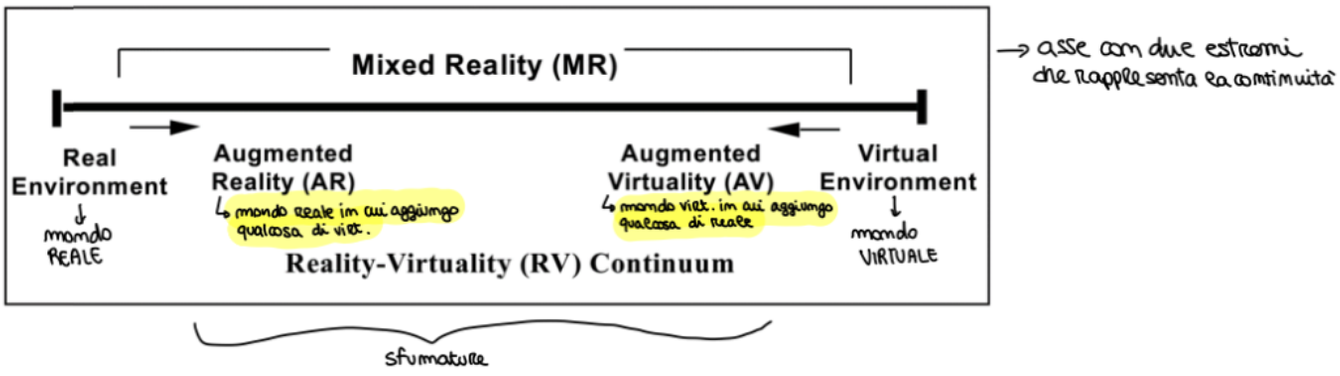
REALTA' VIRTUALE
Ambiente simulato in cui l'utente può interagire con il mondo virtuale.
L'utente riceve stimoli che sono generati artificialmente e ha l'illusione di trovarsi in un mondo virtuale. (es. videogiochi in prima persona).
Livello di immersione basso se si guarda da uno schermo, basta poco per uscire dalla finzione.

Livello di immersione: - basso (monitor)
- parziale (wide sorrounding monitor)
- totale (Samsung Gear VR → HMD Head Mounted Display)

Immagini: - reali: mediate dal computer
- virtuali: interamente create dal computer

VIRTUAL CONTINUUM (continuo virtuale)

Tra i due estremi (virtuale – reale) esistono delle sfumature.



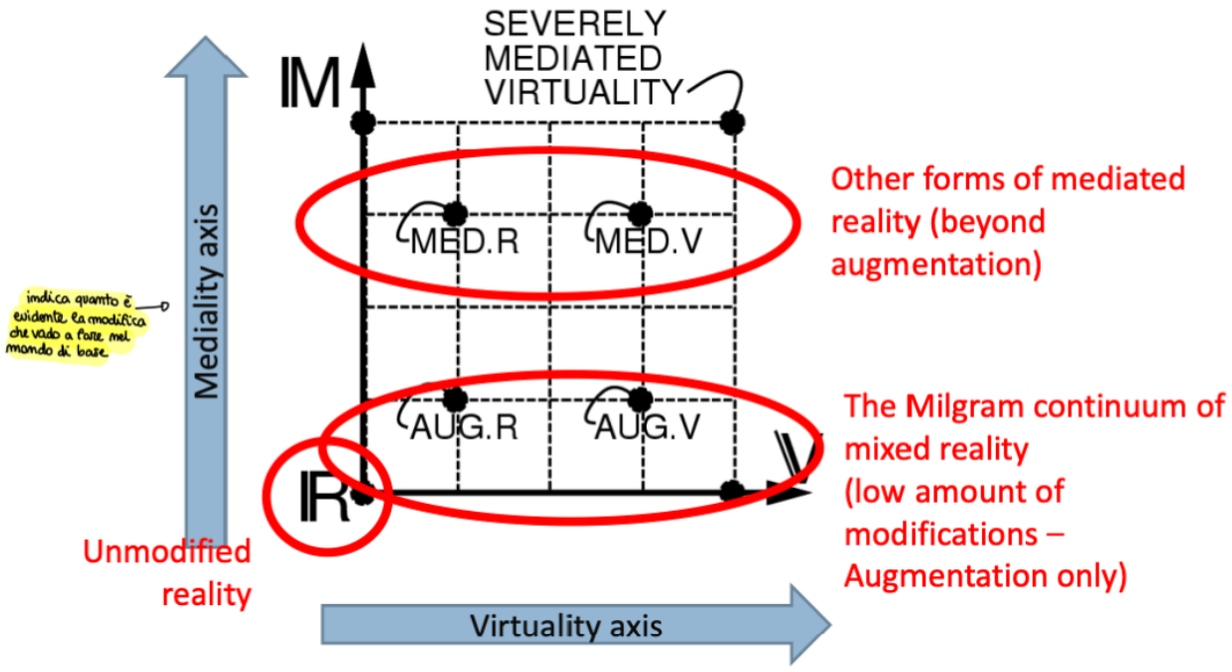
MILGRAM denota con **MIXED REALITY** tutto lo spazio che c'è tra il reale ed il virtuale, ovvero gli estremi dell'asse di Milgram.

Questo modello contiene:
- **AUGMENTED REALITY** (es. app Ikea che mostra gli oggetti nella stanza)
- **AUGMENTED VIRTUALITY** (es. meeting fatti nel mondo virtuale, telegiornali)

Questo modello contiene solo aggiunte al mondo reale/virtuale, non considera le rimozioni.
(es **realtà diminuita** o **realtà modulata**).

MEDIATED REALITY

Mann estende il diagramma di Milgram aggiungendovi una dimensione (la **medialità**) che rappresenta il numero di modifiche, ovvero quanto forti sono le modifiche sul mondo di base.



Asse x : quanto sei vicino al mondo reale o virtuale;
Asse y : quanto forti sono le modifiche applicate al mondo di base.

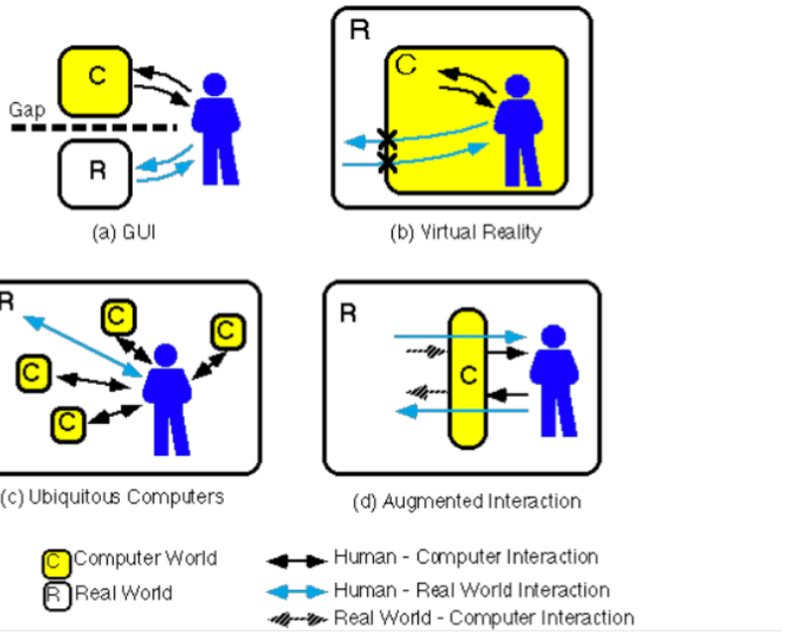
Questo modello comprende :
- **realtà diminuita**: rimuovo alcuni componenti dall'ambiente (importante che sia in tempo reale !)
- **realtà mediata** : altero la realtà sostituendo, es le telecamere notturne ad infrarossi.

REALTA' AUMENTATA

E' un'esperienza **interattiva** (ovvero REALTIME, per essere interattiva), **diretta o indiretta**, di un ambiente nel **mondo reale** dove gli **oggetti che risiedono nel mondo reale sono aumentati** dalle informazioni percettive generate dal computer attraverso diverse modalità sensoriali (es. informazioni visive o audio)

Visione diretta: vedo direttamente sopra al mondo reale, ovvero ho uno "schermo" trasparente (es. Hololens)
Visione indiretta: vedo attraverso lo schermo di un tablet cosa viene inquadrato dalla fotocamera.

Ci sono 4 modalità di interazione con il mondo reale:

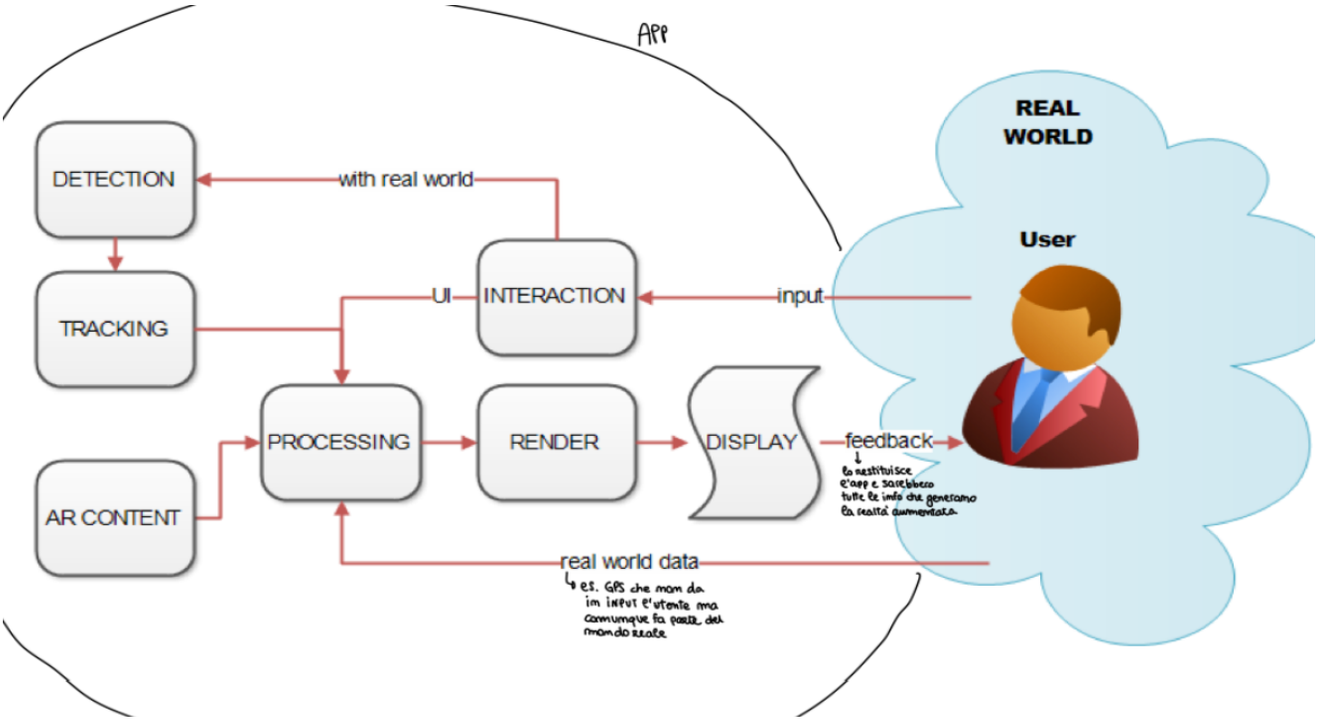


a. Utilizzo tipico di un computer, mondo reale diviso da quello virtuale
b. Utente chiuso all'interno del mondo virtuale e tutto quello con cui interagisce arriva dal mondo virtuale. Non c'è contatto con il mondo reale esterno.
c. -
d. Realtà aumentata, interazione con il mondo virtuale anche se mi sembra di essere nel mondo reale.

Campi applicativi:
- commercio (IKEA o sito che ti fa provare gli occhiali da casa)
- intrattenimento (snap)
- architettura
- turismo (musei)

ARCHITETTURA DI UN SISTEMA DI REALTA' AUMENTATA

Il sistema elabora degli input e restituisce un output sotto forma di informazioni sensoriali.
Il tempo che intercorre tra input e output deve essere il più breve possibile.



INTERAZIONE: può essere

- **diretta**: l'utente interagisce con una GUI o con comandi vocali;
- **indiretta**: sono delle azioni che facciamo nel mondo reale (es. spostamento) e l'app le interpreta tramite accelerometro, giroscopio, GPS..

Per riconoscere all'interno del video esistono due tipi di **marker**:

- marker **fiduciali:** pattern specifici artificiali, solitamente stampati in bianco e nero (per facilitarne il riconoscimento, inseriti nella scena (mondo reale) e hanno forme nette e squadrate. (rilevati da **jsARToolKit**)

- marker **naturali**: il sistema riconosce delle immagini naturali e li usa come marker fiduciali, infatti ogni oggetto può diventare marker → più complesso e può portare a problemi di rilevazione. (**markerless**)
  ↳ occlusione, illuminaz, oggetti simili

Il marker deve essere interpretato per capire come è posizionato nell'ambiente 3D, passaggio dal 2D (fotocamera) al 3D

↳ solo per le INTERAZ. INDIRETTE

DETECTION: non ci focalizza su un solo elemento, si fa una **spatial mapping** ovvero si mappa tutto l'ambiente circostante → costruisco un modello 3D di tutto ciò che mi circonda.

Alcune rilevazioni devono essere fatte su dati complessi come audio e video, difficili da interpretare.
es. COMANDI VOCALI (interaz. DIRETTA)
TRADUZ. AUTOMATICA (interaz. INDIRETTA)

TRACKING: riconosce un input da elaborare (la presenza di un marker) e tiene traccia temporalmente di ciò che è stato tracciato (coerenza temporale, ovvero deve riconoscere anche a distanza di tempo se il tracker è lo stesso di quello che ha scannerizzato prima).

AR CONTENT: tutto il possibile contenuto che posso aggiungere al mondo reale, è un set di tutti i possibili elementi aggiungibili (database), possono essere già inclusi nell'applicazione o trovarsi in una sorgente esterna.

PROCESSING: il cuore del sistema, prende i dati dal mondo reale e li aumenta con i contenuti AR. Un passo fondamentale è la **registration** (allineamento), infatti i contenuti AR e il mondo reale devono essere ben allineati altrimenti rischio di avere poca percezione → la posizione deve aggiornarsi in tempo reale. (three.js)

Si possono creare degli **errori di allineamento**:

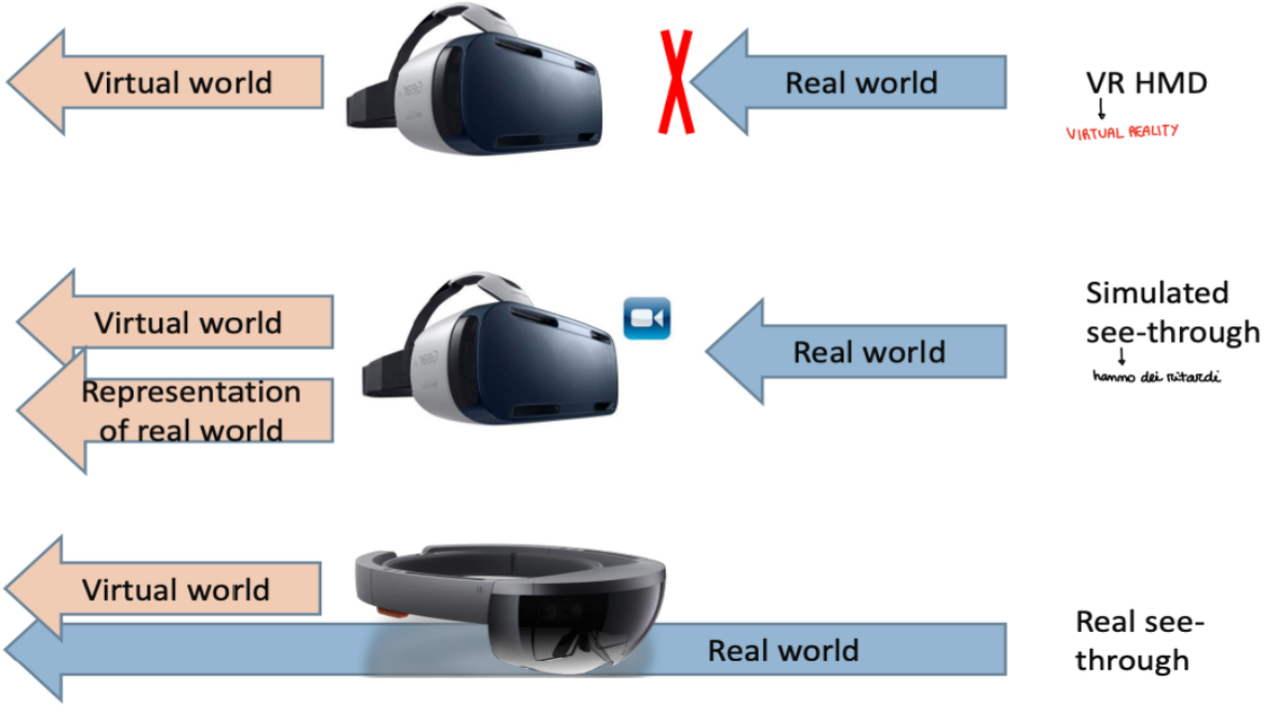
- **statici** : non dipendono dal movimento (dist. ottiche (ricalibro la telecamera offline) e parametri mal impostati (reimposto i parametri all'inizio))
- **dinamici** : ritardi di sistema (cambio sw) o errori di tracking (cambio tracker)

RENDERING: generare l'immagine finale → dal modello 3D ottengo l'immagine da visualizzare. Con un marker fiduciale il processing rileva la posizione del marker (in 3D) e il rendering lo disegna.

DISPLAY: i monitor offrono un basso livello di integrazione con il mondo reale, una migliore integrazione è offerta dagli smartphone che permettono la libertà di movimento.

HEAD MOUNTED DISPLAY

Ne esistono di 3 tipi:

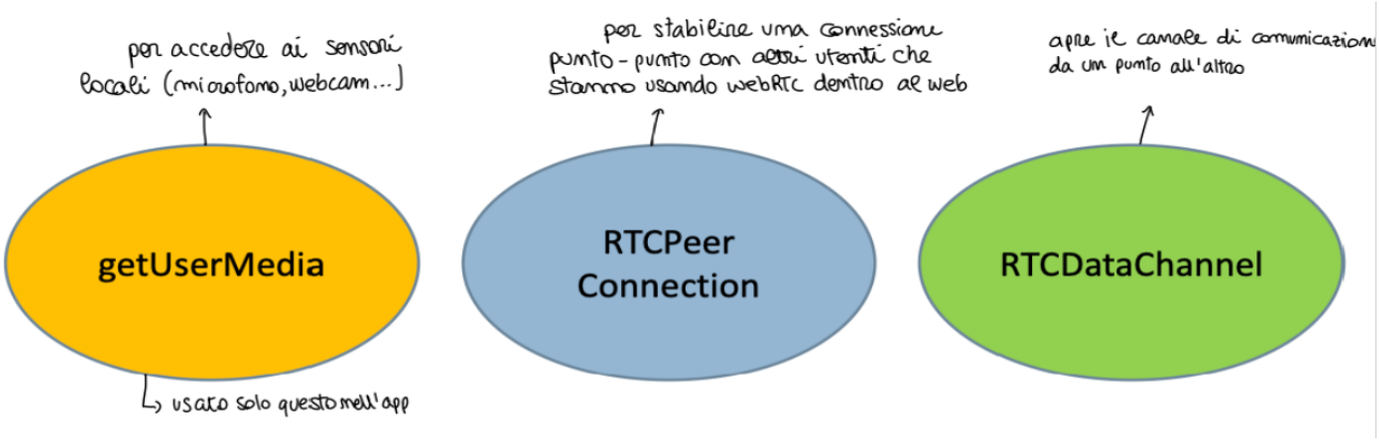


Gli **eyeglasses** sono dei particolari HMD, simili a degli occhiali, che permettono il **see-through** ed il sistema è tutto integrato nei glasses.

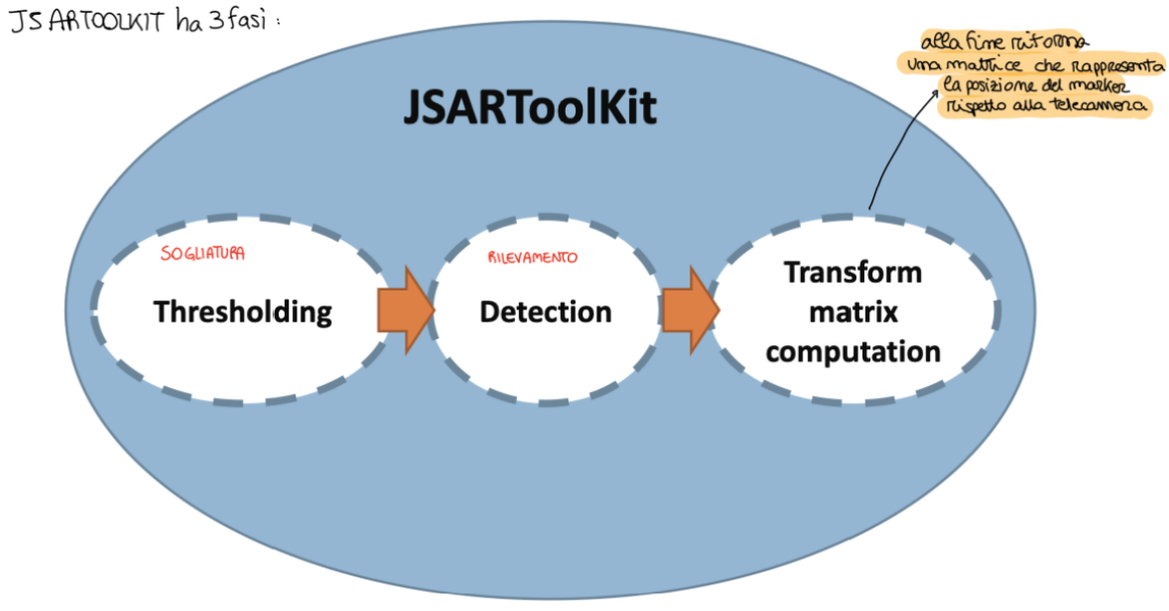
Viene usato HTML 5 perché rispetto al predecessore (HTML 4) permette il supporto ai contenuti multimediali, nello specifico noi abbiamo usato i tag **<canvas>** e **<video>**.

Il tag **<canvas>** è un'area generica di disegno in cui, con API specifiche, posso disegnarvi su figure geometriche ed immagini.
E' una griglia di 4 pixel (rosso,verde,blu,alpha-trasparenza) che ci permette di disegnare contenuti aumentati sopra l'immagine reale.

webRTC serve per implementare applicazioni di REAL TIME COMUNICATION sul web, noi la utilizziamo solo per la funzionalità getUserMedia (accesso alla webcam).
Ha tre componenti fondamentali:



jsARToolKit permette, una volta avere ricevuto un canvas, di ricavare una lista di marker e matrici che descrivono la posizione del marker rispetto alla telecamera



Processo:
- elimino i colori (porto in scala di grigi, passo quindi da un RGB ad un solo campo che per ogni pixel mi dice la sua % di grigio);

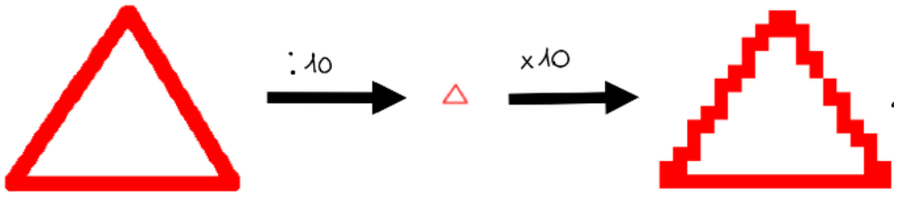
- trasformo in **bianco e nero** (**threshold**) e decido una **soglia** sotto cui tutto quello che sta sotto diventa nero, il resto bianco.

```
var detector = new FLARMultiIdMarkerDetector(param,76);
```

- 76 is the real size of the markers to be detected, in millimeters → quando passiamo da 3d a 2d perdo una dimensione → AMBIGUITÀ vedrei uguali un marker piccolo vicino alla fotocamera e uno grande lontano dalla fotocamera con questa misura aiuto ISARtoolkit a capire dove si trova ie marker

This real-world size is important for detection. Can you figure out why?

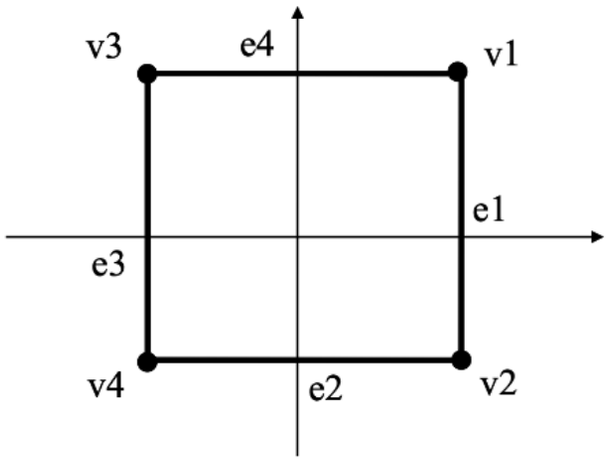
IMMAGINI RASTER: la maggior parte delle fotografie con cui abbiamo a che fare, composte da una griglia di pixel colorati.
Lo svantaggio è che non contengono informazioni semantiche sull'immagine, infatti il computer la vede solo come insieme di pixel e non sa niente dell'immagine.
Quando manipolo l'immagine perdo informazione, infatti se rimpicciolisco di 10 e poi ingrandisco di 10 non avrò mai la risoluzione iniziale. (non perdo informazione solo ruotando di 90 gradi)



IMMAGINI VETTORIALI: modo alternativo per preservare le informazioni geometriche dell'immagine, al posto che salvare un insieme di pixel salvo un insieme di informazioni geometriche (punti (**vertici)** e linee (**archi**)).
Ogni figura viene descritta da due set, archi e vertici.

Vertici →
V = { v1(1,1),
v2(1,-1),
v3(-1,1),
v4(-1,-1) }

Lati →
E = { e1(v1,v2),
e2(v2,v4),
e3(v4,v3),
e4(v3,v1) }



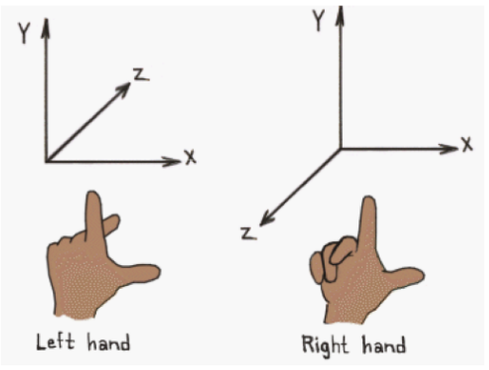
Un esempio di grafica vettoriale sono i font.
Le **curve** possono essere rappresentate in due modi:
immagine diventa più pesante
- approssimate con i segmenti: maggiore è il numero di punti, migliore è l'approsimaz.
- usando le curve di Bézier: $B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$
  $\Rightarrow t \in [0,1]$

curve di Bézier: definisci un punto di inizio e uno di fine, due punti che individuano delle tangenti che mostrano la direzione di entrata e uscita della curva.

⇒ nella grafica VETTORIALE posso avere linee curve mantenendo l'informaz. semantica

Gli artisti 3d lavorano solo su immagini vettoriali in modo da non perdere informazioni durante le trasformazioni ed infine quest'ultime vengono convertite in immagini raster alla fine del processo di modellazione (rendering).

I tre assi definiscono un sistema di riferimento, non tutti i sistemi sono uguali.
Bisogna definire:
- quale asse punta in alto e dove punta l'asse z
- se il sistema è **right-handed** o **left-handed** (non esistono trasformazioni per passare da uno all'altro).



Nel 3D un **segmento** è sempre identificato da due punti (x,y,z)
Un **poligono** è sempre una superficie 2D, perciò potrebbe essere che per più di 3 punti non esista un poligono che passi per essi (se ho vertici che non sono complanari, potrebbe non esistere).

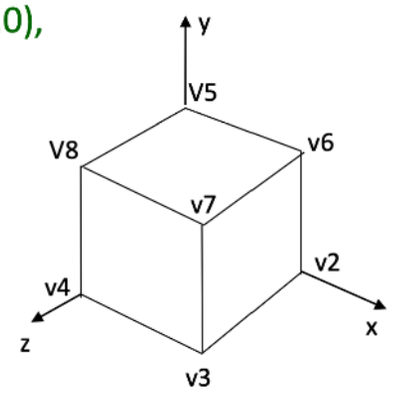
In ogni caso un poligono passante per 3 vertici esiste sempre, ed è il **triangolo**.
Con **TRIANGOLAZIONE** si intende il processo in cui ogni poligono generico può essere scomposto in un insieme di triangoli, la decomposizione non è unica ma il numero di triangoli che lo compongono si.

Pro: non esistono poligoni (che stanno su piani diversi) che non possono essere rappresentati
Con: aumenta la complessità dell'immagine.

Unendo più poligoni creo i **POLIEDRI** che sono insiemi di poligono uniti tra di loro da lati e vertici.
Infatti il poliedro è la struttura più utilizzata nella modellazione 3D per definire forme tridimensionali.

Un poliedro è descritto da un set di vertici, facce (poligoni) e lati.

V = {v1(0,0,0), v2(1,0,0), v3(1,0,1), v4(0,0,1), v5(0,1,0), v6(1,1,0), v7(1,1,1), v8(0,1,1)}



E = {e1(v1,v2), e2(v2,v3), e3(v3,v4), e4(v4,v1), e5(v5,v6), e6(v6,v7), e7(v7,v8), e8(v8,v5), e9(v1,v5), e10(v2,v6), e11(v3,v7), e12(v4,v8)}

P = {p1(e1,e2,e3,e4), p2(e5,e6,e7,e8), p3(e3,e12,e7,e11), p4(e1,e9,e5,e10), p5(e4,e12,e8,e9), p6(e2,e11,e6,e10)}

Nella grafica 3D al posto che i pixel ci sono i **voxel**, che sono cubetti.
Serve per "riempire" i poliedri che, a differenza della classica rappresentazione poligonale, possono essere utilizzati ad esempio in ambiti medici e vanno quindi sezionati all'interno.
La sua controparte è la grafica raster.

Le superfici curve vengono gestite:
- descrizione matematica (NURBS)
- approssimazione poliedrica

Le **trasformazioni** di base sono rappresentabili dal prodotto $matrice * vettore$:
- **rotazioni**
- **cambiamento di scala**
- **traslazioni**

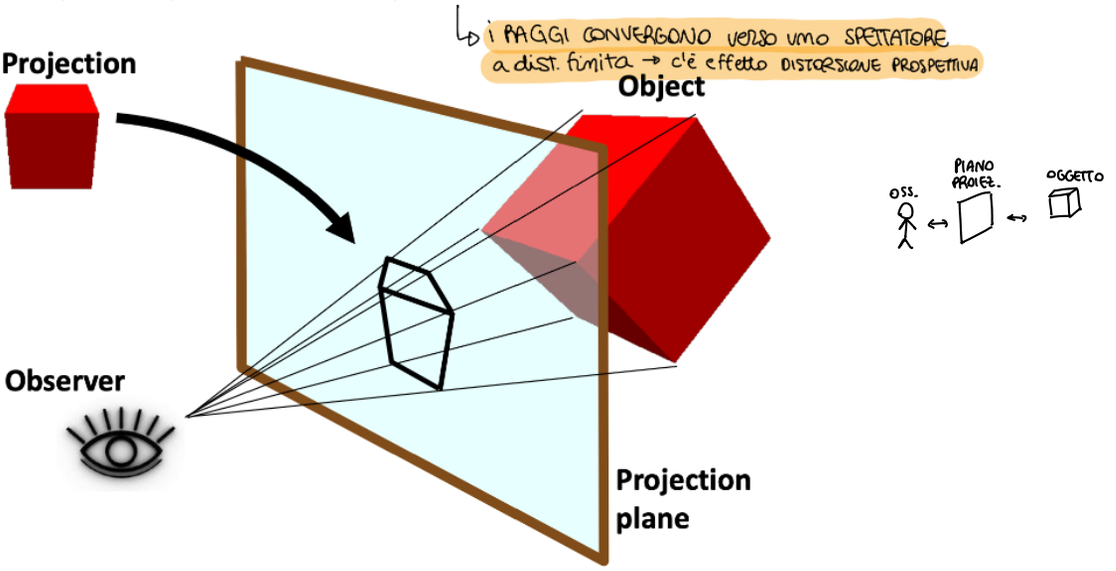
Possono essere:
- **locali**: relative ad un singolo oggetto
- **globali**: relative al sistema di riferimento fissato

Il **pivot** è il centro della trasformazione, ovvero è il punto che non si sposta durante la trasformazione.

I modelli 3D noi li vogliamo rappresentare su uno schermo che è 2D, perciò abbiamo bisogno delle **proiezioni**.

PROIEZIONE PROSPETTICA

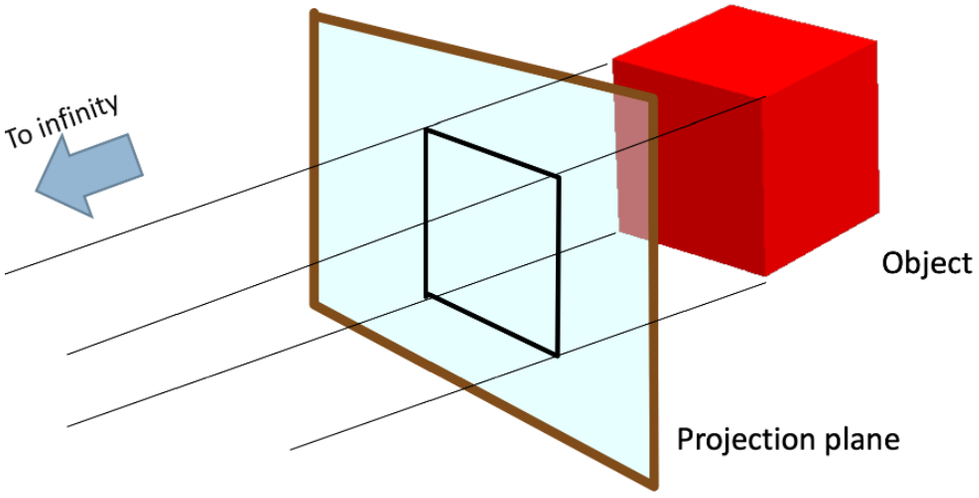
Proiezione in cui l'osservatore è a **distanza finita** dall'oggetto e soffre di **distorsione prospettica**, ovvero gli elementi più lontani tendono ad essere visti ridotti (perché' i **raggi tendono verso lo spettatore** e quindi si perde il senso di parallelismo).



Il parallelismo viene solo rispettato qualora le due linee siano parallele al piano di proiezione.
Le linee parallele nel mondo 3D sembrano non essere più nella proiezione, convergenti verso un punto (PUNTO DI FUGA)

PROIEZIONE ASSONOMETRICA

Proiezione in cui l'osservatore è a distanza infinita dall'oggetto, non soffre di distorsione prospettica e i raggi di luce sono paralleli.



Utile quando è importante mostrare chiaramente le relazioni spaziali tra gli oggetti.
Linee parallele in 3D lo sono anche in proiezioni assonometriche.

Le **proiezioni ortogonali** sono solo delle proiezioni assonometriche applicate sui singoli piani XY, YZ, XZ.

In una proiezione posso mostrare:
- solo vertici
- solo lati
- facce: devono avere un colore ma se tutte le facce hanno lo stesso colore si avrebbe una percezione di un oggetto "piatto".

Per visualizzare un oggetto serve un **materiale**, ovvero l'insieme di cose che descrive come deve essere la **resa visiva** di un oggetto. (colore, trasparenza, luminosità, geometria..)
↳ modo in cui l'oggetto reagisce alla luce

la TEXTURE è un possibile ATTRIBUTO, ovvero un' immagine incollata sulla superficie di un oggetto

WebGL: API che permette di mostrare della grafica 3D interattiva senza usare plug-in esterni. E' molto potente e veloce (opera a basso livello → sfrutta hw della GPU), ma è molto complesso.

Per semplificare l'utilizzo di WebGL usiamo una libreria grafica 3D ad alto livello, **Three.js**

Ogni TRASFORMAZIONE che voglio applicare nella grafica 3D è una TRASFORMAZIONE sui vertici

- CAMBIAMENTO DI SCALA

(non om.)

$$\begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} Sx\,x \\ Sy\,y \\ Sz\,z \end{pmatrix}$$

$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \xrightarrow{h} \begin{pmatrix} xh \\ yh \\ zh \\ h \end{pmatrix} \xrightarrow{:h} \begin{pmatrix} x/h \\ y/h \\ z/h \end{pmatrix}$

(omogenee)

$$\begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- TRASLAZIONI (devo aggiungere una dimensione)

(omogenee)

$$\begin{pmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ h \end{pmatrix} = \begin{pmatrix} x+Tx \\ y+Ty \\ z+Tz \\ h \end{pmatrix}$$

- ROTAZIONI (nel 2D attorno ad un punto, nel 3D attorno ad un asse)
  ↳ ANTIORARIA

(non om.)

ASSE x:
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

(omogenee)
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

ASSE y:
$$\begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}$$
$$\begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

ASSE z:
$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$