## Listes des méthodes et procédures utiles pour le tp1

**couleur.h : classe Couleur**

Couleur(const reel&r,const reel&g,const reel&b);

```
          // constructeur d'une couleur r,g,b
const     reel&     rouge();
          // retourne la composante rouge de la couleur
const     reel&     vert();
          // retourne la composante verte de la couleur
const     reel&     bleu();
          // retourne la composante bleue de la couleur
          // On définie les opérations * + == et != entre
          // deux couleurs
Couleur        operator*(const Couleur&) const;
Couleur        operator+(const Couleur&) const;
booleen        operator==(const Couleur&) const;
booleen        operator!=(const Couleur&) const;

Couleur        operator*(const reel&) const;
      // on défini l'opération * entre une couleur et un
      // réel
```

**exemples:**

```
Couleur blanc = Couleur (1.0,1.0,1.0);
Couleur noir = Couleur (0.0,0.0,0.0);
Couleur c,c1,c2;
reel x,r,g,b;

c1=Couleur(0.5,0.2,1.0);
r=c1.rouge();
g=c1.vert();
b=c1.bleu();
c2=c1+blanc;   c=c1*c2;
x=0.5;
c=c*x; // attention couleur * reel, et non l'inverse.

if (c==noir) c1=c2;
```

**alg.h: classe point**

```
  point(const reel& x, const reel& y, const reel& z);
          // constructeur de point.

     const   reel&   x() const;
          // retourne la coordonnée x du point.

     const   reel&   y() const;
          // retourne la coordonnée y du point.

     const   reel&   z() const;
          // retourne la coordonnée z du point.
     void coordonnees (reel *x, reel *y, reel *z) const;
     // retourne la coordonnée x,y et z du point.
```

**exemples:**

```
     point p1,p2;
     reel x,y,z;

     p1 = point(1.0,1.0,1.0);

     x=point.x();

     p1.coordonnees(&x,&y,&z);
```

**alg.h : classe vecteur**

```
     vecteur(const reel& x, const reel& y, const reel& z);
     // constructeur d'un vecteur avec trois coordonnées
     vecteur(const point& p, const point& q);
          // constructeur d'un vecteur avec deux points

     const   reel&   x() const;
          // retourne la coordonnée x du vecteur

     const   reel&   y() const;
          // retourne la coordonnée y du vecteur

     const   reel&   z() const;
          // retourne la coordonnée z du vecteur
```

```
     void composantes (reel *x, reel *y, reel *z) const;
          // retourne les coordonnées x y et z du vecteur

     reel norme() const;
          // retourne la norme du vecteur

     vecteur unitaire() const;
          // retourne le vecteur unitaire

     void normalise();
          // normalise le vecteur
```

**algutil.h**

```
reel      operator*(const vecteur &v, const vecteur &u);
          // produit scalaire de deux vecteurs
          // retourne un reel

vecteur   operator*(const reel &k, const vecteur &v);
          // multiplication d'un reel par un vecteur
          // retourne un vecteur

vecteur   operator+(const vecteur &v, const vecteur &u);
          // addition de deux vecteurs
          // retourne un vecteur

point     operator+(const point &p, const vecteur &u);
          // addition d'un point avec un vecteur
          // retourne un point

point     operator-(const point &p, const vecteur &u);
          // soustraction d'un vecteur a un point
          // retourne un point

point     operator+(const vecteur &u, const point &p);
          // addition d'un point avec un vecteur
          // retourne un point

vecteur   operator-(const vecteur &v);
          // retourne le vecteur inverse
```

```
vecteur   operator-(const vecteur &v, const vecteur &u);
          // soustraction de deux vecteurs
          // retourne un vecteur

vecteur   operator-(const point &p, const point &q);
          // soustraction de deux points
          // retourne un vecteur exemples:

     point p1,p2,p3;
     vecteur v1,v2,v3;
     reel k;

     k=4.0;
     p1 = point(1.0,2.0,3.0);
     p2 = point(2.0,3.0,4.0);
     v1 = p1 - p2;
     v3 =  v1.unitaire();
     v1.normalise();
     p3 = p1 + v1;
     p3 = p2 + k*v1;
     v2 = -v1;
     v3 = v1 - v2;
     k = c3.norme();
```

**couleurs.h: classe Couleurs**

```
// Attributs non-géométriques (attention, couleurs avec
// un «s» !)

     const     Couleur& diffus() const;
     // retourne le coefficient de reflexion diffuse

     void diffus(const Couleur & c);
     // defini le coefficient de reflexion diffuse

     const     Couleur& speculaire() const;
     // retourne le coefficient de reflexion speculaire

     void speculaire(const Couleur & c);
     // defini le coefficient de reflexion speculaire
```

```
const      Couleur& t_diffus() const;
// retourne le coefficient de transmission diffuse

void t_diffus(const Couleur & c);
// defini le coefficient de transmission diffuse

const      Couleur& t_speculaire() const;
// retourne le coefficient de transmission parfaite
// speculaire

void t_speculaire(const Couleur & c);
// defini le coefficient de transmission parfaite
// speculaire

const      Couleur& reflechi() const;
// retourne le coefficient de reflexion mirroir

void reflechi(const Couleur & c);
// defini le coefficient de reflexion mirroir

const      Couleur& transmis() const;
// retourne le coefficient de transmission

void transmis(const Couleur & c);
// defini le coefficient de transmission
```

**exemples:**

```
Couleurs AttrNonGeo;
Couleur e,diff;
Couleur blanc = Couleur (1.0,1.0,1.0);

diff=AttrNonGeo.diffus();
AttrNonGeo.speculaire(blanc);
```

**objet.h: classe objet**

```
 const Attributs&   attributs () const;
// retourne les attributs de l'objet
```

**attr.h : classe Attributs**

```
      Couleurs couleurs() const;
     // retourne les attributs non-géométriques des
// attributs
```

**exemples:**

```
     Attributs a;
     Objet *o;
     Couleurs couls;
     Couleur diff;


     a = o->attributs();
     couls = a.couleurs();
     diff = couls.diffus();
```

**vision.h: fonction Vision_Normee**

```
Transformation      Vision_Normee (const Camera& p);
     // Retourne la transformation qui amènera un point en
     // coordonnées universelles en coordonnées de vision
     // normees.
```

**exemples:**

```
     Transformation t;

     t=Vision_Normee(p);
```

**transfo.h: classe Transformation**

```
     point     transforme(const point& p) const;
     // retourne un point qui est la transformation de p.
     vecteur transforme(const vecteur& v) const;
     // retourne un vecteur qui est la transformation de v.
```

```
    Transformation inverse() const;    //
retourne la transformation inverse.

 Transformation operator*(const Transformation& t) const;
    // composition de transformations.
```
**exemples:**
```
    Transformation t1,t2,t3;
    point p1,p2;

    t1 = ...;
    t2 = ...;
    t1 = t1 * t2;
    t3 = t1.inverse();  // t3 égal t1 inverse.
    p1 = point(1.0,2.0,3.0);
    p2 = t3.transforme(p1);
```

**inter.h: fonction Objet_Inter**

booleen Objet_Inter (Objet& o, const point& p, const vecteur& v, reel *k, vecteur *vn, Couleurs* c);

```
    // retourne vrai si un rayon defini par p et v intersecte
     // l'objet o et fait les calculs de la distance k
    // (le pt d'intersection = p + v*k), du vecteur
    // normal vn et des attributs non-géométriques c
    // au pt d'intersection.
```
**exemples:**
```
    Objet *obj;
    vecteur DirectionRayon,vn;
    point OrigineRayon;
    reel distance;
    Couleurs attrNonGeo;

    if (Objet_Inter (*obj, OrigineRayon, DirectionRayon,
              &distance, &vn, &attrNonGeo)) ...
```

**camera.h:**

**classe Camera**

```
// classe contenant les paramètres de vision ainsi que les
// lumières

     point              PO() const;
     // retourne le point de l'oeil.

     const entier& NbLumiere() const;
     //Retourne le nombre de lumieres (0 - 8)

     const Couleur& Ambiante(const int& i);
     // retourne l'intensité ambiante de la lumière i
     // (première lumière : i=0)

     const Couleur& Diffuse(const int& i);
     //retourne l'intensité de la lumière i pour le calcul
     // du diffus et du spéculaire.
     // (première lumière : i=0)

     const point& Position(const int& i);
     // retourne la position de la lumière i.
     // (première lumière : i=0)
```