La computer grafica ha 2 tipologie di rendering quello **offline** (immagini, film, fotorealismo) che non è time critical e **realtime** (CAD, giochi, AR) è time critical e per essere considerato relatime deve avere un framerate di 24fps (tra 1 e 5 è rendering interattivo. Gli oggetti 3D sono chiamati **mesh** e sono composti da triangoli, più triangoli ci sono maggiore sarà la risoluzione. Insieme triangoli=mesh, insieme di mesh = oggetto, insieme di oggetti= mondo 3D.

**Raycasting**: è una tecnica che consiste nel lancio di raggi dalla posizione dell'osservatore verso un piano e l'identificazione dell'intersezione con gli oggetti 3D. Questa procedura viene ripetuta per ogni punto sulla risoluzione dello schermo, per determinare se il punto è illuminato o coperto. La fonte di luce lancia raggi verso gli oggetti e se non viene intercettato, il punto viene illuminato.

**Raytracing:** è un’evoluzione del raycasting, ma aggiunge riflessi speculare, rifrazione e ombre. Usa un approccio ricorsivo e si ferma dopo N iterazioni. **Pipeline= world transform ->raytracing**

**Rasterizzazione:** è più veloce del raytracing, ma è più complessa. Si vasa sull’approssimazione di un’immagine su una griglia di pixel. Lo schermo aggiorna lo stesso numero di pixel in altezza e in larghezza.

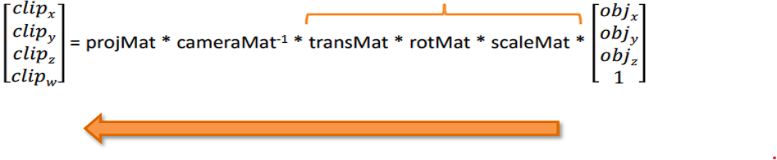
**PP: world transform->lighting->projection transform->clipping->rasterization**

**Le trasformazioni globali** racchidono tutte le operazioni di traslazione, rotazione, scaling; **lighting** consiste nel calcolo dell’illuminazione diretta vertice per vertice; **projection transfom** consiste di proiettare i triangoli sullo schermo passando da 3d a 2d si divide x e y per z e z viene rimosso. Near plane< scena < far plane, il punto di fuga è sulla telecamera nelle telecamere prospettica (tronco di cono). **Clipping** consiste nel rimuovere i vertici che sono oltre il far plane e netro il nearplane. Rieffettua la triangolizzazione dei pezzi di scena che sono renderizzati parzialmente. **Rasterizzazione** dal risultato degli step precedenti otteniamo i vertici da renderizzare tramite “l’algoritmo della linea” si va a ricreare la mesh approssimando la figura sui pixel con i relativi colori.

Pixel: punto su schermo Punto: oggetto matematico in OpenGL punto=pixel. Punto no lunghezza, spessore e direzione

Vettore indica movimento nello spazio (direzione, verso, norma) normalizzazione vettore (componenti/norma). **Se a e b sono normalizzati (tra 0 e 1) l’arco coseno del prodotto scalare è pari all’angolo compreso tra i due vettori, il prodotto è il coseno dell’angolo compreso (usato nelle luci).** Il prodotto vettoriale è usato per il calcolo delle normali. Le matrici servono per le trasformazioni geometriche, su OpenGL si usa la post-moltiplicazione Matrice\*vettore=vettore**, il numero di colonne deve essere uguale alle righe del vettore b** (cambia in base alla libreria). La moltiplicazione tra matrici non è commutativa. Inversione si usa Cayley-Hamilton. Colori RGB o RGBA vettori da 3 dimensioni per RGB e 4 per RGBA dove A è il canale alpha (trasparenza). Lo scaling può essere utilizzato per creare ombre statiche sul piano. Metto a 0 la dimensione che voglio togliere. Oppure posso effettuare flip cambiando di segno(riflessi). Traslazione **vettore posizione+vettore traslazione**, **rotazione =matrice Rotazione\*vettore posizione.** Coordinate omogenee. **Le coordinate omogenee consentono di ottenere tutte le operazioni esprimibili matrice\*vettore in una singola matrice di trasformazione, aggiungendo una dimensione w solitamente pari a 1. Traslazione T\*v, rotazione R\*v, Scaling S\*v, Project P\*v. prodotto rigaM\* colonnaV**

Data la trasformazione **vn=T3\*T2\*T1\*Vp** la moltiplicazione avviene vp\*t1\*t2\*t3

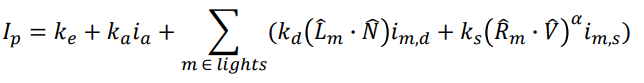
Spazi di coordinate: modello, globali (centro del mondo (0,0,0) trasformazioni\*matrice modello), camera (centro schermo (0,0,0) inversa della camera\*globali), clip-coordinates( matrice proiezione\*eye coordinates) sono tra 0 e 1 (**normalizzazione dividendo per w** si torna a vettori di 3 dimensioni). Matrice di proiezione può essere **ortografica** nella quale all’allontanarsi o avvicinarsi l’oggetto ha dimensione costante oppure **prospettica** nella quale lo scaling varia in base alla lontananza/vicinanza dell’osservatore**. Z nelle proiezioni è invertito cambia in sistema mano sinistra più vicino allo schermo Z positivo**. Il Field of view nella prospettica è l’angolo di visione (angolo del cono). In prospettica z copiato su w e invertito. **Screen coordinates** sono le cordinate sullo schermo e si ottengono: **viewport transfomation\*clip coordinates** . la cordinata z viene utilizzata per il texturing e per lo zbuffer per determinare cosa va davanti a cosa, **posizione rispetto al near plane** (valori tra 0 e 1). Interpolazione matrici si può fare con decomposizione delle matrici, si possono ottenere i vari parametri di scale, rotazione e traslazione; oppure usare i quaternioni (numeri complessi con 3 parti immaginarie). OpenGl no user input, GUI solo rendering ->freeglut consente l’input tramite callback.

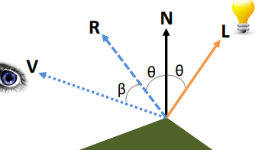
OpenGL è una libreria di API grafiche, **non è uno standard,** per utilizzare opengl bisogna inizializzare un **contesto** ossia un contenitore in cui i driver salvano lo stato delle api e le risorse grafiche da utilizzare. Freeglut è una semplificazione di openGL. Ci sono 3 tipi di buffer:

Immagine che contiene testo, arancia

Descrizione generata automaticamente**Main Buffer:** sono aree di memoria che contengono informazioni legate ai pixel renderizzati su schermo. **Framebuffer:** contiene le immagini da inviare allo schermo. **Back buffer:** è un buffer di supporto al framebuffer in cui vengono caricate le immagini da renderizzare il frame successivo, se venisse caricato subito si vedrebbe flickering causato dall’aggiornamento. **Z-Buffer of Depth buffer:** immagazzina il valore z per ogni pixel. **RGB**(1bit: monocromatico, 4 bit =16 colori, 8 bit = 256 colori), **RGBA** (15bit =215 colori, 16bit =216 colori, 24bit colori), **RGBA**(32bit =24bit+8 bit di alpha, >32bit HDR), l’accesso al framebuffer dipende dal refresh del monitor, l’aggiornamento della memoria è sincronizzato con il monitor per evitare tearing e sfarfallio. Il v-sync sincronizza il framebuffer con il monitor. **Si renderizza tutto sul backbuffer e una volta terminato il rendering si fa swap nel framebuffer favorendo le prestazioni**. GL\_TRIANGLE\_FAN crea triangoli prendendo l’ultimo vertice del triangolo precedente, utile per forme con origine in comune, GL\_TRIANGLES\_STRIP crea triangoli partendo dagli ultimi 2 vertici, triangoli connessi. GL\_VIEWPORT imposta la dimensione della finestra, si richiama quando la finestra viene ridimensionata. glMatrixMode(GL\_MODELVIEW) imposta il contesto in world coordinates, cambiando in PROSPECTIVE o ORTOGRAPHIC si impostano le telecamere dopo matrix mode si può caricare la matrice che si vuole caricare glLoadMatirix()(proiezione, di posizionamento). **OpenGL funziona come macchina a stati**. Posizione=t: x y z indicano le direzioni. **Stencil buffer**, immagini, effetti, maschere, visualizzazione, nascondimento, forme, applicazione, aree specifiche.

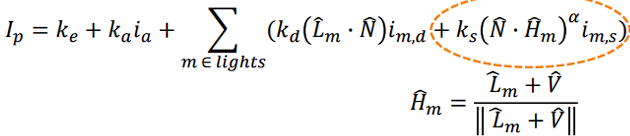
Lookat consente di mantenere il contatto visivo della telecamera con un oggetto. Z buffer z è compreso tra -1 e 1 sono ricalcolati tra 0 e 1. Contiene la distanza perpendicolare per ogni pixel relativa al nearplane, precisione logaritmica. Lo stencil buffer è un buffer opzionale che ha la stessa dimensione di colori dello z buffer e ha una profondità di 8bit, è usato per limitare il rendering di pixel-precise area come riflessi planari. **Illuminazione: ogni vertice ha specificate le coordinate e il vettore normale al piano**, i vertici appartengono a un triangolo quindi a un piano.

****L’output dell’illuminazione è un vertice colore rgb. **Il modello di illuminazione calcola per ogni vertice non per ogni pixel.** Il triangolo normale si ottiene facendo il cross product tra due vettori tra v1-v2 e v1- v3, ottenendo un vettore perpendicolare al piano. Un triangolo ha 2 facce e l’orientamento dipende da come vengono passati i vertici (CCW e CW), il **culling** consente di disabilitare il rendering delle facce interne di un triangolo, migliora le prestazioni. **Il vettore normale va normalizzato**. Vettori normali vengono moltiplicati per per l’inversa della matrice trasposta dell’ultima matrice caricata in **modelview**. Lo scaling richiede la rinormalizzazione, con **glEnable(gl\_Normalize)** si attiva la normalizzazione automatica. Se si ha un vettore normale per vertice viene calcolata la media di tutti i vettori normali in quel punto, distribuendo l’illuminazione per ogni faccia alle quali esso appartiene. La superfice di un mesh non è perfettamente liscia, ma microsfaccettature. L’allineamento delle facce dipende dalla rugosità della superfice. I raggi rimbalzano sulla faccia e determinano l’illuminazione. **Modello di Phong:** illuminazione si compone di **Ambient:** rappresenta la luce ambientale uniforme che circonda l’oggetto, **Diffuse:** rappresenta la quantità di luce che viene riflessa dall’oggetto in modo uniforme in tutte le direzioni (si basa sulla legge del coseno di Lambert) non è importante la posizione dell’osservatore. **Specular:** la luce rimbalza sulla superfice e riflette in una direzione specifica (effetto brillantezza). **Emission:**  è un parametro aggiuntivo che consente di simulare l’emissione di luce, è un glColor con priorità sul modello di illuminazione. **Ip**=superfice, **Ke**=materiale emission, **Ka**=materiale ambientale, **ia**=proprietà ambient della luce, **kd**=materiale difffuse, **Lm**=vettore direzione verso la luce, **Nm**=vettore normale, **ks**=materiale specular, **Rm**= vettore raggio di riflessione perfetta, **V**= vettore direzione osservatore, **alpha**=shininess.In blinn-phong **H:** vettore medio tra osservatore e luce.

Immagine che contiene testo, orologio, calibro

Descrizione generata automaticamenteLa componente speculare è proporzionale al coseno dell’angolo beta compreso tra il vettore della luce e l’osservatore (motivo per cui si fa il prodotto scalare). il modello di blinn è una variante computazionalmente più efficiente, la quale tramite il vettore H che è il vettore che taglia a metà l’angolo luce-occhio, non ha mai unangolo maggiore di 90 gradi. OpenGL usa Blinn-Phong. Le luci direazionali sono luci con direzione infinita, hanno w=0, sono infinite per questo motivo poiché si va a normalizzare dividendo per 0 quindi un valore che tende a infinito, **il vettore posizione è usato come vettore L**. Le omidirectional light sono luci con un cutoff pari a 180 gradi, non hanno una direzione. Le spotlight sono luce che simulano una torcia con un cono di illuminazione settato tramite il cutoff che arriva a un massimo di 90 gradi, l’expont dice come deve essere l’area circostante. L’attenuazione consente di simulare la dispersione della luce nell’ambiente può essere costante, lineare o quadratica.

**Immagine che contiene testo, orologio, arancia

Descrizione generata automaticamenteIf**=illuminazione finale, **Ip**=output modello, **kc**=fattore costatante, **kl**= fattore lineare, **kq**=fattore quadratico, **d**=distanza della fonte di luce. I opengl si possono avere massimo 8 fonti di luce. **L’attenuazione funziona solo con le omni e le spot.** Buona illuminazione aumenta tassellazione oppure usare shaders se il modello si basa sui pixel. Gouraud shading: i vertici sono interpolati sulla superfice, non si vedono i lati se le normali sono correttamente settate. **Texture mapping:** viene moltiplicata per il modello di illuminazione e anch’essa usa delle coordinate. Ossia dei punti di ancoraggio sulla mesh. Durante la rasterizzazione viene aggiunta la texture alla mesh, vram. Texel dati della texture non ancora su schermo, che una volta su di esso sono pixel. Le texture possono essere rettangolari, in passato dovevano essere potenze di 2. Le texture vanno normalizzato per essere indipendenti dalla scala del modello a risoluzioni diverse. S=U=x dimension t=V=ydimenstion r=W=zdimension. Le texture non usano interpolazione lineare, texture malformata altrimenti per ogni triangolo. Si sfrutta la coordinata z per capire come mettere la texture in un contesto 3D. **Texture wrapping**: consente di impostare se lr coordiante vanno oltre il range 0-1, si può ripetere la texture oppure stirare l’ultimo lato. Solitamente sono bitmap RGB. Le texture tile sono texture disegnate per essere ripetute. Con il tiling si riasparmia nella tassellazione. Le texture sono già rasterizzate, allontano dalla texture minificazione(texture più piccola), se mi avvicino magnificazione(ho bisogno di più pixel per replicare la texture). **Filtro Lineare** (Lineare, Nearest) la minificazione causa jittering, magnificazione scattering, nearest prende il centro dl pixel più vicino mentre lineare fa una media pesata dei 4 pixel vicini, di defualt è nearst. Il filtraggio necessita di computazione aggiuntiva, accelerazione hw. **Mipmapping:** vengono creati dei livelli di texturizzazione già filtrati e vengono applicati da opengl in base alla dimensione. Le texture sono prefiltrate diverse volte per avere un migliore LOD. Il mipmapping si può fare in maniera procedurale partendo da un immagine più definita e scalare per i vari livelli. Se si trava tra 2 livelli si fa una media e si intepolano i valori rgb. Se è presente il filtro lineare la mipmap è filtrata 2 volte (bilineare) se si usa il trilineare le mipmap e le sottoimmagini sono interpolate tra i 2 lod più vicini. **Filtro anisotropico**: il filtro anisotropico consente di mantenere la stabilità del filtro trilineare e mantere la qualità. È un filtro pesante perché aumenta i texel per fare le medie. Dipende anche dalla dimensione dei triangoli. In opengl per motivi di performance, carica una volta e riusa più volte, zona di memoria dedicata. **Formati 2D:** raster, vettoriali, ibride(pdf), palette di colori (id identificativo del colore), **color banding**: problema causato dalla mancanza di precisione di gradienti usati per ombreggiare il colore. **Dithering:** è una forma di controllo della distribuzione del rumore per ridurre arteffati causati dalla mancanza di risoluzione. **DUMP:** formato raw non portabile.**PNM:** formato portabile in modo da essere interpretato da un altro utente(PBM,PGM,PPM). **TGA:** è un formato di file immagine che viene utilizzato per immagini a 24 bit e 32 bit, Intestazione: contiene informazioni sul formato, la dimensione e la risoluzione dell'immagine, Dati sulla palette: se l'immagine è a 8 bit, questa sezione contiene la palette di colori utilizzati. Dati sulle immagini: questa sezione contiene i pixel dell'immagine, che possono essere compressi o non compressi. Footer: opzionale, contiene informazioni sulle estensioni del formato TGA. Supporta la compressione come RLE. **JPEG:** Compressione lossy, dati modificati, immagine decompressa non uguale a originale, formato JPEG non consigliato per editing di partenza, trasformazione colore, luminanza e crominanza, trasformata di Fourier, codifica RLE, decompressione, perdita di qualità dipendente dal fattore scelto. **SVG:** formato open standard www, usato nel web ed è basato su xml, può essere renderizzato dai browser. **PDF:**  formato portabile, hardware indipendent, usato per definire documentic on layout statico, supporta grafiche raster e vettoriali e modelli 3D. **Collada**: primo standard, formato .dae (xml). **OBJ**: introdotto da wavefront, formato ascii, non supporta luci, animazioni, camere e gerarchie, mesh unico, ha un file dei materiali associato. **MTL**: contiene informazioni specifiche sui materiali.obj struttura:file mtl, coordinate vertici, coordinate texture, vettori normali, nome oggetto, nome materiale, facce con con**: f vertice/texture/normale.** Il vettore della normale può essere usato da più vertici. **WRL:**formato per realtà virtuale, consente lo scripting dei comportamenti (animazioni), formato testuale. **FBX:** formato binario e ascci, propietario autodesk, molto usato con directx. **3DS:** formato binario basato su chunk di dati, supporta luci, mesh, materiali, animazioni e grafico di scena. **OVO:** formato custom OverVision engine; supporta materiali, texture, mesh, luci.., supporta il mesh bounding. Formato binario basato anch’esso su chunk.