Université Gustave Eiffel — INSTITUT D'ÉLECTRONIQUE ET D'INFORMATIQUE GASPARD-MONGE

Master Informatique 1

TD Infographie n°1

RayTracer 1.0 - utilitaires

Quelques algorithmes de base du RayTracer à tester à part

Préambule : Produit scalaire ($\vec{u}\bullet\vec{v}$) et produit vectoriel ($\vec{u}\wedge\vec{v}$)

On travaille en dimension 3, dans le repère canonique de $\mathbb{R}^3$ $\mathcal{R}(O,x,y,z)$.
Les produits scalaire et vectoriel sont les 2 opérateurs fondamentaux sur les vecteurs.

- **Produit scalaire** $\vec{u}\bullet\vec{v} = (x_u.x_v) + (y_u.y_v) + (z_u.z_v) \quad \in \mathbb{R}$

  Propriétés :
  - $\vec{u}\bullet\vec{v} = \vec{v}\bullet\vec{u}$ (symétrie)
  - $\vec{u}\bullet\vec{u} = {x_u}^2 + {y_u}^2 + {z_u}^2 = \|\vec{u}\|^2$
  - $\vec{u}\bullet(\vec{v}+\vec{w}) = \vec{u}\bullet\vec{v} + \vec{u}\bullet\vec{w}$ (distributivité)
  - $\vec{u}\bullet\vec{v} = 0 \Leftrightarrow$ les vecteurs sont orthogonaux
  - $\vec{u}\bullet\vec{v} = \|\vec{u}\|.\|\vec{v}\|.\cos(\widehat{\vec{u},\vec{v}})$

- **Produit vectoriel** $\vec{u}\wedge\vec{v} = \begin{vmatrix} x_u \\ y_u \\ z_u \end{vmatrix} \wedge \begin{vmatrix} x_v \\ y_v \\ z_v \end{vmatrix} = \begin{vmatrix} y_u.z_v - z_u.y_v \\ -x_u.z_v + z_u.x_v \\ x_u.y_v - y_u.x_v \end{vmatrix} \quad \in \mathbb{R}^3$

  Propriétés (3D) :
  - $\vec{u}\wedge\vec{v} = -\vec{v}\wedge\vec{u}$ (attention : anti-symétrie)
  - $\vec{u}\wedge(\vec{v}+\vec{w}) = \vec{u}\wedge\vec{v} + \vec{u}\wedge\vec{w}$ (distributivité)
  - $\vec{u}\wedge\vec{v} = \vec{0} \Leftrightarrow$ les vecteurs sont colinéaires
  - $\|\vec{u}\wedge\vec{v}\| = \|\vec{u}\|.\|\vec{v}\|.\sin(\widehat{\vec{u},\vec{v}})$

  Propriétés (2D) : le produit vectoriel en dimension 2 n'est pas défini, mais...

  $\vec{u}\wedge\vec{v} = \begin{vmatrix} x_u \\ y_u \\ 0 \end{vmatrix} \wedge \begin{vmatrix} x_v \\ y_v \\ 0 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ x_u.y_v - y_u.x_v \end{vmatrix} \quad \in \mathbb{R}^3$

  On peut donc considérer que, en dimension 2 : $\vec{u}\wedge\vec{v} = \begin{vmatrix} x_u \\ y_u \end{vmatrix} \wedge \begin{vmatrix} x_v \\ y_v \end{vmatrix} = x_u.y_v - y_u.x_v \in \mathbb{R}$

- Vecteur "normés" (i.e. de norme 1)

  en 2D ($\vec{u}\begin{vmatrix} x_u \\ y_u \end{vmatrix}$ et $\vec{v}\begin{vmatrix} x_v \\ y_v \end{vmatrix}$) : $\vec{u}\bullet\vec{v} = \cos(\widehat{\vec{u},\vec{v}})$ et $\vec{u}\wedge\vec{v} = \sin(\widehat{\vec{u},\vec{v}})$

Ces 2 opérateurs, très économiques, donnent donc toutes les informations nécessaires : longueur, orientation, direction, angles ....

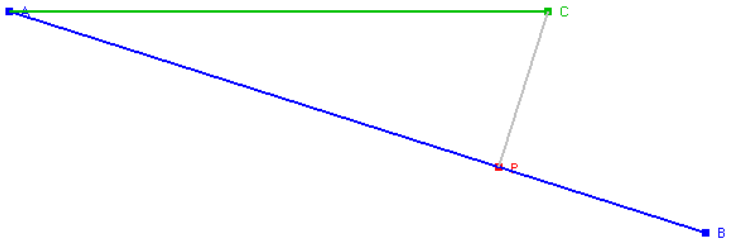
☞ quel que soit le langage de prog., les fonctions `cos` et `sin` sont TRES coûteuses (basées sur des opérations polynômiales). Il faut limiter leur utilisation.

☞ à l'inverse les opérateurs $\bullet$ et $\wedge$ sont très économique : 2 multiplications et 1 addition en 2D.

☞ Lorsqu'on a besoin d'un angle (valeur, orientation...) : passer par ces 2 opérateurs

Exercice 1. (Projection d'un point sur un segment de $\mathbb{R}^2$)
On considère trois points de $\mathbb{R}^2$ $\{A,\ B,\ C\}$

1. En utilisant uniquement le Produit Scalaire 2D, déterminer la position de la projection orthogonale P de C sur (AB)
2. Comment déterminer, en plus si P est entre A et B, avant A ou après B
3. Que devient ce calcul si la droite est définie par le point A et un vecteur $\vec{u}$ normé ($\vec{u} = \overrightarrow{AB}/\|\overrightarrow{AB}\|$) ?
4. Ecrire une fonction `Projection` qui renvoie la projection d'un point C sur une droite AB
   Intégrer cette fonction dans un programme graphique complet, permettant d'illustrer cet algorithme.

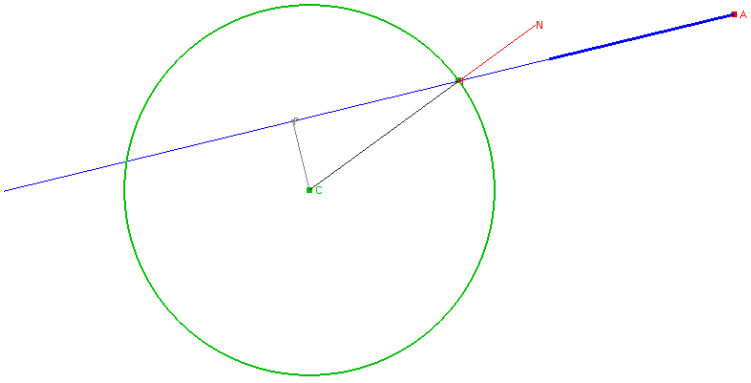


Exercice 2. (Intersection de deux segments de $\mathbb{R}^2$)
On considère quatre points de $\mathbb{R}^2$ $\{A,\ B,\ C,\ D\}$.

1. Donner l'équation de la droite support du segment $[A, B]$.
2. On veut déterminer la valeur de t telle que $P(t)$ soit le point d'intersection de $[A, B]$ et $[C, D]$.
   - Quelle condition doit vérifier t pour que l'intersection entre les droites (A, B) et (C, D) existe et soit située sur le segment $[C, D]$?
   - En utilisant uniquement le Produit Vectoriel 2D, déterminer le point d'intersection des deux segments, s'il existe.
3. Ecrire une fonction `InterSegment` qui calcule le point d'intersection entre deux segments (4 points) passés en arguments. Cette fonction renverra par ailleurs un message booléen selon que l'intersection existe ou non.
   Intégrer cette fonction dans un programme graphique complet, permettant d'illustrer cet algorithme.

Exercice 3. (Intersection Rayon/Cercle dans $\mathbb{R}^2$)
On considère un cercle de centre C, de rayon r et un rayon, issu d'un point A et de direction $\vec{u}$ (normé).

1. En utilisant la fonction de Projection précédente, déterminer le premier (i.e. le plus proche de la source) point d'intersection éventuel du rayon avec le cercle.
   On s'efforcera d'éliminer le plus rapidement possible les cas sans intersection valable.
2. Ecrire une fonction `RayInterCercle` qui renvoie cette intersection, si elle existe, ainsi que la normale au cercle en ce point.
   Intégrer cette fonction dans un programme graphique complet, permettant d'illustrer cet algorithme.



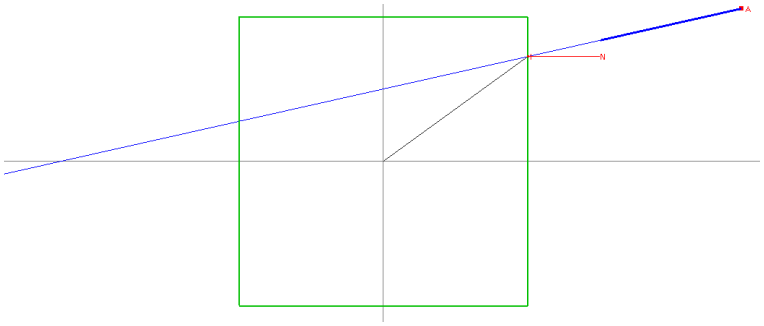
3. Cet algorithme peut-il fonctionner avec une ellipse quelconque ?

▸ **Exercice 4. (Intersection Rayon/Carré dans $\mathbb{R}^2$)**

On considère le carré **canonique** centré sur l'origine Ω, de demi-côté 1 et un rayon $[A, \vec{u})$ (normé).

1. En utilisant la fonction d'intersection de segments précédente (ou une variante simplifiée adaptée au cas du carré canonique), déterminer le premier (i.e. le plus proche de la source) point d'intersection éventuel du rayon avec le carré.
   On s'efforcera d'éliminer le plus rapidement possible les cas sans intersection valable.

2. Ecrire une fonction `RayInterCarre` qui renvoie cette intersection, si elle existe, ainsi que la normale au carré en ce point.
   Intégrer cette fonction dans un programme graphique complet, permettant d'illustrer cet algorithme.



3. Cet algorithme peut-il fonctionner avec un rectangle quelconque ?