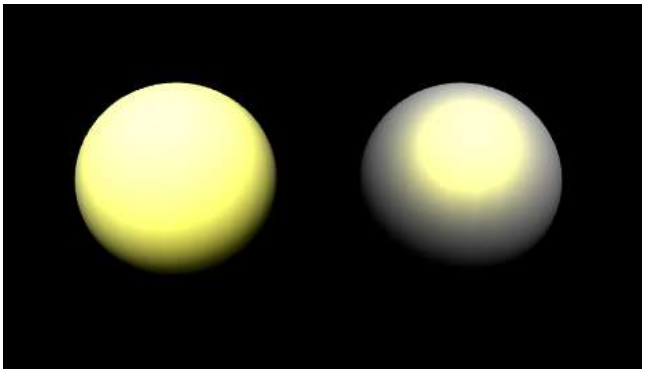
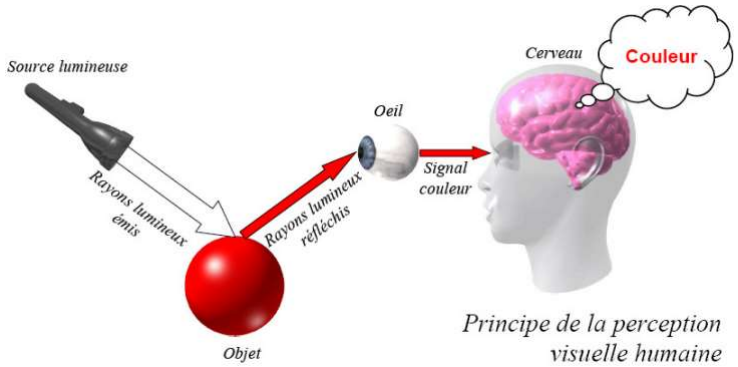


VII – Les shaders

113

VII.1 – Réalité Virtuelle... moins virtuelle !

- Objectif :
  - Rendre les scènes plus réalistes...
  - ... en utilisant un nuanceur (« shader ») : modification des couleurs
  - Représenter l'apparence de l'objet sous l'influence de la lumière
  - Simuler des effets lumineux sur les objets présents dans la scène



Principe de la perception visuelle humaine

114

VII.1 – Calcul de rendu graphique

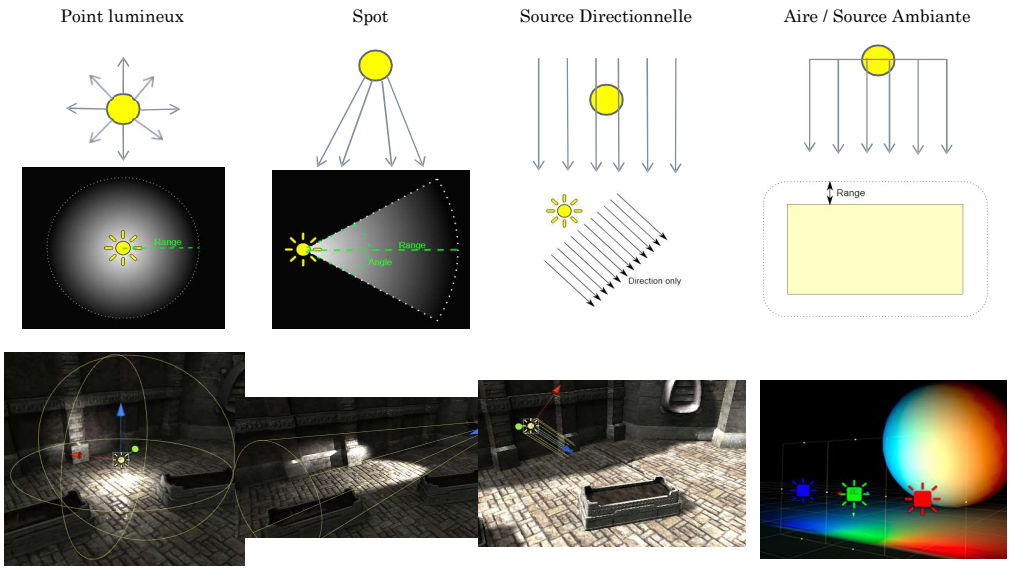
- Principe du calcul du « Rendu graphique » :
  - Exploitation de la carte graphique !
  - ... car généralement des algorithmes complexes, de calcul :
    - d'absorption de la lumière
    - de diffusion,
    - de réflexion,
    - d'ombrage, etc.

- Paramètres principaux pour effectuer ces calculs :
  - types d'**éclairage**
    - diverses formes de sources lumineuses
    - diverses intensités lumineuses
  - caractéristiques des **matériaux** qui constituent les objets
    - capacités à réfléchir ou non la lumière (mat ou brillant, ...)
  - calcul de **l'illumination**
    - analyse de l'apport des sources lumineuses sur l'objet éclairé

115

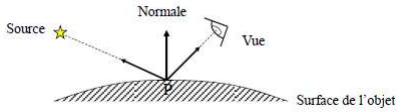
VII.2 – Types d'éclairage

- Différentes formes de sources lumineuses :



VII.3 – Modèle de matériaux

- Utilité du modèle de matériaux
  - Un matériau est vu parce qu'il réfléchit la lumière !



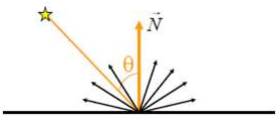
  - Caractérisation d'un matériau par ses capacités à réfléchir la lumière
    - Réflexion ambiante (Ka) : couleur intrinsèque de l'objet
    - Réflexion diffuse (Kd) : couleur émise par réflexion de l'objet
    - Réflexion spéculaire (Ks et Ns) : réflexion de la source et brillance localisée (« shininess »)
  - Exemple : extrait du .MTL de l'objet « dragon »

```
newmtl skin
Ka 0.2286 0.0187 0.0187
Kd 0.1102 0.0328 0.0139
Ks 0.3000 0.3000 0.3000
illum 2         // = 2 : présence spéculaire
Ns 17.8300
```

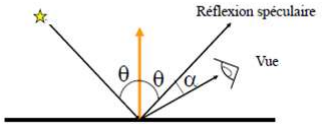
117

VII.3 – Modèle de matériaux et illumination

- Illumination des matériaux
  - Calcul des réflexions du matériau exposé à une source lumineuse
    - Intensité de la source lumineuse : Isl
- Intensités lumineuses résultantes, issues des réflexions
  - Ambiante :
    - réflexion proportionnelle, multi directionnelle
    - $Ia = Isl \times Ka$
  - Diffuse :
    - dépend de l'angle de la source lumineuse
    - $Id = Isl \times Kd \times cos(\theta)$



  - Spéculaire :
    - dépend des angles : de la source et du point de vue
    - $Is = Isl \times Ks \times (cos(\alpha))^{Ns}$

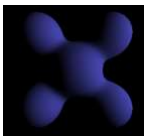


118

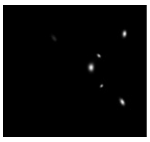
VII.3 – Modèle d'illumination de « Phong »

- Aspect des diverses composantes :
  - Réflexion **Ambiante**
    - Coloriage uniforme avec la couleur de base
    - Pas d'ombrage sur les différentes parties de l'objet
    - Silhouette apparaissant de couleur uniforme, sans relief

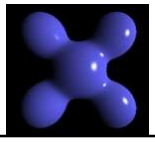
  - Réflexion **Diffuse**
    - La surface diffuse de façon uniforme dans tout l'espace
    - L'aspect d'un point de la surface
      - dépend de l'angle de la source de lumière (ombrage)
      - mais ne dépend pas du point de vue de l'utilisateur



  - Réflexion **Spéculaire**
    - La surface diffuse de façon non uniforme, selon le point de vue
    - La source de lumière apparaît donc comme une tâche brillante
      - Cette tâche est de la couleur de la source (et non de l'objet)
      - Cette tâche se déplace à la surface de l'objet lorsque l'observateur bouge



  - Modèle d'illumination de « Phong »
    - La couleur RGB finale est la somme des 3 !
    - Couleur Finale = Ambiante + Diffuse + Spéculaire



119

VII.4 – Principe d'éclairage dans Processing

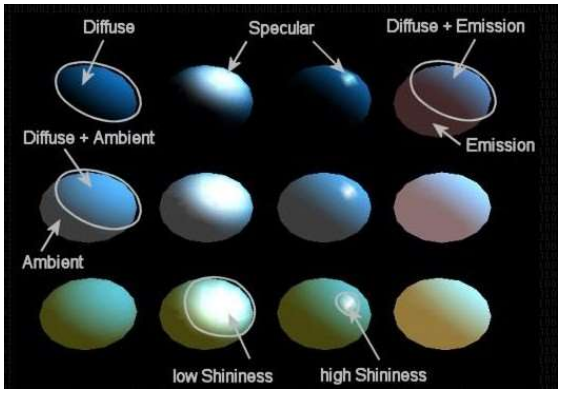
- Principe de l'éclairage des scènes :
  - Ne fonctionne qu'en mode 3D : donc `size(... , ... , P3D);`
  - Sources de lumière blanche... ou avec une couleur RGB quelconque !
  - Toutes les valeurs par défaut, dans Processing : `ligths()`
    https://processing.org/examples/onoff.html

- Autres types d'éclairage, dans Processing :
  - Ambiante : `ambientLight()`
    - Pas de direction privilégiée
    - Généralement utilisée en la combinant avec un autre type de lumière
  - Directionnelle : `directionalLight()`
    - Très dirigée
    - Effet d'ombre sur l'objet
  - Spot : `spotLight()`
    - Identique à la lumière directionnelle, mais avec plus d'options...
    - Choix de l'angle du cône d'éclairage
  - Point : `pointLight()`
    - Identique au spot, mais avec un cône à 180 degrés

120

VII.4 – Modèle d'illumination dans Processing

- Fonctions à appeler avant le dessin de l'objet
  - `ambient()`
    - couleur intrinsèque de l'objet (remplace "fill")
  - `emissive()`
    - couleur renvoyée par l'objet sous un éclairage blanc (joue le rôle de "diffuse")
  - `specular()` et `shininess()`
    - effet de brillance de type "gloss" ("specular" = Ks et "shininess" = Ns)



121

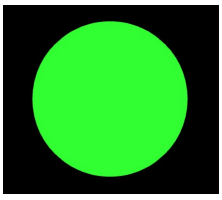
VII.5 – Aperçu du rendu dans Processing

- `ambient()`

  couleur de réflexion **ambiante** : couleur intrinsèque

  Ka

```
ambientLight(255,255,255); // lumiere ambiante en blanc
ambient(50, 255, 50); // reflexion ambiante en vert
fill(255,255,255); // objet en blanc
sphere(180);
```

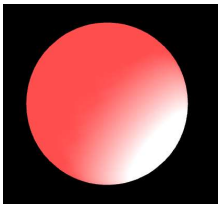


- `emissive()`

  couleur **diffuse** renvoyée par l'objet sous un éclairage blanc

  Kd

```
pointLight(255,255,255, mouseX, mouseY, 120); // lumiere en blanc, obligatoire
emissive(255, 80, 80); // reflexion en rouge
fill(255,255,255); // objet en blanc
sphere(180);
```



122

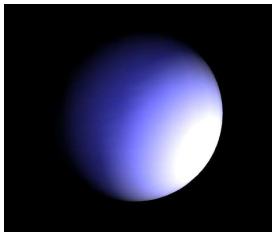
VII.5 – Aperçu du rendu dans Processing

- `specular()`

  réflexion **spéculaire** : effet miroir, de reflet dû à la brillance de l'objet

  Ks

```
lightSpecular(255, 255, 255); // lumiere blanche
specular(80, 80, 255); // reflet bleu fonce
pointLight(255, 255, 255, mouseX, mouseY, 120);
// Attention a l'ordre : pointLight APRES
```

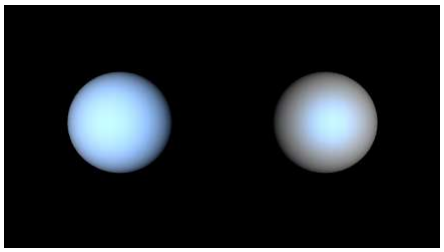


- `shininess()`

  réflexion de la source sur l'objet, réglage effet « gloss »

  Ns

```
lightSpecular(0, 55, 128); // bleu fonce
specular(255, 255, 255); // reflet blanc
directionalLight(200, 200, 200, 0, 0, -1);
translate(width/3, height/2, 0);
shininess(1.0);
sphere(50);  // sphere a gauche
translate(width/3, 0, 0);
shininess(8.0);  // gloss plus faible
sphere(50);  // sphere a droite
```

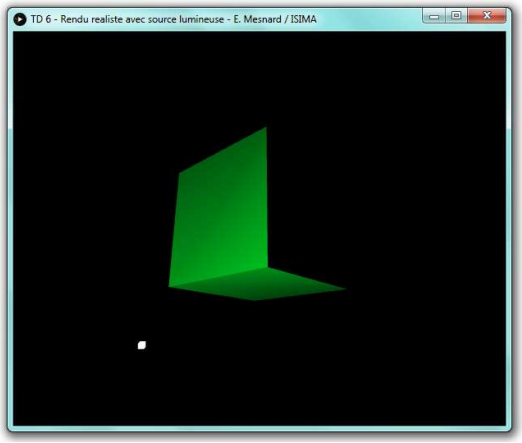


123

TD 8 Réalité Virtuelle : rendu « réaliste »

- Travail demandé :
  - Faire une application qui permet de vérifier, à la souris, l'impact de la position d'une source de lumière de type « `pointLight` », en temps réel.
- Spécifications :
  - Cube 3D blanc (!) tournant
  - Source placée à la souris à la profondeur Z = 120 identique à celle du cube
  - Teinte de la source variable barre espace : On/Off
  - Rappel : `PushMatrix()` et `PopMatrix()` permettent de conserver et restituer les conditions de tracé (repère) Exemple :

```
pushMatrix(); // Sauvegarde repere
translate(50, 20, 120);
box(5); // trace un cube en (50,20,120)
popMatrix();
box(5); // trace un cube en (0,0,0)
```



124