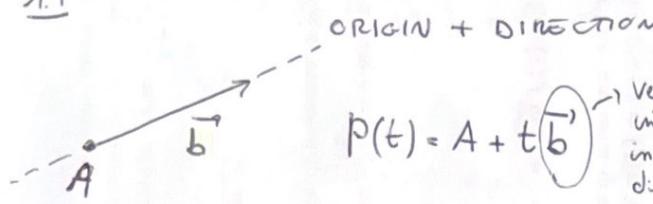


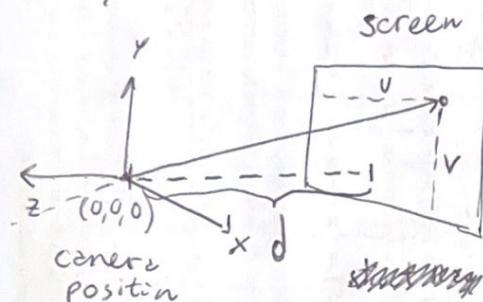
# RAY TRACING

## 1. Ray Class

1.1



## 1.2 Ray in Scene



inviano dei raggi dalla posizione della camera per tutti i pixel della view per andare a computerare il colore associato a ogni pixel

1. manda il ray
2. trova cosa interseca il ray
3. trova il colore associato al ray

per ogni punto dello screen dell'uno invia un rayo partendo dalla posizione della camera (eye)

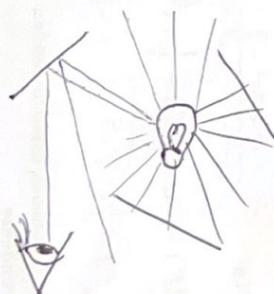
Ci servono

- pos camera
- distanza delle view
- UV coordinates
- dimensioni delle view

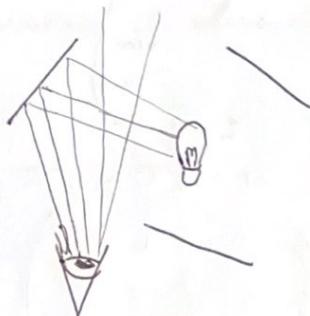
} devono passare tutte queste info alle shader

(QUI PROBABILMENTE DOVREMO FARCI  
QUALCHE CICLO PER CAPIRE BENE  
TUTTE LE INFO ASSOCIATE ALL'PROJECTION  
VIEW)

Perciò ci servono image che riportano quella nostra vista!



se faccio partire i raggi dalla fonte, i raggi che raggiungono gli occhi vengono "speciati"



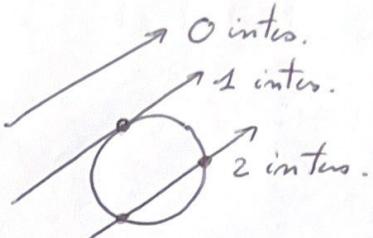
metto ferri partire dal nostro occhio dato che c'è solo quello che possiamo osservare

## 2 Sfere

2.1



abbiamo ora delle sfere nello scene e vogliamo sapere se i nostri raggi vedono queste sfere



$$\begin{aligned} a &= 1 \\ b &= 2D \cdot (O - C) \\ c &= |O - C|^2 - r^2 \end{aligned}$$

$$\begin{cases} P(t) = A + t\vec{b} & \text{eq. ray} \\ x^2 + y^2 + z^2 = r^2 & \text{eq. sphere} \end{cases}$$

$$|P(t) - C|^2 = r^2 \quad \text{equazione analitica inteseive raggi sfera} \Rightarrow t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2.2 non ho quindi aggiunto una funzione dello shader

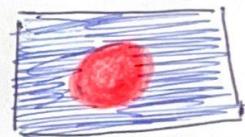
hit-sphere (vec3 center, double radius, Ray ray) of y

Verifica se il raggio interseca la sfera

(ovviamente la sfera la creiamo nel main e passiamo le info legate al centro e raggio con in/out)

Penso che creiamo tante sfere e le passiamo con un array)

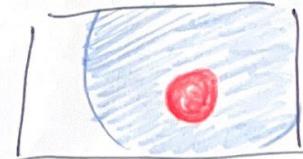
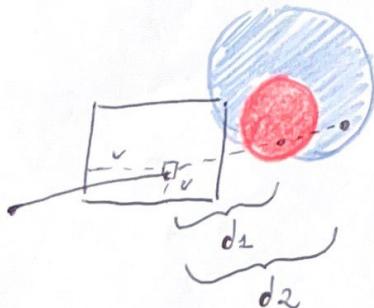
quindi posso dare due cose  
hit-sphere  $\xrightarrow{\text{true}}$  return color. RED  
 $\xrightarrow{\text{false}}$  return color. BLUE



2.3 se ora aggiungo più sfere come col proprio colore

struct Sphere {  
 vec3 centre  
 float radius  
 vec4 color  
}

le creano tutte nel main e le posso nelle sfere, se posso vedere per ogni raggio quale sfera viene colpita e tieni conto delle distanze più bruci e rende il colore della sfera corrispondente

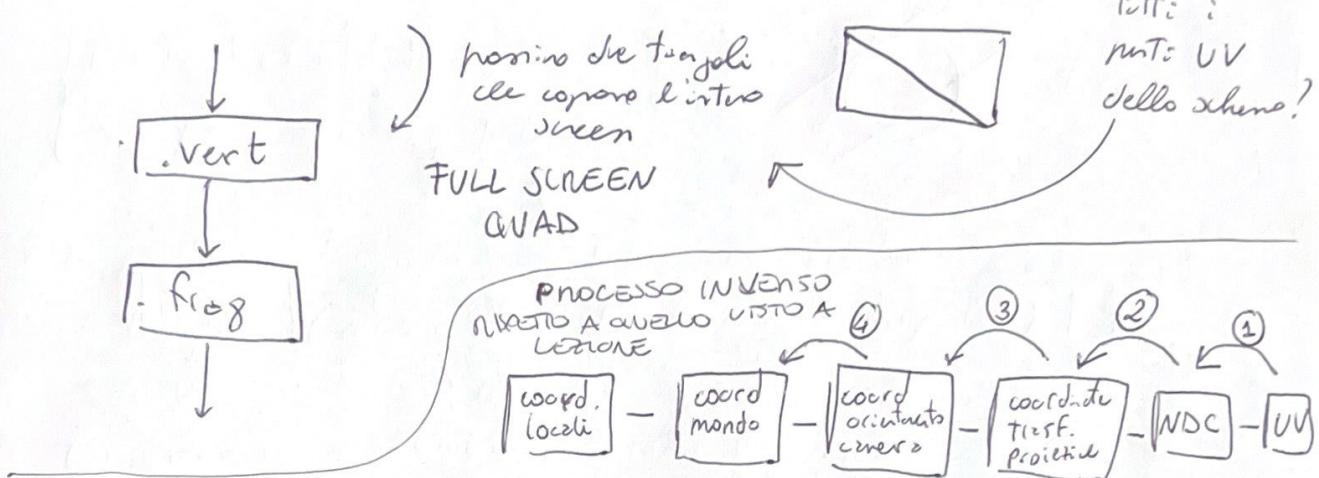


raggio colpisce entrambe, le  $d_1 < d_2$   
quindi renderizzo rosso

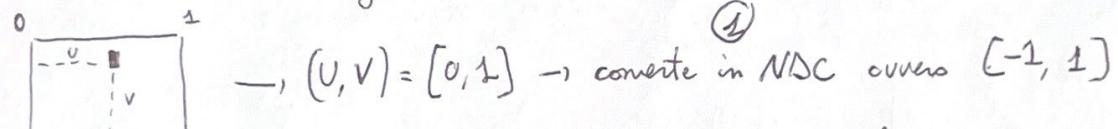
DUE CONCERN SU VULKAN

Tutto quello visto ad ora si fa sulla frog Shred, però il punto è che  
Vulkan non si fa di me. Veri shredder quindi chi sei?

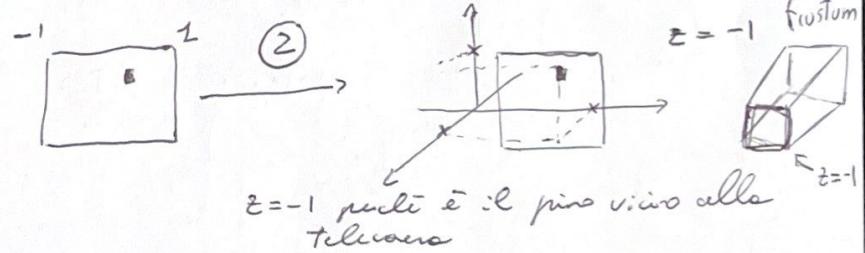
vert he bisigno d: we mesh ma ch mesh posso se ia devo otherwise



cosa fe lo shader fizy?



cra punto nello spazio

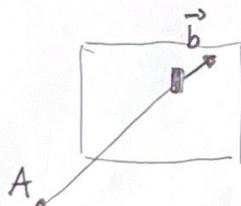


Trofano nuove condizioni rispetto alla vista

\* inverso delle ~~proj~~ Matrix e le riconstruisce  $\frac{x}{w} \frac{y}{w} \frac{z}{w}$

④ cond: este rondo

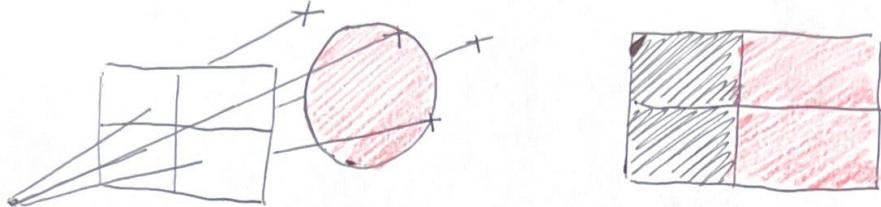
\* inverse view Matrix e non nullizzo per ottenere vettori unitari



FINE V1 PROGETTO

### 3. Aliasing

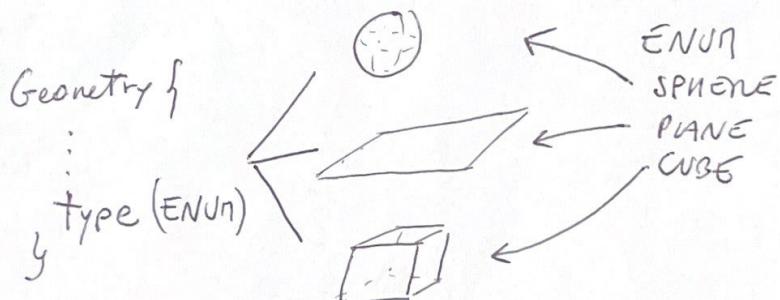
nel documento si introduce anche il problema legato all'point sampling, ovvero stiamo compionando il nostro schermo rispetto al n° di raggi che generiamo e quindi la risoluzione si allarga al n° di raggi.



Noi però abbiamo un raggio per ogni pixel dello schermo quindi non abbiamo problemi. Se volessimo una risoluzione migliore, allora dovremo integrare i risultati: uttrosa a due volte la luce per ottenere una risoluzione migliore.

### STEP SUCCESSIVO

rendere il tutto genérico e non solo con Sfere

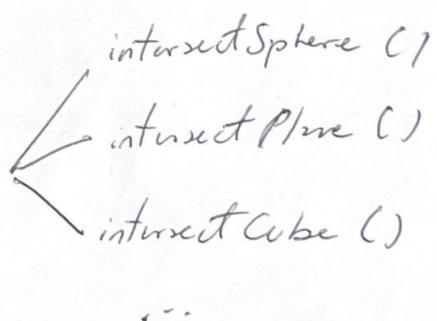


uso un array di Geometry[n]

verifico se il raggio interseca le geometrie

verifico il tipo delle geometrie e sceglio il metodo di calcolo dell'intersezione

(geometria che interseca un raggio con un piano o con un cubo sono sole due, quindi saranno gestiti in modi differenti)



intersezione piano

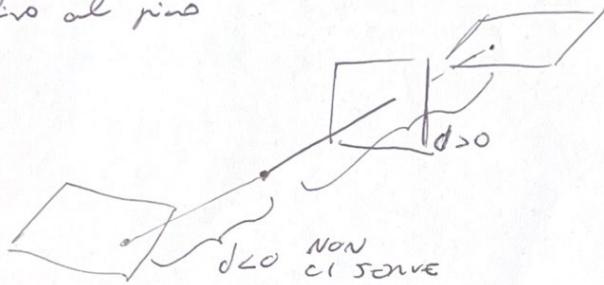
$$\vec{N} \cdot (\vec{P} - \vec{P}_0) = 0 \text{ formula piano} \quad \text{raggio } \vec{R}(t) = \vec{O} + \vec{t} \vec{D}$$

1) verifica che non sia // al piano se  $\vec{N}$  e  $\vec{t}$  sono //  
 $\text{dot}(\vec{N}, \vec{t}) ? = 0$

2) sostituisce all'equazione del piano l'angolo del raggio  $\theta$

$$\vec{N} \cdot (\vec{P} - \vec{O}) = x \\ \text{ottiene un raggi che va da } \vec{O} \text{ a } \vec{P}$$

3) verifica che sia > 0 la distanza lungo la direzione del raggio  
fra al piano



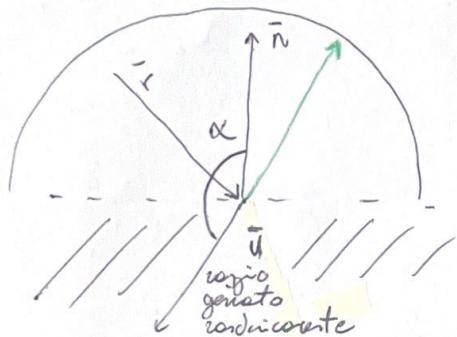
#### 4. Diffuse Reflections materiali con superfici non perfettamente lisce

Diffuse objects non emettono luce, ne bensì riflettono e recuperano il colore degli oggetti circostanti.

Quello che fare è riflettere i raggi che arrivano.

Il più superficie esempio di diffuse refrazione è quello di una sfera con spigoli parallelati in giù dirette

Ora come si generano i raggi connessi all'interno dell'intera sfera, dato che i raggi non si intersecano?



superficie abbinano:

$$\text{dot}(\bar{n}, \bar{o})$$

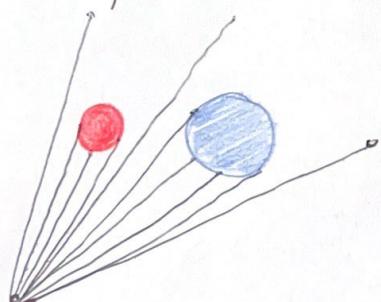
se  $\text{dot} < 0$  allora l'angolo  $\alpha$  tra  $\bar{n}$  e  $\bar{o}$  è  $> 90^\circ$ , e quindi significa che il raggio è uno di quelli rossi di riva

→ in questo caso invertendo il raggio e ottieniamo quello conetto

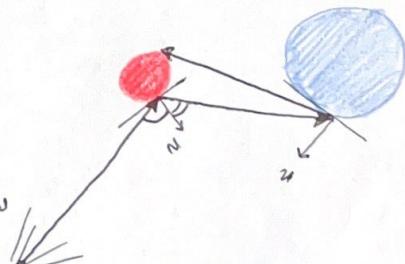
quello che bisogna fare ora è un piano ricorsivo.

Fino ad ora infatti ci sono limitati a ricevere i raggi dalla sfera e se battava qualcosa perdeva il colore dell'oggetto colpito.

Ora invece già volta le colpisce su qualcosa vado a riconoscere quale raggio con direzione corrisponde e rendere ricorsivo



questo  
solo se  
in raggio,  
ripetere su  
tutti i  
raggi

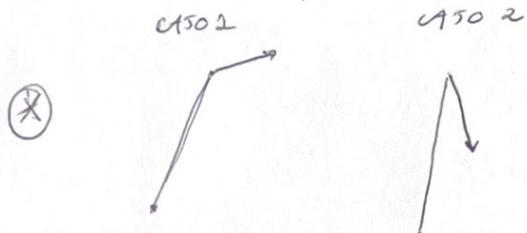


Iniziamo per ora a dire che

- se colpisce qualcosa ritorna  $\frac{1}{2}$  colore delle diverse ricerche
- se non colpisce nulla il colore del cielo

## APPUNTI SULLA SHADON

- per generare un valore randomico partendo da un seed usare i PCG [www.pcg-random.org](http://www.pcg-random.org)
- inoltre negli shader lavorare con una distibuzione uniforme per estrarre un valore. Altro: quello che accade è che



appena spostate le telecamere il nuovo  
raggio potrebbe avere diverse totalmente  
diverse, negli invece segue una  
distibuzione Normale

con seed genero un qualcosa a parte dalle coordinate dei pixel che non  
sono dirette loro

\* Le SHADON NON SUPPORTANO LA RICORSIONE, QUINDI SE' DA FARLE CON  
UN CICLO

inoltre il ciclo torna solo quando il raggio non colpisce più qualcosa,  
ma è troppo  $\rightarrow$  mettiamo MAX\_DEPTH = 10

\* la soluzioe poi segue anche la lambertian distribution, ovvero che un  
raggio rifleso da un materiale diffuso è proporzionale al  $\cos(\phi)$ , ovvero il  
cos dell'angolo tra  $\vec{n}$  e raggio rifleso.

TRADOTTO: "E' più probabile che il raggio sia rifleso vicino alla normale,  
piuttosto che lontano"

ver.direction = hit.normal + random.Direction()

## §.1 Light

ora se esegui il codice ottieni una cosa come



Ora vogliamo introdurre una forte luce, di modo tale

che ottieni effetti di gesto tinto  
effetto di ombre sugli oggetti che non  
sono fonte di luce.



struct RayTracingMaterial {

color → se riflette luce != nero altrimenti se fonte di luce → nero

emissionColor → binario se fonte di luce

straight

}

quindi se nel vedi dei raggi avrai 2 variabili

color = 1

incomingLight = 0

for - - - material material

incomingL += (KemissinColor \* Klight) \* color

color += material.color

di luce noi moltiplichiamo il colore + il colore del reticolo da l'ha con il Ray.

In più teniamo ditta luce che viene in moto

Considera un solo raggi

↓ binario = fonte

- se incontra luce incoming += (1.0 \* 1.0) \* 1.0 → colore binario

- se incontra oggetto incoming += (0.0 \* 1.0) + 1.0 → nero ombre  
di oggetto

## ACCONCINENTI WCÓ

-1 quando ho aperto la luce mi sono accorto che c'era una sorta di pattern nei punti di variazioni illusori sul pavimento, come delle sfere concentriche

CENTRIFIC ANTIFACTINO

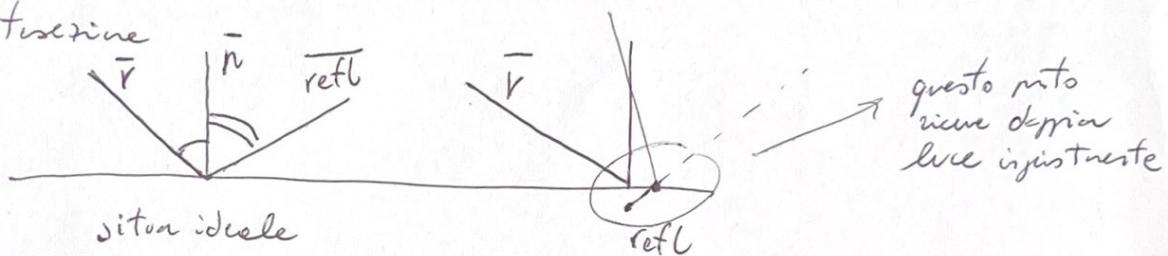
o

SOF INTERSECTION PROBLEM



poteva non leggere alla jettazione dei neri cosenali, dato che anche solo un po' di sovrapposizione si vede bene come la jettazione di neri cosenali sia giusta.

Il problema è legato al 2° raggiungimento. Se effettua il suo moto attraverso un punto = intersection point, per problemi di anastoreto, la stessa superficie potrebbe essere rilettata una seconda volta, generando quindi un falso punto J: intersection



avvenuto quindi in offset al punto J: partenza

FINE VERSI ONE 2

## 5. Progressive Rendering

ovviamente la nostra immagine è sfocata perché invia un solo ray per pixel  
 Se incrementa il numero di ray per pixel inviati a riflettere la visualizzazione, ne pagherà tanti in funzione di FPS, con 100 ray per pixel l'immagine è Faccetta sullo schermo.

### 5a

Se stiamo lavorando con un rendering, per cui la sorgente di luce, una soluzione è il Progressive rendering.

Invece di inviare 100 ray per pixel a gli fuori, se invia solo 1, ma ritraso il colore del vecchio frame per aggiornarlo di volta con il valore nuovo, dunque tale da migliorare non a non de peggiorare il tempo l'immagine

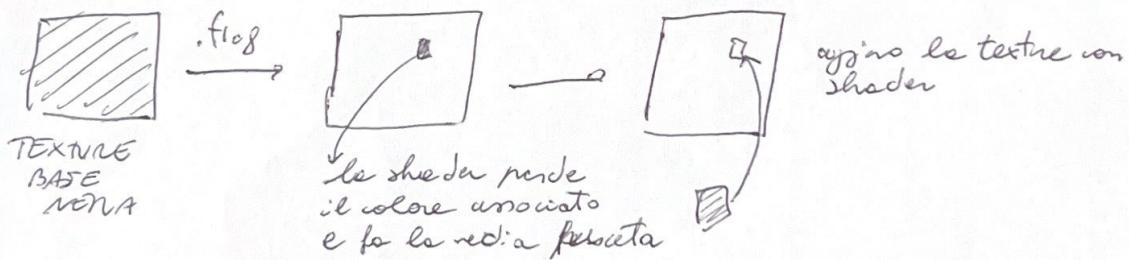
PROBLEMA VULKAN: non esistono delle variabili globali tra i shaders, quindi dovranno passare alle shaders una cosa come

struct f

{ vec4 color // align = 16 }

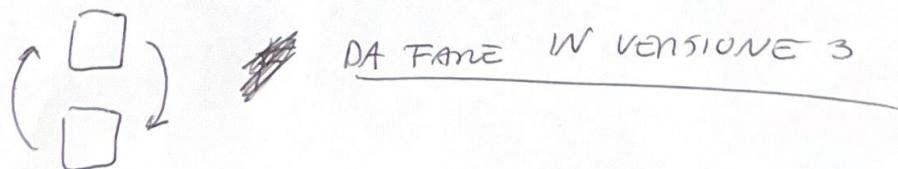
e poi creare un buffer con prendere width \* height \* sizeof() = 7,68 Mbytes  
 troppo grande e troppo costoso

=> PASSARE TUTTO SU UNA TEXTURE



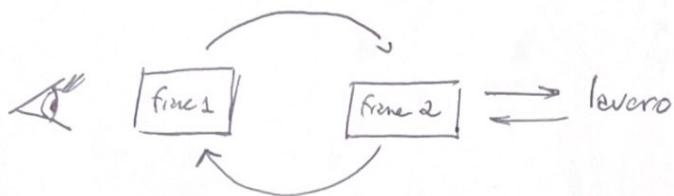
PROBLEMA: Ultimo step non si può fare con uno shader, deve in Frame Buffer

dobbiamo trovare in modo di lavorare con due texture da switchare (rig nowz)



## 5.6 Progressive Rendering con Texture

L'idea è sfruttare le swap chain vista a lezione



Se voglio ad esempio ordine a fare fade ~~the~~ nero a bianco

a. Mostro immagine A  $(0, 0, 0)$

b. nel prossimo aggiorno l'immagine B  $(1, 1, 1)$

c. Swapriamo e così via

quello che faccio in b è  $A + (1, 1, 1)$

Command

nel creare `IVBuffer`, prima di andare a iniziare il rendering pass (prima facili se se poi Vulkan vuole me sotto istanze del subspace)

in metode `copyImage` ha

VKCmdPipeline  
Barrier

1) usa le barrier per la sincronizzazione per layout dst e src V

2) fa la copia dell'immagine con `VKCmdCopyImage()`

3) ancora due barrier

PRACTICAMENTE E' DA SUDE 65 DI L21

1) le barrier servono perché l'elenco ordine a trasmettere le `VkImage` da un layout a un altro specifico

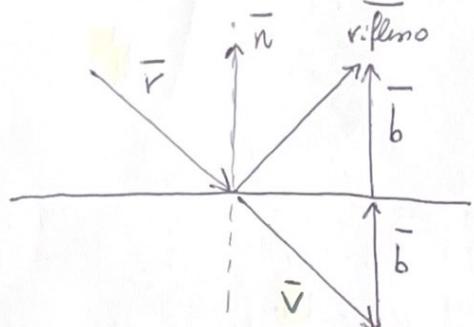
`VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL` = immagine per color attachment

`PRESENT_KHR` = da presentare sullo swapchain

`TRANSITION_DST_OPTIMAL` = immagine destinata per una `copy` operation

## 6. Reti di Specchi:

Implementano gli specchi, in questo caso il raggio non ha più una direttrice randomica quando colpisce la superficie

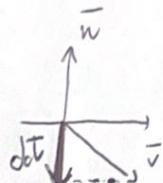


con che il rifleso?

$$\text{rifleso} = \bar{v} + 2\bar{b}$$

$\bar{v} = \bar{v}$  ovviamente

poi  $\text{dot}(\bar{v}, \bar{n})$



mettiamo davanti a invertire il segno, e lo

risultato  $\times n$

$$\text{rifleso} = \bar{v} - 2 \times \text{dot}(v, n) \times n = \bar{b}$$

ora insinno un nuovo valore nel RayTracing Network smooth = [0, 1]

→ 0 significa che il reticolo non si comporta da specchio

→ 1 che è totalmente un specchio

## VULKAN

quando calcolo la nuova direttrice del raggio, devo a calcolare quelle su il diffuse Network (la randomica) ~~sempre~~ ma anche quelle su gli specchi e faccio

lerp rispetto allo smooth tra esse  
(interpolazione)

in GLSL è  $\text{mix}(a, b, c)$

$$c=0 \rightarrow a$$

$c=0.5$  metà tra a e b

$$c=1 \rightarrow b$$

## 7. Riferimenti Dielettrici

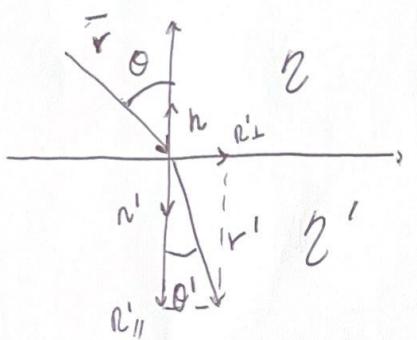
acqua, vetro... rettuali che riflette la luce ne la rifugge o anche la quantità di luce che viene rifatta dipende dall'indice di rifrazione

VETRO  $i = 1.5 - 1.7$ , DIANANTE  $i = 2.4 \dots$

se poi hai un dielettrico dentro un dielettrico (es. nella vetro in acqua) il nuovo indice sarà  $\frac{\text{indice rettuale}}{\text{indice del contenitore}}$

### Rifrazione

Snell Law



$$n \cdot \sin \theta = n' \cdot \sin \theta'$$

$$R'_H = -\sqrt{1 - |R'_L|^2 n}$$

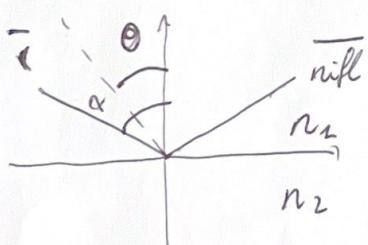
$$\bar{R}'_L = \frac{n}{n'} (\bar{R} + \cos \theta \bar{n})$$

valore

questo sarà la componente  $\perp$  di  $\bar{R}_L$  è  $\bar{R}_L = (\bar{R}' \cdot \bar{n}) \bar{n}$  e noi dev'abbagliarci per il rapporto tra gli indici  $\frac{n}{n'}$

$$\bar{R}_H = \bar{R}' - (\bar{R}' \cdot \bar{n}) \frac{\bar{n}}{\bar{n}'}$$

ricordiamoci pure che ~~se~~ c'è l'angolo critico, ovvero



in questo caso non c'è rifrazione, solo riflessione

$$\theta = \arcsin \left( \frac{n_2}{n_1} \right) \text{ solo se } n_2 < n_1$$

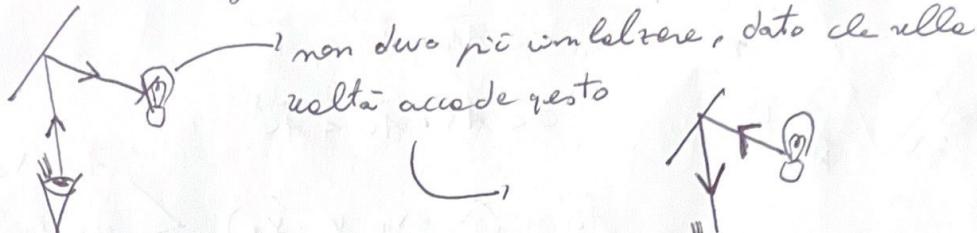
$\alpha > \theta$  più solo riflessione

RIFLESSIONE TOTALE SOLO SE  $n_2 > n_1$

in VULKAN

ma valta trovato il rettangolo del colpo h

1) se  $h.\text{emissionStrength} > 0.0$  è una luce e non sono issue dal vhl

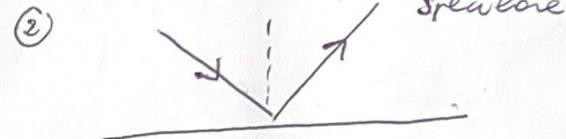
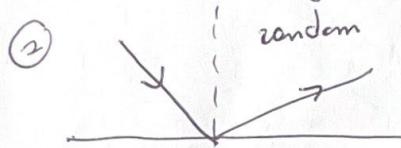


IN QUESTO CASO UCCOCO IL COLORE

2) se  $R.\text{dielectric Constant} < 1$

NON È UN DIELETTRICO  $\rightarrow$  NO LEGGE SNELL

dovrò decidere se il raggi si flessa forte



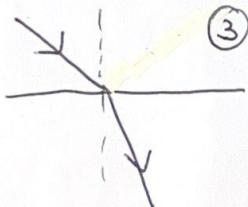
sulle cose delle smoothness del materiale

- più vicino a 1 più ②
- più vicino a 0 più ①

3) se ho DIELETTRICO

a) verifio se sono nel caso di total reflection, ovvero se  
 $n_1 > n_2$  e  $\theta_i > \theta_{\text{crit}}$  SE VERSO ACCORDI ②

b) se nono avere riflessione allora devo rendere visibile se applicare riflessione o rifrazione sulle cose delle smoothness

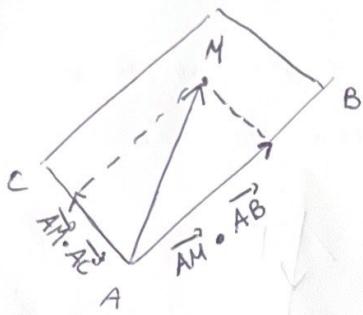


2-3 mi servono la nuova direzione del raggio

pos del raggio = hit\_pos + ray\_dir \* 0,002 per evitare il self intersect.

moltiplico il colore globale del pixel + colore del rettangolo

## Interscruz piano limitato



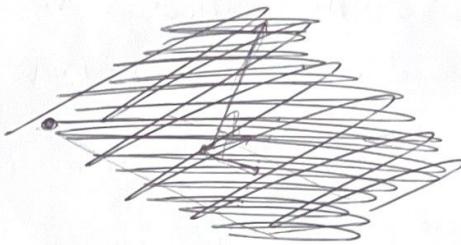
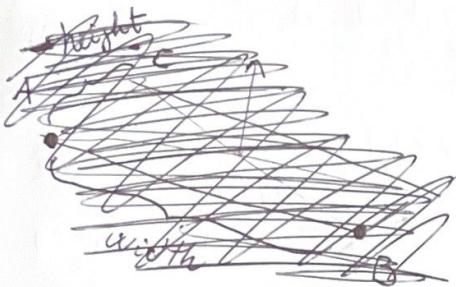
dove avere che

$$0 < \vec{AM} \cdot \vec{AB} < \vec{AB}$$

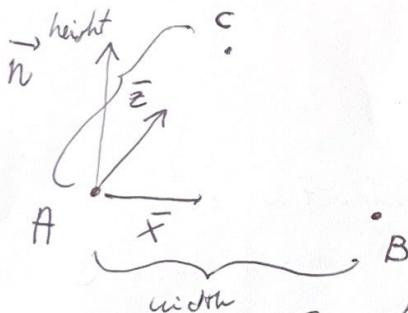
$$0 < \vec{AN} \cdot \vec{AC} < \vec{AC}$$

$$\vec{AB} = (x_2 - x_1, y_2 - y_1) \quad \begin{cases} A = (x_1, y_1) \\ B = (x_2, y_2) \end{cases}$$

ora il problema è trovare le coordinate del punto B



ho cercato per passare point, normale e gli alti. Se ass:



di modo che per trovare B e C basta fare

$$B = A + \bar{x} \cdot \text{width}$$

$$C = A + \bar{z} \cdot \text{height}$$

