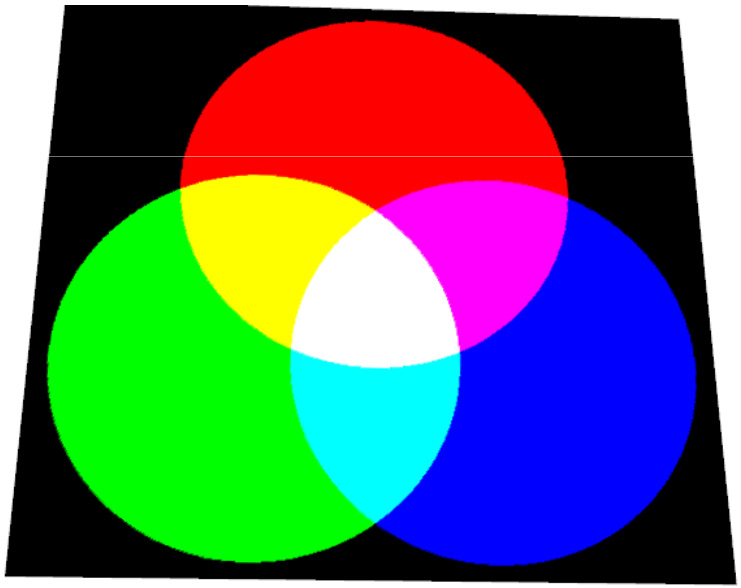
Gestion des couleurs et des matériaux

La chaine de traitement programmable

- Génération des facettes à partir de points
- Lissage par le shader: application d'un traitement spécifique à chaque sommet
- Génération de l'image sur l'écran (pixelisation)
- Lissage de fragment. Test de visibilité
- Calcul de la couleur pour la surface en question

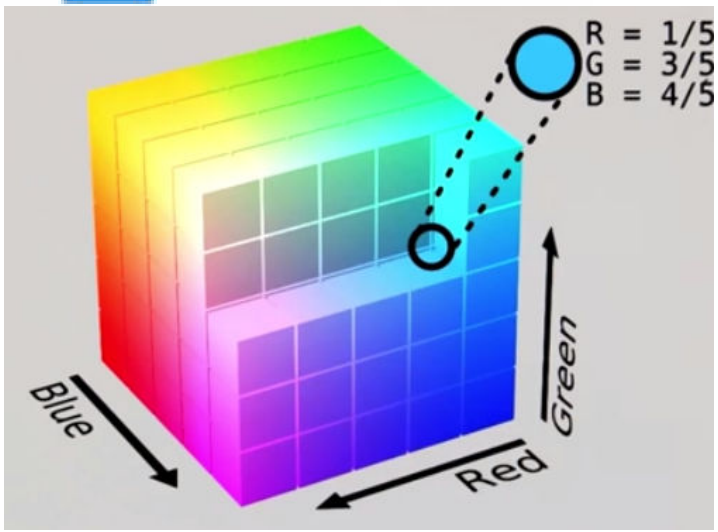
Rappel sur la représentation des couleurs

- CMYK vs RGB
- Démo



Couleur sous THREE.js

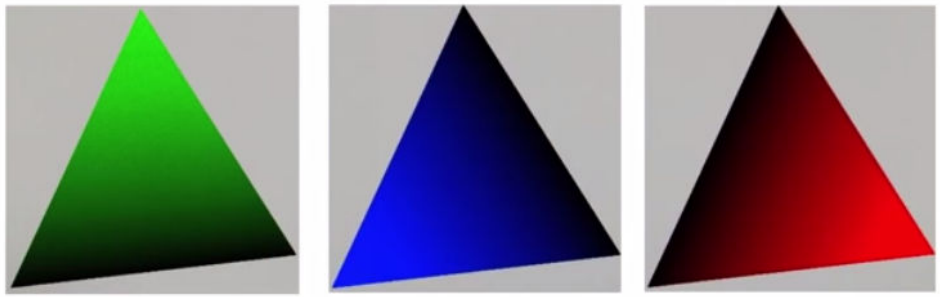
```
var sphereMaterial = new THREE.MeshLambertMaterial();

sphereMaterial.color.r = 1.0;
sphereMaterial.color.g = 0.0;
sphereMaterial.color.b = 0.0;


sphereMaterial.color.setRGB(0.972, 0.749, 0.141);

sphereMaterial.color.setHex(0x1280FF);
```

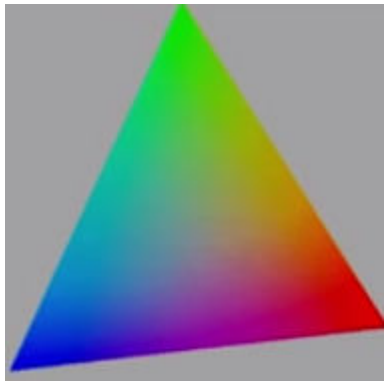


Exercice d'interpolation couleur

- Définition d'un triangle couleur au coins
  - P1 Rouge
  - P2 Bleu
  - P3 Vert

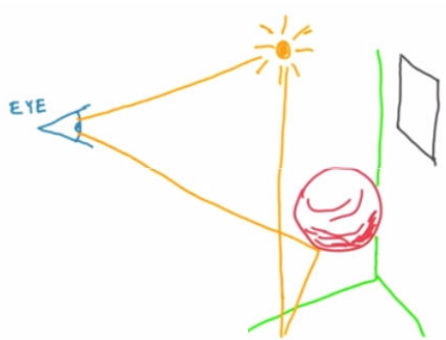
```
var color1 = new THREE.Color( 0xF08000 ); // orange
var color2 = new THREE.Color( 0x808000 ); // olive
var color3 = new THREE.Color( 0x0982FF ); // bright blue

geometry.faces[0].vertexColors = [ color1, color2, color3 ];
```





Modèle d'illumination

- Facteurs de définition d'un matériau:
  - Émissive
  - Ambiante
  - Diffuse
  - spéculaire



SURFACE COLOR =

EMISSIVE +

AMBIENT +

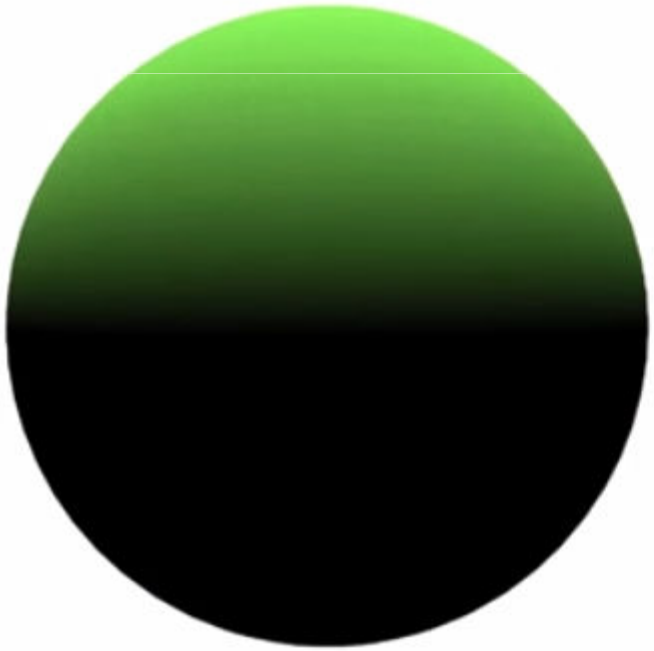
FOR EACH LIGHT:

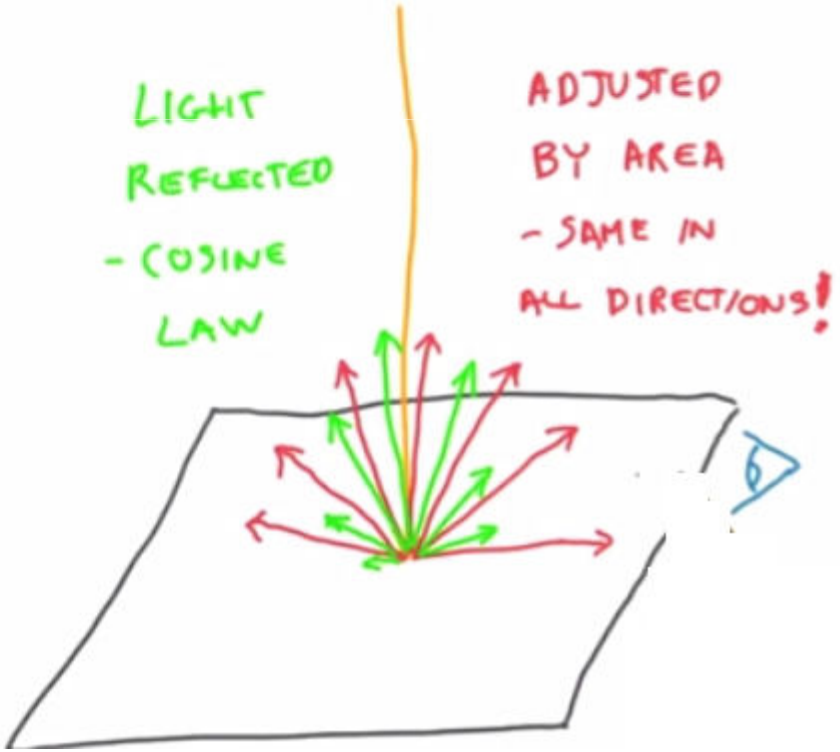
DIFFUSE(LIGHT) + SPECULAR(LIGHT, VIEW)

$$C = E + A + \sum (D(L) + S(L,V))$$

Lumière sur un matériau diffus

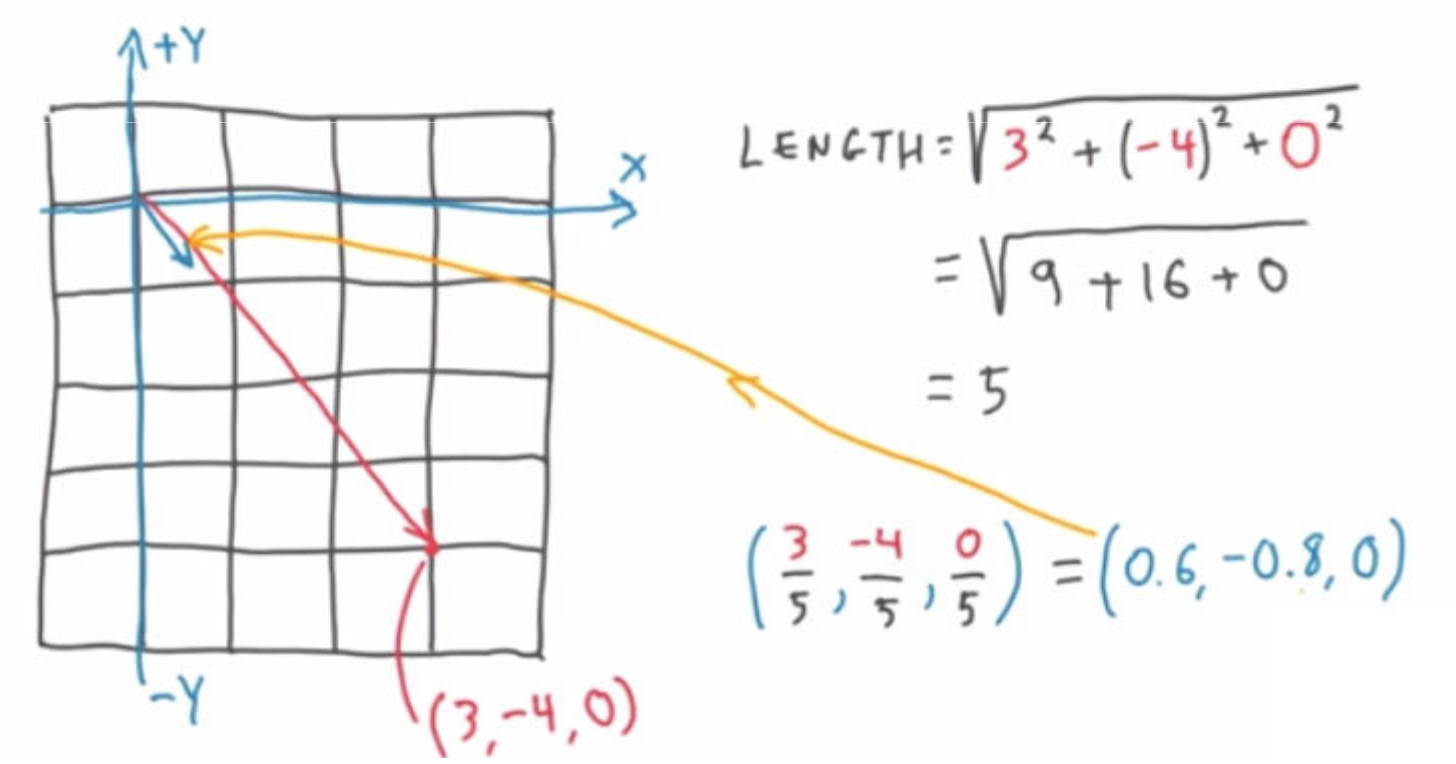
- Le cosinus est obtenu rapidement par le produit scalaire



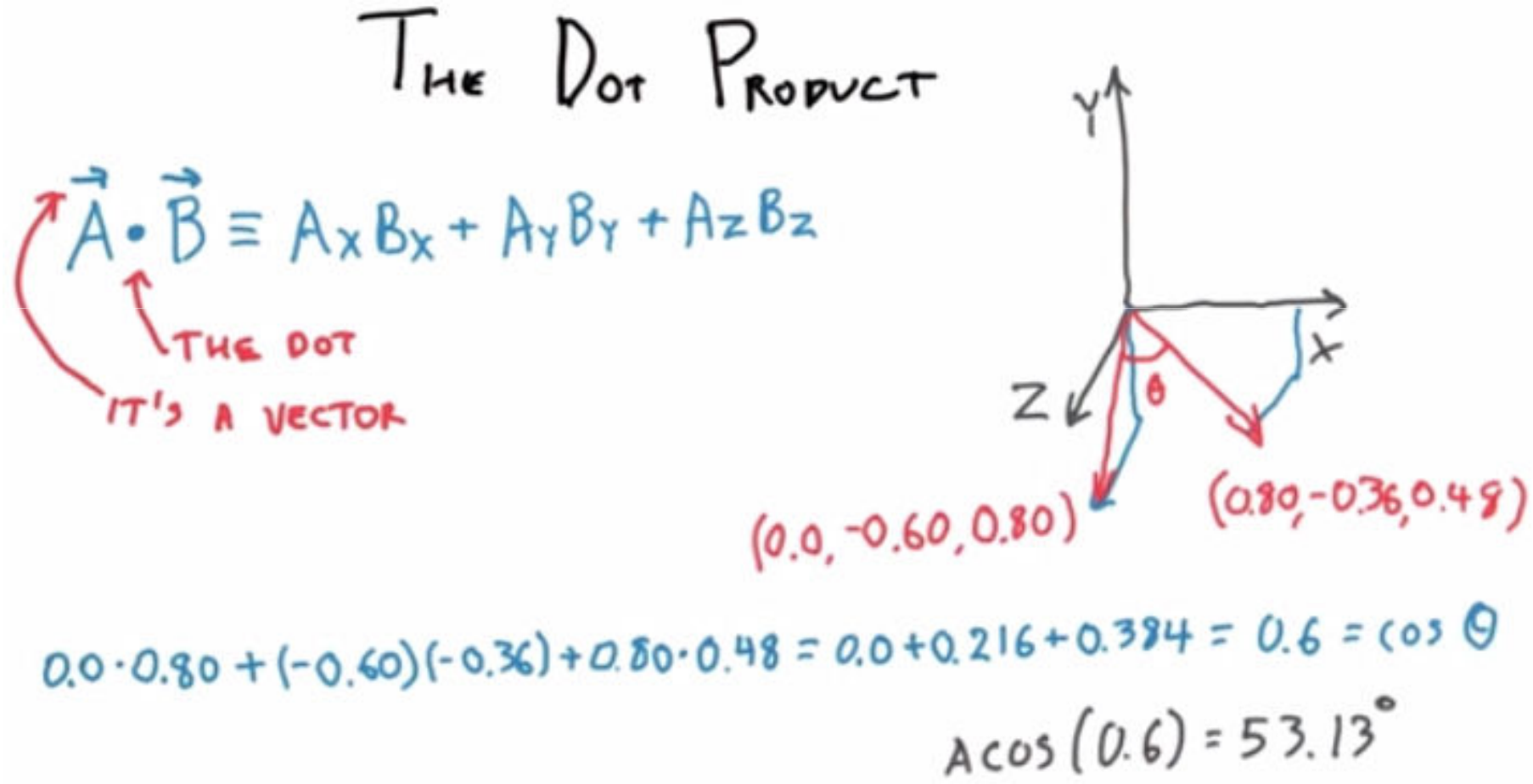


Normalisaition d’un vecteur

- Produit scalaire
- C’est quoi la longueur de (-6, 15, 10)?

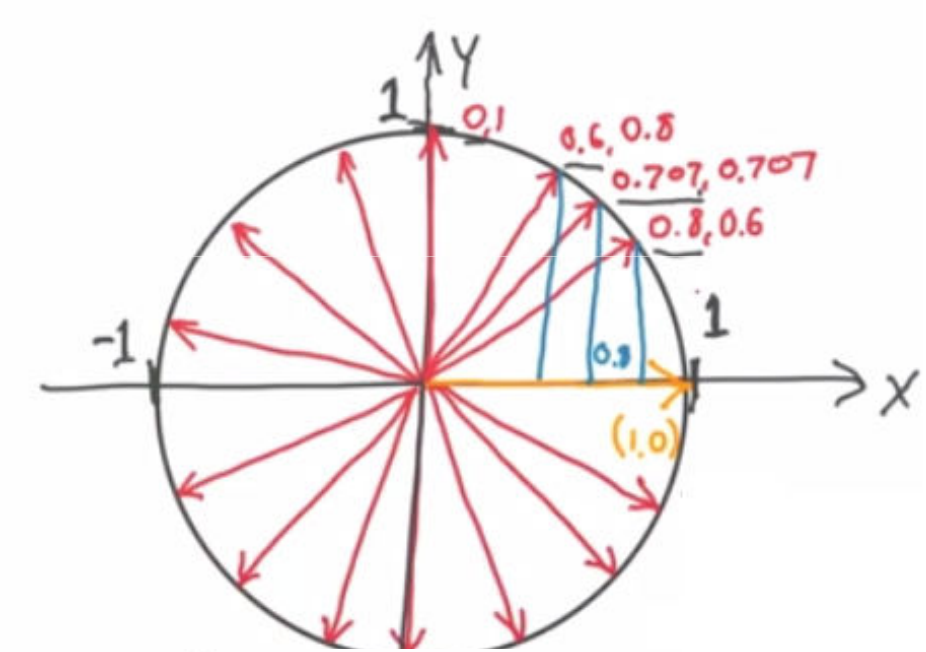


Produit scalire



Explication du produit scalaire

- Fonctionne sur les vecteurs normalisés

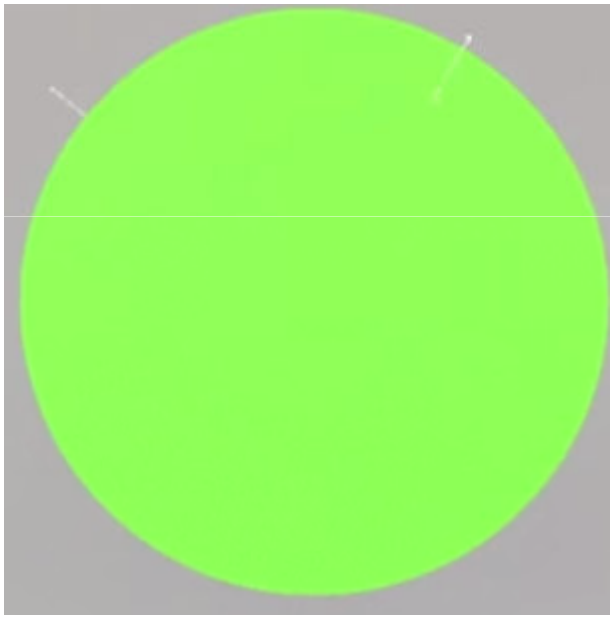


$$C = \text{AMBIENT} + \text{COLOR} \cdot \sum (\vec{N} \cdot \vec{L_i})$$

```
material = new THREE.MeshBasicMaterial( { color: 0x80fc66, shading: THREE.FlatShading } );
```

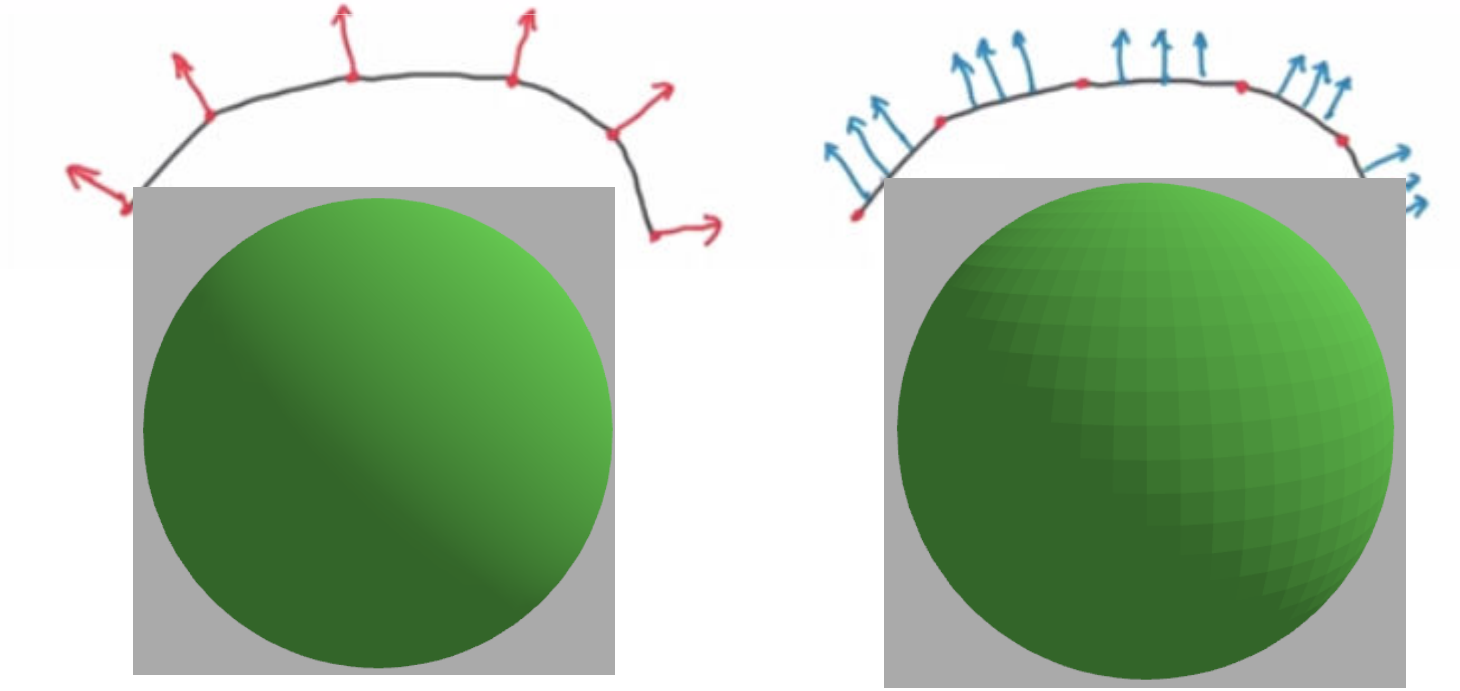
Exercice Rajout lumière diffuse

- Modèle Oren-nayar-blinn



Lissage des normales

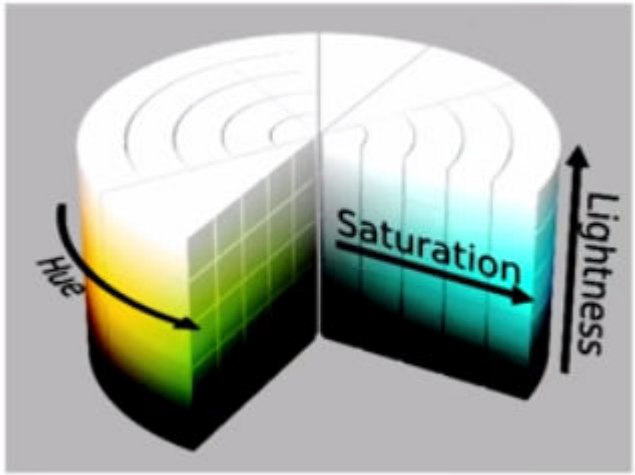
- Lissage depuis les facettes vs depuis normales aux sommets (THREE.FlatShadin / SmoothShading)



Démo Ka, Kd et HSL

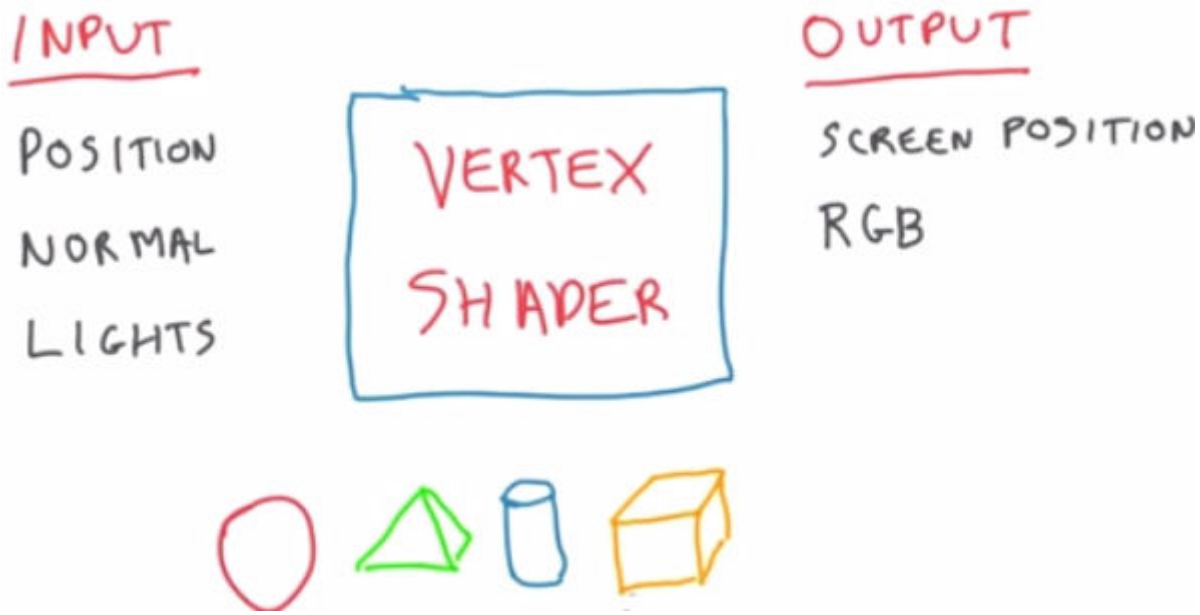
- Ka: contribution de la lumière ambiante
- Kd: contribution de la lumière diffuse
- Système de couleurs HSL
- Demo: Lambert

$$\text{FRAGMENT COLOR} = K_A \cdot \text{MATERIAL} + K_D \cdot \text{MATERIAL} \cdot (\vec{N} \cdot \vec{L})$$



Pré-calcul (Baking)

- On peu enregister les couleurs déjà obtenues par calcul (lissage) depuis les normales (shading)



Quiz

- Quand peut-ont enregistrer les valeurs RGB pré-calculées par sommet(vertex)?
  - ☐ Si l'orientation de l'objet ne change pas
  - ☐ Si la position de l'œil (caméra) et la lumière ne changent pas
  - ☐ Si la direction de la lumière et l'orientation des objets ne changent pas
  - ☐ Si la position de l'œil et l'orientation des objets ne changent pas

Matériau spéculaires

- Modèle Blinn-Phong
- Apparence différente selon angle de vision
- Demo: Spéculaire