> .....	28
ANGOLI DI EULERO .....	29
<i>Definizione</i> .....	29
<i>Significato algebrico</i> .....	29
<b>4. INTERSEZIONI .....</b>	<b>31</b>
RAGGIO .....	31
INTERSEZIONE RAGGIO - PIANO.....	31

INTERSEZIONE RAGGIO - SFERA .....	32
<i>Metodo analitico/algebrico</i> .....	32
Normale al punto di intersezione .....	33
<i>Metodo geometrico</i> .....	33
Passo 1: capire se l'origine del raggio è interna o esterna alla sfera .....	33
Passo 2: esclusione dell'intersezione in caso di raggio in direzione opposta .....	34
Passo 3: punto del raggio più vicino alla sfera e sua distanza dal centro della sfera.....	34
Passo 4: calcolo delle intersezioni e delle normali .....	34
INTERSEZIONE RAGGIO - TRIANGOLO.....	34
INTERSEZIONE RAGGIO - VOLUME DI CONTENIMENTO (METODO SLABS) .....	35
INTERSEZIONE RAGGIO - POLIGONO .....	36
INTERSEZIONE PIANO - BOX .....	37
<i>Metodo algebrico</i> .....	38
<i>Metodo geometrico</i> .....	38
AABB.....	38
OBB.....	38
INTERSEZIONI FRA VOLUMI DI CONTENIMENTO.....	39
<i>Intersezione sfera - sfera</i> .....	39
<i>Intersezione AABB - AABB</i> .....	39
<i>Intersezione k-DOP – k-DOP</i> .....	39
<i>Separating axis theorem in 2D</i> .....	39
<i>Intersezione OBB - OBB</i> .....	40
INTERSEZIONE FRA LINEE .....	42
2D .....	42
3D .....	43
DYNAMIC INTERSECTION TESTING.....	43
<i>Sfera - piano</i> .....	44
<i>Sfera - sfera</i> .....	45
<b>5. GESTIONE DELLA FISICA .....</b>	<b>47</b>
INTRODUZIONE.....	47
LEGGI DI NEWTON (~ 1700) .....	48
<i>Sistemi particellari</i> .....	48
SIMULAZIONE FISICA.....	49
<i>Esempio 3D di un movimento di un proiettile</i> .....	49
<i>Equazioni in forma chiusa</i> .....	50
<i>Approssimazione numerica</i> .....	50
(1) Serie di Taylor.....	50
(2) Integrazione di Eulero .....	50
Esempio di simulazione numerica .....	51
(3) Integrazione di Runge-Kutta (RK2, RK4).....	51
RISPOSTA ALLA COLLISIONE SENZA ATTRITO .....	52
<i>Impulso</i> .....	52
<i>Coefficiente di restituzione</i> .....	54
<i>Calcolare la velocità dopo la collisione</i> .....	54
<b>6. PIPELINE DI RENDERING.....</b>	<b>57</b>
PIPELINE .....	57
CICLI DI RENDERING E FRAME RATE .....	58
<i>Capacità della pipeline</i> .....	59
MODULI DELLA PIPELINE DI RENDERING .....	60
<i>Stadio applicativo</i> .....	60
<i>Stadio geometrico</i> .....	61

(1) Trasformazione dei modelli (model & view transform) .....	61
(2) Computazione della luce (vertex shading) .....	62
(3) Proiezione della scena .....	63
(4) Clipping .....	64
(5) Screen mapping .....	64
<i>Stadio di rasterizzazione</i> .....	65
(1) Triangle setup .....	66
(2) Triangle traversal (scan conversion) .....	66
(3) Pixel shading .....	66
(4) Merging .....	66
PROGRAMMABILITÀ DELLA SCHEDA GRAFICA .....	68
<i>A programmable GPU pipeline</i> .....	68
Vertex shader .....	68
Geometry shader .....	69
OpenGL .....	69
<i>Shader programming model</i> .....	71
COME DEFINIRE UNA MESH USANDO TRIANGOLI .....	72
<b>7. MODELLI DI ILLUMINAZIONE .....</b>	<b>75</b>
ILLUMINAZIONE AMBIENTALE .....	76
RIFLESSIONE DIFFUSIVA (MODELLO LAMBERTIANO) .....	77
RIFLESSIONE SPECULARE .....	79
<i>Modello di Blinn Phong</i> .....	80
MODELLI DI OMBREGGIATURA .....	81
<i>Flat shading</i> .....	81
<i>Gouraud shading</i> .....	81
<i>Phong shading</i> .....	82
MODELLI DI ILLUMINAZIONE GLOBALE .....	83
(1) <i>Radiosity</i> .....	83
<i>Ray casting</i> .....	89
(2) <i>Ray tracing</i> .....	90
<i>Shadow rendering</i> .....	92
Volumi d'ombra .....	93
Mappe d'ombra .....	93
<i>Ambient occlusion</i> .....	94
<b>8. STEREOSCOPIA .....</b>	<b>95</b>
ELEMENTI DELLA STEREOSCOPIA .....	95
<i>Binocular disparity</i> .....	95
<i>Convergency e accomodation</i> .....	96
<i>Psicologia</i> .....	96
ELEMENTI DI FISIOLOGIA DELLA VISIONE .....	97
<i>L'occhio: composizione e principio di funzionamento</i> .....	97
<i>La retina</i> .....	99
<i>Fusione sensoriale</i> .....	99
<i>La percezione della profondità, la stereopsi</i> .....	102
TECNICHE STEREOSCOPICHE .....	103
<i>I principi di funzionamento</i> .....	103
<i>Modello della visione</i> .....	105
<i>Criticità legate alla tecnica stereoscopica</i> .....	109
DISPLAY .....	111
<i>Autostereoscopici</i> .....	111
Barriera parallasse .....	111

Lenti cilindriche .....	111
<i>Non-autostereoscopici</i> .....	112
Passivi .....	112
Attivi .....	113
<b>9. LA RIPRESA STEREOSCOPICA.....</b>	<b>115</b>
INTRODUZIONE.....	115
<i>I sistemi di riferimento</i> .....	116
La camera ottica .....	116
Il sistema di riferimento per la scena ripresa – coordinate scena .....	117
Il sistema di riferimento per i sensori – coordinate immagine .....	118
Il sistema di riferimento per la visione – coordinate osservatore .....	118
<i>La camera ottica come modello geometrico di telecamera generica</i> .....	120
Geometria della camera ottica .....	120
<i>Relazioni tra sensore e schermo del display</i> .....	122
<i>La geometria di visualizzazione</i> .....	122
LA GEOMETRIA DI RIPRESA.....	125
<i>Assi ottici paralleli</i> .....	125
<i>Assi ottici convergenti</i> .....	128
CONCLUSIONI .....	131
ELEMENTI DI GEOMETRIA EPIPOLARE.....	131
PROIEZIONE DA 3D A 2D .....	133
<i>Proiezione ortografica</i> .....	133
<i>Proiezione prospettica</i> .....	133
<i>Proiezione prospettica e calibrazione</i> .....	134
APPLICAZIONI DELLA STEREOSCOPIA.....	135
<i>Image-based rendering</i> .....	135
<i>Free viewpoint video</i> .....	136
Generazione tramite mappa di profondità ottenuta da sensore .....	136
Generazione tramite mappa di profondità ottenuta mediante depth estimation .....	137
<b>10. AUDIO 3D .....</b>	<b>138</b>
INTRODUZIONE.....	138
<i>Sistema uditivo umano</i> .....	138
<i>Percezione del suono e della sua direzione</i> .....	138
<i>Metodi per la creazione dell'audio immersivo</i> .....	139
TRASFORMATA DI FOURIER.....	141
<i>Numeri complessi</i> .....	141
<i>Serie di Fourier</i> .....	141
<i>Impulsi e proprietà di sifting</i> .....	142
<i>Trasformata di Fourier di una variabile continua</i> .....	143
<i>Trasformate di Fourier utili</i> .....	143
Esempio 1: ottenere la trasformata di Fourier di una semplice funzione .....	143
Esempio 2: trasformata di Fourier di un impulso .....	144
Esempio 3: trasformata di Fourier di un treno di impulsi .....	144
<i>Convoluzione</i> .....	146
<i>Campionamento</i> .....	147
<i>Discrete Fourier transform</i> .....	149
SISTEMA AUDIO.....	149
<i>Risposta all'impulso</i> .....	150
<i>Funzione di trasferimento</i> .....	150
3D AUDIO SIGNAL PROCESSING.....	151
<i>HRTF</i> .....	151

<i>Misurazioni sperimentali</i> .....	152
<i>Gestione del sistema con casse</i> .....	152
<b>11. SPEECH RECOGNITION .....</b>	<b>155</b>

# 1. Introduzione

## Panoramica

Quando si parla di **realità virtuale** ci si riferisce a una simulazione computerizzata che mira a costruire un'immagine del mondo (un mondo virtuale) che appaia ai nostri sensi il più vicino possibile al mondo reale (seppur molto diverso dal mondo reale che circonda l'utente).

Per far sì che il partecipante percepisca le immagini sintetizzate come reali, il sistema si occupa di modificare la scena virtuale seguendo il movimento e le azioni compiute dall'utente, in modo da dare all'utente la sensazione di essere immerso o di essere presente all'interno della simulazione.

In modo più formale un sistema di realtà virtuale può essere definito come un mezzo composto di simulazioni interattive computerizzate, che percepisce la posizione e le azioni del partecipante, fornendo dei feedback sintetici a uno o più sensi, dando la sensazione di essere immersi nella simulazione.

Notiamo che la definizione formale dice "fornendo dei feedback sintetici a uno o più sensi", intendendo che il sistema può comunicare con l'utente non solo tramite la vista (il senso classico), ma anche tramite suoni, vibrazioni o altri mezzi.

Le vibrazioni, o tutte le sensazioni fisiche (percepibili tramite la pelle) che possono essere sfruttate per la comunicazione lavorano sui **sensi aptici** (il tatto).

Meno di frequente si stimolano sensi come l'olfatto e il gusto, di difficile riproduzione allo stato dell'arte.

Quando il termine realtà virtuale fu coniato, produsse la grande aspettativa di poter sintetizzare un mondo virtuale indistinguibile da quello reale.

Ad oggi, tale aspettativa non è ancora soddisfatta, in quanto la tecnologia non è ancora abbastanza sviluppata da generare tali mondi virtuali.

Pertanto, l'obiettivo è diventato costruire delle rappresentazioni accettabili di parti del mondo reale.

Tornando alle caratteristiche di un sistema di realtà virtuale possiamo dire che un buon SRV deve costruire un ambiente che sia percepibile e realistico, con sufficiente interattività per eseguire dei compiti specifici in un modo efficiente e comodo.

I fattori principali sono:

- 1) **Immersione**: fortemente legata alla configurazione fisica dell'interfaccia utente, origina la classificazione in:
  - **fully immersive**: con sistemi fisici quali gli **HMD (Head-Mounted display)** come il casco. Questi sistemi mirano a isolare completamente l'utente dal mondo reale, con la speranza che questo contribuisca alla credibilità della simulazione. L'uso di HMD porta però (con lo stato dell'arte attuale) a problemi fisici negli utenti che li usano, quali nausea e capogiro.
  - **semi-immersive**: con sistemi fisici quali la proiezione su grandi schermi; esempio ne è il **CAVE**, un ambiente di lavoro multiutente circondato da schermi dove viene proiettato il mondo virtuale; meno realistico del precedente permette simulazione in gruppo, per valutare il lavoro di squadra.
  - **non-immersive**: con sistemi fisici basati su desktop; sicuramente i meno realistici, hanno grande popolarità per il loro costo basso, la facilità di uso e di installazione; esempi ne sono i videogames.

Tale classificazione dipende da quanto l'utente percepisce (vede, sente, tocca) del mondo reale durante la simulazione.

- 2) **Presenza:** la presenza (o il senso della presenza) è un concetto soggettivo che indica la sensazione provata dall'utente di trovarsi all'interno del mondo virtuale. Tale sensazione può essere provocata in diversi modi (immagini, suoni, feedback aptici...) attraverso i differenti stimoli inviati al cervello che li capisce e li considera come coerenti fra loro. La sensazione di presenza è riconoscibile quando l'utente crede e agisce nel mondo virtuale, nel modo in cui crede e agisce nel mondo reale; ciò provoca effetti di coinvolgimento e reazioni emotive nell'utente, fatti che hanno portato a pensare di usare i SRV a scopi terapeutici, per la gestione delle paure. La sensazione di presenza può essere influenzata da diversi fattori, in relazione al mondo che si sta costruendo: in un ambiente medico/chirurgico è importante il tatto, in un'orchestra l'audio. La presenza deve essere distinta dal **coinvolgimento**, il quale dipende principalmente dal tipo di relazione che si instaura fra l'utente e il contenuto della simulazione (se è interessato o meno), mentre la presenza viene a crearsi quando il livello di realismo della scena è sufficientemente alto.
- 3) **Mondo virtuale**
- 4) **Feedback sensoriale**
- 5) **Interattività:** possibilità di muoversi nella scena 3D e manipolare gli oggetti
- 6) **Real time:** le azioni modificano immediatamente (o quasi immediatamente) lo stato della scena.

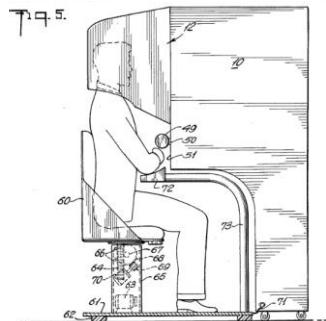
In particolare l'immersione, l'interazione e il real-time definiscono quello che è chiamato il **triangolo della realtà virtuale**:



## Cenni storici

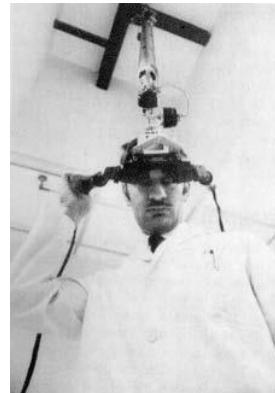
La storia della realtà virtuale inizia negli anni '60: il sistema (**sensorama**) creato dal cinematografo Morton Heilig permetteva all'utente di sedersi davanti ad uno schermo e scegliere quale veicolo guidare (una filmato di guida preregistrato); l'immersione è data da un sistema audio 3D e generatori di odori e di vento.

L'utente era un osservatore passivo.



Successivamente nel 1968 nasce il primo HMD (creato da Sutherland) che presentava all'utente scene 3D: il sistema traccia il movimento della testa dell'utente e aggiorna le immagini di conseguenza.

Le immagini erano molto lontane dalla realtà ma meglio che niente.



Successivamente Sutherland sviluppa anche lo **Sketchpad** considerato l'antenato dei programmi moderni di progettazione computerizzata (CAD). Sutherland è comunemente riconosciuto come il padre dei sistemi di realtà virtuale.



Il ricercatore Myron Krueger dell'Università del Wisconsin stava sperimentando una prospettiva diversa sui sistemi di realtà virtuali, che egli definiva "Realtà artificiale".

Mentre il display head-mounted di Sutherland era particolarmente adatto per un punto di vista in prima persona nel mondo virtuale, la realtà artificiale di Krueger forniva una visione in seconda persona di un mondo virtuale in cui i partecipanti potevano guardarsi nel mondo.

Il passo seguente nel 1978 fu di creare una mappa interattiva, simile alle attuali street view di Google maps, in cui poter navigare su percorsi determinati precedentemente, simulando la guida nella città di Aspen (Colorado).

Il sistema utilizzato fotografa tutte le strade della città, usando quattro telecamere che puntano in direzioni diverse montate su un camion. L'utente può muoversi in quattro direzioni all'interno della simulazione. Questi erano gli inizi degli ambienti virtuali interattivi.



I sistemi di realtà virtuale hanno subito una grande spinta anche dalla NASA, nel cui centro di ricerca sono stati creati sistemi per l'apprendimento dei piloti per le missioni spaziali.

Dagli anni 80 la strada è diventata tutta in discesa con le nuove innovazioni tecnologiche, fino alla commercializzazione di sistemi quali gli **oculus rift**.

## Tipi di sistemi RV

1. Windows on World: sono detti anche Desktop VR.

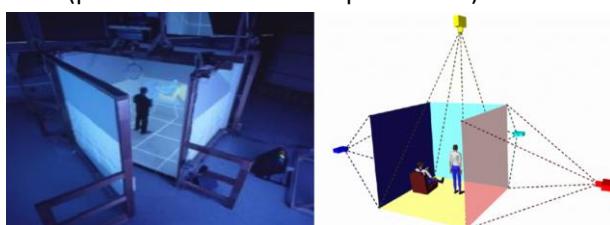
2. VR immersivo: l'immersione è totale, il rapporto con l'ambiente fisico reale è annullato.

Gli strumenti fisici che possono essere usati sono:

- **HMD**: L'*Head-Mounted Display* è un elmetto, o una maschera che mostra immagini stereo (2 canali diversi per occhio destro e sinistro) sugli LCD e può includere un sistema audio basato su cuffie e un head-tracker per tracciare il movimento della testa.  
Può essere usato sia per i sistemi immersivi che per quelli di realtà aumentata.  
Ha il vantaggio di essere molto immersivo e permettere buone libertà all'utente, con gli svantaggi che risulta invasivo e pesante da portare, è difficile da condividere con altri utenti e crea fenomeni di distorsione sui bordi.  
Lo svantaggio più significativo però è che l'utente che lo porta percepisce delle sensazioni visive diverse da ciò che gli comunica il corpo con gli altri sensi (principalmente l'equilibrio) provocandogli dei disorientamenti con conseguente nausea e capogiro.



- **CAVE**: altro metodo è usare un CAVE (*Cave Automatic Virtual Environment*) cioè un ambiente (solitamente una stanza) tappezzata di schermi (solitamente si coprono le pareti frontali e laterali e più di rado quelle sopra e sotto).  
Con questo metodo l'utente è immerso nella scena e l'ambiente virtuale (sia visivo che audio) mutano in base ai suoi spostamenti nella stanza e alle sue azioni (il suo movimento è tracciato).  
I vantaggi sono che l'utente ha la massima libertà di movimento e il costo del sistema è limitato rispetto ad altri sistemi immersivi, anche se data la grandezza degli schermi e la quantità di dettagli che devono essere presenti per rendere la scena realistica il costo del sistema può aumentare (per farne aumentare le prestazioni).



- **BOOM**: alternativamente ai precedenti si può usare un *Binocular Omni-Orientation Monitor*, che si presenta come un binocolo montato su un braccio articolato in modo da poter muovere il binocolo stesso in ogni direzione.

Con questo metodo i movimenti sono più lenti rispetto all'HMD perché è necessario muovere il braccio meccanico e non solo scuotere la testa; questo diminuisce il rischio di lag nel tracciamento della posizione e semplifica il sistema, richiedendo anche meno velocità e immediatezza nella risposta.

Naturalmente l'immersione è limitata in quanto si muove di pari passo con la libertà data all'utente.

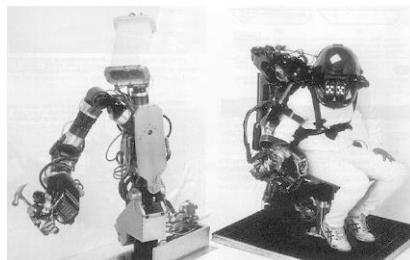


- **Dispositivi aptici:** i dispositivi aptici si occupano della gestione del senso del tatto (per gli esempi successivi consideriamo come dispositivo aptico un guanto indossabile i cui spostamenti possono essere limitati da un braccio meccanico collegato). Lavorando a diretto contatto con la pelle umana si pongono l'obiettivo di simulare ciò che accade quando un utente si muove fisicamente nella scena toccando un oggetto presente. Non si limita a ciò ma viene anche usato per la gestione delle sensazioni di calore o di freddo date dalla scena e dai suoi oggetti. Ma non solo, infatti simulano anche le forze attive o resistenti degli oggetti nella scena (ad esempio non si può trapassare un oggetto con la propria mano, ma il guanto viene bloccato quando si tocca l'oggetto, dando la sensazione di aver impattato contro qualcosa). Può anche lavorare sul senso di pesantezza (spingendo il guanto verso il basso con una certa forza) di un oggetto o sulla viscosità o fluidità del mezzo in cui si sta mettendo la mano.
- Il guanto di cui abbiamo parlato è un dispositivo **selfgrounded**, cioè indossabile dall'utente. Naturalmente non viene usato solo per inviare una risposta del sistema all'utente ma anche come input verso il sistema tramite i sensori di cui è provvisto.

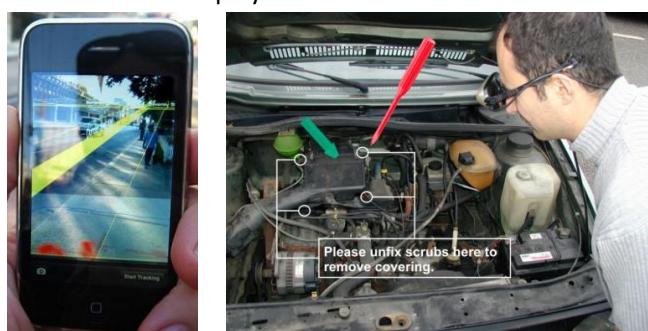


Questi dispositivi permettono una gestione naturale del movimento da parte dell'utente, fornendo un alto livello di libertà.

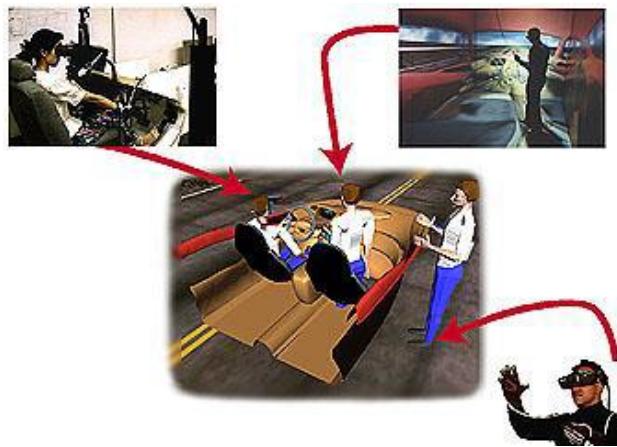
- **Telepresenza:** l'utente si connette con un sistema meccanico, per il controllo di tale sistema da remoto in situazioni e ambienti pericolosi. In questo caso l'ambiente in cui si lavora è reale ma lontano da quello dell'utente.



- **Realtà aumentata:** una combinazione di realtà e mondo virtuale sovrapposti, ad esempio nell'applicazione di Pokemon go (o negli oculus rift). Come nella telepresenza questo sistema permette di avere un nuovo punto di vista sul mondo reale, o di aggiungere informazioni su/intorno ciò che si vede. Uno strumento fisico classico è l'**HUD (Head-Up Display)** in cui un display trasparente viene posto davanti all'utente in modo da mostrare la realtà retrostante e aggiungere eventuali informazioni mostrate sul display.

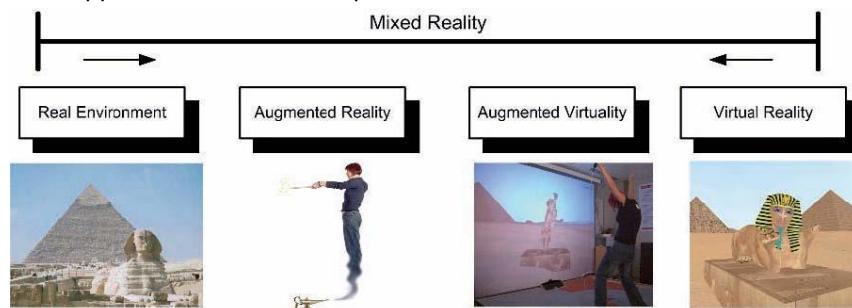


- **VR distribuito:** il mondo simulato è distribuito su diversi sistemi interconnessi, in modo che più utenti possano collaborare nello stesso mondo virtuale pur trovandosi fisicamente lontani.



I sistemi di realtà virtuale possono essere anche classificati in base alla quantità di realtà e di virtualità presente nella scena: a partire dall'ambiente reale ci si sposta verso l'ambiente completamente virtuale passando per la realtà aumentata e per la virtualità ambientata.

Classificazione delle applicazioni in base alla quantità di realtà e di virtualità contenuta:



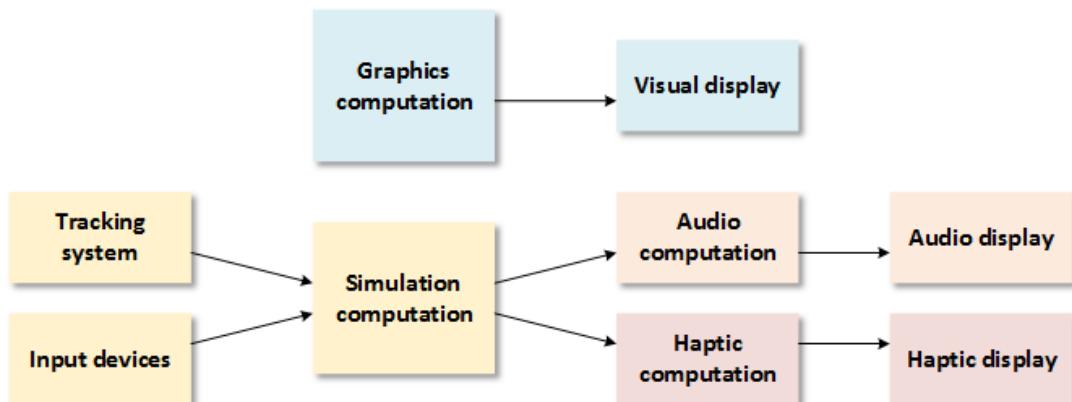

---

### Caratteristiche tecniche dei sistemi di realtà virtuale

---

I sistemi di realtà virtuale richiedono componenti sia hardware che software: il sistema si occupa di ricevere gli eventuali input, di effettuare i calcoli riguardo la fisica del mondo virtuale e in generale la sua simulazione, e di renderizzare lo stato del mondo in modo visivo, acustico, aptico...

Un tipico sistema VR:



Per far sì che il sistema permetta un'interazione in real-time, è necessario che abbia alcuni prerequisiti di potenza specifici, sia nel caso in cui sia implementato su un singolo computer, sia se è distribuito su più computer.

Nel caso in cui i prerequisiti non siano soddisfatti si possono verificare dei ritardi fastidiosi: il tempo che intercorre fra l'occorrere di un evento e il momento in cui la simulazione mostra il suo effetto risulta troppo lungo rispetto ad una situazione reale, peggiorando la simulazione.

Una soluzione può essere l'uso di più CPU contemporaneamente.

Anche il tempo che intercorre fra la costruzione di immagini correlate della stessa scena presentate su schermi diversi (ad esempio quelli per l'occhio destro e sinistro dell'oculus rift) deve essere minimo per evitare effetti innaturali di incongruenza.

---

## Gestione degli input e del tracciamento dell'utente

---

Perché un sistema simuli la realtà l'utente deve poterci interagire, inviandogli degli input.

A differenza degli input inviati solitamente ad un programma per pc, in un sistema di realtà virtuale non è solo necessario che l'utente comunichi le proprie decisioni (cosa fare, **cognitive input**) ma anche che alcune parti del suo corpo siano tracciate (**user monitoring**).

Uno dei dispositivi più usati è il **sensore di posizione**, che può essere usato per tracciare una posizione approssimativa dell'utente, ma anche un posizione dettagliata di una sua sezione (un dito o una mano). I sensori possono essere di vari tipi, elettromagnetici, ottivi, ultrasonici, neurali/muscolari, meccanici, etc... ciascuno con i propri vantaggi e le proprie limitazioni.

Qualsiasi sia il tipo di sensore usato è importante tener conto di: accuratezza e precisione della posizione riportata dal sensore, materiali che creano interferenza, dimensione e ingombro del sensore.

Nessun tipo di sensore conosciuto finora fornisce ottimi risultati in tutte e tre le caratteristiche elencate, ma alcuni offrono compromessi migliori di altri, in particolare conoscendo l'ambiente in cui dovrà lavorare.

È suggeribile cercare con attenzione il miglior sensore di posizione, poiché errori in questo ambito sono fastidiosi tanto quanto i lag del sistema.

Analizziamone ora nel dettaglio alcuni tipi:

- 1) **Elettromagnetici**: sono sensori molto popolari; il sensore indossato dall'utente è collegato con un filo al sistema, oppure (in quelli più recenti e costosi) con connessione radio, per dare maggiore libertà di movimento all'utente.
- 2) **Meccanici**: sensori più invadenti dei precedenti usano una connessione fisica fra l'utente e il sistema (ne sono un esempio i BOOM visti in precedenza); questi sensori sono molto accurati, incrementando la sensazione di immersione, con lo svantaggio che deve esserci un contatto fisico fra l'utente e il dispositivo, cosa che spesso ne impedisce i movimenti naturali.
- 3) **Guanti**: poco ingombranti e molto precisi vengono usati prevalentemente per cogliere la configurazione della mano, piuttosto che la sua posizione; i sensori disposti sulle diverse articolazioni delle mani permettono di capire la posizione delle dita della mano e la posizione/rotazione della mano stessa; i materiali usati sono solitamente delle strisce di metallo (che aumentano la loro tensione quando piegati) o le fibre ottiche (che fanno passare meno luce quando piegate).
- 4) **Ultrasonico**: usando un sistema di *trasmittenti* (speakers) e *riceventi* (microfoni) si invia il segnale dai primi ai secondi, valutando il tempo di trasmissione; tempi diversi equivalgono a distanze diverse nella coppia; per fornire un'informazione completa è necessario inserire diverse coppie sugli assi X, Y, Z per disegnare l'utente nella sua interezza.

- 5) **Ottico**: usa la *computer vision* per riconoscere l'utente nella figura che gli viene presentata e cogliere di conseguenza la sua posizione; non è un sistema adattabile ad ogni ambiente perché l'utente deve essere riconoscibile, contrastato con lo sfondo (il contrasto può essere creato ad arte tramite dei led indossabili).
- 6) **Inerziale o giroscopico**: vengono usati per riportare movimenti relativi e non posizioni assolute; un esempio frequente è quello negli HMD, in cui è importante conoscere le angolazioni rispetto all'asse principale, invece di sapere la posizione della testa nel mondo. Solitamente hanno il vantaggio di laggare poco.
- 7) **Neurali e muscolari**: posizionati sulla pelle, percepiscono i cambiamenti neurali o muscolari che avvengono sottopelle; un esempio è il sensore posizionato sul braccio che rileva quando la mano viene stretta a pugno.
- 8) **Altri**: altri input possono essere costruiti per permettere all'utente una maggiore azione, ad esempio una pulsantiera premibile per fargli effettuare salti o per sparare nel mondo virtuale.

## VR Software

Tuffiamoci ora nel mondo dei software di realtà virtuale analizzandone le componenti principali:

- **Motore fisico**: usato per la simulazione delle leggi della natura nel mondo virtuale, non segue necessariamente le leggi del mondo reale (ad esempio si può impedire all'utente di camminare sulle pareti, ma se il personaggio è Spiderman deve poterlo fare).
- **Librerie di rendering**: usate per convertire la scena generata nel sistema virtuale in immagini bidimensionali visibili dall'utente; comprende gli algoritmi di rendering che vedremo nel seguito, che non si riferiscono solo al rendering visivo, ma a quello genericamente multisensoriale (includendo tatto e audio); per quanto riguarda la parte visiva si occupano anche di generare la corretta (e naturale) luminosità e ombreggiatura della scena, per generare un realistico effetto tridimensionale; ultima loro funzionalità è la **collision detection** (che vedremo in seguito).
- **Librerie VR**: usate per tracciare la posizione e gli input dell'utente e aggiornare il sistema in modo adeguato.
- **Free VR**: una libreria che funge da interfaccia/integrazione per la realtà virtuale completamente open-source, disegnata per lavorare con input e output diversi, adattabile ai diversi sistemi e dispositivi usati.  
Il suo obiettivo (genericamente è l'obiettivo dell'open-source) è quello di far condividere a più utenti possibili la ricerca VR, usando hardware e software diversi.
- **VR Juggler**: un'altra libreria open-source; framework di sviluppo di sistemi di realtà virtuale altamente scalabile, che può essere usata su PC come su sistemi molto complessi; la sua flessibilità la rende atta ad essere usata in molte configurazioni di sistemi VR come i desktop, gli HMD e i CAVE; inoltre è portatile su quasi qualsiasi piattaforma (Linux, Windows, Solaris, Mac OS X e altre).

Scendiamo ancora più nel tecnico per parlare del linguaggio usato: il **VRML** (*Virtual Reality Modeling Language*), un linguaggio standard per le simulazioni interattive con il web; permette di creare mondi virtuali via Internet, o di gestire parti di applicazioni tramite il collegamento via Internet (ad esempio il reperimento delle informazioni dell'oculus rift).

I file usati per questo linguaggio sono di diversi tipi: xml-based come X3D, 3DMLW e COLLADA o javascript-based come O3D (creato da Google) oppure ancora pdf-based come U3D.

---

## Rappresentazione e interazione dell'utente

---

### Rappresentazione

Come detto all'inizio di questo capitolo la realtà virtuale è un mezzo per la comunicazione, ma può variare in base alla comunicazione che si vuole ottenere: si può pensare di simulare un mondo reale, il più verosimile possibile, oppure un mondo fantastico, con oggetti mai visti prima, oppure ancora cercare di mostrare il mondo reale in un modo in cui non siamo abituati a vederlo, ad esempio mostrando ciò che può vedere qualcuno che ha subito un danno cerebrale.

È necessario creare una **mappa** fra i concetti che si vogliono descrivere nel mondo virtuale e il modo in cui vengono presentati all'utente tramite i suoi molteplici sensi; la complessità di questa mappa è limitata dalle capacità del sistema usato, ma anche dagli strumenti di output a disposizione (solo video, anche audio o addirittura tatto).

Solitamente una rappresentazione molto realistica (e quindi molto dettagliata) è complessa da computare, provocando una diminuzione del real-time, e viceversa.

Cercare un *trade-off* tra la bontà della rappresentazione e l'efficienza temporale è una delle priorità di chi costruisce il sistema.

Alcuni trucchi per arrivare ad un buon compromesso sono:

- l'uso di **texture mapping** (posiziona delle immagini sugli oggetti, con l'obiettivo di dargli un aspetto più realistico, senza un costo eccessivo)
- la gestione del livello di dettaglio (diminuire o aumentare il dettaglio degli oggetti per velocizzare o rallentare il sistema)
- la decimazione dei poligoni (meno poligoni ci sono meno lavoro deve fare il rendering).

### Interazione dell'utente

---

L'introduzione della realtà virtuale ha portato alla nascita di molti nuovi metodi di interazione non utilizzabili precedentemente.

Allo stesso tempo molte delle interazioni dei sistemi computerizzati tradizionali possono essere utilizzati nei sistemi di realtà virtuale.

Facciamo ora una carrellata delle principali categorie di interazione:

- 1) **Interazione diretta:** dove l'utente mimando le azioni le fa nel sistema reale; ad esempio per spostare un oggetto allunga la mano, stringe le dita, alza la mano, la sposta nella posizione desiderata, l'abbassa e apre la mano.
- 2) **Interazione fisica:** in cui si usa un sistema fisico di input, tipo il mouse nei sistemi standard o un guanto nei sistemi di RV.
- 3) **Interazione virtuale:** in cui il sistema usato per l'input non è reale ma virtuale.  
Ad esempio per sparare un razzo è necessario premere un bottone che fa parte della scena e il nostro dito deve "premere" tale bottone (un esempio standard è l'equalizzatore delle casse del PC).
- 4) **Interazione via agente:** comunicare al sistema tramite un agente, solitamente un altro sistema complesso computerizzato (tipo Siri).

In base all'effetto dell'interazione nel sistema possiamo suddividere l'interazione in:

- **Effettuare una selezione** usando la posizione di una parte del corpo, ad esempio indicando un punto, oppure usando la voce.
- **Manipolazione del mondo virtuale** dopo aver selezionato un oggetto lo si può manipolare, spostandolo, schiacciandolo, cambiando colore, distruggendolo, dividendolo per crearne di nuovi...  
Le azioni fattibili dipendono dal sistema e dall'oggetto ma sono classificabili in quelle fatte direttamente sull'oggetto (schiacciandolo con la mano) e quelle esercitate in modo soprannaturale (scegliendo il nuovo colore da un menù a tendina).
- **Navigazione** per muoversi da un posto all'altro, attraverso le componenti di viaggio (con cui ci si sposta) e **wayfinding** (con cui si informa il mondo su dove si vuole andare).  
Lo spostamento può essere fatto in diversi modi, quali:
  - lo **spostamento fisico** (si cammina)
  - il **ride-along** (spostandosi solo su percorsi segnalati)
  - il **towrope** (come il ride-along con ancora meno libertà di allontanarsi dal centro della via tracciata)
  - il **fly-through** (massima libertà in ogni direzione)
  - il **pilot-through** (l'utente da gli input con dei pulsanti)
  - **move-the-world** (sposto il mondo e non l'utente)
  - **scale-the-world** (si riduce il mondo, ci si sposta di poco e si ingrandisce il mondo)
  - **put-me-here** (si comunica la posizione e il sistema fa il resto).

---

## Campi di applicazione

---

Come visto nelle sezioni precedenti i sistemi di realtà virtuale possono essere adattati a differenti situazioni con diverse configurazioni, dalle più semplici alle più complesse.

I loro campi di applicazione possono essere suddivisi in macrocategorie:

- 1) **Virtual prototyping** con l'obiettivo di generare applicazioni che limitino i costi di produzione degli oggetti da lanciare sul mercato; permettono la valutazione ergonomica (l'usabilità), costruibilità (la facilità di produzione dell'oggetto) e estetica del nuovo oggetto da lanciare.  
In questi casi il realismo è molto più importante dell'interattività.
- 2) **Visualization** di un grande ammontare di dati (pensiamo alle ricerche scientifiche) difficilmente visualizzabili in un modo che permetta all'utente di comprenderne il significato.  
Può inoltre essere usato in campo medico per visualizzare il mondo attraverso un'ampia varietà di "lenti".
- 3) **Training** per attività che sono rare, costose e pure pericolose (quali il volo).
- 4) **Entertainment** principalmente nei videogames, con risultati incredibili ad oggi nel confronto prezzo/performance.
- 5) **Medico** apprendimento di attività mediche poco invasive quali la sutura, ma anche visualizzazione di analisi mediche, ad esempio per la simulazione dell'analisi ai raggi X e la visualizzazione dei risultati.  
Queste applicazioni sono utili per effettuare apprendimento con infermieri o medici, annullando i rischi causati dalle fasi di apprendimento su pazienti veri; altro uso è quello del controllo remoto, attraverso il quale un chirurgo può effettuare un'operazione controllando il robot che esegue fisicamente l'operazione anche se è dall'altra parte del mondo.
- 6) **Altri** quali i campi medici, artistici e dell'educazione.

---

## Trend attuali

---

Al termine di questo capitolo cerchiamo di capire quali sono i trend attuali e futuri che segue questa branca.

allo stato dell'arte, i trend nei sistemi di realtà virtuale mirano a convertire le applicazioni esistenti nella realtà virtuale, con l'obiettivo di allontanarsi dal codice scritto per rendere accessibili certe tecnologie a persone non esperte del settore.

Altro obiettivo è quello di sviluppare applicazioni di realtà virtuale con un motore di gioco, che permette di creare facilmente mondi virtuali e di svilupparne la navigazione.

Inoltre ci si è concentrati anche sulla riduzione dei costi dei device di input, creandone ad arte per gli specifici usi, dispositivi con capacità limitate ma utili all'occorrenza.

Esempi ne sono il **Nintendo Wiimote** (o Wii Remote) e il **Microsoft Kinect**.

In particolare il secondo è formato da due camere di input e un microfono: le due camere si occupano del colore (valori RGB) e della posizione tramite una camera ad infrarossi (IR); il microfono viene usato per la speech recognition.

La camera per il riconoscimento della posizione riconosce l'utente in base agli sbalzi di profondità (dall'utente al background), inoltre riconosce le parti del corpo facendo uso di conoscenze a priori sulla struttura umana.

Così come per i device di input, anche le schede grafiche stanno calando di prezzo, rendendo accessibile ad un pubblico più vasto i frutti della realtà virtuale.

Per quanto riguarda l'output ci si indirizza sui display in modalità stereo passiva: al posto di produrre due immagini diverse per l'occhio destro e quello sinistro (ad esempio simulando l'uso di due camere nella scena), si costruisce una sola immagine che viene poi filtrata (il termine tecnico è *polarizzata*) in modo diverso prima che raggiunga gli occhi (esempio ne sono gli occhialini di 3D, che tramite una sola immagine proiettata, filtrata con occhialini rossi e blu, danno la sensazione di tridimensionalità).

Ancora, più che la realtà virtuale si sta spostando il focus sulla realtà aumentata, fornita su supporti portabili a mano, date le informazioni che questa può fornire e l'ingombro minimo che si ha.

Altre ancora sono le funzionalità da citare che si inizia a sfruttare, quale il tracciamento dell'occhio, l'interazione utente/ambiente tramite wireless, l'uso di display distribuiti (per mostrare grandi quantità di informazioni contemporaneamente), il multiutente (sia in ambito di realtà virtuale che di realtà aumentata) e altri ancora su cui non ci soffermiamo.



## 2. Volumi di contenimento (*bounding box*, BV)

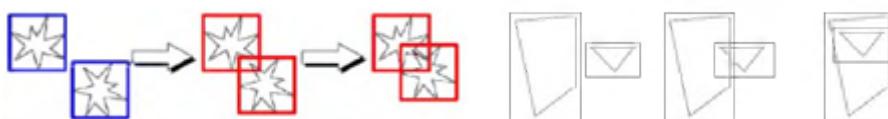
Vedremo nel capitolo successivo come determinare le intersezioni fra raggi di varia natura e oggetti della scena, passando attraverso le primitive che le costituiscono.

### Volumi di contenimento

Gli oggetti però possono essere molti e molto complessi, perciò data una collezione di oggetti è utile trovare un **volume di contenimento** che la racchiuda per minimizzare il costo del calcolo delle intersezioni o degli algoritmi di **collision detection**.

Un buon volume di contenimento deve avere alcune caratteristiche fondamentali:

- essere sufficientemente aderente all'oggetto contenuto per approssimare bene l'oggetto
- essere geometricamente semplice per evitare di ricadere in costi eccessivi di calcolo
- le sue trasformazioni devono essere facili da calcolare (es. rotazione)
- occupare poco spazio di memorizzazione.



Si racchiudono gli oggetti in solidi (volumi) che li contengono con i quali sia facile testare l'intersezione: se non c'è intersezione con il bounding volume non c'è intersezione con l'oggetto racchiuso.

Questo non rende sub-lineare la complessità della ricerca delle intersezioni, ma semplifica le operazioni, e dunque sortisce nella pratica un miglioramento dei tempi.



C'è un ovvio *trade-off* tra precisione del bounding box all'oggetto e costo del test.

Una misura quantitativa usata per misurare la bontà di un certo bounding box è il **void volume** (*volume vuoto*) ovvero la differenza tra il volume del BV e il volume dell'oggetto.

Il rilevamento dell'intersezione (**intersection detection**) consiste nel rilevare se due oggetti si intersecano/si sovrappongono nello spazio ed avviene eseguendo dei test di intersezione.

È un problema che si verifica nella grafica computerizzata in molte forme, tra cui:

- clipping
- eliminazione del volume di visualizzazione (**view-volume culling**): consiste nello scartare in modo efficiente e conservativo i poligoni (o gli oggetti) che non intersecano il volume di vista
- ray-tracing
- rilevamento delle collisioni (**collision detection**) ovvero il rilevamento dell'intersezione tra un insieme di oggetti in movimento.

Possono esserci 3 relazioni spaziali tra due oggetti:

- non c'è intersezione (*disjoint, exclusion*)
- intersezione parziale (*overlap*)
- contenimento (*inclusione*).



Dato un insieme di oggetti, trovare un volume di contenimento appropriato è importante per ridurre al minimo i costi di intersezione.

La probabilità che un raggio arbitrario possa colpire un qualsiasi oggetto convesso è proporzionale all'area di superficie dell'oggetto.

Riducendo al minimo questa zona aumenta l'efficienza di qualsiasi algoritmo di intersezione, in quanto un esito negativo (il raggio non interseca il BV) è più veloce che calcolare l'intersezione.

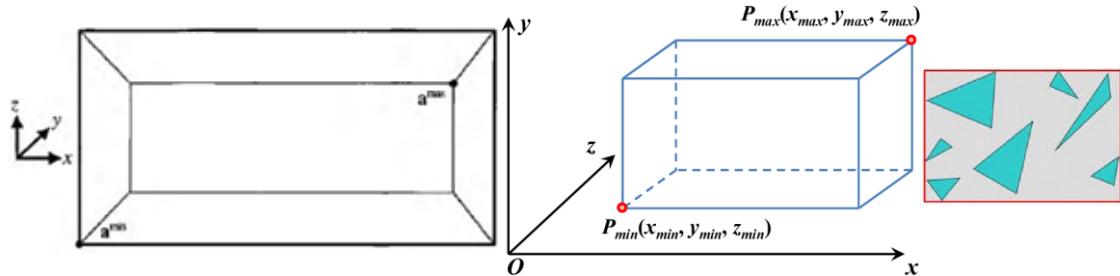
Spesso quindi è meglio ridurre al minimo il volume di ogni BV per gli algoritmi di rilevazione delle collisioni.

Esistono diversi volumi di contenimento: l'AABB, l'OBB, il  $k$ -DOP e la sfera.

### **AABB (Axis Aligned Bounding Box, parallelepipedo con i lati paralleli agli assi)**

Un AABB è un parallelepipedo orientato in maniera tale da avere le facce orientate normali agli assi del sistema di riferimento.

Un AABB  $A$  è definito da due punti estremi  $a^{min}$  e  $a^{max}$  con  $a_i^{min} \leq a_i^{max}$ ,  $\forall i \in x, y, z$ .



#### **CREAZIONE DI UN AABB**

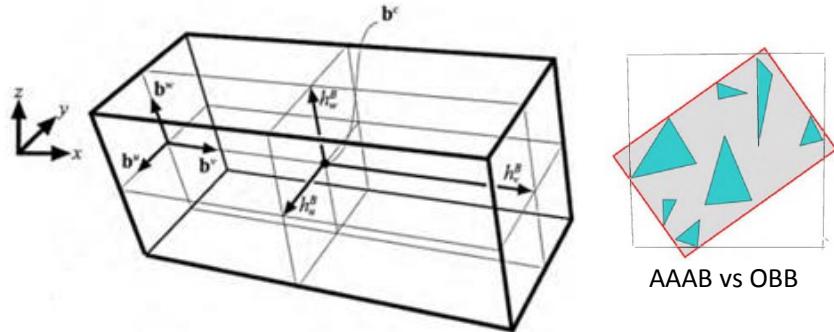
L'AABB facilmente creabile attorno ad un oggetto considerando l'estensione massima e minima dell'oggetto lungo ciascun asse  $x, y, z$ .

### **OBB (Oriented Bounding Box, parallelepipedo generico)**

In alternativa abbiamo l'**OBB** cioè un parallelepipedo come l'AABB allineato però nella direzione dell'oggetto.

Un OBB  $B$  può essere descritto da un punto centrale  $b^c$  e da tre vettori normalizzati e orientati positivamente che descrivono le direzioni del box.

I tre vettori sono  $b^u, b^v, b^w$  e hanno lunghezza  $h_u^B, h_v^B, h_w^B$ .



#### **CREAZIONE DI UN OBB**

Il meccanismo principale per determinarlo è considerare delle tecniche statistiche come la **principal component analysis**, una tecnica di regressione statistica che trova le 3 principali direzioni di propagazione dei dati (o dei componenti dell'oggetto sulla scena) che possono essere usate come assi a cui allineare l'OBB.

## ***k*-DOP (Discrete Oriented Polytope)**

Più complesso è il ***k*-DO**, che crea un poligono a *k* vertici attorno all'oggetto.

È l'intersezione di un insieme di coppie di piani paralleli dove ogni coppia di piani è chiamata **slab**.

È definito da  $k/2$  (con  $k$  pari) normali normalizzare  $\mathbf{n}_i$  con  $1 \leq i \leq k/2$ .

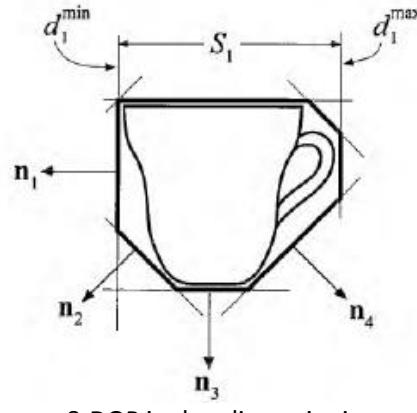
Ogni normale ha associati due valori  $d_i^{\min}$  e  $d_i^{\max}$  con  $d_i^{\min} < d_i^{\max}$ .

Ogni tripletta  $S_i = \{\mathbf{n}_i, d_i^{\min}, d_i^{\max}\}$  definisce uno slab  $S_i$  che è il volume tra due piani  $\pi_i^{\min}$  e  $\pi_i^{\max}$ :

$$\begin{aligned}\pi_i^{\min}: \mathbf{n}_i \cdot \mathbf{x} + d_i^{\min} &= 0 \\ \pi_i^{\max}: \mathbf{n}_i \cdot \mathbf{x} + d_i^{\max} &= 0\end{aligned}$$

Il volume *k*-DOP è poi l'intersezione tra tutti gli slab:

$$\bigcup_{1 \leq i \leq \frac{k}{2}} S_i$$



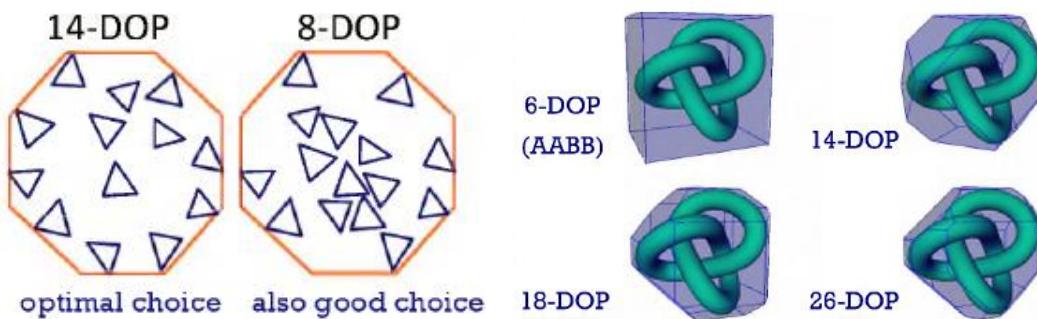
8-DOP in due dimensioni.

### **CREAZIONE DI UN *k*-DOP**

Per creare un *k*-DOP proietti i vertici su ogni normale  $\mathbf{n}_i$  del *k*-DOP e i valori estremi (*min*, *max*) di queste proiezioni sono memorizzati in  $d_{min}^i$  e  $d_{max}^i$ .

Questi due valori definiscono lo slab più piccolo per quella direzione.

Effettuiamo lo stesso calcolo per ogni direzione determinando tutti gli slab del *k*-DOP ed insieme, tutti questi valori definiscono un *k*-DOP minimo.



## **Sfera**

Uno degli oggetti più usati è la **sfera**.

È invariante alle rotazioni e dal punto di vista della memoria è il bounding volume meno costoso.

### **CREAZIONE DI UNA SFERA**

La creazione può avvenire in diversi modi:

- **determinare un AABB e poi usare il centro dell'AABB come centro della sfera e la diagonale dell'AABB come raggio.**

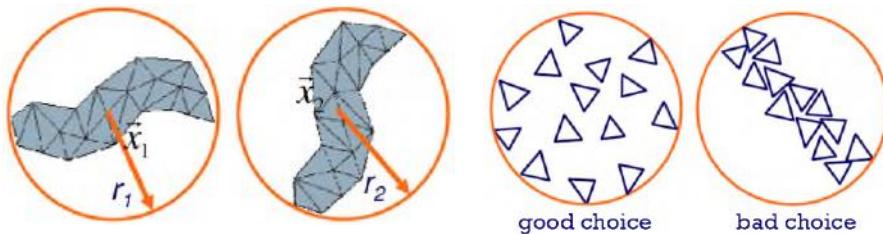
È un metodo veloce ma di solito inefficiente, meglio centrare la sfera nell'AABB e determinarne il raggio considerando la massima estensione dell'oggetto contenuto.

- **algoritmo di Ritter:** l'idea è trovare il vertice che sia minimo e un vertice che sia massimo lungo ciascuno degli assi  $x$ ,  $y$  e  $z$ .

Tra queste 3 coppie prendi la coppia che ha distanza massima tra minimo e massimo e crea la sfera che viene centrata nel punto di mezzo e ha raggio  $r$  pari alla metà della distanza.

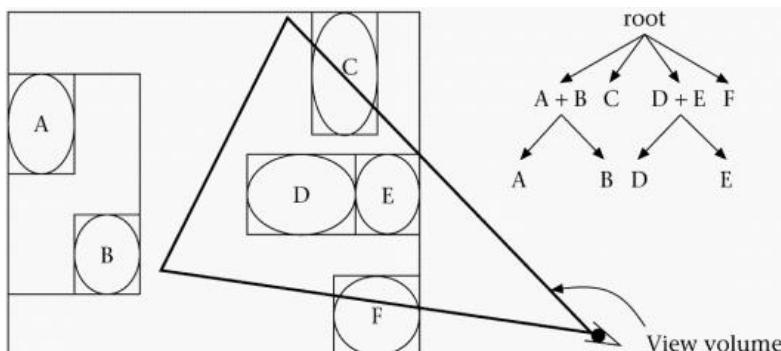
Determiniamo poi la distanza  $d$  di ogni altro vertice dal centro della sfera: ogni volta che ne troviamo uno fuori dalla sfera muoviamo il centro della sfera verso quel vertice di  $(d - r)/2$ , settando il raggio a  $(d + r)/2$ .

Dopo aver effettuato questi accorgimenti per ogni vertice dell'oggetto c'è garanzia che la sfera contenga tutto l'oggetto.



### Gerarchie di bounding volumes (BHV)

Si costruiscono **gerarchie di bounding volumes** dove al livello più alto si ha un volume che racchiude tutta la scena, ed al livello più basso si hanno bounding volumes per i singoli oggetti.



Le gerarchie sono un approccio per migliorare le performance del collision detection quando sono in presenza di molti oggetti od oggetti complessi.

Sono in grado di ridurre la complessità dell'algoritmo da  $O(n^2)$  a  $O(n \log n)$  o anche  $O(n)$ .

### 3. Basi geometriche

#### Retta

La più semplice linea è una **retta**  $r$  i cui punti  $P(x, y)$  soddisfano l'equazione:

$$ax + by + c = 0$$

dove  $a, b, c$  sono coefficienti opportuni che caratterizzano una specifica retta.

- Se  $c = 0$  la retta passa per l'origine
- se  $a = 0$  la retta è parallela all'asse  $x$  e sarà espressa nella forma  $y = k$  dove  $k$  è una costante
- se  $b = 0$  la retta è parallela all'asse  $y$ , cioè  $x = k$ .

L'equazione di una retta può essere rappresentata sotto varie forme a seconda degli elementi che si vogliono mettere in evidenza o di cui si disponga per la costruzione.

Fissati per esempio due punti  $P(x_p, y_p)$  e  $Q(x_q, y_q)$ , la retta che passa per essi è data da:

$$\begin{aligned} \frac{x - x_p}{x_q - x_p} &= \frac{y - y_p}{y_q - y_p} \\ (x - x_p)(y_q - y_p) &= (y - y_p)(x_q - x_p) \\ x(y_q - y_p) - y(x_q - x_p) - x_p y_q + x_q y_p &= 0 \end{aligned}$$

Nel caso in cui si richieda che la retta stacchi sugli assi due segmenti dati, cioè passi per i punti  $P(p, 0)$  e  $Q(0, q)$ , l'equazione della retta è data da:

$$\frac{x}{p} + \frac{y}{q} = 1$$

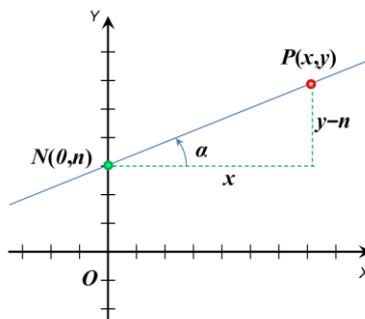
Quando invece la retta stacchi sull'asse  $y$  un segmento  $n$  e formi con l'asse  $x$  un angolo  $\alpha$ , posto  $m = \tan \alpha$ , detto **coefficiente angolare** o **pendenza**, possiamo scrivere:

$$y = mx + n$$

È facile vedere che da  $ax + by + c = 0$  si ottiene:

$$m = -\frac{a}{b}$$

$$n = -\frac{c}{b}$$



Un modo molto utilizzato nella computer grafica per rappresentare una retta è quello parametrico, cioè:

$$x = x_0 + at \quad y = y_0 + bt$$

dove  $x_0, y_0$  sono le coordinate del punto iniziale,  $t$  rappresenta il parametro con valore  $\in \mathbb{R}$ , mentre  $a$  e  $b$  sono opportuni coefficienti (vedremo in seguito interpretabili come componenti del vettore che specifica la direzione della retta).

Nel caso di un segmento che collega due punti  $P(x_p, y_p)$  e  $Q(x_q, y_q)$  l'equazione della retta a cui appartiene il segmento è:

$$x = x_p + \underbrace{(x_q - x_p)}_a t \quad y = y_p + \underbrace{(y_q - y_p)}_b t$$

e per percorrere il segmento  $\overline{PQ}$  il valore di  $t$  è compreso fra 0 e 1.

Quando  $t = 0$  il punto è quello iniziale, mentre per  $t = 1$  ci si trova nel punto finale del segmento.

Gli altri valori di  $t$  che soddisfano la condizione  $0 < t < 1$  corrispondono ai punti compresi fra  $P$  e  $Q$ .

Per  $t < 0$  o  $t > 1$  si percorre la retta passante per  $P$  e  $Q$  esternamente al segmento.

---

## Definizione di una retta mediante vettori

---

Dato un punto  $P_0(x_0, y_0, z_0)$  ed un vettore  $\nu = (v_x, v_y, v_z)$  è possibile scrivere l'equazione parametrica della retta  $r$  passante per il punto e avente direzione specificata dal vettore:

$$\mathbf{r}(t) = P_0 + \nu t$$

dove  $t \in \mathbb{R}$ .

Se  $t$  assume valori non negativi si percorre la retta da  $P_0$  nel verso del vettore  $\nu$ , mentre valori non positivi fanno spostare, a partire da  $P_0$ , nel verso opposto a  $\nu$ .

Se il vettore  $\nu$  è definito mediante due punti  $P_1(x_1, y_1, z_1)$  e  $P_2(x_2, y_2, z_2)$  (uno dei due punti può coincidere con  $P_0$ ) allora  $\nu = P_2 - P_1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$  e si ha:

$$\mathbf{r}(t) = P_0 + (P_2 - P_1)t$$

ovvero in modo esplicito:

$$x = x_0 + (x_2 - x_1)t$$

$$y = y_0 + (y_2 - y_1)t$$

$$z = z_0 + (z_2 - z_1)t$$

---

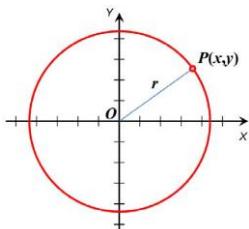
## Circonferenza

---

Oltre alla retta, una linea di notevole interesse sul piano è la **circonferenza** di centro in  $O$  e raggio  $r$ :

$$x^2 + y^2 = r^2$$

cioè i punti  $P(x, y)$  della circonferenza hanno distanza costante  $r$  dall'origine.

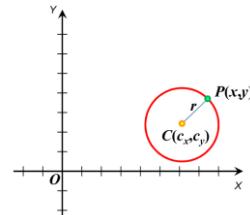


Se invece la circonferenza ha centro in  $C(c_x, c_y)$  e raggio  $r$  l'equazione è data da:

$$(x - c_x)^2 + (y - c_y)^2 = r^2$$

cioè:

$$x^2 + y^2 - 2xc_x - 2yc_y + c_x^2 + c_y^2 - r^2 = 0$$



---

## Superficie sferica

---

L'equazione della **superficie sferica** di centro nell'origine è:

$$x^2 + y^2 + z^2 = r^2$$

Nel caso in cui il centro sia in  $C(c_x, c_y, c_z)$  e il raggio sia  $r$  l'equazione diventa:

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = r^2$$

All'interno di una sfera possono trovare allocazione oggetti ed essere facilmente individuati nel volume di contenimento.

Gli oggetti si trovano all'interno della sfera se le loro coordinate, sostituite nell'equazione producono un valore minore di  $r^2$ .

---

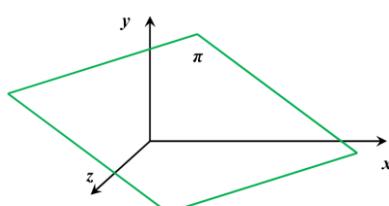
## Piano

---

L'equazione del piano è:

$$\pi: ax + by + cz + d = 0$$

con  $a, b, c, d$  costanti che caratterizzano un determinato piano  $\pi$ .



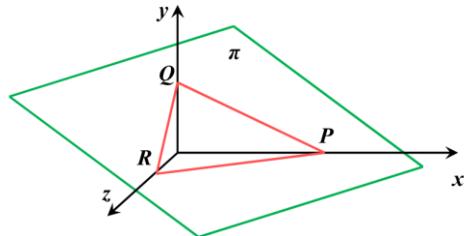
Se un punto  $P'(x', y', z')$  giace sul piano, è soddisfatta l'equazione di cui sopra.

Se si trova al di sotto del piano vale  $ax' + by' + cz' + d < 0$ , mentre se si trova al di sopra del piano si ha  $ax' + by' + cz' + d > 0$ .

In generale un poligono giace su un piano, quindi tutti i vertici e i punti degli spigoli soddisfano l'equazione del piano, come anche i punti all'interno.

In particolare, l'equazione del piano che passa per i punti  $P(p, 0, 0)$ ,  $Q(0, q, q)$  ed  $R(0, 0, r)$  è data da:

$$\frac{x}{p} + \frac{y}{q} + \frac{z}{r} = 1$$



Per quanto riguarda una retta nello spazio, questa è rappresentata dall'intersezione di due piani:

$$a_1x + b_1y + c_1z + d_1 = 0$$

$$a_2x + b_2y + c_2z + d_2 = 0$$

## Le trasformazioni bidimensionali

Le figure bidimensionali possono essere sottoposte a trasformazioni geometriche elementari che consistono in:

- traslazioni
- rotazioni
- cambiamenti di scala.

### Traslazione

La traslazione consiste nel movimento di una figura parallelamente agli assi coordinati e si realizza addizionando a ogni punto  $P(x, y)$  della figura, le quantità  $t_x$  e  $t_y$  corrispondenti agli spostamenti orizzontali e verticali.

Le nuove coordinate  $P'(x', y')$  sono espresse dalle relazioni:

$$x' = x + t_x$$

$$y' = y + t_y$$

In forma matriciale, indicando con  $\mathbf{P} = [x, y]^T$  il vettore colonna a due componenti che rappresenta il punto del piano  $P(x, y)$ , possiamo scrivere:

$$\mathbf{P}' = \mathbf{P} + [t_x, t_y]^T$$

dove  $[t_x, t_y]^T$  è il vettore di traslazione.

Occorre osservare che ogni punto di un oggetto è traslato delle stesse quantità: si tratta di una trasformazione detta di corpo rigido.

Nel caso di un segmento, si ottiene la versione traslata applicando le formule di cui sopra agli estremi del segmento.

Nel caso di un poligono, ogni vertice è sottoposto alla traslazione.

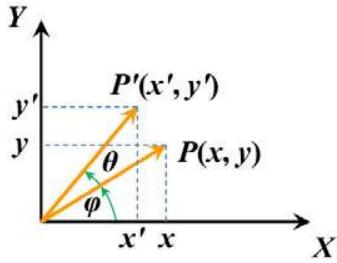
### Rotazione

Consideriamo un segmento con un estremo nell'origine degli assi cartesiani e l'altro in  $P(x, y)$ .

La rotazione del segmento di un angolo  $\theta$  attorno all'origine in verso antiorario (senso positivo) definisce le coordinate  $(x', y')$  dell'estremo  $P'$  secondo le seguenti relazioni:

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta - y \cos \theta$$



Infatti, osservando la figura si possono scrivere le seguenti uguaglianze:

$$r = \sqrt{x^2 + y^2} \quad x = r \cos \varphi \quad y = r \sin \varphi$$

$$x' = r \cos(\varphi + \theta) \quad y' = r \sin(\varphi + \theta)$$

$$x' = r(\cos \theta \cos \varphi - \sin \theta \sin \varphi) \quad y' = r(\cos \theta \sin \varphi - \sin \theta \cos \varphi)$$

$$x' = x \cos \theta - y \sin \theta \quad y' = x \sin \theta - y \cos \theta$$

In forma matriciale possiamo scrivere:

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

cioè:

$$\mathbf{P}' = \mathbf{R} \cdot \mathbf{P}$$

dove  $\mathbf{R}$  è la matrice di rotazione.

$\mathbf{P}$  e  $\mathbf{P}'$  sono i vettori colonna che definiscono l'estremo del segmento di coordinate  $P(x, y)$  e  $P'(x', y')$ .

Nel caso di rotazioni orarie (verso negativo) la matrice di rotazione è modificata considerando che valgono le seguenti relazioni:

$$\begin{aligned} \cos(-\theta) &= \cos \theta \\ \sin(-\theta) &= -\sin \theta \end{aligned}$$

per cui diventa:

$$\mathbf{R} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

in caso di rotazione di un angolo  $\theta$  in **verso orario**.

## Cambiamento di scala

Il cambiamento di scala, detto scalamento, ha lo scopo di ingrandire o ridurre una figura.

Esso è realizzato utilizzando una matrice diagonale; se i coefficienti sono maggiori di 1 l'effetto è una espansione, viceversa si ha una compressione.

La trasformazione di scala avviene sempre rispetto all'origine per cui la figura sottoposta a compressione si avvicina all'origine e viceversa si allontana nel caso di espansione.

Indicando con  $s_x$  il fattore di scala rispetto all'asse  $x$  e con  $s_y$  quello rispetto all'asse  $y$  possiamo scrivere:

$$\begin{aligned} x' &= x s_x \\ y' &= y s_y \end{aligned}$$

che in forma matriciale diventa:

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

Nel caso in cui  $s_x = s_y$  il cambiamento di scala è detto *omogeneo*, altrimenti è detto *differito*.

---

## Coordinate omogenee

---

Per ovviare all'inconveniente di aver a che fare con operatori matriciali diversi a seconda del tipo di trasformazione che si vuole realizzare, è opportuno introdurre al posto delle coordinate cartesiane, le *coordinate omogenee* che fanno passare dallo spazio bi-dimensionale a quello tridimensionale.

La rappresentazione di un punto nel piano bidimensionale con coordinate ordinarie  $\begin{bmatrix} x \\ y \end{bmatrix}$  è definita dalla

terna di coordinate omogenee  $\begin{bmatrix} X \\ Y \\ W \end{bmatrix}$  dove  $W \neq 0$  e tale che  $x = \frac{X}{W}$  e  $y = \frac{Y}{W}$ , nonché da tutte le terne:

$$\begin{bmatrix} kX \\ kY \\ kW \end{bmatrix}$$

con  $k$  numero reale diverso da zero; ciò significa che esiste nello spazio tridimensionale una infinità di rappresentazioni equivalenti di un punto nel piano.

La relazione tra un punto di coordinate omogenee  $\begin{bmatrix} X \\ Y \\ W \end{bmatrix}$  e il suo corrispondente nel piano cartesiano è

data da:

$$x = \frac{X}{W} \quad y = \frac{Y}{W}$$

ma, per evitare di dover eseguire queste divisioni si preferisce utilizzare la **rappresentazione omogenea normalizzata**:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

con  $X = x, Y = y$  e  $W = 1$ .

Un modo intuitivo per immaginare le coordinate omogenee così definite (alle quali faremo riferimento in seguito) consiste nel considerare un sistema di assi cartesiani  $x, y$  in uno spazio tridimensionale con coordinata  $z$  sempre uguale a 1 (piano di equazione  $z = 1$ ).

Le trasformazioni geometriche espresse in coordinate omogenee saranno anch'esse determinate da matrici con un'unità in più per ogni dimensione, saranno cioè matrici di dimensione  $3 \times 3$ .

Analizziamo le diverse trasformazioni in questa nuova rappresentazione.

### Traslazione in coordinate omogenee

---

Per traslare un punto  $P$  di coordinate omogenee  $[x, y, 1]^T$  di una quantità  $t_x$  lungo l'asse  $x$  e di  $t_y$  lungo l'asse  $y$  occorre eseguire il prodotto:

$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

### Rotazione in coordinate omogenee

---

La rotazione del punto  $P$  di un angolo  $\theta$  è espressa dalla relazione:

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

### Cambiamento di scala in coordinate omogenee

---

Lo scalamento di un punto  $P$  di un fattore  $s_x$  relativamente all'asse  $x$  e di un fattore  $s_y$  relativamente all'asse  $y$  è realizzato dal prodotto:

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

---

## Vettori

---

### Prodotto scalare

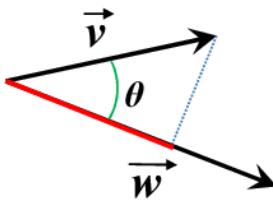
Dati due vettori  $\vec{v}$  e  $\vec{w}$  il *prodotto scalare* (o prodotto interno) è definito come:

$$\vec{v} \cdot \vec{w} = |\vec{v}| |\vec{w}| \cos \theta$$

cioè il prodotto dei moduli (lunghezze dei vettori) per il coseno dell'angolo tra i vettori (compreso tra 0 e  $\pi$ ) oppure, in altro modo, come il prodotto del modulo di un vettore per la proiezione dell'altro sulla direzione del primo.

Il risultato è un valore scalare (da qui il nome del prodotto), che nella rappresentazione algebrica è la somma dei prodotti delle componenti omologhe:

$$\vec{v} \cdot \vec{w} = v_x w_x + v_y w_y + v_z w_z$$



Casi particolari si riferiscono al valore di  $\theta = 0$  (vettori paralleli) per cui il prodotto scalare di due versori vale 1 mentre quello di due vettori è il prodotto del loro modulo; nel caso in cui  $\theta = \frac{\pi}{2}$  (vettori ortogonali) il prodotto scalare è nullo.

### Prodotto vettoriale

Un altro prodotto che si può realizzare tra due vettori è il *prodotto vettoriale* (o prodotto esterno) che restituisce un vettore e si indica con  $\vec{v} \wedge \vec{w}$ .

Se l'angolo tra i due vettori è  $\theta$  (con  $0 \leq \theta \leq \pi$ ) il modulo del prodotto è definito come:

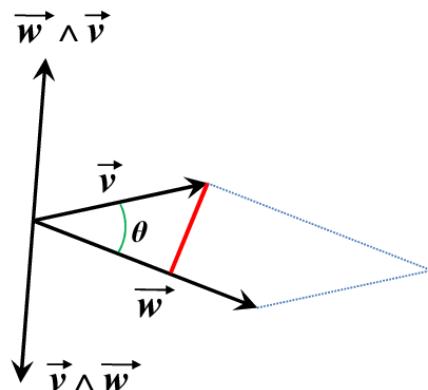
$$|\vec{v} \wedge \vec{w}| = |\vec{v}| |\vec{w}| |\sin \theta|$$

che è l'area del parallelogramma di lati  $\vec{v}$  e  $\vec{w}$  mentre la direzione è perpendicolare al piano di  $\vec{v}$  e  $\vec{w}$  e il verso si determina con la regola della mano destra: se il pollice è nella direzione di  $\vec{v}$  e l'indice in quella di  $\vec{w}$ , allora il medio definisce il verso.

Quindi se  $\vec{n}$  è un versore ortogonale al piano contenente  $\vec{v}$  e  $\vec{w}$  il cui verso è definito con la regola della mano destra, allora:

$$\vec{v} \wedge \vec{w} = |\vec{v}| |\vec{w}| |\sin \theta| \vec{n}$$

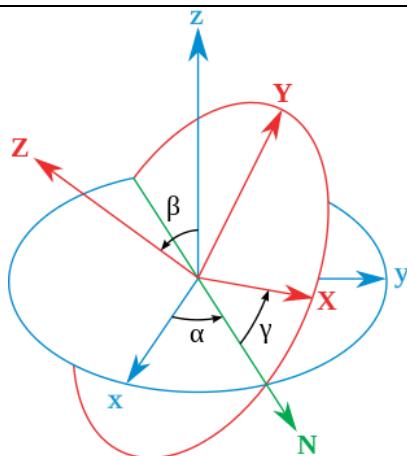
Il prodotto vettoriale non è commutativo, ma *anticommutativo*, cioè  $\vec{v} \wedge \vec{w} = -\vec{w} \wedge \vec{v}$ .



## Angoli di Eulero

Gli angoli di Eulero sono stati introdotti per descrivere l'orientamento di un corpo rigido nello spazio.

### Definizione



Angoli di Eulero - Il sistema fisso  $(xyz)$  è rappresentato in blu, e il sistema ruotato  $(XYZ)$  è rappresentato in rosso.  
La linea dei nodi, indicata con  $N$ , è rappresentata in verde.

Gli angoli di Eulero descrivono la posizione di un sistema di riferimento  $XYZ$  solidale con un corpo rigido attraverso una serie di rotazioni a partire da un sistema di riferimento fisso  $xyz$ .

I due sistemi di riferimento coincidono nell'origine.

Se i piani  $xy$  e  $XY$  sono distinti, si intersecano in una retta (passante per l'origine) detta *linea dei nodi* ( $N$ ).

Se i piani coincidono, si definisce la linea dei nodi  $N$  come l'asse  $X$ .

Gli angoli di Eulero sono i tre angoli seguenti:

- $\alpha$  è l'angolo tra l'asse  $x$  e la linea dei nodi.  
Detto angolo di *precessione*, è definito in  $[0, 2\pi]$  oppure in  $[-\pi, \pi]$
- $\beta$  è l'angolo tra gli assi  $z$  e  $Z$ .  
Detto angolo di *nutazione*, è definito in  $[0, \pi]$  oppure in  $[-\frac{\pi}{2}, \frac{\pi}{2}]$
- $\gamma$  è l'angolo tra la linea dei nodi e l'asse  $X$ .  
Detto angolo di *rotazione propria*, è definito in  $[0, 2\pi)$  oppure in  $[-\pi, \pi)$ .

### Significato algebrico

Dal punto di vista dell'algebra lineare il passaggio dal sistema di riferimento fisso  $xyz$  a quello ruotato  $XYZ$  equivale ad operare un cambiamento di base, ovvero passare da una base  $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$  (con  $\hat{e}_1$ ,  $\hat{e}_2$  ed  $\hat{e}_3$  rispettivamente i versori degli assi  $x$ ,  $y$  e  $z$ ) ad una  $(\hat{i}, \hat{j}, \hat{k})$  (ovvero i versori degli assi  $X$ ,  $Y$  e  $Z$ ) tramite una matrice di cambiamento di base  $A$ .

La matrice  $A$  è una matrice quadrata  $3 \times 3$  ortogonale che rappresenta una rotazione nello spazio.

Gli angoli di Eulero permettono di rappresentare  $A$  in forma relativamente semplice come moltiplicazione di 3 matrici di rotazione lungo i tre assi  $x$ ,  $y$  e  $z$ .

In altre parole, la rotazione descritta da  $A$  può essere effettuata in tre passi distinti:

1. Rotazione intorno all'asse  $z$  di un angolo  $\alpha$ , facendo così coincidere l'asse  $x$  con  $N$ .

Ciò comporterà un cambiamento di base  $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \rightarrow (\hat{e}'_1, \hat{e}'_2, \hat{e}'_3)$ , con  $\hat{e}'_1 = \hat{n}$  ed  $\hat{e}'_3 = \hat{e}_3$ , avendo chiamato  $\hat{n}$  il versore della linea dei nodi.

La matrice di rotazione sarà:

$$R_\alpha = \begin{pmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. Rotazione intorno all'asse  $N$  di un angolo  $\beta$ , con un conseguente cambio di base  $(\hat{n}, \hat{e}'_2, \hat{e}'_3) \rightarrow (\hat{n}, \hat{e}''_2, \hat{e}''_3)$ .

Stavolta la matrice di rotazione sarà:

$$R_\beta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \beta & \sin \beta \\ 0 & -\sin \beta & \cos \beta \end{pmatrix}$$

3. Rotazione intorno all'asse  $Z$  di un angolo  $\gamma$ .

Il cambiamento di base sarà  $(\hat{n}, \hat{e}''_2, \hat{e}''_3) \rightarrow (\hat{i}, \hat{j}, \hat{k})$ , con  $\hat{k} = \hat{e}''_3$ , e la matrice di rotazione:

$$R_\gamma = \begin{pmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Il cambiamento di base sopra descritto si può formalizzare come:

$$\begin{pmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{pmatrix} = R_\gamma R_\beta R_\alpha \begin{pmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \end{pmatrix}$$

mentre il passaggio inverso:

$$\begin{pmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \end{pmatrix} = (R_\gamma R_\beta R_\alpha)^T \begin{pmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{pmatrix} = R_\alpha^T R_\beta^T R_\gamma^T \begin{pmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{pmatrix}$$

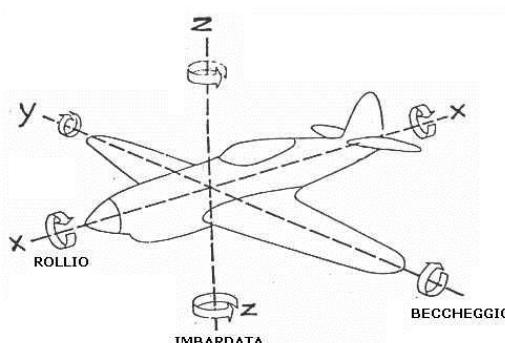
Occorre puntualizzare che la sequenza descritta in questa sezione è solo una delle 12 possibili per operare il cambiamento di base indicato.

Dal nome degli assi intorno ai quali si sono effettuate le singole rotazioni, essa prende il nome di  $ZXZ$ .

Le altre possibili sono  $XZX$ ,  $XYX$ ,  $YXY$ ,  $YZY$ ,  $ZYZ$ ,  $XZY$ ,  $XYZ$ ,  $YXZ$ ,  $ZYX$  e  $ZXY$ .

Le 12 sequenze sono ottenute tramite tutte le permutazioni possibili con gli assi uguali non consecutivi.

Una particolare variante degli angoli di Eulero, usata in aeronautica e in robotica, è quella degli angoli di Tait-Bryan. In questo caso gli angoli  $\alpha$ ,  $\beta$  e  $\gamma$  prendono il nome, rispettivamente, di angoli di imbardata, rollio e beccheggio.



## 4. Intersezioni

Per utilizzare gli algoritmi di **ray tracing** e **radiosity** visti in precedenza, è necessario riuscire a determinare in modo completo e corretto le intersezioni fra i raggi (quelli luminosi o quelli che partono dall'osservatore) e i box di contenimento degli oggetti della scena.

---

### Raggio

---

Un **raggio** è rappresentato parametricamente come combinazione dell'origine e di un vettore direzione unitario  $\mathbf{d}$ :

$$\mathbf{R}(t) = \mathbf{R}_0 + t\mathbf{R}_d$$

dove  $\mathbf{R}_0 = (x_0, y_0, z_0)$  è l'origine,  $\mathbf{R}_d = (x_d, y_d, z_d)$  è la sua direzione e  $t > 0$ .

Per semplicità possiamo pensare di scomporre il raggio nelle sue tre componenti  $(x, y, z)$  scomponendo in queste componenti sia il punto iniziale che il vettore di direzione:

$$\begin{aligned}x(t) &= x_0 + tx_d \\y(t) &= y_0 + ty_d \\z(t) &= z_0 + tz_d\end{aligned}$$

Al posto di un vettore direzione possiamo avere a disposizione due punti  $p_1$  e  $p_2$  che si trovano sul raggio, avendo come equazione:

$$\begin{aligned}x(t) &= x_1 + t(x_2 - x_1) \\y(t) &= y_1 + t(y_2 - y_1) \\z(t) &= z_1 + t(z_2 - z_1)\end{aligned}$$

Spesso conviene normalizzare la direzione del raggio.

I raggi sono un ottimo strumento per le rilevazioni delle intersezioni perché sono dotati di una semplice equazione parametrica.

I raggi sono un oggetto geometrico importante utilizzato nel rilevamento dell'intersezione perché:

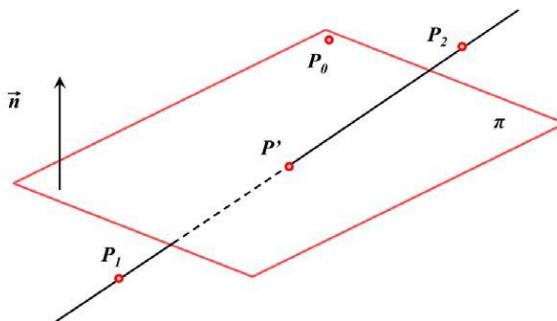
- a volte sono buoni modelli di percorsi praticati da oggetti in movimento.
- possono essere utilizzati come un metodo computazionalmente efficiente per eseguire un rilevamento approssimativo dell'intersezione.

---

### Intersezione raggio - piano

---

Dato un piano caratterizzato da un punto  $P_0$  di componenti  $P_0(x_0, y_0, z_0)$  e da una normale  $\mathbf{n}$  di componenti  $(n_x, n_y, n_z)$ , possiamo usarne l'equazione per ricavare il punto di intersezione fra il piano e il raggio partendo dalle equazioni dei due oggetti.



Equazione del piano:

$$Ax + By + Cz + D = 0$$

dove il vettore  $(A, B, C)$  rappresenta la normale al piano, oppure:

$$n_x(x - x_0) + n_y(y - y_0) + n_z(z - z_0) = 0$$

$$\nearrow \quad x(t) = x_0 + tx_d$$

$$\text{L'equazione del raggio era } \mathbf{R}(t) = \mathbf{R}_0 + t\mathbf{R}_d \rightarrow \begin{aligned} y(t) &= y_0 + ty_d \\ \searrow \quad z(t) &= z_0 + tz_d \end{aligned}$$

Ora sostituisco le equazioni del raggio nell'equazione del piano e la risolvo per  $t$ :

$$Ax + By + Cz + D = 0$$

$$A(x_0 + tx_d) + B(y_0 + ty_d) + C(z_0 + tz_d) + D = 0$$

$$Ax_0 + Atx_d + By_0 + Bty_d + Cz_0 + Ctz_d + D = 0$$

$$t(Ax_d + By_d + Cz_d) + Ax_0 + By_0 + Cz_0 + D = 0$$

$$t(Ax_d + By_d + Cz_d) = -(Ax_0 + By_0 + Cz_0 + D)$$

$$t = -\frac{Ax_0 + By_0 + Cz_0 + D}{Ax_d + By_d + Cz_d} = -\frac{N\mathbf{R}_0 + D}{N\mathbf{R}_d}$$

- Se  $N\mathbf{R}_d = 0$  allora il raggio e il piano sono paralleli, perciò non c'è intersezione
- se  $t < 0$  l'intersezione è precedente al punto di partenza del raggio e quindi non c'è intersezione
- se  $t > 0$  allora è un  $t_i$  (cioè il  $t$  dell'intersezione) e le componenti del punto di intersezione sono date da  $(x, y, z) = \mathbf{R}_0 + t_i\mathbf{R}_d$ .

Questo metodo di calcolo dell'intersezione può essere utile per verificare se un raggio interseca una determinata faccia di un oggetto della scena: si determina il piano su cui è posata la faccia interessata e l'intersezione con il raggio.

Se esiste l'intersezione, ciò non implica che il raggio intersechi la faccia stessa, per questo motivo è necessario fare ulteriori controlli di contenimento del punto di intersezione nella faccia.

### Intersezione raggio - sfera

$$\nearrow \quad x(t) = x_0 + tx_d$$

$$\text{Consideriamo il solito raggio determinato dall'equazione } \mathbf{R}(t) = \mathbf{R}_0 + t\mathbf{R}_d \rightarrow \begin{aligned} y(t) &= y_0 + ty_d \\ \searrow \quad z(t) &= z_0 + tz_d \end{aligned}$$

e una sfera caratterizzata dall'equazione  $|P - C|^2 \leq r_s^2$  dove:

- $C(x_c, y_c, z_c)$  è il centro della sfera
- $P(x_p, y_p, z_p)$  un punto qualsiasi posto nello spazio all'interno e sulla superficie della sfera
- $r_s$  il raggio della sfera.

Per il calcolo dell'intersezione fra raggio e sfera presentiamo ora due metodi alternativi: il metodo analitico/algebrico ed il metodo geometrico.

### Metodo analitico/algebrico

Il *metodo analitico* prevede la sostituzione dell'equazione della retta in quella della sfera  $|P - C|^2 = r^2$ :

$$|(\mathbf{R}_0 + t\mathbf{R}_d) - C|^2 = r^2$$

$$(\mathbf{R}_0 + t\mathbf{R}_d)(\mathbf{R}_0 + t\mathbf{R}_d) - C - C = r^2$$

$$[\mathbf{R}_d + (\mathbf{R}_0 - C)][\mathbf{R}_d + (\mathbf{R}_0 - C)] = r^2$$

$$t^2\mathbf{R}_d \cdot \mathbf{R}_d + 2t\mathbf{R}_d(\mathbf{R}_0 - C) + (\mathbf{R}_0 - C)(\mathbf{R}_0 - C) - r^2 = 0$$

ed essendo  $\mathbf{d}$  un versore e quindi  $\mathbf{d} \cdot \mathbf{d} = 1$ :

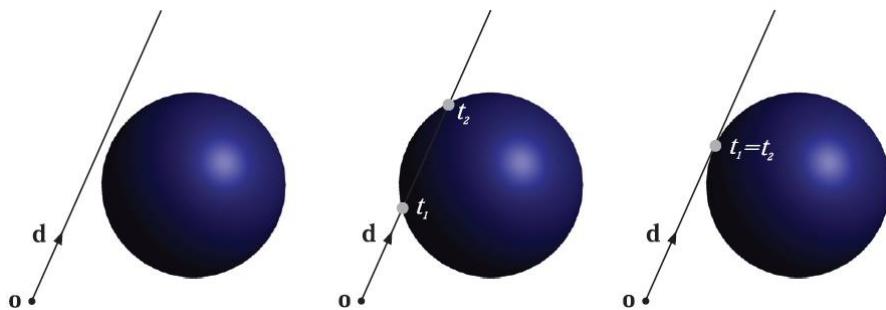
$$t^2 + \underbrace{2\mathbf{d}(R_0 - C)t}_{B} + \underbrace{(R_0 - C)(R_0 - C) - r^2}_{C} = 0$$

Ottengo dunque un'equazione di secondo grado in  $t$ :

$$At^2 + Bt + C = 0$$

Tale eq. può avere come soluzione:

- **nessun valore reale**: il raggio non interseca la sfera
- **1 valore (2 coincidenti)**: il raggio è tangente alla sfera
- **2 valori distinti**: il raggio è secante alla sfera.



Ottenuti i valori di  $t$  questi vengono sostituiti nell'equazione del raggio per ottenere il punto di intersezione.

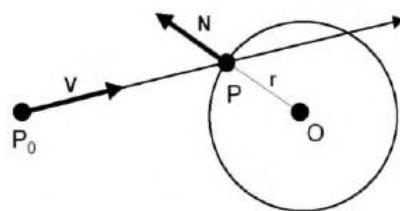
#### NORMALE AL PUNTO DI INTERSEZIONE

In alcune situazioni può essere utile conoscere la normale alla superficie nel punto di intersezione, ad esempio quando abbiamo bisogno di calcolare l'equazione di illuminazione in un determinato punto della superficie.

Per ottenere la normale è necessario calcolare:

$$\mathbf{N} = \frac{\mathbf{P} - \mathbf{O}}{|\mathbf{P} - \mathbf{O}|}$$

dove  $P$  è il punto di intersezione e  $O$  il centro della sfera.



#### Metodo geometrico

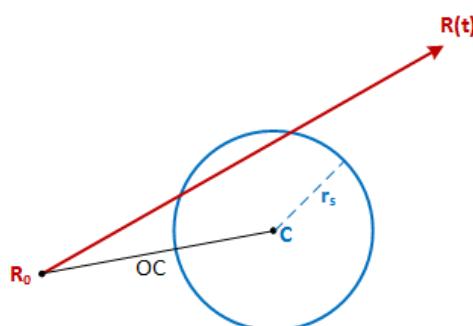
---

##### PASSO 1: CAPIRE SE L'ORIGINE DEL RAGGIO È INTERNA O ESTERNA ALLA SFERA

Determiniamo il vettore che congiunge l'origine del raggio al centro della sfera  $OC = C - R_0$ .

Calcoliamo la norma del vettore come  $\|OC\|^2$ .

Se  $\|OC\|^2 < r_s^2$  il raggio parte internamente alla sfera (si va direttamente al passo 4) altrimenti l'origine è esterna.

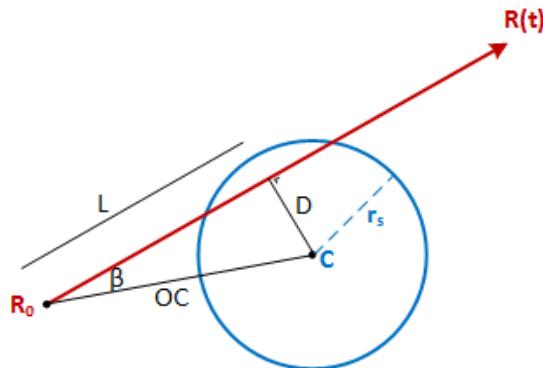


## PASSO 2: ESCLUSIONE DELL'INTERSEZIONE IN CASO DI RAGGIO IN DIREZIONE OPPSTA

Calcoliamo la proiezione del vettore  $OC$  sul raggio  $R(t)$ , che è anche la distanza da  $R_0$  al punto più vicino al centro della sfera:

$$L = \|OC\| \cos \beta = \|OC\| \cdot \|R_d\| \cos \beta = OC \cdot R_d$$

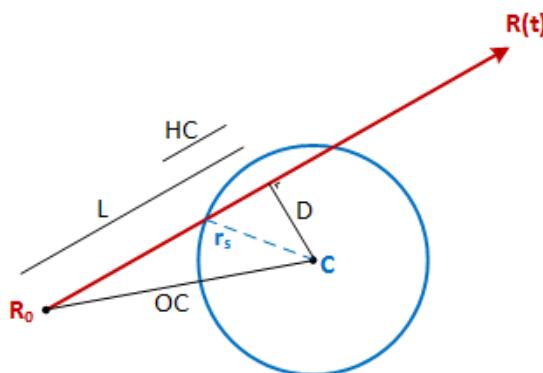
Se il raggio è esterno alla sfera e punta lontano da essa  $L$  deve essere  $< 0$  e il raggio manca la sfera.



## PASSO 3: PUNTO DEL RAGGIO PIÙ VICINO ALLA SFERA E SUA DISTANZA DAL CENTRO DELLA SFERA

Determiniamo ora  $D^2 = \|OC\|^2 - L^2$  cioè il quadrato della distanza fra il centro della sfera e il punto sul raggio che passa più vicino al centro della sfera.

- Se  $D^2 < r_s^2$  allora il raggio è *secante* e possiamo determinare le due intersezioni facendo uso di  $HC^2 = r_s^2 - D^2$  che rappresenta la distanza fra i punti di intersezione e il punto del raggio più vicino al centro della sfera.
- Se  $D^2 > r_s^2$  allora il raggio non interseca la sfera.
- Se  $D^2 = r_s^2$  allora il raggio è tangente.



## PASSO 4: CALCOLO DELLE INTERSEZIONI E DELLE NORMALI

Determiniamo il parametro  $t$  del raggio come  $t = L - HC$  e  $t = L + HC$ .

Tramite i parametri troviamo le intersezioni come  $I = (x, y, z) = R_0 + tR_d$ .

Infine riportiamo il calcolo della normale nel punto come (negato se il raggio ha origine dentro la sfera):

$$N = \frac{I - S_c}{S_r}$$

### Intersezione raggio - triangolo

Come detto precedentemente, definire che c'è un'intersezione fra il piano su cui giace la faccia di un oggetto e un raggio non implica che il raggio intersechi la faccia stessa.

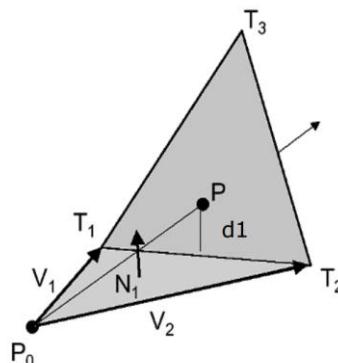
Considerando la faccia come un triangolo (la primitiva più di moda): come possiamo definire l'intersezione fra il raggio e il triangolo?

Dato un triangolo con vertici  $(T_1, T_2, T_3)$  ed il raggio definito da  $\mathbf{R}(t) = P_0 + t\mathbf{R}_d$ :

- 1) calcoliamo i valori  $V_1 = T_1 - P_0$  e  $V_2 = T_2 - P_0$
- 2) calcoliamo poi la normale alla faccia definita dai vertici  $P_0, T_1, T_2$  (con la regola della mano destra e il prodotto scalare  $N_1 = V_2 \times V_1$ )
- 3) ne calcoliamo il versore dividendolo per il suo modulo  $N_1 = \frac{N_1}{\text{mod}(N_1)}$
- 4) infine calcolo  $d_1 = (P - P_0)N_1$ .

Effettuo questi calcoli per ogni coppia di vertici (ora abbiamo usato  $T_1$  e  $T_2$  ma poi useremo  $T_2$  e  $T_3$  e infine  $T_3$  e  $T_1$ ) e definiamo  $d_1, d_2$  e  $d_3$ : se tutti e tre sono maggiori di 0 allora il raggio interseca il triangolo, altrimenti passa fuori.

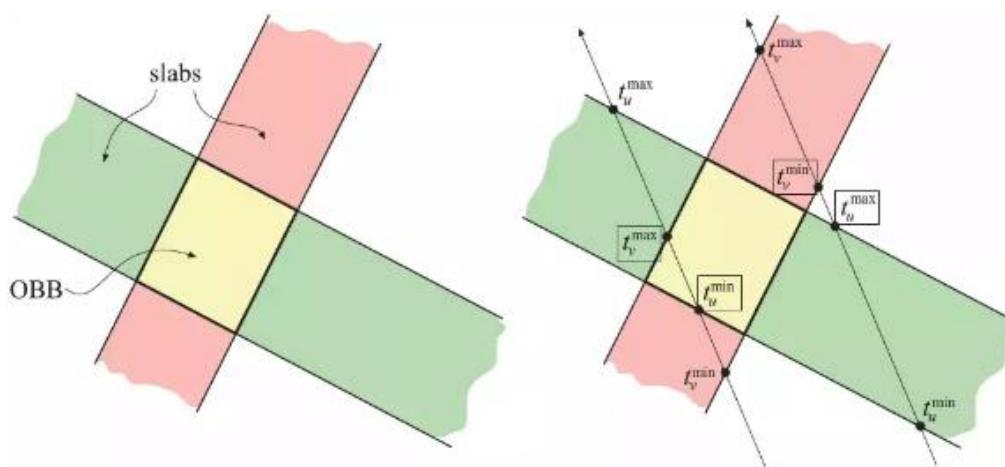
Se calcoliamo le normali alle facce con la regola della mano sinistra i valori  $d_1, d_2$  e  $d_3$  dovranno essere tutti e tre minori di 0.



### Intersezione raggio - volume di contenimento (metodo slabs)

Ciascun **AABB** o **OBB** è caratterizzato da coppie di piani paralleli fra loro che prendono il nome di **slab**: è naturale pensare quindi che il calcolo dell'intersezione fra i raggi e i volumi di contenimento passi per il calcolo dell'intersezione fra i raggi e gli slab.

Consideriamo per generalità l'intersezione fra un raggio e un OBB ma il meccanismo può essere usato per diversi volumi di contenimento, quali AABB o  $k$ -dop.



- 1) Si considera un box come un insieme di 3 slabs in 3D (2 nel 2D).
- 2) Per ogni slab, le intersezioni sono caratterizzate da un valore  $t$  minimo  $t_i^{\min}$  e massimo  $t_i^{\max}$  per  $i = u, v, w$  in un sistema cartesiano associato ad assi  $u, v$  e  $w$ .
- 3) Calcoliamo quindi i valori di  $t^{\min} = \max(t_u^{\min}, t_v^{\min}, t_w^{\min})$  e  $t^{\max} = \min(t_u^{\max}, t_v^{\max}, t_w^{\max})$ .
- 4) Se  $t^{\min} \leq t^{\max}$  c'è intersezione tra raggio e box, in caso contrario non c'è intersezione.

## Intersezione raggio - poligono

Estendiamo ora ciò che è stato detto nelle sezioni precedenti per il calcolo dell'intersezione fra raggio e il triangolo, analizzando il calcolo dell'intersezione fra un raggio e un poligono a  $n$  vertici.

Il poligono in questione può essere visto come una lista ordinata di vertici  $\{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{n-1}\}$  dove ogni vertice crea uno spigolo con il vertice successivo.

Il piano su cui è posato il poligono è definito come  $\pi_p = \mathbf{n}_p \mathbf{P} + d_p = 0$ .

Consideriamo per prima l'intersezione fra il raggio e  $\pi_p$ , calcolata facilmente sostituendo  $\mathbf{P}$  con il raggio:

$$\mathbf{n}_p(\mathbf{o} + t\mathbf{d}) + d_p = 0$$

che dà un valore di  $t$  pari a:

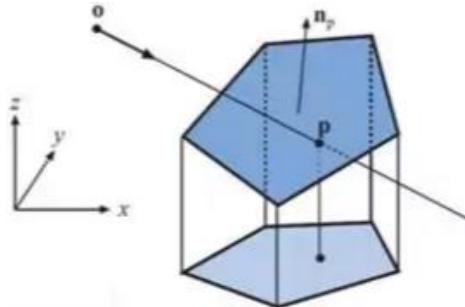
$$t = \frac{-d_p - \mathbf{n}_p \cdot \mathbf{o}}{\mathbf{n}_p \cdot \mathbf{d}}$$

Se il denominatore è minore di un epsilon molto piccolo  $|\mathbf{n}_p \cdot \mathbf{d}| < \varepsilon$ , allora il raggio è considerato parallelo al piano su cui è posato il poligono e perciò non c'è intersezione.

In caso contrario il punto di intersezione  $\mathbf{P}$  è calcolato come  $\mathbf{P} = \mathbf{o} + t\mathbf{d}$  usando il valore di  $t$  calcolato sopra.

Il passo successivo è determinare se l'intersezione  $\mathbf{P}$  è interna o meno al poligono, calcolo che può essere fatto passando da tre a due dimensioni.

Questo può essere fatto proiettando tutti i vertici e  $\mathbf{P}$  su uno dei piani costruibili tramite gli assi (xy or xz or yz), quello in cui l'area della proiezione del poligono è massimizzata<sup>1</sup>.



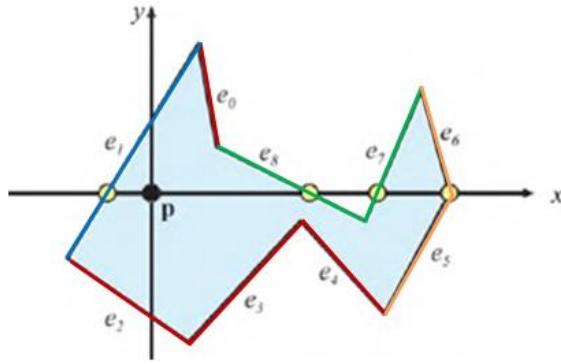
Ora che ci troviamo in uno spazio bidimensionale possiamo fare un passo successivo: trasliamo ciascun punto del poligono di un delta pari alle coordinate di  $\mathbf{P}$ , in modo che  $\mathbf{P}$  diventi l'origine.

Matematicamente per ciascun vertice  $\mathbf{P}_{old}$  si calcola il rispettivo punto traslato  $\mathbf{P}_{new} = \mathbf{P}_{old} - \mathbf{P}$ .

A questo punto, per capire se l'origine è interna o meno al nuovo poligono abbiamo solo bisogno di camminare sul bordo del poligono e contare il numero di volte in cui gli spigoli del poligono attraversano l'asse  $x$  ([regola pari-dispari](#) o [crossing test](#)): se il numero di attraversamenti è pari allora l'origine è all'interno.

<sup>1</sup> Per determinare quale piano usare si considera la variazione di coordinate dei vertici e si determina quale coordinata provoca la minor variazione (cerchiamo il piano formato dagli assi "più parallelo" a  $\pi$ ).

Quella coordinata viene ignorata e vengono mantenute solo le altre due.



I lati  $e_0, e_2, e_3, e_4$  non attraversano l'asse  $x$ .

L'intersezione tra il lato  $e_1$  e l'asse  $x$  deve essere calcolata ma non ci porterà ad un crossing dato che l'intersezione ha una componente  $x$  negativa.

I lati  $e_7$  ed  $e_8$  incrementeranno il numero di crossing visto che i vertici di questi lati hanno componente  $x$  positiva.

Infine i lati  $e_5$  ed  $e_6$  condividono un vertice che ha  $y = 0$  e  $x > 0$  e quindi insieme incrementeranno il numero di crossing di uno.

Considerando i vertici sull'asse  $x$  come sopra il raggio,  $e_5$  è classificato come attraversante il raggio,  $e_6$  come sopra il raggio.

Algoritmo:

1. rappresentiamo un bordo come raggi  $B_i = v_{i+1} + t_b(v_{i+1} - v_i)$
2. far partire un raggio dall'intersezione  $P$  in ogni direzione  $R = P + t_p(1,0)$  funziona bene
3. per intersecare il raggio  $R$  con il poligono  $P$ , interseziona  $R$  con ogni confine
4. un'intersezione è valida sse  $t_p \geq 0$  e  $0 \leq t_b < 1.0$
5. se il numero di intersezioni è:
  - dispari → dentro
  - pari → fuori.

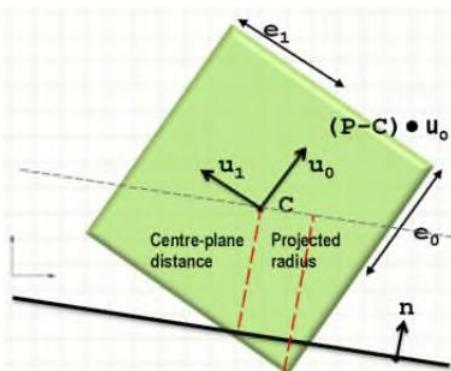
### Intersezione piano - box

Ora cerchiamo le intersezioni fra piani e box di contenimento.

Dato che i vertici del box rappresentano i punti più lontani dal centro del box, questi devono essere confrontati solo per la distanza rispetto al piano.

Proiettando ciascun vertice lungo il piano normale e accumulando il risultato, si ottiene il raggio di proiezione massimo.

Il confronto di questo con la distanza del centro di OBB dal piano determina se c'è intersezione.



## Metodo algebrico

Un primo semplice modo per calcolare se un box interseca un piano  $\pi = \mathbf{n}P + d = 0$  è quello di inserire tutti i vertici del box nell'equazione del piano: se troviamo sia valori positivi che negativi, allora i vertici si trovano da ambe le parti del piano perciò c'è intersezione.

## Metodo geometrico

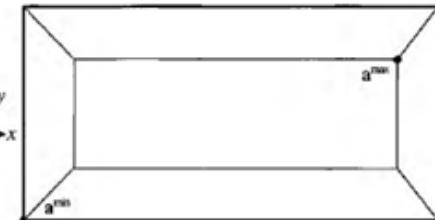
Un metodo un po' più complesso è quello di cercare di confrontare la posizione del box con quella del piano, confrontando l'estensione massima del box (lungo la sua diagonale maggiore) con la distanza fra il centro del box e il piano, calcolato lungo la sua normale.

### AABB

Assumiamo di lavorare con un AABB chiamato  $A$ , centrato in un punto  $c$  e di semi-diagonale (distanza dal centro a un vertice)  $h$ .

Possiamo notare che  $c$  e  $h$  sono facilmente derivabili a partire dall'angolo minimo e massimo (rispettivamente  $a_{min}$  e  $a_{max}$ ) come:

$$c = \frac{a_{max} + a_{min}}{2} \quad h = \frac{a_{max} - a_{min}}{2}$$



Dato  $A$  confrontiamolo con un piano di equazione  $\pi = \mathbf{n}P + d = 0$ .

L'idea è di calcolare l'**estensione** ( $e$ ) di  $A$  quanto viene proiettato sulla normale al piano  $\mathbf{n}$ .

Questo viene fatto calcolando il prodotto scalare fra ciascuna semi-diagonale  $\mathbf{g}_i$  e la normale  $\mathbf{n}$ :

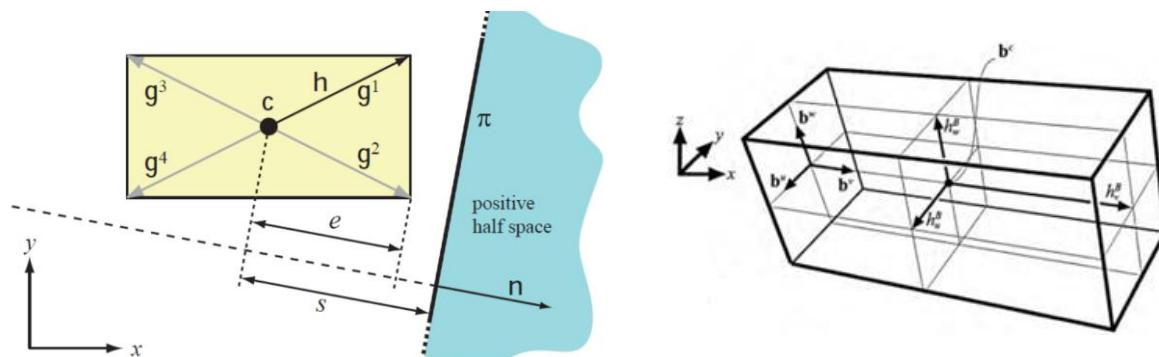
$$e = h_x |\mathbf{n}_x| + h_y |\mathbf{n}_y| + h_z |\mathbf{n}_z|$$

Una volta calcolata e la confrontiamo con la distanza fra il centro di  $A$  e il piano, determinata come:

$$s = \mathbf{c}\mathbf{n} + d$$

Se consideriamo che il "fuori" del piano è il suo semispazio positivo, possiamo determinare che:

- il box è interamente fuori dal piano se  $s - e > 0$
- mentre è interamente dentro  $s + e < 0$
- se nessuna delle due condizioni è soddisfatta allora il box interseca il piano.



### OBB

Il metodo geometrico illustrato per l'AABB può essere esteso agli OBB, modificando la definizione di estensione, che diventa:

$$e = \mathbf{h}B_u |\mathbf{n}b_u| + \mathbf{h}B_v |\mathbf{n}b_v| + \mathbf{h}B_w |\mathbf{n}b_w|$$

---

## Intersezioni fra volumi di contenimento

---

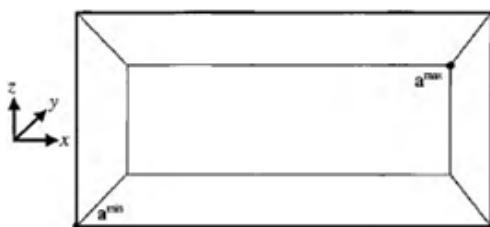
### Intersezione sfera - sfera

Per determinare la presenza di un'intersezione si può facilmente calcolare la distanza fra i centri delle due sfere e confrontare tale distanza con la somma dei rispettivi raggi:

- se la distanza è maggiore allora non c'è intersezione
- se è uguale c'è un punto di intersezione
- se è minore allora ci c'è una circonferenza di intersezione.

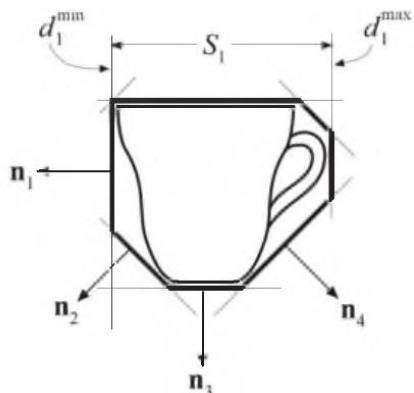
### Intersezione AABB - AABB

Dato che ciascun AABB è riconoscibile tramite due punti (i vertici opposti) possiamo usare questi punti per determinare l'intersezione: avendo due AABB che chiamiamo  $A$  e  $B$ , li consideriamo disgiunti se  $a^{\min} > b^{\max}$  oppure  $b^{\min} > a^{\max}$ , altrimenti c'è intersezione.



### Intersezione $k$ -DOP – $k$ -DOP

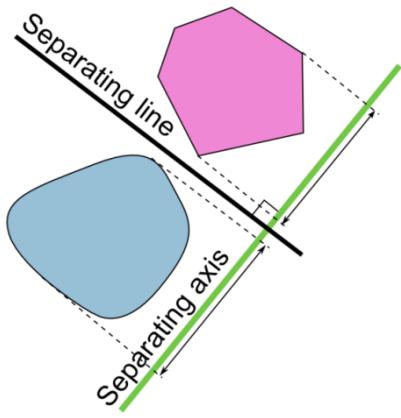
Dati due  $k$ -DOP  $A$  e  $B$  si testano tutte le coppie di slab ( $SA_i, SB_i$ ) per l'intersezione ( $s_i = SA_i \cap SB_i$ ). Se l'intersezione è sempre vuota  $s_i = \emptyset$  allora i  $k$ -DOP sono disgiunti, altrimenti si dichiarano come intersecati se e solo se le intersezioni sono non vuote per  $1 \leq i \leq \frac{k}{2}$ .



### Separating axis theorem in 2D

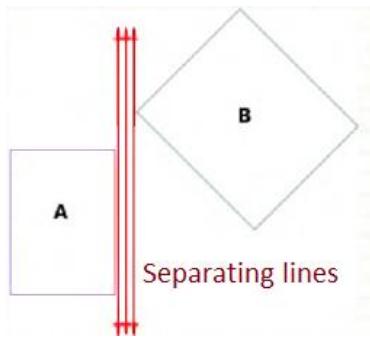
Il teorema dell'asse di separazione ci informa che due poligoni convessi non si intersecano se e solo se esiste una linea tale per cui le proiezioni dei due poligoni sulla linea non si intersecano.

Tale linea prende il nome di **asse di separazione** (*separating axis*).

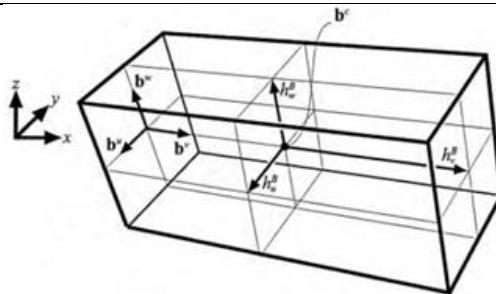


Equivalentemente il teorema può anche essere esplicato come: due poligoni non si intersecano solo se esiste una linea che divide i poligoni in modo che si trovino interamente sui due lati opposti della linea, che viene detta linea di separazione ed è **perpendicolare** all'asse di separazione.

Notiamo come è possibile che ci siano più assi di separazione o linee di separazione paralleli fra loro, ma per semplicità consideriamo ogni insieme di segmenti paralleli come un segmento solo.



### Intersezione OBB - OBB



L'algoritmo per determinare l'intersezione fra due OBB si basa sul teorema dell'asse di separazione.

Il test viene fatto nel sistema di coordinate formate dal centro di  $A$  e dagli assi.

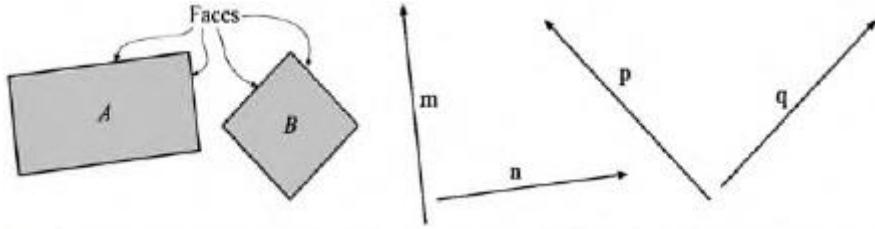
Questo significa che l'origine è  $\alpha^c = (0,0,0)$  e che gli assi principali nel sistema di coordinate sono  $\alpha^u = (1,0,0)$ ,  $\alpha^v = (0,1,0)$  e  $\alpha^w = (0,0,1)$ .

Inoltre si assume che  $B$  sia posizionato relativamente ad  $A$ , con una traslazione  $t$  e una rotazione  $R$  (matrice).

Secondo il test dell'asse di separazione, è sufficiente trovare un asse che separa  $A$  e  $B$  per essere sicuri che essi siano disgiunti (non si sovrappongono).

Per determinare l'asse di separazione testiamo 15 assi:

- 3 a partire dalle facce di  $A$  (perpendicolari alle facce)
- 3 a partire da quelle di  $B$  (perpendicolari alle facce)
- $9 = 3 \cdot 3$  come combinazione degli spigoli di  $A$  e  $B$ .



Per determinare se due OBB  $A$  e  $B$  si sovrappongono, è possibile utilizzare il test dell'asse di separazione.

Qui è mostrato in due dimensioni.

Gli assi separatori devono essere ortogonali alle facce di  $A$  e  $B$ .

Gli assi  $m$  ed  $n$  sono ortogonali alle facce di  $A$ , mentre gli assi  $p$  e  $q$  sono ortogonali alle facce di  $B$ .

Gli OBB vengono quindi proiettati sugli assi. Se entrambe le proiezioni si sovrappongono su tutti gli assi, gli OBB si sovrappongono, altrimenti no.

Quindi è sufficiente trovare un asse che separa le proiezioni per sapere che gli OBB non si sovrappongono.

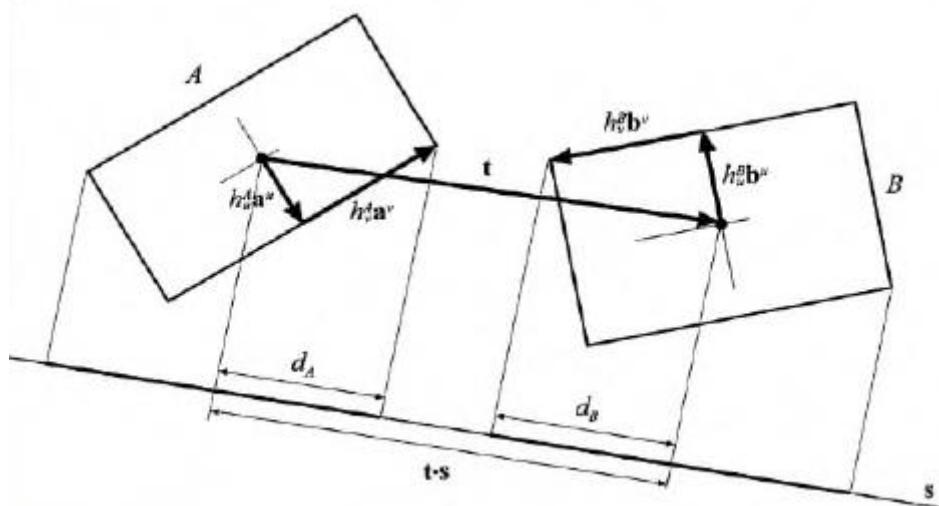
In questo esempio, l'asse  $n$  è l'unico asse che separa le proiezioni.

Come conseguenza dell'ortonormalità della matrice  $\mathbf{A} = (\mathbf{a}^u \ \mathbf{a}^v \ \mathbf{a}^w)$ , i potenziali assi separatori che dovrebbero essere ortogonali alle facce di  $A$ , sono semplicemente gli assi  $\mathbf{a}^u, \mathbf{a}^v, \mathbf{a}^w$ .

Lo stesso vale per  $B$ .

I rimanenti 9 assi potenziali, formati da un lato ciascuno da entrambi  $A$  e  $B$ , sono quindi  $\mathbf{c}^{ij} = \mathbf{a}^i \times \mathbf{b}^j$ ,  $\forall i, j \in \{u, v, w\}$ .

Assumiamo sia  $s$  un potenziale asse di separazione e adottiamo la notazione della seguente figura.



I *radii*  $d_A$  e  $d_B$  degli OBB sull'asse  $s$  sono ottenuti da semplici proiezioni:

$$d_A = \sum_{i \in \{u, v, w\}} h_i^A |a^i \cdot s|$$

$$d_B = \sum_{i \in \{u, v, w\}} h_i^B |b^i \cdot s|$$

Ricorda che  $h_i^A$  e  $h_i^B$  sono sempre positivi e quindi non c'è bisogno di calcolare i loro valori assoluti.

Se e solo se  $s$  è un asse di separazione, allora gli intervalli sugli assi sono disgiunti, cioè:

$$|t \cdot s| > d_A + d_B$$

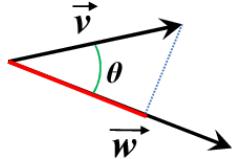
Riassumendo, per ciascun asse si calcola la proiezione su di esso di  $A$  e  $B$ : se le proiezioni sono disgiunte allora l'asse trovato è un asse di separazione e gli OBB sono disgiunti.

Se nessun asse determinato può essere usato come asse di separazione allora gli OBB si intersecano.

## Intersezione fra linee

Per valutare l'eventuale intersezione fra due linee abbiamo bisogno dell'**operatore perpendicolare** (**prodotto scalare** · in 2D e **prodotto vettoriale** × in 3D).

### Prodotto scalare (prodotto interno)



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

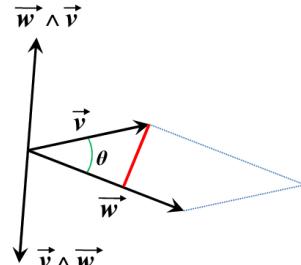
cioè il prodotto dei moduli per il cos dell'angolo tra i vettori ( $0 \leq \theta \leq \pi = 180^\circ$ ) oppure come il prodotto del modulo di un vettore per la proiezione dell'altro sulla direzione del primo.

Il risultato è un valore scalare ed è:

$$\mathbf{v} \cdot \mathbf{w} = v_x w_x + v_y w_y + v_z w_z$$

- Se  $\mathbf{v} \perp \mathbf{w} \rightarrow 0$
- Se  $\mathbf{v} \parallel \mathbf{w} \rightarrow 1$ .

### Prodotto vettoriale (prodotto esterno)



$$|\mathbf{v} \times \mathbf{w}| = |\mathbf{v}| |\mathbf{w}| \sin \theta$$

$\mathbf{v} \times \mathbf{w}$  è un vettore che ha:

- modulo  $|\mathbf{v} \times \mathbf{w}|$  che indica l'area del parallelogramma di lati  $\mathbf{v}$  e  $\mathbf{w}$
- direzione perpendicolare al piano di  $\mathbf{v}$  e  $\mathbf{w}$
- verso si determina con la regola della mano destra: se il pollice è nella direzione di  $\mathbf{v}$  e l'indice in quella di  $\mathbf{w}$ , allora il medio definisce il verso.

## 2D

Consideriamo due linee:

$$\begin{aligned}\mathbf{r}_1(s) &= \mathbf{o}_1 + s\mathbf{d}_1 \\ \mathbf{r}_2(t) &= \mathbf{o}_2 + t\mathbf{d}_2\end{aligned}$$

dove  $\mathbf{o}$  e  $\mathbf{d}$  sono vettori a 2 dimensioni.

Dato che  $a \cdot a^\perp = 0$ , l'intersezione tra  $\mathbf{r}_1(s)$  e  $\mathbf{r}_2(t)$  si calcola tramite i seguenti passi:

$$1. \quad \mathbf{r}_1(s) = \mathbf{r}_2(t) \quad \text{cioè poniamo l'uguaglianza fra le due linee}$$

$$2. \quad \mathbf{o}_1 + s\mathbf{d}_1 = \mathbf{o}_2 + t\mathbf{d}_2 \quad \text{espandendo il passo 1}$$

$$3. \quad \begin{cases} s\mathbf{d}_1 \cdot \mathbf{d}_2^\perp = (\mathbf{o}_2 - \mathbf{o}_1) \cdot \mathbf{d}_2^\perp \\ t\mathbf{d}_2 \cdot \mathbf{d}_1^\perp = (\mathbf{o}_1 - \mathbf{o}_2) \cdot \mathbf{d}_1^\perp \end{cases}$$

$$4. \quad \begin{cases} s = \frac{(\mathbf{o}_2 - \mathbf{o}_1) \cdot \mathbf{d}_2^\perp}{\mathbf{d}_1 \cdot \mathbf{d}_2^\perp} \\ t = \frac{(\mathbf{o}_1 - \mathbf{o}_2) \cdot \mathbf{d}_1^\perp}{\mathbf{d}_2 \cdot \mathbf{d}_1^\perp} \end{cases}$$

Se  $\mathbf{d}_1 \cdot \mathbf{d}_2^\perp = 0$  allora le linee sono parallele e non c'è intersezione.

Se le linee hanno lunghezza infinita, allora tutti i valori di  $s$  e  $t$  sono validi ma per segmenti di linee di lunghezza  $l_1$  ed  $l_2$  (che cominciano a  $s = 0$  e  $t = 0$  e finiscono a  $s = l_1$  e  $t = l_2$ ), abbiamo una intersezione valida sse  $0 \leq s \leq l_1$  e  $0 \leq t \leq l_2$ .

Ovvero se le rette sono tracciate solo in un intervallo di  $s$  e  $t$ , allora c'è intersezione solo se entrambi i parametri sono contenuti nei propri intervalli.

Oppure, se setti  $\mathbf{o}_1 = \mathbf{p}_1$  e  $\mathbf{d}_1 = \mathbf{p}_2 - \mathbf{p}_1$  (ovvero il segmento comincia a  $\mathbf{p}_1$  e termina a  $\mathbf{p}_2$ ) e facciamo la stessa cosa con  $\mathbf{r}_2$  che comincia a  $\mathbf{q}_1$  e termina a  $\mathbf{q}_2$ , allora c'è una valida intersezione sse  $0 \leq s \leq 1$  e  $0 \leq t \leq 1$ .

## 3D

---

Consideriamo nuovamente le due rette con equazione:

$$\begin{aligned}\mathbf{r}_1(s) &= \mathbf{o}_1 + s\mathbf{d}_1 \\ \mathbf{r}_2(t) &= \mathbf{o}_2 + t\mathbf{d}_2\end{aligned}$$

dove  $\mathbf{o}$  e  $\mathbf{d}$  sono vettori a 3 dimensioni.

In questo caso l'operatore *perp* è il prodotto vettoriale  $\times$ .

Dato che  $a \cdot a^\perp = 0$ , l'intersezione tra  $\mathbf{r}_1(s)$  e  $\mathbf{r}_2(t)$  si calcola tramite i seguenti passi:

1.  $\mathbf{r}_1(s) = \mathbf{r}_2(t)$  cioè poniamo l'uguaglianza fra le due linee

2.  $\mathbf{o}_1 + s\mathbf{d}_1 = \mathbf{o}_2 + t\mathbf{d}_2$  espandendo il passo 1

3. 
$$\begin{cases} s\mathbf{d}_1 \times \mathbf{d}_2^\perp = (\mathbf{o}_2 - \mathbf{o}_1) \times \mathbf{d}_2^\perp \\ t\mathbf{d}_2 \times \mathbf{d}_1^\perp = (\mathbf{o}_1 - \mathbf{o}_2) \times \mathbf{d}_1^\perp \end{cases}$$

4. 
$$\begin{cases} \{s(\mathbf{d}_1 \times \mathbf{d}_2) \cdot (\mathbf{d}_1 \times \mathbf{d}_2) = ((\mathbf{o}_2 - \mathbf{o}_1) \times \mathbf{d}_2) \cdot (\mathbf{d}_1 \times \mathbf{d}_2) \\ t(\mathbf{d}_2 \times \mathbf{d}_1) \cdot (\mathbf{d}_2 \times \mathbf{d}_1) = ((\mathbf{o}_1 - \mathbf{o}_2) \times \mathbf{d}_1) \cdot (\mathbf{d}_2 \times \mathbf{d}_1) \end{cases}$$

5. 
$$\begin{cases} s = \frac{\det(\mathbf{o}_2 - \mathbf{o}_1, \mathbf{d}_2, \mathbf{d}_1 \times \mathbf{d}_2)}{\|\mathbf{d}_1 \times \mathbf{d}_2\|^2} \\ t = \frac{\det(\mathbf{o}_2 - \mathbf{o}_1, \mathbf{d}_1, \mathbf{d}_2 \times \mathbf{d}_1)}{\|\mathbf{d}_1 \times \mathbf{d}_2\|^2} \end{cases}$$

Il passo 3 deriva dalla sottrazione di  $\mathbf{o}_1$  ( $\mathbf{o}_2$ ) da entrambi i lati e dal prodotto vettoriale con  $\mathbf{d}_1$  ( $\mathbf{d}_2$ ).

Il passo 4 si ottiene facendo il prodotto scalare con  $\mathbf{d}_1 \times \mathbf{d}_2$  ( $\mathbf{d}_2 \times \mathbf{d}_1$ ).

Il passo 5 (la soluzione) si trova riscrivendo il lato destro come determinanti (e cambiando alcune segni nell'equazione sotto) e poi dividendo per il termine poso a destra di  $s$  ( $t$ ).

Se il denominatore  $\|\mathbf{d}_1 \times \mathbf{d}_2\|^2 = 0$  allora le linee sono parallele.

Se le linee non sono posate sullo stesso piano, allora  $s$  e  $t$  rappresentano i punti sulle rispettive rette che si trovano più vicini l'uno all'altro.

Come nel caso bidimensionale, se le linee sono trattate come segmenti di lunghezza finita e quindi i rispettivi parametri  $s$  e  $t$  assumono valori in intervalli limitati, è necessario che i valori trovati siano compresi nei relativi intervalli, altrimenti non c'è intersezione.

---

## Dynamic intersection testing

---

Fino ad ora è stato considerato solo il test di intersezione statica, ciò significa che tutti gli oggetti coinvolti non si muovono durante il test.

Tuttavia, questo non è sempre uno scenario realistico, specialmente in quanto rendiamo i fotogrammi a tempi discreti.

Supponiamo di avere una palla che si trova al tempo  $t$  da un lato di una porta chiusa e che al tempo  $t + \Delta t$  può muoversi sull'altro lato della porta.

Per dare significato al movimento è necessario rilevare la collisione.

Per fare ciò si possono effettuare molti test frazionando l'intervallo di tempo  $\Delta t$ , ma questo può produrre un numero elevato di calcoli, senza la garanzia di trovare la collisione.

Un test dinamico di intersezione è stato progettato per far fronte a questo problema.

Questa sezione fornisce un'introduzione all'argomento.

Supponiamo che un oggetto  $A$  si muove con velocità  $\mathbf{v}_A$  ed uno oggetto  $B$  con velocità  $\mathbf{v}_B$  dove la velocità è la quantità di un oggetto che si è spostato durante il frame.

Per semplificare i calcoli, supponiamo invece che  $A$  sia in movimento e che  $B$  sia fermo.

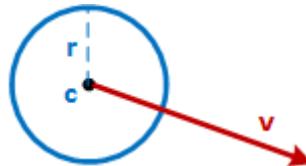
Per compensare la velocità di  $B$ , la velocità di  $A$  è allora:  $\mathbf{v} = \mathbf{v}_A - \mathbf{v}_B$ .

Come tale, negli algoritmi viene data velocità ad un solo oggetto.

Le intersezioni calcolate finora ci sono utili quando dobbiamo calcolare le **collisioni**, cioè intersezioni che avvengono fra gli oggetti della scena a causa del loro movimento in una certa direzione.

## Sfera - piano

---



Abbiamo una sfera centrata in  $c$  e di raggio  $r$  che si muove ad una velocità  $\mathbf{v}$  in un intervallo di tempo  $\Delta t$ .

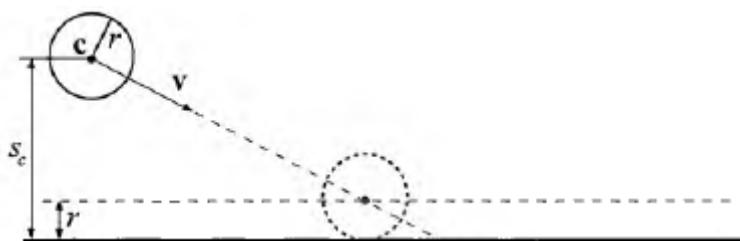
La sfera nel prossimo frame risulta quindi centrata in  $e = c + \Delta t \mathbf{v}$ .

Per semplicità assumiamo che  $\Delta t = 1$  e che il tempo di inizio del movimento sia  $t = 0$ .

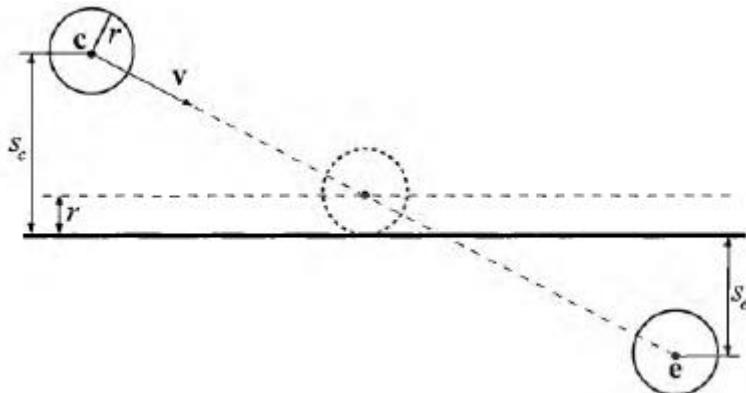
La sfera collide con il piano  $\pi: nx + d = 0$ ?

Calcoliamo per utilità futura i seguenti valori:

- $s_c$ : la distanza (con segno) fra il centro della sfera è il piano.  
È calcolata sostituendo il centro della sfera nell'equazione del piano e sottraendo il raggio della sfera a questo valore.  
Indica quanto si può muovere la sfera (perpendicolarmente rispetto al piano) prima di incontrare il piano.



- $s_e$ : simile ad  $s_c$  ma calcolato a partire dalla posizione finale  $e$ .



Se  $s_c s_e > 0$  cioè la sfera iniziale e quella finale si trovano dalla stessa parte del piano e se  $|s_c| > r$  e  $|s_e| > r$ , non c'è intersezione (sono dalla stessa parte e troppo lontano dal piano perché un punto sulla superficie possa collidere).

In questo caso la sfera può essere spostata in  $e$  senza problemi.

Altrimenti, la posizione della sfera e il tempo esatto in cui avviene la prima collisione, sono calcolati tramite il parametro  $t$ :

$$t = \frac{s_c - r}{s_c - s_e}$$

Il centro della sfera è in  $c + t\mathbf{v}$ .

Per dare senso alla collisione si può pensare di riflettere il vettore della velocità  $\mathbf{v}$  lungo la normale al piano impattato, e spostare la sfera lungo questa nuova direzione:

$$(1-t)\mathbf{r}$$

dove  $1-t$  è il tempo rimanente per il fotogramma successivo dalla collisione, ed  $\mathbf{r}$  è il vettore di riflessione.

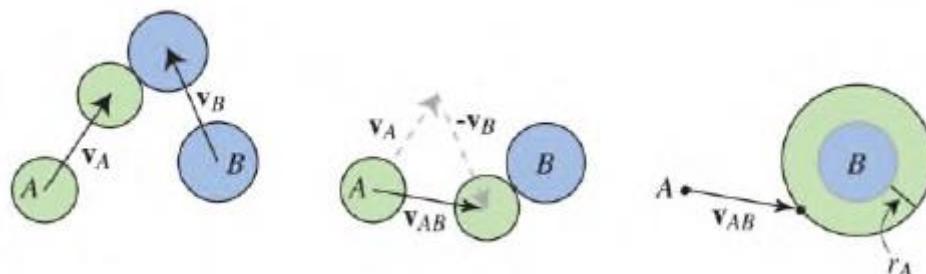
### Sfera - sfera

Consideriamo la collisione fra due sfere in movimento: con alcune accortezze possiamo ridurre la collisione fra le due sfere ( $A$  e  $B$ ) al test di intersezione fra un raggio e una sfera ferma: ciò può essere fatto sottraendo la velocità  $A$  sia ad  $A$  che a  $B$ .

In questo modo  $A$  risulta ferma, mentre  $B$  ha una nuova velocità data dalla differenza fra  $\mathbf{v}_B$  e  $\mathbf{v}_A$ .

Notiamo che questo calcolo non modifica la posizione della collisione fra le due sfere.

Il passo successivo è aggiungere il raggio di  $B$  ad  $A$ , in modo da trasformare  $B$  in un semplice raggio e  $A$  in una sfera di raggio  $r_A + r_B$ .



La figura a sinistra mostra due sfere che si muovono e collidono.

Al centro la sfera  $B$  è stata fatta statica sottraendo la sua velocità da entrambe le sfere.

Nota che le posizioni relative delle sfere al punto di collisione rimangono le stesse.

A destra, il raggio  $r_A$  della sfera  $A$  viene aggiunto al raggio di  $B$  e sottratto da se stesso: la sfera in movimento  $A$  può essere considerata come un raggio.

Ora abbiamo la situazione in cui la sfera  $B$  è statica ed è più grande, e la sfera  $A$  è un punto che si muove lungo una linea retta, cioè un raggio.

Dato ciò, possiamo calcolare il tempo  $t$  di intersezione come:

$$(\mathbf{v}_{AB} \cdot \mathbf{v}_{AB})t^2 + 2(\mathbf{l} \cdot \mathbf{v}_{AB})t + \mathbf{l} \cdot \mathbf{l} - (r_A + r_B)^2 = 0$$

dove  $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$  ed  $\mathbf{l} = \mathbf{c}_A - \mathbf{c}_B$  dove  $\mathbf{c}_A$  e  $\mathbf{c}_B$  sono i centri delle sfere.

Possiamo scrivere:

$$a = (\mathbf{v}_{AB} \cdot \mathbf{v}_{AB})$$

$$b = 2(\mathbf{l} \cdot \mathbf{v}_{AB})$$

$$c = \mathbf{l} \cdot \mathbf{l} - (r_A + r_B)^2$$

che sono valori che uso nell'equazione quadratica  $at^2 + bt + c = 0$ .

Usiamo la soluzione Press:

$$q = -\frac{1}{2} \left( b + sign(b) \sqrt{b^2 - 4ac} \right)$$

$$\text{dove } sign(b) = \begin{cases} 1 & b \geq 0 \\ -1 & \text{else} \end{cases}.$$

Quindi:

$$t_0 = \frac{q}{a} \quad t_1 = \frac{c}{q}$$

Il più piccolo valore nel range  $[t_0, t_1]$  che si trova in  $[0, 1]$  (il tempo del frame) è il tempo della prima intersezione.

Ricavando  $t$  e sostituendolo nelle equazioni di movimento delle sfere otteniamo i punti:

$$\mathbf{p}_A(t) = \mathbf{c}_A + t \cdot \mathbf{v}_A$$

$$\mathbf{p}_B(t) = \mathbf{c}_B + t \cdot \mathbf{v}_B$$

che indicano le posizioni delle due sfere al momento della collisione.

# 5. Gestione della fisica

## Introduzione

Come detto in precedenza una delle fasi di rendering più importanti è legato al movimento degli oggetti e alla loro interazione, cioè la gestione della fisica e delle sue leggi nel corso della simulazione.

La quantità di fisica necessaria per la simulazione dipende dalla simulazione scelta (per la gestione di una folla di persone è tanta, per una palla che scorre su un piano inclinato è poca).

Come visto nel capitolo sulla pipeline del rendering, la gestione della fisica avviene tramite una sotto-pipeline e viene effettuata dalla GPU (scheda grafica) e dalla sua sottoarea **PPU (Phisic processing unit)**.

Nel caso in cui non sia la GPU a lavorare sulla fisica, se ne occuperà la CPU nella fase di *application* (solitamente avviene nel *geometry shade* con la GPU).

La PPU, è un microprocessore dedicato per la gestione del calcolo della fisica degli oggetti virtuali, specialmente per quanto riguarda i calcoli di fisica presenti nel motore grafico di un videogioco.

È quindi un'enorme rivoluzione, soprattutto nel campo della grafica tridimensionale, in quanto tutti i movimenti degli oggetti, la dinamica dei fluidi, la simulazione del movimento di capelli e abiti, la distruzione di oggetti e tutto quello che riguarda il tridimensionale non viene più elaborato dalla CPU del computer, ma dal processore della scheda, ottimizzato per quel tipo di calcoli.

Alla CPU viene quindi tolta una determinata mole di lavoro, rendendo disponibile tale potenza risparmiata per l'impiego in altre operazioni, tra cui la gestione del sistema operativo, del motore di gioco, dell'intelligenza artificiale ecc.

La fisica è molto utile per le operazioni di:

- collision detection (cogliere l'istante in cui avviene la collisione e gestire l'urto in modo opportuno)
- per simulare la gravità o la presenza di corpi puntiformi o effetti particellari
- per determinare le caratteristiche di corpi rigidi e non rigidi (quali i tessuti)
- per simulare corpi in giunzione fra loro (ad esempio le articolazioni).

Un **motore fisico (physics engine)** è un programma informatico che simula un modello fisico Newtoniano utilizzando variabili come massa, velocità, frizione alla resistenza del vento e altro. Il motore, utilizzando questi dati e le leggi Newtoniane, simula il comportamento degli oggetti sottoposte alle forze del mondo (reale o immaginario).

È possibile definire almeno due classi di motori fisici:

- **high-precision**: non lavorano in tempo reale, con l'obiettivo di raggiungere il miglior risultato possibile; utile in ambito scientifico e in tutte le applicazioni non real-time.  
L'accuratezza della simulazione è massima.
- **real-time**: richiedono un **framerate** (numero di immagini prodotte al secondo) molto alto, pertanto si limitano a raggiungere risultati soddisfacenti, anche se subottimi.  
La velocità della simulazione è la massima possibile.  
La velocità di elaborazione si scontra però con la complessità della scena, pertanto si applicano (come nelle tecniche di illuminazione) delle prefasi di semplificazione degli oggetti (ad esempio introdurre i bounding box per creare un involucro geometricamente semplice all'oggetto) che permettono un'approssimazione del risultato finale in un tempo molto più breve.

---

## Leggi di Newton (~ 1700)

---

Ricapitoliamo ora alcune delle leggi della fisica che vengono usate in fase di simulazione:

- (1) **prima legge di Newton**: un corpo mantiene il proprio stato di quiete o di moto rettilineo uniforme, finché una forza non agisce su di esso
- (2) **seconda legge di Newton**: se su un corpo agisce una forza o un sistema di forze, la forza risultante applicata al corpo possiede direzione e verso della sua accelerazione e, in modulo, è direttamente proporzionale al modulo la sua accelerazione  $F = m \cdot a$
- (3) **terza legge di Newton**: ad ogni azione corrisponde una reazione uguale e contraria.

Generalmente un oggetto non si ferma naturalmente ma sono le forze che lo portano a fermarsi.

Le **forze** che interagiscono possono essere applicate da un soggetto esterno, dalla gravità, dai campi elettromagnetici e seguono la formula definita dalla seconda legge di Newton:

$$F = m \cdot a \quad [F \rightarrow N, \text{Newton} \quad m \rightarrow kg \quad a \rightarrow m/s^2]$$

Da cui possono calcolare l'**accelerazione**:

$$a = \frac{F}{m}$$

oppure:

$$a(t) = \frac{dv}{dt} = a$$

L'accelerazione così calcolata viene usata per modificare la **velocità** tramite l'equazione:

$$v(t) = v_0 + \int_0^t dv = v_0 + \int_0^t a dt = v_0 + at$$

la quale è usata per modificare la **posizione** dell'oggetto in movimento tramite l'equazione:

$$x = x_0 + (v + at)t$$

applicata per  $x, y$  e  $z$  con  $t$  il delta time.

La **cinematica** descrive il moto dei corpi senza però indagare le cause del moto.

Le forze possono essere applicate su:

- **corpi puntiformi**, la nostra approssimazione iniziale, che ci permette di gestire solo movimenti lineari (ad esempio l'*effetto particellare*)
- **corpi rigidi**, la cui forma non cambia ma di cui è necessario gestire angoli e spostamenti lineari (ad esempio i personaggi di un gioco)
- **corpi morbidi**, che cambiano posizione, orientamento e forma (ad esempio i vestiti o i liquidi).

### Sistemi particellari

---

Un **sistema particellare** è un programma che simula e ricrea le dinamiche fisiche e il comportamento delle particelle.

Queste possono essere di varia natura e genere: fumo, acqua, sabbia, vapore, fuoco, polvere, nuvole ma anche milioni di insetti o decine di migliaia di foglie, ecc.

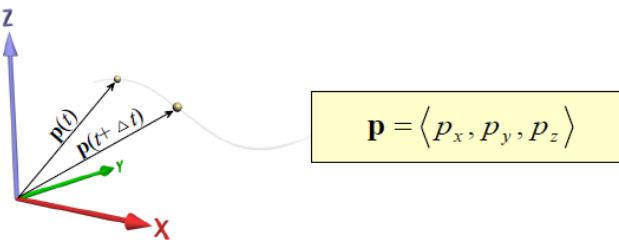
Un sistema particellare calcola tutte le dinamiche che avvengono tra esse, ad esempio eventi atmosferici come vento e attrito, ma anche gravità e collisioni.

Una **particella** è una sfera di raggio piccolo e finito con una superficie perfettamente liscia e priva di attrito.

Una particella permette solo traslazioni nello spazio e non rotazioni.

In cinematica è descritta da:

- **posizione**  $(p_x, p_y, p_z)$  che muta nel tempo



- **velocità** [m/s]

- velocità media:

$$v_m(t) = \frac{\text{spazio percorso}}{\text{tempo trascorso}} = \frac{\mathbf{p}(t + \Delta t) - \mathbf{p}(t)}{\Delta t}$$

- velocità istantanea: derivata prima della posizione in un periodo di tempo

$$v(t) = \lim_{\Delta t \rightarrow 0} \frac{\text{spazio percorso}}{\text{tempo trascorso}} = \frac{\mathbf{p}(t + \Delta t) - \mathbf{p}(t)}{\Delta t} = \frac{d\mathbf{p}(t)}{dt}$$

e quindi la posizione è l'integrale della velocità sul tempo

- **accelerazione** [m/s<sup>2</sup>]

- derivata prima della velocità o derivata seconda della posizione in un periodo di tempo:

$$\mathbf{a}(t) = \frac{d}{dt} v(t) = \frac{d^2}{dt^2} \mathbf{p}(t)$$

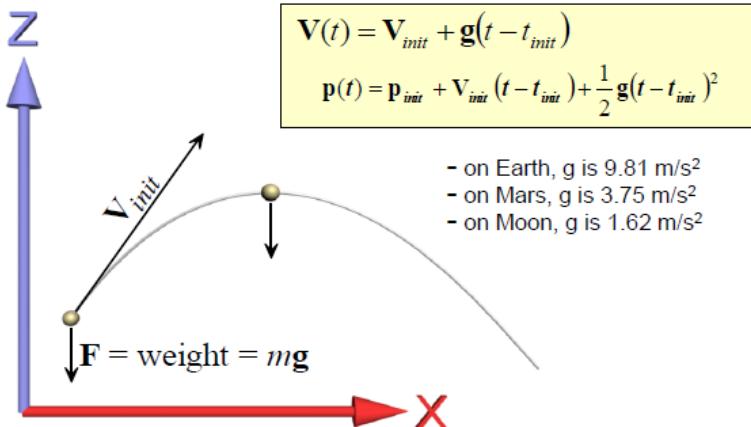
## Simulazione fisica

Quando parliamo di **simulazione fisica** parliamo di un **ciclo di movimento**: l'applicazione di una forza  $\mathbf{F}(t)$  causa un'accelerazione  $\mathbf{a}(t)$  che modifica la velocità  $v(t)$  di un oggetto mutandone la sua posizione.

La simulazione deve risolvere le variazioni che intervengono nel tempo, generando in ogni frame la posizione degli oggetti, in modo da renderizzarli correttamente sullo schermo.

### Esempio 3D di un movimento di un proiettile

Data una velocità iniziale  $\mathbf{v}_{init}$  ed una posizione iniziale  $\mathbf{p}_{init}$  a tempo  $t_{init}$ , voglio conoscere i valori a tempo  $t$ .



## Equazioni in forma chiusa

---

In certi casi, quando non è presente rotazione, le equazioni da gestire in fase di simulazione vengono definite in **forma chiusa**, in quanto qualsiasi riproduzione del ciclo di movimento porta allo stesso risultato, rendendo necessario determinare solo il tempo corrente rispetto al tempo iniziale e nessun altro parametro.

Quando le equazioni non sono in forma chiusa (ad esempio accelerazione e velocità cambiano in base ad alcuni parametri particolari e non sono costanti) è necessario usare l'*approssimazione numerica* per risolvere le equazioni.

## Approssimazione numerica

---

L'**approssimazione numerica** rappresenta una serie di tecniche per risolvere incrementalmente le equazioni del moto quando le forze applicate ad un oggetto non sono costanti o quando non c'è una soluzione in forma chiusa.

Il problema principale nell'applicazione delle leggi fisiche è che richiedono calcoli complessi infattibili nel tempo concesso da un ciclo di rendering.

Per evitare ciò si usano dei metodi numerici, che approssimano i risultati esatti tramite calcoli numerici più semplici e più adatti ad una macchina a precisione finita.

Ad esempio considerando una qualsiasi funzione  $S$  calcolata in un istante  $t$  avremo un certo valore definito da  $S(t)$ .

Posso usare diverse tecniche di approssimazione come:

- (1) Taylor
- (2) Eulero
- (3) Runge-Kutta.

In ogni caso, data una particolare approssimazione numerica, il suo ordine è  $n$  quando il suo termine dell'errore è  $O(\Delta t)^{n+1}$ .

### (1) SERIE DI TAYLOR

Per calcolare il valore all'iterazione successiva  $S(t + \Delta t)$  possiamo usare l'espansione tramite serie di Taylor:

$$S(t + \Delta t) = S(t) + \Delta t \frac{d}{dt} S(t) + \frac{(\Delta t)^2}{2!} \frac{d^2}{dt^2} S(t) + \dots$$

Se tronchiamo rimuovendo i tempini più grandi:

$$S(t + \Delta t) = S(t) + \Delta t \frac{d}{dt} S(t) + \frac{(\Delta t)^2}{2!} \frac{d^2}{dt^2} S(t) + \dots + O(\Delta t)^n$$

dove  $O(\Delta t)^n$  è l'**errore di troncamento**.

### (2) INTEGRAZIONE DI EULERO

Altro metodo è usare l'integrazione di Eulero, che mira ad approssimare la funzione in  $S(t + \Delta t)$  usando il valore esatto in  $t$  e la pendenza della funzione (la tangente alla funzione) nel punto  $t$ , troncando quindi la serie di Taylor dopo il secondo addendo.

Purtroppo questo metodo provoca una crescita dell'errore allontanandosi dall'istante iniziale.

## ESEMPIO DI SIMULAZIONE NUMERICA

Consideriamo una particella che sta accelerando di  $10 \text{ m/s}^2$  da un certo punto iniziale: in 10 secondi deve avere esattamente velocità 100 m/s.

L'equazione in forma chiusa che descrive la distanza coperta dopo 10 s è:

$$s = 0.5 \cdot at^2 = 0.5 \cdot (10) \cdot (10)^2 = 0.5 \cdot (10) \cdot (100) = 500 \text{ m}$$

Lo pseudocodice che integra le equazioni del moto rispetto al tempo, semplicemente aggiungendo il cambiamento di posizione e la velocità ai valori precedenti ad ogni step  $t$  (integrazione di Eulero) è:

```
float t = 0;
float dt = 1;
float velocity = 0;
float position = 0;
float force = 10;
float mass = 1;

while(t <= 10) {
    position = position + velocity*dt;
    velocity = velocity + (force/mass)*dt;
    t = t + dt;
}
```

Quindi se usiamo l'integrazione di Eulero invece che risolvere esattamente come abbiamo fatto prima, otteniamo:

```
t = 0: position = 0      velocity = 0
t = 1: position = 0      velocity = 10
t = 2: position = 10     velocity = 20
t = 3: position = 30     velocity = 30
t = 4: position = 60     velocity = 40
t = 5: position = 100    velocity = 50
t = 6: position = 150    velocity = 60
t = 7: position = 210    velocity = 70
t = 8: position = 280    velocity = 80
t = 9: position = 360    velocity = 90
t = 10: position = 450   velocity = 100
```

Quindi quando usiamo l'integrazione di Eulero, abbiamo un'accuratezza di 50 metri per questo semplice calcolo dopo solo 10 secondi, inoltre questo errore continua a crescere nel tempo.

Una soluzione a questo problema è ridurre il time step.

Ad esempio, possiamo ridurre il time step a  $dt = 1/100^{\text{th}}$  di secondi ad ogni integrazione di Eulero anziché un secondo.

Facendo così otterremo un risultato molto più vicino al valore esatto, tuttavia, l'errore continuerà a crescere nel tempo.

Considerando che questa è una simulazione estremamente semplice, le cose si complicano e l'errore cresce nel caso di simulazioni più complesse.

## (3) INTEGRAZIONE DI RUNGE-KUTTA (RK2, RK4)

Alternativamente al metodo di Eulero è possibile usare quello di Runge-Kutta (quello implementato nelle GPU) che diminuisce notevolmente l'errore al crescere di  $t$ .

Anche con questo metodo si usano solo le derivate prime ma se ne combinano un insieme (solitamente quelle nel punto  $t$ , nel punto  $t_0$  iniziale e nel punto  $t_n$  finale) per generare dei valori che mediati correttamente ci diano una pendenza più accurata del tratto di funzione fra  $t$  e  $\Delta t$ .

Successivamente si usa l'integrazione di Eulero usando la pendenza (=derivata prima) dataci dal Runge-Kutta per ottenere un risultato più accurato.

**RK2:** valuta le derivate al timestamp iniziale e finale e poi combina queste derivate campionate facendone una media:

$$\begin{aligned}y' &= f(y(t), t) \\y(t_0) &= y_0 \\K_1 &= f(y_n, t_n) \\K_2 &= f(y_n + K_1 \Delta t, t_n + \Delta t) \\y_{m+1} &= y_n + \frac{\Delta t}{2}(K_1 + K_2)\end{aligned}$$

**RK4:** L'integrazione di RK più usata è la RK4, in cui si definiscono quattro valori di  $k$ :

- $k_1$  è la pendenza all'inizio dell'intervallo
- $k_2$  la pendenza a metà dell'intervallo (usando  $k_1$  per stimare il punto medio)
- $k_3$  la pendenza a metà dell'intervallo (usando  $k_2$  per stimare il punto medio)
- $k_4$  la pendenza alla fine dell'intervallo (usando  $k_3$  per stimare il punto finale).

I 4  $k$  vengono pesati in modo diverso per determinare la pendenza finale da usare, dando maggior peso ai due  $k$  centrali e meno a quelli all'estremo (diamo peso 1, 2, 2, 1).

RK4 è un metodo di ordine 4 e quindi il suo errore è dell'ordine  $\Delta t^5$ .

### Risposta alla collisione senza attrito

Un **momento lineare** (o **quantità di moto**) di un oggetto massivo è una grandezza vettoriale definita come il prodotto della massa dell'oggetto per la sua velocità:

$$\text{momento} \quad \mathbf{p} = m \cdot \mathbf{v} \quad [kg \cdot m/s]$$

Si tratta di una grandezza fisica conservativa, ovvero che rimane uguale nel tempo in assenza di forze esterne al sistema applicate all'oggetto.

La derivata prima del momento è la forza applicata all'oggetto:

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}(t)$$

L'importanza della quantità di moto è espressa dal secondo principio della dinamica, dal quale si evince che la forza applicata ad un punto materiale è pari alla derivata della quantità di moto del punto stesso rispetto al tempo.

Infatti, supponendo la massa costante:

$$\mathbf{F}(t) = \frac{d\mathbf{p}}{dt} = m \frac{d\mathbf{v}(t)}{dt} = m \cdot \mathbf{a}(t)$$

Considera due particelle che collidono.

Per la durata della collisione (solitamente molto breve), entrambe le particelle si applicano una forza a vicenda.

### Impulso

Viene definito **impulso** la variazione della quantità di moto di un corpo che viene sottoposto ad un urto con un altro corpo.

In altre parole è l'effettiva quantità di moto trasmessa al corpo urtato al momento dell'urto.

Le quantità di moto iniziale e finale utili per calcolare l'impulso consistono nel prodotto della massa del corpo per la velocità finale (nel primo membro) e per la velocità iniziale (nel secondo membro).

Dunque per calcolare l'impulso in genere si usa misurare massa e velocità del corpo prima del contatto e trarre i dati iniziali e ripetere l'operazione dopo il contatto.

Sfruttando la seconda legge della dinamica di Newton e la legge della cinematica di un moto rettilineo uniforme si ha che:

$$\mathbf{F}(t) = \frac{d\mathbf{p}}{dt}$$

Integrando rispetto al tempo entrambi i membri si ottiene:

$$\Delta\mathbf{p} = \int_0^{\Delta t} \mathbf{F} dt$$

$$m_1\mathbf{v}_1^+ = m_1\mathbf{v}_1^- + \Lambda \quad \text{eq. 1}$$

dove  $m_1\mathbf{v}_1^+$  è il momento lineare della prima particella subito prima della collisione ed  $m_1\mathbf{v}_1^-$  è il momento lineare della prima particella subito dopo la collisione.

$\Lambda$  è l'**impulso lineare**.

Il terzo principio della dinamica di Newton dice che ad ogni azione corrisponde una reazione uguale ed opposta quindi la particella 2 ha stessa magnitudo ma direzione opposta ( $-\Lambda$ ):

$$m_2\mathbf{v}_2^+ = m_2\mathbf{v}_2^- - \Lambda \quad \text{eq. 2}$$

Possiamo risolvere queste equazioni se conosciamo  $\Lambda$ .

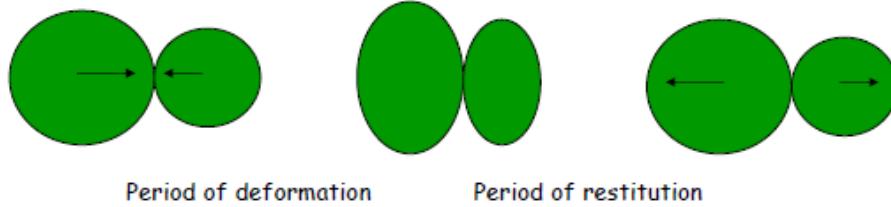
In assenza di attrito, l'impulso lineare agisce completamente lungo la superficie unitaria normale al vettore nel punti di contatto:

$$\Lambda = \Lambda_s \mathbf{n}$$

dove  $\mathbf{n}$  è la superficie unitaria normale al vettore e  $\Lambda_s$  è il valore scalare dell'impulso.

Quindi abbiamo due equazioni con 3 incognite ( $\mathbf{v}_1^+, \mathbf{v}_2^+, \Lambda_s$ ): abbiamo bisogno di una terza equazione.

Osserva il comportamento nella realtà di due oggetti quando collidono.



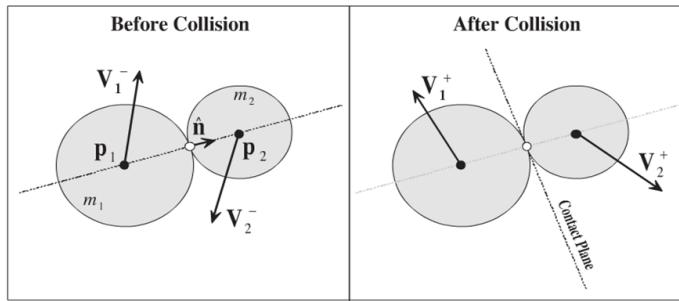
Durante l'impatto iniziale, entrambi gli oggetti sperimentano un periodo di deformazione in cui le loro forme si comprimono e si deformano in risposta alla forza di collisione.

La terza equazione è l'approssimazione della risposta del materiale agli oggetti che collidono:

$$(\mathbf{v}_1^+ - \mathbf{v}_2^+) \mathbf{n} = -\varepsilon (\mathbf{v}_1^- - \mathbf{v}_2^-) \mathbf{n} \quad \text{eq. 3}$$

dove  $\varepsilon$  è il **coefficiente di restituzione**, relativo alla conservazione o alla perdita di energia cinetica.

- Se  $\varepsilon = 1$ , la collisione è perfettamente elastica cioè gli oggetti rimbalzano e la somma delle energie cinetiche delle particelle è la stessa prima e dopo la collisione.
- Se  $\varepsilon = 0$ , la collisione è perfettamente plastica cioè gli oggetti non hanno alcun periodo di restituzione e hanno una perdita massima di energia cinetica.
- Nella vita reale, dipende dai materiali (es. una palla da tennis sulla racchetta ha  $\varepsilon = 0.85$ ).



## Coefficiente di restituzione

È una misura dell'elasticità della collisione: misura quanta parte dell'energia cinetica prima della collisione degli oggetti rimane come energia cinetica anche dopo la collisione.

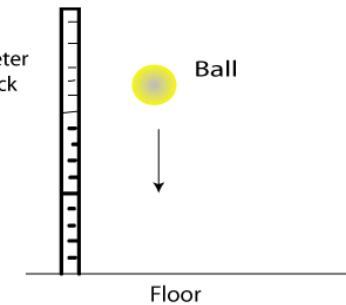
È definito come rapporto delle differenze delle velocità prima e dopo la collisione:

$$\varepsilon = \frac{v_1^+ - v_2^+}{v_1^- - v_2^-}$$

Per un oggetto che colpisce un altro oggetto inamovibile (es. il pavimento):

$$\varepsilon = \sqrt{\frac{h}{H}}$$

dove  $h$  è l'altezza del rimbalzo e  $H$  l'altezza della caduta.



object	H (cm)	$h_1$ (cm)	$h_2$ (cm)	$h_3$ (cm)	$h_4$ (cm)	$h_5$ (cm)	$h_{ave}$ (cm)	c.o.r.
range golf ball	92	67	66	68	68	70	67.8	0.858
tennis ball	92	47	46	45	48	47	46.6	0.712
billiard ball	92	60	55	61	59	62	59.4	0.804
hand ball	92	51	51	52	53	53	52.0	0.752
wooden ball	92	31	38	36	32	30	33.4	0.603
steel ball bearing	92	32	33	34	32	33	32.8	0.597
glass marble	92	37	40	43	39	40	39.8	0.658
ball of rubber bands	92	62	63	64	62	64	63.0	0.828
hollow, hard plastic ball	92	47	44	43	42	42	43.6	0.688

Esperimento fatto con diverse palle:

- lascia cadere una palla da una altezza predefinita (92 cm)
- misura il rimbalzo
- ripeti 5 volte e calcolane la media.

## Calcolare la velocità dopo la collisione

Risolvendo le 3 equazioni precedenti, otteniamo:

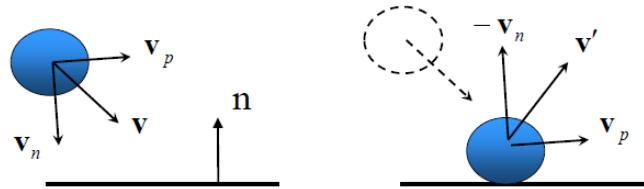
$$\begin{cases} m_1 v_1^+ = m_1 v_1^- + \Lambda \\ m_2 v_2^+ = m_2 v_2^- - \Lambda \\ (v_1^+ - v_2^+) \mathbf{n} = -\varepsilon (v_1^- - v_2^-) \mathbf{n} \end{cases} \rightarrow \Lambda = -\left(\frac{m_1 m_2 (1 + \varepsilon) (v_1^- - v_2^-) \hat{\mathbf{n}}}{m_1 + m_2}\right) \hat{\mathbf{n}} \quad Eq. 4$$

Per calcolare le velocità dopo la collisione, applico il risultato dell'eq. 4 alle eq. 1 e 2 e divido per  $m_1$  e  $m_2$  rispettivamente.

È consigliabile considerare un sistema di coordinate che include  $\mathbf{n}$  e il piano di contatto tangente alle superfici dell'oggetto nel punto di contatto.

Per una collisione senza attrito, le velocità delle componenti parallele al piano di contatto non cambiano. Deve essere calcolata solo la componente parallela alla direzione normale di contatto.

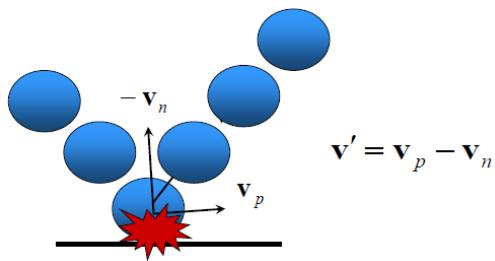
Il metodo può essere generalizzato per 3 dimensioni.



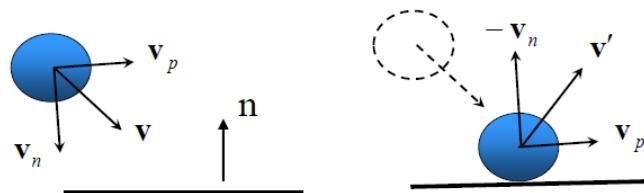
$$\mathbf{v} = \mathbf{v}_n + \mathbf{v}_p$$

$$\mathbf{v}_n = (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$$

$$\mathbf{v}_p = \mathbf{v} - \mathbf{v}_n$$



$$\mathbf{v}' = \mathbf{v}_p - \mathbf{v}_n$$

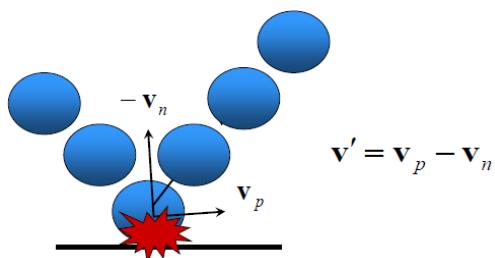


$$\mathbf{v} = \mathbf{v}_n + \mathbf{v}_p$$

$$\mathbf{v}_n = (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$$

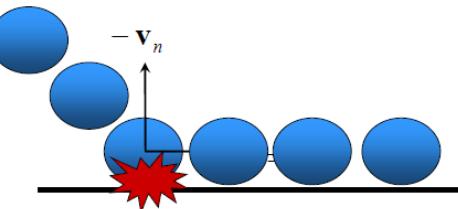
$$\mathbf{v}_p = \mathbf{v} - \mathbf{v}_n$$

$$\boxed{\mathbf{v}' = \mathbf{v}_p - k\mathbf{v}_n}$$



$$\mathbf{v}' = \mathbf{v}_p - \mathbf{v}_n$$

$$\mathbf{v}' = \mathbf{v}_p - 0.0\mathbf{v}_n$$





## 6. Pipeline di rendering

Il **VR engine** (*motore VR*) è un componente chiave del sistema VR che legge i dispositivi di input, accede a database task-dependent, aggiorna lo stato del mondo virtuale ed invia i risultati ai display di output.

È un'astrazione: può significare un computer, più computer collocati insieme o molti computer remoti che collaborano in una simulazione in modo distribuito.

La peculiarità del real-time del VR richiede un motore VR molto potente in modo da assicurare:

- velocità di aggiornamento della grafica e aptica (30 fps e centinaia di Hz)
- bassa latenza (< 100 ms per evitare il malore degli utenti durante di simulazione).

Il nucleo di tale architettura è la **pipeline di rendering**.

I compiti che devono essere svolti sono (in ordine sparso):

- proiettare tutti gli oggetti 3D sul piano immagine (*trasformazioni geometriche*)
- determinare quali primitive o parti di primitive sono visibili (*rimozione delle superfici nascoste*)
- determinare quali pixel occupa una determinata primitiva (*scan conversion*)
- calcolare il colore di ogni punto di una superficie visibile (*illuminazione, shading, texture mapping*).

---

### Pipeline

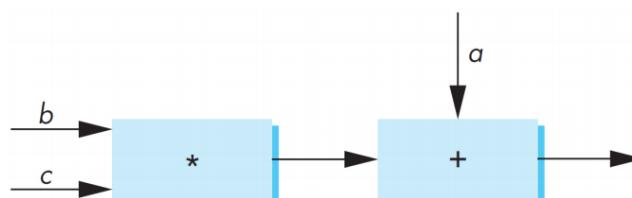
---

La **pipeline** è una sequenza di operazioni o blocchi funzionali (anche in numero molto vasto) che devono essere eseguiti in ordine.

Caratteristica e vantaggio della pipeline è che può essere applicata ad un numero molto vasto di oggetti simili fra loro che sono sottoposti alla "catena di montaggio".

Ciascun modulo della catena di montaggio lavora in modo indipendente dagli altri, lavorando su oggetti che sono forniti in output dai moduli precedenti della pipeline.

La pipeline ci permette di **parallelizzare** le operazioni, in quanto i moduli possono lavorare su parti differenti di dati, senza vincoli (o con pochi vincoli) legati agli altri moduli.



Nell'esempio in figura supponiamo di avere una ALU<sup>2</sup> che computa un'operazione alla volta.

L'espressione che vogliamo calcolare è  $a + (b \cdot c)$ .

Se usiamo l'ALU necessitiamo di 2 cicli di clock (moltiplicazione e somma).

Invece nel caso in cui usiamo una pipeline, supponendo di somministrare molte triple  $(a, b, c)$  da calcolare, raddoppiamo la velocità di calcolo del sistema: mentre il secondo modulo della pipeline calcola la somma, il primo può portarsi avanti e calcolare la moltiplicazione della tripletta successiva.

---

<sup>2</sup> L'unità aritmetico-logica (**ALU**) è una tipologia particolare di processore digitale che si contraddistingue per essere preposta all'esecuzione di operazioni aritmetiche o logiche.

L'ALU è una componente fondamentale della CPU e della GPU ed è presente nell'architettura di Von Neumann. Attualmente l'ALU è tipicamente implementata come parte di un microprocessore monolitico. Le moderne CPU e GPU sono dotate di ALU molto sofisticate e un singolo microprocessore può contenere anche più di un'ALU.

Se aggiungiamo altri moduli aumentiamo ancora la capacità del nostro sistema (**throughput**<sup>3</sup>) ma aumentiamo anche la latenza (cioè il tempo) necessaria ad elaborare il singolo elemento (che deve percorrere l'intera pipeline).

Il vantaggio si ha quindi quando si cerca di elaborare un gran numero di elementi dello stesso tipo ma diversi fra loro (ad esempio i pixel dell'immagine).

Idealmente, un sistema senza pipeline che viene suddiviso in  $n$  passi di pipeline può dare un accelerazione di un fattore di  $n$ .

Questo aumento delle prestazioni è la ragione principale per utilizzare la pipelining.

Le fasi della pipeline vengono eseguite in parallelo ma vengono bloccate finché la fase più lenta ha finito il suo compito (*collo di bottiglia*), infatti il throughput dei dati è determinato dal processo più lento.

Vantaggi della pipeline:

- ▲ la *modularità* che permette di:
  - fare elaborazione parallele (aumenta il parallelismo)
  - fare una separazione logica dei diversi componenti
- ▲ è necessaria solo una *conoscenza locale* sul singolo elemento o sul singolo gruppo di elementi (es. pochi pixel dell'immagine).

Svantaggi della pipeline:

- ▼ flessibilità limitata
- ▼ alcuni algoritmi potrebbero richiedere un diverso ordinamento dei blocchi della pipeline
- ▼ la conoscenza locale non permette di gestire alcuni processi quali quelli legati all'ombreggiatura e all'illuminazione globale<sup>4</sup> in quanto richiedono una conoscenza dell'intera scena.

---

## Cicli di rendering e frame rate

---

La **pipeline del rendering** è un modello astratto che ci permette di modellare il **ciclo di rendering** del motore grafico.

Per dare l'impressione della fluidità delle immagini presentate, queste devono essere presentate in rapida successione perciò tale ciclo deve essere ripetuto un numero molto alto di volte al secondo.

Il tasso con cui vengono renderizzate le immagini prende il nome di **framerate** (cioè il numero di immagini che vengono presentate in ogni secondo) e deve essere di almeno 30fps (frame per secondo): tale numero è dovuto alle caratteristiche del sistema visivo umano, in cui la persistenza dell'immagine sulla retina è di qualche decina di millisecondi.

Se il tempo di persistenza del frame è minore del tempo di persistenza dell'immagine sulla retina si ha la sensazione della fluidità.

Il framerate può essere più alto in caso di **interfaccia aptica**<sup>5</sup>.

---

<sup>3</sup> Il **throughput** non è da confondersi con la "capacità" del link: sia la capacità che il throughput si esprimono in bit/s, ma mentre la prima esprime la frequenza trasmisiva massima alla quale i dati possono viaggiare, il throughput è un indice dell'"effettivo" utilizzo della capacità del link. Il throughput è la quantità di dati trasmessi in una unità di tempo, mentre la capacità dipende esclusivamente da quanta informazione è disponibile sul canale nella trasmissione.

<sup>4</sup> Un **modello di illuminazione locale** determina come interagiscono le sorgenti luminose (e i loro raggi) e le superfici della scena. I modelli globali usano i modelli locali estendendone le capacità, definendo anche le mutue interazioni luminose indirette fra i diversi oggetti della scena (es. raggio riflesso da una superficie su un'altra). Il dettaglio è rimandato a sezioni successive.

<sup>5</sup> Un'**interfaccia aptica** è un'interfaccia di input che è anche interfaccia di output, ad esempio un braccio meccanico con un certo numero di gradi di libertà, il quale reagisce, in questo caso con una forza che modella l'ambiente fisico in cui si muove il braccio nell'ambiente virtuale.

Per aumentare la fluidità percepita dall'utente, il sistema deve avere una bassa latenza, cioè un basso tempo di reazione/risposta del sistema all'input dell'utente (< 100 ms).

In ogni ciclo di rendering è necessario accedere agli input trasmessi dell'utente (in modo banale con tastiera e mouse) e di conseguenza adattare l'ambiente virtuale (aggiornare il sistema virtuale e renderizzarlo) in modo coerente con gli input, per dare all'utente un feedback su ciò che è stato fatto.

A livello più basso le azioni da fare sono:

- definire e limitare lo spazio di attività nel sistema
- descrivere gli oggetti presenti nello spazio attraverso la loro geometria
- definire la/e telecamera/e virtuale che riprende lo spazio
- calcolare sul piano immagine come la scena viene vista dalla telecamera:
  - proiezioni degli oggetti sul piano di visualizzazione
  - capire quali oggetti sono visibili e quali no
  - determinare i pixel coperti dagli oggetti cioè la *scan conversion*
  - computare il colore, la luce e le ombreggiature su ogni sezione visibile.

La pipeline che vedremo è una **pipeline funzionale**: ciascun blocco contiene una delle operazioni che devono essere eseguite sui dati per definirne la proiezione sul piano immagine.

Ha un livello di astrazione molto elevato poiché la sua implementazione dipende dall'architettura hardware e dallo strato software ad esso collegato.

Le API fornite da tale strato di software per la programmazione della scheda grafica fanno riferimento di solito all'OpenGL o al Direct 3D in base alla scheda grafica che si ha a disposizione.

## Capacità della pipeline

---

Come calcoliamo la capacità della pipeline per ogni unità di tempo?

Di solito la capacità viene misurata in frames per secondo ma può anche essere rappresentata usando gli Hertz (Hz) che definisce la frequenza di aggiornamento dell'immagine come 1/secondi.

Gli Hz vengono usati di solito quando il valore non varia nel tempo (ad esempio per gli schermi che sono ad un rate fisso), a differenza dei framerate.

Infatti solitamente il frame rate può variare, in base alla complessità della scena da renderizzare.

Per questo il framerate può indicare sia il tasso per un determinato frame ma anche una media della performance durante l'uso del sistema.

Per velocizzare il sistema si possono usare alcuni trucchi quali quello di semplificare la geometria per gli oggetti lontani (i cui dettagli si perdono) e lasciarla complessa per gli oggetti vicini.

La capacità è determinata dal modulo più lento all'interno della pipeline che ne è il collo di bottiglia.

Dato il tempo impiegato da tale modulo (es. 20 millisecondi), il framerate viene calcolato come l'inverso del tempo impiegato: 1/secondi ( $1/0.020 = 50 \text{ Hz}$  o  $50 \text{ fps}$ ).

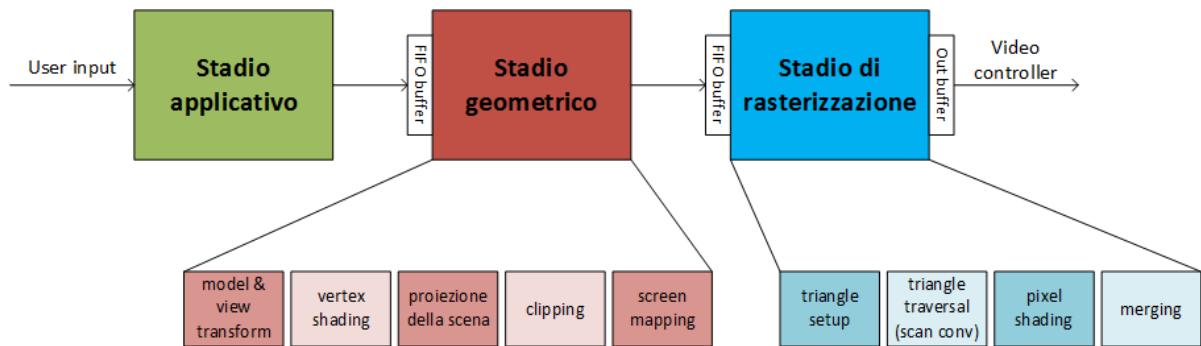
Tale valore deve però essere confrontato con il rate di visualizzazione dell'hardware: supponiamo che il framerate della pipeline di renderizzazione sia di 16fps; se il monitor a disposizione lavora a 60Hz può visualizzare 60 immagini diverse al secondo, o 30 se le immagini vengono mantenute il doppio del tempo, o 15 se vengono mantenute il quadruplo del tempo.

In questo caso il rate usato del monitor è di 15Hz, che è la velocità massima di output gestibile dallo schermo, inferiore alla capacità di renderizzazione (in questo caso 16Hz).

## Moduli della pipeline di rendering

La pipeline segue 3 stadi principali che possono essere a loro volta suddivisi in altri sottostadi (messi in pipeline o in parallelo).

- **stadio applicativo**: riceve gli input e gestisce la variazione degli oggetti nella scena
- **stadio geometrico**: elabora le primitive della scena (triangoli e altre...)
- **stadio di rasterizzazione**: preleva i dati precedenti e li elabora a livello di pixel per convertire la scena in un'immagine visualizzabile sul piano immagine.



### Stadio applicativo

In questo stadio ricade tutto ciò che viene scritto da programma ed è interamente gestito dalla CPU.

Ciò include:

- la gestione dell'input (dai vari dispositivi)
- la gestione degli spostamenti della camera e degli oggetti nella scena
- l'eventuale interazione fra gli oggetti o il loro movimento (animazione)
- la risposta fisica degli oggetti all'input (si pensi ai dispositivi aptici descritti in precedenza). La gestione della fisica può ricadere in questo modulo ed essere effettuata via software o essere gestita da una pipeline fisica che usa un motore fisico gestito in parte o totalmente dalla GPU (scheda grafica).

Uno dei problemi principali in una simulazione grafica è mantenere un frame rate sopra un certo livello pur avendo una scena molto complessa.

Tale problema può essere risolto in due modi:

- semplificazione dei modelli (**LOD = Level Of Detail**) ovvero usare meno poligoni.  
Per alcune implementazioni tale semplificazione viene effettuata nello stage della geometria usando direttamente la scheda grafica (dall'activex 10 in avanti), all'interno del sottomodulo **geometry shader**
- rappresentare i dati con precisione singola e non doppia.

Lo stadio applicativo dà in input allo stadio geometrico delle primitive di rendering come punti, righe e triangoli.

Una conseguenza dell'applicazione basata sul software di questa fase è che non è suddivisa in sottostadi, così come invece lo sono lo stadio geometrico e di rasterizzazione.

Tuttavia, per aumentare le prestazioni, questa fase viene spesso eseguita in parallelo su più core di processori.

Nella progettazione della CPU, questa è chiamata una **costruzione superscalare** in quanto è in grado di eseguire diversi processi allo stesso tempo nello stesso stadio.

## Stadio geometrico

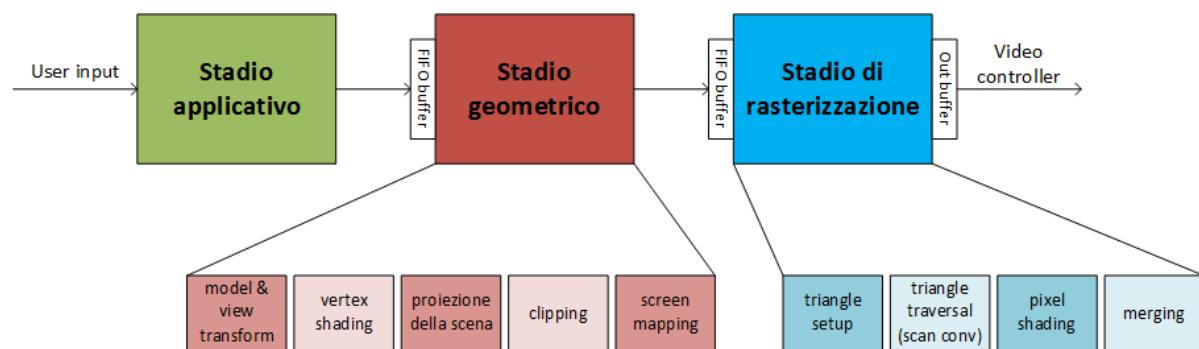
Lo stadio precedente fornisce a questo stadio una rappresentazione della scena aggiornata tramite primitive di cui la più semplice è il triangolo (un insieme di 3 vertici).

Tutte le operazioni qui svolte sono gestite dall'hardware (di solito dalla GPU):

- definizione del sistema di riferimento:
  - riferimento destroso ( $x, y, z$  = pollice, indice e medio della mano destra) o
  - riferimento sinistrorso ( $x, y, z$  = pollice, indice e medio della mano sinistra)
- definizione dei vertici visti come punti rappresentati da vettori a 3 componenti ( $x, y, z$ ) o 4 (di cui l'ultima è usata per effettuare in modo più semplice le trasformazioni) se abbiamo le coordinate omogenee ( $x, y, z, j$ )
- definizione degli oggetti in base alle coordinate centrate nel sistema di riferimento locale dell'oggetto stesso
- trasformazione delle coordinate precedenti in quelle del sistema di riferimento globale
- gestione delle trasformazioni delle primitive attraverso la trasformazione dei loro vertici.

L'output di questo stadio sono primitive geometriche 2D ovvero poligoni 2D.

Date le API fornite dalla scheda grafica, lo stadio geometrico è altamente programmabile o configurabile tramite i *vertex* e i *geometry shader*.



Di solito lo stadio geometrico è suddiviso in 5 sottostadi:

- (1) trasformazione dei modelli e della vista (model & view transform)
- (2) computazione della luce (vertex shading)
- (3) proiezione della scena
- (4) clipping
- (5) screen mapping.

### (1) TRASFORMAZIONE DEI MODELLI (MODEL & VIEW TRANSFORM)

Gli oggetti del modello a disposizione sono descritti tramite le loro primitive (solitamente triangoli, definiti dai vertici, definiti dalle loro coordinate nello spazio locale dell'oggetto).

La fase di trasformazione riceve il modello dell'oggetto nello *spazio locale* e lo rimappa nello *spazio globale*, applicando le eventuali trasformazioni di rotazione, scalamento e traslazione.

Tali trasformazioni vengono effettuate tramite matrici di trasformazioni applicate alle coordinate omogenee dei vertici delle primitive<sup>6</sup>.

Le trasformazioni vengono applicate a tutti gli oggetti, telecamera compresa.

<sup>6</sup> L'utilizzo delle coordinate omogenee ci permette di unire le operazioni di rotazioni, traslazione e scalamento tramite una sola matrice: la matrice è una 4x4, in cui la sezione 3x3 in alto a sinistra definisce i coefficienti per il rotoscalamento, mentre la quarta colonna e la quarta riga definiscono i coefficienti di traslazione.

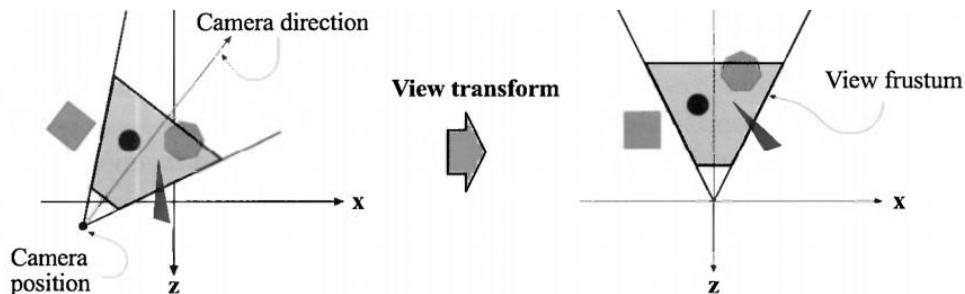
Il posizionamento della telecamera genera la creazione di un volume che viene detto **volume di visualizzazione (view frustum)** ed è un tronco di piramide la cui punta (che viene troncata) è definita dalla telecamera.

Il tronco di piramide è compreso fra:

- il **near plane** e il **far plane** che definiscono il piano più vicino e quello più lontano del volume (le basi del tronco di piramide) e
- i **top, down, right, left plane** (i piani laterali del tronco di piramide).

Dopo aver posizionato la telecamera c'è una nuova fase di trasformazione dei modelli, in cui il sistema di riferimento passa da quello globale a quello della telecamera, come mostrato in figura.

I due passaggi nei sistemi di riferimento vengono effettuati tramite matrici di trasformazione e pertanto possono essere computate in un colpo solo usando una matrice di trasformazione che è il prodotto di quelle delle singole trasformazioni.



## (2) COMPUTAZIONE DELLA LUCE (VERTEX SHADING)

Per produrre una scena realistica bisogna modellare l'aspetto degli oggetti.

Questo include il materiale dell'oggetto e l'effetto di qualsiasi sorgente luminosa che illumina l'oggetto.

I materiali e le luci possono essere modellate in molti modi, dai colori semplici ad elaborazioni di rappresentazioni delle descrizioni fisiche.

Questo è noto come **shading** (ombreggiatura): esso comporta l'elaborazione di un'equazione di ombreggiatura in vari punti dell'oggetto.

Tipicamente alcuni di questi calcoli vengono eseguiti durante la fase geometrica sui vertici di un modello (a livello di vertici) mentre altri possono essere eseguiti durante la rasterizzazione (a livello di pixel).

Questo sottostadio calcola le informazioni della luce a **livello di vertici** delle primitive che ha ricevuto dal sottostadio precedente: riceve  $n$  vertici e restituisce gli  $n$  vertici con le relative equazioni di illuminazione.

Calcola il colore della superficie basandosi su:

- a quale oggetto appartiene il vertice
- modello di illuminazione
- tipo e numero di sorgenti luminose che ci sono nella scena e come interagiscono con l'oggetto
- quali sono le caratteristiche fisiche dell'oggetto (se la superficie è opaca o trasparente, se riflette o diffonde la luce o entrambe)
- effetti atmosferici come la nebbia o il fumo.

I calcoli delle ombre sono di solito considerati nello spazio globale ma in pratica è a volte conveniente trasformare le entità rilevanti (come la telecamera e le sorgenti luminose) nello spazio della telecamera, in quanto le relative relazioni tra le entità vengono preservate se vengono trasformate nello stesso spazio.

I modelli di illuminazione usati in questo sottostadio sono (in ordine di complessità e semplicità):

- wire-frame models (effettua il calcolo ma non renderizza nulla)
- flat shaded models (l'illuminazione viene calcolata nella normale alla superficie e tale illuminazione è costante su tutta la superficie)
- Gouraud
- Phong shaded (modelli più complessi che usano l'illuminazione calcolata nei vertici e la mediano, tramite interpolazioni, nei diversi punti nella superficie).

### (3) PROIEZIONE DELLA SCENA

La scena viene proiettata sul piano immagine con l'obiettivo di trasformare ciascun punto posizionato nel sistema globale in una sua proiezione sul piano.

In questo momento non lavoriamo ancora sul piano immagine ma in un volume di contenimento che contiene ancora la componente di profondità ( $z$ ) usata nel seguito per risolvere i problemi di visibilità. Il volume di visualizzazione creato nel primo sottostadio viene trasformato, tramite matrici di trasformazioni, in un **volume di vista canonico** (cubo unitario), compreso fra i punti estremi  $(-1, -1, -1)$  e  $(1, 1, 1)$ .

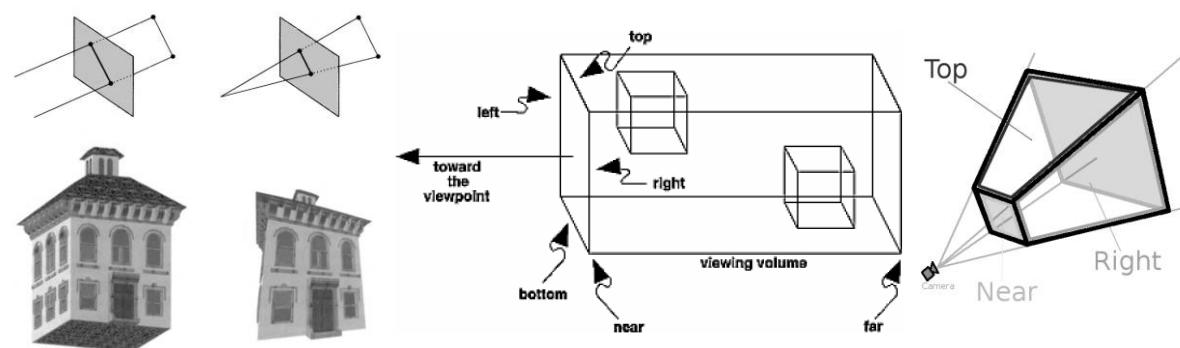
In base al tipo di telecamera usata vengono effettuate delle proiezioni diverse:

- se la telecamera è prospettica, il volume di visualizzazione è un tronco di piramide (**frustum**), e la proiezione effettuata è una proiezione prospettica
  - più lontano un oggetto si trova dalla telecamera e più piccolo appare dopo la proiezione
  - le linee parallele possono convergere all'orizzonte
  - la trasformazione prospettica simula il modo in cui si percepiscono le dimensioni degli oggetti
- se la telecamera è ortografica o parallela, il volume di visualizzazione è un cubo, e la proiezione è ortogonale
  - le linee parallele rimangono parallele dopo la trasformazione che è una combinazione di traslazioni e rotazioni.

Entrambe le trasformazioni (ortografica e prospettica) possono essere costruite con matrici  $4 \times 4$  e, dopo una trasformazione, si dice che i modelli sono in **coordinate del dispositivo normalizzate**.

Anche se queste matrici trasformano un volume in un altro, vengono chiamate proiezioni perché dopo la visualizzazione la coordinata  $z$  non viene memorizzata nell'immagine generata.

In questo modo i modelli sono proiettati da tre a due dimensioni.



Proiezione ortografica e prospettica.

#### (4) CLIPPING

Selezione dei punti e delle primitive che rientrano nell'area visibile (VV, **view volume**) dalla camera (parzialmente o totalmente), con l'esclusione di tutti gli altri.

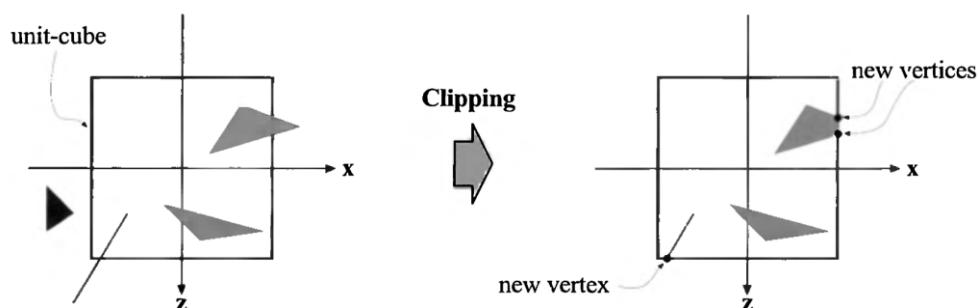
Le primitive:

- completamente all'interno → sono mantenute
- completamente all'esterno → vengono scartate
- parzialmente all'interno → vengono scomposte nei vertici e viene mantenuta la parte all'interno del volume.

Ad esempio, una linea che ha un vertice all'esterno e uno all'interno del volume di visualizzazione, dovrebbe essere tagliata: il vertice che è esterno è sostituito da un vertice che si trova nell'intersezione tra la linea e il VV.

Il vantaggio di eseguire la trasformazione del volume di visualizzazione e la proiezione prima del clipping è che le primitive vengono sempre tagliate contro il cubo unitario.

A differenza delle precedenti fasi geometriche, tipicamente eseguite da unità di elaborazione programmabili, la fase di clipping (nonché la successiva fase di screen mapping) viene solitamente elaborata mediante hardware fisso.



#### (5) SCREEN MAPPING

Solo le primitive (clippate) all'interno del volume di visualizzazione vengono trasmesse alla fase di screen mapping e le coordinate sono ancora tridimensionali.

Le coordinate  $x$  e  $y$  di ogni primitiva vengono trasformate per formare le **coordinate dello schermo (screen coordinates)**.

Le screen coordinates insieme alle coordinate  $z$  vengono chiamate anche **coordinate di finestra (windows coordinates)**.

Assumendo una finestra di rendering con l'angolo minimo a  $(x_1, y_1)$  e angolo massimo a  $(x_2, y_2)$ , dove  $x_1 < x_2$  e  $y_1 < y_2$ , lo screen mapping è una traduzione seguita da un'operazione di scalatura.

La coordinata  $z$  non è influenzata da questa mappatura.

Le nuove screen coordinates  $x$  e  $y$ , insieme alla coordinata  $z$  (con  $-1 < z < 1$ ), vengono trasmesse alla fase di rasterizzazione.

Riassumendo, questo sottostadio converte le componenti  $x, y$  delle primitive (comprese fra  $-1$  e  $1$  all'interno del volume di contenimento) in posizioni riga, colonna dei pixel dello schermo (comprese fra  $x_1, x_2$  e  $y_1, y_2$  in base alle dimensioni dello schermo).

Questa fase non è programmabile perché eseguita in hardware.

Tutte le API hanno valori di posizione dei pixel che aumentano da sinistra a destra.

La posizione dello zero per i bordi superiore e inferiore è diversa tra OpenGL e DirectX:

- OpenGL favorisce il sistema cartesiano in tutto, trattando l'angolo inferiore sinistro come elemento più basso
- DirectX definisce di solito l'angolo superiore sinistro come questo elemento.

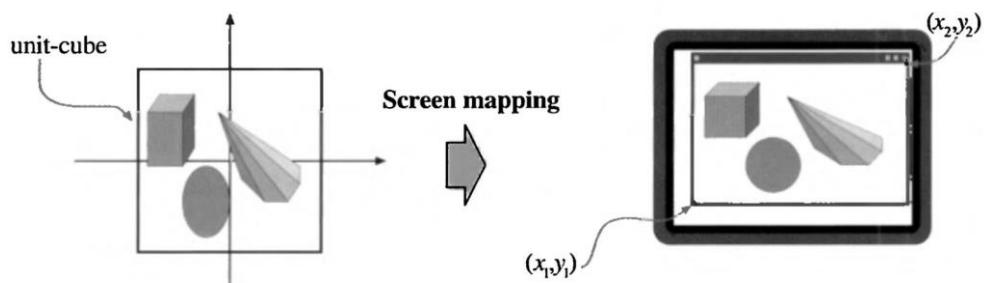
Inoltre ci sono delle differenze anche sullo scegliere il centro del pixel:

- DirectX9 e i suoi predecessori utilizzano un sistema di coordinate dove 0.0 è il centro del pixel, il che significa che l'intervallo di pixel  $[0, 9]$  copre l'intervallo  $[-0.5, 9.5]$ .
- OpenGL e DirectX10 usano 0.0 per il bordo sinistro del pixel più a sinistra. Il centro di questo pixel è a 0.5. Quindi il range di pixel  $[0, 9]$  copre l'intervallo  $[0.0, 10.0]$ .

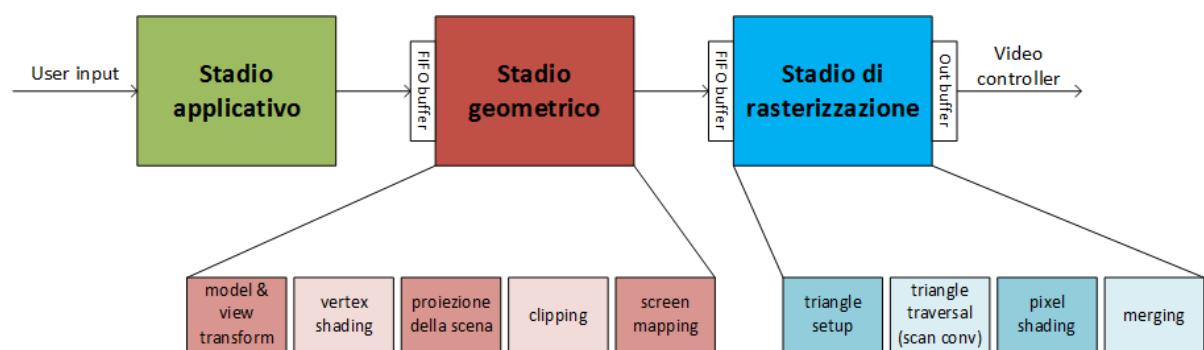
Le conversioni sono semplici:

```
d = floor(c);
c = d + 0.5;
```

dove  $d$  è l'indice discreto (*intero*) del pixel e  $c$  è il suo valore continuo (*floating point*).



## Stadio di rasterizzazione



Dati i vertici trasformati e proiettati con i relativi dati di shading (dalla fase geometrica), l'obiettivo della fase di rasterizzazione è quello di calcolare e impostare i colori dei pixel su cui giace l'oggetto.

Questo processo viene chiamato rasterizzazione o scan conversion ed è dunque la conversione da vertici bidimensionali nello spazio dello schermo (ciascuno con un valore z (profondità) e varie informazioni di ombreggiatura associate a ciascun vertice) in pixel sullo schermo.

È completamente implementato nell'hardware.

Converte l'output delle informazioni sui vertici in informazioni sui pixel necessarie al display video.

Lo stadio di rasterizzazione può essere suddiviso in:

- (1) triangle setup
- (2) triangle traversal (scan conversion)
- (3) pixel shading
- (4) merging.

## (1) TRIANGLE SETUP

Unisce i vertici in modo da ricostruire la singola primitiva (cioè il singolo triangolo).

Tale operazione non è modificabile in quanto è effettuata direttamente dall'hardware della scheda grafica.

## (2) TRIANGLE TRAVERSAL (SCAN CONVERSION)

Determina quali pixel ricadono all'interno di una certa primitiva.

Vengono definiti i **frammenti** cioè insieme di dati (posizione, colore, texture, profondità...) associati al singolo pixel per colorarlo correttamente.

Come prima, tale operazione non è modificabile.

## (3) PIXEL SHADING

Si occupa di definire il colore del singolo pixel, valutando texture, materiale degli oggetti e le loro componenti di illuminazione.

Questa fase è gestita dalla sezione programmabile della GPU (scheda grafica).

## (4) MERGING

Le informazioni per ogni pixel vengono memorizzate nel color buffer ovvero un array rettangolare di colori (componente R, G e B per ogni colore).

La fase di merging combina il colore del frammento prodotto dalla fase di shading con il colore attualmente memorizzato nel buffer.

A differenza della fase di shading, la sotto-unità di GPU che in genere esegue questa fase non è completamente programmabile, tuttavia è altamente configurabile, consentendo vari effetti.

Questa sottofase risolve anche la visibilità: quando l'intera scena è stata renderizzata, il buffer di colore dovrebbe contenere i colori delle primitive nella scena che sono visibili dal punto di vista della telecamera.

La visibilità viene risolta utilizzando l'algoritmo **z-buffer**.

Lo z-buffer è una matrice in cui ogni elemento, associato a un pixel dell'immagine da produrre, contiene la profondità della parte della scena più vicina all'osservatore che viene visualizzata dal pixel relativo.

```
Inizializza tutti gli elementi dello z-buffer a distanza grande (in teoria  $\infty$ )
Inizializza tutti gli elementi del color buffer al colore di background
Per ogni oggetto O
    Per ogni pixel P che contiene la proiezione di O //scan conversion di O
        Se (profondità della porzione di O associata a P < profondità memorizzata
            nell'elemento dello z-buffer associato a P)
            Calcola lo shading dell'oggetto nel punto;
            Aggiorna il colore nel color buffer per la posizione P;
            Aggiorna la profondità nello z-buffer per la posizione P;
    Fine per ogni pixel P
Fine per ogni oggetto O
```

È uno stadio configurabile, ma non programmabile che permette risolvere i problemi di visibilità degli oggetti utilizzando l'informazione sulla coordinata z per capire quali sono le primitive più vicine alle telecamere.

Per gli oggetti opachi è indifferente l'ordine in cui vengono scandite le primitive: si confrontano le z dei diversi oggetti (delle diverse primitive) per valutare qual è la più vicina, scartando tutte le altre.

La cosa è più complessa per gli oggetti non opachi.

Se stiamo considerando oggetti semitrasparenti o traslucidi non possiamo semplicemente scartare le informazioni relative agli oggetti più lontani che sono coperti, bensì miscelare le informazioni degli oggetti vicini e trasparenti con quelli lontani.

Per primitive semitrasparenti l'ordine è molto importante e si usa il metodo back-to-front: scandisco gli oggetti dal più lontano al più vicino e vado ad aggiornare l'informazione del colore nel color-buffer in base all'informazione che ho già presente nel buffer, l'oggetto dietro, e a quella della primitiva corrente che sto considerando.

Il nostro buffer colore contiene le informazioni RGB, ma anche un quarto canale, l'**alpha channel**, l'indice di opacità che ci dice quanto un oggetto è semitrasparente e ci serve proprio per capire come effettuare il merging in modo iterativo, per elaborare le informazioni di visibilità degli oggetti a partire dal più lontano al più vicino (ordine back-to-front).

Come viene utilizzato alpha per inserire il valore del colore del pixel nel frame buffer?

$$C'_D = A_S C_S + (1 - A_S) C_D$$

dove:

- $C_S$  rappresenta il colore del pixel che è stato determinato negli stadi precedenti
- $A_S$  rappresenta il valore di opacità del pixel che è stato determinato negli stadi precedenti
- $C_D$  il valore del colore del pixel inserito precedentemente nel buffer, esso viene sostituito dal nuovo  $C'_{DS}$
- se  $A_S = 0$  vuol dire che la primitiva che sto considerando è completamente trasparente, quindi  $C'_D$  sarà uguale al precedente  $C_D$  cioè il colore del pixel dietro.

Un'altra struttura dati importante è lo **stencil buffer** che viene usata per creare alcuni tipi di effetti, ottimizzare il rendering e aumentare il livello di realismo: l'uso dello stencil buffer è uno dei metodi utilizzabili per calcolare le ombre nell'immagine che dev'essere renderizzata.

È un buffer che viene utilizzato in modalità binaria (ogni elemento può assumere valore 0 o 1) che ci permette di capire se un pixel è interessato da un'ombra oppure no.

Come possiamo generare un'ombra?

Un'ombra troppo netta è non realistica e ha senso unicamente se abbiamo una sorgente luminosa totalmente puntiforme ma nei casi reali questo non succede mai quindi è utile attenuare le ombre per aumentare il livello di realismo.

La pipeline di rendering funziona utilizzando un doppio **buffer di colore**, un **front-buffer** e un **back-buffer**.

La scena nell'intera sequenza viene renderizzata nel back-buffer poi avviene uno swap dei buffer, viene presentato in visualizzazione, cioè sullo schermo, il front-buffer appena riempito e siamo pronti per renderizzare il nuovo frame che viene costruito nel back-buffer.

Questo serve a evitare che la scheda grafica che riempie il buffer crei un effetti di **flickering** cioè sfarfallio dovuto all'elaborazione in tempo reale del frame.

## Programmabilità della scheda grafica

In che modo si programma la funzionalità della pipeline? Utilizzando degli shader.

Quando è stata introdotta la scheda grafica, per le varie versioni delle librerie grafiche, il linguaggio di programmazione era di basso livello (simile all'Assembler) e la programmabilità era maggiormente legata allo stadio di rendering.

DirectX9 ha apportato notevoli novità definendo un linguaggio di shading ad alto livello (simile al C).

Questo ha facilitato di molto la programmabilità degli shader.

Il passo successivo, introdotto da DirectX10, ha permesso di unificare gli shader (*vertex* e *pixel* che vedremo nel seguito), inoltre ha introdotto il geometry shader che sta tra *vertex* e *pixel shader* e permette di creare degli effetti particolari mediante la creazione o eliminazione di vertici o pixel.

Linguaggi tipici per gli shader sono:

- **CG** C for graphics
- **HSSL** introdotto da Microsoft che è sostanzialmente un'implementazione di CG
- **GLSL** è un'implementazione di OpenGL.

Quelli elencati sono **linguaggi di medio livello (IL, intermediate language)**, sostanzialmente indipendente dall'architettura.

Il legame con la specifica architettura avviene al livello più basso, con l'interpretazione del linguaggio intermedio da parte delle librerie che fanno funzionare la scheda grafica ovvero i driver dei vari dispositivi.

## A programmable GPU pipeline



I moduli verdi sono totalmente programmabili, quelli azzurri sono a funzionalità fissa cablati sul silicio, quelli gialli sono quelli configurabili anche se non completamente programmabili.

La GPU implementa le fasi geometrica e di rasterizzazione precedentemente descritte.

Sono suddivisi in diverse fasi hardware con diversi gradi di configurabilità o programmabilità.

Le fasi fisiche sono suddivise in modo leggermente diverso rispetto alle fasi funzionali presentate per la pipeline generale.

### VERTEX SHADER

La funzionalità principale del **Vertex shader** è quella di effettuare il mapping delle coordinate nei vari sistemi di riferimento (locale dell'oggetto -> globale -> locale della telecamera).

Dato che è completamente programmabile è possibile modificare il modo in cui lo shader effettua il mapping cioè mettere mano alle trasformazioni che possono subire i vertici.

Lo shader permette inoltre di creare degli effetti di colorazione e luminosità particolari che vanno oltre alle operazioni standard.

Cosa non può fare un Vertex shader? Non può creare o distruggere vertici (cosa che può fare il Geometry shader).



## GEOMETRY SHADER

Il **Geometry shader** riceve i dati (che possono essere punti e primitive) dal Vertex shader e ha la capacità di distruggere primitive o crearne di nuove.

Viene usato con buoni risultati per la creazione di sistemi particellari in cui i punti vengono creati, spostati e poi distrutti.

Esempi ne sono le superfici che devono simulare l'acqua, i capelli o i fili di un prato, l'effetto della polvere che cade o del fumo che si sposta.

Il Geometry shader esegue anche la gestione del livello di dettaglio (**LOD**): se l'oggetto è molto grande (in termini di numero di poligoni che lo compongono) ed è molto lontano dalla telecamera è assolutamente inutile effettuare il calcolo su tutte le primitive dell'oggetto.

Per il periodo di tempo in cui l'oggetto si trova a tale distanza può essere sostituito da un oggetto con geometria più semplice.

Dunque le primitive possono essere distrutte e poi ricreate.

L'input del Geometry shader è una singola primitiva a  $n$ -vertex:

- un punto ( $n = 1$ )
- un segmento di linea ( $n = 2$ )
- un triangolo ( $n = 3$ )
- con fino a  $n$  ulteriori vertici che agiscono come punti di controllo.

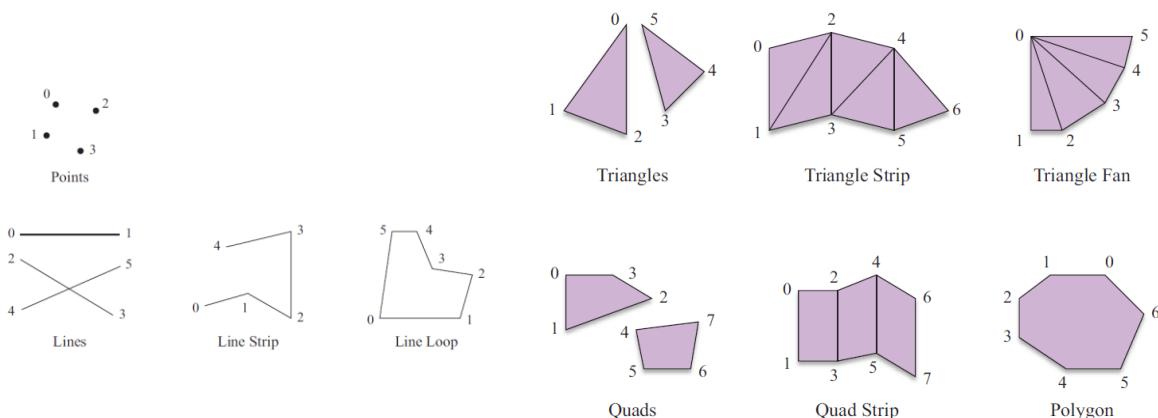
L'output è zero o più primitive (anche di tipo diverso dall'originale ricevuta in input).

Per esempio, il Geometry shader potrebbe convertire i punti in due quadrati (due triangoli) o trasformare i triangoli in altri triangoli ma, in opzione, eliminare alcuni triangoli e così via.

## OPENGL

La figura illustra i tipi di primitive grafiche definiti dalla libreria OpenGL.

L'hardware grafico è in grado di rendere un insieme di singoli punti, una serie di segmenti di linea o un gruppo di poligoni riempiti.



L'unità di elaborazione grafica (GPU) esegue le istruzioni indipendentemente dall'unità di elaborazione centrale (CPU).

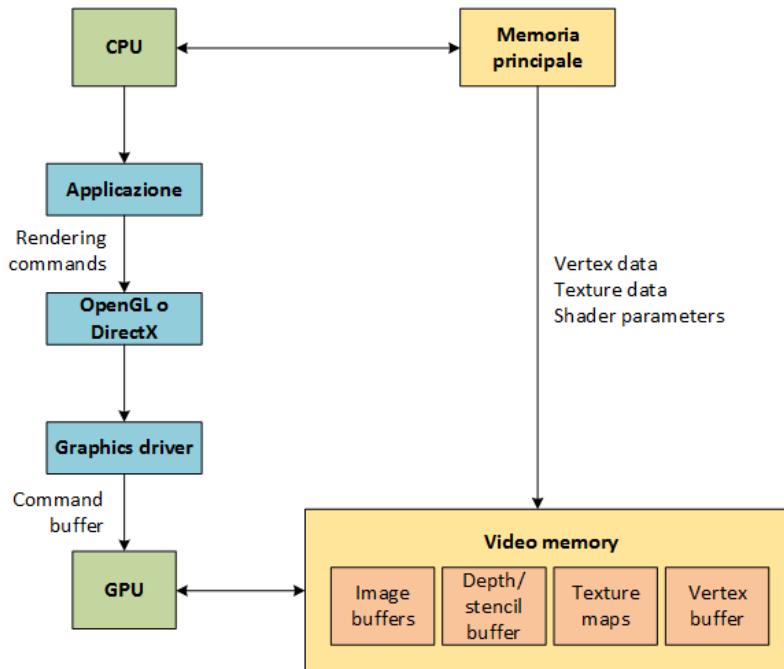
La CPU invia comandi di rendering alla GPU, che effettua le operazioni di rendering mentre la CPU continua con altre operazioni (**operazioni asincrone**).

Un'applicazione comunica con la GPU inviando comandi a una libreria di rendering, ad esempio OpenGL, che a sua volta invia comandi a un driver che sa parlare con la GPU nella sua lingua madre.

L'interfaccia alla libreria di rendering (OpenGL) è chiamata **HAL (Hardware Abstraction Layer)** perché espone un insieme comune di funzioni che possono essere utilizzate per rendere una scena su qualsiasi hardware grafico che supporti l'architettura OpenGL.

Il driver trasforma le chiamate di funzione OpenGL in un codice che la GPU può capire.

Questa figura illustra le comunicazioni che avvengono tra la CPU e la GPU:



Una scheda grafica 3D ha un proprio nucleo di memoria, comunemente chiamato **VRAM (Video Random Access Memory)**.

OpenGL utilizza diversi buffer per mantenere in memoria le informazioni riguardanti i pixel dell'area di rendering.

I buffer differiscono tra loro per il tipo di informazioni che contengono o per la loro posizione.

L'insieme di tutti i buffer si definisce OpenGL Frame Buffer.

Il ruolo svolto dai buffer in OpenGL è essenziale, a seconda del loro ruolo essi memorizzano particolari informazioni su ogni pixel del frame che si sta renderizzando.

Ci sono diversi tipi di dati che possono essere trovati quasi sempre nella memoria della scheda grafica quando è in esecuzione un'applicazione grafica 3D:

- **color buffer:** il buffer del colore è senza dubbio il più importante, esso è indispensabile dato che memorizza il colore di ogni pixel del frame (RGBA).  
Normalmente la dimensione del color buffer è di 32 bit, 8 bit (un byte dunque) per ognuna delle componenti.  
È però possibile creare buffer non simmetrici o addirittura privi di un canale (a volte il canale Alpha viene impostato a 0 bit se non si necessita di effettuare trasparenze o effetti simili).  
È possibile richiedere a OpenGL più di un color buffer, specificando l'opzione Double Buffer e/o Stereo Buffer.  
Quando chiediamo ad OpenGL un double buffer essa crea due buffer colore, uno principale (**Front buffer**) e uno nascosto (**Back buffer**).  
Le operazioni di disegno avvengono sul back buffer invisibile all'utente, quando il frame è completo il due buffer vengono swappati trasformando il back buffer nel buffer principale visualizzato a schermo e il front buffer nel buffer nascosto
- **front image buffer:** contiene i dati dei pixel che sono visibili nella visualizzazione

- **z-buffer:** è il buffer della profondità.  
Ad ogni pixel è associata una determinata profondità che naturalmente dipende dalla sua posizione all'interno della scena.  
Un pixel appartenente ad un oggetto molto lontano un alto valore di profondità mentre un oggetto molto vicino alla "camera" avrà un valore molto basso.  
Questo buffer è indispensabile per visualizzare gli oggetti in base alla distanza dal punto di vista invece che in base all'ordine nel quale vengono disegnati
- **stencil buffer:** permette di renderizzare sul frame buffer attraverso una maschera che ci consente di disegnare solo su alcune porzioni del frame.  
Per creare una maschera dobbiamo renderizzare sullo stencil buffer, per ogni pixel che andremo a modificare verrà memorizzato un 1.  
Renderizzando sempre sullo stesso frame un nuovo oggetto possiamo indicare di modificare un pixel solo se il valore relativo al dato pixel nello stencil buffer è uguale 1 ovvero appartenente alla maschera.  
In questo modo il nuovo oggetto verrà renderizzato solo all'interno dell'area precedentemente delimitata.  
In base alla dimensione in bit dello stencil buffer è possibile creare diversi strati di maschere.  
Viene utilizzato per le ombre realistiche, riflessioni, filtri, rendering all'interno di forme irregolari
- **texture maps:** le immagini che vengono applicate alla superficie di un oggetto per dare maggiori dettagli visivi.  
Nelle applicazioni di rendering avanzate, le texture maps possono contenere informazioni diverse rispetto a una semplice immagine.  
Ad esempio, una *bump map* contiene vettori che rappresentano le pendenze che variano in base alle diverse posizioni sulla superficie di un oggetto.

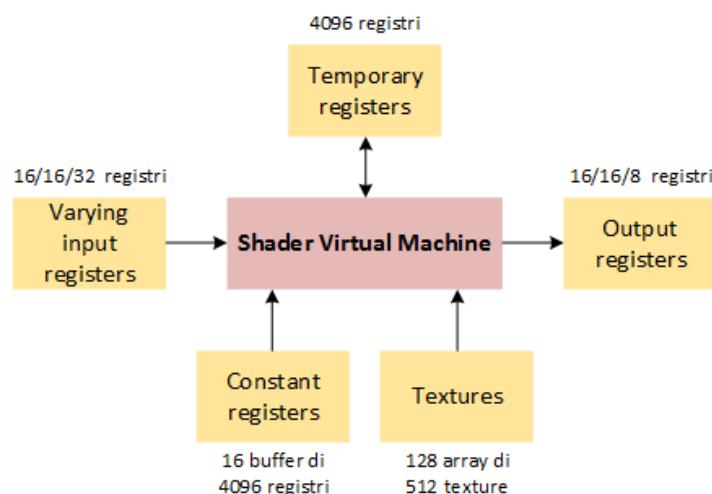
Le applicazioni non sono costrette ad utilizzare tutti i buffer disponibili, infatti ogni programma OpenGL prima di iniziare qualsiasi operazione di rendering si occupa di selezionare e attivare un Frame Buffer adatto alle proprie esigenze.

## Shader programming model

---

Una fase di shader programmabile ha due tipi di input:

- **input uniformi** che restano costanti durante le chiamate
- **input variabili** che sono diversi per ogni vertice o pixel.



Un shader può accedere alla RAM indirettamente tramite i registri.  
 Tutti i registri della GPU sono in formato SIMD a 128 bit.  
 Ogni registro è in grado di tenere quattro valori a virgola mobile o interi a 32 bit.  
 Inoltre ogni registro può contenere un vettore a quattro elementi in coordinate omogenee o un colore in formato RGBA, con ogni componente in formato a virgola mobile a 32 bit.  
 Le maschere possono essere rappresentate da gruppi di tre o quattro registri.  
 Un registro della GPU può anche essere utilizzato per contenere un singolo scalare a 32 bit, nel qual caso il valore viene normalmente replicato in tutti e quattro i campi a 32 bit.  
 Alcune GPU possono operare su campi a 16 bit, conosciuti come **half** (metà).

Gli shader effettuano le modifiche usando un insieme di informazioni memorizzate nei registri:

- **registri di input**: ricevono le informazioni in input, ad esempio nel vertex shader ricevono i vertici
- **registri di costanti**: tutto ciò che non è modificabile viene gestito con dei registri costanti che permettono di elaborare la primitiva tenendo costante il parametro in questione
- **registri temporanei**: usati come appoggio per i calcoli intermedi
- **registri di output**: contengono le informazioni date in output dallo shader, quali le nuove posizioni dei vertici, il loro colore, le coordinate della texture associata, etc.

Ciascuno di questi registri contiene un'informazione di 32 bit in formato **SIMD (Single Instruction Multiple Data)**, e sui quali può essere effettuata la medesima istruzione.

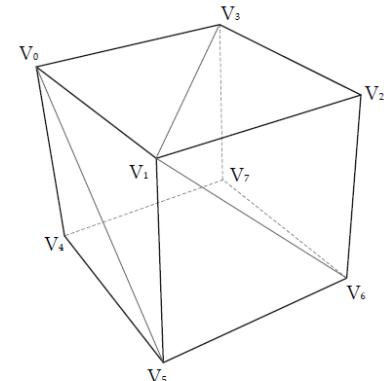
---

### Come definire una mesh usando triangoli

---

Il modo più semplice per definire una mesh è semplicemente quello di elencare i vertici in gruppi di tre.

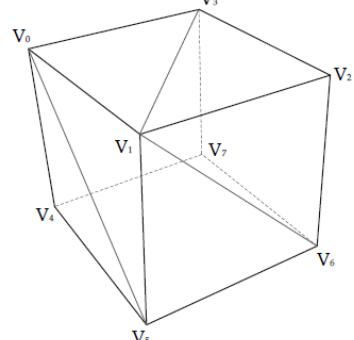
Questa è una tecnica molto efficiente quando si debba memorizzare un singolo poligono.  
 Nel caso di una rete di poligoni, le coordinate dei vertici condivisi sono duplicate con ovvio spreco di memoria e perdita di efficienza, in quanto i vertici comuni vengono disegnati più volte.



[V <sub>0</sub>   V <sub>1</sub>   V <sub>3</sub> ]	[V <sub>1</sub>   V <sub>2</sub>   V <sub>3</sub> ]	[V <sub>0</sub>   V <sub>5</sub>   V <sub>1</sub> ]	...	[V <sub>5</sub>   V <sub>7</sub>   V <sub>6</sub> ]
---	---	---	-----	---

Nel caso invece in cui i poligoni siano definiti tramite una lista di puntatori a una sequenza di vertici, ciascun vertice della rete di poligoni è memorizzato una sola volta nella lista dei vertici.

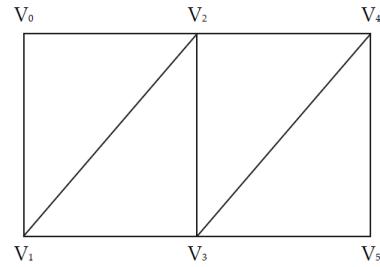
Poiché ciascun vertice è memorizzato una sola volta, viene risparmiata una considerevole quantità di memoria. Le coordinate dei vertici possono inoltre essere facilmente cambiate.



Vertices	[V <sub>0</sub>   V <sub>1</sub>   V <sub>2</sub>   V <sub>3</sub>   V <sub>4</sub>   V <sub>5</sub>   V <sub>6</sub>   V <sub>7</sub> ]
Indices	[0   1   3   1   2   3   0   5   1   ...   5   7   6]

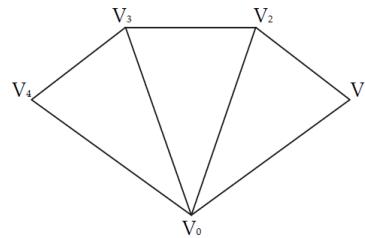
Alcune volte per renderizzare si usano strutture dati specializzate conosciute come **triangle strips** e **triangle fans**.

Entrambe queste strutture dati eliminano la necessità di un index buffer, pur riducendo in qualche misura la duplicazione dei vertici.



Vertices [V<sub>0</sub> | V<sub>1</sub> | V<sub>2</sub> | V<sub>3</sub> | V<sub>4</sub> | V<sub>5</sub>]

Interpreted  
as triangles: [0 1 2] [1 3 2] [2 3 4] [3 5 4]



Vertices [V<sub>0</sub> | V<sub>1</sub> | V<sub>2</sub> | V<sub>3</sub> | V<sub>4</sub>]

Interpreted  
as triangles: [0 1 2] [0 2 3] [0 3 4]



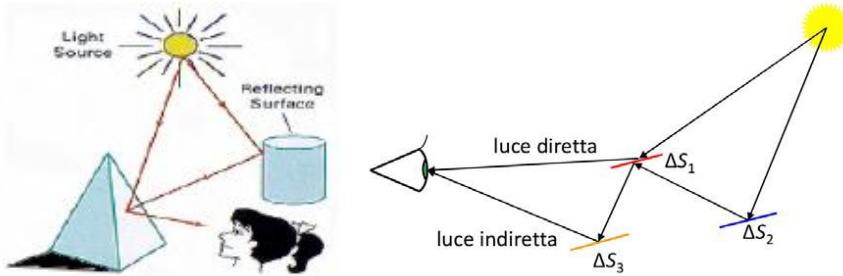
## 7. Modelli di illuminazione

Generare un corretto modello di illuminazione permette di aumentare il realismo dell'immagine, simulando gli effetti creati dall'interazione dei diversi tipi di luce con le superfici presenti nella scena. Gli attori che consideriamo per generare un modello sono:

- **sorgenti luminose**: caratterizzate da posizione nello spazio, spettro elettromagnetico, forma e direzione della luce
- **superfici**: caratterizzate da posizione nello spazio (di importanza notevole è la normale alla superficie), proprietà di riflettanza, diffusione, trasparenza del materiale e l'eventuale interazione con le altre superfici della scena
- **sistema visivo umano e piano immagine**: caratterizzato dalla posizione e dallo spettro di sensibilità del sistema.

Un **modello di illuminazione** viene utilizzato per calcolare l'intensità della luce che si riflette in un dato punto di una superficie.

Un **metodo di rendering** utilizza i calcoli di intensità del modello di illuminazione per determinare l'intensità della luce in tutti i pixel dell'immagine.



I raggi luminosi che provengono direttamente da una sorgente luminosa e colpiscono la superficie  $\Delta S$  costituiscono l'**illuminazione diretta**.

I raggi luminosi che interessano una superficie elementare  $\Delta S$  e provengono da altri elementi di superfici costituiscono invece l'**illuminazione indiretta**.

È detta **ombra** l'effetto prodotto da oggetti che bloccano i raggi luminosi provenienti da sorgenti di luce.

Le sorgenti luminose possono essere così discriminate:

- **point source**: sorgente puntiforme che emette luce in tutte le direzioni con la stessa intensità (viene detta isotropa).

Data la sorgente in  $P_S$  e un punto nello spazio  $P_0$  il raggio di luce che parte dalla sorgente e colpisce il punto  $P_0$  può essere descritto dall'equazione:

$$r_1(t) = P_S + (P_0 - P_S)t$$

- **parallel source**: i raggi luminosi che vengono emessi dalla sorgente sono paralleli tra loro.

Sono utili per modellare sorgenti molto lontane dall'oggetto (ad esempio il sole).

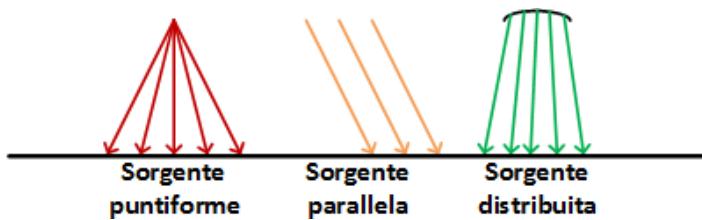
Un raggio con direzione  $d$  che parte dalla sorgente e colpisce il punto  $P_0$  può essere rappresentato dall'equazione:

$$r_2(t) = P_0 + dt$$

- **distributed source**: i raggi luminosi sono originati da una superficie ben definita nello spazio.

Sono utili per modellare sorgenti estese quali i neon; un tipo particolare sono gli **spot**, in cui la sorgente è puntiforme ma i raggi vengono emessi solo in un cono di direzioni.

In base al tipo di sorgente i raggi possono essere descritti dalle equazioni  $r_1(t)$  o  $r_2(t)$ .



Ricordandoci che il modello di illuminazione deve essere applicato in fase di rendering, l'obiettivo rimane quello di calcolare il modello nel modo più veloce possibile, in modo da poter generare dei risultati sufficienti nel tempo concesso dal ciclo di rendering.

Per velocizzare questi calcoli si usano dei modelli semplificati, nei quali viene limitato il livello di dettaglio degli oggetti oppure non vengono gestite le riflessioni di luce fra oggetti o le ombre.

Tale metodi di semplificazione portano però a dei risultati subottimali e poco realistici perciò vengono usati di rado.

Vediamo ora le componenti di illuminazione usate per la costruzione di un modello completo.

L'interazione tra la luce e i materiali di cui è costituita una superficie  $\Delta S$  può essere schematizzata nel seguente modo:

- **riflessione speculare**: la luce è riflessa in una singola direzione che dipende dall'angolo di incidenza del raggio sulla superficie e dalla normale alla superficie stessa
- **riflessione**: la luce incidente viene restituita in un singolo angolo solido molto piccolo con direzione che dipende dall'angolo di incidenza del raggio sulla superficie e dalla normale alla superficie stessa
- **diffusione**: la luce è dispersa in tutte le direzioni e se l'intensità è la stessa si parla di perfetta diffusione
- **rifrazione**: la superficie si lascia attraversare dalla luce che emerge poi da un'altra superficie dell'oggetto
- **assorbimento**: la luce non viene restituita.

L'interazione di una radiazione luminosa con una superficie dipende:

- a) dalla lunghezza d'onda della radiazione luminosa
- b) dall'angolo di incidenza tra superficie e radiazione luminosa
- c) dalla natura e microstruttura del materiale che costituisce la superficie.

La simulazione del fenomeno dell'illuminazione (*lighting*) delle singole  $\Delta S$  considerando solo i raggi luminosi provenienti dalle sorgenti di luce è indicata come **illuminazione locale**.

Se si tiene invece conto del contributo dovuto a fenomeni di interazione della luce con gli oggetti presenti nella scena si parla di **illuminazione globale**.

Il pregio dell'illuminazione locale consiste nel fatto che gli algoritmi sono sovente semplificati e veloci ma per contro non gestiscono le ombre e nemmeno i fenomeni di inter-riflessione e rifrazione.

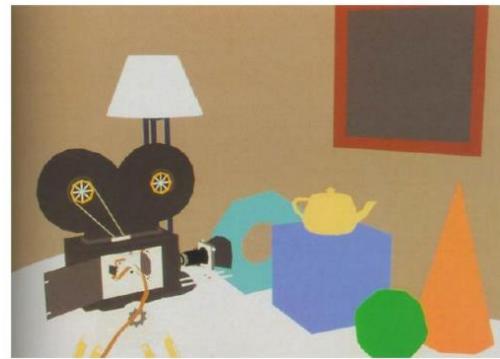
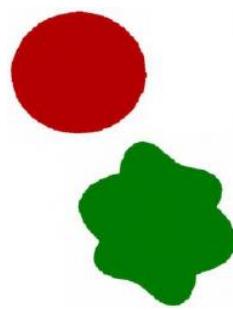
## Illuminazione ambientale

L'**illuminazione ambientale** è un tipo particolare di illuminazione non-direzionale cioè un'illuminazione di sfondo.

Tale illuminazione è utile per illuminare zone non colpite da sorgenti luminose, in modo che non vengano renderizzate come aree completamente nere.

L'illuminazione così costruita è la stessa per ogni punto di ogni oggetto della scena in quanto possiamo immaginare di "accendere le superfici" indipendentemente dalla loro posizione.

Da sola quindi non basta visto che gli oggetti risultano piatti.



Esempio di scena illuminata unicamente con la luce ambientale.

L'intensità riflessa  $I_{amb}$  di ogni punto della superficie è:

$$I_{amb} = K_a I_a$$

dove  $I_a$  è l'intensità della luce ambientale mentre  $K_a \in [0,1]$  definisce il **coefficiente ambientale** cioè la predisposizione della superficie a riflettere la luce ambientale (valore che dipende dal materiale e dalle caratteristiche fisiche della superficie).

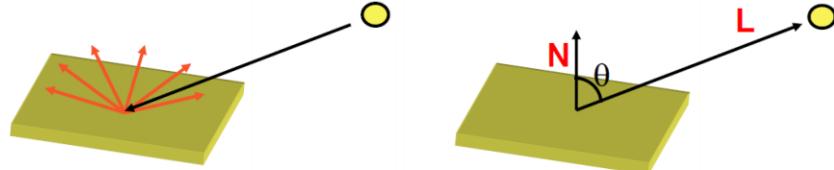
Sia  $K_a$  che  $I_a$  possono essere funzioni del colore, calcolando  $l_{amb}^r$ ,  $l_{amb}^g$ ,  $l_{amb}^b$ .

Ciò può essere utile per definire il colore dell'oggetto<sup>7</sup>.

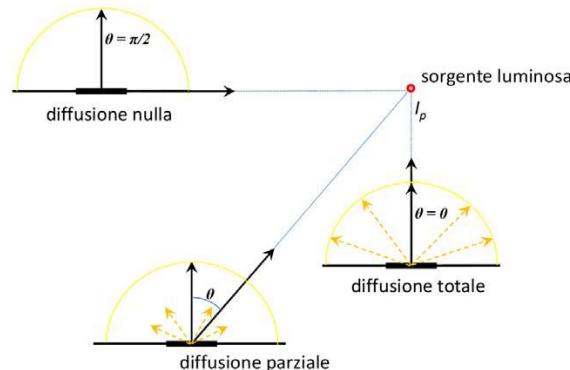
### Riflessione diffusiva (modello Lambertiano)

In aggiunta all'illuminazione ambientale si può considerare il modello Lambertiano: il **modello di illuminazione di Lambert** prevede che una superficie colpita da un raggio luminoso rifletta tale raggio in ogni direzione.

La quantità di energia luminosa emessa dal punto colpito, dipende dall'angolo  $\theta$  fra il raggio luminoso che incide  $L$  e la normale  $N$  alla superficie in quel punto.



La superficie risulta illuminata in ogni direzione la si guardi e l'intensità luminosa che viene colta dipende dall'angolo fra la normale alla superficie e il raggio incidente.



Incidenza a vari angoli di un raggio di luce su una superficie e relativa diffusione (Lambert).

<sup>7</sup> Si ricordi che un oggetto che vediamo di colore rosso, è un oggetto che assorbe/non riflette il colore rosso ( $K_{amb}^r = 0$ ), mentre riflette gli altri colori ( $K_{amb}^g$  e  $K_{amb}^b > 0$ ).

Tanto minore è l'angolo  $\theta$  fra i due vettori, tanto maggiore è l'intensità della luce diffusa. L'intensità è nulla se l'angolo è maggiore di 90 gradi, in quanto il raggio arriva da una sorgente parallela o dietro la superficie.

Matematicamente il modello è descrivibile come:

$$I_{diff} = K_d I_p \cos \theta = K_d I_p (N \cdot L)$$

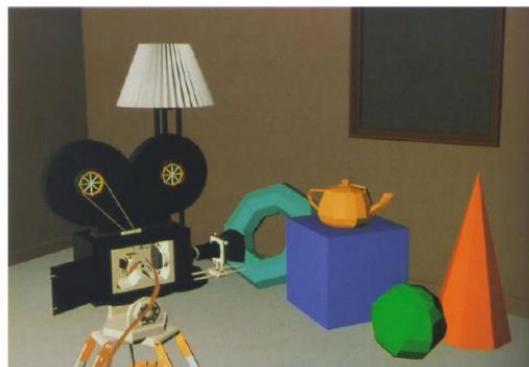
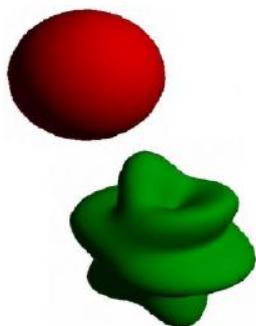
dove  $I_p$  è l'intensità del raggio luminoso incidente,  $K_d \in [0,1]$  rappresenta il coefficiente di diffusione della superficie,  $N$  è il vettore che identifica la normale alla superficie,  $L$  è il vettore che identifica la direzione del raggio luminoso<sup>8</sup>.

La superficie è visibile con la stessa luminosità da qualsiasi parte la si guardi: una superficie completamente diffusiva ha una rugosità tale per cui le microrugosità fanno riflettere il raggio in modo analogo su tutte le direzioni, senza una direzione di preferenza.

Il modello di Lambert è abbastanza facile da applicare ma non esistono oggetti ideali che siano completamente descrivibili con tale modello.

Solitamente la riflessione diffusiva viene utilizzata nel calcolo del modello di illuminazione, ma viene affiancata ad altri modelli.

Il modello Lambertiano, rispetto all'uso della sola illuminazione ambientale, produce un notevole incremento del senso di tridimensionalità con effetti diversi in base alla direzione della sorgente luminosa.



Esempio di scena illuminata con la luce ambientale e con il modello Lambertiano.

Gli aspetti negativi del modello sono:

- ▼ le immagini sono molto contrastate come accade quando si fotografi una scena in una giornata di sole molto luminosa
- ▼ le parti illuminate diffondono una grande quantità di energia mentre quelle in ombra appaiono molto scure con pochissimi particolari.

Solitamente ci sono due tipi di sorgenti luminose: una luce ambientale in background e una sorgente luminosa.

Il modello che unisce il Lambertiano all'ambientale produce un'illuminazione locale che matematicamente può essere descritta come segue:

$$I = I_{diff} + I_{amb} = K_d I_p (N \cdot L) + K_a I_a$$

Questo modello deve essere replicato per ogni canale di colore  $I_R, I_G, I_B$ .

---

<sup>8</sup>  $\cos \theta = NL$  solo perché  $N$  e  $L$  sono versori, cioè vettori con modulo unitario.

## Riflessione speculare

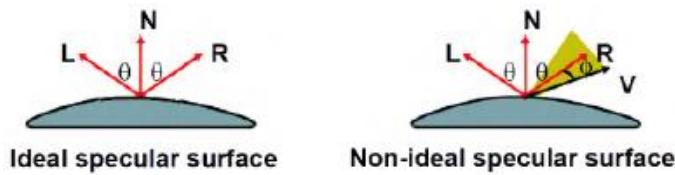
La diffusione non è la sola possibile interazione fra la luce e le superfici: ogni superficie è dotata di un certo grado di riflessione che è massima in superfici completamente lisce (quali gli specchi) e diminuisce con l'incremento di rugosità del materiale (in modo opposto alla diffusione).

Il modello di riflessione speculare di cui parliamo ora viene usato per descrivere gli effetti della riflessione speculare della luce su superfici completamente speculari (o simili): ad ogni raggio di luce incidente sulla superficie corrisponde un solo raggio emesso, il quale è complanare a quello incidente rispetto alla normale alla superficie.

Come nel caso del modello Lambertiano, anche qui sappiamo che, dato un raggio incidente, la luce viene emessa non in un unico raggio (le superfici perfettamente speculari sono rare quanto quelle completamente diffuse) ma in un cono di direzioni.

Pertanto, dato un raggio luminoso incidente  $L$ , il cono di direzioni è centrato in un vettore  $R$  (il raggio riflesso della perfetta riflessione speculare).

L'intensità diminuisce allontanandosi dalla direzione di  $R$ .

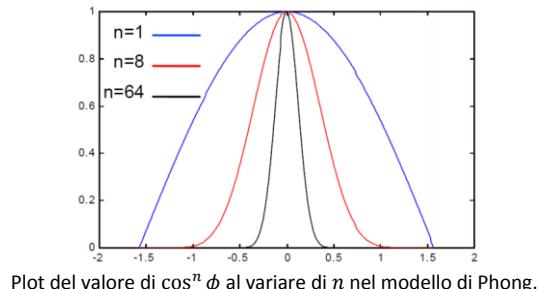


Il modello che descrive questi fenomeni è chiamato **modello di Phong** ed è:

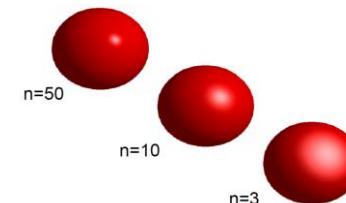
$$I_{spec} = K_s I_p \cos^n \theta = K_s I_p (R \cdot V)^n$$

dove  $K_s$  descrive le caratteristiche di riflettanza della superficie ed  $n$  è un parametro di riflessione speculare della superficie<sup>9</sup>.

Come si può notare l' $I_{spec}$  è massima se l'osservatore (la cui direzione è caratterizzata da  $V$ ) si trova in corrispondenza del raggio riflesso (la cui direzione è caratterizzata da  $R$ ), cioè quando  $R \cdot V = 1$ .



Plot del valore di  $\cos^n \phi$  al variare di  $n$  nel modello di Phong.



Variazione dell'effetto della riflessione al variare di  $n$ .

Aggiungendo il modello di Phong a quelli visti precedentemente l'equazione che descrive il modello di illuminazione costruito finora è:

$$I = I_{diff} + I_{spec} + I_{amb} = K_d I_p (N \cdot L) + K_s I_p (R \cdot V)^n + K_a I_a$$

Se avessimo  $k$  sorgenti luminose nella scena, l'illuminazione diventerebbe:

$$I = I_{amb} + \sum_k I_{diff}^k + I_{spec}^k$$

<sup>9</sup> Il valore usato per  $n$  definisce la capacità di riflessione speculare della superficie: per uno specchio riflettente  $n \rightarrow \infty$ , per una superficie diffusiva  $n \rightarrow 0$ .

Come si può notare nelle due figure, a valori alti di  $n \cos^n \phi$  diventa un impulso, provocando un cono di riflessione molto piccolo (tende ad esserci solo un raggio riflesso per ogni raggio incidente); per valori bassi invece il cono di riflessione si apre, provocando una diffusione della luce.

## Modello di Blinn Phong

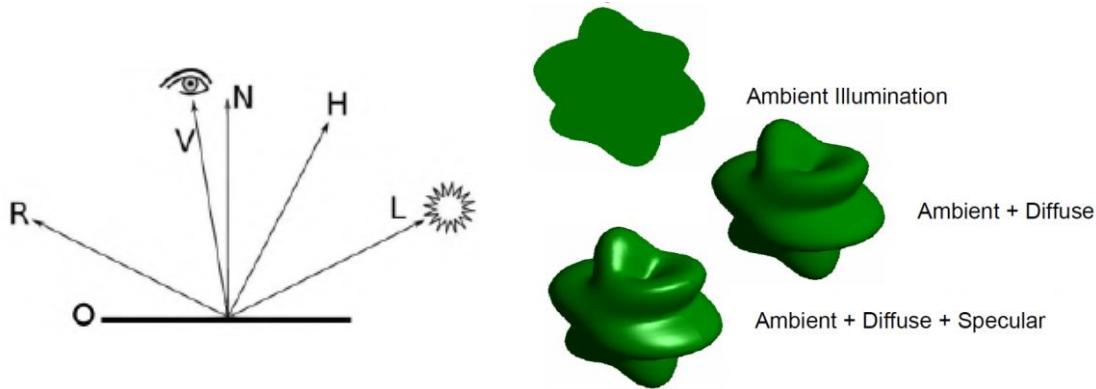
Uno dei problemi legati al modello di Phong è che  $R$  (il raggio riflesso) ci è sconosciuto e dovremmo andare a calcolarlo ad ogni ciclo.

Per evitare ciò si usa un'approssimazione che è definita dal **modello di Blinn Phong** in cui l'illuminazione è definita da:

$$I_{spec} = K_s(N \cdot H)^n \quad \text{con} \quad H = \frac{L + V}{|L + V|}$$

Il vantaggio è che  $H$  viene calcolato una volta sola per tutti i punti e tutte le superfici, in quanto se la luce non si sposta e l'osservatore non si sposta il calcolo va bene per ogni superficie, a differenza del raggio riflesso  $R$ , che dipende dalla superficie considerata.

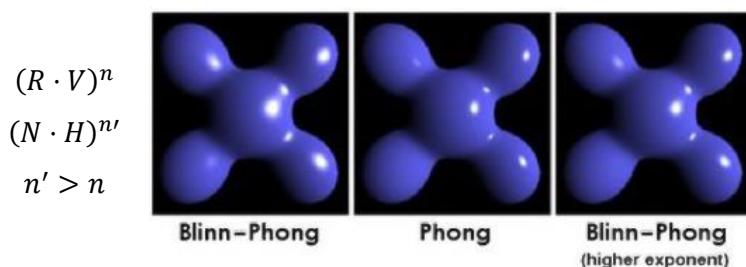
Per questo motivo all'interno dei software in uso oggi si usa il modello di Blinn Phong al posto di quello di Phong.



Una considerazione da fare è che l'angolo compreso fra  $N$  e  $H$  (Blinn Phong) e quello fra  $R$  e  $V$  (Phong) non sono lo stesso, ma il primo è esattamente il doppio del secondo:

$$\begin{aligned} 2 \cdot \widehat{OR} + \widehat{RV} + 2 \cdot \widehat{VH} &= \pi \\ \widehat{VH} &= \widehat{NH} + \left( \frac{\pi}{2} - \widehat{RV} - \widehat{OR} \right) \\ &\downarrow \\ 2 \cdot \widehat{OR} + \widehat{RV} + 2 \cdot \left( \widehat{NH} + \left( \frac{\pi}{2} - \widehat{RV} - \widehat{OR} \right) \right) &= \pi \\ 2 \cdot \widehat{OR} + \widehat{RV} + 2 \widehat{NH} + \pi - 2 \widehat{RV} - 2 \widehat{OR} &= \pi \\ \widehat{RV} &= 2 \widehat{NH} \end{aligned}$$

Data la differenza potremmo pensare che non si può sostituire il primo al secondo ma tale differenza può essere ignorata aumentando il valore dell'esponente  $n$ : rivalutando  $n$  otteniamo dei valori di riflettanza molto simili, nonostante la differenza fra gli angoli.



## Modelli di ombreggiatura

I modelli di illuminazione visti finora vengono elaborati dalla scheda grafica ma a differenza di ciò che abbiamo visto finora, la GPU non elabora i singoli punti degli oggetti ma lavora su vertici, primitive e superfici, in quanto il calcolo su ogni punto sarebbe eccessivamente costoso.

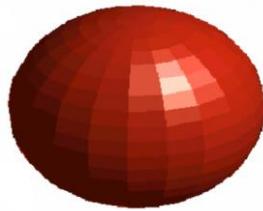
Ciò accade perché se consideriamo una primitiva, l'equazione di illuminazione si mantiene tale (o molto simile) in ogni suo punto.

Pertanto si può pensare di calcolare il modello di illuminazione in un punto della superficie ed estenderlo a tutta la primitiva oppure di interpolare il valore del modello di illuminazione calcolato in alcuni punti chiave della primitiva per ottenere il valore negli altri punti.

### Flat shading

Il modello più semplice è quello piatto, in cui l'intensità è calcolata una volta sola per ogni superficie, supponendo che l'oggetto sia un poliedro (unione di  $k$  primitive piane),  $N \cdot L$  (normale alla superficie e direzione della luce) si mantiene costante in ogni superficie, ma anche  $V \cdot R$  (punto di vista dell'osservatore e raggio riflesso) sono costanti.

Se la superficie è descritta da pochi poligoni, l'effetto è di frammentare la figura.



### Gouraud shading

L'idea dell'ombreggiatura di Gouraud è di calcolare l'equazione di illuminazione sui vertici della primitiva e poi interpolare i valori così ottenuti per determinare l'illuminazione negli altri punti.

Per fare ciò è necessario:

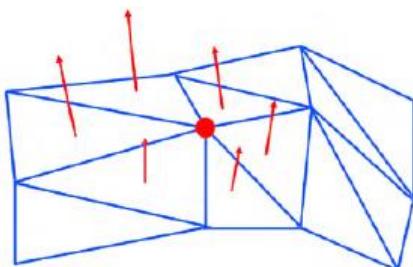
1. determinare la normale ad ogni vertice della primitiva
2. calcolare il valore delle equazioni di illuminazione in questi vertici
3. interpolare tali valori.

La normale al vertice è calcolata come la media delle normali di tutti i poligoni a cui appartiene il vertice considerato.

Matematicamente è calcolata come:

$$N_v = \frac{\sum_k N_k}{|\sum_k N_k|}$$

dove  $N_k$  sono le normali delle primitive adiacenti.



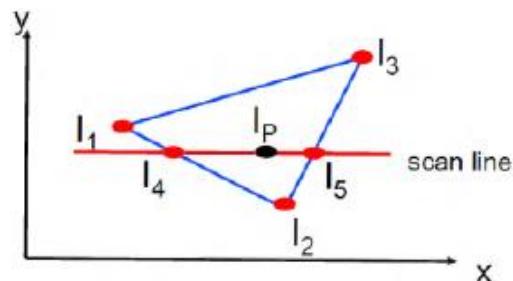
L'interpolazione in un punto appartenente alla primitiva, viene calcolata come:

- media dei valori nei vertici pesata rispetto alla distanza del punto dai vertici, se il punto è sullo spigolo del poligono
- media dei valori sugli spigoli pesata rispetto alla distanza del punto dagli spigoli, se il punto non è sullo spigolo.

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

$$I_P = \frac{x_5 - x_P}{x_5 - x_4} I_4 + \frac{x_P - x_4}{x_5 - x_4} I_5$$

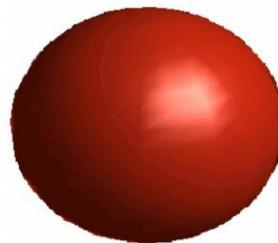


Questo tipo di ombreggiatura gestisce bene la componente diffusiva, ma in modo pessimo quella riflettente.

Ciò accade perché, se usiamo come riferimento il modello di Phong, l'equazione di illuminazione per un punto contenuto nella primitiva si basa sulla normale al poligono, mentre attraverso l'ombreggiatura di Gouraud, l'equazione si basa sull'interpolazione, la quale a sua volta usa le normali nei vertici.

Il risultato è quindi diverso.

Inoltre, non appena ci si sposta da un poligono a quello adiacente, il cambiamento di un vertice (quello non terminale dello spigolo che separa i due poligoni) provoca un grande effetto di discontinuità, molto fastidioso.



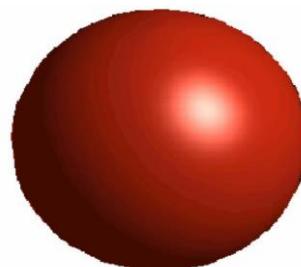
### **Phong shading**

Come soluzione ai problemi presentati nel modello precedente si usa il modello di Phong:

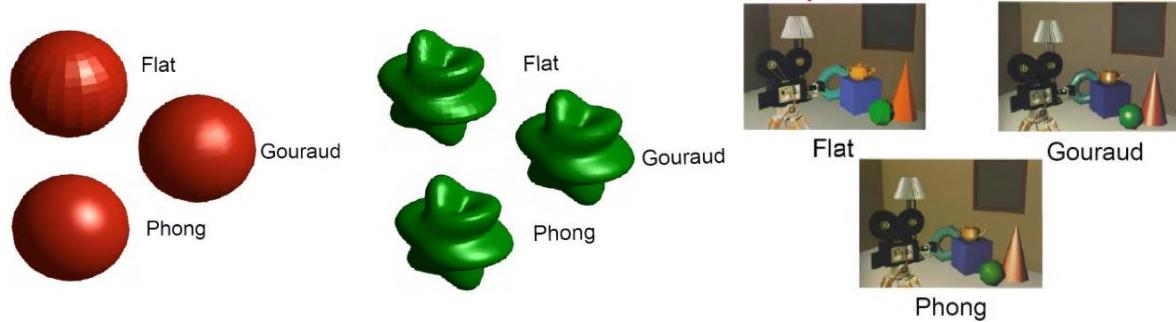
1. si determinano le normali nei vertici
2. si determina per ogni pixel che interessa il poligono considerato il vettore normale attraverso l'interpolazione di quelle calcolate nei vertici
3. si applica a tali normali il modello di illuminazione preferito.

Ciò aumenta la complessità dei calcoli, ma evita che la superficie risulti piatta.

Questo modello di ombreggiatura gestisce in modo efficiente sia la componente diffusiva che quella riflettente, poiché i modelli di illuminazione applicati (tipo Phong) lavorano su normali che sono simili a quelle reali.



Confronti:



## Modelli di illuminazione globale

Nei modelli presi in esame finora non si è tenuto conto dello scambio energetico fra i diversi oggetti provocati dalla riflessione della luce fra gli oggetti.

Allo stesso modo abbiamo ignorato i fenomeni di ombreggiatura, intesi come fenomeni provocati da un oggetto che occlude la luce ad un altro oggetto.

Ora, con lo studio dei modelli globali, terremo in considerazione anche questi fenomeni, con il fine di rendere la scena il più realistica possibile.

I modelli visti finora non vengono scartati ma sono alla base dei modelli globali: un esempio è considerare un oggetto della scena colpito da un raggio, come sorgente luminosa per un oggetto secondario.

Le tecniche di illuminazione globale sono solitamente molto lenti ma sono in grado di produrre immagini molto realistiche.

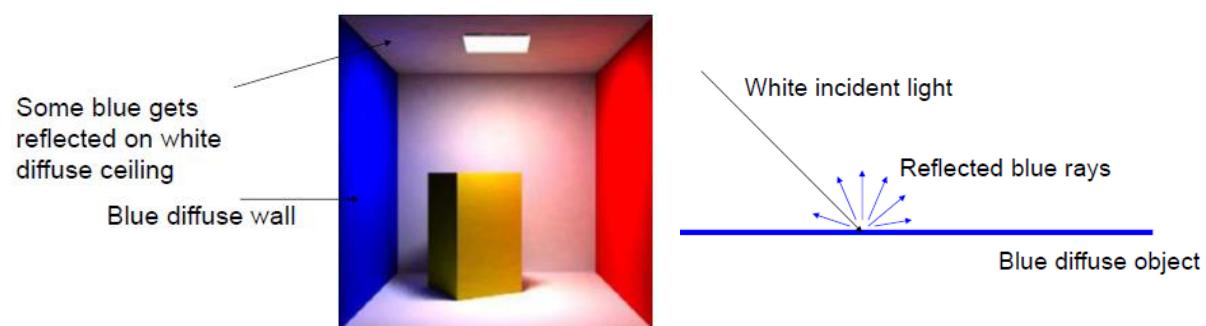
Le tecniche principali sono:

- (1) *radiosity* (che ha come modello locale di base quello Lambertiano)
- (2) *ray tracing* (che ha come modello locale di base quello di Phong).

### (1) Radiosity

Ricordiamo che nel modello Lambertiano, l'intensità della luce diffusa dipende dall'angolo  $\theta$  che si forma fra la normale alla superficie e il raggio incidente ( $\theta = 0 \rightarrow \cos \theta = 1$ ,  $\theta = \frac{\pi}{2} = 90^\circ \rightarrow \cos \theta = 0$ ).

La **radiosity** modella la mutua informazione fra superfici con proprietà diffuse, considerando quindi anche le superfici che non sono intercettate direttamente da un raggio luminoso.

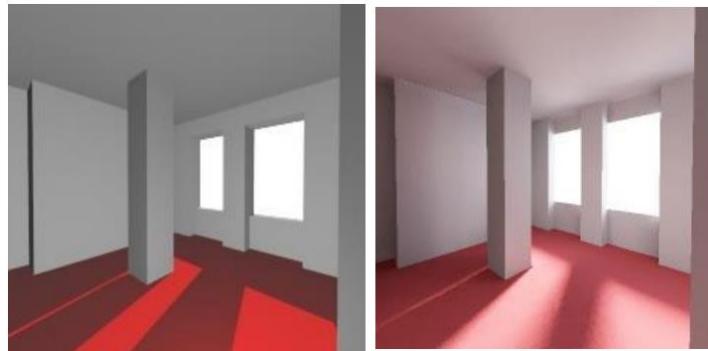


Il modello locale usato dalla radiosity è il modello Lambertiano: quando una superficie diffusiva viene colpita da un raggio luminoso, questa diffonde la luce in ogni direzione, con intensità proporzionale all'intensità del raggio incidente e dall'angolo formato fra la normale alla superficie e il raggio incidente. Il colore della luce riflessa dipende dal colore del raggio incidente e dal colore della superficie.

La radiosity migliora il realismo della scena, rendendo le superfici meno piatte/uniformi.

Inoltre agisce positivamente quando la posizione degli oggetti rispetto alle sorgenti luminose crea delle ombre nella scena: le ombre non devono risultare nette ma essere sfumate, date le caratteristiche di diffusione degli oggetti della scena, e la radiosity è il meccanismo perfetto per farlo.

Confronto fra un'immagine creata senza radiosity (sulla sinistra) e con radiosity (sulla destra).



Questo approccio viene spiegato con un esempio: una stanza illuminata da un raggio di sole che passa attraverso una finestra.

Usando un modello di illuminazione globale:

- il sole viene modellato con una luce puntiforme
- include una luce d'ambiente
- le ombre sono troppo nette (sharp)
- le superfici sono illuminate troppo uniformemente.

Con l'illuminazione globale della radiosity:

- per simulare la sorgente luminosa si usa una immagine del cielo posta come texture sul piano della finestra
- l'intera stanza è illuminata e visibile, anche quelle superfici lontane dal sole
- le ombre sono morbide (soft)
- il cambiamento di luminosità sul muro di sinistra è sottile
- il soffitto è rosa chiaro.

L'algoritmo della radiosity, per semplicità, considera tutte le superfici come se fossero perfettamente diffuse (Lambertiane), non trasparenti né riflessive.

Per prima cosa le superfici vengono suddivise in un numero finito di quadrilateri o triangoli (che prendono il nome di **patch**), ad ognuno dei quali è associata una costante di diffusione.

L'ammontare dell'energia diffusa da ogni patch può essere calcolato sulla base del coefficiente di diffusione della patch e dall'energia che la patch considerata riceve da tutte le altre patch della scena.

Le energie ricevute dipendono dall'angolo fra le normali delle coppie di patch ( $p_1, p_k$ ) dove  $p_1$  è la patch considerata e  $p_k$  è la  $k$ -esima altra patch; il valore calcolato confrontando le coppie prende il nome di fattore di forma (cioè la frazione di energia emessa dalla singola  $p_k$  che influisce sulla patch  $p_1$ ), e tiene in considerazione anche l'angolo del raggio incidente sulle patch  $p_k$  (influente visto che usiamo il modello di Lambert).

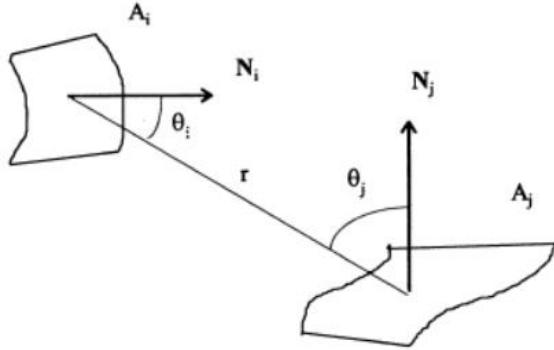
Fatto il primo passo per calcolare l'energia diffusa dalla patch  $p_1$  è necessario calcolare l'influenza che ha l'energia emessa da questa sulle altre patch.

Ovvero il valore della radiosity  $M_i$  per l' $i$ -esima patch (misurata in energia per unità unitaria) è:

$$M_i = M_i^E + \rho_{e,i} \sum_{j=1}^n (F_{ij} \cdot M_j)$$

dove:

- $M_i$  è l'output della patch  $i$
- $M_i^E$  è la diffusione dovuta a una sorgente luminosa
- $\rho_{e,i}$  è il coefficiente di diffusione della patch
- $n$  è il numero totale di patch nella scena
- $F_{ij}$  è il fattore di forma che descrive la relazione geometrica fra la patch  $i$  e quella  $j$
- $M_j$  è l'energia diffusa dalla patch  $j$ .



Come viene calcolato matematicamente il form factor?

Esso dipende dalla presenza di ostacoli fra due patches, dalla loro dimensione e dal relativo orientamento, dalla distanza e dalla forma.

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos \theta_i \cdot \cos \theta_j}{\pi \cdot r^2} v_{ij} \quad dA_j \quad dA_i$$

dove  $v_{ij}$  è il coefficiente di visibilità fra la patch  $A_i$  e quella  $A_j$ , che vale 1 se le superfici si vedono, 0 altrimenti:

$$v_{ij} = \begin{cases} 1 & \text{se } A_i \text{ e } A_j \text{ possono vedersi a vicenda} \\ 0 & \text{altrimenti} \end{cases}$$

Per calcolare la radiosity di tutte le patch della scena è necessario calcolare le matrici di  $n^2$  fattori di forma, riuscendo poi a calcolare l'output della patch i come prodotto matriciale:

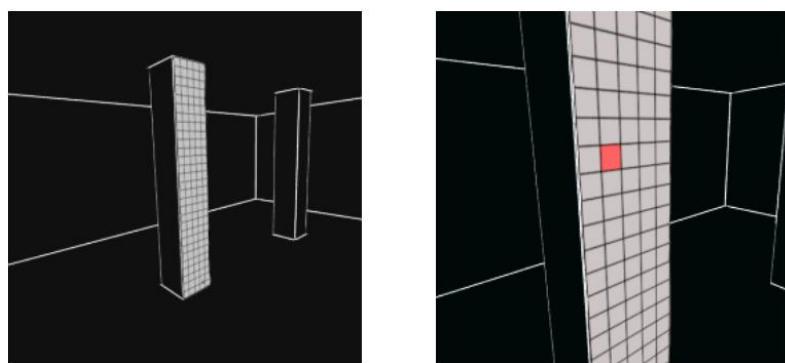
$$M_i - \rho_{i,e} \cdot \sum_{j=1}^n (F_{ij} \cdot M_j) = M_i^E \quad (I - \rho_e F)M = M^E$$

o:

$$\begin{bmatrix} 1 - \rho_{e,1}F_{1,1} & -\rho_{e,1}F_{1,2} & \dots & \rho_{e,1}F_{1,n} \\ \rho_{e,2}F_{2,1} & 1 - \rho_{e,2}F_{2,2} & \dots & -\rho_{e,2}F_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_{e,n}F_{n,1} & -\rho_{e,n}F_{n,2} & \dots & 1 - \rho_{e,n}F_{n,n} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} M_1^E \\ M_2^E \\ \vdots \\ M_n^E \end{bmatrix}$$

Al posto di effettuare questo calcolo possiamo usare un modello iterativo come Gauss-Seidel, in cui si inizializza il vettore  $M$  (di output) ad un valore casuale (tutti 0) e ad ogni passo si migliora il valore del vettore, raffinandolo a partire dai valori precedenti.

Dopo un numero sufficiente di iterazioni il vettore  $M$  contiene valori molto simili alla soluzione ottimale, in quanto il metodo converge verso tale soluzione.



Suddivisione delle patch sulla superficie considerata

Consideriamo ora un certo numero di patch che costituiscono la nostra scena (pavimento, colonne, finestre): alcune di queste sono classificabili come patch che emettono energia propria (patch delle finestre) altre invece non emettono energia (per quest'ultime la componente  $M_e$  dell'equazione della radiosity è nulla).

Pensiamo al primo passo del nostro processo iterativo considerando che ci saranno i vari tipi di patch nella scena.

Per calcolare la quantità di energia luminosa emessa da ogni patch supponiamo di porci dal punto di vista della patch stessa, come se collocassimo una telecamera sulla patch, e calcolare cosa la patch vede all'interno della scena.

Se consideriamo una patch non viene colpita dai raggi emessi dalla sorgente luminosa essa non riceverà alcun tipo di energia luminosa e quindi sarà nera.

Se invece consideriamo una patch che viene colpita dai raggi emessi dalla sorgente luminosa, essa riceverà una parte di luce diretta e quindi sarà in parte (o in tutto) illuminata.

Queste considerazione e i relativi calcoli vengono effettuati per tutte le patch in modo da aggiungere i vari contributi luminosi alla sommatoria globale.

Avendo  $n$  patch è necessario fare  $n$  considerazione e calcoli di questo tipo (uno per patch).

Terminate le iterazioni su tutte le patch si sommano tutti i singoli contributi luminosi.

Dopo la prima iterazione, avrò come risultato un'illuminazione diretta cioè come se avessi applicato soltanto il modello di illuminazione locale, considerando solo l'interazione diretta fra sorgente luminosa e superfici.

Per questo motivo ci sarà un gran numero di patch completamente nere e avrò un effetto di "blocchettatura" dovuto al fatto che abbiamo un numero di patch limitato.

Supponiamo ora di iniziare il secondo passo di iterazione: riconsideriamo le patch analizzate nel passo precedente che non vedevano nulla perché non colpite dalla sorgente: ora alcune saranno illuminate perché "vedranno" le patch illuminate direttamente dalle sorgenti di cui abbiamo calcolato le equazioni di illuminazione all'iterazione precedente.

Così otterrò una nuova illuminazione con le patch direttamente esposte alla luce e quelle che ricevono luce indiretta.

Nel terzo passo rifacciamo la stessa cosa e, di nuovo, patch che prima non vedevano niente ora saranno illuminate indirettamente da patch a loro volta illuminate indirettamente.

Possiamo andare avanti in questo modo fin quando il nostro metodo non converge cioè fin quando, il risultato ottenuto al passo  $m$  è molto simile a quello ottenuto nel passo  $m + 1$ .

Come detto sopra, l'oggetto viene classificato come oggetto che emette luce (sorgente) o come oggetto che non emette luce.

La quantità di energia luminosa emessa da  $M_i^E$  sarà indicata con `emission`.

Quando una luce colpisce una superficie, parte della sua energia viene assorbita e parte viene riflessa.

Il rapporto tra la quantità di energia incidente ed energia diffusa rappresenta il *coefficiente diffusivo* già menzionato sopra e sarà indicato come `reflectance`.

In ogni fase dell'algoritmo, dobbiamo tener conto di due quantità: la quantità di luce `incidente` e la quantità di luce `trasmessa`:

```
Incident = sum of all light that a patch can see  
transmitted = (incident*reflectance) + emission
```

La patch può essere descritta usando una struttura:

```
structure PATCH
    emission
    reflectance
    incident
    transmitted
end structure
```

Algoritmo :

```
load scene
divide each surface into equal sized patches

initialise patches:
for each Patch in the scene
    if this patch is a light then
        patch.emission = some amount of light
    else
        patch.emission = black
    end if
    patch.transmitted = patch.emission
end for

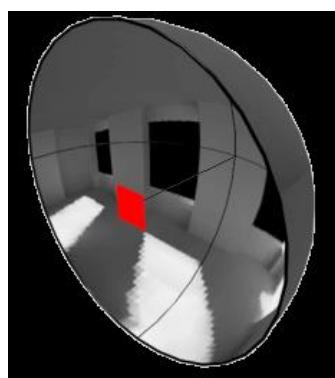
Passes_Loop:
each patch collects light from the scene
for each Patch in the scene
    render the scene from the POV of this patch patch.incident = sum of incident
    light in rendering (see Calc_Incident_Light)
end for

calculate transmitted light from each patch:
for each Patch in the scene
    I = patch.incident
    R = patch.reflectance
    E = patch.emission
    patch.transmitted = (I*R) + E
end for

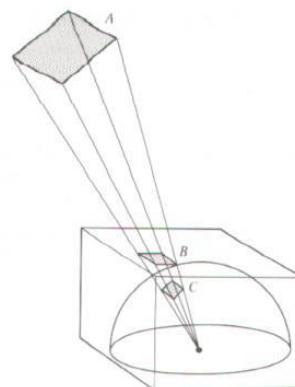
Have we done enough passes?
    if not then goto Passes_Loop
```

Quando calcolo ogni passo dal punto di vista di ogni patch, considero la patch corrente come se fosse al centro di un'emisfera e guardo che cosa è presente in quell'emisfera.

L'emisfera è una forma computazionalmente complessa, perciò possiamo approssimarla considerando un emicubo per determinare la quantità di luce incidente alla patch.



Emisfera



Emicubo

Come renderizziamo sull'emicubo?

Possiamo pensare di suddividere il nostro emicubo in 5 diversi piani immaginari: in realtà quindi dobbiamo fare 5 rendering per ogni patch.

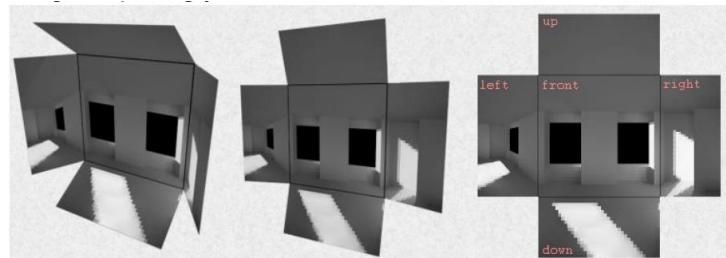
La telecamera viene posizionata sulla patch in direzione della normale alla superficie quindi faremo un render frontale, poi ruotiamo la telecamera di 90° gradi a destra e facciamo il render di destra, poi di sinistra, di sopra e di sotto.

Se un oggetto però si trova su uno spigolo dell'emicubo avremo vari contributi nei render che coinvolgono quello spigolo: perciò in qualche modo è necessario attenuare il contributo del rendering in prossimità degli spigoli dell'emicubo.

Questo viene fatto usando una maschera di compensazione: il risultato dei 5 render della patch lo moltiplichiamo per la maschera compensativa.

Dobbiamo tenere conto anche del fatto che se un oggetto si trova perfettamente di fronte alla patch (cioè in direzione della normale) allora il contributo ricevuto e poi diffuso sarà maggiore rispetto al caso in cui si trovi in posizione angolata (lo sappiamo tenendo conto del modello Lambertiano).

Usiamo perciò un'altra maschera compensativa in modo che il render frontale abbia un peso maggiore rispetto agli altri 4.

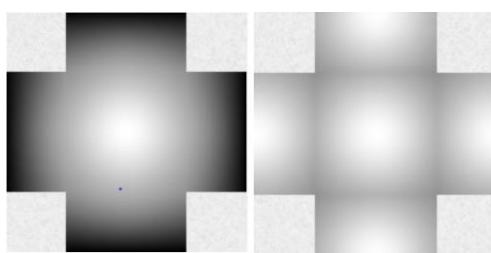


Esempio di emicubo

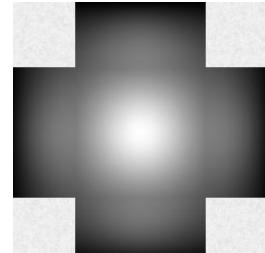
Combinando le due maschere compensative descritte, ottengo una terza maschera complessiva che posso utilizzare direttamente con i 5 render per ottenere il render finale della singola patch.

L'algoritmo per il calcolo del modello di radiosity viene precalcolato per l'intera scena, in questo modo l'utente che si muove nella scena può apprezzare la corretta illuminazione in tutta la stanza.

Però se ci muoviamo dal punto iniziale non verremmo considerati come nuovo elemento riflettente o diffusivo, la radiosity è calcolata all'inizio considerando tutti gli elementi come statici.



Maschere di compensazione per la riduzione dell'intensità negli angoli e l'incremento dell'intensità sulla faccia frontale.

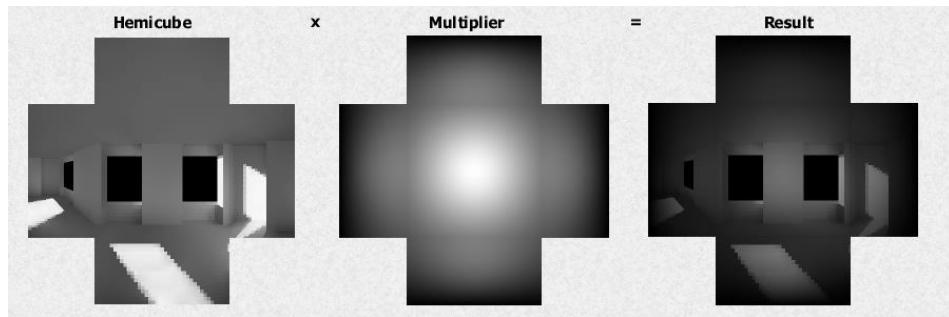


Combinazione (moltiplicazione delle due precedenti maschere).

Terminato l'algoritmo possiamo rappresentare correttamente le patch su cui abbiamo lavorato usando due texture: la prima rappresenta la tessitura originale della patch a prescindere dall'illuminazione, la seconda contiene le caratteristiche calcolate dell'illuminazione.

Le due texture vengono combinate per ottenere il risultato finale.

L'effetto della maschera risultante è:



## Ray casting

Un algoritmo antesignano del *ray tracing* è quello di **ray casting**.

In questo metodo di rendering viene calcolata, *per ogni pixel*, l'intersezione con gli oggetti nella scena di un raggio che parte dall'occhio e attraversa il pixel in esame: le caratteristiche e l'illuminazione del punto di intersezione del primo oggetto incontrato a partire dall'occhio determinano il colore da attribuire al pixel (cioè lo *shading*, calcolato con algoritmi di illuminazione), senza proseguire con ulteriori riflessioni. In particolare, l'illuminazione è quella dovuta unicamente alle varie sorgenti di luce e non si considerano riflessioni.

Sostanzialmente l'algoritmo deve calcolare intersezioni tra rette e oggetti, questi ultimi tipicamente specificati da una equazione implicita.

L'algoritmo è costituito da due cicli nidificati: quello esterno scorre i pixel dell'immagine da produrre, mentre quello interno percorre tutti gli oggetti nella scena valutandone i punti di intersezione con il raggio condotto dal pixel in esame (il calcolo dell'intersezione è un processo più oneroso che fare la scan conversion<sup>10</sup> di un poligono, approccio usato nell'algoritmo dello z-buffer).

Per maggiore efficienza e per evitare il calcolo di molte intersezioni vuote, si sono sviluppati algoritmi che calcolano l'intersezione solo con gli oggetti che hanno probabilità di intersecare il raggio, usando strutture dati di appoggio, quali ad esempio un **octree**.

```
Per ogni pixel P
    Per ogni oggetto O
        Se (il raggio attraverso P interseca O AND
            l'intersezione è più vicina di quella calcolata fino ad ora)
            Calcola lo shading di O nel punto di intersezione; [A]
            Aggiorna il colore di P;
        Fine per ogni oggetto O
    [B]
Fine per ogni pixel P
```

Nel punto dell'algoritmo denotato [A] si ha il calcolo dello shading del pixel, che è tipicamente un processo computazionalmente oneroso: nel caso si trovi un'intersezione più vicina rispetto ad una calcolata in precedenza, lo shading deve essere rieseguito, in quanto quello fatto in precedenza deve essere sostituito.

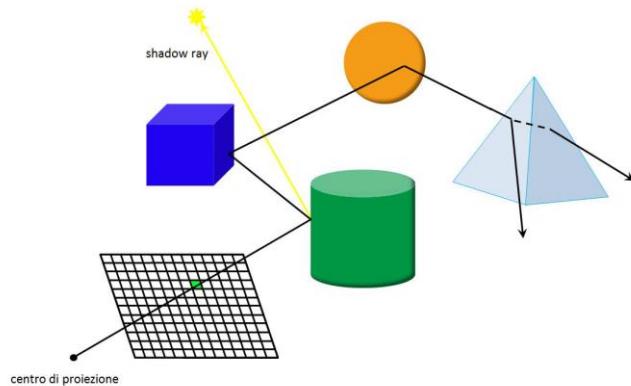
Si vede quindi che tale struttura è inefficiente: si può arrivare quindi al *deferred shading* che nel punto [A] tiene traccia solo del punto di intersezione più vicino per il pixel in esame (salvando texture, colore e vettore normale), e al termine del ciclo su tutti gli oggetti e prima di passare al pixel successivo effettua il calcolo dello shading del pixel (nel punto denotato [B]), facendo così un uso efficiente delle risorse di calcolo.

Per effettuare il rendering non serve memorizzare lo stato di più di un pixel alla volta, evitando la necessità di un buffer.

<sup>10</sup> La **scan conversion** di un poligono consiste nel determinare quali siano i pixel di un'immagine che fanno parte della proiezione del poligono su di essa.

## (2) Ray tracing

Il **ray tracing** parte dall'idea del ray casting e migliora la qualità del risultato proseguendo la traiettoria di ogni raggio facendolo proseguire nelle riflessioni e rifrazioni in base agli oggetti incontrati nella scena. Ovviamente in questo modo si tiene conto della profondità (come nel ray casting) ma anche di ombre, sorgenti di luce e oggetti trasparenti.



L'algoritmo di **recursive ray tracing** (proposto da Whitted nel 1979) prosegue il raggio ad ogni intersezione con un oggetto, secondo le regole della riflessione e della rifrazione, ripetendo il processo sul nuovo raggio che si crea (due raggi nel caso di rifrazione), da cui il nome *ricorsivo*.

Per creare mediante recursive ray tracing un'immagine bidimensionale di una scena modellata matematicamente occorre definire un **punto di vista** (POV, centro di proiezione) e un piano di proiezione della scena costituito da una matrice di pixel.

Dal centro di proiezione vengono fatti partire dei raggi, uno per ogni pixel e, nella prima accezione, passante per il suo centro: quando il **raggio primario** incontra un oggetto viene **riflesso** (ed eventualmente **rifratto**) generando quello che viene chiamato **raggio secondario** (si hanno due raggi secondari in caso di riflessione e rifrazione).

Il **raggio riflesso** è determinato in base alla legge di riflessione, per cui è complanare al raggio incidente e alla normale alla superficie nel punto di incidenza, ed ha un angolo di riflessione uguale a quello di incidenza.

Il **raggio rifratto**, nel caso di materiale trasparente, segue la **legge di Snell** per cui detti  $\eta_1$  e  $\eta_2$  gli indici di rifrazione dei materiali a contatto attraversati dal raggio, allora gli angoli di incidenza  $\theta_1$  e di rifrazione  $\theta_2$  rispetto alla normale nel punto di attraversamento dei due materiali sono calcolati come:

$$\eta_1 \sin \theta_1 = \eta_2 \sin \theta_2$$

Ogni raggio secondario viene nuovamente intersecato con il primo oggetto che incontra e ne vengono calcolati i raggi riflesso e rifratto, producendo raggi di ordine tre.

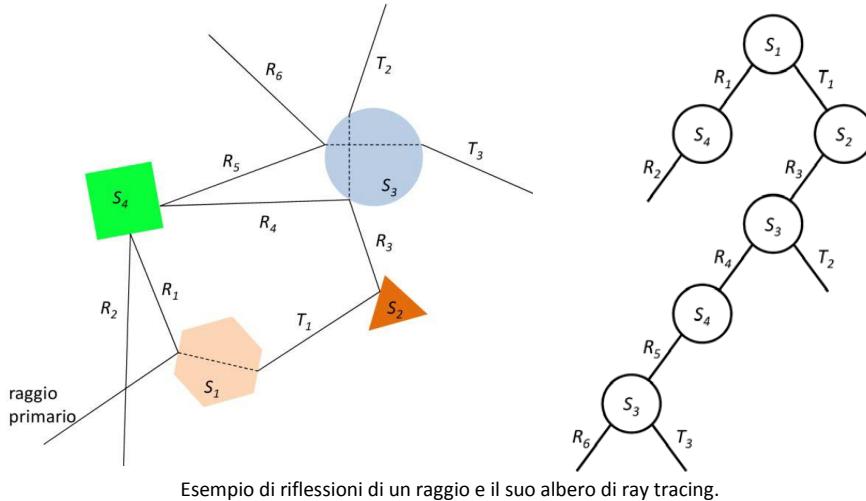
Il processo viene ripetuto, creando un **albero di ray tracing** (da ogni nodo che rappresenta un raggio discendono al massimo due nodi figli che rappresentano rispettivamente il raggio riflesso e quello rifratto).

La profondità dell'albero deve essere limitata mediante un parametro dell'algoritmo.

Inoltre il raggio termina di essere riflesso (e rifratto) nel caso intersechi una sorgente di luce non riflettente.

L'intensità e il colore da attribuire al pixel associato al raggio primario sono determinati a partire dalle foglie dell'albero di ray tracing risalendo fino alla radice dell'albero, sommando i contributi dei nodi figli, attenuati in funzione della distanza dal punto rappresentato dal nodo padre, a quello del nodo padre: quando questo processo giunge al nodo radice si hanno intensità e colore da attribuire al pixel attraversato dal raggio primario.

- Se il raggio non attraversa alcun oggetto allora al pixel vengono assegnati intensità e colore dello sfondo (o un valore di default attribuito allo sfondo).
- Se il raggio primario interseca una sorgente di luce non riflettente, al pixel può essere assegnata l'intensità della sorgente.



In figura (a) è riportato un semplice esempio di riflessioni e rifrazioni di un raggio in una scena composta da quattro oggetti.

Gli oggetti  $S_1$  e  $S_3$  sono trasparenti per cui si creano un raggio riflesso e uno rifratto (per semplicità nell'illustrare il procedimento si considera solo il secondo raggio rifratto, cioè quello uscente dall'oggetto e senza alcuna riflessione all'interno dell'oggetto stesso).

Nella figura (b) viene riportato l'albero di ray tracing per il raggio.

L'intensità e il colore di ogni intersezione *dovuti all'illuminazione* sono calcolati sommando il contributo di quella di ambiente, di quella Lambertiana dovuta alla diffusione della luce proveniente dalle sorgenti di luce, e di quella speculare dovuta alla riflessione della luce, anche in questo ultimo caso delle sorgenti luminose.

Il raggio può quindi avere diversi comportamenti:

- **il raggio non colpisce nulla e va all'infinito** (viene stoppato con  $t$  sufficientemente grande e viene assegnato al pixel attraversato il colore di background)
- **il raggio colpisce un oggetto che non riflette ma diffonde solo** (il raggio viene ignorato e al pixel viene assegnato un colore in base alle informazioni ottenute con la radiosity, che si occupa dei modelli diffusivi)
- **il raggio colpisce un oggetto riflettente**: in questo caso il raggio viene geometricamente riflesso sulla superficie e si continua a tracciare il raggio riflesso.  
Quello che chiamiamo qui raggio riflesso è in realtà quello incidente, in quanto tramite il ray tracing percorriamo i raggi al contrario dall'osservatore alla sorgente<sup>11</sup>.  
Il raggio può essere fermato dopo un numero fissato di riflessioni
- **il raggio colpisce un oggetto rifrattante** (trasparenti o semi trasparenti): i raggi proseguono il loro percorso in base alle caratteristiche fisiche del materiale attraversato secondo le **legge di Snell**.

---

<sup>11</sup> Nel ray tracing si segue il percorso inverso dei raggi per una questione di convenienza: solo una minima parte dei raggi emessi dalla sorgente e riflessi dai vari oggetti della scena giunge all'osservatore. Perciò conviene partire direttamente da questi raggi e fare il percorso inverso.

Per questo motivo prende il nome di Backward ray tracing.

L'algoritmo è molto complesso e oneroso, perciò viene sottoposto a numerose ottimizzazioni, le quali lavorano principalmente sulle fasi di calcolo delle intersezioni fra i raggi e gli oggetti della scena:

- **bounding volume**: volumi semplici (cubi o parallelepipedi) che racchiudono un oggetto della scena.  
Il test preliminare valuta se il raggio interseca il bounding volume: se si, si procede con l'intersezione fra il raggio e il vero oggetto; in caso contrario si evitano i calcoli complessi e si scarta l'oggetto
- **OcTree fisico**: con il quale la scena viene decomposta in 8 sottoscene valutabili singolarmente (utile anche per fare cache quando cambiano solo poche parti della scena al variare del tempo) le quali possono essere ridivise in modo ricorsivo
- **OcTree del piano immagine**: analogo a quello fisico per il piano immagine.

Quando un raggio colpisce una superficie, può generare fino a 4 nuovi tipi di raggio:

- (a) **pixel ray**: raggio che parte dall'osservatore, attraversa il piano immagine e interseca un oggetto.
- (b) **shadow ray**: raggio che congiunge le sorgenti luminose con il punto colpito dal pixel ray.  
Se l'oggetto non colpisce la sorgente ma altri oggetti sappiamo che il punto è in ombra perché nel cammino tra il punto colpito dal pixel ray e la sorgente luminosa, c'è un altro oggetto.
- (c) **reflection ray**: raggio riflesso che parte dal punto colpito dal pixel ray (se l'oggetto riflette il raggio cioè se è lucido).
- (d) **transparency ray**: raggio (trasmesso) che attraversa l'oggetto a partire dal punto colpito dal pixel ray (se l'oggetto è trasparente o semitrasparente).

In generale, per trovare il colore della luce che lascia una superficie, è necessario considerare la combinazione di diversi tipi di luce incidente su quella superficie.

Il colore della luce emessa è funzione di:

- la luce proveniente dalle sorgenti
- la luce riflessa dagli oggetti (riflessioni)
- la luce che passa attraverso gli oggetti (rifrazioni).

È possibile determinare i colori della luce riflessa e trasmessa considerando le proprietà degli oggetti da cui sono originate.

## Shadow rendering

---

Se è necessario creare delle ombre possiamo usare lo *stencil buffer* nelle tecniche d:

- volumi d'ombra
- mappe d'ombra.

In entrambe le tecniche, gli oggetti della scena sono generalmente suddivisi in tre categorie:

- oggetti che fanno ombra
- oggetti che ricevono ombra
- oggetti interamente esclusi quando si presentano ombre.

Anche le luci sono taggiate per indicare se devono oppure no generare ombre.

## VOLUMI D'OMBRA

Immaginiamo di spostarci dal punto di vista dell'oggetto a quello della luce.

A questo punto abbiamo la possibilità, utilizzando le tecniche di intersezione fra geometrie semplici, di creare il volume d'ombra che l'oggetto crea: tracciamo dei raggi verso il bordo dell'oggetto che partono dalla luce e andranno a colpire l'oggetto.

Il risultato sarà effettivamente un volume d'ombra visto come un certo numero di geometrie elementari che vengono combinate fra loro.

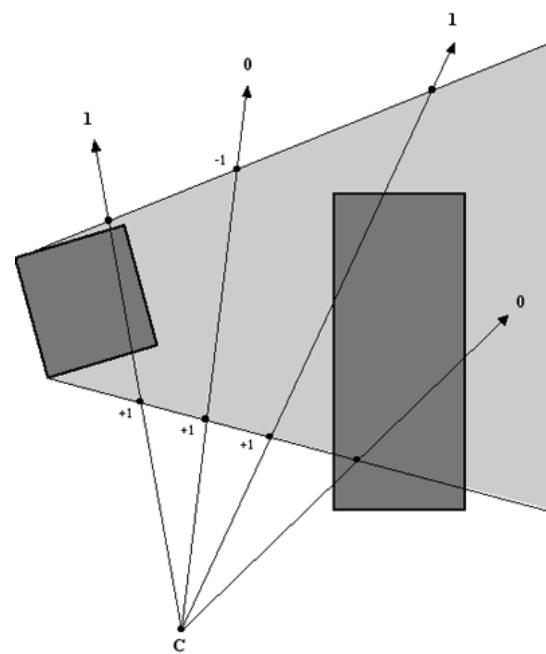
A questo punto affianchiamo al volume d'ombra lo *stencil buffer* che memorizza le informazioni necessarie per determinare se un oggetto è in ombra: ci rimettiamo nel punto di vista dell'osservatore e tracciamo un raggio verso la primitiva dell'oggetto di cui vogliamo sapere se è in ombra oppure no.

Questo raggio potrà dunque andare a intersecare il volume di ombra in due modi:

- in modalità entrante
- in modalità uscente<sup>12</sup>.

Consideriamo ora i raggi che partono dall'osservatore, attraversano la griglia di pixel e entrano nella scena: ogni volta che un raggio entra nel volume d'ombra incrementiamo il valore nello *stencil buffer* (inizializzato a 0), se esce il valore nello stencil buffer viene decrementato.

Quando finisco quest'operazione per tutti i pixel lo *stencil buffer* contiene valori a zero o diversi da zero; quelli diversi da zero mi dicono quali pixel dovrà ombreggiare.



## MAPPE D'OMBRA

Un test di profondità per frammento viene eseguito dal punto di vista della luce anziché dal punto di vista della telecamera.

La scena è renderizzata in due fasi:

1. si genera una texture di mappa d'ombra renderizzando la scena dal punto di vista della sorgente luminosa e si salva salvando il contenuto del depth buffer
2. la scena viene renderizzata come al solito e la mappa d'ombra viene utilizzata per determinare se ogni frammento è in ombra oppure no.

A ogni frammento nella scena, la mappa d'ombra ci dice se la luce viene ostruita da una geometria più vicina alla sorgente luminosa.

Una mappa d'ombra contiene solo informazioni di profondità.

<sup>12</sup> Per capire se il raggio dell'osservatore entra o esce dal volume d'ombra lo confrontiamo con la normale alla superficie intercettata dal raggio: se il verso del raggio è opposto rispetto alla normale uscente del volume d'ombra significa che sto entrando nel volume d'ombra, se invece attraverso la superficie uscente del volume d'ombra i versi saranno concordati.

Per suo rendering:

- si applica una proiezione prospettica per le sorgenti luminose puntiformi
- si applica una proiezione ortografica per le sorgenti luminose direzionali.

La scena viene interpretata come al solito dal punto di vista della fotocamera.

Per ogni vertice di ogni triangolo viene calcolata la sua posizione nello spazio luminoso (utilizzando lo stesso "spazio di visualizzazione" utilizzato nella generazione della mappa d'ombra).

Queste coordinate possono essere interpolate attraverso il triangolo, proprio come qualsiasi altro attributo vertice.

Questo dà la posizione di ogni frammento nello spazio luminoso.

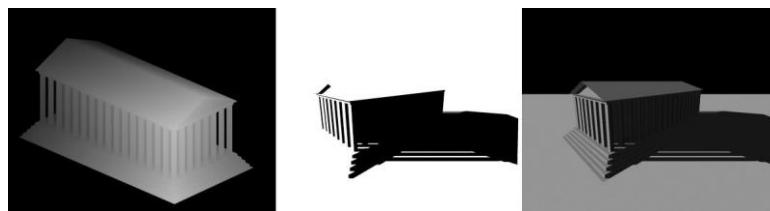
Per specificare se un determinato frammento è in ombra o no, convertiamo le coordinate  $(x, y)$  del frammento di luce in coordinate di texture  $(u, v)$  all'interno della mappa d'ombra.

Poi confrontiamo la coordinata  $z$  dello spazio della luce con la profondità memorizzata nel corrispondente texel nella mappa d'ombra.

Se lo spazio di luce  $z$  del frammento è più lontano dalla luce rispetto al texel nella mappa d'ombra, allora deve essere occluso da un altro pezzo di geometria più vicino alla sorgente luminosa (è in ombra).

Se lo spazio di luce  $z$  del frammento è più vicino alla sorgente luminosa rispetto al texel nella mappa d'ombra, allora non è occluso e non è in ombra.

Il colore del frammento può essere regolato di conseguenza.



## Ambient occlusion

L'**ambient occlusion** è un metodo di shading che contribuisce a conferire realismo ai modelli di riflessione locale in quanto tiene conto dell'attenuazione luminosa in prossimità di volumi occlusi.

Diversamente da metodi locali come il Phong shading, l'ambient occlusion è un metodo globale, cioè l'illuminazione di ogni punto è funzione della geometria della scena.

Ad ogni modo è un'approssimazione grezza dell'intera illuminazione globale. L'aspetto generato dalla sola ambient occlusion è simile a quello di un oggetto in un giorno nuvoloso.

Solitamente l'ambient occlusion viene calcolata tracciando raggi in ogni direzione dalla superficie.

I raggi che raggiungono lo sfondo o il "cielo" aumentano la luminosità della superficie, mentre quelli che intercettano un altro oggetto non aggiungono alcuna illuminazione.

Di conseguenza i punti circondati da molte altre geometrie vengono renderizzati in ombra, mentre i punti più liberi da ingombri risultano più chiari.

L'ambient occlusion è legata all'**accessibility shading**, che si basa sulla facilità con cui un certo elemento (polvere, luce ecc...) può raggiungere ogni punto di una superficie (si pensi ad un oggetto poroso in cui i buchi meno accessibili tendono a rimanere coperti di polvere).

Una buona caratteristica di questo metodo di shading è quella di offrire una migliore percezione della forma tridimensionale degli oggetti mostrati.

Questo fatto è riportato dai risultati di esperimenti che dimostrano la superiore resa della profondità prodotta da uniforme illuminazione "sky light" diffusa rispetto alla "direct lighting".



# 8. Stereoscopia

## Elementi della stereoscopia

Nel capitolo sull'audio 3D abbiamo capito come ingannare l'utente facendogli percepire un suono proveniente da una sorgente in movimento.

Allo stesso modo possiamo ingannarne la vista, simulando non solo la profondità ma anche la tridimensionalità degli oggetti della scena, come se non fossero immagini piatte sullo schermo di visualizzazione ma oggetti che da esso fuoriescono o che vi si trovano dietro.

La **stereoscopia**, in termini commerciali prende il nome di **3D** e si basa sull'idea di produrre due viste differenti del mondo, simulando le immagini che derivano dall'occhio destro e da quello sinistro, per poi accoppiarle nel nostro sistema visivo.

Per fare ciò è necessario capire quali sono gli elementi che vengono usati dall'essere umano per riconoscere la tridimensionalità, infatti conoscendoli possiamo ricostruire un sistema visivo che fornisca la sensazione di tridimensionalità.

Iniziamo dunque la carrellata di elementi, partendo da quelli fisici, per proseguire con quelli psicologici.

La percezione umana della profondità è una combinazione di molti fattori:

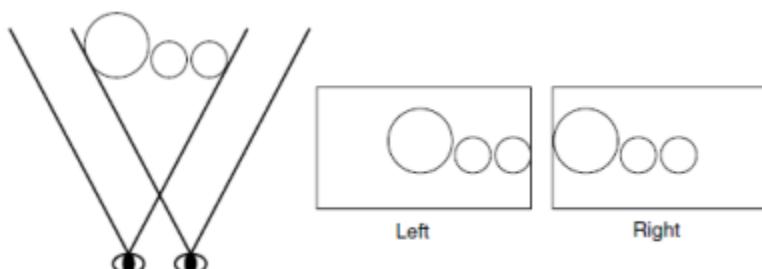
- disparità binoculare
- convergenza
- accomodation
- altre esperienze geometriche e legate alla memoria.

### Binocular disparity

Il primo elemento fisico che valutiamo è la **disparità binoculare**, una sensazione neuronale basata sul fatto che ciascun essere umano è dotato di due occhi posti ad una certa distanza fra loro.

Le immagini che vengono percepite dall'occhio destro e dall'occhio sinistro dell'essere umano sono leggermente diverse fra loro, in particolare risultano traslate in base alla **distanza interoculare**.

L'immagine finale che viene prodotta dal nostro cervello deriva dall'unione delle due percepite, come se fossimo dotati di **visione ciclopica**<sup>13</sup>, che mantiene la sensazione di profondità derivante dalla diversità delle immagini.



Disparità binoculare fra l'occhio destro e sinistro

Nel sistema che miriamo a costruire possiamo simulare tale situazione usando 2 camere leggermente spostate fra loro di qualche centimetro (più generalmente della distanza interoculare media), producendo, tramite i rispettivi *view frustum*, due immagini simili a quelle generate dagli occhi.

Valutando singolarmente le due immagini prodotte dalle camere (o alternativamente dagli occhi) si può notare come gli oggetti lontani mantengono la loro posizione indipendentemente dalla camera con cui li si osserva, mentre quelli vicini si spostano in base alla camera con cui lo stiamo osservando.

<sup>13</sup> Date le due immagini percepite ne viene generata una sola, come se fossimo dotati di un solo occhio centrale come i ciclopi.

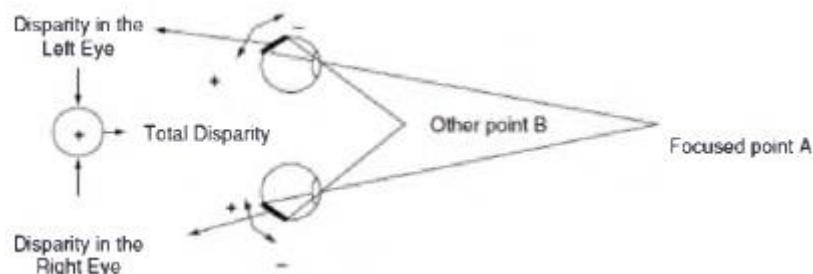
Questa differenza (è la **disparità binoculare**) ci fornisce un'informazione sulla profondità a cui si trova l'oggetto, cioè la distanza che intercorre fra l'oggetto e l'osservatore.

Tale disparità è riconoscibile solo entro i 15 metri, dopo i quali l'immagine rimane ferma, qualsiasi sia la camera con cui la si osserva.

Possiamo notare che se volessimo allungare la soglia dei 15 metri dovremmo avere la testa più grande e gli occhi più lontani fra loro.

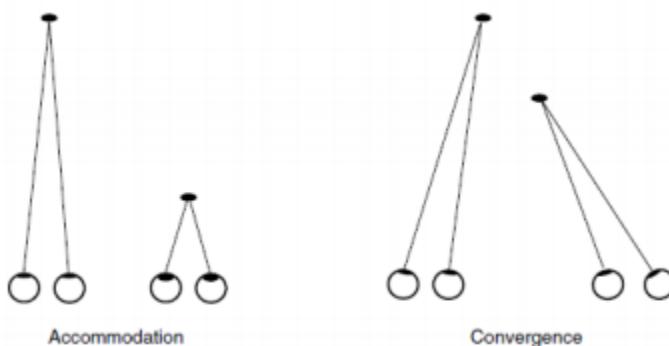
Visto che superati i 15 metri l'informazione data dalla disparità binoculare è nulla è necessario usare altri strumenti.

Una rappresentazione della disparità binoculare:



### Convergencys e accommodation

In aiuto alla disparità possiamo considerare **convergenza** e **accomodazione**, fenomeni muscolari volontari messi in atto dal sistema visivo osservando un certo punto nello spazio.



L'**accommodation** riguarda il fenomeno di compressione e rilassamento della cornea durante la messa a fuoco.

La **convergenza** riguarda invece la rotazione della testa e dei bulbi oculari che convergono verso l'oggetto di interesse: la **convergenza** aumenta se l'oggetto è vicino all'osservatore, diminuisce se è lontano, al punto che i raggi di osservazione (che partono dagli occhi e terminano sull'oggetto osservato) tendono a diventare paralleli.

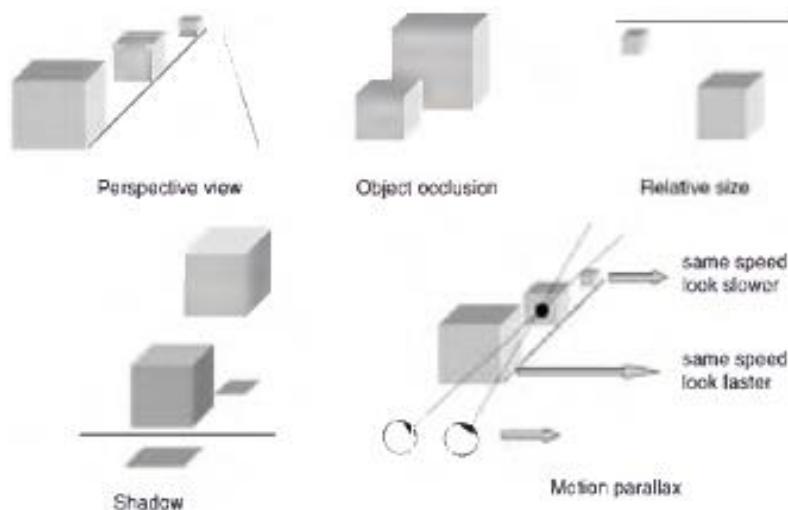
**Accommodation** e **convergenza** portano informazioni aggiuntive sulla profondità di un oggetto ma tali informazioni sono utili solo fra i 5 e i 10 metri di distanza dall'osservatore.

### Psicologia

La **psicologia** è di grande utilità per la comprensione della profondità, ed è usata in particolare per confermare o confutare tramite l'esperienza le informazioni provenienti dai fenomeni fisici visti finora. In molti casi, per la percezione e la comprensione di ciò che stiamo osservando, la psicologia gioca un ruolo più importante delle informazioni che percepiamo tramite fenomeni fisici, e ne sono un esempio le illusioni ottiche.

Elenchiamo ora alcuni aspetti da tenere in considerazione durante la costruzione di un'immagine, per renderla realistica dal punto di vista psicologico:

- **Prospettiva:** le dimensioni degli oggetti cambiano in base alla distanza dall'osservatore
- **Occlusione fra gli oggetti:** oggetti vicini possono impedire la visibilità di oggetti lontani
- **Presenza e corretta configurazione delle ombre**
- **Velocità proporzionale alla distanza:** la velocità degli oggetti viene percepita in modo diverso rispetto alla distanza dall'osservatore (l'aereo sembra muoversi più lentamente dell'auto)
- **Prospettiva atmosferica:** gli strati di atmosfera o di smog rendono gli oggetti lontani più sfocati e meno definiti di quelli vicini
- **Linea dell'orizzonte:** gli oggetti posti sulla linea dell'orizzonte vengono percepiti come più lontani dall'osservatore
- **Dimensioni familiari:** per il confronto fra le dimensioni degli oggetti si fa riferimento alle dimensioni a cui siamo abituati a vedere gli oggetti
- **Livello di dettaglio:** gli oggetti più lontani vengono percepiti come meno dettagliati di quelli più vicini.



Dato che il sistema visivo deve combinare la fisiologia con la psicologia per costruire delle immagini tridimensionali realistiche è importante che gli aspetti elencati vengano rispettati per non creare disorientamento nell'utente.

Alcuni di questi elementi sono più importanti di altri, ma in generale sono aspetti additivi, cioè in cui la bontà del risultato dipende dalla somma dei contributi apportati da ogni elemento.

---

## Elementi di fisiologia della visione

---

### L'occhio: composizione e principio di funzionamento

---

L'occhio è l'organo deputato alla vista.

Le figure 1 e 2 riportano dei disegni esplicativi della sua composizione.

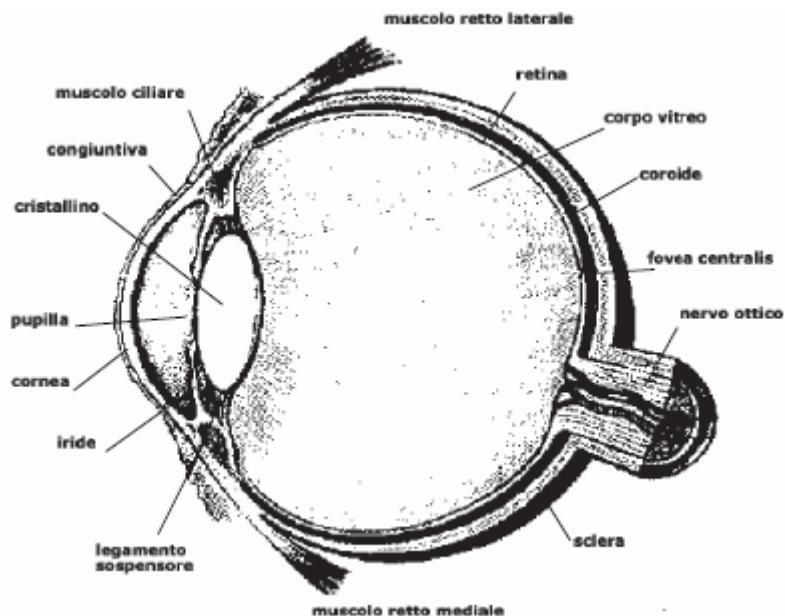


Fig. 1 - Sezione dell'occhio

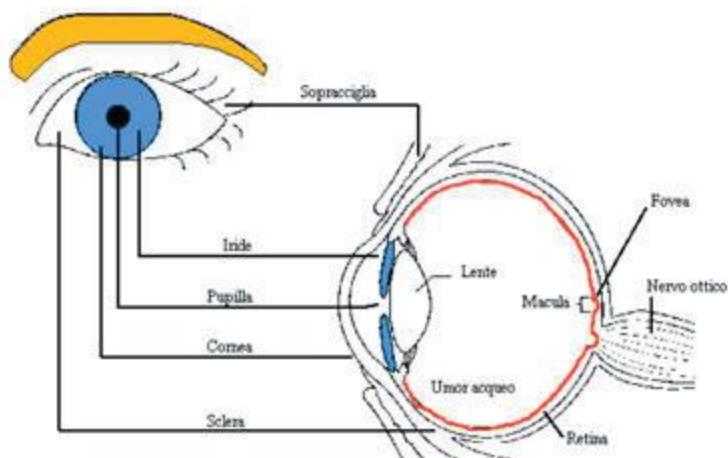


Fig. 2 - Composizione schematica dell'occhio

Esso è composto da un bulbo e da alcuni annessi come i muscoli estrinseci, che servono per effettuare la rotazione del bulbo nella sua sede, e le ghiandole, le vie lacrimali e le palpebre, che servono per la lubrificazione della sclera.

Il bulbo, di forma grossolanamente sferica, è costituito da tre pareti chiamate tonache che, procedendo dall'esterno verso l'interno, sono la sclera, la coroide e la retina; quest'ultima ospita i recettori della luce. L'immagine è focalizzata sulla retina dal sistema di lenti composto dai mezzi diottrici - dall'esterno verso l'interno: la **cornea**, l'**umor acqueo**, il **cristallino** ed il **corpo vitreo**.

Di questi solo il cristallino ha la possibilità di deformarsi, su sollecitazione dei muscoli intrinseci, per ottenere l'esatta focalizzazione dell'immagine sulla retina.

L'**iride** è un diaframma che possiede nella sua parte centrale una soluzione di continuo circolare, la **pupilla**, avente calibro variabile.

In questo modo l'iride regola, in seguito ad un riflesso detto **fotomotore**, l'intensità di luce che raggiunge la retina entro limiti ottimali.

I **fotorecettori** della retina, che si differenziano in **coni** e **bastoncelli**, raccolgono lo stimolo luminoso e lo trasformano in eccitazione nervosa tramite reazioni biochimiche.

Questo stimolo viene poi convogliato, tramite il nervo ottico, ai centri corticali deputati alla visione presenti nel cervello.

## La retina

La retina è composta di vari strati: il più interno di questi rappresenta il **fondo dell'occhio** (*fundus*) ed è a contatto con il corpo vitreo.

Questa parte, di colore rosso come si vede in figura 3, contiene la **macula lutea**, delimitata da un piccolo anello chiaro brillante, e la **fovea centralis**, o **fovea**, indicata dalla freccia.

Nella retina sono presenti due tipi di recettori:

- i **bastoncelli**: sono distribuiti nelle aree periferiche della retina e sono deputati alla visione notturna ed in bianco e nero
- i **coni**: sono destinati alla visione dettagliata (lettura, oggetti distanti) ed alla percezione dei colori.

La fovea, in particolare, è costituita quasi esclusivamente da coni e, grazie alla elevata densità di recettori, è caratterizzata dalla massima acuità visiva.

La sua principale direzione visiva corrisponde al “dritto-davanti”.

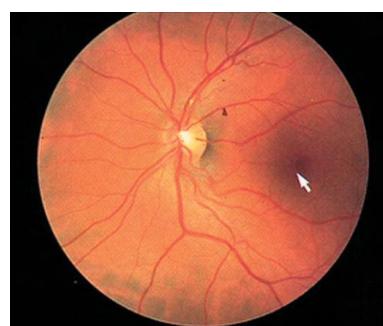


Fig. 3 - Fotografia di una porzione della retina. La freccia indica la fovea

## Fusione sensoriale

Nonostante l'uomo possieda due occhi, non vede doppio grazie al processo di  **fusione (sensoriale)**<sup>14</sup> operato dai centri celebrali preposti alla vista.

Tale processo permette il riconoscimento degli oggetti osservati ed è quindi fondamentale per l'apprendimento e, conseguentemente, per la conoscenza.

L'immagine di ogni punto visibile costituisce un oggetto osservato, chiamato **punto oggetto**, viene proiettata sulla retina in una coppia di punti, uno per ciascun occhio, chiamati **punti retinici**.

Quando si fissa su un oggetto, gli assi ottici oculari convergono, intersecandosi in un particolare punto chiamato **di fissazione**.

La sua immagine proiettata sulle retine, in particolare sulle fovee, si forma su una coppia di punti retinici, uno per ciascun occhio, dai quali originano stimoli nervosi che il sistema della visione “fonde” in un singolo punto.

Per questo motivo tali punti retinici sono chiamati **corrispondenti**; non sono simmetrici da un punto di vista anatomico, ma risultano funzionalmente accoppiati a causa del fenomeno della fusione sensoriale.

Per ogni punto di fissazione esiste una curva, chiamata **oroptero** composta da tutti i punti dello spazio reale per i quali si verifica la fusione.

Ciò si ottiene poiché le loro immagini proiettate sulle retine si formano su coppie di punti retinici corrispondenti, uno per ciascuna retina.

In altre parole, tutti i punti dello spazio reale giacenti sull'oroptero vengono percepiti come punti singoli.

<sup>14</sup> La “fusione motoria” contribuisce a mantenere a livello foveale le due immagini grazie al continuo allineamento degli assi oculari dovuto all’azione della muscolatura estrinseca.

Siccome per riconoscere gli oggetti osservati è necessario che questi siano percepiti come unici, la corteccia celebrale continuamente governa la direzione di osservazione tramite i movimenti degli occhi, affinché le immagini cadano su punti retinici corrispondenti.

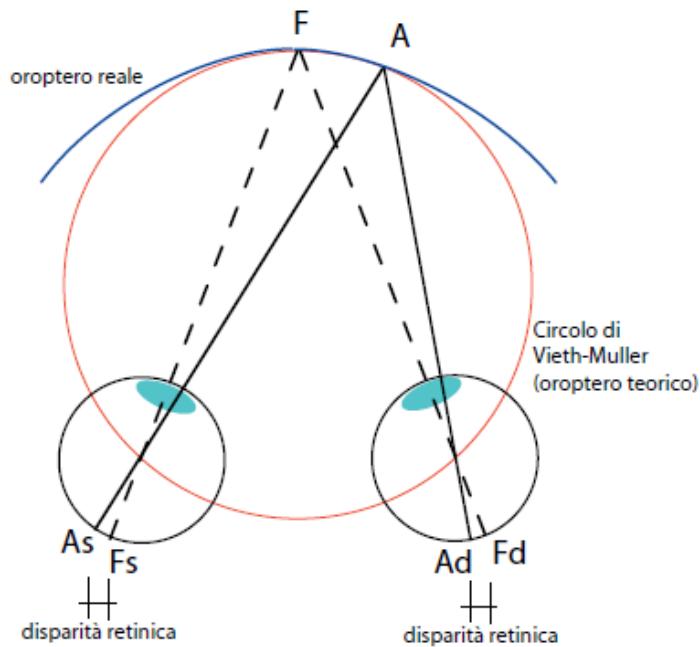


Fig. 4 - Oroptero e punti retinici corrispondenti

Analizzando la figura 4, si nota che il punto di fissazione  $F$  proietta la sua immagine sui punti retinici  $F_s$  e  $F_d$ , giacenti rispettivamente sulla retina sinistra e sulla retina destra.

Siccome i punti retinici associati al punto di fissazione sono punti corrispondenti, questo viene recepito come unico.

Considerando un altro punto giacente sull'oroptero, ad esempio il punto  $A$ , si nota che esso forma la propria immagine su una coppia di punti retinici corrispondenti, che – rispetto ai precedenti – risultano situati internamente nell'occhio destro e esternamente nell'occhio sinistro.

La distanza tra punti retinici, per esempio tra  $F_s$  e  $A_s$  oppure tra  $F_d$  e  $A_d$ , si chiama **disparità retinica**.

Grazie all'accoppiamento dei punti retinici corrispondenti, il punto  $A$  verrà ancora percepito come un unico punto, ma la posizione dei punti retinici  $A_s$  e  $A_d$  rispetto ai punti retinici corrispondenti al punto  $F$ , cioè, rispettivamente,  $F_s$  e  $F_d$ , inducono nell'osservatore la percezione che il punto  $A$  sia situato a destra di  $F^{15}$ .

La fusione non avviene solamente per i punti giacenti sull'oroptero, ma anche per quelli che si trovano in una ristretta zona, chiamata **area di Panum**, che si estende sia davanti che dietro di esso, e che ha una forma simile a quella illustrata in figura 5.

Un punto  $B$  incluso nell'area di Panum viene quindi ancora percepito come unico.

<sup>15</sup> In realtà, anche se in modo non così preciso, la percezione della direzione visiva è ottenuta anche con un occhio solo: si parla di "direzione visiva oculocentrica".

In questo caso il sistema della visione effettua l'analisi della disparità retinica generata, nel solo occhio considerato, dalle immagini dei punti osservati.

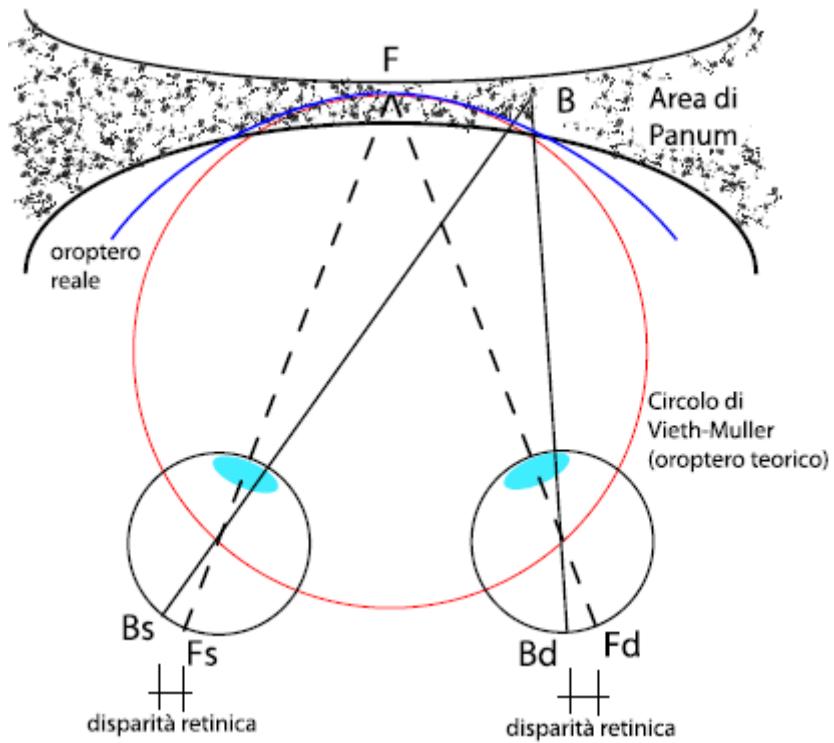


Fig. 5 - Area di Panum

Tuttavia, i centri preposti alla visione, analizzando le disparità retiniche tra le sue immagini sulle retine,  $B_s$  e  $B_d$ , e quelle relative al punto di fissazione  $F$ , rispettivamente  $F_s$  e  $F_d$ , ricavano la percezione di maggiore o minore distanza rispetto al punto di fissazione.

In figura 5, il punto oggetto  $B$ , situato al di là dell'oroptero, è correttamente percepito più distante di  $F$ . I punti oggetto che si trovano al di fuori dell'area di Panum vengono visti come doppi, fenomeno chiamato **diplopia**.

Le immagini di tali punti oggetto si formano su punti retinici che, non essendo accoppiati, si chiamano **disparati** e non portano alla fusione.

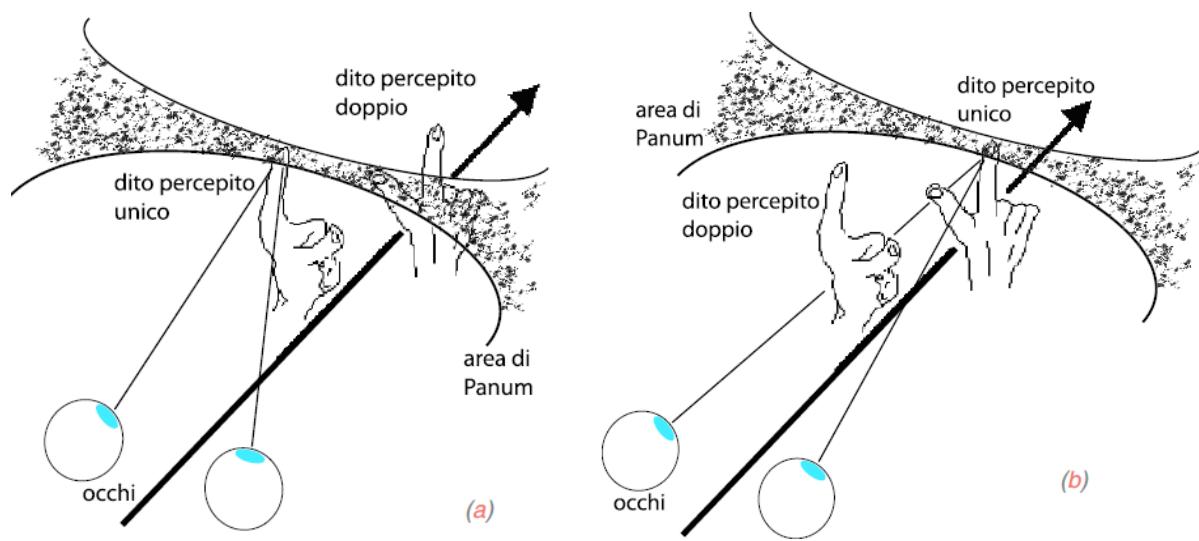


Fig. 6 - Esperimento per riprodurre l'effetto della "diplopia"

Per renderci conto di questo fenomeno possiamo effettuare un semplice esperimento illustrato nelle figure 6a e 6b: mettiamo davanti ai nostri occhi ad una distanza di circa 20 cm. l'indice della mano sinistra.

Se a questo punto noi lo fissiamo lo vedremo singolo; se frapponiamo tra il nostro volto e il dito l'indice della mano destra noi vedremo quest'ultimo come doppio (diplopia crociata) ed identica sensazione avremo se lo posizioneremo tra l'indice della mano sinistra e l'infinito (diplopia omonima).

Cambiando punto di fissazione la diplopia si verifica per altri oggetti, quindi, proseguendo l'esempio, ponendo l'attenzione all'indice della mano destra (senza spostarla), vedremo doppio il dito della mano sinistra.

Se un oggetto percepito come doppio è posto tra gli occhi e l'oroptero parleremo di **disparità crociata** mentre se è posto tra l'oroptero e l'infinito parleremo di **disparità omonima**.

Abitualmente non ci accorgiamo della diplopia cosiddetta **fisiologica** poiché la nostra attenzione è focalizzata sull'oggetto che stiamo osservando e quindi il nostro cervello trascura le immagini doppie.

### **La percezione della profondità, la stereopsi**

---

La percezione della profondità non dipende solamente dalla visione binoculare, ma è frutto di un insieme di elaborazioni mentali e di fenomeni percettivi e fisiologici la cui conoscenza permette di ricreare artificialmente la sensazione di distanza, oppure creare situazioni e ambienti paradossali. Le arti visive, in particolare la pittura, la fotografia e la cinematografia, hanno spesso sfruttato tali processi, a volte anche effettuando artificiosi elaborazioni sul materiale visivo, per ottenere la sensazione di distanza desiderata.

In particolare, vi sono molti **elementi monoculari**, cioè osservabili ed efficaci anche con un occhio solo, che concorrono al giudizio della distanza relativa tra gli oggetti osservati; alcuni di essi sono:

- Il **movimento parallattico**: la velocità di spostamento di un oggetto vicino sembra maggiore di quella di uno lontano.
- La **prospettiva lineare**: un oggetto di grandezza costante sottende angoli progressivamente minori (quindi è visto più piccolo), man mano che si allontana.  
L'esempio più classico è la convergenza apparente di due rette parallele che si allontanano dall'osservatore.
- La **sovraposizione dei contorni**: un oggetto che interrompa i contorni di un altro viene percepito come anteposto all'altro.  
Questo è un fattore molto importante per le inquadrature nelle riprese stereoscopiche.
- La **distribuzione delle luci e delle ombre**: il chiaro-scuro genera una impressione di rilievo e quindi di profondità relativa.
- La **familiarità con oggetti noti**: la distanza tra oggetti noti viene giudicata anche in base alla loro grandezza apparente.
- La **prospettiva aerea**: la foschia atmosferica influenza il contrasto e il colore degli oggetti situati più lontano. Inoltre ne sfuma i contorni.  
Il fenomeno viene spesso utilizzato in fotografi a mettendo a fuoco il soggetto principale e sfocando ad arte gli altri piani, in particolare lo sfondo.
- L'**accomodazione** del cristallino, ovvero la modificazione della sua forma per mettere a fuoco l'immagine sulla retina, a carico della muscolatura intrinseca.

Gli **elementi binoculari**, cioè quelli che vengono percepiti con ambedue gli occhi, aggiungono precisione e qualità alla percezione della distanza.

Il principale elemento binoculare è la **convergenza degli assi ottici oculari**, permessa dalla muscolatura estrinseca.

Peraltro, la convergenza su punti distanti più di 30 metri dall'osservatore (angolo di convergenza pari a circa 4°, ossia 0.062°) è ottenuta con spostamenti angolari del globo oculare troppo piccoli per essere sentiti e misurati.

Di maggiore interesse per la stereoscopia è la **stereopsi**, ossia la visione tridimensionale che origina dall'analisi delle piccole differenze tra le immagini recepite dai due occhi.

Nella realtà, infatti, un oggetto occupa un volume non nullo e quindi viene percepito come un insieme di punti oggetto che proiettano la propria immagine su altrettante coppie di punti retinici.

Solo un punto oggetto può essere il punto di fissazione: quello in cui convergono gli assi ottici oculari. I punti oggetto situati nei suoi pressi cadono o sull'oroptero o nell'area di Panum e vengono quindi fusi in punti singoli.

Tuttavia generano disparità retiniche la cui analisi e misura da parte del sistema percettivo conferiscono all'oggetto impressione di solidità e di profondità.

La stereopsi risulta perciò dalla lieve disparità con cui sono visti gli oggetti purchè siano compresi nell'area di Panum, ossia purché siano percepiti senza diplopia.

In altre parole ancora, la stereopsi è la visione tridimensionale che origina dalla stimolazione simultanea di punti retinici orizzontalmente disparati da parte di punti oggetto situati nell'ambito delle aree di Panum.

L'**acutezza stereoscopica** è la disparità minima oltre la quale non si produce alcun effetto stereoscopico. Valori pari a circa 15-30 secondi d'arco, ossia  $0.004^{\circ}$ - $0.008^{\circ}$ , sono ritenuti eccellenti.

Si noti che l'angolo di acutezza stereoscopica è circa 15 volte minore del minimo angolo di convergenza oculare sensibile.

Questo sembrerebbe implicare che la percezione della profondità è maggiormente dovuta alla stereopsi rispetto ad altri fenomeni, in particolare rispetto alla convergenza oculare.

---

## Tecniche stereoscopiche

---

Gli aspetti psicologici visti nella sezione precedente possono essere adeguatamente gestiti con una buona costruzione della scena e un buon rendering, assegnando ad ogni oggetto il giusto grado di complessità e di realismo.

I software che se ne occupano possono generare problemi di performance per il real time ma sono comunque sufficienti per scene non eccessivamente complessi.

Gli aspetti fisici, quali la disparità binoculare, la convergenza e l'accomodation sono gestiti dalla parte hardware che ad oggi è la parte che genera la maggior parte dei problemi tecnici ed è responsabile delle limitazioni presenti in questo ramo della realtà virtuale.

Iniziamo a valutare come costruire un corretto sistema hardware considerando un modello semplice di disparità binoculare ignorando per ora l'accomodation e la convergenza.

---

## I principi di funzionamento

---

Per **stereoscopia** si intende genericamente la tecnica che si prefigge di riprodurre il fenomeno della stereopsi.

Le tecnologie e le tecniche con le quali si ottiene questo fine sono molto disparate e, evidentemente, dipendono dalla tecnologia disponibile – si tenga conto che sono passati circa 170 anni dal primo strumento stereoscopico – ma tutte si basano sulla riproduzione artificiale della situazione reale che genera la stereopsi al fine di indurre nell'osservatore un'apparente percezione della profondità.

In particolare, le tecniche stereoscopiche mirano a riprodurre artificialmente i diversi angoli sotto cui, in una situazione reale, i due occhi vedrebbero i diversi punti che formano l'oggetto sotto osservazione.

Fattore comune di tutte le tecniche è l'uso di due immagini per volta, una per l'occhio destro e una per quello sinistro.

Per questo motivo si parla spesso di **coppia stereoscopica**, sia che si tratti di immagini fisse che di sequenze in movimento.

Le varie tecniche differiscono invece in modo sostanziale per la modalità con la quale la singola immagine della coppia stereoscopica viene veicolata all'occhio corretto.

Un altro punto importante è legato al punto di fissazione.

Questo è il punto reale in cui convergono gli assi ottici oculari quando si osserva un oggetto, e cade – per costituzione dell'occhio e per la stessa definizione di asse ottico oculare – al centro della scena osservata in ambedue gli occhi.

Quindi, tenendo conto che l'apertura di campo è identica per ambedue gli occhi, questi vedono la stessa scena (in altre parole: gli oggetti costituenti la scena vengono visti in posizioni grosso modo coincidenti) pur con le piccole differenze dovute alla parallasse - peraltro sfruttate dalla stereopsi.

Ciò giustifica la tecnica, pressoché universalmente utilizzata, di riprodurre le due immagini costituenti la coppia stereoscopica in posizioni spazialmente coincidenti (per esempio sullo stesso schermo), sfruttando opportune tecnologie per separare le immagini relative ai due occhi e veicolarle quindi all'occhio corretto; con altre tecniche le immagini vengono proiettate direttamente all'occhio corretto, ma in tali casi si richiede all'osservatore un adattamento per rendere spazialmente coincidenti le due immagini in modo artificioso (un esempio sono gli stereogrammi).

La figura 8 rappresenta la situazione in cui si trova un osservatore che stia osservando un oggetto del mondo reale come illustrato in figura 7.

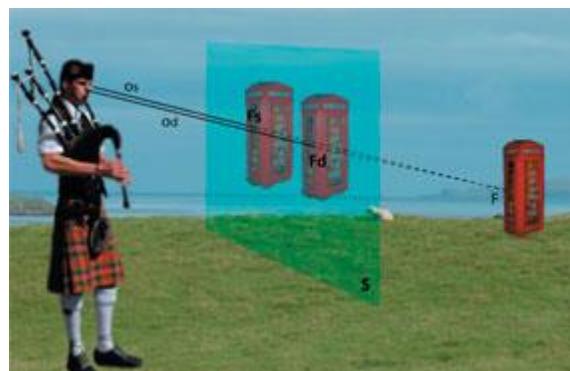


Fig. 7 - Illustrazione schematica della geometria dell'osservazione di un oggetto reale.

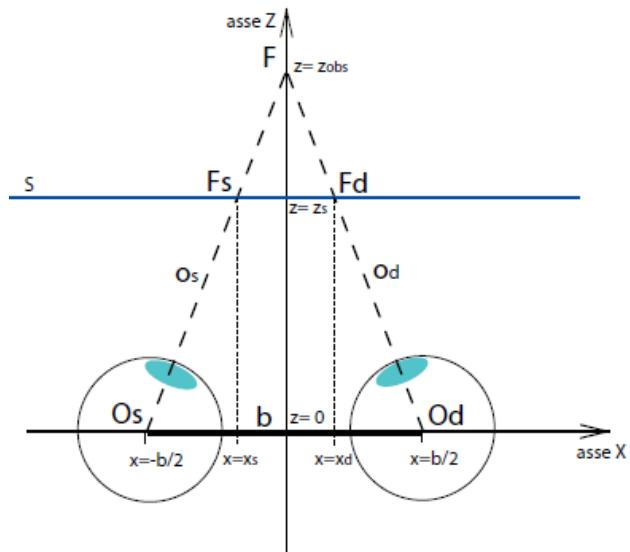


Fig. 8 - Modello descrittivo dell'osservazione reale, piano  $S$  interposto tra punto  $F$  e osservatore.

Per schematizzare, dell'osservatore si considerano solo gli occhi, in particolare il centro di rotazione degli assi ottici oculari,  $O_s$  e  $O_d$ , rispettivamente per l'occhio sinistro e per l'occhio destro, e dell'oggetto solo un punto, un **punto oggetto**, indicato con  $F$  e situato per semplicità di trattazione sull'asse  $Z$ .

L'osservatore, più precisamente il punto medio del segmento  $b$  che ha estremi in  $O_s$  e  $O_d$ , viene posto nell'origine di un sistema di assi cartesiani, il cui asse  $X$  comprende  $O_s$  e  $O_d$ , e corrisponde quindi nominalmente alla direzione **orizzontale trasversale**, l'asse  $Z$  corrisponde alle direzione "longitudinale davanti" all'osservatore, e l'asse  $Y$  (non visibile in figura) corrisponde alla direzione verticale che non riveste importanza per la derivazione che segue e può essere quindi trascurato.

Si noti che il piano  $XZ$  contiene gli assi ottici degli occhi ed è quindi conseguentemente inclinato rispetto all'*orizzontale*.

Inoltre, per motivi che saranno più chiari nel seguito, si consideri anche un piano  $S$ , ortogonale all'asse  $Z$ , interposto tra osservatore e punto osservato.

Quando l'osservatore fissa il punto  $F$ , gli assi ottici oculari,  $o_s$  e  $o_d$ , convergono su tale punto, e intersecano il piano  $S$  nei punti  $F_s$  e  $F_d$ ; questi, per come sono stati ricavati, corrispondono alle proiezioni del punto  $F$  sul piano  $S$  viste, rispettivamente, dall'occhio sinistro e dall'occhio destro. Si noti che in questo caso  $F_s$  è *a sinistra* di  $F_d$ .

Se il punto  $F$  risultasse interposto tra piano  $S$  e osservatore, si avrebbe la situazione illustrata in figura 9.

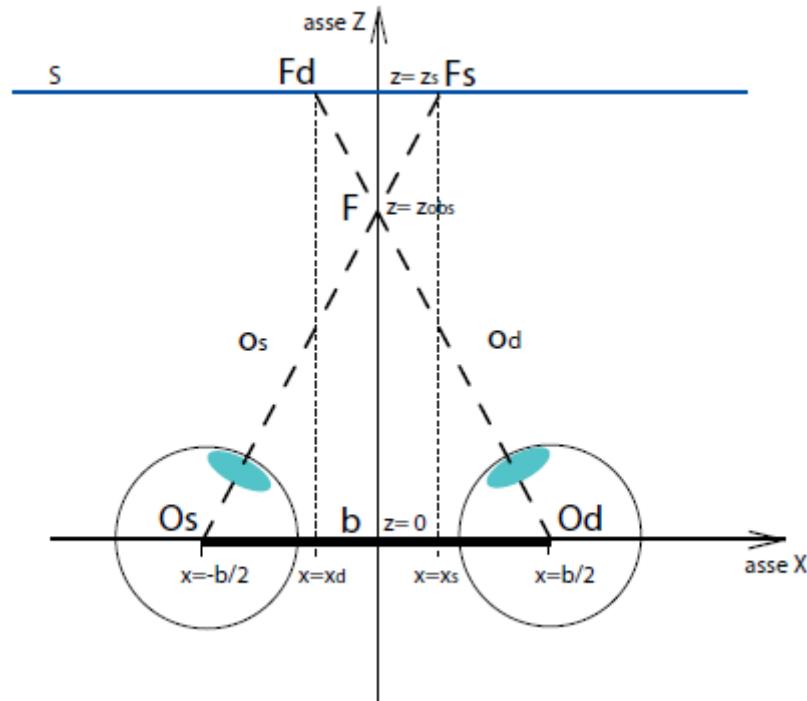


Fig. 9 - Modello descrittivo dell'osservazione reale, punto  $F$  interposto tra piano  $S$  e osservatore.

Le intersezioni con il piano  $S$  dei prolungamenti oltre al punto di fissazione  $F$  degli assi ottici oculari  $o_s$  e  $o_d$  sono ancora indicate con  $F_s$  e  $F_d$ .

Si noti che in questo caso  $F_s$  risulta *a destra* e  $F_d$  risulta *a sinistra*.

La stereoscopia si basa sull'assunto che, per ottenere la percezione della profondità, sia sufficiente proporre all'osservatore una riproduzione della realtà basata sulle proiezioni dei punti oggetto su uno schermo (bidimensionale), veicolate all'occhio corretto mediante opportune tecniche.

In questo differisce profondamente da quelle tecniche, quali l'olografia, che si basano invece sulla riproduzione della configurazione tridimensionale reale dei punti oggetto.

Rispetto a queste, la stereoscopia richiede minori risorse tecnologiche, tanto che da tempo è una realtà anche industriale, ancorché di nicchia.

Tuttavia, soffre di problemi dovuti all'interazione negativa tra la scena apparente, che sfrutta alcuni elementi binoculari, in particolare la stereopsi, e altri elementi non sempre controllabili ed evitabili, dovuti soprattutto alle condizioni reali di visione e alle tecniche di generazione della coppia stereoscopica.

## Modello della visione

---

In questo paragrafo si introduce un modello geometrico per la visione di una coppia stereoscopica, illustrata in linea di principio nella figura 10.

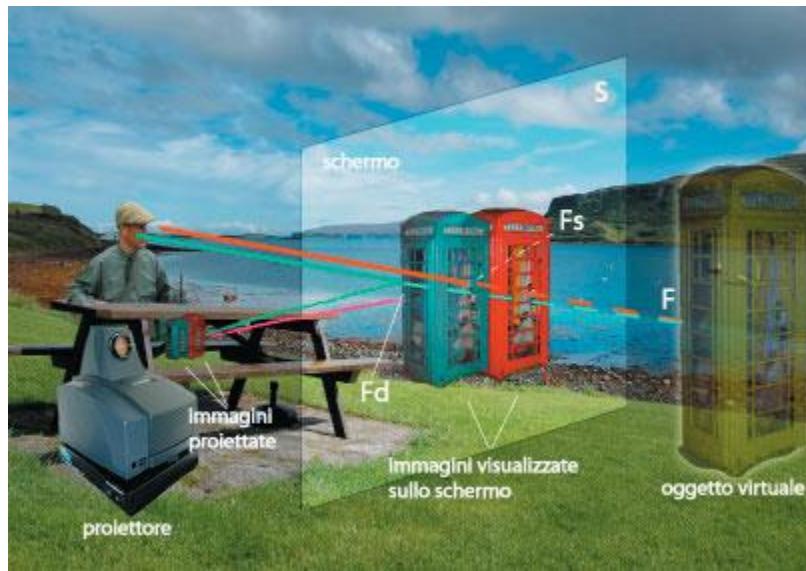


Fig. 10 - Illustrazione schematica della presentazione della coppia stereoscopica.

La figura 11 illustra schematicamente la configurazione di proiezione della coppia stereoscopica.

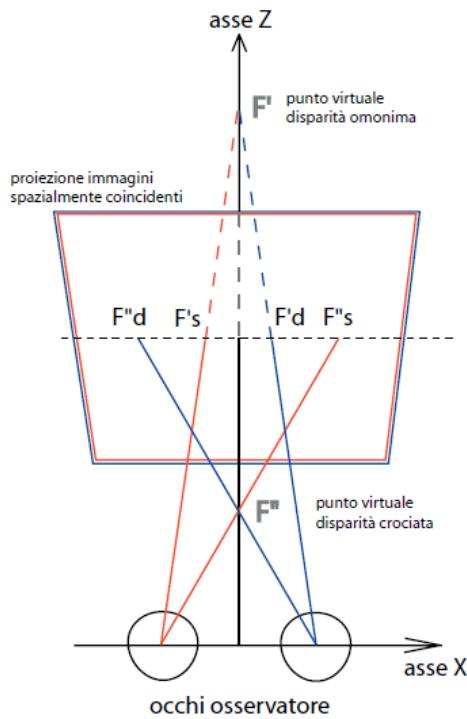


Fig. 11 - Configurazione schematica della proiezione della coppia stereoscopica.

Si noti la coincidenza spaziale tra le due immagini costituenti la coppia stereoscopica.

Tali immagini sono composte dalle proiezioni sullo schermo reale  $S$  delle immagini delle proiezioni dei punti oggetto sull'ipotetico piano  $S$  introdotto nel paragrafo precedente.

In figura, l'immagine da veicolare all'occhio sinistro è di colore rosso, quella da veicolare all'occhio destro è di colore blu.

Per distinguere i due casi sopra illustrati,  $F'_x$  si riferisce alla situazione in cui il piano  $S$  è interposto tra punto e osservatore,  $F''_x$  si riferisce invece all'altro caso.

I punti  $F'_s, F'_d, F''_s$  e  $F''_d$  sono realmente proiettati sullo schermo (reale), e, almeno in linea di principio, sono ricavati come proiezioni dei punti reali sull'ipotetico piano  $S$  introdotto nel paragrafo precedente. La figura 12 illustra la situazione vista dall'alto.

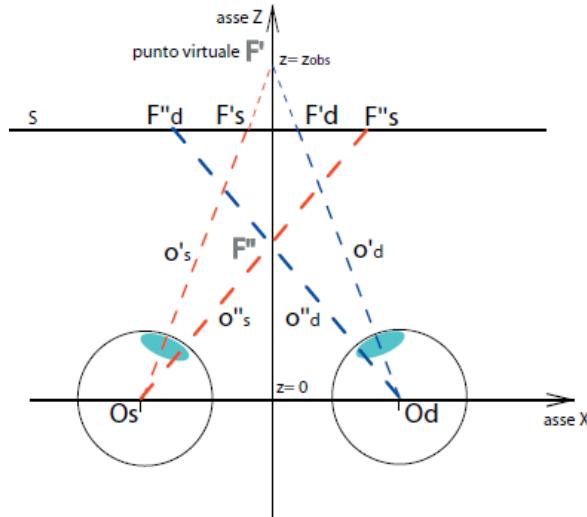


Fig. 12 - Schematizzazione della visione apparente del mondo reale

Quando l'osservatore considera i punti  $F'_s$  e  $F'_d$ , gli assi ottici  $o_s$  e  $o_d$  convergono verso il punto immaginario  $F'$  situato **al di là** dello schermo.

Invece quando l'osservatore considera i punti  $F''_s$  e  $F''_d$ , gli assi ottici  $o_s$  e  $o_d$  convergono verso il punto immaginario  $F''$  situato **al di qua** dello schermo, tra schermo e osservatore.

L'illusione cui tende la tecnica è quella di stimolare nell'osservatore la sola percezione di due punti,  $F'$  e  $F''$ , nell'esempio situati a distanza diversa e, rispettivamente, **al di là** e **al di qua** dello schermo, tramite la proiezione di immagini generate in modo opportuno.

Tale configurazione geometrica può essere modellata come rappresentato in figura 13 nel caso di punto percepito **al di là** dello schermo.

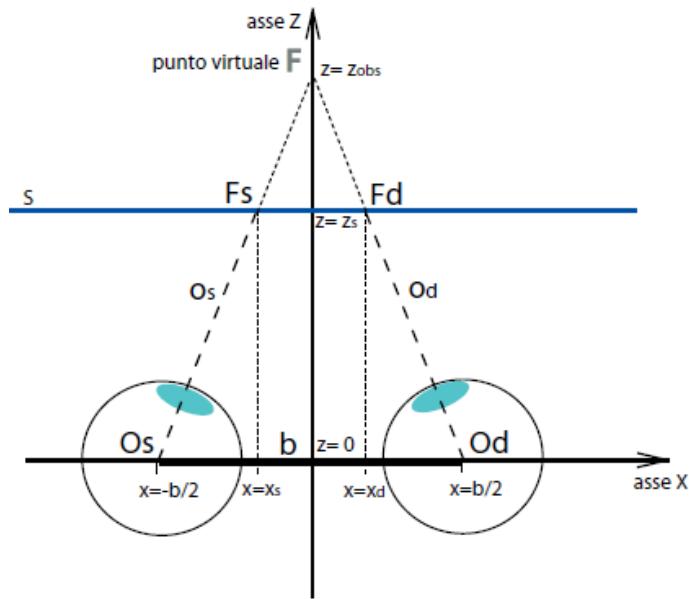


Fig. 13 - Modello della visione - punto "al di là" dello schermo.

Utilizzando la similitudine dei triangoli  $O_s F O_d$ , e  $F_s F F_d$ , tenendo conto che la lunghezza dei segmenti deve essere sempre positiva, si ottiene:

$$\frac{x_d - x_s}{z_{obs} - z_s} = \frac{\frac{b}{2} - \left(-\frac{b}{2}\right)}{x_{obs}} \quad (1)$$

che, risolta evidenziando la distanza  $z_{obs}$  tra punto e osservatore, dà la seguente espressione (le variabili usate sono illustrate nel seguito):

$$z_{obs} = z_s \frac{b}{b + x_s - x_d} \quad (2)$$

La configurazione in cui il punto viene percepito tra lo schermo e l'osservatore è invece modellata geometricamente come in figura 14.

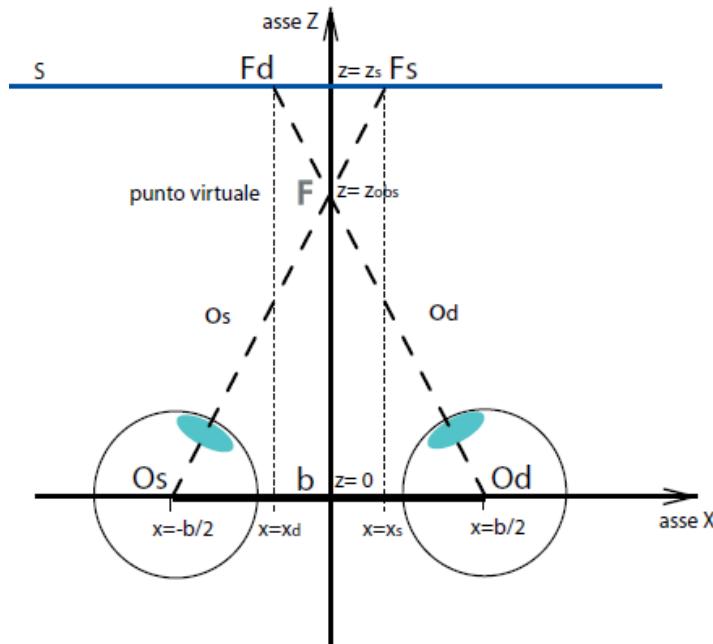


Fig. 14 - Modello della visione - punto "al di qua" dello schermo.

Utilizzando anche in questo caso la similitudine dei triangoli  $O_s F O_d$ , e  $F_s F F_d$ , tenendo conto che la lunghezza dei segmenti deve essere sempre positiva, si ottiene:

$$\frac{x_s - x_d}{z_s - z_{obs}} = \frac{\frac{b}{2} - \left(-\frac{b}{2}\right)}{z_{obs}} \quad (3)$$

che, risolta evidenziando la distanza  $z_{obs}$  tra punto e osservatore, dà la seguente espressione (le variabili usate sono illustrate nel seguito):

$$z_{obs} = z_s \frac{b}{b + x_s - x_d} \quad (4)$$

Si noti che le formule (2) e (4) sono formalmente identiche, perciò i due casi si possono modellare con la stessa formula che qui si riporta per chiarezza:

$$z_{obs} = z_s \frac{b}{b + x_s - x_d} \quad (5)$$

dove:

- $z_{obs}$  = distanza tra osservatore e punto di fissazione percepito  $F$ , coincidente con la coordinata  $Z$  del punto stesso
- $z_s$  = distanza tra osservatore e schermo  $S$
- $x_s, x_d$  = ascisse punti proiettati  $F_{s,d}$
- $b$  = distanza tra i centri di rotazione degli assi ottici; non differisce sostanzialmente dalla distanza interpupillare.

La distanza percepita  $z_{obs}$  del punto apparente dall'osservatore è proporzionale alla distanza  $z_s$  tra osservatore e schermo, è all'incirca inversamente proporzionale alla distanza tra i punti proiettati  $x_s - x_d$  e dipende in modo più complesso dalla distanza interpupillare  $b$ .

Quest'ultima dipende dall'osservatore.

In un maschio adulto varia da 5.77 cm a 6.96 cm, con una media di 6.32 cm e una variazione percentuale pari a circa il 20%. I bambini presentano ovviamente valori inferiori.

La distanza tra punti proiettati  $x_s - x_d$  è generata ad arte per generare l'effetto di profondità desiderato.

È l'unico parametro in linea di principio controllabile da chi produce le immagini.

Si noti però che per generare esattamente l'effetto tridimensionale desiderato, in sede di produzione si dovrebbe poter conoscere esattamente la dimensione dello schermo su cui avverrà la proiezione.

Infatti, a parità di distanza  $x_s - x_d$ , misurata in sede di produzione in termini di pixel, la proiezione su schermi di dimensioni differenti mostrerà differenti valori per la distanza  $x_s - x_d$  misurata in termini di lunghezza lineare.

In ogni caso, il valore della distanza  $x_s - x_d$  non deve superare quello della distanza interpupillare altrimenti gli occhi sarebbero forzati a divergere in modo innaturale.

In formule, si deve rispettare la diseguaglianza seguente:

$$x_d - x_s \leq b \quad (6)$$

Nel caso in cui i punti proiettati risultassero coincidenti, il punto oggetto apparente sarebbe percepito adagiato sullo schermo.

Il modello ci conferma l'esperienza; infatti, ponendo nella (5):

$$x_s - x_d = 0 \quad (7)$$

si ottiene:

$$z_{obs} = z_s \quad (8)$$

La distanza  $z_s$  tra osservatore schermo può essere controllata solo in parte poiché, in genere, non si può intervenire sulle condizioni di visione.

Tuttavia, per limitare le differenze di percezione la distanza dallo schermo degli spettatori non deve essere troppo diversa.

Si noti che nei teatri IMAX le poltrone sono disposte su un settore di corona circolare non eccessivamente profondo.

### Criticità legate alla tecnica stereoscopica

---

Dall'analisi della formula (5), che descrive il modello geometrico della visione, emerge come sia praticamente impossibile, data la non controllabilità dei vari parametri, riprodurre fedelmente la geometria cui un osservatore reale sarebbe spettatore in una situazione reale.

Ciò si potrebbe teoricamente ottenere se tutti i parametri, anche e soprattutto quelli relativi alla ripresa, non considerati in questo lavoro, potessero essere misurati e/o controllati (per esempio: simulazioni personalizzate in ambienti controllati e per osservatori singoli).

Nel campo televisivo si devono invece considerare, per i vari parametri, dei limiti non troppo stringenti: non si potrà quindi pretendere dal sistema una assoluta fedeltà quantitativa, ma si otterrà una rappresentazione in certa misura anche percepita come artificiosa.

Un esempio è dato dal fenomeno detto **effetto teatro delle marionette** (*puppet theater effect*), consistente nella visualizzazione di oggetti conosciuti, soprattutto le persone, con una grandezza innaturalmente inferiore a quanto ci si aspetterebbe in relazione alla distanza percepita.

La stereoscopia è afflitta inoltre da fattori disturbanti dovuti all'artificiosità della tecnica.

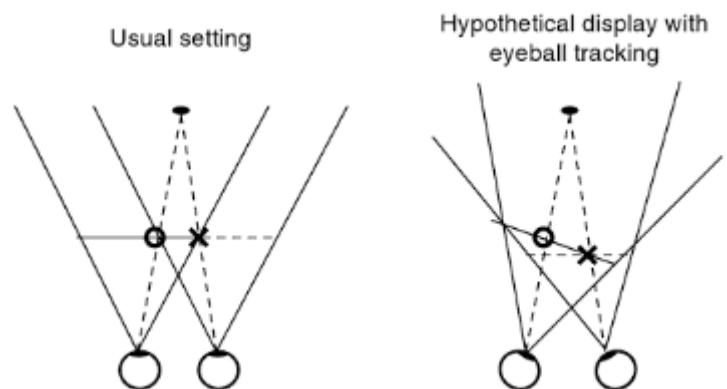
Per esempio, infatti, sebbene si utilizzi la stereopsi e si possano convenientemente adottare i classici elementi monoculari già noti nel campo televisivo, in sede di visualizzazione l'occhio si adatta, sia come punto di fissazione che come distanza, sullo schermo, e da ciò possono nascere incongruenze tra le sensazioni reali e le sensazioni apparenti suscite dalla proiezione stereoscopica.

I numerosi fattori disturbanti possono provocare affaticamento se lievi, ma possono anche portare alla perdita della percezione della prospettiva se di entità eccessiva.

Considerando il singolo utente è possibile creare un buon sistema, facendo un preciso tracking dell'utente stesso.

In caso di un numero elevato di spettatori (ad esempio al cinema) si effettua il calcolo mediando la posizione degli utenti.

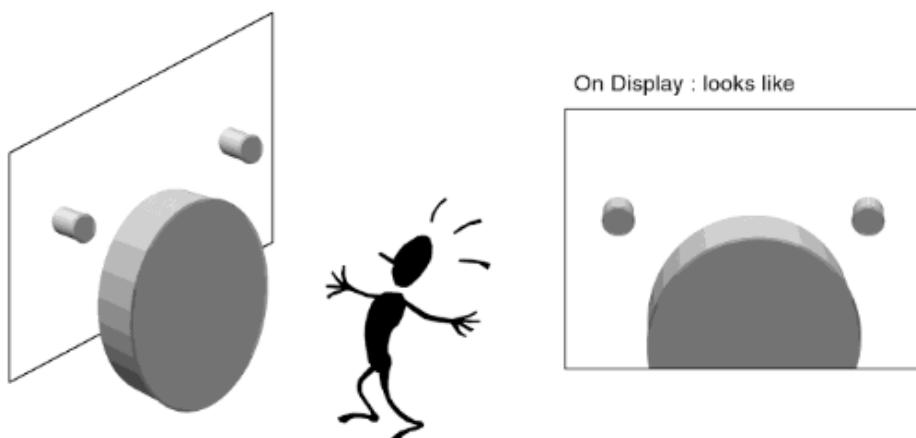
Inoltre è necessario tenere conto che se la linea fra gli occhi non è parallela allo schermo, è necessario gestire sensazioni di profondità diverse data la posizione dell'utente rispetto allo schermo, effettuando un tracking obbligatorio continuo.



Come la disparità dev'essere calcolata se si conosce il punto attuale di convergenza, per esempio, grazie al tracking degli occhi.

In ultimo, ricordo che è cosa buona e giusta, quando si usa la disparità per generare immagini che escono dallo schermo, controllare che l'immagine sia completamente contenuta nello schermo.

In caso contrario gli oggetti che escono risultano tagliati e il senso di immersione va a banane.



L'effetto del boundary: gli oggetti con la parallasse negativa sembrano essere dietro la superficie dello schermo con parallasse positiva; gli oggetti dovrebbero essere collocati al centro dello schermo e dimensionati correttamente se possibile per evitare questo effetto.

## Display

Elenchiamo ora alcuni dei display maggiormente usati nella pratica in quanto economicamente sostenibili:

- **autostereoscopici**: barriera parallasse e lenti cilindriche
- **non-autostereoscopici**: passivi ed attivi
- **head-mounted display (HMD)**: è uno schermo montato sulla testa dello spettatore attraverso un casco ad hoc
- **display multipli (CAVE)**.

### Autostereoscopici

Mirano a dare il senso della profondità e della tridimensionalità solo attraverso lo schermo, senza dover fare indossare nulla agli utenti.

I più utilizzati sono barriera parallasse e lenti cilindriche.

#### BARRIERA PARALLASSE

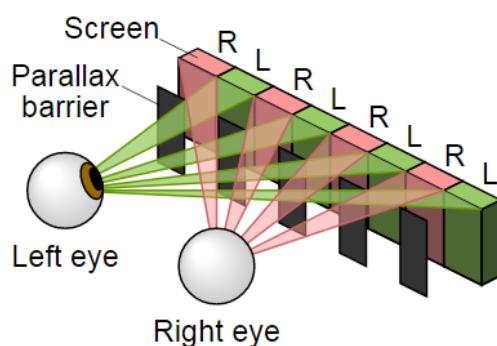
Davanti allo schermo viene posizionata la **barriera parallasse**, una struttura simile ad una rete che oscura parti diverse dello schermo in base all'angolazione da cui la si guarda.

La rete è sufficientemente fine da far sì che l'immagine percepita sembri continua e non reticolata, se vista da una distanza minima.

L'immagine che viene visualizzata sullo schermo è formata dalle immagini che devono colpire l'occhio destro e quello sinistro (le chiameremo *L* e *R*), mescolate in base alla struttura della rete, in modo che osservando lo schermo da una certa distanza e angolazione l'occhio destro percepisce solo l'immagine *R*, mentre il sinistro solo *L*.

La posizione dell'utente però deve essere fissata.

Un esempio di applicazione è il Nintendo.



#### LENTI CILINDRICHE

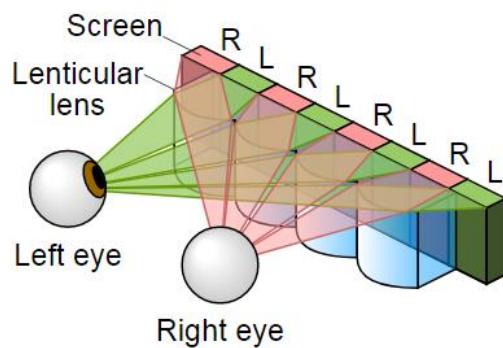
Così come veniva fatto nelle figurine delle patatine, davanti allo schermo sono posizionate delle lenti cilindriche, le quali proiettano immagini diverse in base all'angolo da cui sono osservate.

L'immagine presente sullo schermo è formata da tutte le immagini che vogliono essere viste dalle diverse angolazioni mescolate in modo adeguato fra loro.

Con una configurazione corretta è possibile costruire 60 viste, fornendo 30 punti di vista diversi della stessa scena (1 per occhio).

L'unico difetto è che muovendosi attorno allo schermo e osservando le lenti da diverse angolazioni si crea fra un'immagine e la sua consecutiva una transizione, riducendo il senso di continuità che si avrebbe ruotando attorno ad un vero oggetto 3D.

Inoltre le lenti riducono la luminosità.



## Non-autostereoscopici

Usano un supporto fissato sull'utente e si distinguono in passivi ed attivi.

### PASSIVI

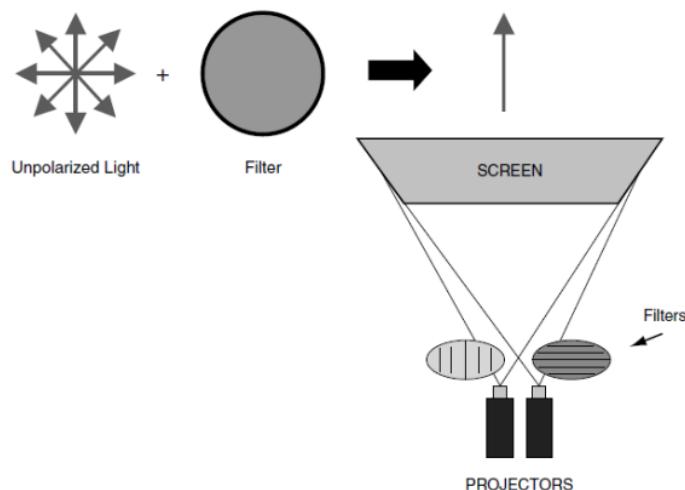
Non compiono direttamente un'attività ma filtrano le immagini ricevute.

Ne sono esempi quelli:

- **bicolore**: vengono composte immagini sovrapposte di colore rosso e ciano, che vengono nascoste dalla lente avente lo stesso colore



- **polarizzati**: vengono usate lenti a diversa polarizzazione (orizzontale o verticale) e create immagini con diverse polarizzazioni per riga, o proiettando immagini a diversa polarizzazione. La **polarizzazione** è la direzione in cui si amplia l'onda luminosa. Problema dei sistemi polarizzati è che si verifica spesso un effetto di cross-polarizzazione, che mostra ad un occhio le immagini riservate all'altro, provocando un effetto di sfarfallio dei bordi detto **ghosting**.



## ATTIVI

Sono strumenti quali occhiali o caschi che proiettano nelle due lenti le immagini riservate all'occhio destro e a quello sinistro.

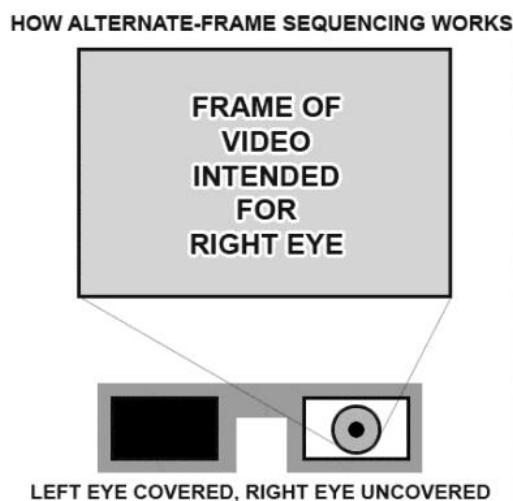
Una prima versione prevede di sincronizzare l'occhiale con lo strumento di visualizzazione (es. la tv) in modo che questo emetta a frame alterni immagini riservate all'occhio destro e sinistro e contemporaneamente l'occhiale oscuri una delle due lenti per abilitare solo un occhio.

Con un frame rate sufficientemente alto si ha la sensazione della continuità delle immagini, ma è un supporto pesante e deve essere sincronizzato.

Successivamente si è pensato a sistemi quali *l'head mounted* (il classico oculus rift), nel quale in ogni lente viene proiettata l'immagine desiderata.

L'idea è la stessa alla base dei binocoli visibili al museo del cinema, solo con una migliore risoluzione e portabilità.

Il problema è solo quello di costruire immagini molto vicine agli occhi, risolvibile tramite l'uso di lenti ottiche apposite.





## 9. La ripresa stereoscopica

Finora abbiamo visto come costruire un sistema stereoscopico per la visualizzazione, ma dobbiamo capire ora come produrre le immagini da proiettare sugli schermi.

I modelli che vediamo ora possono essere usati per la cattura di immagini con videocamere, ma anche per la sintesi di nuove immagini partendo da immagini memorizzate.

Nel campo televisivo è comune utilizzare una coppia di telecamere - reali oppure, nell'ambito della computer grafica, virtuali – per generare i segnali costituenti una “coppia stereoscopica”.

Un sistema di ripresa composto da due telecamere può essere disposto solamente secondo due configurazioni geometriche: infatti, posto che devono giacere sullo stesso piano, gli assi ottici delle telecamere possono essere paralleli o convergere in un punto.

In questo articolo si analizzano le proprietà geometriche delle due configurazioni, ricavando le relazioni che descrivono la conversione tra lo spazio reale della scena ripresa e lo spazio percepito in fase di visualizzazione.

---

### Introduzione

---

In questo paragrafo si illustrano i principi di funzionamento della stereoscopia.

Questi si basano su alcune caratteristiche della visione umana; in particolare sulla convergenza oculare verso un punto (*punto di fissazione*), in genere ubicato in posizione centrale rispetto all'oggetto di interesse, e sull'analisi delle disparità tra i punti retinici corrispondenti (*stereopsi*), resa possibile dal fenomeno della  *fusione sensoriale* grazie alla quale l'immagine che si focalizza sui punti retinici corrispondenti viene percepita come appartenente allo stesso oggetto.

La prima caratteristica implica che le due scene viste dai due occhi siano percepite come spazialmente coincidenti.

La seconda permette di valutare la posizione spaziale dei punti costituenti la superficie visibile dell'oggetto rispetto al punto di fissazione e quindi di percepire la tridimensionalità dell'oggetto osservato.

Le tecniche di visualizzazione stereoscopica riproducono virtualmente la *geometria di osservazione* - ossia la configurazione geometrica che si ha durante l'osservazione di una scena da parte di un osservatore reale - in modo da stimolare la visione tridimensionale stereoscopica nell'osservatore.

Per ottenere ciò, le tecniche stereoscopiche che potremmo definire “tradizionali” utilizzano una coppia di segnali, chiamata *coppia stereoscopica*, veicolando il segnale corretto all'occhio corrispondente.

Negli stereoscopi in uso nel passato come sorgenti di segnale si utilizzavano coppie di disegni o di fotografie; in campo cinematografico si svilupparono tecniche specifiche, per lo più basate su coppie di cineprese; nel campo televisivo è comune utilizzare una coppia di telecamere - reali oppure, nell'ambito della computer grafica, virtuali.

Analizzando il processo di ripresa, emerge che nella conversione tra lo spazio reale e le immagini stereoscopiche si producono delle distorsioni, alcune delle quali dovute alla configurazione geometrica di ripresa, ossia all'orientazione e alla posizione dei sensori rispetto alla scena.

L'analisi delle distorsioni sarà tema di un prossimo articolo, mentre in questo lavoro si analizzano le configurazioni geometriche di ripresa stereoscopica esaminandone le caratteristiche ubicate su una linea orizzontale.

In condizioni di normalità fisica, gli assi oculari sono complanari e convergenti verso il punto di convergenza; solo in presenza di patologie possono risultare sghembi. Tale condizione impone delle restrizioni alle possibili configurazioni della geometria di osservazione.

Nella ripresa stereoscopica, al posto degli occhi dell'ipotetico osservatore si pongono dei sensori. Pertanto la geometria di ripresa eredita alcune caratteristiche di quella di osservazione; in particolare si verificano dei legami geometrici tra i punti della scena ripresa tridimensionale e le relative proiezioni sui piani bidimensionali dei sensori, tali da porre delle restrizioni sul loro posizionamento.

Tali legami sono analizzati nell'ambito della *geometria epipolare*.

Nell'analisi del processo di ripresa, si assume che i sensori possano essere modellati come se fossero delle *camere ottiche*.

Questo semplifica l'analisi geometrica perché si evita di dover prendere in considerazione le caratteristiche geometriche delle ottiche reali e, soprattutto, le loro distorsioni e aberrazioni.

## I sistemi di riferimento

Per effettuare le analisi geometriche necessarie a stabilire le relazioni tra i punti della scena ripresa e i punti della scena percepita, conviene definire dei sistemi di riferimento cartesiani opportunamente posizionati e orientati.

Il sistema di riferimento per i sensori corrisponde a quello utilizzato nel modello basato sulla camera ottica e si può considerare adagiato sul piano immagine.

Ovviamente ci saranno due sistemi di riferimento, uno per ogni camera considerata (nella stereoscopia tradizionale si usano due camere).

Il sistema di riferimento per la scena invece è unico e vale per ambedue le telecamere.

Per questo motivo se ne pone l'origine in un punto centrale, ubicato sul piano di simmetria dell'apparato, come meglio illustrato nel paragrafo seguente.

Per derivare i legami tra i punti immagine e la scena percepita conviene inoltre introdurre un terzo sistema di riferimento utilizzato per descrivere la geometria di visione.

## LA CAMERA OTTICA

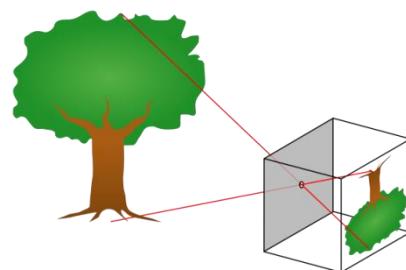
Detta anche camera oscura, in inglese **pin hole camera**, è un dispositivo ottico che sta alla base di tutta la tecnica fotografica.

La camera oscura più semplice è formata da una scatola chiusa, con un piccolo foro, detto stenopeico, su una faccia dal quale entra la luce.

Quest'ultima forma sulla faccia opposta l'immagine reale capovolta di quanto si trova davanti al foro. Più il foro è piccolo, più l'immagine è nitida e definita, ma le dimensioni del foro non si possono ridurre più di un certo limite per evitare fenomeni di diffrazione e per lasciare entrare una quantità di luce sufficiente per gli usi pratici.

Il pregio maggiore di una camera così semplice è che tutti gli oggetti sono a fuoco a prescindere dalla loro distanza, ancorché le immagini possano essere poco nitide a causa delle dimensioni finite del foro stenopeico; in altre parole, il foro stenopeico si comporta come un obiettivo che non ha una sua lunghezza focale specifica e che ha profondità di campo illimitata.

La limitata quantità di luce entrante nel foro limita l'uso di tale camera alla ripresa di oggetti fissi, per i quali si possono adottare tempi di esposizione relativamente lunghi.



## IL SISTEMA DI RIFERIMENTO PER LA SCENA RIPRESA – COORDINATE SCENA

Al fine di individuare la posizione spaziale dei punti costituenti la scena, chiamati *punti oggetto*, si definisce un sistema cartesiano tridimensionale, illustrato in figura 1, che ha:

- l'origine sul punto medio del segmento congiungente i centri ottici dei due sensori (cioè i fori stenopeici dei modelli adottati per le telecamere)
- l'asse delle ascisse  $X$  adagiato sullo stesso segmento, con valori positivi verso sinistra (guardando verso l'asse delle  $Z$ ).  
Nell'uso comune si trova in posizione orizzontale
- l'asse delle ordinate  $Y$  perpendicolare all'asse delle ascisse e rivolto verso l'alto.  
Nell'uso comune si trova lungo la verticale
- infine, l'asse  $Z$  perpendicolare ai due assi precedenti e rivolto verso la scena ripresa.

I sensori sono posizionati simmetricamente rispetto al piano  $ZY$  in ambedue le configurazioni che si analizzeranno in questo lavoro.

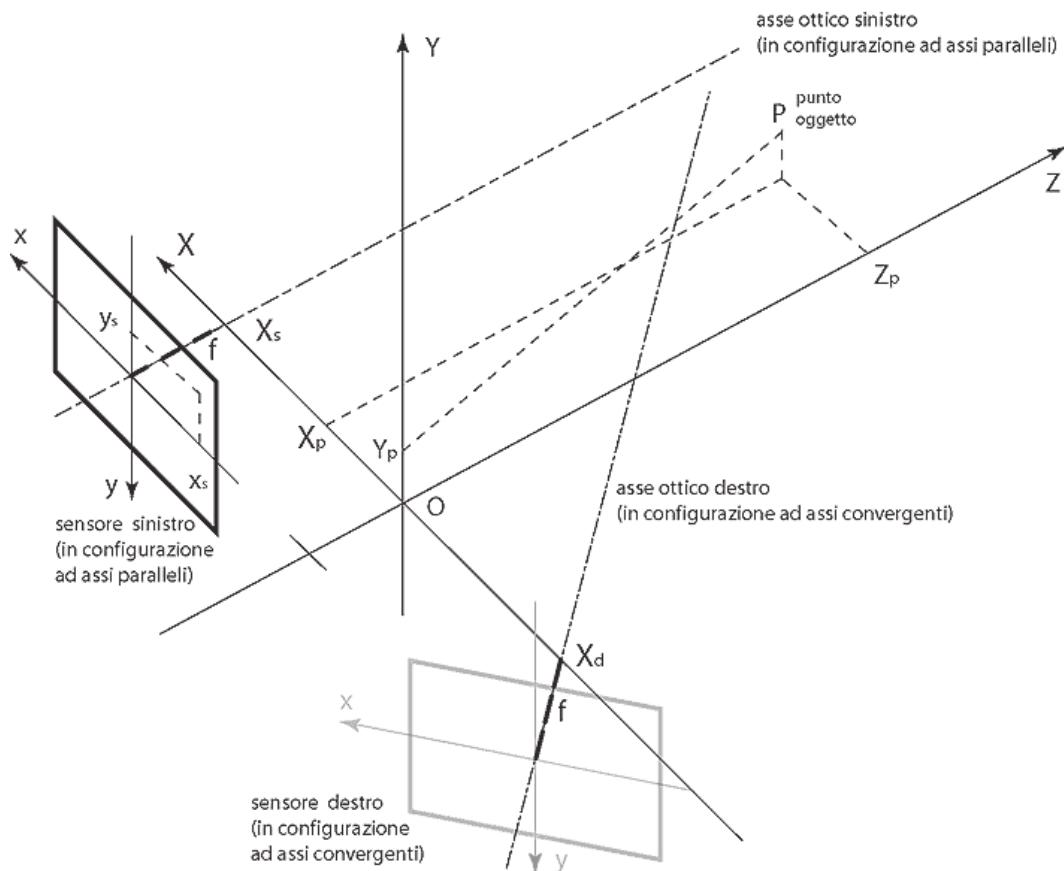


Fig. 1 – Sistema di riferimento per l'individuazione dei punti oggetto (spazio oggetto).

Per migliore comprensione, in figura si riporta anche una coppia di sensori, con il relativo sistema di coordinate locali. Il sensore sinistro è disposto secondo la configurazione ad assi ottici paralleli e quello destro secondo la configurazione ad assi ottici convergenti.

In particolare i centri ottici delle camere ricadono nei punti  $(X_d, 0, 0)$  e  $(X_s, 0, 0)$  dove  $X_s = -X_d = \frac{b}{2}$ , e la distanza tra il centro del sensore e il centro ottico è pari alla distanza focale  $f$ . Le due differenti configurazioni geometriche di ripresa si ottengono ruotando i sensori attorno ad un asse verticale passante per il centro ottico del modello relativo.

I principali parametri geometrici di cui tenere conto sono:

- $X_p, Y_p, Z_p$  sono le coordinate del punto oggetto rispetto al sistema di riferimento  $XYZ$ .  $Z_p$  è la coordinata  $Z$  del punto oggetto.  
In prima approssimazione si può considerare uguale alla distanza del punto oggetto dalle telecamere.
- $b = X_s - X_d$  è la distanza tra i centri ottici delle telecamere, che nel modello della camera ottica sono costituiti dal foro stenopeico; tale parametro si chiama *base stereoscopica*.
- $f$  è la lunghezza focale delle ottiche delle telecamere.  
Nel modello della camera ottica consiste nella distanza tra il foro stenopeico (centro ottico) e il piano del sensore.

### IL SISTEMA DI RIFERIMENTO PER I SENSORI – COORDINATE IMMAGINE

Per individuare la posizione dei punti costituenti la proiezione dei *punti oggetto* sulla superficie dei sensori, chiamati *punti immagine*, si definisce un sistema cartesiano bidimensionale, illustrato in figura 2, che ha l'origine nel centro del sensore, e, rispetto al sistema di coordinate definito per la scena ( $X, Y, Z$ ), presenta l'asse delle ascisse  $x$  orientato verso sinistra, e l'asse delle ordinate  $y$  orientato verso il basso, come riportato in figura 1.

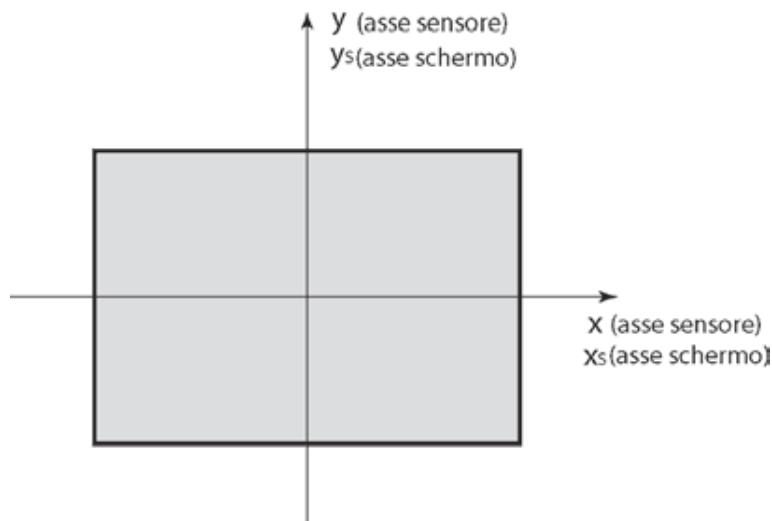


Fig. 2 – Sistema di riferimento per i punti immagine (per il sensore e per lo schermo).

L'orientazione degli assi  $x$  e  $y$  illustrata in figura 1 permette di realizzare facilmente la rotazione di  $\pi$  (180°) attorno all'asse ottico del sensore, necessaria per “raddrizzare” l'immagine proiettata sul sensore stesso che, nel modello di camera ottica, risulta rovesciata rispetto alla scena ripresa.

A parte il fattore moltiplicativo  $M$  di cui si parla nel paragrafo successivo, questo sistema di riferimento è utilizzabile anche per individuare un punto sullo schermo dell'apparato visualizzatore.

In questo caso, per evitare confusione, si indicano gli assi con i simboli  $x_S$  e  $y_S$  (la  $S$  in apice è maiuscola).

### IL SISTEMA DI RIFERIMENTO PER LA VISIONE – COORDINATE OSSERVATORE

Per l'analisi della configurazione geometrica posta in essere in fase di visione si introduce un ulteriore sistema di riferimento, illustrato in figura 3a.

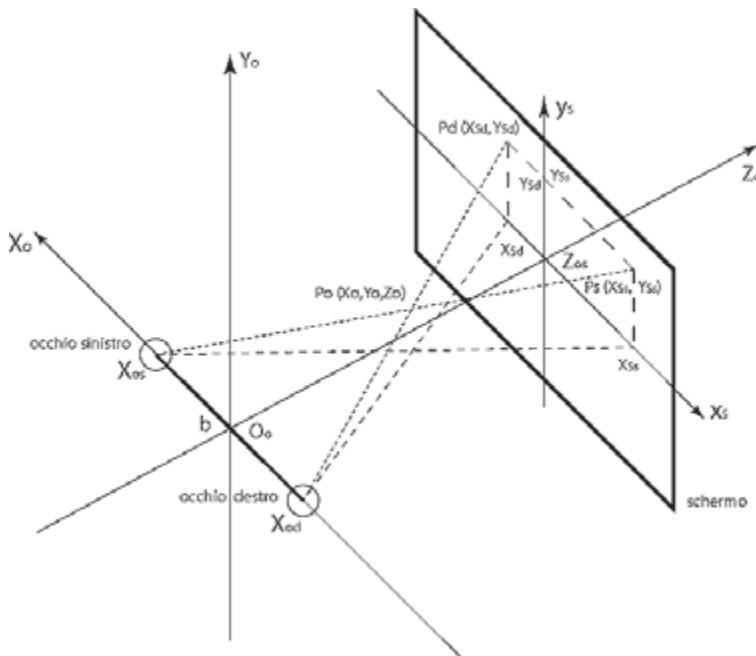


Fig. 3a – Sistema di riferimento per l'analisi geometrica della configurazione di visione.

Il centro  $O_o$  del sistema di coordinate cartesiane tridimensionali  $(X_o, Y_o, Z_o)$  è posto nel punto medio del segmento che unisce i centri ottici oculari, a loro volta ubicati sull'asse  $X_o$  in corrispondenza delle ascisse  $X_{od}$  e  $X_{os}$ , rispettivamente per l'occhio destro e quello sinistro.

Lo schermo su cui viene visualizzata la coppia stereoscopica è posto sul piano  $Z_o = Z_{os}$ , parallelo al piano  $X_o Y_o$ .

Il centro del sistema di coordinate definite su di esso si trova, per semplicità, sull'asse  $Z_o$ , e gli assi  $x_S$  e  $y_S$  sono paralleli, rispettivamente agli assi  $X_o$  e  $Y_o$ .

Il segmento  $e = X_{os} - X_{od}$  rappresenta la distanza tra i centri ottici oculari e corrisponde alla base stereoscopica dell'osservatore.

L'ordine dei termini è scelto opportunamente per ottenere un valore positivo, considerando il verso dell'asse  $X_o$ .

Il punto  $P$  ripreso dalla scena reale è proiettato sullo schermo nei punti  $P_d$  e  $P_s$  tramite, rispettivamente, la componente destra e quella sinistra della coppia stereoscopica.

L'osservatore vede con l'occhio destro l'immagine di  $P_d$  e con quello sinistro l'immagine di  $P_s$ , percependo un singolo punto  $P_o$  posizionato in  $(X_o, Y_o, Z_o)$ .

Si possono introdurre altre due grandezze molto importanti, ossia la *disparità orizzontale*,  $\delta_{So}$ , e la *disparità verticale*,  $\delta_{Sv}$ , definite come differenze di coordinate relative allo schermo e date dalle relazioni seguenti:

- 1a)  $\delta_{So} = x_{sd} - x_{ss}$
- 1b)  $\delta_{Sv} = y_{sd} - y_{ss}$

La prima è quella che genera la percezione della distanza e della profondità e quindi è una grandezza desiderata, che si cerca di preservare nell'intera filiera dalla ripresa alla visualizzazione.

La seconda genera affaticamento visivo senza dare informazioni.

È pertanto indesiderata, ma in certe situazioni è inevitabile.

Quando disparità orizzontale assume valori positivi, si parla di *disparità omonima*; gli assi ottici oculari si intersecano ad una distanza dall'osservatore maggiore della distanza dello schermo e conseguentemente il punto  $P_o$  viene percepito "al di là" dello schermo stesso.

Al contrario, quando disparità orizzontale assume valori negativi, si parla di *disparità crociata*; gli assi ottici oculari si intersecano ad una distanza dall'osservatore inferiore alla distanza dello schermo e conseguentemente il punto  $P_o$  viene percepito "al di qua" dello schermo, in una posizione intermedia tra l'osservatore e lo schermo stesso.

### La camera ottica come modello geometrico di telecamera generica

---

Per analizzare le relazioni geometriche intercorrenti tra le coordinate tridimensionali di un punto della scena ripresa e le coordinate della sua proiezione sul piano immagine, si utilizza un modello basato sulla camera ottica ideale.

Questo modello non include alcuna distorsione dovuta alle lenti degli obiettivi. Inoltre il piano immagine è considerato continuo, mentre molti sensori attuali, essendo costituiti da celle, presentano, di fatto, solo coordinate quantizzate.

Date le semplificazioni assunte, è da considerarsi un modello ideale di telecamera; tuttavia è rappresentativo e utile perché permette di concentrarsi sulla geometria complessiva evitando le complicazioni dovute alla complessa geometria ottica degli obiettivi reali e agli inevitabili fattori spuri, in particolare le distorsioni e le aberrazioni, che si verificano nella pratica.

#### GEOMETRIA DELLA CAMERA OTTICA

La geometria utilizzata per il modello della camera ottica è illustrato nella figura, dove sono anche riportati i componenti ed i parametri principali, in particolare:

- Un **sistema di coordinate cartesiane tridimensionali** con origine  $O$  nel punto focale della camera ottica (ossia il foro stenopeico).  
I tre assi del sistema di riferimento sono indicati con  $X$ ,  $Y$  e  $Z$ . L'asse  $Z$  è rivolto verso la direzione in cui punta la camera ed è coincidente con l'asse ottico della stessa.
- Un **piano immagine** su cui viene proiettata, attraverso il foro stenopeico, la scena tridimensionale ripresa.  
Tale piano è parallelo al piano  $XY$  ed è posizionato ad una distanza  $f$  dall'origine  $O$  nella parte delle  $Z$  negative.  
Il parametro  $f$  è chiamato *lunghezza focale* della camera.
- Un **punto  $P$  appartenente alla scena** tridimensionale ripresa, con coordinate  $(X_p, Y_p, Z_p)$  relative al sistema d'assi cartesiani  $XYZ$ .
- Una **linea di proiezione** del punto  $P$  verso la camera, passante per il punto  $O$  (foro stenopeico).
- La proiezione del punto  $P$  sul piano immagine, individuata con la lettera  $p$ , e determinata dalla intersezione della linea di proiezione con il piano immagine.
- Un **sistema di coordinate cartesiane bidimensionali** con origine  $o$  nel centro del piano immagine (cioè del sensore).  
I due assi del sistema di riferimento sono indicati con  $x$  e  $y$ , e le coordinate del punto  $p$  rispetto a questo sistema di assi sono  $(x_p, y_p)$ .  
In figura sono indicati i versi opportuni perché - dopo la rotazione di  $\pi$  ( $180^\circ$ ) del piano immagine attorno all'asse ottico, necessaria per "raddrizzare" l'immagine in fase di visualizzazione - si ottenga un sistema con l'orientamento usuale degli assi cartesiani.

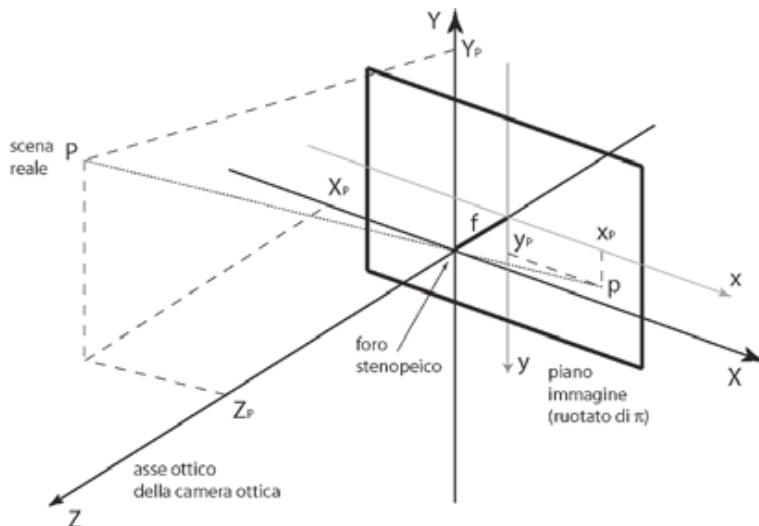


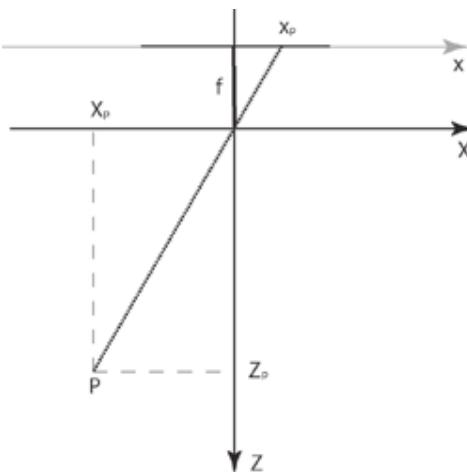
Illustrazione della geometria della camera ottica

Il foro stenopeico, attraverso il quale passano tutte le linee di proiezione, si considera infinitamente piccolo.

In una camera ottica reale non potrebbe avere dimensioni inferiori a certi limiti - finiti - sia per lasciare passare abbastanza luce per gli usi pratici, sia per evitare fenomeni di diffrazione luminosa. Tale punto coincide con l'origine  $O$  del sistema di coordinate cartesiane tridimensionali  $(X, Y, Z)$  usate per individuare i punti  $P$  della scena e viene chiamato anche con il termine punto focale della camera (*camera focal point*) o centro della camera (*camera center*).

La figura in questa pagina illustra la configurazione geometrica usata per ricavare le coordinate  $(x, y)$  del punto proiezione  $p$  a partire dalle coordinate tridimensionali  $(X, Y, Z)$  del punto della scena  $P$ .

Tale configurazione è ricavata da quella complessiva illustrata nella figura precedente tramite proiezione su un piano parallelo a quello definito dagli assi  $X$  e  $Z$ .



Geometria semplificata del modello di camera ottica (vista dall'asse  $Y$ ).

Dall'analisi della figura si possono ricavare la seguenti relazioni:

- 1a)  $x_p = -\frac{fX_p}{Z_p}$
- 1b)  $y_p = -\frac{fY_p}{Z_p}$

Le relazioni 1a) e 1b) possono essere compattate nella pratica notazione matriciale seguente:

$$2) \quad \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \frac{f}{Z_p} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \end{bmatrix}$$

La mappatura dalla scena 3D sul piano immagine 2D operata dalla camera è una proiezione prospettica seguita da una rotazione di  $180^\circ$  del piano immagine attorno all'asse ottico (è come se si osservasse l'immagine sul retro del sensore ponendosi a testa in giù).

Le dimensioni degli oggetti proiettati dipendono dalla loro distanza dalla camera (per la precisione dal valore della coordinata  $Z_p$ ) e dalla lunghezza focale  $f$ .

Per ottenere direttamente immagini non ruotate si può:

- ruotare il sistema di coordinate locali di  $180^\circ$ , posizione adottata nei paragrafi precedenti tramite il particolare orientamento degli assi  $x$  e  $y$ .
- considerare non un piano immagine reale, che è posizionato in  $Z = -f$ , bensì un piano immagine virtuale posizionato in  $Z = +f$ .

Tale posizione semplifica alcune formulazioni geometriche ed è utilizzata nell'ambito della geometria epipolare.

### **Relazioni tra sensore e schermo del display**

---

L'immagine proiettata sul sensore in fase di ripresa viene rovesciata e proiettata sullo schermo del display in fase di visualizzazione.

Non ne viene modificata la forma, ma l'immagine è sempre ingrandita di un fattore  $M$  pari al rapporto tra la lunghezza della diagonale  $D$  dello schermo del display e quella della diagonale  $d$  del sensore:

$$M = \frac{D}{d} \quad 2)$$

Quindi le coordinate definite sullo schermo differiscono da quelle definite sui sensori dello stesso fattore moltiplicativo:

$$x_{Sd,s} = M x_{d,s} \quad 3a)$$

$$y_{Sd,s} = M y_{d,s} \quad 3b)$$

Per rendersi conto del valore di tale fattore moltiplicativo si consideri che i televisori HDTV o "HD ready" attuali sono dotati di schermi la cui diagonale varia tra i 32" e i 50", mentre i sensori delle telecamere professionali hanno diagonale pari a 1/2" o 3/4".

Quindi il fattore  $M$  ha un valore che varia, orientativamente, tra 43 e 100.

### **La geometria di visualizzazione**

---

Nell'introduzione si sono riportate le basi della tecnica stereoscopica e si riportano relazioni che legano la disparità orizzontale con la percezione della distanza del punto visualizzato.

In questo paragrafo si completa tale derivazione aggiungendo anche le altre dimensioni; si introducono infatti le relazioni che legano la posizione percepita del punto visualizzato con la sua ubicazione nello spazio tridimensionale della scena.

Il modello geometrico adottato per queste derivazioni è illustrato in figura 3a.

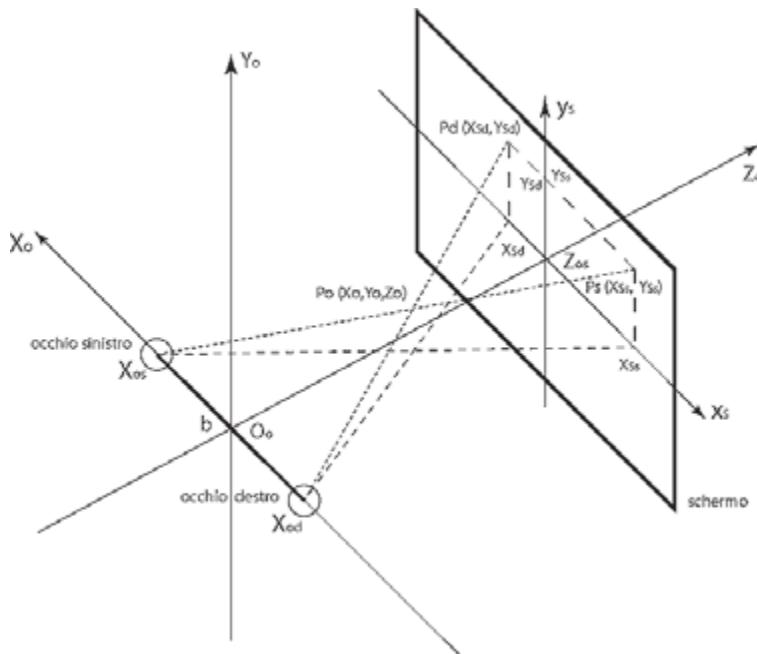


Fig. 3a – Sistema di riferimento per l'analisi geometrica della configurazione di visione.

La figura 3b riporta la proiezione, su un piano parallelo al piano  $X_oZ_o$ , della configurazione tridimensionale di visione illustrata in figura 3a.

In figura 3b sono illustrati i due casi tipici.

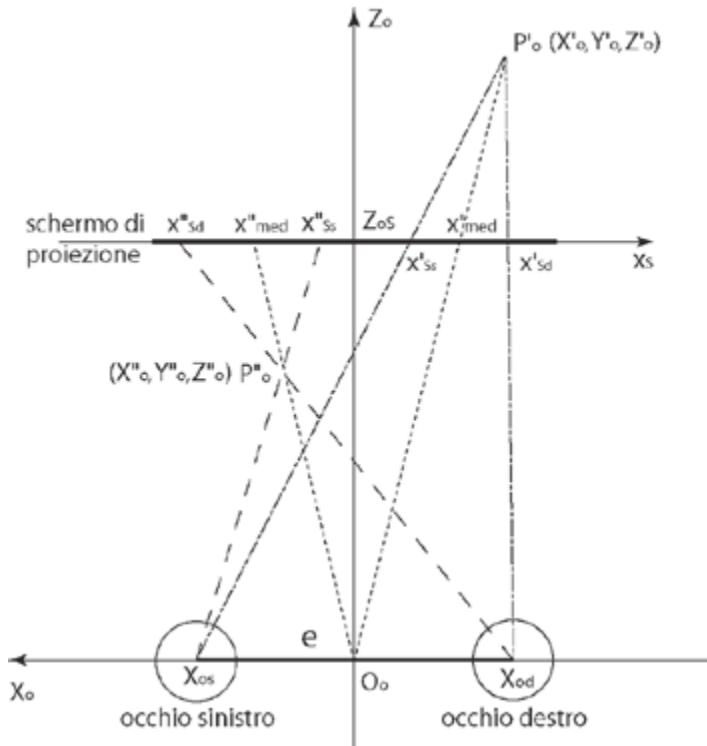


Fig. 3b – Proiezione su un piano orizzontale della configurazione di visione.

Nel caso in cui  $x_{sd} > x_{ss}$ , cioè quando il punto proiettato relativo all'occhio destro si trova più a destra di quello relativo all'occhio sinistro, si ha *disparità omonima*, e il punto visualizzato viene percepito "al di là" dello schermo (caso contrassegnato dall'apice singolo).

Al contrario, se  $x_{sd} < x_{ss}$ , cioè quando il punto proiettato relativo all'occhio sinistro si trova più a destra di quello relativo all'occhio destro, si ha *disparità crociata*.

Gli assi ottici oculari si incrociano in un punto posizionato tra l'osservatore e lo schermo e il punto visualizzato viene conseguentemente percepito "al di qua" dello schermo (caso contrassegnato da ")).

La coordinata  $Z_o$  dei punti  $P'$  e  $P''$  è data dalla seguente relazione, già derivata nell'introduzione, che risulta valida per ambedue i tipi di disparità per cui si tralasciano gli apici:

$$Z_o = Z_{os} \frac{e}{e - \delta_{so}} \quad 5a)$$

Dove:

- $Z_o$  è la distanza del punto percepito
- $Z_{os}$  è la distanza dello schermo dall'osservatore
- $e = X_{os} - X_{od}$  è la distanza interoculare dell'osservatore
- $\delta_{so} = x_{sd} - x_{ss}$  è la *disparità orizzontale* valutata tra punti immagine proiettati sullo schermo.

Si noti che la distanza del punto percepito dall'osservatore dipende solamente dalla distanza tra schermo e osservatore e dalla disparità orizzontale presentata dai punti proiettati sullo schermo, a prescindere dalla sua dimensione.

La coordinata  $X_o$  del punto percepito è data dalla seguente relazione, ricavata per similitudine tra i triangoli  $O_oZ_oP_o$  e  $O_oZ_oS_{xmed}$ , dove  $x_{med}$  è il punto medio tra  $x_{sd}$  e  $x_{ss}$ .

Il segno meno tiene conto dell'opposto verso degli assi  $X_o$  e  $x_s$ .

Anche in questo caso la relazione vale per i due tipi di disparità per cui si tralasciano gli apici:

$$X_o = -X_{med} \frac{e}{e - \delta_{so}} \quad 5b)$$

Dove:

- $X_o$  è l'ascissa del punto percepito
- $x_{med} = \frac{x_{sd} - x_{ss}}{2}$  è il punto medio tra  $x_{sd}$  e  $x_{ss}$
- $e = X_{os} - X_{od}$  è la distanza interoculare dell'osservatore
- $\delta_{so} = x_{sd} - x_{ss}$  è la *disparità orizzontale* valutata tra punti immagine proiettati sullo schermo.

Si noti che l'ascissa del punto percepito non dipende dalla distanza tra schermo e osservatore, ma dall'ubicazione dei punti proiettati sullo schermo e dalla loro disparità orizzontale.

La coordinata  $Y_o$  è data da una relazione che si ricava facilmente tramite la similitudine di triangoli (vedi la figura 3c).

Anche in questo caso la relazione vale per i due tipi di disparità per cui si tralasciano gli apici:

$$Y_o = y_s \frac{e}{e - \delta_{so}} = \frac{y_{sd} + y_{ss}}{2} \frac{e}{e - \delta_{so}} \quad 5c)$$

Dove:

- $y_o$  è l'ordinata del punto percepito
- $y_s$  è l'ordinata del punto proiettato rispetto allo schermo
- $e = X_{os} - X_{od}$  è la distanza interoculare dell'osservatore
- $\delta_{so} = x_{sd} - x_{ss}$  è la *disparità orizzontale* valutata tra punti immagine proiettati sullo schermo.

Si noti che, analogamente all'ascissa, anche l'ordinata del punto percepito non dipende dalla distanza tra schermo e osservatore, ma solo dall'ubicazione dei punti proiettati sullo schermo e dalla loro disparità orizzontale.

In caso di presenza di disparità verticale,  $y_s$  viene valutato, per convenzione, come valor medio delle ordinate dei due punti proiettati:

$$y_s = \frac{y_{sd} + y_{ss}}{2}$$

Si noti che in questo caso la visione può essere disturbata e potrebbe essere difficoltoso posizionare esattamente il punto percepito.

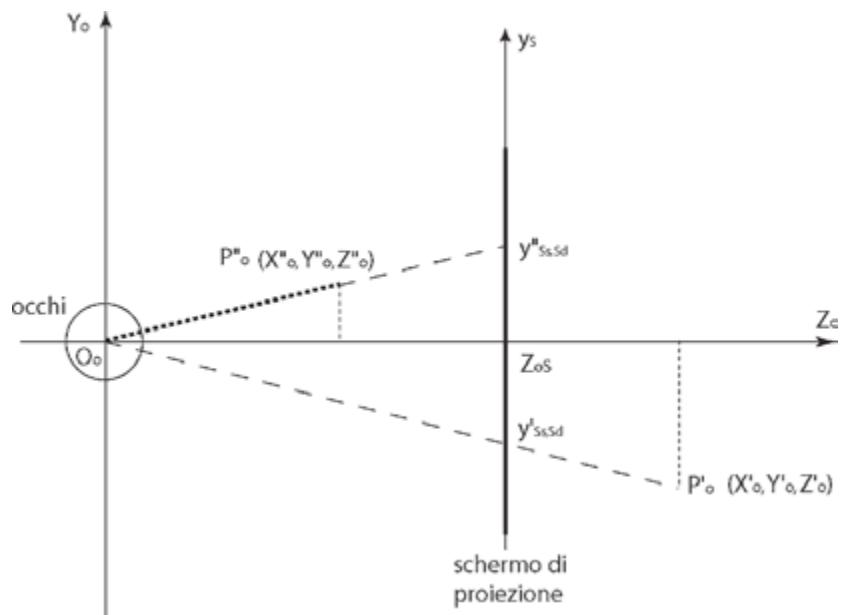


Fig. 3c - Proiezione su un piano verticale della configurazione di visione.

## La geometria di ripresa

### Aassi ottici paralleli

Questa configurazione, illustrata in figura 4a, è caratterizzata dal fatto che gli assi ottici delle telecamere sono paralleli. Per motivi di semplicità costruttiva e operativa si fa in modo che i sensori siano anche complanari.

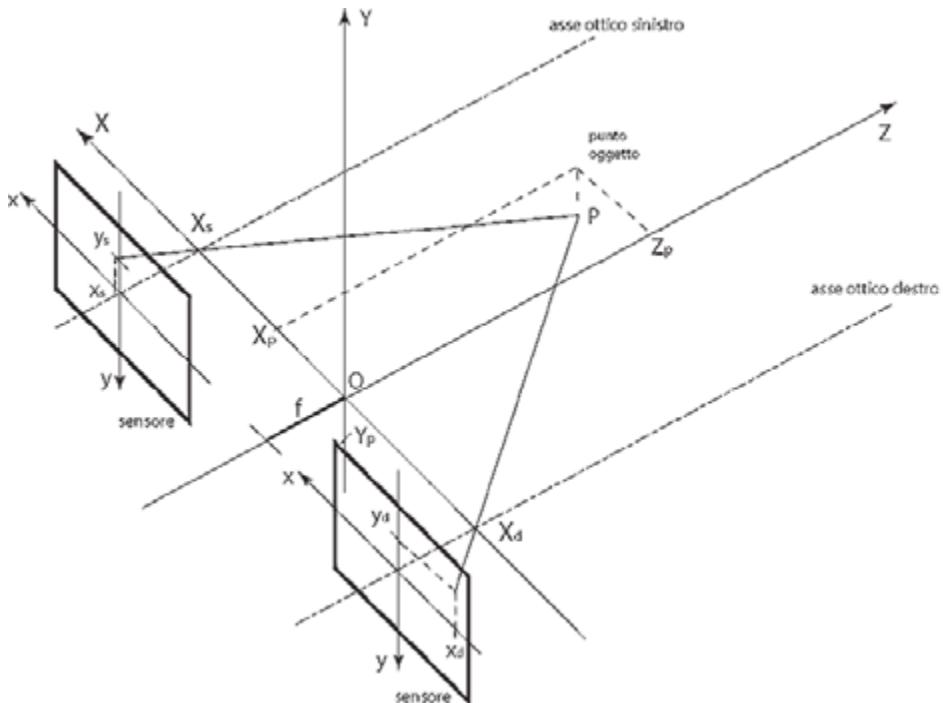


Fig. 4a – Configurazione ad assi ottici paralleli

La principale caratteristica per quanto riguarda la stereoscopia deriva direttamente dalla specifica configurazione geometrica.

Gli assi ottici, essendo rette parallele, "convergono all'infinito"; per questo motivo, i punti della scena ripresa posti "all'infinito" - in pratica ad una distanza relativamente grande dalle telecamere - formano sui due sensori immagini ubicate nelle stesse posizioni (relativamente alle coordinate locali dei sensori  $x, y$ ).

In fase di visualizzazione, sullo schermo del display tali punti risultano coincidenti e quindi vengono percepiti "sullo schermo" (cioè alla stessa distanza dello schermo stesso).

In linea di massima questo effetto è da considerarsi un difetto perché limita allo spazio tra lo schermo e lo spettatore il volume in cui è possibile far percepire gli oggetti visualizzati.

Pertanto, con una facile operazione di postproduzione, è possibile inserire una traslazione orizzontale per minimizzare tale problema e utilizzare anche lo spazio percepito "al di là" dello schermo.

L'analisi del modello geometrico della configurazione di ripresa, utilizzando il modello delle camera oscura al posto delle telecamere, permette di definire le relazioni intercorrenti tra la posizione spaziale del generico punto oggetto, cioè dei punti costituenti la superficie degli oggetti ripresi, e le coordinate del punto immagine, cioè della sua proiezione sulla superficie dei sensori.

Le coordinate  $(x_d, y_d)$ ,  $(x_s, y_s)$  dei punti immagine proiettati sui sensori sono ricavate dalle seguenti relazioni, nota la posizione  $(X_p, Y_p, Z_p)$  del relativo punto oggetto:

- 6a)  $x_d = -\frac{f}{Z_p} \left( X_p + \frac{b}{2} \right)$
- 6b)  $x_s = -\frac{f}{Z_p} \left( X_p - \frac{b}{2} \right)$
- 7a)  $y_d = -\frac{f}{Z_p} Y_p$
- 7b)  $y_s = -\frac{f}{Z_p} Y_p$

Dove le grandezze usate sono state illustrate nel paragrafo precedente e sono riportate anche nelle figure 4a, 4b e 4c che illustrano la configurazione di ripresa ad assi ottici paralleli sotto diversi punti di vista.

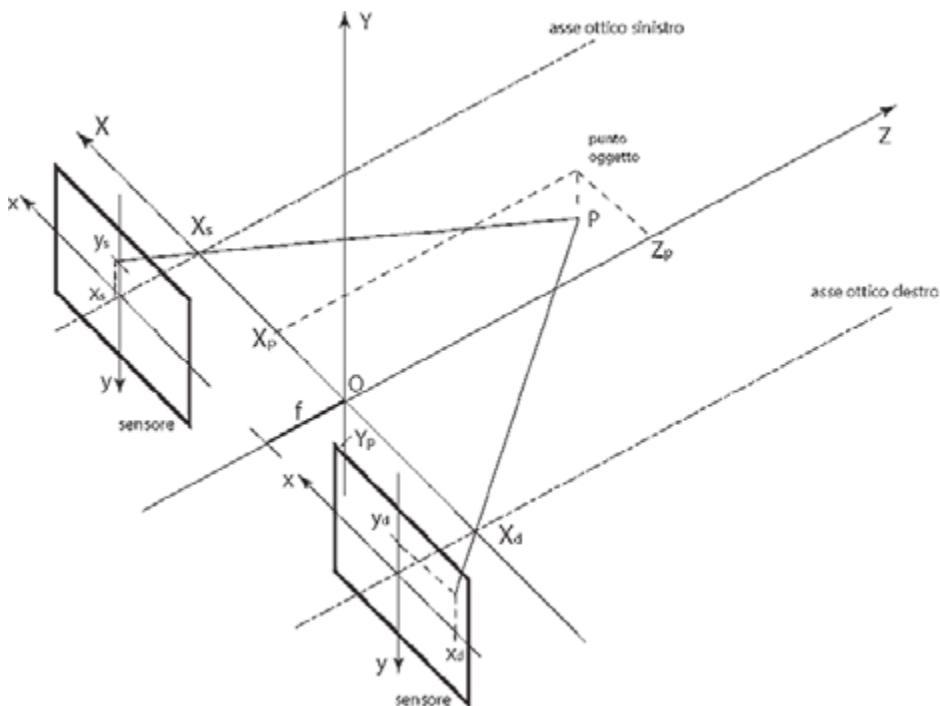


Fig. 4a – Configurazione ad assi ottici paralleli

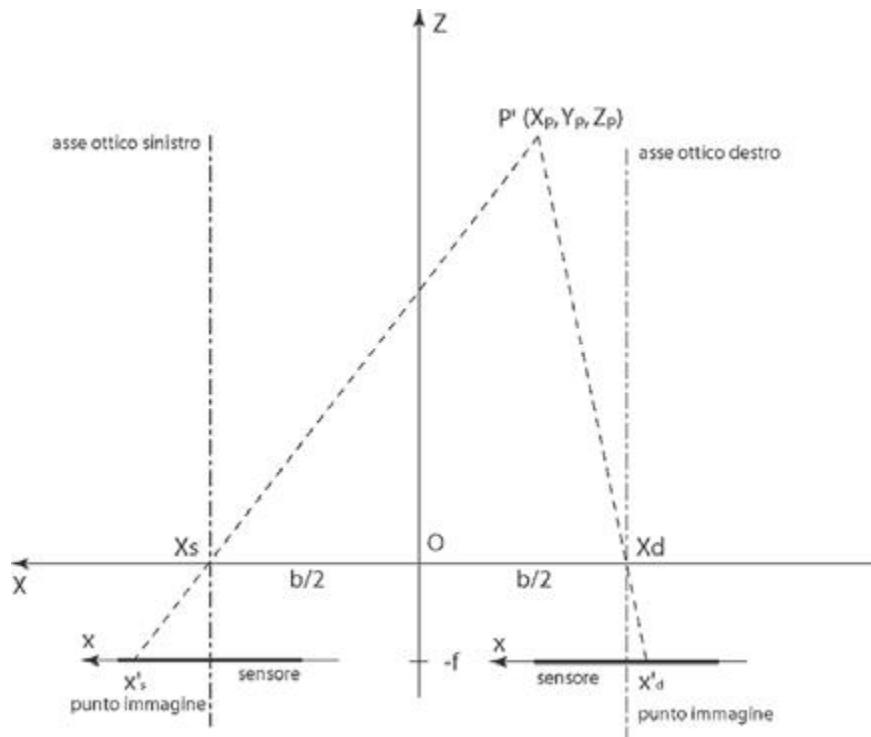


Fig. 4b – Configurazione ad assi ottici paralleli - vista dall'alto

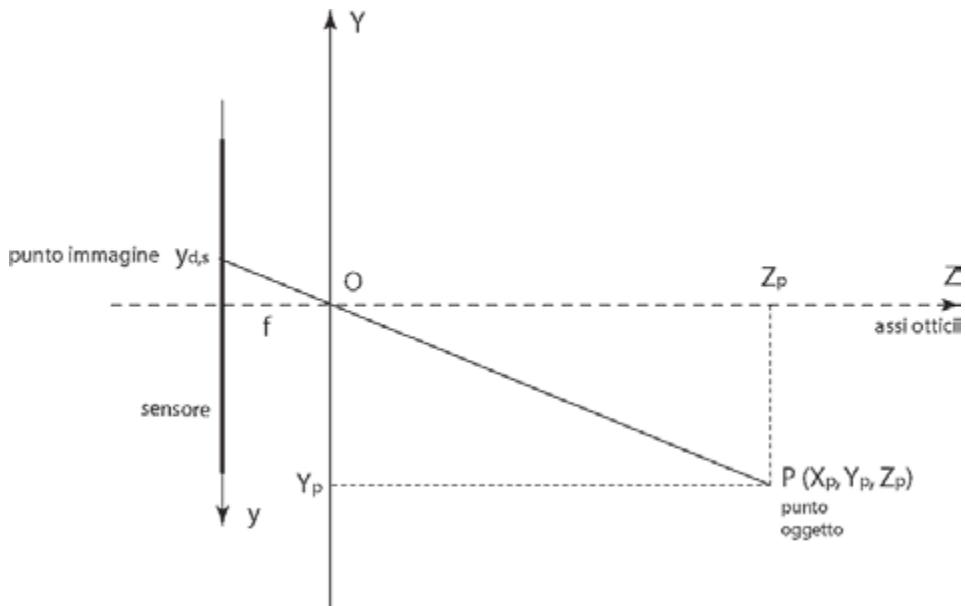


Fig. 4c – Configurazione ad assi ottici paralleli - vista di lato

Dalla figura 4c emerge che le proiezioni sui due sensori di ogni punto della scena presentano le stesse coordinate rispetto alle coordinate locali.

Pertanto non si generano disparità verticali dovute alla conversione tra lo spazio reale e le immagini stereoscopiche proiettate sui sensori; questa caratteristica positiva è uno dei principali punti di forza di questa configurazione di ripresa.

In particolare, le disparità orizzontale e verticale,  $\delta_o$  e  $\delta_v$ , sono date dalle seguenti relazioni:

- 8a)  $\delta_o = x_d - x_s = -\frac{f}{Z_p} b$
- 8b)  $\delta_v = y_d - y_s = 0$

dalla cui analisi emergono due interessanti conseguenze:

- dalla relazione 8a) si evince che, data una lunghezza focale  $f$  e una base stereoscopica  $b$ , la disparità orizzontale  $\delta_o$  dipende solamente dal valore della coordinata  $Z_p$  del punto oggetto, in prima approssimazione uguale alla distanza dell'oggetto dalle telecamere.  
La lunghezza focale  $f$  funge da fattore moltiplicativo in proporzione diretta con la disparità.
- dalla relazione 8b) emerge che in questa configurazione di ripresa, nei limiti dei modelli usati, ed in particolare in assenza di distorsioni dovute alle ottiche, non si generano disparità verticali.

### Assi ottici convergenti

Questa configurazione, illustrata in figura 5a, è caratterizzata dal fatto che gli assi ottici delle telecamere si intersecano in un punto (*punto di convergenza*), similmente agli assi ottici oculari, che convergono verso il punto di fissazione.

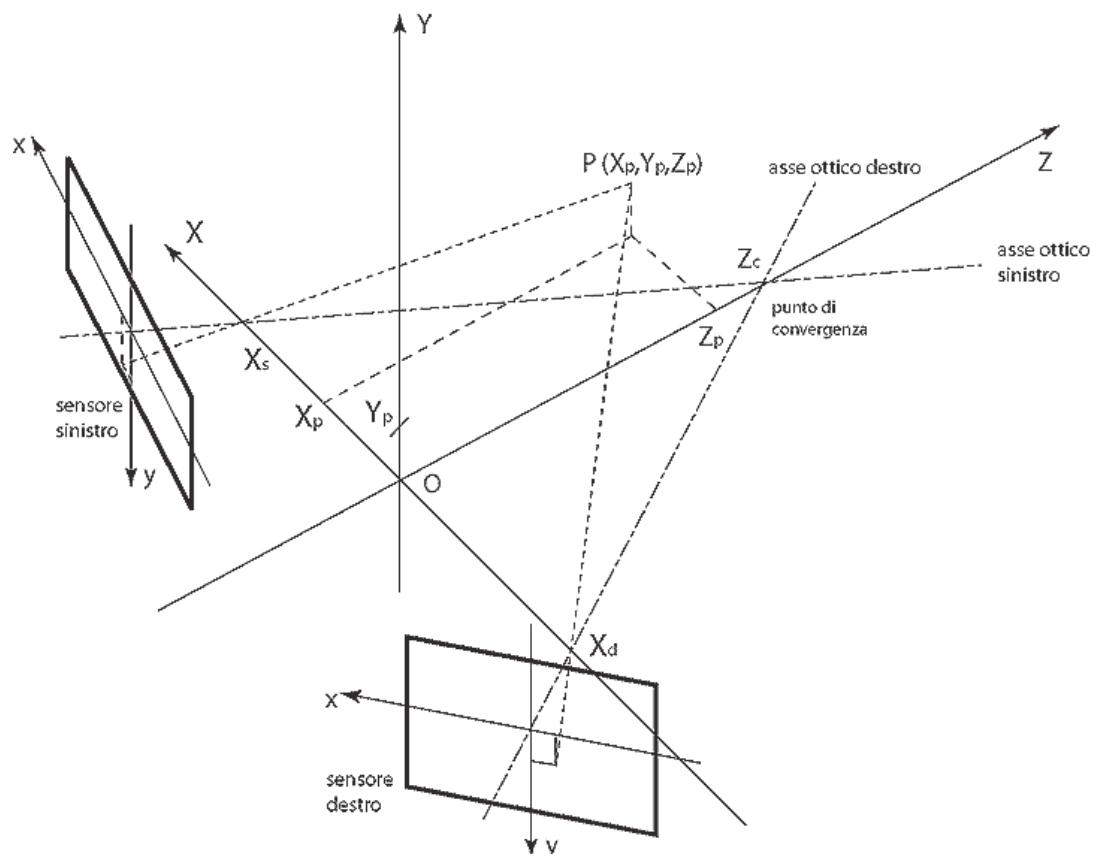


Fig. 5a - Configurazione ad assi ottici convergenti.

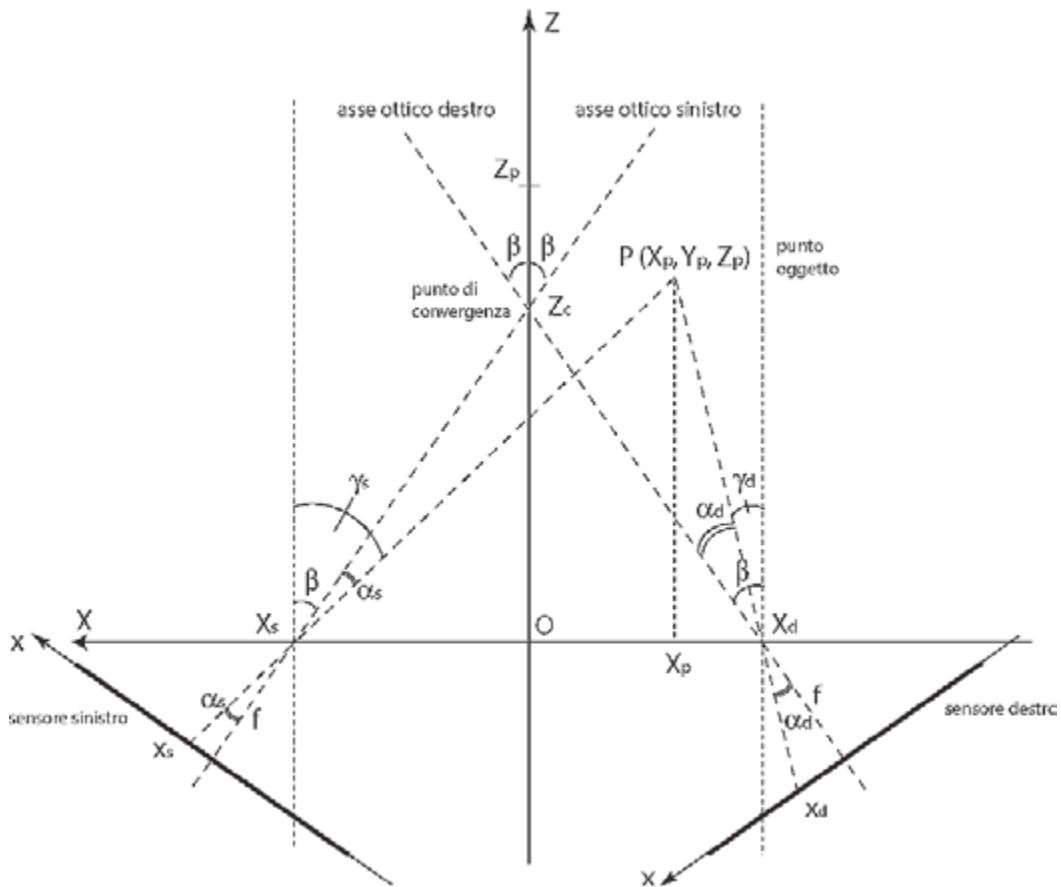


Fig. 5b - Configurazione ad assi ottici convergenti – vista dall’alto.

Per motivi di semplicità costruttiva e operativa si fa in modo che i sensori siano simmetrici rispetto al piano  $ZY$ ; pertanto li si fa ruotare, in verso opposto, attorno al proprio asse verticale della stessa quantità, l’angolo di convergenza  $\beta$ .

Questo è definito come l’angolo compreso tra l’asse ottico e l’asse  $Z$ ; è positivo se misurato in senso antiorario per la camera destra e in senso orario per la camera sinistra.

Si assume che abbia sempre valore positivo o nullo (il questo caso gli assi ottici risultano paralleli).

Analogamente al caso precedente, l’analisi del modello geometrico della configurazione di ripresa, utilizzando il modello delle camera oscura al posto delle telecamere, permette di definire le relazioni intercorrenti tra la posizione spaziale del generico punto oggetto, cioè dei punti costituenti la superficie degli oggetti ripresi, e le coordinate del punto immagine, cioè della sua proiezione sulla superficie dei sensori.

Dall’analisi della figura 5b, che illustra la configurazione in oggetto vista dall’alto, si ricavano le relazioni 9a) e 9b) che danno le ascisse dei punti immagine rispetto al sistema di coordinate dei sensori:

$$x_d = -f \cdot \operatorname{tg} \left( \operatorname{arctg} \frac{X_p + \frac{b}{2}}{Z_p} - \beta \right) \quad 9a)$$

$$x_s = -f \cdot \operatorname{tg} \left( \operatorname{arctg} \frac{X_p - \frac{b}{2}}{Z_p} + \beta \right) \quad 9b)$$

Mentre per ricavare le relazioni 10a) e 10b), valide per le ordinate dei punti immagine, conviene analizzare la figura 5a:

$$y_d = \frac{fY_p}{Z_p} \frac{\cos\left(\arctg \frac{X_p + \frac{b}{2}}{Z_p}\right)}{\cos\left(\arctg \frac{X_p + \frac{b}{2}}{Z_p} - \beta\right)} = \frac{fY_p}{Z_p \cos \beta + \left(X_p + \frac{b}{2}\right) \sin \beta} \quad 10a)$$

$$y_s = \frac{fY_p}{Z_p} \frac{\cos\left(\arctg \frac{X_p - \frac{b}{2}}{Z_p}\right)}{\cos\left(\arctg \frac{X_p - \frac{b}{2}}{Z_p} + \beta\right)} = \frac{fY_p}{Z_p \cos \beta - \left(X_p - \frac{b}{2}\right) \sin \beta} \quad 10b)$$

Dove:

- $X_p, Y_p, Z_p$  sono le coordinate del punto oggetto considerato
- $f$  è la lunghezza focale delle camere
- $b$  è la base stereoscopica
- $\beta = \arctg\left(\frac{b}{2Z_c}\right)$  è l'angolo di convergenza degli assi ottici.

Si noti che ponendo  $\beta = 0$ , cioè annullando il valore dell'angolo di convergenza, in altre parole ponendo gli assi ottici paralleli, le relazioni 9a,b) e 10a,b) si riducono rispettivamente alle 6a,b) e 7a,b), ricavate nel punto precedente.

Le disparità orizzontale e verticale (sul sensore),  $\delta_o$  e  $\delta_v$ , sono date dalle seguenti relazioni:

$$\delta_o = -f \left( \tg\left[\arctg \frac{X_p + \frac{b}{2}}{Z_p} - \beta\right] - \tg\left[\arctg \frac{X_p - \frac{b}{2}}{Z_p} + \beta\right] \right) \quad 11a)$$

$$\delta_v = -\frac{2fY_p X_p \sin \beta}{[Z_p \cos \beta + \left(X_p + \frac{b}{2}\right) \sin \beta][Z_p \cos \beta - \left(X_p - \frac{b}{2}\right) \sin \beta]} \quad 11b)$$

Dalla cui analisi emerge che:

- Se l'angolo di convergenza è nullo,  $\beta = 0$ , ovvero si è in configurazione ad assi ottici paralleli, si ottiene:  $\delta_o = \frac{-fb}{Z_p}$  e  $\delta_v = 0$ , cioè le relazioni 8a) e 8b), come è lecito aspettarsi.
- La convergenza degli assi ottici delle camere implica l'ingenerarsi di una disparità verticale dovuta alla distorsione trapezoidale (in inglese: *key-stone distortion*) che si verifica quando si inclina il sensore rispetto alla scena ripresa.

Il suo valore è nullo:

- per ogni punto della scena ripresa giacente sul piano ZX ( $Y_p = 0$ )
- per ogni punto della scena ripresa giacente sul piano ZY ( $X_p = 0$ )
- per ogni punto posto “all’infinito” ( $Z_p \rightarrow \infty$ )
- La disparità verticale risulta direttamente proporzionale alla lunghezza focale,  $f$ , e all’ordinata del punto,  $Y_p$ , e dipende in modo non banale dalla posizione del punto ripreso (coordinate  $X_p$  e  $Z_p$ ), dalla base stereoscopica  $b$  e dall’angolo di convergenza degli assi ottici delle camere  $\beta$ .

## Conclusioni

La configurazione di ripresa ad assi ottici paralleli è interessante per via della semplicità costruttiva dell'apparato di ripresa e per la semplicità della formalizzazione delle sue proprietà geometriche. Ha inoltre proprietà interessanti soprattutto per quanto riguarda l'eliminazione di alcune distorsioni, tema che verrà trattato in un prossimo articolo.

Presenta alcune limitazioni per quanto riguardano lo spazio percepito in fase di visualizzazione, che si possono superare con un adeguato – e semplice – intervento in fase di post produzione; in particolare, è necessario traslare orizzontalmente le immagini costituenti la coppia stereoscopica.

Tuttavia, la configurazione ad assi ottici convergenti è quella che più si avvicina alla geometria della visione umana, e presenta interessanti proprietà, quale quella di far percepire l'elemento di interesse sullo schermo, con parte della scena visualizzata "al di qua" dello schermo e parte "al di là" di esso. Sfrutta quindi in modo nativo tutto lo spazio percepibile in fase di visualizzazione, senza richiedere operazioni di post processing.

Tuttavia è affetta da alcune distorsioni, inevitabili proprio perché derivanti dalla geometria stessa della configurazione di ripresa.

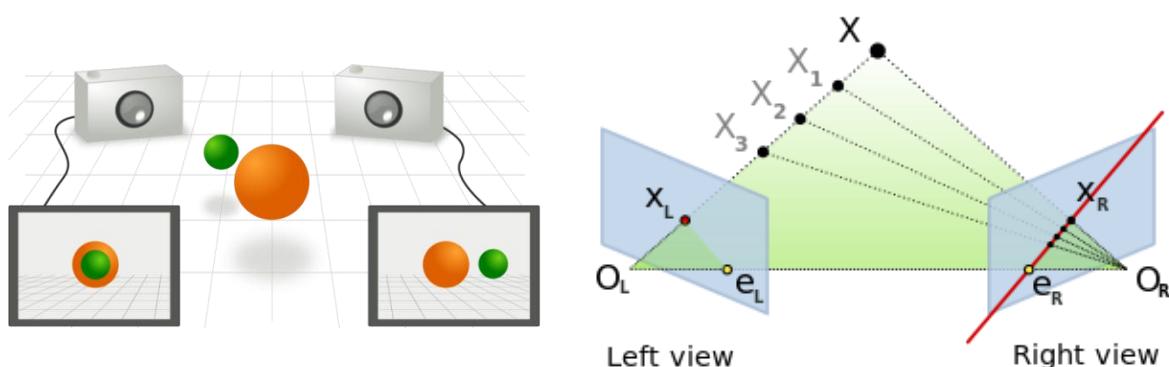
Tutti i pro e i contro delle diverse configurazioni di ripresa non sono facilmente valutabili sulla carta, soprattutto per quanto riguarda la tolleranza dell'osservatore rispetto alle eventuali distorsioni.

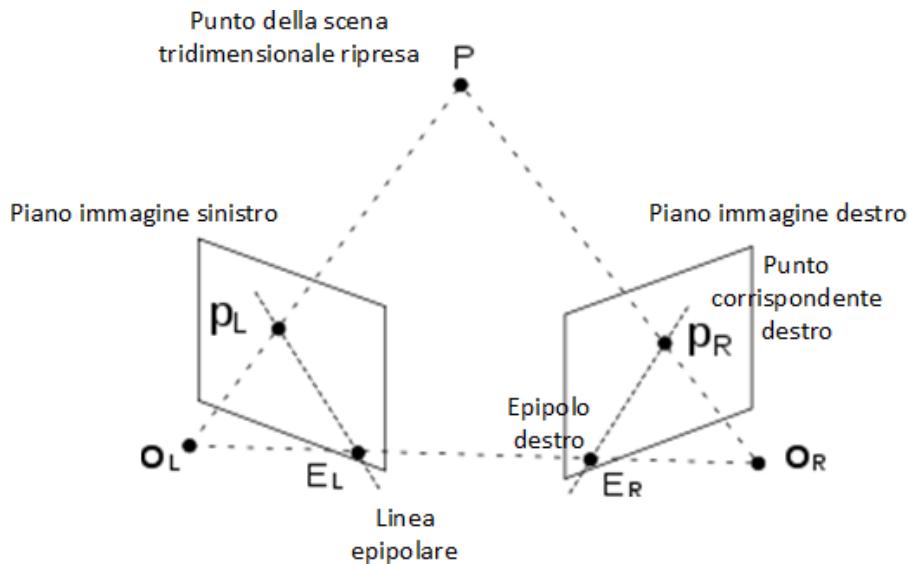
Per questo motivo, è prevista una campagna di sperimentazione in cui si effettueranno riprese con un sistema configurabile nelle due modalità, variandone i parametri caratteristici, al fine di individuare le modalità di ripresa ottimali nelle varie situazioni.

## Elementi di geometria epipolare

Dati due piani immagine, a distanza  $f$  dai fuochi  $O_L$  e  $O_R$ , cerchiamo di capire come si muovono le proiezioni dei punti della scena sui due piani immagine: tracciamo un raggio che parte da  $O_L$  e, passando per il punto  $X_L$  del piano immagine, entra nella scena: incrementando il valore di  $z$  possiamo trovare infiniti punti a diversa profondità, ciascuno dei quali viene proiettato in  $X_L$ .

Per necessità futura chiamiamo questo insieme di punti  $ZL_{inf}$ .





Lavorando con una sola camera possiamo notare che passando dalla scena tridimensionale al piano immagine si perde l'informazione della profondità e di conseguenza il senso di tridimensionalità.

Tale senso di tridimensionalità è fornito dal secondo sensore  $O_R$ : tracciando i raggi che partono da  $O_R$  e intersecano i punti nell'insieme  $ZL_{inf}$  determiniamo le proiezioni di questi ultimi sul piano immagine di destra.

Le proiezioni coinvolgono un pixel diverso per ciascun punto in  $ZL_{inf}$ , facendo rientrare dalla finestra il senso di tridimensionalità perso con una sola camera.

Naturalmente la bicameralità non permette di vedere tutti gli oggetti presenti nella scena, poiché alcuni possono essere occlusi alla vista qualsiasi sia la camera con cui li si guarda.

Fatta questa breve introduzione, definiamo come **epipoli** i due punti  $E_L$  e  $E_R$ : l'epipolo  $E_L$  è la proiezione del centro di proiezione  $O_R$  sul piano immagine di sinistra, cioè l'intersezione fra la retta che congiunge le due camere e il piano immagine di sinistra; viceversa vale per l'epipolo di destra  $E_R$ .

La linea epipolare  $EL_L$  è definita come la retta che unisce l'epipolo  $E_L$  alla proiezione di un punto  $P$  sul piano immagine di sinistra (il piano immagine dell'epipolo).

Su  $EL_L$  troviamo le proiezioni dei punti  $ZR_{inf}$ , punti che si trovano sulla retta che congiunge  $P$  alla camera  $O_R$  a diverse profondità  $z$ .

Viceversa vale per la linea epipolare di destra  $EL_R$  (segnata in rosso).

Definiamo per completezza il piano epipolare come quello su cui sono posati i punti  $X$ ,  $X_L$  e  $X_R$ .

La geometria epipolare viene usata per capire la corrispondenza fra i pixel del piano immagine destro e sinistro che identificano lo stesso punto  $P$  nella scena, infatti se conosco le posizioni delle camere posso capire la corrispondenza fra i pixel e se conosco la corrispondenza fra i pixel posso conoscere le posizioni delle camere.

## Proiezione da 3D a 2D

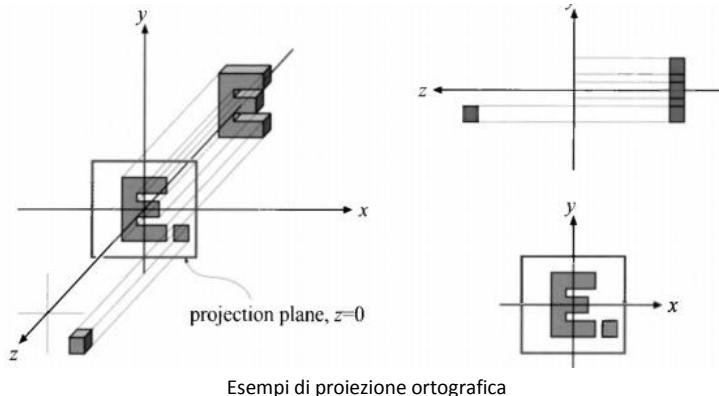
Affrontiamo ora la proiezione dal punto di vista puramente geometrico, definendola come la trasformazione dei punti della scena, per portarli tutto su uno stesso piano a  $z$  fissata.

I metodi proiettivi sono principalmente due: proiezione ortografica e prospettica.

### Proiezione ortografica

La proiezione ortografica è detta anche proiezione parallela: ciascun punto della scena è proiettato tramite raggi paralleli fra loro.

Questo metodo ha il vantaggio di mantenere il parallelismo fra le rette e le distanze fra i punti.



Esempi di proiezione ortografica

Il metodo più semplice per realizzarla è pensare di proiettare i punti sul piano  $z = 0$  attraverso la seguente matrice di trasformazione  $P_0$  che elimina la componente  $z$  dei punti:

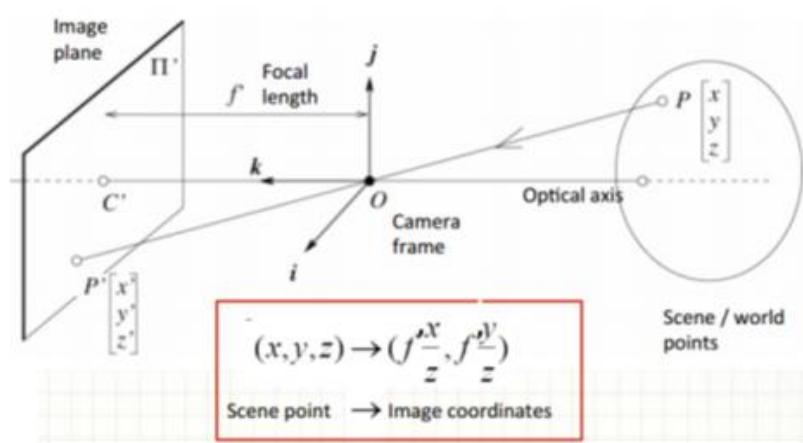
$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Questo tipo di trasformazione però non è invertibile (visto che la matrice ha determinante = 0), pertanto l'informazione sulla profondità dei punti è persa per sempre.

La proiezione ortografica quindi fa perdere completamente il senso di profondità, anche se mantiene le dimensioni fisiche degli oggetti della scena.

### Proiezione prospettica

Passiamo ora alla proiezione prospettica, come vista nelle sezioni precedenti nel modello geometrico della camera oscura. Per completezza riportiamo il modello geometrico visto in precedenza in una configurazione leggermente diversa in figura:



Modello geometrico della proiezione prospettica

Consideriamo un punto  $P$  di cui vogliamo conoscere la proiezione  $P'$ , grazie ad un raggio proiettivo passante per il punto  $O$  distante  $f'$  (distanza focale) dal piano immagine.

Ricordiamo che se il piano immagine è finito non può contenere le proiezioni di tutti gli oggetti della scena ma ne vedrà solo alcuni.

Per determinare le coordinate di  $P'$  possiamo usare le formule viste in precedenza oppure usiamo una matrice di mapping, che rimappa le tre componenti di  $P$ , mantenendo  $x$  e  $y$  uguali e determinando

$$z' = \frac{z}{f'}$$

Consideriamo queste come coordinate omogenee e passiamo da coordinate a 3 componenti a coordinate a 2 componenti dividendo le prime due componenti per la terza.

Il risultato di tale operazione è mostrato in figura.

Da notare che con questo modello si mantiene una certa informazione sulla profondità, anche se non si può ricavare la vera distanza degli oggetti dalla camera, visto che la matrice potrebbe essere riscalata da un fattore di moltiplicazione qualsiasi.

## Proiezione prospettica e calibrazione

---

Il modello prospettico qui presentato è lontano dalla realtà perché è sempre necessario considerare le caratteristiche della camera che si usa, valutandone i suoi parametri intrinseci ed estrinseci.

Consideriamo come parametri estrinseci quelli che riguardano la relazione fra la camera e il mondo esterno, distinguibili da quelli intrinseci che riguardano invece le caratteristiche della camera stessa.

Come parametri estrinseci da tenere in considerazione ci sono:

- la camera deve essere centrata in  $(0,0,0)$ ; se così non accade è necessario effettuare delle traslazioni
- la camera deve essere allineata con gli assi; se così non accade è necessario effettuare delle rotazioni.

Fra quelli intrinseci elenchiamo invece:

- la camera mantiene le proporzioni di unità rispetto al mondo esterno
- il centro ottico del piano immagine è in posizione  $(0,0)$
- non c'è rotazione interna alla camera ("skew"); ciò può accadere nel caso in cui il sensore sia mal saldato nella camera e posizionando la camera dritta, l'immagine che ne deriva è storta.

$$\begin{aligned} x &= K[R \ t]X \\ w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{aligned}$$

Tutti questi parametri vengono utilizzati per generare due matrici associate alla camera e ai suoi parametri intrinseci ed estrinseci, in cui:

- $u_0, v_0$  vengono usati per traslare il centro ottico in  $(0,0)$
- $\alpha, \beta$  vengono usate per mantenere le proporzioni di unità
- $s$  viene usato per gestire l'eventuale "skew"
- $t_x, t_y, t_z$  vengono usati per la traslazione della camera in  $(0,0,0)$
- $r_{11} \dots r_{33}$  vengono usati per la rotazione della camera in modo che sia allineata agli assi.

I parametri intrinseci sono dati dalla camera stessa, mentre quelli estrinseci ci sono sconosciuti in fase di acquisizione.

La matrice che considera i parametri è una  $3 \times 4$ , formata da 12 valori da fissare; 11 di questi sono i gradi di libertà fissati dai parametri intrinseci ed estrinseci elencati sopra, l'ultimo è un'informazione sulla profondità: questa informazione non è ricavabile dall'immagine, perciò scopriamo che la matrice può essere soggetta ad uno scalamento qualsiasi che non ne modifica le caratteristiche.

Alternativamente si può usare una matrice  $4 \times 4$ , che matrice può essere usata per mappare dal mondo 3D alle coordinate omogenee del piano immagine della proiezione, includendo questa volta anche la disparità, cioè  $x_s$ .

Usare la  $4 \times 4$  è utile perché abbiamo una funzione della disparità della distanza e quindi anche nel piano immagine abbiamo l'informazione della profondità che di solito perdiamo in fase di proiezione.

Grazie a questa informazione possiamo invertire la matrice di proiezione e ritrovare la posizione del punto acquisito nella scena.

L'informazione della profondità è acquisibile solo se abbiamo a disposizione un sensore di profondità (tipo quello fissato sul Kinect).

Oltre a ritrovare la posizione del punto nello spazio abbiamo la possibilità di passare dai punti di un piano immagine a quelli di un altro, calcolando i primi come funzione dei secondi (dati i secondi) e viceversa.

---

## Applicazioni della stereoscopia

---

### Image-based rendering

Una delle principali applicazioni della stereoscopia è di estendere il rendering derivante dalla grafica sintetica aggiungendo elementi reali, o, viceversa, partire da immagini reali ed estenderle con elementi renderizzati sinteticamente.

Nel primo caso, tramite la computer graphics genero il modello geometrico e luminoso della scena, modello una camera sintetica e genero un'immagine realistica.

Modellando una seconda camera si può generare un rendering stereoscopico.

Nel secondo caso, in ambiti di computer vision, parto da una scena reale e con camere reali cerco di definire un modello geometrico.

È necessario usare almeno 2 camere per ricostruire la posizione nello spazio tridimensionale di ciascun punto, sulla base della posizione del punto sui due piani immagine.

I due casi mostrati possono essere combinati per generare applicazioni di realtà aumentata: costruiamo il modello a partire da una scena reale, la modifichiamo sinteticamente aggiungendo elementi grafici a piacere e poi sintetizziamo l'immagine (o le immagini in situazioni di stereoscopia) per falsare la realtà acquisita.

I problemi di questa tecnica sono che:

- il modello geometrico della realtà basato su 2 camere è molto approssimativo, in quanto servirebbero molte camere per ridurre l'approssimazione aggiungendo molta complessità al processo
- la grafica ha il problema opposto e cioè è abbastanza gestibile ma non rende fotorealistica la scena.

La soluzione è fondere le immagini acquisite con parti di modelli per generare immagini di realtà aumentata.

## Free viewpoint video

---

Di applicazioni così ce ne sono a bizzeffe, soprattutto per cinematografia: l'utente reale si muove, ripreso da  $n$  camere che lo trackkano in modo sufficiente da poter ricostruire il suo movimento in ambiente grafico, cosa che permette di modificare l'utente "grafico" a piacere.

Per far ciò cerchiamo di passare dal modello standard di scena filmata da un singolo punto di vista con coordinate  $[x, y, t]$  (cioè le coordinate sul piano immagine e il tempo) ad un free viewpoint video con  $n$  camere allineate orizzontalmente, o disposte in altro modo, che acquisiscono contemporaneamente filmati da posizioni diverse.

Per velocizzare l'acquisizione possiamo usare meno camere alleggerendo il carico del software.

Le posizioni che non sono acquisite direttamente da una camera sono generate tramite interpolazione...

Come? Partiamo da una configurazione iniziale in cui supponiamo di avere due sole camere che osservano una scena:

- 1) dobbiamo sapere la corrispondenza dei punti sui 2 piani immagine (prendendo punti in zone altamente informative)
- 2) se conosciamo  $x_0$  e  $x_1$  trovate da 1) possiamo conoscere  $p$  (coordinata del punto nella scena) e...
- 3) conoscendo  $p$  da 2) possiamo generare la sua proiezione su qualsiasi altro piano immagine virtuale (fittizio) conoscendo la posizione del piano immagine.

### Generazione tramite mappa di profondità ottenuta da sensore

Gli algoritmi che eseguono questi passi seguono le regole geometriche/matematiche viste nelle sezioni precedenti: supponiamo che le due camere siano a distanza  $b$  (baseline) ma non siano convergenti e acquisiamo 2 scene con 2 sensori di profondità identici<sup>16</sup>.

A questo punto conoscendo anche la distanza dell'oggetto proiettato possiamo costruire una mappa delle distanze, vista come un'immagine a profondità 8 bit: a colori bianchi sono associati oggetti a distanza minima, a colori scuri sono associati oggetti lontani.

Possiamo ricavare il colore in funzione della distanza, associando al colore minimo e massimo  $z_{near}$  e  $z_{far}$ , cioè la distanza minima e massima acquisita dal sensore e le altre di conseguenza.

Ricavo perciò  $1/z$  come  $d(isparità)/255$ .

Ora che abbiamo a disposizione un'immagine della scena, associata alla sua mappa di profondità, possiamo cercare di generare una seconda immagine.

I punti dell'immagine originale sappiamo dove rimapparli in base alla traslazione orizzontale calcolata sulla base della distanza.

Ci sono però altri problemi da affrontare:

- Le aree occluse nell'immagine originale (a differenza di ciò che accade nel modello grafico) non le conosciamo e quindi quando simuliamo lo spostamento della camera non sappiamo come completare la parte di scena che ora vediamo (quella che vogliamo generare) e prima non vedevamo.  
Una delle tecniche principali per risolvere questo problema è fare interpolazione, cosa che molte volte genera pixel poco sensati.
- Gli shift dei punti conosciuti possono non essere valori interi di pixel e quindi è necessario capire in quale pixel riposizionare il vecchio punto immagine.

---

<sup>16</sup> È un sogno bellissimo quello di avere due sensori identici, visto che ogni sensore è ovviamente unico per l'elettronica interna di cui è fatto.

Per questo eseguiamo una fase di pre-processing per correggere i difetti conosciuti del sensore e del white-balance (bilanciamento del bianco), e infine per riallineare le immagini in caso di disallineamento.

Per fare ciò si usano tecniche di interpolazione, oppure si acquisisce un'immagine a doppia risoluzione (nello spazio di un pixel ce ne sono 4), cosa che ci permette più libertà di movimento.

- Gli shift dei punti conosciuti possono rimappare nella stessa posizione più pixel originali: è necessario usare la mappa di profondità per capire quale pixel mostrare (cioè come colorare il nuovo pixel).

### GENERAZIONE TRAMITE MAPPA DI PROFONDITÀ OTTENUTA MEDIANTE DEPTH ESTIMATION

Al posto di usare un sensori di profondità è possibile generare la mappa di profondità partendo da due immagini scostate fra loro, conoscendo la corrispondenza fra i pixel.

Per ogni pixel nella prima immagine (definito da  $x$ ) definisco le linee epipolari e cerco sulle linee il punto corrispondente al pixel iniziale (definito  $x'$ ), cercando di fare un match nell'intorno del primo pixel a meno di variazioni di illuminazione o difetti delle camere.

Conoscendo  $f$ ,  $b$  e  $d$  (fuoco, baseline e disparità) posso ricavare  $z$  come proporzionale a  $\frac{1}{x-x'}$  (a meno delle costanti  $f$  e  $b$ ).

La corrispondenza fra i pixel dei due piani immagini la trovo muovendomi sulle linee epipolari, confrontando un intorno del pixel originale con degli intorni centrati su tutti i pixel sulla linea epipolare. Un meccanismo di correlazione prevede che la differenza fra i due blocchetti (intorno del pixel originale e di quello considerato) è tanto più piccola quanto più i blocchi sono simili.

Quando trovo la differenza minima posso supporre di aver trovato la corrispondenza.

A questo punto è possibile calcolare la disparità come differenza fra le nuove coordinate trovate e quelle vecchie.

La differenza di dimensione dell'intorno del pixel incide sui risultati della correlazione: le finestre piccole prendono più cantonate in presenza di tante aree simili anche non corrispondenti, ma hanno il vantaggio di riconoscere più dettagli a rischio di farsi fregare in caso di immagini con molto rumore; le finestre più grandi diminuiscono sia i dettagli che gli errori, a scapito della risoluzione che è poverissima.

Il metodo della correlazione fallisce in aree di background, con pattern ripetitivi o texture frequenti.

Per migliorare le prestazioni possiamo usare alcune informazioni conosciute a priori:

1. per ciascun punto deve esserci una sola corrispondenza
2. i punti appartenenti ad uno stesso oggetto seguono un ordine che deve essere mantenuto nelle due immagini
3. imponiamo che le misure di disparità non cambino drasticamente da punto a punto (la mappa di profondità deve esser smoothness), evitando correlazioni fra punti molto lontani fra loro.

Per mettere insieme questi dati si usa un “funzionale di energia”:

$$F = a \cdot e_{data} + b \cdot e_{smooth}$$

che pesa l'informazione derivante dal SSD con lo a smoothness della mappa di profondità tramite i parametri  $a$  e  $b$ .

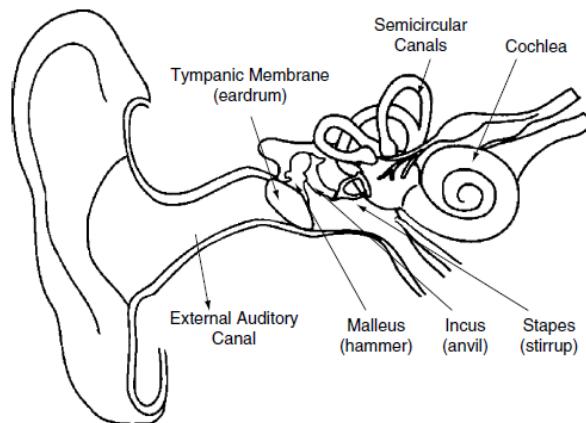
La corrispondenza è trovata cercando il minimo del funzionale di energia, garantendoci risultati molto vicini all'ottimo.

# 10. Audio 3D

## Introduzione

All'interno della costruzione di un sistema di realtà virtuale gioca un ruolo significativo la presentazione acustica: una distribuzione accurata del suono permette infatti agli utenti una migliore immersione nel mondo, poiché l'utente è abituato a percepire continuamente suoni nella propria realtà.

### Sistema uditivo umano



La figura mostra il sistema uditivo umano, nel quale il suono segue il seguente percorso:

1. le onde sonore mettono in vibrazione la membrana del **timpano**
2. i tre ossicini successivi (**staffa**, **incudine** e **martello**) passano la vibrazione alla **coclea**, una struttura a forma di chiocciola, contenente un fluido in cui si diffondono le vibrazioni
3. le vibrazioni provocano l'eccitazione di **cellule ciliate** poste sulla membrana della coclea, le quali generano un impulso nervoso sul **nervo uditivo**
4. l'impulso nervoso è inviato al cervello.

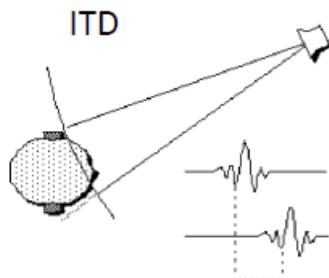
### Percezione del suono e della sua direzione

Una volta compresa la struttura del nostro sistema uditivo, la domanda successiva che ci poniamo è: come percepiamo la direzione del suono?

Ciò ci permette di capire come può essere percepito un suono 3D.

Le caratteristiche che ci fanno distinguere due direzioni diverse di provenienza del suono sono:

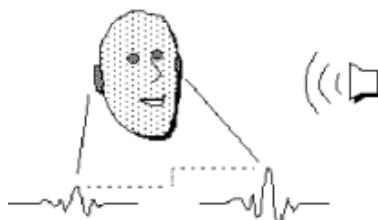
- **volume**: la differenza del volume del suono (cioè dell'ampiezza/energia dell'onda sonora) che percepiamo fra un orecchio e l'altro ci permette di cogliere la differente direzione.  
Inoltre, dato che l'intensità diminuisce con il quadrato della distanza, le differenze di intensità ci permettono di distinguere sorgenti lontane da quelle vicine.
- **ritardo**: l'onda sonora viaggia nello spazio, pertanto arriva in tempi diversi nelle due orecchie. Tale differenza di tempo viene chiamato **ritardo**, **differenza di fase** o **Interaural Time Delay (ITD)**.  
Ampi ritardi provocano l'**effetto eco**.



Visualizzazione in frequenza dell'Interaural Time Delay

- **spettro di frequenza:** le onde sonore che arrivano da un lato della testa incontrano l'orecchio più vicino, attraversano la scatola cranica e infine raggiungono l'orecchio più lontano. Il passaggio delle onde attraverso la scatola cranica provoca una variazione dello spettro in frequenza dell'onda sonora, in quanto cambia il materiale attraversato. Questo fenomeno prende il nome di **Interaural Intensity Difference (IID)**.

IID



Visualizzazione in frequenza dell'Interaural Intensity Difference

- **altre riflessioni o assorbimenti:** se la stanza in cui viene emesso il suono non è isolata, le onde vengono riflesse o assorbite (in base al materiale) sulle pareti della stanza, aggiungendo informazioni. Ciò può essere simulato producendo suoni ravvicinati nel tempo, emulando così il rimbalzo sul sistema esterno delle onde sonore.

## Metodi per la creazione dell'audio immersivo

Date le informazioni che abbiamo raccolto finora possiamo chiederci quali metodi possiamo usare per creare un **audio immersivo**.

L'obiettivo che ci poniamo per la costruzione di un audio immersivo è di effettuare un tracking completo dell'onda nella posizione dell'utente: se siamo in condizioni di controllare completamente l'onda sonora emessa (effetti di fase, ampiezza e gamma di frequenza), nella punto e nel momento in cui questa raggiunge l'utente, possiamo ottenere un buon risultato.

Per fare ciò le possibilità sono:

- **audio soundfield:** la soluzione più complessa prevede di emulare l'intero sistema audio nella stanza.  
In questo metodo è difficile effettuare il tracking del singolo utente perché sarebbe necessario mutare il ruolo degli speaker in base alla posizione dell'utente stesso (pensiamo ad un utente che gira su se stesso, lo speaker destro deve diventare progressivamente quello sinistro e viceversa).  
Può essere usato in modo abbastanza efficiente se si costruisce un suono nel quale la posizione dell'utente è una posizione mediata fra quelle possibili, rispetto alla posizione degli speaker.

- **audio binaurale**: è la soluzione più facile perché lo strumento fisico usato come speaker sono delle **cuffie**: ciascun utente è dotato di cuffie e qualsiasi suo movimento è seguito dagli speaker in modo fisico.

Ciò ci consente di controllare lo stimolo sonoro direttamente sulle due orecchie usando solo la **stereofonia** (due canali di emissione del suono), costruendo un tracking personalizzato sul singolo utente e privo di interferenze esterne.

Evita il fenomeno del **cross-talk**.

L'unico svantaggio è che ogni utente deve essere provvisto di cuffie.

La stereofonia può essere simulata con buoni risultati gestendo il volume sull'orecchio destro e sinistro in base all'angolo compreso fra la direzione del suono e la normale alla fronte dell'utente, come vedremo in dettaglio più avanti.

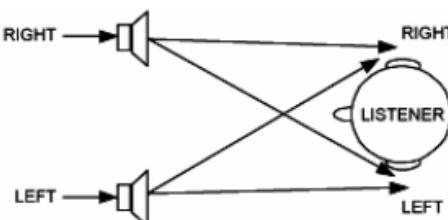
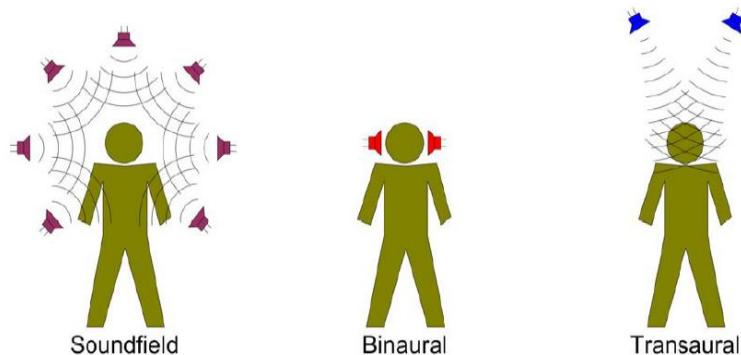
- **audio transaurale**: impiega un sistema di altoparlanti a due canali.

Ha il vantaggio di non dover indossare delle cuffie e quindi offre una sensazione di naturalezza e apertura offerta da un ascolto a campo diffuso.

La libertà di movimento associata all'ascolto *transaurale* può però compromettere la resa del sistema audio 3D: se si considera l'ITD, risulta evidente come un movimento dell'ascoltatore verso un altoparlante costituisca una distorsione non prevedibile dal progettista e che quindi falserà la corretta collocazione della sorgente. Viene così a individuarsi una regione spaziale limitata, detta **sweet spot**, all'interno della quale le informazioni spaziali riprodotte saranno correttamente interpretate dal soggetto.

Al di fuori di essa, invece, la resa del sistema non è prevedibile ma in generale non porterà alla corretta spazializzazione.

È soggetto al fenomeno del cross-talk.



Fenomeno di cross-talk nell'ascolto transaurale.

Alla fine degli anni 90 la costruzione di audio 3D come quello illustrato necessita di hardware complesso, affiancato ad un software semplice.

Negli anni successivi si cerca di spostare la complessità dall'hardware al software, e si mira in particolare all'isolamento dell'utente dal mondo esterno, in modo da sintetizzargli più facilmente il suono desiderato nelle cuffie.

Spostare il focus sul software porta alla nascita degli studi sul Discrete Signal Processing.

---

## Trasformata di Fourier

---

Chi è Jean Baptiste Joseph Fourier?

Nato in Francia nel 1768 ha contribuito allo studio del segnale periodico nei primi anni del 1800.

### Numeri complessi

Un **numero complesso**  $C$  è definito come:

$$C = R + jI \quad R, I \in \mathbb{R}, j \in \mathbb{I}, \quad j = \sqrt{-1}, j^2 = 1$$

$R$  è la parte reale del numero complesso mentre  $I$  è la parte immaginaria.

I numeri reali sono un sottoinsieme dei numeri complessi in cui  $I = 0$ .

Il **coniugato** di un numero complesso  $C$  è:

$$C^* = R - jI$$

I numeri complessi possono essere visti geometricamente come punti sul piano (piano complesso) la cui ascissa è  $R$  mentre l'ordinata è  $I$ .

Un numero complesso può essere espresso in coordinate polari:

$$C = |C|(\cos \theta + j \sin \theta) \quad |C| = \sqrt{R^2 + I^2}$$

dove  $|C|$  rappresenta la lunghezza del vettore che congiunge l'origine degli assi con il numero complesso  $C$  di coordinate  $(R, I)$  e  $\theta$  è l'angolo tra il vettore e l'asse  $x$ .

Dal grafico si nota che  $\tan \theta = \left(\frac{I}{R}\right)$  e  $\theta = \arctg \left(\frac{I}{R}\right)$ .

La funzione  $\arctg$  ritorna un angolo compreso in  $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ .

Ma siccome  $R$  ed  $I$  possono essere positive e negative indipendentemente, dobbiamo ottenere angoli nel range  $[-\pi, \pi]$ , quindi si utilizza la funzione  $\text{atan2}(\text{Imag}, \text{Real})$ .

Usando la **formula di Eulero**:

$$e^{j\theta} = \cos \theta + j \sin \theta \quad e = 2.71828 \dots$$

abbiamo una rappresentazione dei numeri reali in coordinate polari:

$$C = |C|e^{j\theta}$$

dove  $|C|$  e  $\theta$  sono rappresentati come prima.

### Serie di Fourier

La **serie di Fourier** è una somma di  $\sin$  e  $\cos$  moltiplicati per coefficienti appropriati, che esprime una funzione  $f(t)$  di una variabile periodica  $t$  con periodo  $T$ .

Ogni funzione periodica può essere scritta come somma di  $\sin$  e  $\cos$  di differenti ampiezze e frequenze.

Se c'è un solo tipo ( $\sin$  o  $\cos$ ) allora è coinvolta anche la **fase**.

La serie di Fourier ha la seguente forma:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{T}t} = \sum_{n=-\infty}^{\infty} c_n e^{j\omega nt} \quad n \in \mathbb{Z}, \omega = \frac{2\pi}{T}$$

dove i coefficienti sono:

$$c_n = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{-j\frac{2\pi n}{T}t} dt = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{-j\omega nt} dt \quad n = 0, \pm 1, \pm 2, \dots$$

Il fatto che  $f(t)$  sia composta da  $\sin$  e  $\cos$  deriva dalla **formula di Eulero**:

$$e^{j\omega} = \cos \omega + j \sin \omega$$

## Impulsi e proprietà di sifting

Un concetto fondamentale collegato alla trasformata di Fourier è quello dell'impulso e delle sue proprietà di sifting.

Un **impulso unitario (delta di Dirac)** di una variabile continua  $f$ , posizionato a  $t = 0$  e denotato da  $\delta(t)$  è definito come:

$$\delta(t) = \begin{cases} \infty & t = 0 \\ 0 & t \neq 0 \end{cases}$$

e deve soddisfare l'identità:

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

Quindi un impulso può essere visto come un picco di ampiezza infinita e durata nulla, avente una superficie unitaria.

La **proprietà di sifting** fornisce il valore della funzione  $f(t)$  nella posizione dell'impulso  $t = t_0$  quindi:

$$\text{per } t = 0: \int_{-\infty}^{\infty} f(t)\delta(t) dt = f(0)$$

$$\text{per } t = t_0: \int_{-\infty}^{\infty} f(t)\delta(t - t_0) dt = f(t_0)$$

Le proprietà viste fin ora si riferiscono al continuo ma noi, trattando immagini digitali, abbiamo a che fare con quantità discrete quindi le formule viste prima per poter descrivere un singolo **impulso discreto**  $\delta(t)$  diventano:

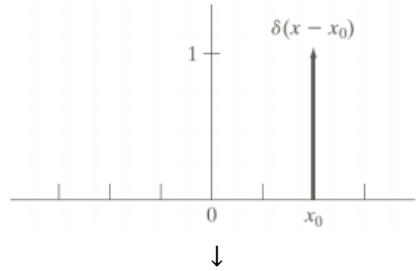
$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases}$$

$$\sum_{x=-\infty}^{\infty} \delta(x) = 1$$

La **proprietà di sifting** per variabili discrete diventa:

$$\text{per } x = 0: \sum_{x=-\infty}^{\infty} f(x)\delta(x) = f(0)$$

$$\text{per } x = x_0: \sum_{x=-\infty}^{\infty} f(x)\delta(x - x_0) = f(x_0) \quad (\text{generica})$$



Un singolo impulso discreto posizionato in  $x = x_0$ .

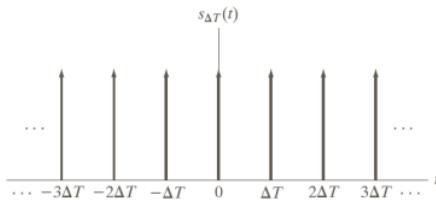
La variabile  $x$  è discreta e  $\delta$  è 0 in qualsiasi posizione, tranne che in  $x = x_0$ .

spike centrato dove l'argomento di  $\delta$  si annulla. In questo caso:

$$x - x_0 = 0 \quad \text{per } x = x_0$$

Un **treno di impulsi** è la somma di infiniti impulsi periodici  $\Delta T$  unitari e separati:

$$S_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$



## Trasformata di Fourier di una variabile continua

La **trasformata di Fourier** di una funzione continua  $f(t)$  di una variabile continua  $t$  è definita dalla seguente equazione:

$$\mathcal{F}\{f(t)\} = F(\mu) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t} dt$$

La trasformata può essere indicata con  $F(\mu)$  perché è una funzione della sola variabile  $\mu$ , anch'essa continua, infatti  $t$  è la variabile di integrazione.

La condizione sufficiente per l'esistenza è:

$$\int_{-\infty}^{\infty} [f(t)]^2 dt < 0$$

L'**antitrasformata di Fourier** (o **trasformata inversa**) permette di risalire a  $f(t)$  data  $F(\mu)$  in questo modo:

$$f(t) = \int_{-\infty}^{\infty} F(\mu)e^{j2\pi\mu t} d\mu$$

Le due equazioni che abbiamo appena visto individuano la **coppia di Fourier**.

Se  $F(\mu)$  è la trasformata di Fourier di  $f(t)$ , allora  $f(-\mu)$  è la trasformata di Fourier di  $F(t)$ .

$$\int_{-\infty}^{\infty} F(t)e^{-j2\pi\mu t} dt = \int_{-\infty}^{\infty} F(t)e^{j2\pi(-\mu)t} dt = f(-\mu)$$

L'equazione della trasformata di Fourier, grazie alla formula di Eulero, può essere riscritta come:

$$F(\mu) = \int_{-\infty}^{\infty} f(t)[\cos(2\pi\mu t) - j \sin(2\pi\mu t)] dt$$

Dopo l'integrazione, l'unica variabile rimasta è la frequenza, per questo motivo il dominio della trasformata di Fourier viene chiamato **dominio della frequenza**.

## Trasformate di Fourier utili

### ESEMPIO 1: OTTENERE LA TRASFORMATA DI FOURIER DI UNA SEMPLICE FUNZIONE

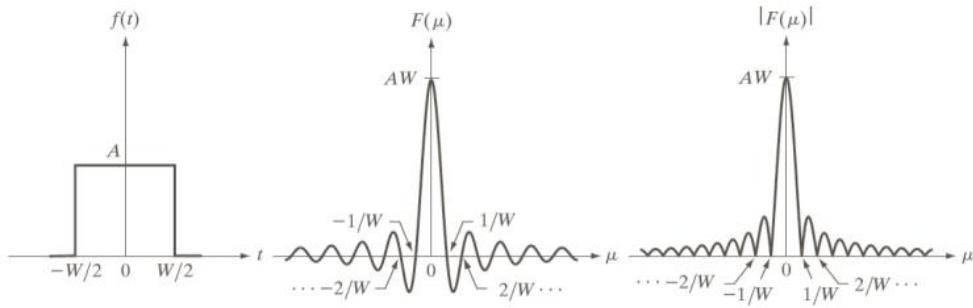
Data la funzione  $f(t)$  che è rappresentata in figura (a), la sua trasformata è:

$$f(t) = \begin{cases} A & -\frac{W}{2} \leq t \leq \frac{W}{2} \\ 0 & \text{altrimenti} \end{cases} \quad F(\mu) = AW \frac{\sin(\pi\mu W)}{\pi\mu W}$$

Una funzione

La sua trasformata di Fourier

Lo spettro



Il problema della trasformata di Fourier è che solitamente contiene termini complessi e, per agevolare la visualizzazione, si lavora con la magnitudo della trasformata (cioè una quantità reale) chiamata **spettro di Fourier** (o **spettro della frequenza**).

$$|F(\mu)| = AT \frac{\sin(\pi\mu W)}{\pi\mu W}$$

Notiamo che la funzione tende a  $\infty$  sia per valori positivi che negativi di  $\mu$ .

Le posizioni degli zeri di entrambe le funzioni (trasformata e spettro) sono inversamente proporzionali alla larghezza  $W$  della funzione box, la cui altezza dei lobi decresce in funzione della distanza dall'origine.

### ESEMPIO 2: TRASFORMATA DI FOURIER DI UN IMPULSO

- La trasformata di Fourier di un *impulso posizionato nell'origine* ( $t = 0$ ) del dominio spaziale diventa una costante nel dominio della frequenza:

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t) e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{\infty} \underbrace{e^{-j2\pi\mu t}}_{f(t)} \delta(t) dt \quad \text{dalla sifting property } \int_{-\infty}^{\infty} f(t)\delta(t)dt = f(0) \\ &= e^{-j2\pi\mu 0} \\ &= e^0 = 1 \end{aligned}$$

- Se invece l'*impulso è posizionato in  $t = t_0$*  allora diventa:

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t - t_0) e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{\infty} \underbrace{e^{-j2\pi\mu t}}_{f(t)} \delta(t - t_0) dt \quad \text{dalla sifting property } \int_{-\infty}^{\infty} f(t)\delta(t - t_0)dt = f(t_0) \\ &= e^{-j2\pi\mu t_0} \\ &= e^{j(-2\pi\mu t_0)} \\ &= \cos(-2\pi\mu t_0) + j \sin(-2\pi\mu t_0) \quad \begin{matrix} \cos \text{ è pari } \cos(-x) = \cos(x) \\ \sin \text{ è dispari } \sin(-x) = -\sin(x) \end{matrix} \\ &= \cos(2\pi\mu t_0) - j \sin(2\pi\mu t_0) \end{aligned}$$

### ESEMPIO 3: TRASFORMATA DI FOURIER DI UN TRENO DI IMPULSI

Quando parleremo di campionamento, faremo uso della trasformata di Fourier di un treno di impulsi periodico.

Ricavare tale trasformata non è così immediato come nel caso dei singoli impulsi ed è utile approfondirne nel dettaglio la derivazione.

Cominciamo col notare che l'unica differenza tra la formula della trasformata di Fourier e quella dell'antitrasformata è il segno dell'esponente, infatti:

$$\begin{array}{ll} \text{Trasformata:} & F(\mu) = \int_{-\infty}^{\infty} f(t) e^{-j2\pi\mu t} dt \\ & \\ \text{Antitrasformata:} & f(t) = \int_{-\infty}^{\infty} F(\mu) e^{j2\pi\mu t} dt \end{array}$$

In questo modo, se una funzione  $f(t)$  ha la trasformata di Fourier  $F(\mu)$  successivamente  $F(t)$  ha trasformata  $f(-\mu)$ .

Utilizzando tale simmetria e dato che, come mostrato sopra, la trasformata di Fourier di un impulso  $\delta(t - t_0)$  è  $e^{-j2\pi\mu t_0}$ , ne segue che la funzione  $e^{-j2\pi\mu t_0 t}$  ha come trasformata  $\delta(-\mu - t_0)$ .

Supponiamo che  $a = -t_0$ , ne segue che la trasformata di  $e^{j2\pi\mu a t}$  è  $\delta(-\mu - a) = \delta(\mu - a)$  dove il secondo termine è vero perché  $\delta$  non è uguale a zero solo quando  $\mu = a$ , che è lo stesso risultato sia per  $\delta(-\mu + a)$  che per  $\delta(\mu - a)$ , le due forme sono quindi equivalenti.

Il treno di impulsi  $S_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$  è periodico con periodo  $\Delta T$  e può essere espresso come una serie di Fourier:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{\Delta T} t} \quad c_n = \frac{1}{\Delta T} \int_{-\frac{\Delta T}{2}}^{\frac{\Delta T}{2}} s_{\Delta T}(t) e^{j\frac{2\pi n}{\Delta T} t} dt$$

Facendo riferimento alla figura del treno di impulsi, si nota che l'integrale nell'intervallo  $\left[-\frac{\Delta T}{2}, \frac{\Delta T}{2}\right]$  comprende solo l'impulso di  $s_{\Delta T}(t)$ , collocato nell'origine.

Di conseguenza, la precedente equazione  $c_n$  diventa:

$$c_n = \frac{1}{\Delta T} \int_{-\frac{\Delta T}{2}}^{\frac{\Delta T}{2}} \delta(t) e^{j\frac{2\pi n}{\Delta T} t} dt = \frac{1}{\Delta T} e^0 = \frac{1}{\Delta T}$$

L'espansione in serie di Fourier diventa:

$$s_{\Delta T}(t) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T} t}$$

Il nostro obiettivo è quello di ottenere la trasformata di Fourier di questa espressione.

Visto che la somma è un processo lineare, ottenere la F.t. di una somma è la stessa cosa che ottenere la somma delle F.t. delle singole componenti.

Queste componenti sono degli esponenziali che, come visto in questo esempio, sono:

$$\mathcal{F}\left\{e^{j2\pi t \frac{n}{\Delta T}}\right\} = \delta\left(\mu - \frac{n}{\Delta T}\right)$$

Così  $S(\mu)$ , la F.t. di un treno di impulsi periodico  $s_{\Delta T}(t)$  è:

$$\begin{aligned} S(\mu) &= \mathcal{F}\{s_{\Delta T}(t)\} = \mathcal{F}\left\{\frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{-j\frac{2\pi n}{\Delta T} t}\right\} = \frac{1}{\Delta T} \mathcal{F}\left\{\sum_{n=-\infty}^{\infty} e^{-j\frac{2\pi n}{\Delta T} t}\right\} \\ &= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right) \end{aligned}$$

Questo risultato fondamentale ci dice che la trasformata di Fourier di un treno di impulsi con periodo  $\Delta T$  è anch'essa un treno di impulsi, il cui periodo è  $\frac{1}{\Delta T}$ .

Questa proporzionalità inversa tra i periodi di  $s_{\Delta T}(t)$  e  $S(\mu)$  è analoga a quanto visto nella figura di Esempio 1, nel rapporto tra una funzione box e la sua trasformata.

## Convoluzione

---

La **convoluzione** di due funzioni prevede il flipping (rotazione di 180°) di una funzione rispetto all'origine e il suo scorrimento sull'altra, durante il quale si effettua un calcolo.

La convoluzione di due funzioni continue  $f(t)$  e  $h(t)$ , di una variabile continua  $t$ , è:

$$c(t) = f(t) * h(t) = \int_{-\infty}^{\infty} f(\tau)h(t - \tau) d\tau$$

dove:

- il segno meno serve per il flipping
- $t$  è lo spostamento necessario per lo scorrimento di una funzione sull'altra
- $\tau$  è la variabile di integrazione.

La trasformata di Fourier della convoluzione risulta essere:

$$\mathcal{F}\{f(t) * h(t)\} = H(\mu)F(\mu)$$

dove  $f(t)$  e  $h(t)$  possono essere invertite, quindi la convoluzione gode della **proprietà commutativa**.

In dettaglio, se  $t$  indica il dominio spaziale e  $\mu$  il dominio della frequenza, la trasformata di Fourier della convoluzione di due funzioni nel dominio spaziale è uguale al prodotto delle due funzioni nel dominio della frequenza della trasformata di Fourier delle due funzioni.

Viceversa, dal prodotto di due trasformate otteniamo la convoluzione nel dominio spaziale calcolando l'antitrasformata di Fourier.

Ora abbiamo tutte le basi per esporre le due parti del teorema della convoluzione che è la base del filtraggio nel dominio della frequenza.

### 1° parte

$$f(t) * h(t) \Leftrightarrow H(\mu)F(\mu)$$

Indica che l'espressione di destra si ottiene attraverso la trasformata di Fourier dell'espressione di sinistra, mentre l'espressione di sinistra si ottiene attraverso l'inverso della trasformata di Fourier dell'espressione di destra.

### 2° parte

$$f(t)h(t) \Leftrightarrow H(\mu) * F(\mu)$$

Indica che la convoluzione nel dominio della frequenza è analoga alla moltiplicazione nel dominio spaziale e che sono relazionate, rispettivamente, dalla trasformata di Fourier e dalla sua inversa.

La convoluzione gode delle seguenti proprietà:

- commutatività
- associatività
- distributività.

## Campionamento

Con le nozioni apprese finora possiamo affrontare il problema del campionamento.

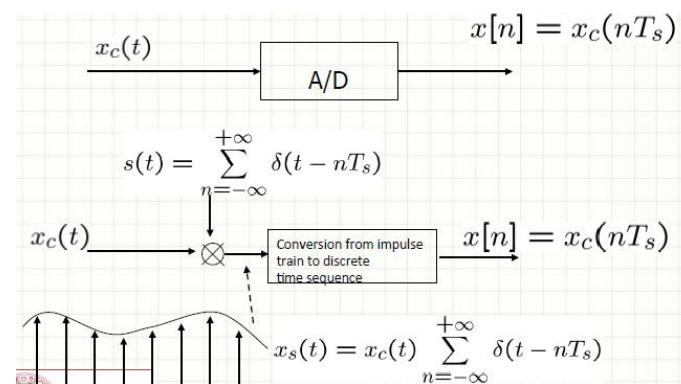
Nel caso in cui si desideri memorizzare un segnale la prima idea è quella di memorizzarne la forma analogica: tale metodo però richiede l'accumulo di un gran numero di informazioni, con l'effetto di occupare uno spazio eccessivo.

Per ovviare a tale problema è possibile effettuare il campionamento, con l'idea di memorizzare solo alcuni tratti del segnale, in modo tale da poter ricostruire il segnale (o una sua forma molto simile) nel momento del bisogno.

Per evitare perdita di informazioni attraverso il campionamento è necessario campionare ad una frequenza doppia rispetto alla frequenza massima del segnale (**teorema di Nyquist**).

Dal segnale  $X$  originale in ingresso passiamo ad un segnale campionato  $X_c$  tramite il convertitore analogico/digitale, che campiona con periodo  $T_s$  e frequenza  $\frac{1}{T_s}$ .

Per effettuare il campionamento usiamo l'astrazione della delta di Dirac: moltiplichiamo il segnale per il treno di impulsi (con periodo infinitesimo). I segnali campionati vengono immagazzinati nel vettore  $x[n]$ .



Se proviamo a vedere in frequenza cosa avviene determiniamo le trasformate di Fourier di  $X_c$  (segnale originale),  $S$  (treno di impulsi) e  $X_s$  (segnale campionato): il prodotto dei segnali in un dominio è la convoluzione dei segnali nell'altro dominio.

La trasformata del segnale campionato è una somma infinita del segnale originale traslato di  $1/T_s$  (per le proprietà di convoluzione fra un treno di impulsi e un segnale).

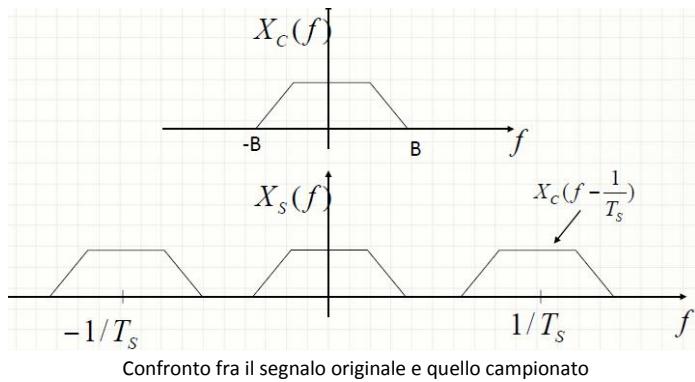
Supponiamo che oltre ad un certo limite di frequenze  $B$  il segnale in frequenza sia nullo.

In questo caso dopo aver fatto il campionamento abbiamo una somma dello stesso segnale originale traslato di  $\frac{1}{T_s}$ .

Se le repliche sono sufficientemente distanti, cioè  $T_s$  molto piccolo, i segnali non si sovrappongono ed è possibile estrarre il segnale originale, prelevando solo le frequenze fra  $-B$  e  $B$ .

Le trasformate di Fourier di un segnale originale, un treno di impulsi e del campionamento sono:

$$\begin{aligned}\mathcal{F}\{x_c(t)\} &= X_c(f) = \int_{-\infty}^{\infty} x_c(t) e^{-j2\pi ft} \\ \mathcal{F}\{s(t)\} &= S(f) = \frac{1}{T_s} \sum_{-\infty}^{\infty} \delta\left(f - \frac{n}{T_s}\right) \\ \mathcal{F}\{x_s(t)\} &= X_s(f) = \frac{1}{T_s} \sum_{-\infty}^{\infty} X_c\left(f - \frac{n}{T_s}\right)\end{aligned}$$



Per estrarre il segnale originale effettuo un filtraggio nelle frequenze, tagliando alla metà della frequenza usata nel campionamento.

Per farlo si fa il prodotto della trasformata del segnale campionato con un filtro  $H$ , o altresì effettuiamo una convoluzione nel dominio del tempo.

Il filtro  $H$  viene chiamato funzione di trasferimento, in quanto mantiene alcune frequenze ma ne elimina altre.

È importante però campionare ad una frequenza tale da evitare di sovrapporre i segnali originali nel segnale campionato.

Infatti, se si campiona in modo non adeguato al limite fra due sequenze translate c'è l'**aliasing**: vengono sommate frequenze diverse fra loro che deformano lo spettro e quindi rovinano il segnale che viene estratto.

Per evitare l'effetto aliasing è necessario far sì che la banda occupata dal segnale sia di dimensione

$$B < \frac{1}{2T_s}.$$

Il problema nell'aliasing è che dal punto di vista teorico nessun segnale è compreso in una banda finita di frequenze: quando viene acquisito un suono, nelle fasi iniziali e finali dell'acquisizione c'è una transizione rapida del suono, che crea una frequenza altissima, con un relativo spettro che ondeggi all'infinito.

Da una certa frequenza in poi però tali sinusoidi hanno ampiezza quasi nulla e perciò possono essere ignorate.

In più, il segnale che noi consideriamo come originale (quello che campioniamo) può essere precedentemente filtrato tramite un filtraggio analogico (l'**anti-aliasing filter**) del segnale che pulisce le frequenze spurie: in pratica inseriamo una porta sufficientemente stretta, che limita l'ingresso delle frequenze ad una soglia della metà della frequenza di campionamento.

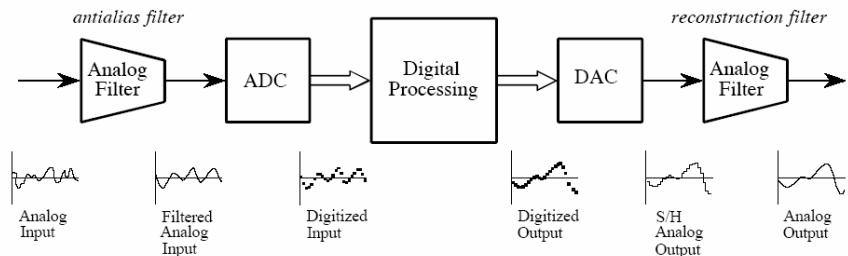
All'uscita invece considero tutti i campioni che ho memorizzato e interpolo ogni campione singolarmente.

L'interpolante apporta informazioni nell'intorno del punto campionato, ma non lontano.

Sommendo tutte le interpolanti riottengo il segnale originale (o un segnale molto simile, non distinguibile dall'orecchio umano).

Le differenze dal segnale originale sono provocate dalla poca informazione che è memorizzata con il campionamento e dalla precisione finita della macchina.

Come ultima precisazione possiamo notare che il campionamento nella realtà non viene effettuato con delle delta di Dirac ideali, ma con dei campionamenti di lunghezza minima possibile nel tempo: tale lunghezza deve essere abbastanza piccola per avere un segnale puro, ma non troppo perché campionamenti infinitesimi portano un contributo di rumore eccessivo.



## Discrete Fourier transform

L'obiettivo della **trasformata di Fourier discreta** è quella di determinare una approssimazione numerica della trasformata di Fourier.

Nella DFT l'integrale di Fourier è sostituito dalla sommatoria sui campioni effettuati.

I segnali sono acquisiti con passo  $T_0$ ; le sinusoidi sono discrete, campionate con  $m$  che va da 0 a  $N - 1$ .  $n$  permette di discretizzare la frequenza, in quanto l'insieme di frequenze utilizzate è generato come  $nf_0$ .

$f_0$  e  $T_0$  rappresentano la frequenza minima di acquisizione e il periodo minimo di acquisizione e sono solitamente normalizzate a 1 per rapidità di calcolo.

$$\bar{x}(mT_0) = \frac{1}{T} \sum_{n=0}^{N-1} \bar{x}(nf_0) e^{jm\frac{2\pi}{N}n}$$

$$\bar{X}(nf_0) = \frac{T}{N} \sum_{m=0}^{N-1} \bar{x}(mT_0) e^{-jn\frac{2\pi}{N}m}$$

In realtà, degli  $n$  campioni in frequenza prelevati dalla DFT consideriamo solo la prima metà, perché la seconda comprende la prima replica della parte di spettro prima di 0.

Il dominio della DFT è quindi formato da  $n/2$  valori, tutti con parte reale e immaginaria.

---

## Sistema audio

---

$$y[n] = T\{x[n]\}$$



Dato un sistema audio, questo può godere di diverse caratteristiche:

- **lineare**: un sistema viene detto lineare se vale il principio di sovrapposizione degli effetti per qualsiasi costante  $a$  e  $b$ :

$$T(a x_1[n] + b x_2[n]) = aT(x_1[n]) + bT(x_2[n])$$

- **senza memoria**: un sistema viene detto senza memoria se l'output  $y[n]$  per ogni  $n$  dipende solamente dal suo input  $x[n]$  per qualsiasi  $n$

- **BIBO (Bounded IN Bonded OUT)**: sistema non distruttivo, se il volume in ingresso è limitato lo stesso deve succedere in uscita

$$|x[n]| \leq B_x, \quad \forall n \quad \leftrightarrow \quad |T\{x[n]\}| \leq B_y, \quad \forall n$$

- **tempo invariante:** un sistema è tempo invariante se calcolare la trasformazione per un segnale  $x(n)$  non dipende dal tempo, altresì possiamo dire che una traslazione in ingresso implica una traslazione in uscita.

Tale sistema viene detto anche **LTI (Linear Time Invariant)**:

$$y[n] = T\{x[n]\} \quad T\{x[n - n_0]\} = y[n - n_0]$$

I diversi tipi di LTI possono essere combinati fra loro per determinare sistemi più complessi: le combinazioni tradizionali sono in serie o in parallelo.

### Risposta all'impulso

Un sistema LTI è univocamente definito dalla sua risposta all'impulso  $h[n]$ .

$$\begin{aligned}
 y[n] &= T\{x[n]\} \\
 &= T\left\{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right\} \\
 &= \sum_{k=-\infty}^{\infty} x[k]T\{\delta[n-k]\} \\
 &= \boxed{\sum_{k=-\infty}^{\infty} x[k]h[n-k]}
 \end{aligned}$$

Time invariance

Discrete convolution

La figura mostra il calcolo relativo alla risposta all'impulso: l'output  $y[n]$  è definito come la trasformazione dell'input  $x[n]$ .

A sua volta questo può essere approssimato in modo discreto con la somma dei segnali moltiplicati per la delta di Dirac.

Visto che le  $x[k]$  sono costanti rispetto alla trasformazione le porto fuori (questo perché se il sistema è lineare la trasformazione della somma è la somma delle trasformazioni).

Determino così una convoluzione discreta fra  $x[k]$  e  $h[n - k]$  che definisce la risposta all'impulso del sistema.

Ciò che viene fatto nella convoluzione è una somma pesata dell' $x$  con il filtro  $h$ , traslando il filtro come finestra mobile rispetto agli  $x$  a disposizione (vedi esempio sulle slide).

La **convoluzione discreta** fra una sequenza di  $x[n]$  di lunghezza  $N_1$  e  $h[n]$  di lunghezza  $N_2$  è definita come con  $y[n]$  sequenza di lunghezza  $N_1 + N_2 - 1$ .

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$

### Funzione di trasferimento

In frequenza un sistema lineare tempo invariante prevede l'elaborazione dello spettro di frequenza in ingresso con  $H(n)$ , definita funzione di trasferimento (che altro non è che la trasformata di Fourier di  $h(n)$ ).

Tale funzione è un filtro, i sistemi lineari però sono invarianti rispetto alle sinusoidi, pertanto la frequenza non può essere modificata tramite il filtro ma può essere solo ritardata, anticipata, attenuata o amplificata.

All'ingresso del sistema non viene ricevuto un segnale con una sola frequenza ma un suono formato da un insieme di frequenze, perciò le funzioni di trasferimento possono modificare il suono lavorando in modo diverso sulle diverse frequenze.

---

## 3D audio signal processing

---

Come detto precedentemente, per effettuare un tracking corretto del suono sull'utente (in particolare sulle due orecchie), è necessario riuscire a controllare l'intensità e il ritardo del suono sui due padiglioni auricolari.

Ma ciò non è sufficiente: anche la lo spettro di frequenza del suono muta quando le onde sonore attraversano la testa.

Tale cambiamento è provocato dal cambio di materiale attraversato, visto che è la scatola cranica è composta di acqua, mentre il mondo esterno d'aria.

L'acqua infatti attutisce alcune frequenze, modificando lo spettro.

### HRTF

---

Dato ciò è stata seguita una nuova teoria audio 3D: l'**Head-Related Transfer Function (HRTF)**.

Tale teoria si concentra sul fatto che la percezione è basata sulla differenza di distribuzione di energia delle onde sonore nel dominio della frequenza.

Per gestire tale differenza, come prima approssimazione, possiamo costruire un sistema audio lineare: posizioniamo una sorgente sonora in un punto nello spazio e generiamo un segnale continuo  $x(t)$  con diverse ampiezze al variare del tempo.

Successivamente consideriamo la distanza che intercorre fra la sorgente ( $x$ ) e il microfono ( $y$ ) (analogo dell'orecchio) e valutiamo l'onda che riceviamo.

Il percorso che l'onda intercorre fra la sorgente e il microfono ci è conosciuto (il suono si sposta in modo rettilineo), ma non abbiamo conoscenza sui materiali attraversati: supponiamo però che ci siano solo deformazioni lineari e, di conseguenza, che il segnale che otteniamo dipenda solo dalla risposta all'impulso  $h(t)$  di un certo filtro lineare.

Nella configurazione del sistema di test proviamo poi a usare noi stessi come microfono, raccogliendo  $y$  destro e  $y$  sinistro nelle posizioni delle nostre orecchie (supponiamo di trovarci in un ambiente insonorizzato, con una sorgente sonora che arriva da destra).

La coppia di risposte all'impulso fra l'orecchio sinistro e quello destro lo chiamiamo **HRIR (Head-Related Impulse Response)** nel dominio del tempo e HRTF nel dominio della frequenza (determiniamo  $H(f)$  che è la trasformata di Fourier di  $h(t)$ ).

Le due componenti (destra e sinistra) non si differenziano solo per ritardo o attenuazione del segnale, ma anche per l'applicazione di un filtro provocato dal fatto che frequenze diverse vengono modificate in modo diverso dalla nostra testa e dal nostro orecchio.

Grazie all'HRTF possiamo simulare con un modello lineare ciò che accade quando un suono raggiunge le nostre orecchie, poiché cattura la distribuzione dell'energia nel dominio della frequenza di un suono che arriva da un punto particolare nello spazio (dipende dalla posizione della sorgente).

Inoltre l'HRTF ci da un'informazione sulle frequenze a cui siamo più sensibili.

L'HRTF ci può essere utile per riprodurre o sintetizzare un segnale binario a partire da un segnale mono, tenendo a mente che dobbiamo avere conoscenza sulla posizione della sorgente (visto che l'HRTF cambia in base alla posizione), che non tutti gli utenti percepiscono il suono allo stesso modo (ma possiamo fare una media delle possibili percezioni), e che simuliamo che il suono arrivi direttamente alle orecchio tramite delle cuffie (invece che usare un modello in libertà che aumenterebbe la complessità).

## Misurazioni sperimentali

Per determinare le diverse risposte all'impulso, in base alla variazione di posizione della sorgente, si è proceduto sperimentalmente, cioè provando in un laboratorio isolato a misurare l'HRTF.

Tali misurazioni si sono poste l'obiettivo di determinare la differenza fra il segnale emesso e quello sintetizzato quando la sorgente si sposta nello spazio in uno spazio limitato intorno all'utente.

Per effettuare tali misurazioni si sono usate sia persone vere (con due microfoni posti davanti ai timpani) che manichini (con i microfoni al posto dei timpani), e si è proceduto modificando azimut (destra-sinistra) e elevazione della sorgente per capire come cambia il suono registrato.

Nelle misurazioni si fa ruotare la sorgente attorno alla testa del soggetto a distanza fissa (ciò perché modificare la distanza implica solo attenuare il suono, cosa che può essere facilmente fatta matematicamente).

Tali misurazioni hanno rivelato che siamo più sensibili alle variazioni di azimut e meno a quelle di elevazione.

I test che sono stati effettuati nei laboratori del MIT nel 1995 e poi dall'UC Davis CIPIC Laboratory hanno permesso di costruire dei databases contenenti dati corretti, utilizzabili da chiunque in momenti successivi.

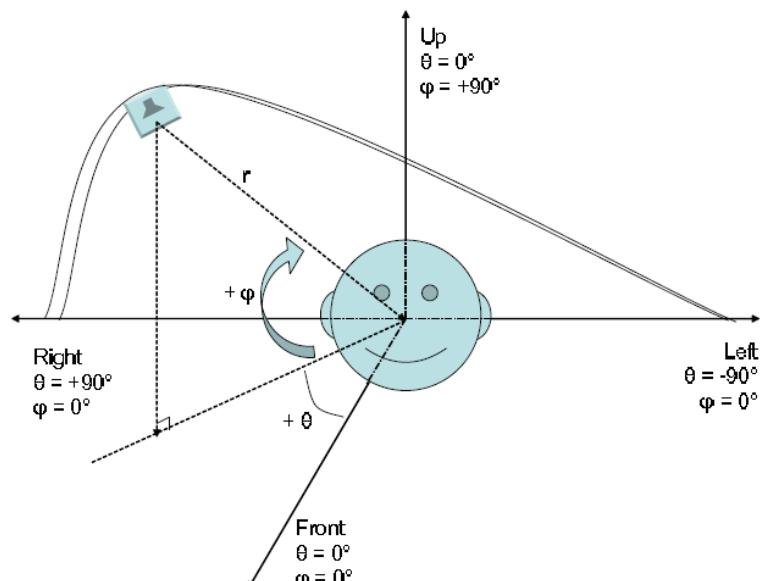
Tali misurazioni hanno usato come suono di base dei rumori bianchi, i quali contengono tutte le frequenze e permettono di cogliere immediatamente come vengono gestite le diverse frequenze e stimare quindi la relativa funzione di trasformazione.

Grazie alle misurazioni effettuate è possibile definire dei grafici che mostrano (al variare di elevazione o di azimuth) la variazione della frequenza, cioè la sua attenuazione o amplificazione.

Le misurazioni naturalmente non possono essere effettuate nel continuo, cioè per qualsiasi valore possibile di azimuth ed elevazione, ma solo in un insieme finito di punti (ad esempio 1250 per il CIPIC).

Tale limitazione implica che non tutta la sfera di possibili posizioni attorno alla testa è coperta.

Se è necessario dare l'impressione del movimento nel continuo della sorgente è necessario interpolare/predire le misurazioni effettuate nei punti non archiviati in db, a partire da quelli archiviati.



Rappresentazione grafica del metodo di misurazione

## Gestione del sistema con casse

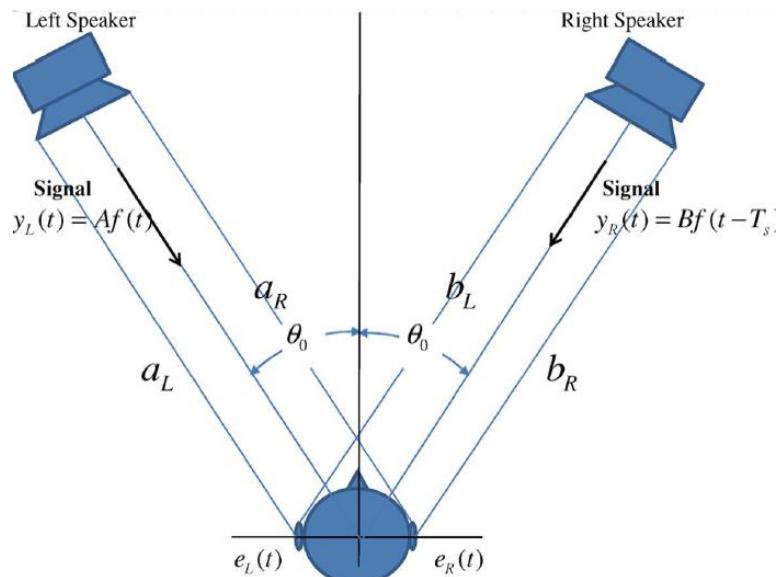
Finora abbiamo supposto di costruire un sistema audio 3D che utilizzasse delle cuffie come strumento fisico.

Se supponiamo invece di utilizzare degli speaker lontani (ad esempio delle casse), i suoni uscenti dalle due casse vengono rilevate da entrambe le orecchie, dando origine al **fenomeno del cross-talk**.

Volendo controllare completamente il suono che arriva sul singolo orecchio, avendo piena e precisa conoscenza sulla posizione dell'utente e dei suoi microfoni, conoscendo il ritardo fra un orecchio e l'altro, possiamo cancellare il suono che non dovrebbe arrivare ad un orecchio ma ci arriva lo stesso. Per fare ciò creiamo un suono ad hoc, in opposizione di fase con quello che vogliamo cancellare, tale da azzerarlo.

Questo fenomeno prende il nome di **cross-talk cancellation**: ogni cassa cancella/attenua il suono dell'altra casa nell'orecchio che non dovrebbe sentire quel suono.

Questo risultato si può ottenere utilizzando un filtro.



Rappresentazione grafica della costruzione di un sistema con speaker lontani e gestione della cross-talk cancellation



## 11. Speech recognition

Gli studi sulla **speech recognition** iniziano negli anni '60, quando nascono i primi convertitori analogici digitali, nei laboratori Bell.

La speech recognition può essere molto utile quando non esistono altri metodi per comunicare con la macchina, usando il parlato come input.

Ciò è possibile perché, a differenza della musica, il segnale audio prodotto dal parlato è tale da distinguere bene le pause (rumore di fondo), dalle vocali (caratterizzate da sinusoidi), dalle consonanti (caratterizzate da un segnale rumoroso).

Proprio per queste caratteristiche del segnale audio, la prima idea per il riconoscimento del parlato era la segmentazione dello spettro: lo spettro in frequenza viene segmentato in modo che ogni segmento contenga una lettera (riconoscibile per una data distribuzione di frequenza); dal riconoscimento delle lettere si passa alla formazione di parole (matchate con meccanismi di massima verosimiglianza con parole memorizzate in un vocabolario) e quindi di frasi.

Usare uno spettrogramma per riconoscere e considerare segmenti della trasformata è stato quindi il punto di partenza degli studi.

Il problema fondamentale di questo metodo è che il segnale raccolto è condizionabile da moltissimi fattori che lo rendono instabile: vocabolario limitato, diverse interpretazioni, soggettività delle frequenze in base al soggetto, variazione delle parole, confusione fra le parole, rumore o riverbero...

Inoltre le frequenze con cui viene emessa una singola lettera da diversi soggetti può variare molto (pensiamo alla frequenza di un uomo da quella di una bambina).

Oltre tutto la lunghezza con cui vengono pronunciate le lettere dipende dal soggetto, perciò anche la stessa segmentazione dello spettro per il riconoscimento delle lettere è difficile.

Il passo successivo, nel 1975, è quello di utilizzare un vocabolario di 1000 parole e riconoscerne almeno il 90 su 100.

Le parole che vengono raccolte sono parole isolate, non inserite nel contesto di una frase.

Questo passo avanti in realtà porta di nuovo ad un vicolo cieco, a causa della mancanza di criteri di valutazioni standard oggettivi.

La base di conoscenza sulla base della quale vengono riconosciute le lettere e le parole viene fatta inizialmente a mano da un linguista, il quale crea un sistema di regole che permettono di capire se il riconoscimento è valido o no.

Successivamente la base di conoscenza viene costruita automaticamente tramite una fase di apprendimento.

Anche questa strada subisce un profondo stop, data la mancanza di tecnologia abbastanza sviluppata e di un criterio di valutazione universale.

Dal punto di vista della procedura pratica il processo segue le fasi di:

- **digitalizzazione**: per convertire il segnale da analogico a digitale
- **signal processing**: per separare il parlato dal rumore di fondo (cosa che viene fatta anche tramite l'hardware, usando un secondo microfono che registra il rumore di fondo da eliminare)
- **fonetica**: per cogliere la variabilità del parlato umano (soggetto a accenti, emozione, raffreddore, stile del parlato, intonazione...)
- **fonologia**: per il riconoscimento e la distinzione dei suoni (ad esempio le "e" aperte o chiuse; sono i simbolini che ci sono sul vocabolario)

- **lessicologia e sintassi:** per il disambiguamento delle parole simili
- **sintassi:** per interpretare le caratteristiche della sintassi
- **prosology:** per riconoscere le parole: parole isolate sono molto più semplici da riconoscere del parlato continuo.

I sistemi di speech recognition visto fin'ora sono speaker dependent system: richiedono una fase di training (solitamente 5/10 minuti) per adattare il sistema all'utente che lo usa.

Questa fase li rende più robusti ma meno convenienti e portabili.

L'idea quindi è quella di spostarsi su sistemi *speaker independent* che, anche al costo di un minore riconoscimento, si adatta a qualsiasi utente.

Per effettuare questo passaggio si passa dal sistema a regole ad un sistema basato sull'approccio statistico: le regole usate precedentemente restituiscono un valore percentuale di correttezza, invece di un secco sì/no. Inoltre il match del segmento viene effettuato con una grande quantità di dati memorizzati in db, che forniscono una statistica ricca.

Inoltre, data la grande mole di dati a disposizione, è possibile utilizzare il *machine learning* per creare le regole e un meccanismo di massima verosimiglianza per trovare la parola più verosimile.

