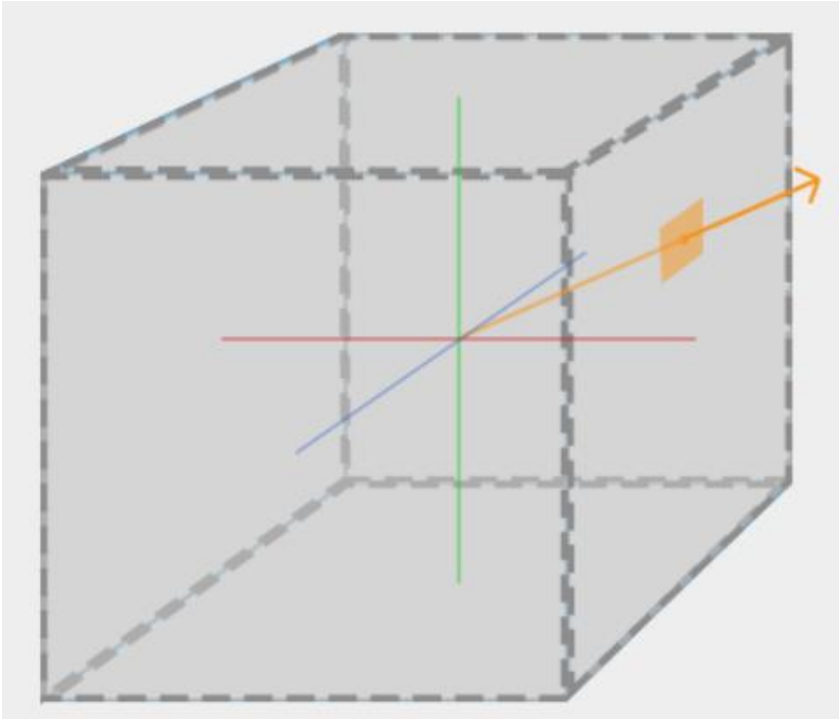
Cubemap

Una cubemap è una texture che contiene 6 singole texture 2D che formano ciascuna un lato di un cubo: un cubo strutturato.

Le cubemap hanno la proprietà utile di poter essere campionate usando un vettore di direzione. Supponiamo di avere un cubo unitario 1x1x1 ed un vettore direzione che ha l'origine nel centro del cubo.

Il campionamento di un valore di texture dalla cubemap con un vettore di direzione, avviene nel seguente modo:



Se immaginiamo di avere una cubo a cui applichiamo una tale cubemap, questo vettore di direzione è uguale alla posizione (interpolata) del vertice locale del cubo.

In questo modo possiamo campionare la cubemap usando i vettori di posizione effettiva del cubo, purché il cubo sia centrato sull'origine.

Quando campioniamo una cubemap, consideriamo quindi tutte le posizioni dei vertici del cubo come coordinate di texture.

Creare una cubemap

- Una cubemap è una texture come qualsiasi altra, quindi per crearne una generiamo una texture e la leghiamo al target GL_TEXTURE_CUBE_MAP prima di eseguire ulteriori operazioni sulla texture.

- **unsigned int** textureID;
- glGenTextures(1, &textureID);
- glBindTexture(GL_TEXTURE_CUBE_MAP, textureID);

Poiché una cubemap contiene 6 texture, una per ogni faccia, dobbiamo chiamare glTexImage2D sei volte impostandone i parametri.

Questa volta, tuttavia, dobbiamo impostare il parametro *target* texture in modo che corrisponda a una faccia specifica della mappa cubica, dicendo a OpenGL per quale parte della mappa cubica stiamo creando una trama. Questo significa che dobbiamo chiamare glTexImage2D una volta per ogni faccia della mappa cubica

Poiché abbiamo 6 facce, OpenGL fornisce 6 differenti target, per ogni faccia del cubemap

Texture target	Orientation
GL_TEXTURE_CUBE_MAP_POSITIVE_X	Right
GL_TEXTURE_CUBE_MAP_NEGATIVE_X	Left
GL_TEXTURE_CUBE_MAP_POSITIVE_Y	Top
GL_TEXTURE_CUBE_MAP_NEGATIVE_Y	Bottom
GL_TEXTURE_CUBE_MAP_POSITIVE_Z	Back
GL_TEXTURE_CUBE_MAP_NEGATIVE_Z	Front

potremmo eseguirne il ciclo iniziando con GL_TEXTURE_CUBE_MAP_POSITIVE_X e incrementando l'enum di 1 ogni iterazione, su ogni target della texture:

```
vector<string> faces
{
"right.jpg",
"left.jpg",
"top.jpg",
"bottom.jpg",
"front.jpg",
"back.jpg"
};
```

```
int width, height, nrChannels;
unsigned char *data;
for(GLuint i = 0; i < textures_faces.size(); i++)
 {
data = stbi_load(faces[i].c_str(),  &width, &height, &nrChannels, 0);

 glTexImage2D( GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0, GL_RGB, width, height, 0,
GL_RGB, GL_UNSIGNED_BYTE, data );

}
```

Specificare wrapping e filtering

```
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);
```

GL_TEXTURE_WRAP_R imposta il metodo di wrapping per la coordinata R della texture che corrisponde alla terza dimensione della texture.

Impostiamo il metodo di wrapping su GL_CLAMP_TO_EDGE poiché le coordinate di texture che sono esattamente tra due facce potrebbero non colpire una faccia esatta (a causa di alcune limitazioni hardware), quindi usando GL_CLAMP_TO_EDGE OpenGL restituisce sempre i loro valori di bordo ogni volta che campioniamo tra le facce.

All'interno del fragment shader dobbiamo anche usare un diverso campionatore del tipo samplerCube, ma questa volta usando un vettore di direzione vec3 invece di un vec2. :

Esempio di fragment shader che usa una cubemap

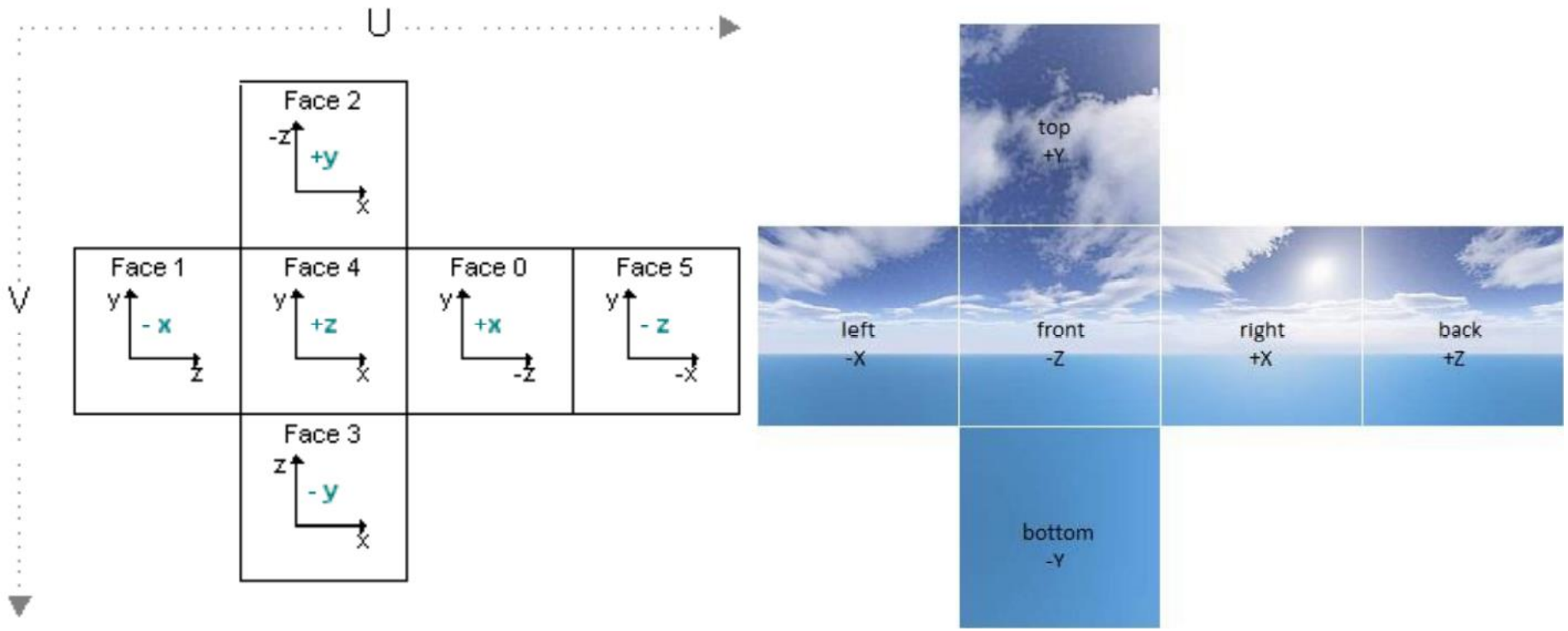
```
in vec3 textureDir; // vettore direzione che rappresenta una coordinata 3D di texure
uniform samplerCube cubemap; // campionatore di texture di tipo cubemap

void main()
{
    FragColor = texture(cubemap, textureDir);
}
```

Uso di una cubemap per creare uno skybox

- Uno skybox è un cubo (grande) che racchiude l'intera scena e contiene 6 immagini di un ambiente circostante, dando al giocatore l'illusione che l'ambiente in cui si trova sia in una realtà molto più grande di quella reale.

- Alcuni esempi di skybox utilizzati nei videogiochi sono immagini di montagne, di nuvole o di un cielo notturno stellato.

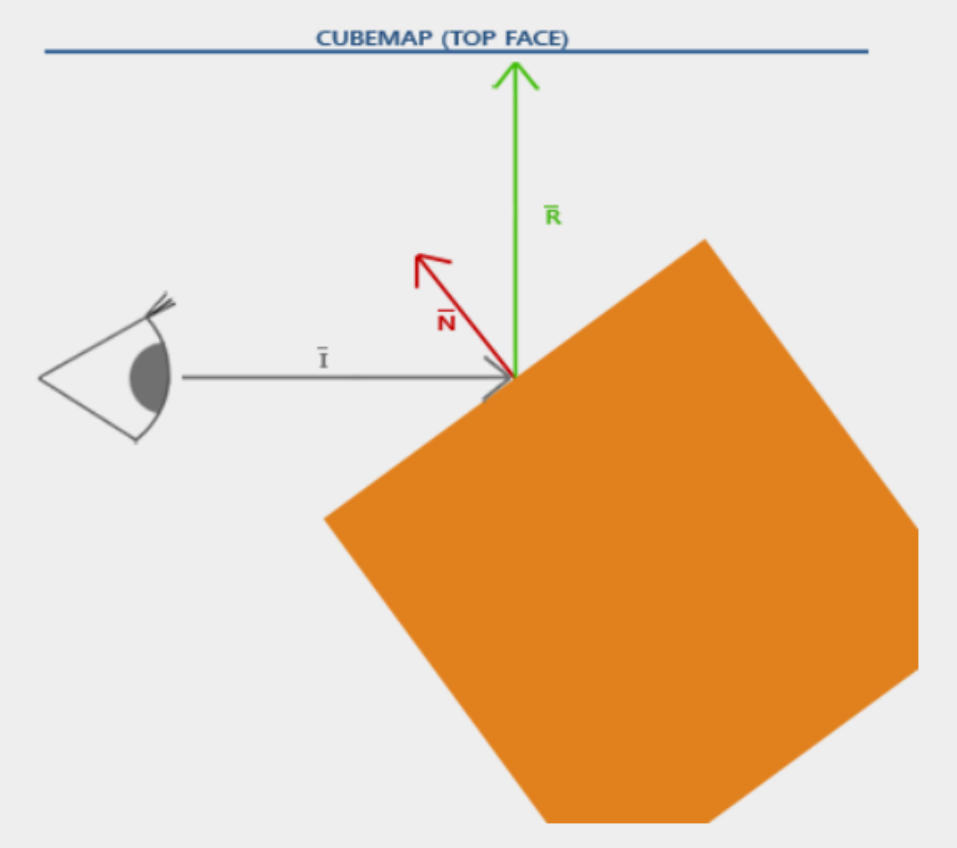

U
V
Face 2
-z
+y
x
Face 1
y
- x
z
Face 4
y
+z
x
Face 0
y
+x
-z
Face 5
y
- z
-x
Face 3
z
- y
x
top
+Y
left
-X
front
-Z
right
+X
back
+Z
bottom
-Y


top
left
front
right
back
bottom

Mappatura ambientale

La riflessione è la proprietà di un oggetto (o parte di un oggetto) che riflette l'ambiente circostante ad es. i colori dell'oggetto sono più o meno uguali al suo ambiente in base all'angolo del visualizzatore.

Dato un ambiente cubico, in cui è stata mappata una cubemap, vediamo adesso come campionare la cubemap da associare ad un oggetto, in maniera tale che rifletta l'ambiente c e:

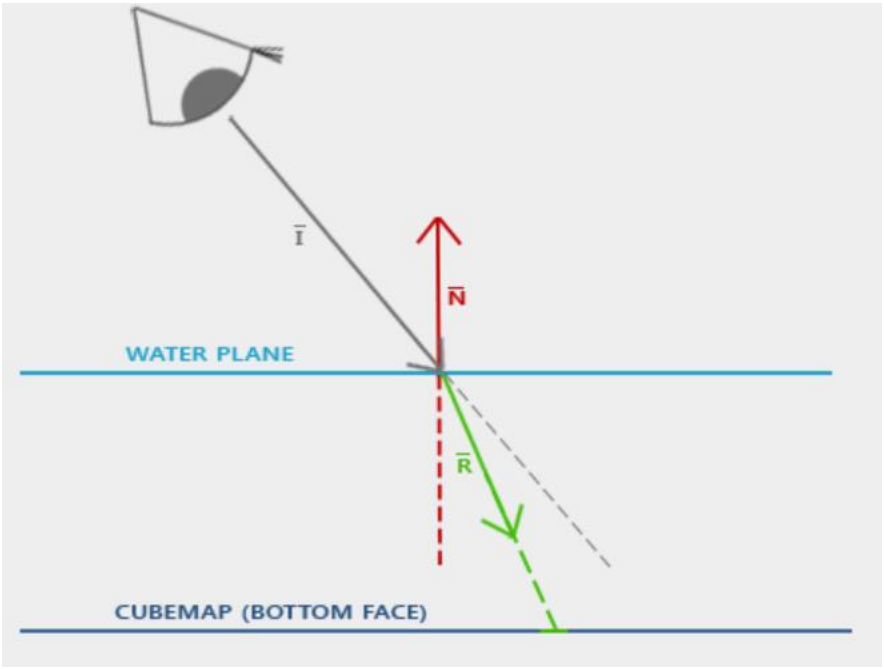


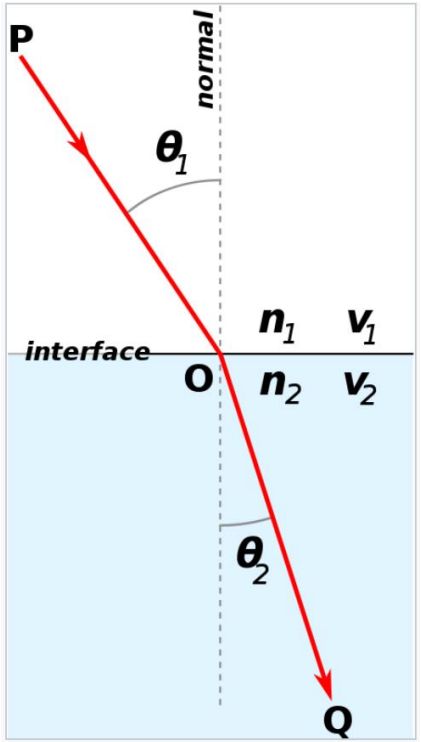

```
I = normalize(Position - cameraPos);
R = reflect(I, normalize(Normal));
 //R è la direzione lungo cui campionare la cubemap
FragColor = vec4(texture(skybox, R).rgb, 1.0);
```

Questo codice va scritto nel fragment shader relativo all'oggetto che si vuole rifletta la cubemap dell'ambiente circostante.

Un'altra forma di mappatura dell'ambiente è chiamata **rifrazione** ed è simile alla riflessione.

La rifrazione è il cambiamento di direzione della luce dovuto al cambiamento del materiale attraverso il quale scorre la luce. La rifrazione è ciò che vediamo comunemente con superfici simili all'acqua in cui la luce non penetra direttamente, ma si piega leggermente.





Legge di Snell

$$\frac{\sin\theta_2}{\sin\theta_1} = \frac{v_2}{v_1} = \frac{n_1}{n_2}$$

La rifrazione è abbastanza facile da implementare utilizzando la funzione refract incorporata di GLSL che prevede un vettore normale, una direzione di vista e un rapporto tra gli indici di rifrazione di entrambi i materiali.

L'indice di rifrazione determina quanto si distorce la luce passando da un materiale ad un altro. Ogni materiale ha il proprio indice di rifrazione. Nella tabella seguente è riportato un elenco degli indici di rifrazione più comuni:

Materiale	Indice di rifrazione
Aria	1.00
Acqua	1.33
Ghiaccio	1.309
Vetro	1.52
Diamante	2.42

Nel fragment shader

```
void main()
{
    float ratio = 1.00 / 1.52;
    vec3 I = normalize(Position - cameraPos);
    vec3 R = refract(I, normalize(Normal), ratio);
    FragColor = vec4(texture(skybox, R).rgb, 1.0);
}
```