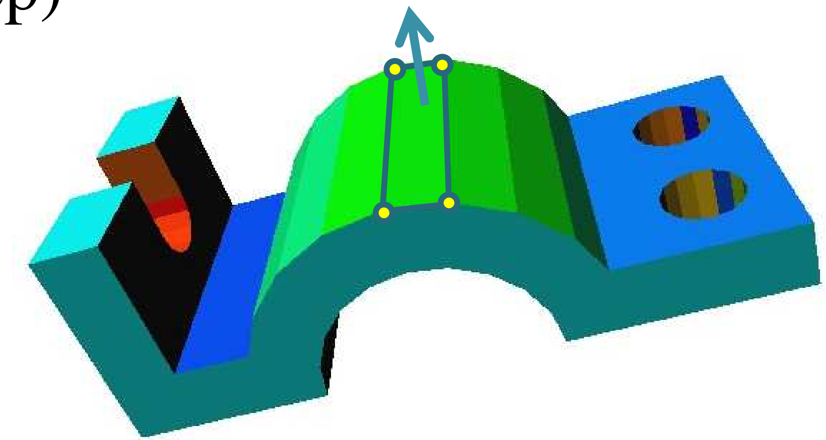
Boundary representation (Brep)

Représentation frontière



- 1 ière partie – 2D : Polygones

1. Rappels de géométrie
2. Représentation informatique
3. Algorithmes: calcul de l'aire, localisation d'un point…
4. Intersections et collisions
5. Opérations booléennes
6. Transformations et coordonnées homogènes
7. Bibliographie, TP…

1. Rappels de géométrie

1. Produits vectoriel, produit scalaire
2. Les 3 représentations d'une courbe
3. Propriétés géométriques et topologiques des Polygones

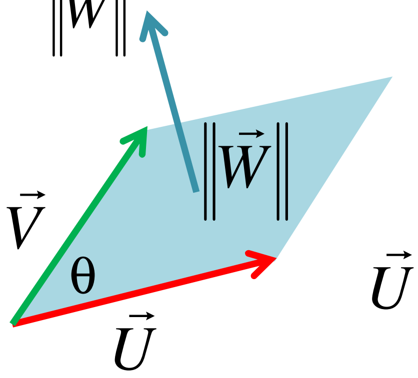
Produit Vectoriel (3D)

Rappels de géométrie (1)

$$\vec{W} = \vec{U} \wedge \vec{V}$$

$$\vec{w} = \frac{\vec{W}}{\left\|\vec{W}\right\|}$$

$$\left\|\vec{W}\right\| = \left|\left\|\vec{U}\right\|\left\|\vec{V}\right\|\sin(\theta)\right|$$



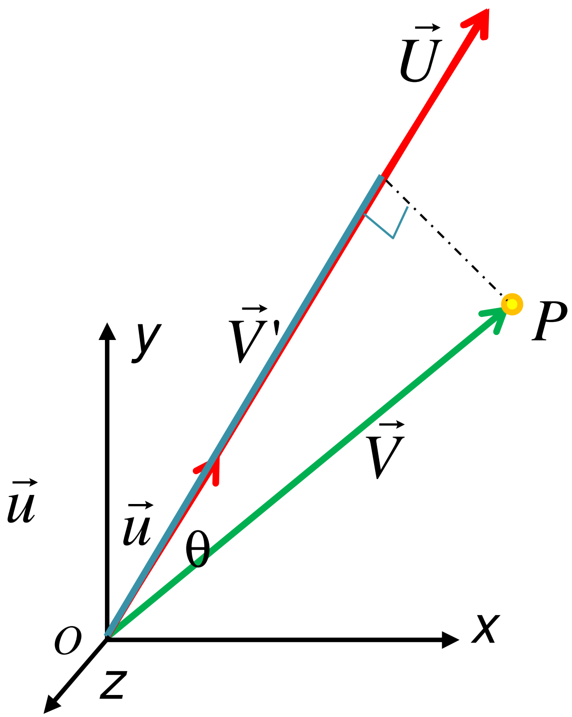
$$\vec{U} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \quad W = \begin{pmatrix} w_x = u_y.v_z - u_z.v_y \\ w_y = u_z.v_x - u_x.v_z \\ \boxed{w_z = u_x.v_y - u_y.v_x} \end{pmatrix} \text{2D}$$

$$M_u V = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} -u_z v_y + u_y v_z \\ u_z v_x - u_x v_z \\ -u_y v_x + u_x v_y \end{pmatrix}$$

Unity : W=Vector3.Cross (U,V)

Scilab : W=cross (U,V)

Produit scalaire



$$w = \vec{U}.\vec{V}$$

$$w = \left\|\vec{U}\right\|\left\|\vec{V}\right\|\cos(\theta)$$

$$\left\|\vec{V}\right\|^2 = \vec{V}.\vec{V}$$

Projection sur l'axe $\vec{V}' = \left\|\vec{V}\right\|\cos(\theta)\vec{u}$

$w = {}^tU * V$ $\quad w = u_x.v_x + u_y.v_y + u_z.v_z$

$$P_u * V = \begin{pmatrix} u_x^2 & u_x.u_y & u_x.u_z \\ u_y.u_x & u_y^2 & u_y.u_z \\ u_z.u_x & u_z.u_y & u_z^2 \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} w.u_x \\ w.u_y \\ w.u_z \end{pmatrix}$$

*Scilab : a = U*V'*

P_u est la matrice de projection sur l'axe u (unitaire)

1.2. Représentations d'une courbe

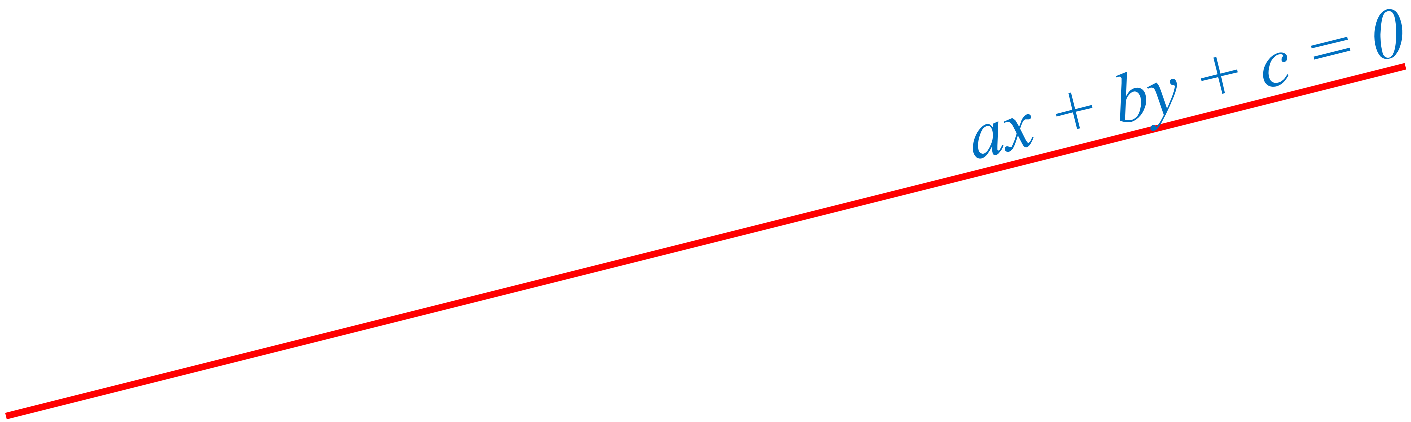
Exemple de la droite

- **Représentation algébrique**

  $y = A\,x + B$ ou $x = A\,y + B$ (explicite)

  $a\,x + b\,y + c = 0$ (implicite)

  $a^2 + b^2 = 1$ (normalisée)



1.2. Représentations d'une courbe

Exemple de la droite

- Représentation algébrique

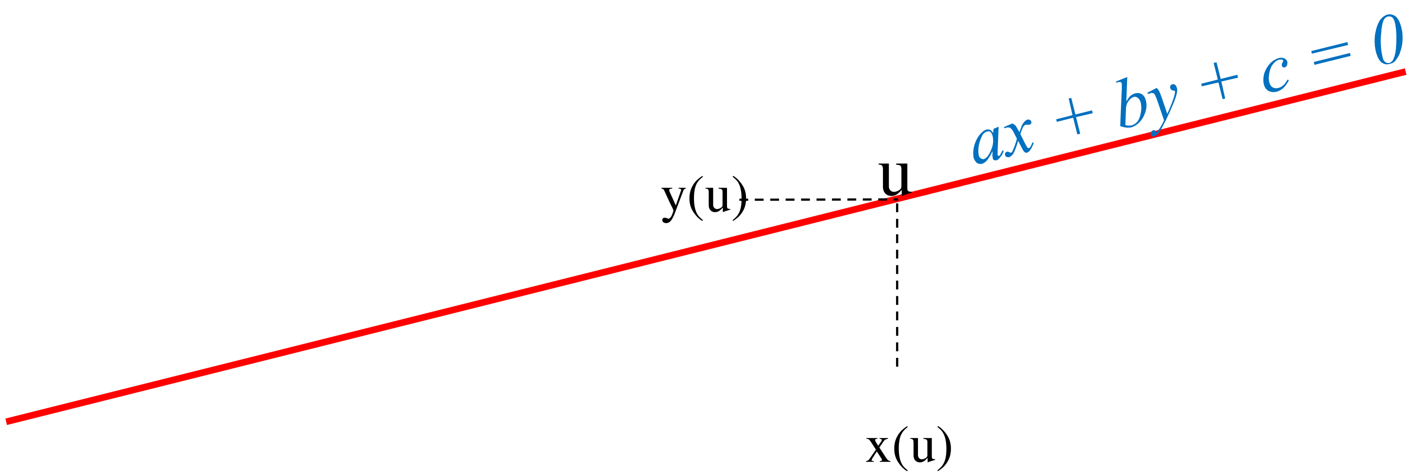
$y = A\,x + B$ ou $x = A\,y + B$ (explicite)

$a\,x + b\,y + c = 0$ (implicite)

$a^2 + b^2 = 1$ (normalisée)

- **Représentation paramétrique**

$$x(u) = a_0 * u + b_0$$
$$y(u) = a_1 * u + b_1$$



1.2. Représentations d’une courbe

Exemple de la droite

- Représentation algébrique

$$y = A\,x + B \qquad \text{ou} \qquad x = A\,y + B \quad \text{(explicite)}$$

$$a\,x + b\,y + c = 0 \qquad \text{(implicite)}$$

$$a^2 + b^2 = 1 \ \text{(normalisée)}$$

- Représentation paramétrique

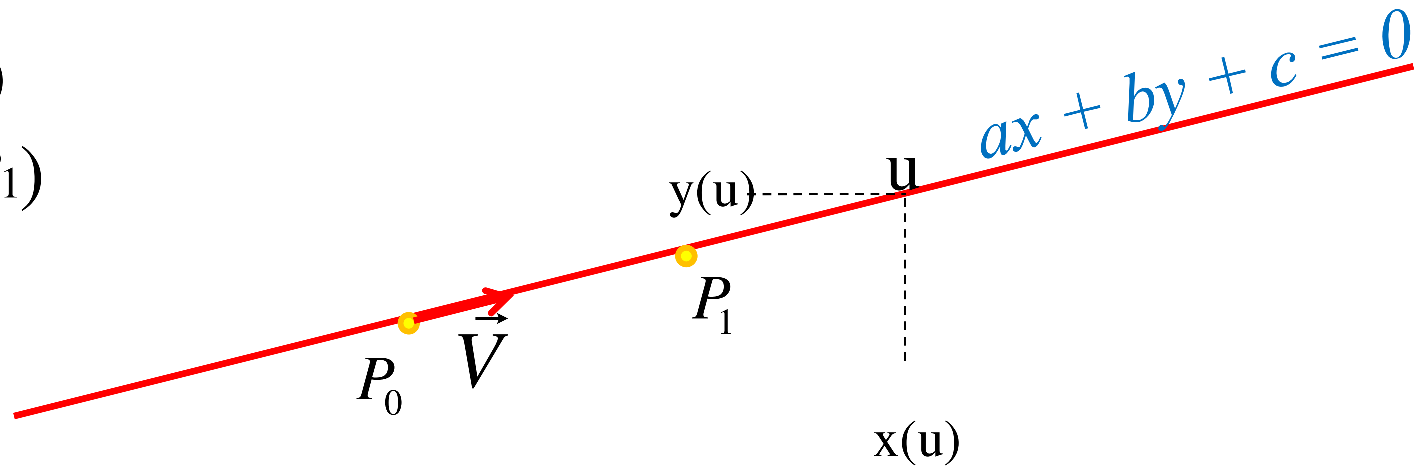
$$x(u) = a_0 * u + b_0$$

$$y(u) = a_1 * u + b_1$$

- **Représentation vectorielle :**

$$(O, \vec{V})$$

$$(P_0, P_1)$$



Représentations d'une courbe

Exemple de la droite

- Représentation algébrique

  y= A x + B ou x= A y + B (explicite)

  a x + b y + c = 0 (

$$P \in D(O,\vec{V}) \Leftrightarrow \vec{V} \wedge \overrightarrow{OP} = 0$$

- Représentation paramétrique

  $x(u) = a_0 * u + b_0$

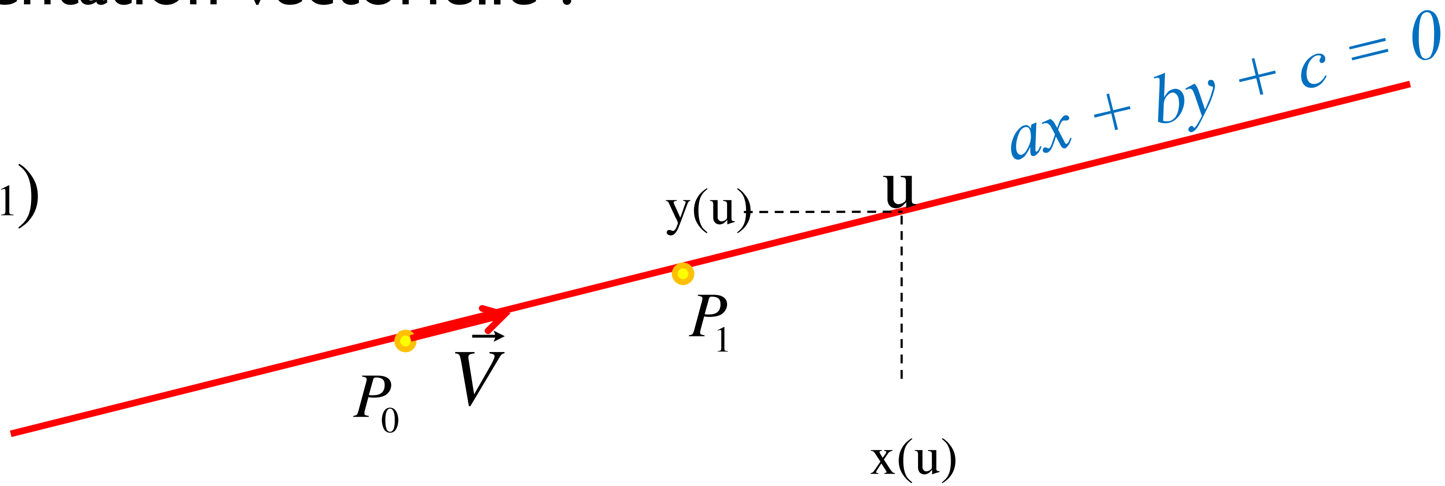
  $y(u) = a_1 * u + b_1$

$$P(u) = O + u\vec{V}$$

- Représentation vectorielle :

  $(O,\vec{V})$

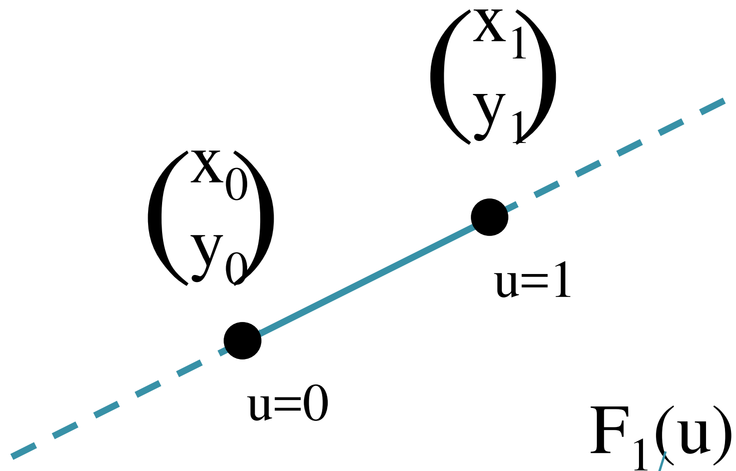
  (P_0, P_1)



Courbe paramétrique polynomiale

Exemple d'une droite ou d'un segment (P_0, P_1)

$$\begin{cases} x(u) = a_0 * u + b_0 \\ y(u) = a_1 * u + b_1 \end{cases}$$



$$\begin{cases} x(u) = x_1 * ? + x_0 * ? \\ y(u) = y_1 * ? + y_0 * ? \end{cases}$$

Choix de u : $x(u=0) = x_0$

$x(u=1) = x_1$

$F_1(u)$ $F_0(u)$

$$\begin{cases} x(u) = x_1 * u + x_0 * (1-u) \\ y(u) = y_1 * u + y_0 * (1-u) \end{cases}$$

Une courbe polynomiale paramétrique contrôlée par les points

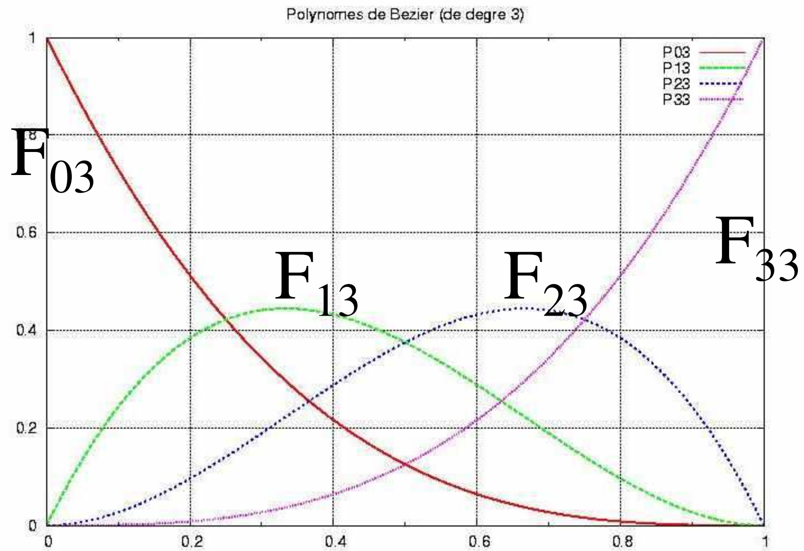
$$\boxed{X(u) = \sum_{i=0}^{m} P_i \, F_{im}(u)}$$

Courbes paramétriques polynomiales

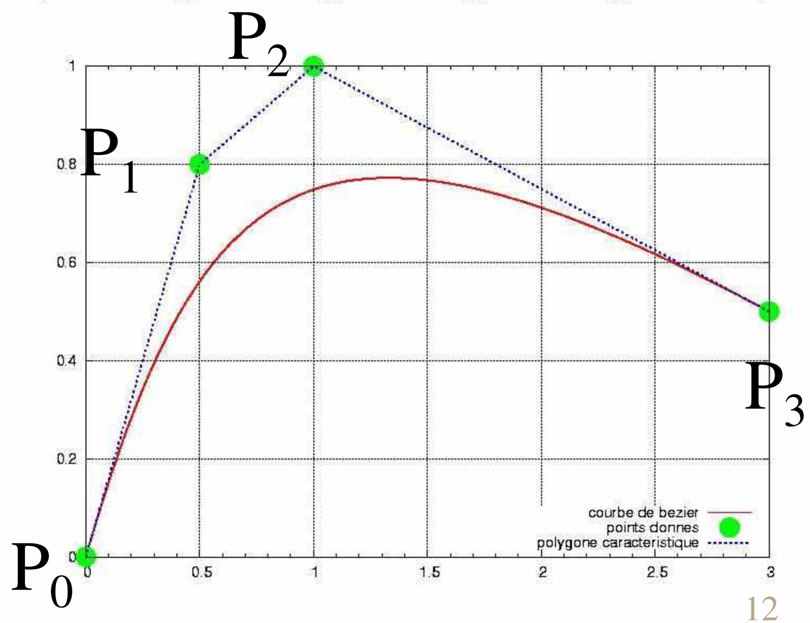
Exemple de la courbe de Bezier

$$\boxed{X(u) = \sum_{i=0}^{m} P_i \, F_{im}(u)}$$

$$F_{im}(u)=(m!/((m-i)!\ i!))\ u^i(1-u)^{m-i}$$



$$\begin{pmatrix} x(u) \\ y(u) \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \end{pmatrix} \begin{vmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 \\ u \\ u^2 \\ u^3 \end{vmatrix}$$



Convention : $0^0 = 1$

Représentations du cercle

- Représentation algébrique

$$a*(x-x_0)^2 + b*(y - y_0)^2 +- r^2 = 0 \qquad \text{(implicite)}$$

$$a^2 + b^2 = 1 \text{ (normalisée)}$$

- Représentation paramétrique

$$x(t) = r * \cos(u) + x_0$$
$$y(t) = r * \sin(u) + y_0$$

- Représentation vectorielle

(O,r)
ou
(P_1,P_2,P_3)

Représentations du cercle

- Représentation algébrique

$a*(x-x_0)^2 + b*(y - y_0)^2 +- r^2 = 0$ (implicite)

$$P \in C(O,r) \Leftrightarrow \left\| \overrightarrow{OP} \right\| = r$$

- Représentation paramétrique

$x(t) = r * \cos(u) + x_0$
$y(t) = r * \sin(u) + y_0$

$$P(u) = O + r\vec{V}(u)$$

- Représentation vectorielle

(O,r)

ou

(P_1,P_2,P_3)

Pas de courbe polynomiale décrivant un cercle !
Il faut des courbes rationnelles (NURBS)
La seule conique polynomiale est la parabole

Problèmes élémentaires sur les droites

1. Distance *d* d'un point *P(x,y)* à une droite ?

2. Position d'un point *P(x,y)* par rapport au ½ espace limité par une droite ?

3. Intersection de 2 droites ?

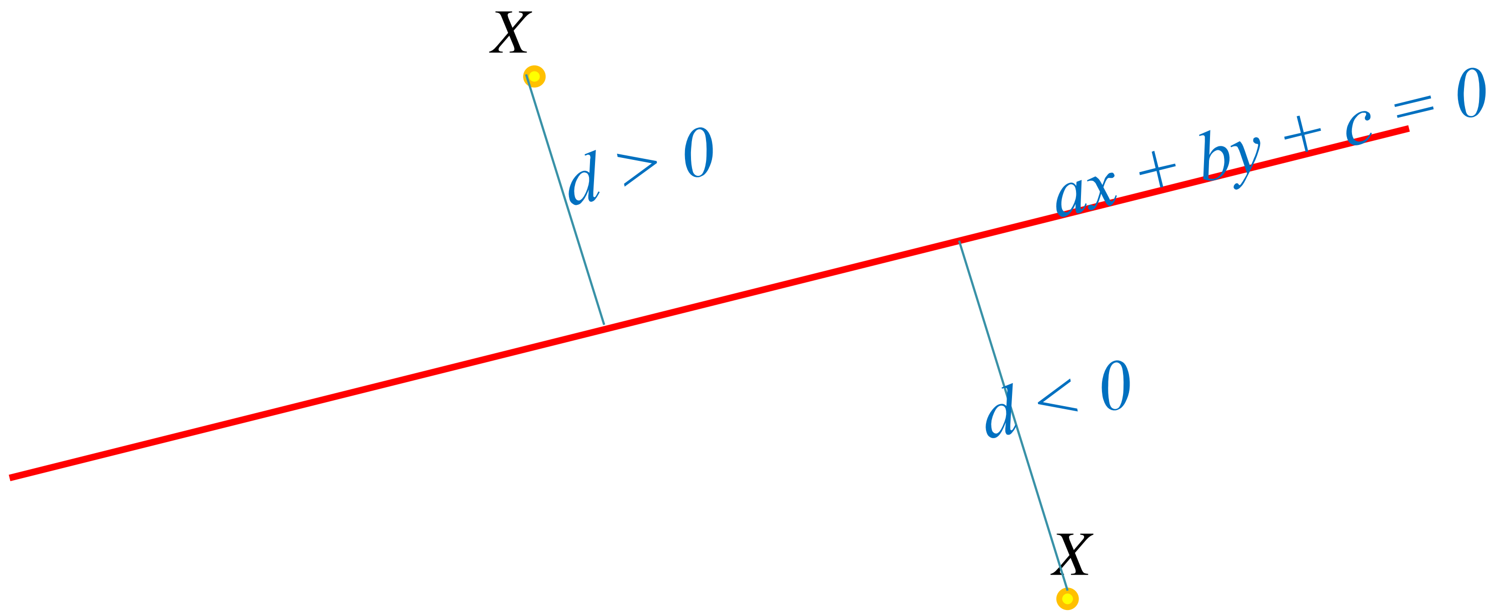
4. « « d'une droite et d'un segment ?

Position et distance d'un Point/Droite

Avec la représentation algébrique ?

$X(\mathbf{x},\mathbf{y})$

$ax + by + c = d$

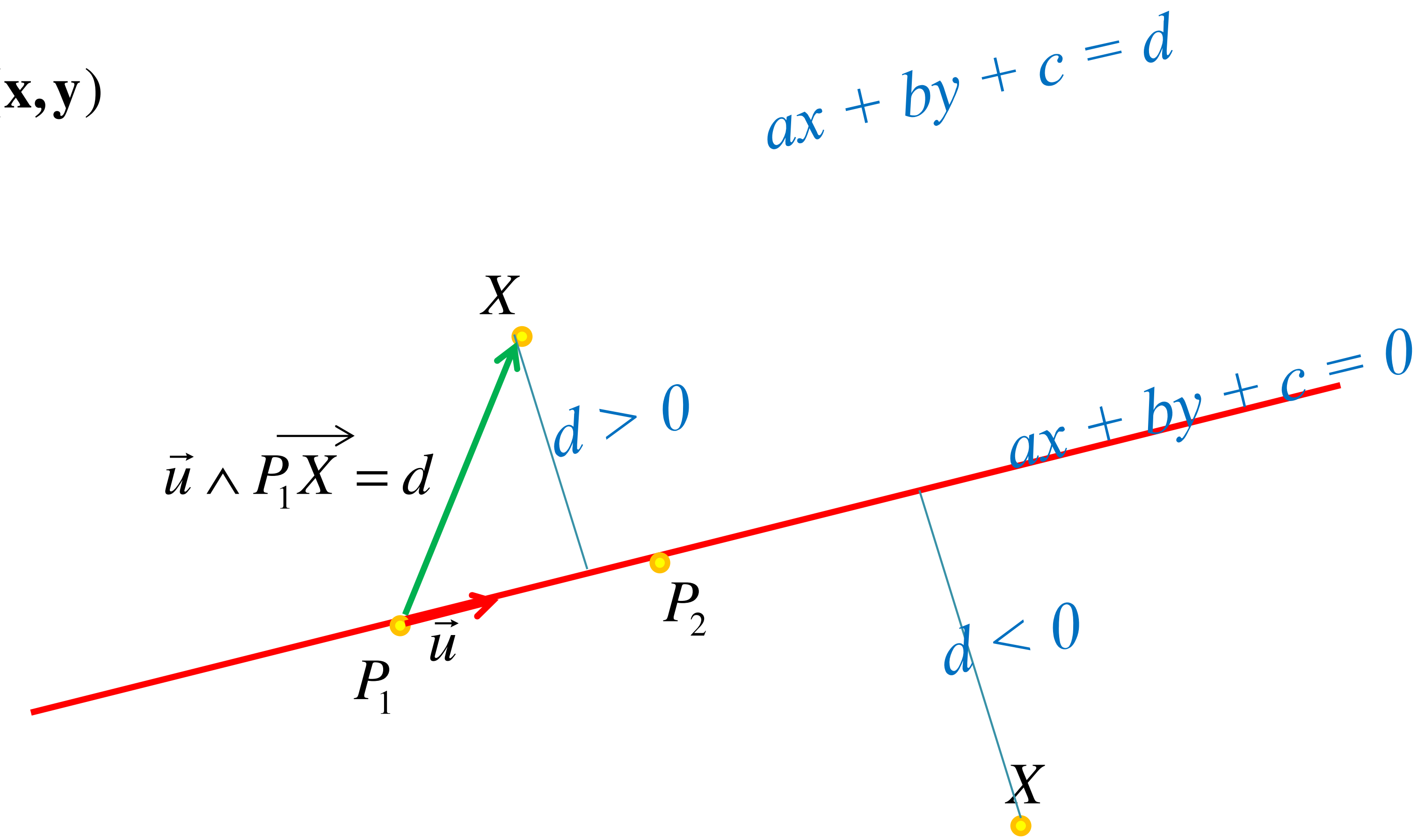


Avec $a^2 + b^2 = 1$

Position et distance d'un Point/Droite

Avec la représentation vectorielle ?

$X(\mathbf{x},\mathbf{y})$



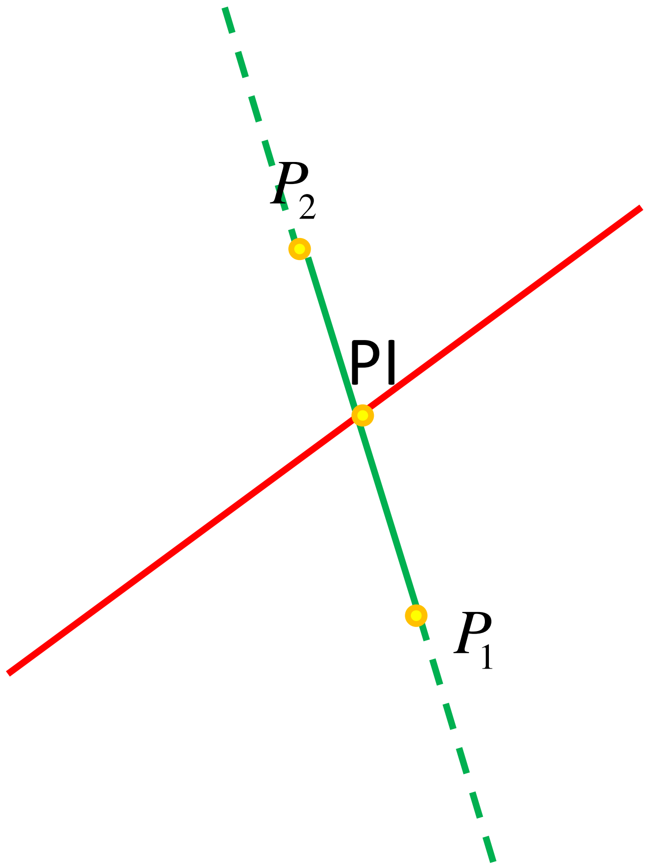
$\overrightarrow{P_1P_2} \wedge \overrightarrow{P_1X} = 0$ ➔ ax+ by +c = 0

Avec $a^2 + b^2 = 1$

Point d'intersection d'un segment avec une droite

Droite : $Ax+By+C=0$

Droite : $x(t)=a_1u+b_1$
$y(t)=a_2u+b_2$



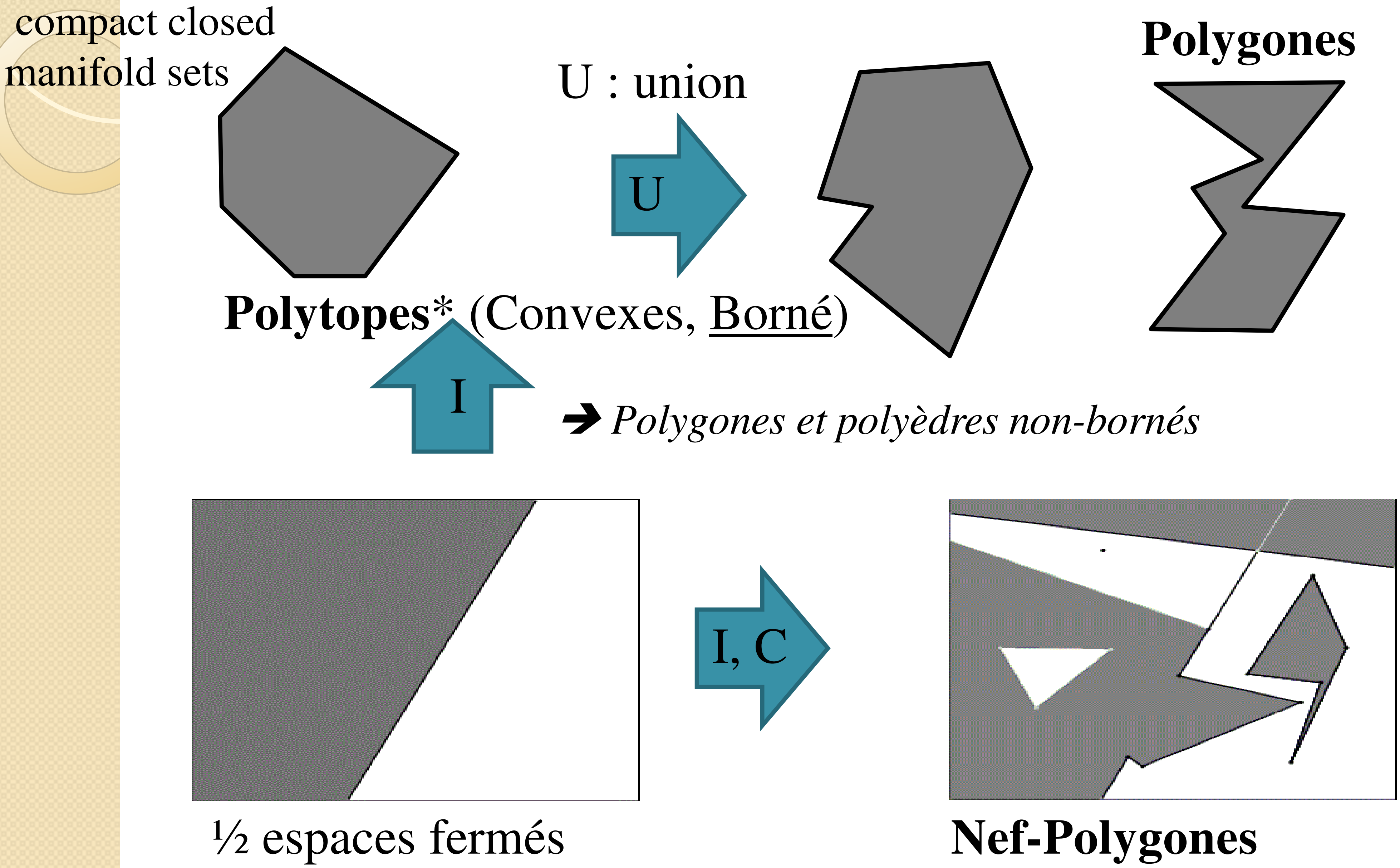
équation linéaire :

$$u_i = -\frac{(Ab_1+Bb_2+C)}{Aa_1+Ba_2} \Longrightarrow PI\,(x(u_i),\, y(u_i))$$

Que dire si : $u_i > 1$ ou $u_i < 0$?

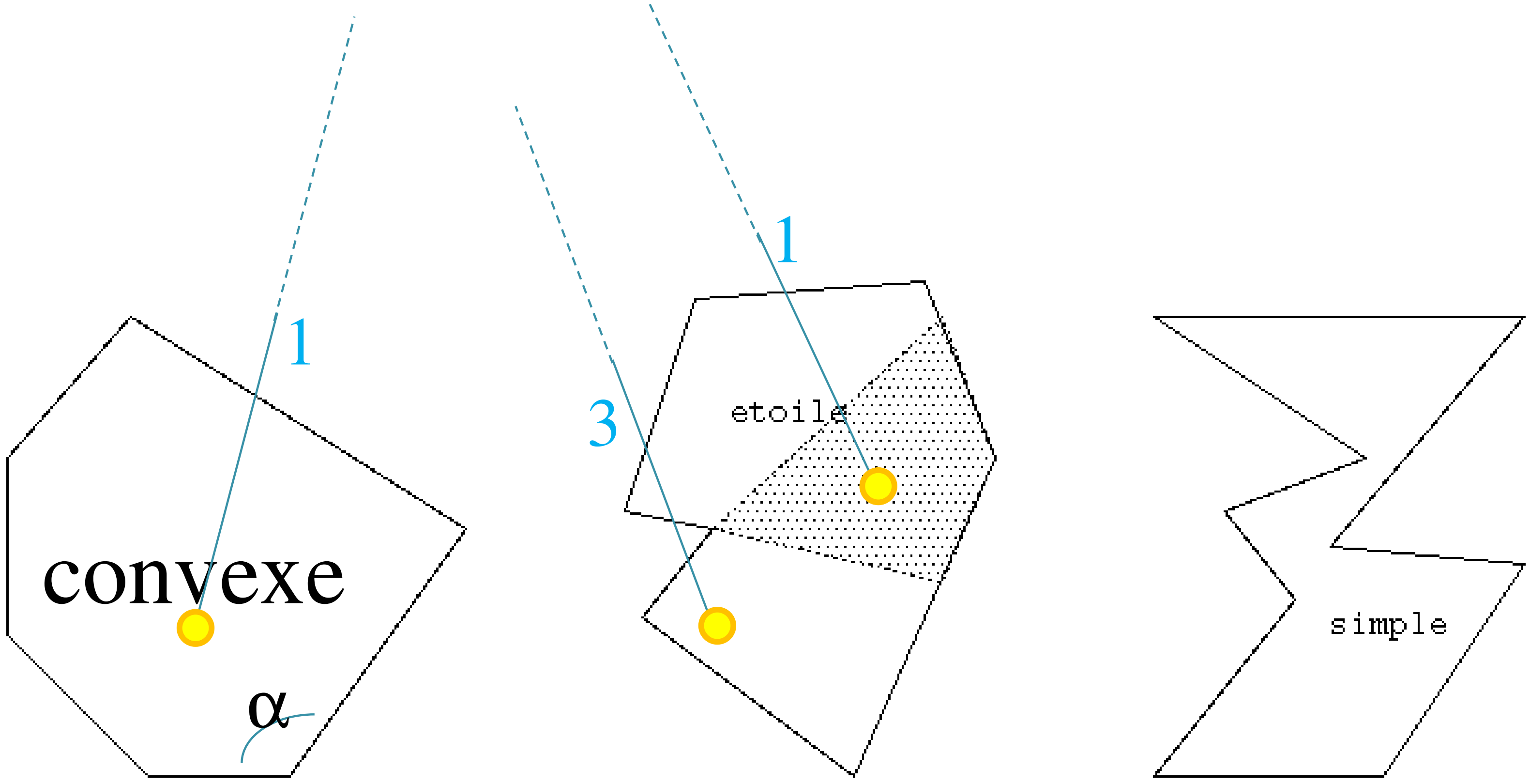
Et pour le cercle ?

1.3. Polygones, Polyèdres



I,C : Intersection et Complément de ½ espaces ouverts

Convexité (géométrie)



α <= 180°

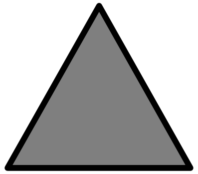
étoilé

simple

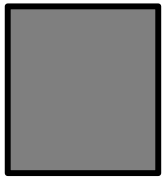
Polygone ou polyèdres

Polygones simples et angles

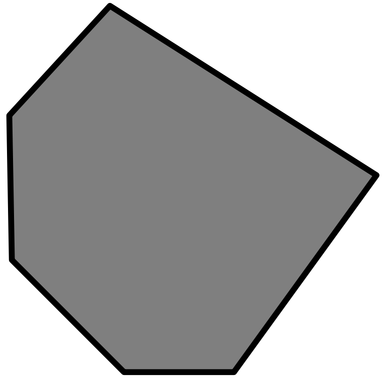
Somme des angles internes = (n-2)*180°



$(3-2)*180° = 3*60°$

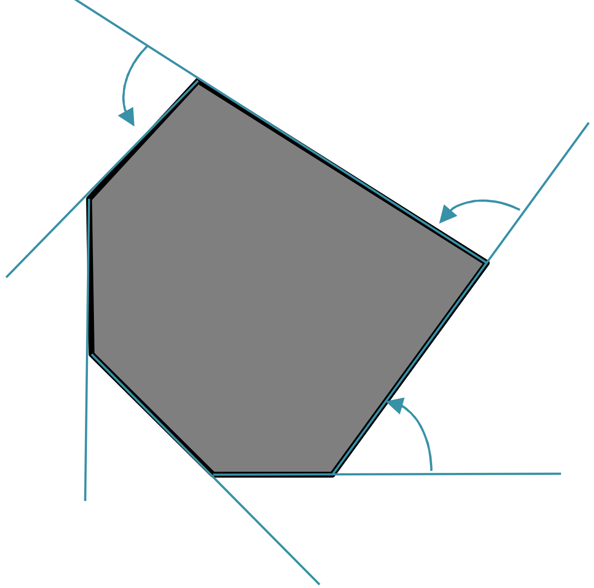


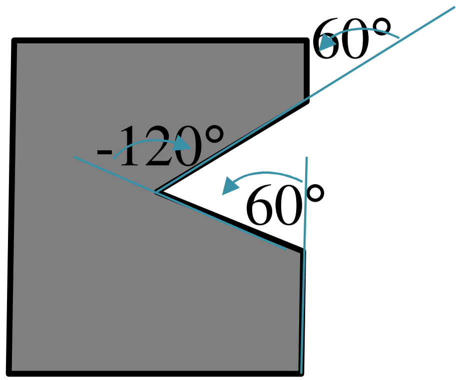
$(4-2)*180° = 4*90°$



$(6-2)*180° = 720°$

Somme des angles externes = 360°

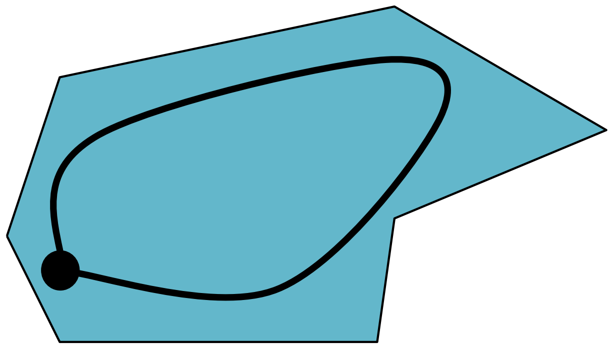


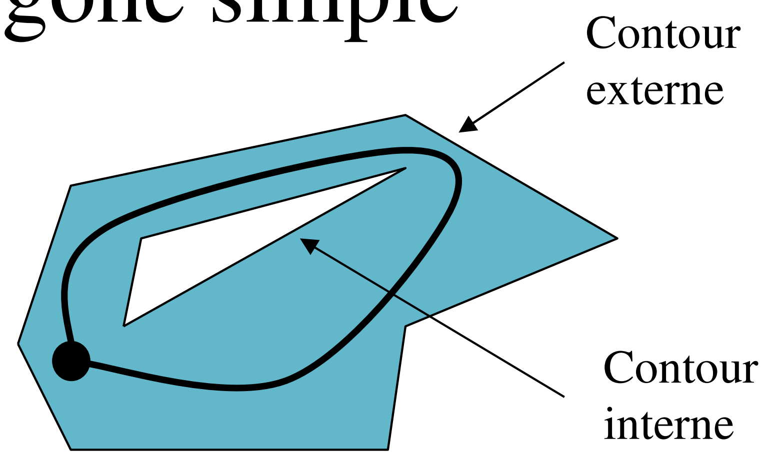


Connexité (topologie)

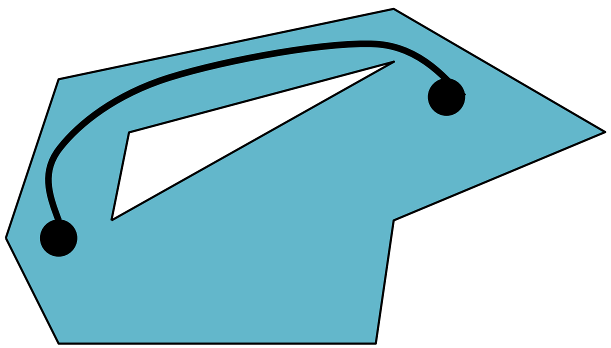
lacets non-contractiles

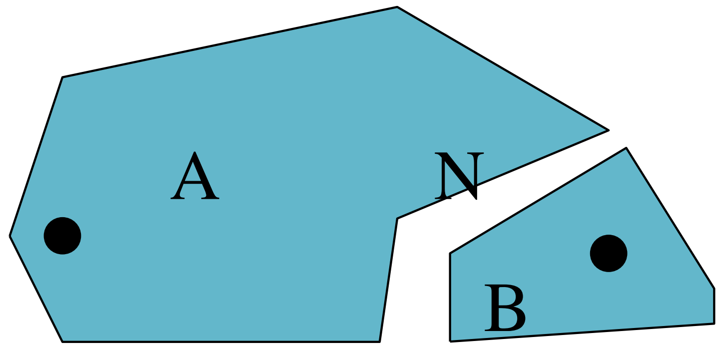
- Connexité simple - polygone simple





- Connexité par arc - composante connexe

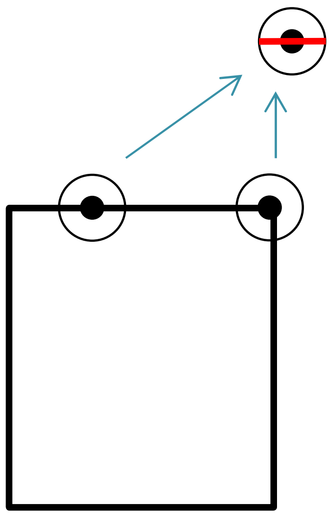




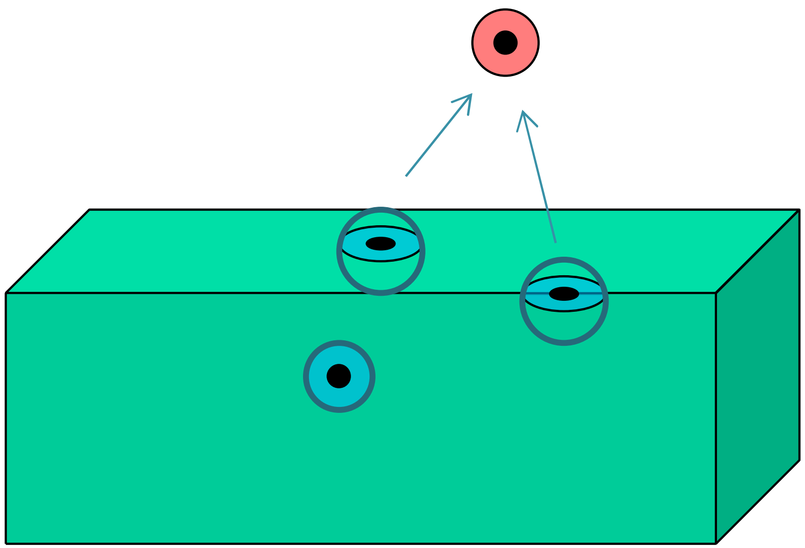
N non connexe ⇔ Existe A,B séparés, N=AuB
Si A et B fermés, A,B séparés ⇔ AnB=0

Variétés (manifold)

- Une variété est de dimension 1 ssi le voisinage d'un point est ***homéomorphe*** à un segment
- Une variété est de dimension 2 ssi le voisinage d'un point est ***homéomorphe*** à un disque



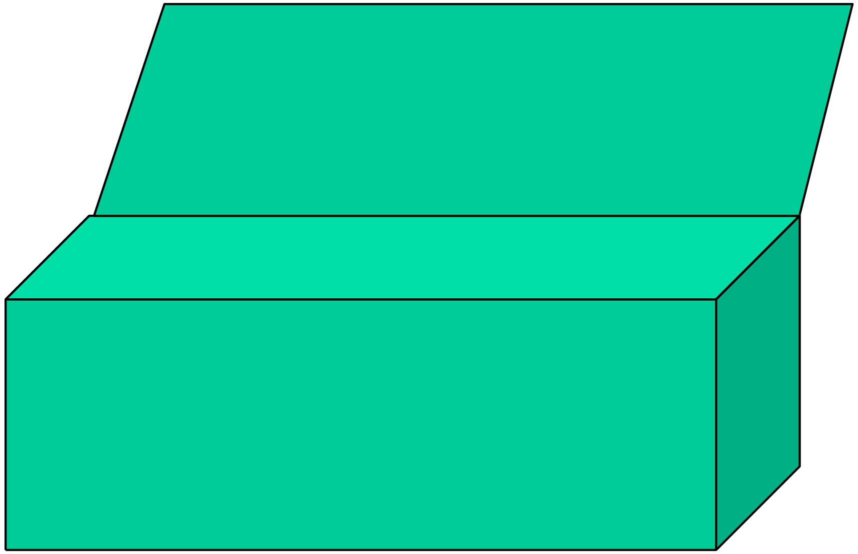
2D : frontière 1-manifold



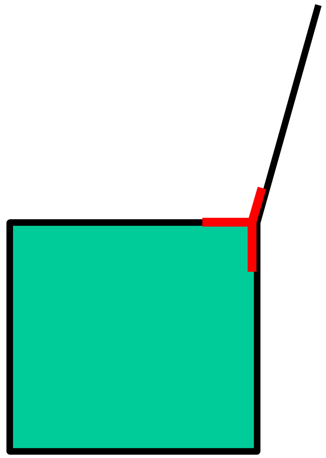
3D : frontière 2-manifold

Question : Frontières “manifold” ?

La frontière de ces objets est-elle une varieté de dimension D-1 ?



3D

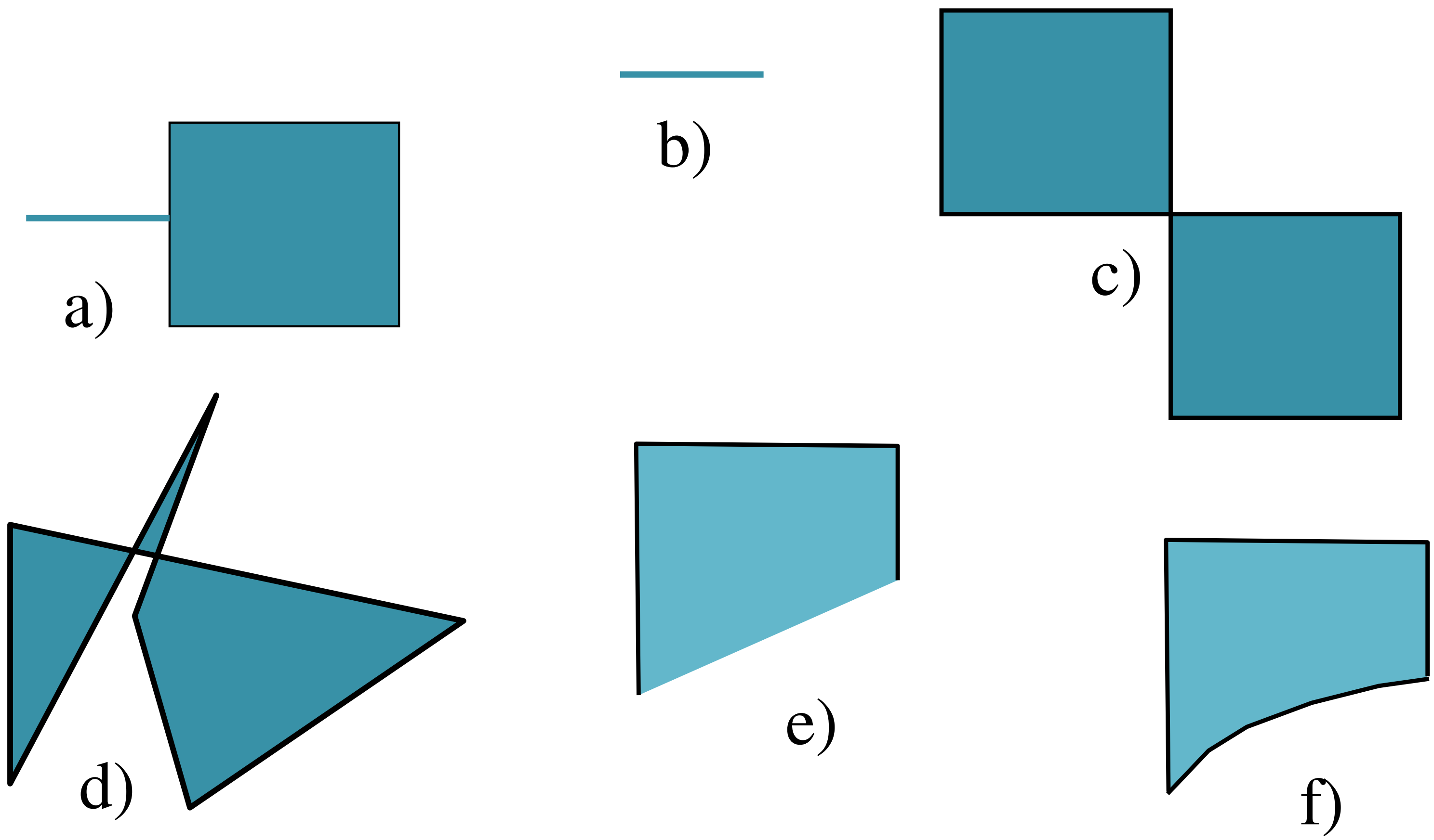


2D

Trouvez un objet 2D régulier mais dont la frontière n’est pas une varieté de dimension 1 (1-manifold) ?

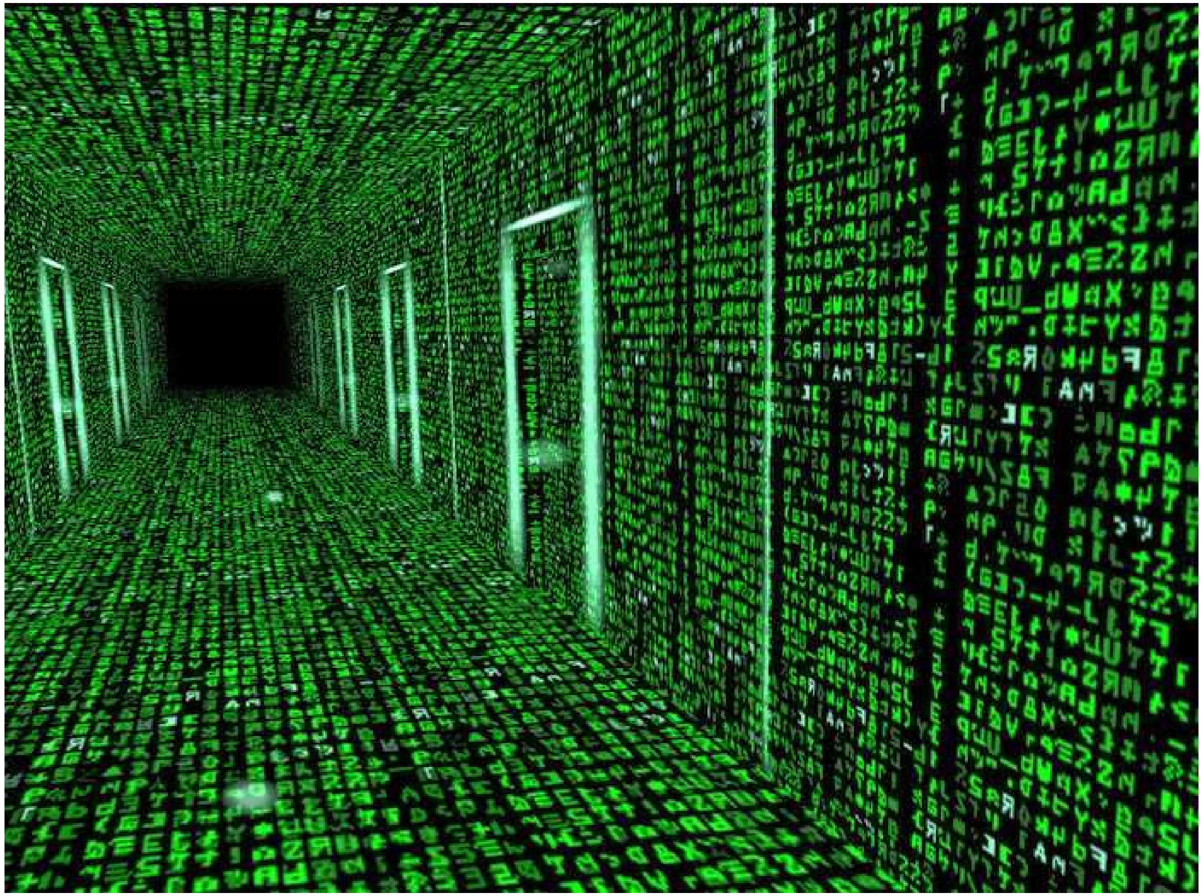
Questions : polygones « valides »

Donnez les caractéristiques des polygones suivant.
Lesquels ne sont pas valide et pourquoi ?



Ensembles régulier : A = Adh(Int(A))

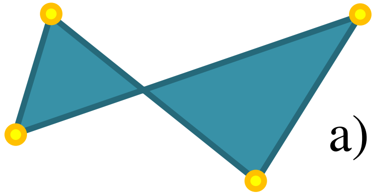
2. Représentation informatique



Contraintes d 'intégrité

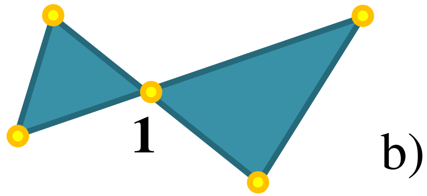
Ensembles régulier : A = Adh(Int(A)), Polygones

La frontière doit être une variété de dimension 1 (1-manifold)



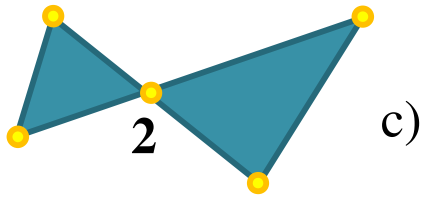
- P1: Intersection réduites aux sommets communs

Pas de polygones repliés

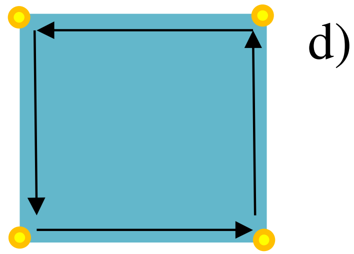


- P2 : Un sommet appartient exactement à 2 arêtes

Polygones régulier ***à frontière 1-manifold***



- P3 : Il existe une orientation des arêtes telle que chaque sommet est origine d'une arête et extrémité d'une autre.



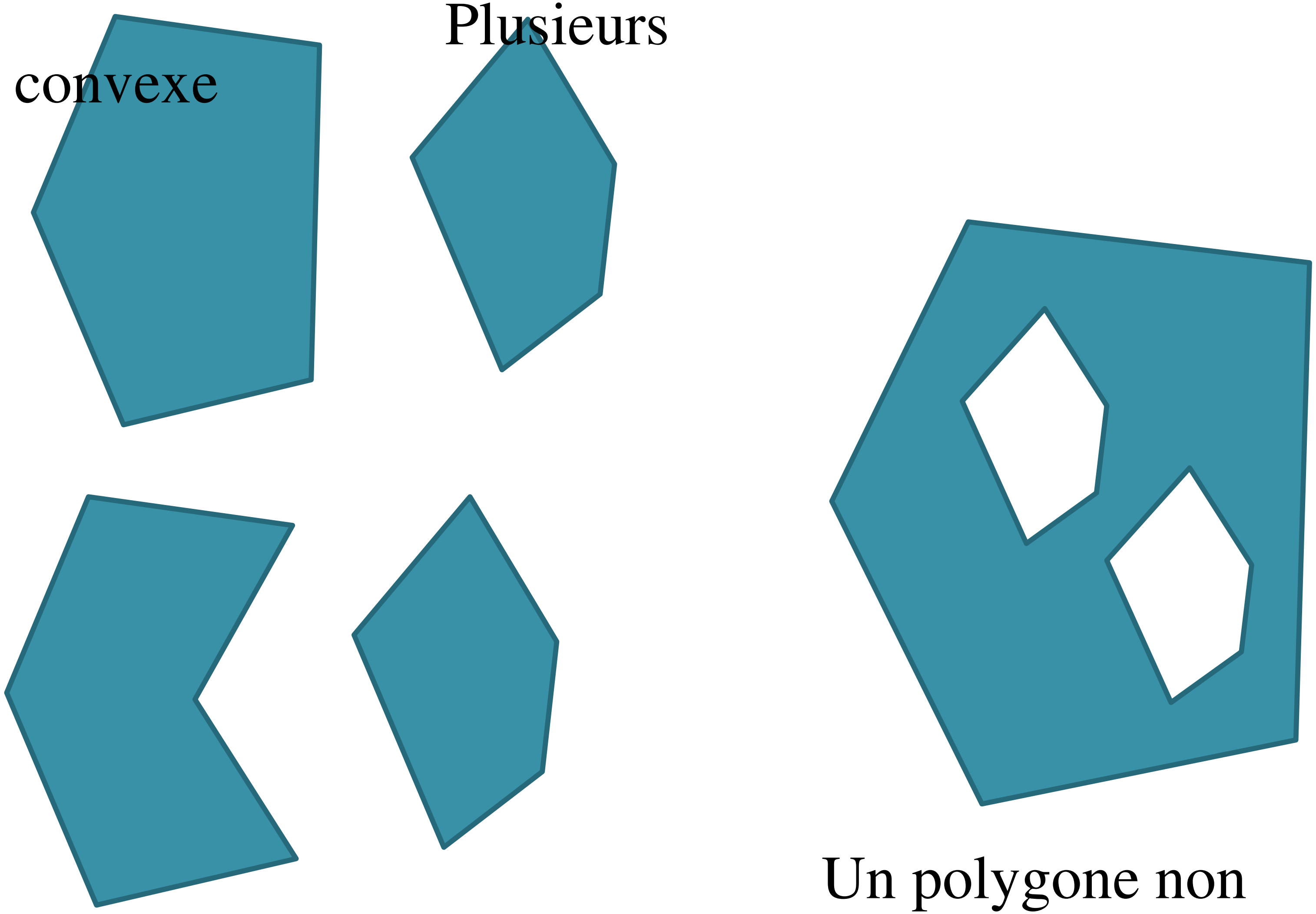
Notons que les PP ➔ l'intérieur et l'extérieur sont connus au niveau d'une arête : l'orientation est dite « clockwise » or « counter clockwise » en fonction de la convention

Structure nécessaire ?

(pour décrire un polygone dans les différents cas)

Un convexe

Plusieurs



Un polygone simplement connexe

Un polygone non simplement connexe

Description des polygones

Fichiers de données (utilisateurs) pour PovRay

Polygon { nb_points

$< x_{11},y_{11} >$, … , $<, x_{1i},y_{1i} >$ … $< x_{11},y_{11}>$,

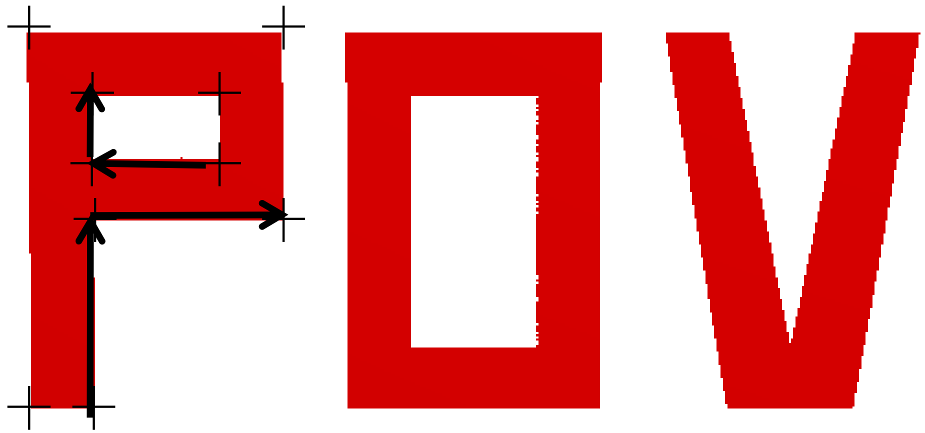
$< x_{21},y_{21} >$, … , $<, x_{2i},y_{2i} >$ … $< x_{21},y_{21}>$,

$< x_{j1},y_{j1} >$, … , $<, x_{ji},y_{ji} >$ … $< x_{j1},y_{j1}>$

}

Scilab :

xfpolys(xpols,ypols[,fill])



Comment sait-on que l'on est sur le bord d'un trou ?

Comment peut-on connaitre le contour extérieur à partir d'un trou ?

➔ Des algorithmes !

Représentation informatique

Structure simple avec des tableaux

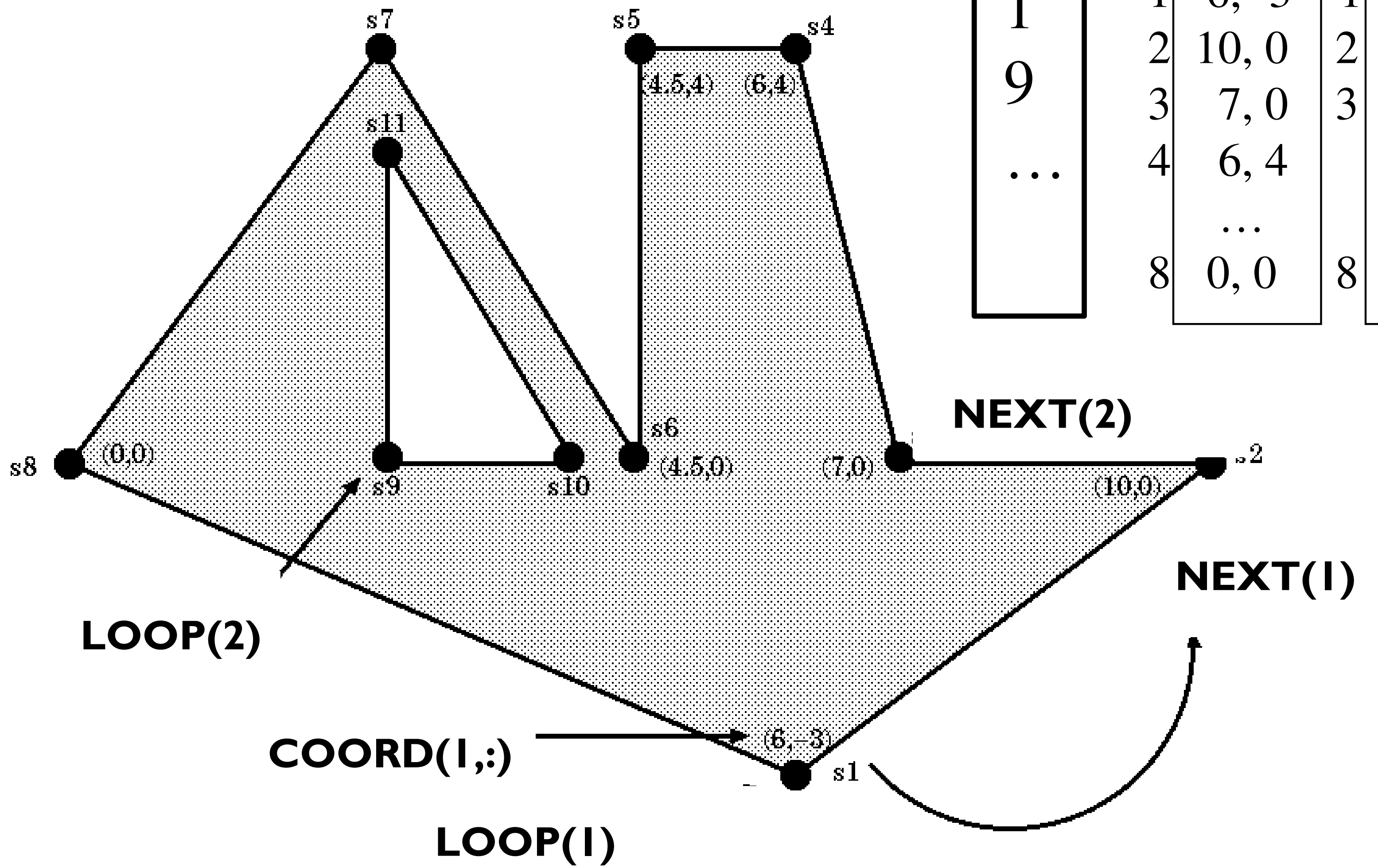
COORD= [6, -3; 10, 0; 7, 0; …];

EDGES=[1,2,3,4…,1]

LOOP
1
9
…

	COORD
1	6, -3
2	10, 0
3	7, 0
4	6, 4
	…
8	0, 0

	NEXT
1	2
2	3
3	4
	…
	…
8	1

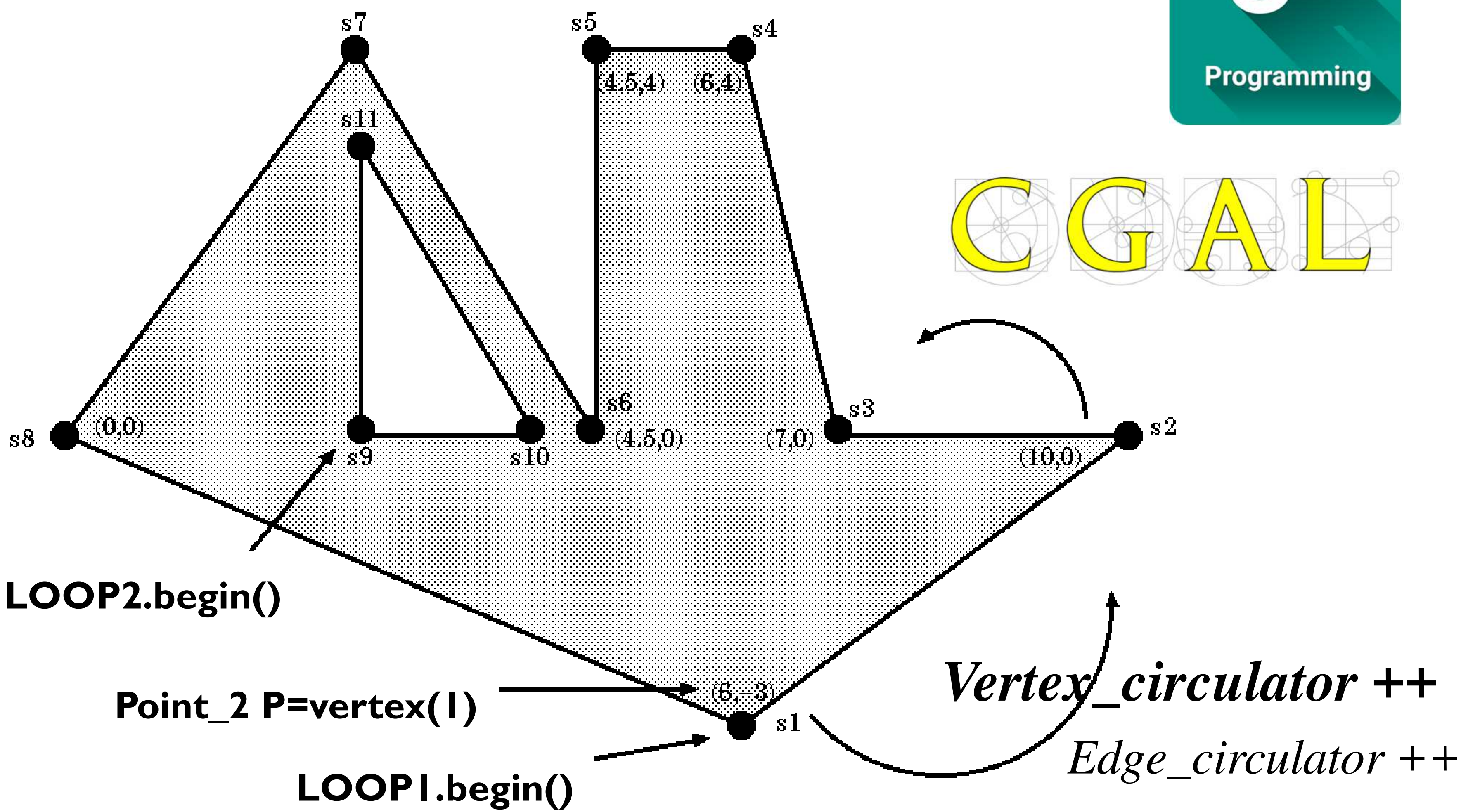


Représentation informatique

Structure C++ CGAL

CGAL::Polygon_2< **Point_2<Kernel>**, Container >

CGAL::Polygon_with_holes_2< Kernel, Container >



3. Interrogation du modèle
(questions élémentaires)

CGAL (polygon package)

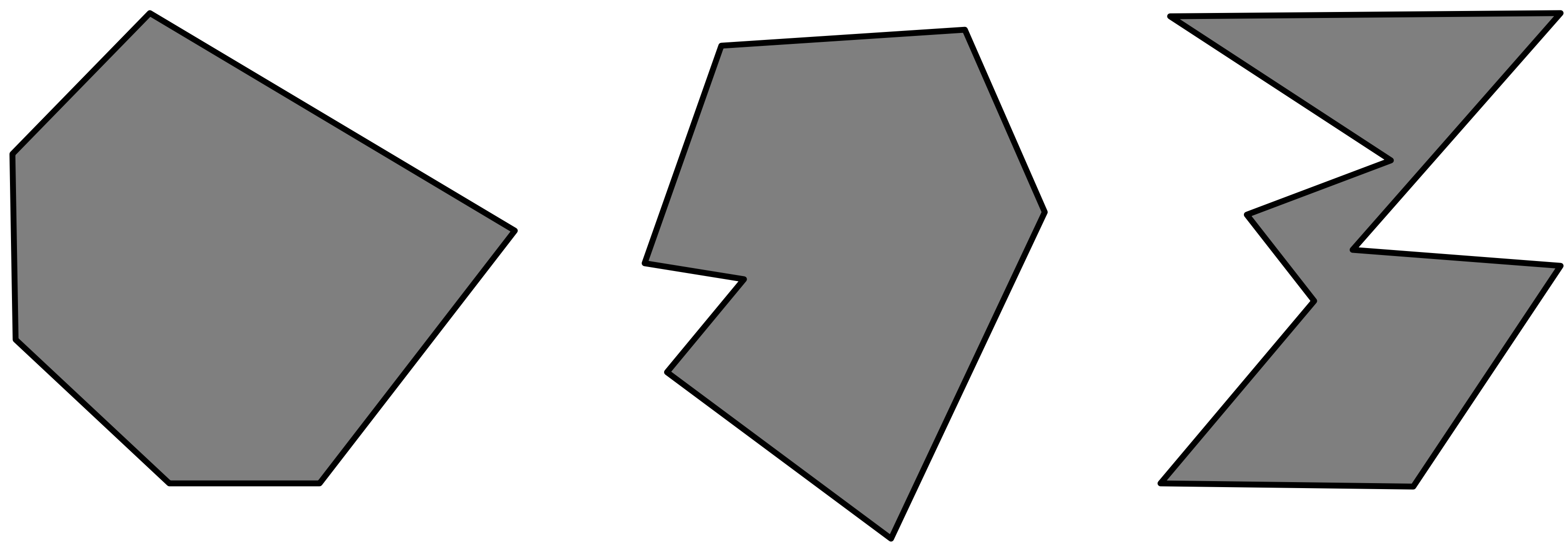
The following algorithms are available:

- find the leftmost, rightmost, topmost and bottommost vertex.
- compute the (signed) area. **Algo ?**
- check if a polygon is simple.
- check if a polygon is convex.
- find the orientation (clockwise or counterclockwise)
- check if a point lies inside a polygon. **Algo ?**

http://doc.cgal.org/latest/Polygon/classCGAL_1_1Polygon__2.html

Exercice : programmer l'algorithme is_convex()

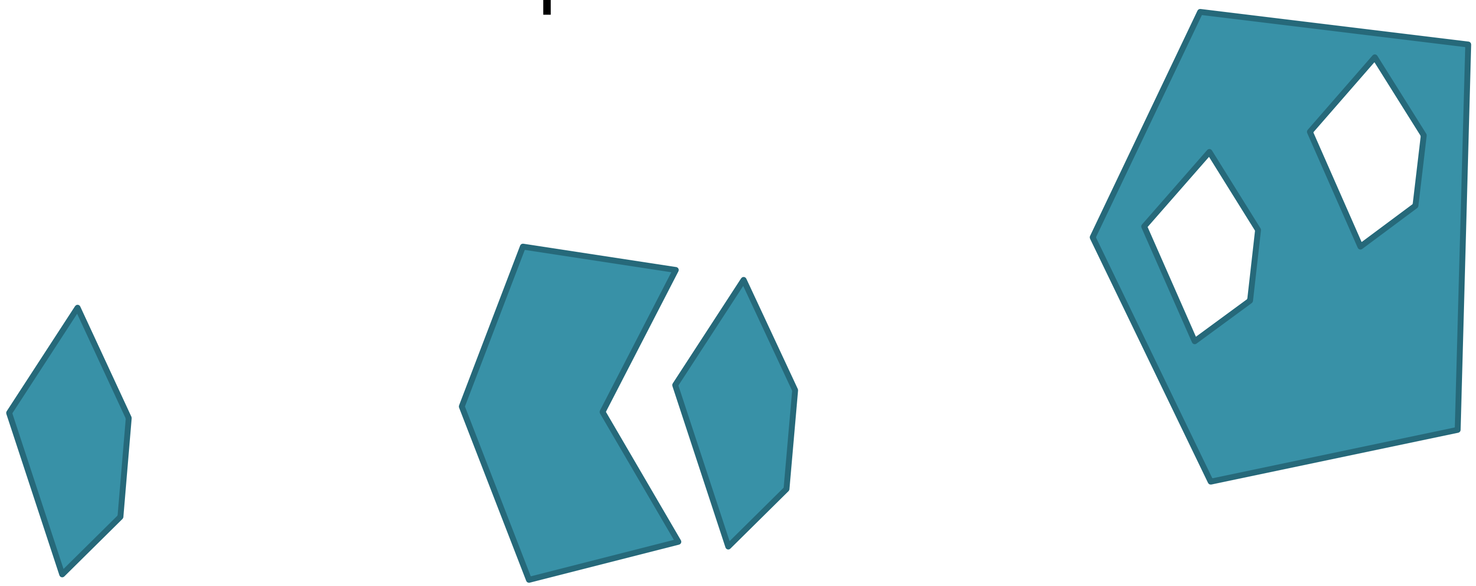
Ecrire un algorithme qui détermine si un polygone simple est convexe (sur une des structures de données présentées) ?



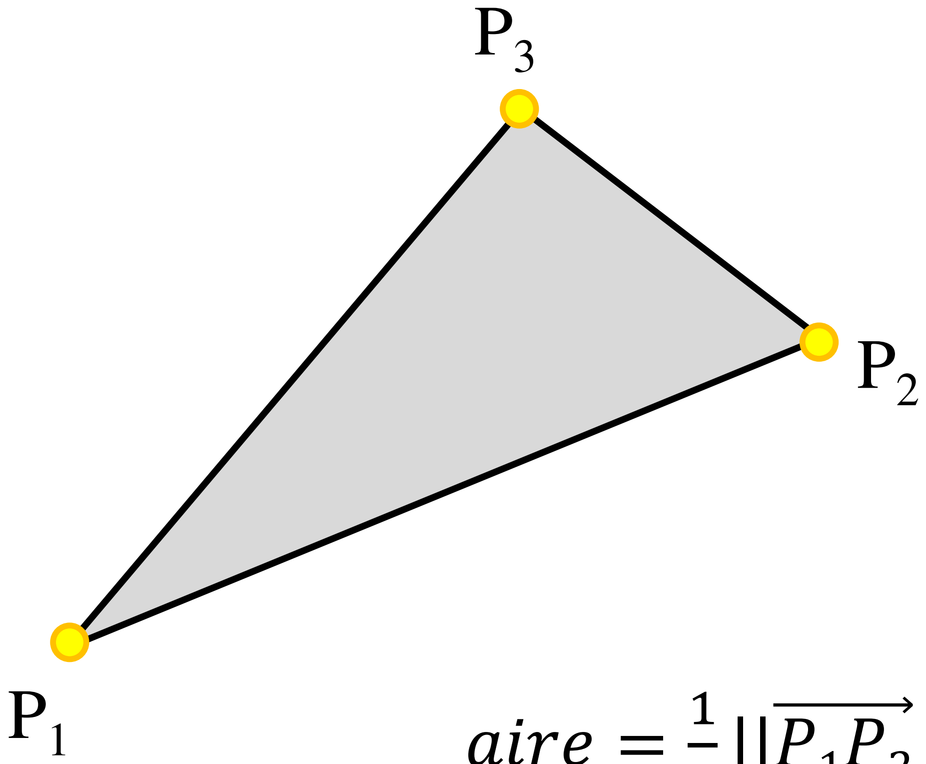
Exercice : programmer l'algorithme is_connex()

Ecrire un algorithme qui détermine si un des polygones représentés (par une des structures de données) est simplement connexe?

Idem sur une simple liste d'arêtes ?

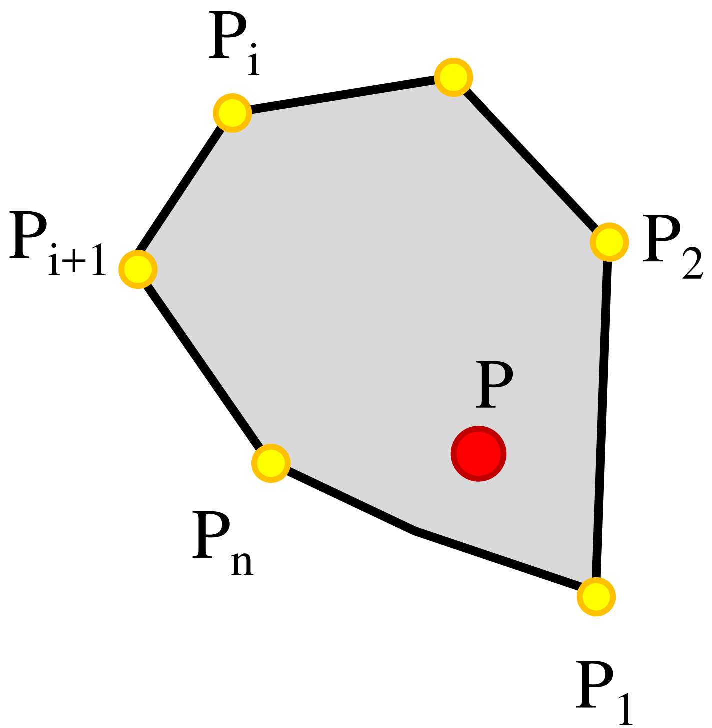


L'aire d'un triangle ?



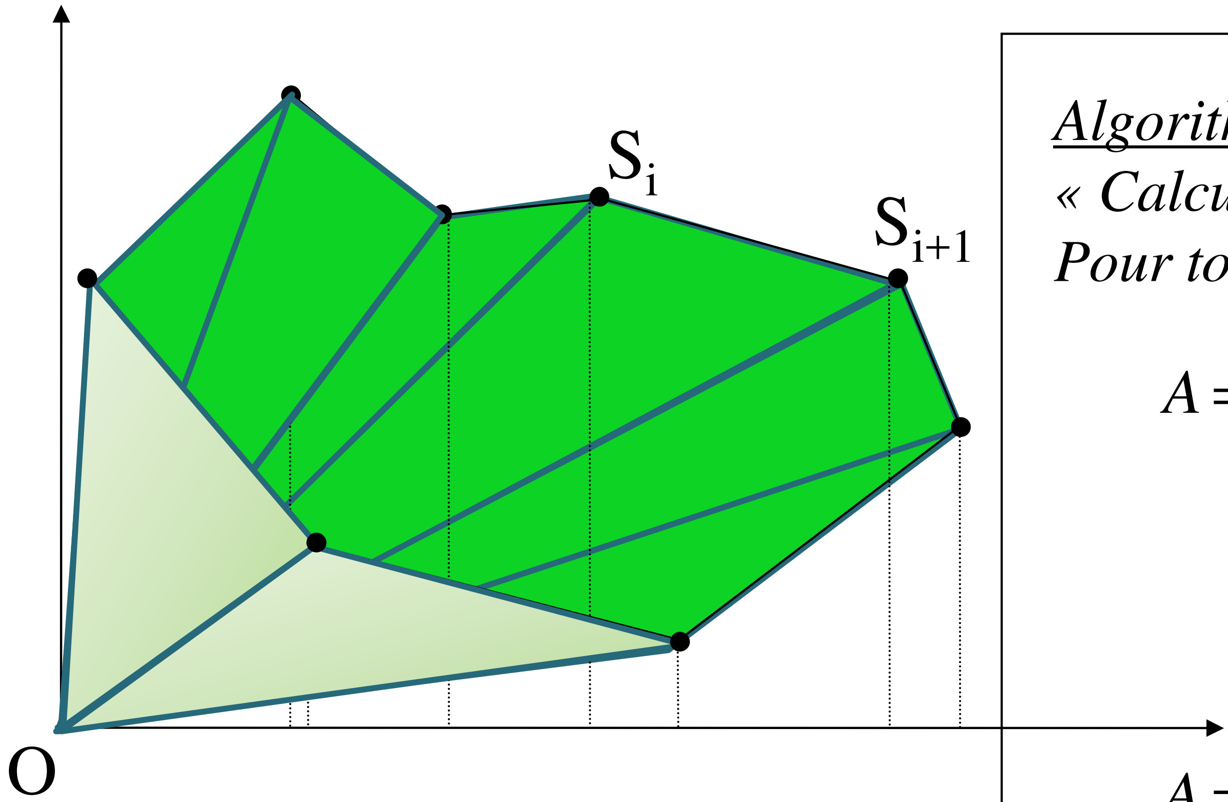
$$aire = \frac{1}{2}||\overrightarrow{P_1P_2} \wedge \overrightarrow{P_2P_3}||$$

L'Aire d'un convexe ?



Calcul de l 'aire

Approche pragmatique (vectorielle)



Algorithme :

« Calcul » du point O, A=0

Pour toutes les sommets du poly

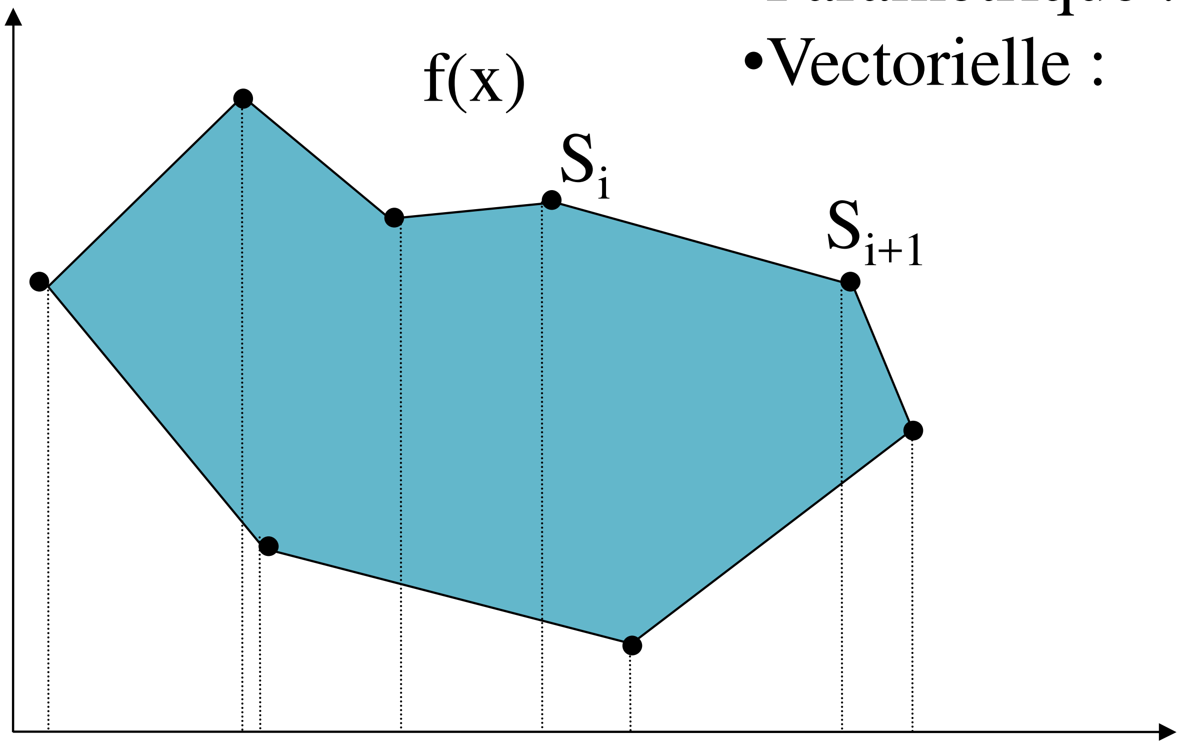
$$A = A + \frac{1}{2}\overrightarrow{OS_i}\Lambda\overrightarrow{S_iS_{i+1}}$$

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)$$

Calcul de l 'aire

Sur une représentation :

- Algébrique : f(x) = y = A x + B
- Paramétrique : $x_i(t)$, $y_i(t)$
- Vectorielle : $\overrightarrow{S_i S_{i+1}}$

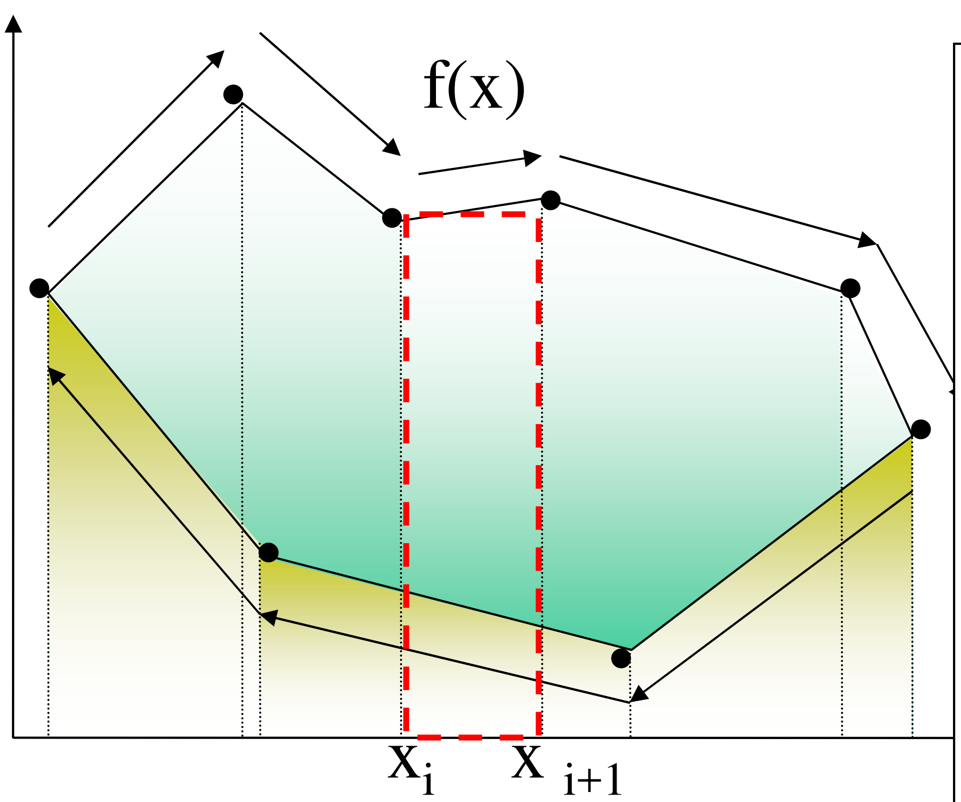


Skip

Calcul de l 'aire

Approche triviale : représentation algébrique explicite : y = A x + B

$$\sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)$$



<u>Exemple cas linéaire</u> :

$f(x) = (x-x_i)(y_{i+1}-y_i)/(x_{i+1}-x_i) + y_i$

Ou encore

$f(x) = Ax+B$

avec $A = (y_{i+1}-y_i)/(x_{i+1}-x_i)$

et $B = (x_{i+1}\, y_i - x_i y_{i+1})/(x_{i+1}-x_i)$

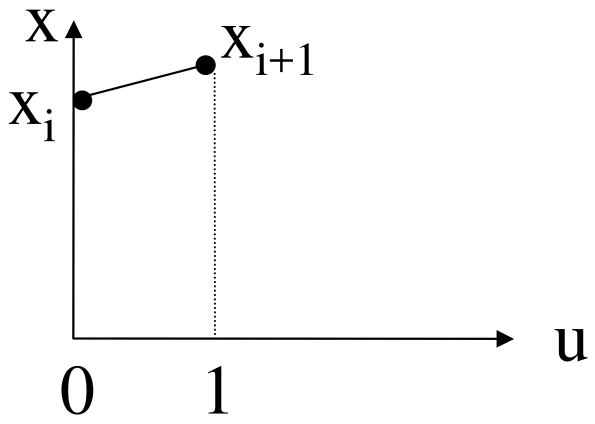
$$\int_{x_i}^{x_{i+1}} (Ax+B)dx =$$

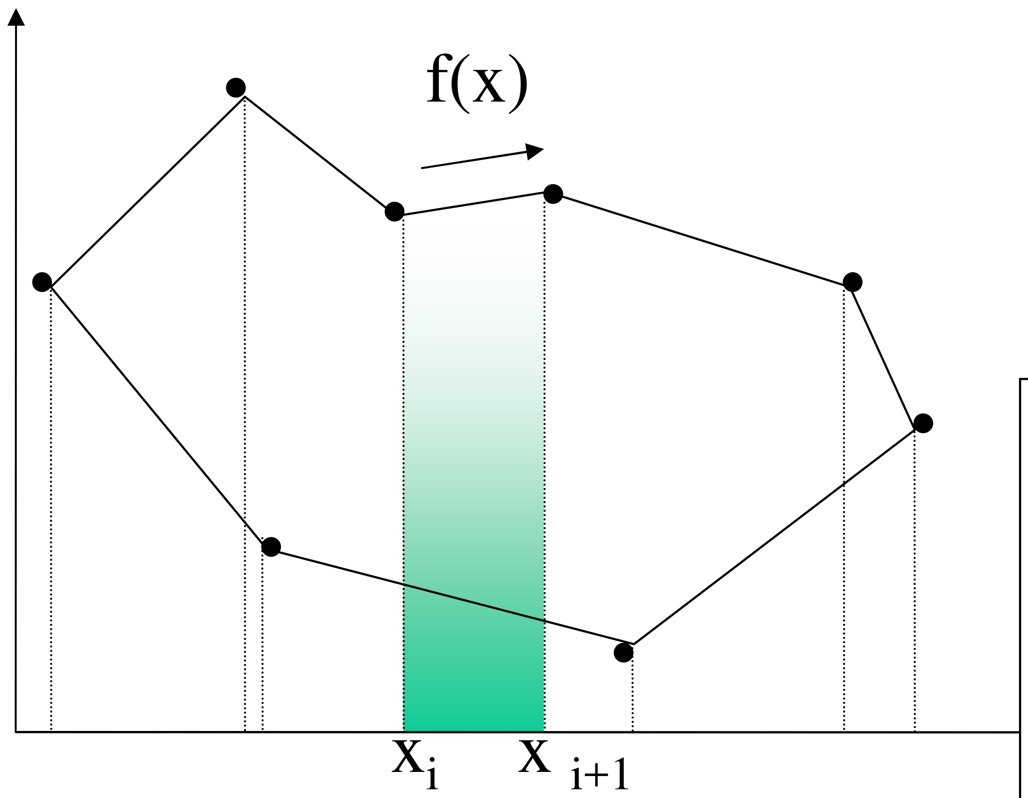
$$\boxed{\frac{1}{2}(y_{i+1}+y_i)(x_{i+1}-x_i)}$$

Calcul de l'Aire

Avec changement de variable : représentation paramétrique

$$\int_{x=x_i}^{x_{i+1}} f(x)dx = \int_{u=0}^{1} f(t(u))t'(u)du$$





x = t(u),   u dans [0:1]

Exemple cas linéaire :

$t(u) \quad = ux_{i+1} + (1\text{-}u)x_i$

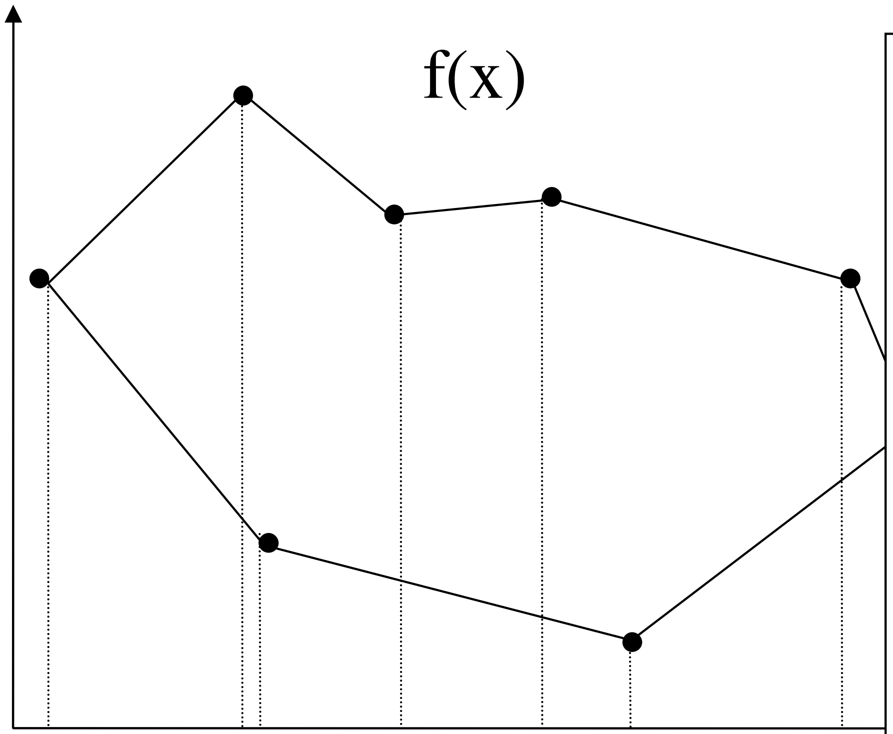
$f(t(u)) = uy_{i+1} + (1\text{-}u)y_i$

$$\int (uy_{i+1} + (1-u)y_i)(x_{i+1} - x_i)du =$$

$$\frac{1}{2}(y_{i+1} + y_i)(x_{i+1} - x_i)$$

Aire & Centre de gravité

Formules (rappels)



$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)$$

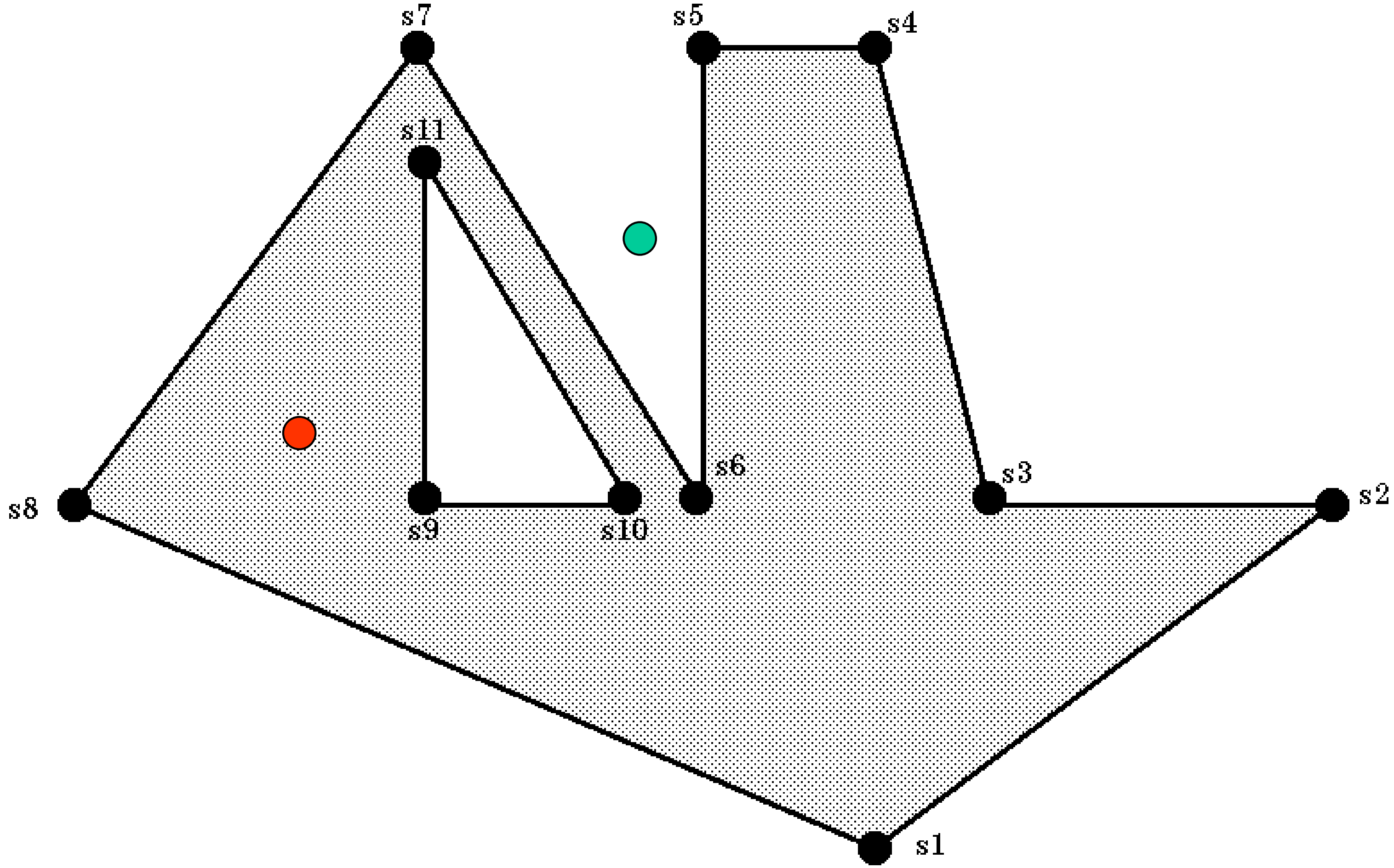
$$x_G = \frac{1}{6A}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$y_G = \frac{1}{6A}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

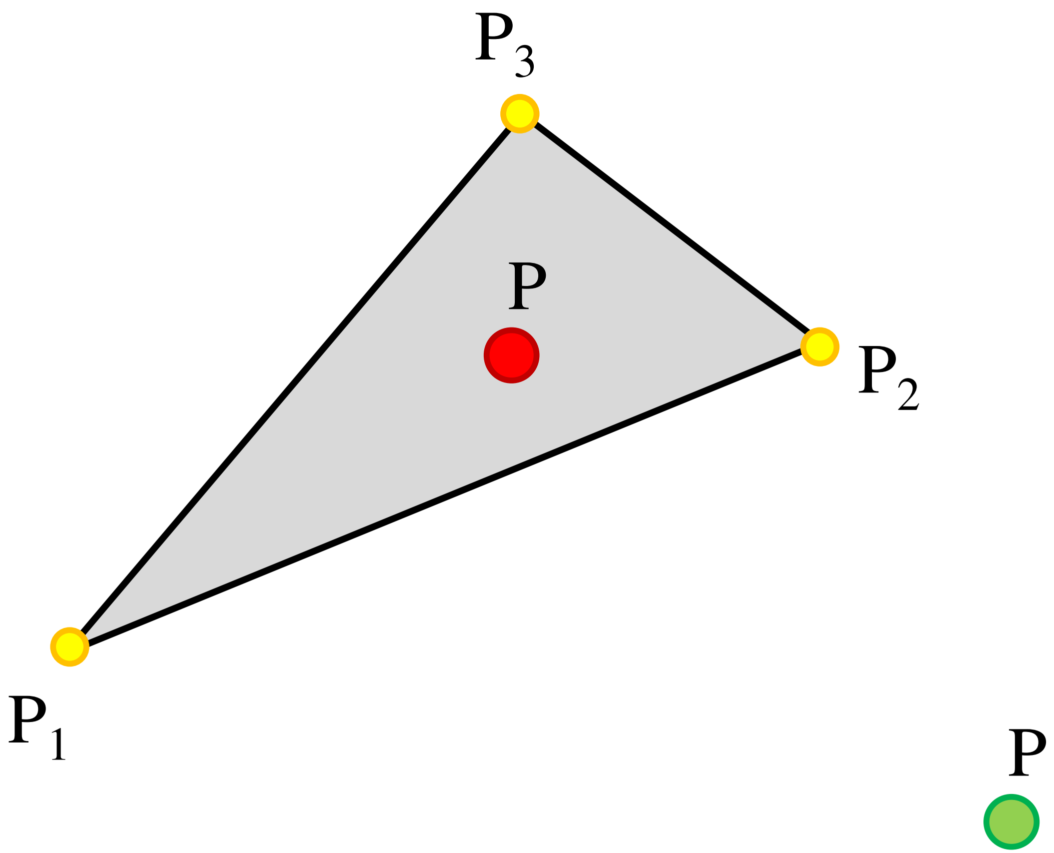
Exercice : algorithme de calcul de l'aire d'un polygone « courbe »

- Calcul de l'aire d'un polygone dont la frontière est décrite par des courbes :
- C.a.d. écrire l'intégrale correspondant à la représentation paramétrique de courbes (de Bezier d'ordre 3 par exemple).

Point dans polygone ?



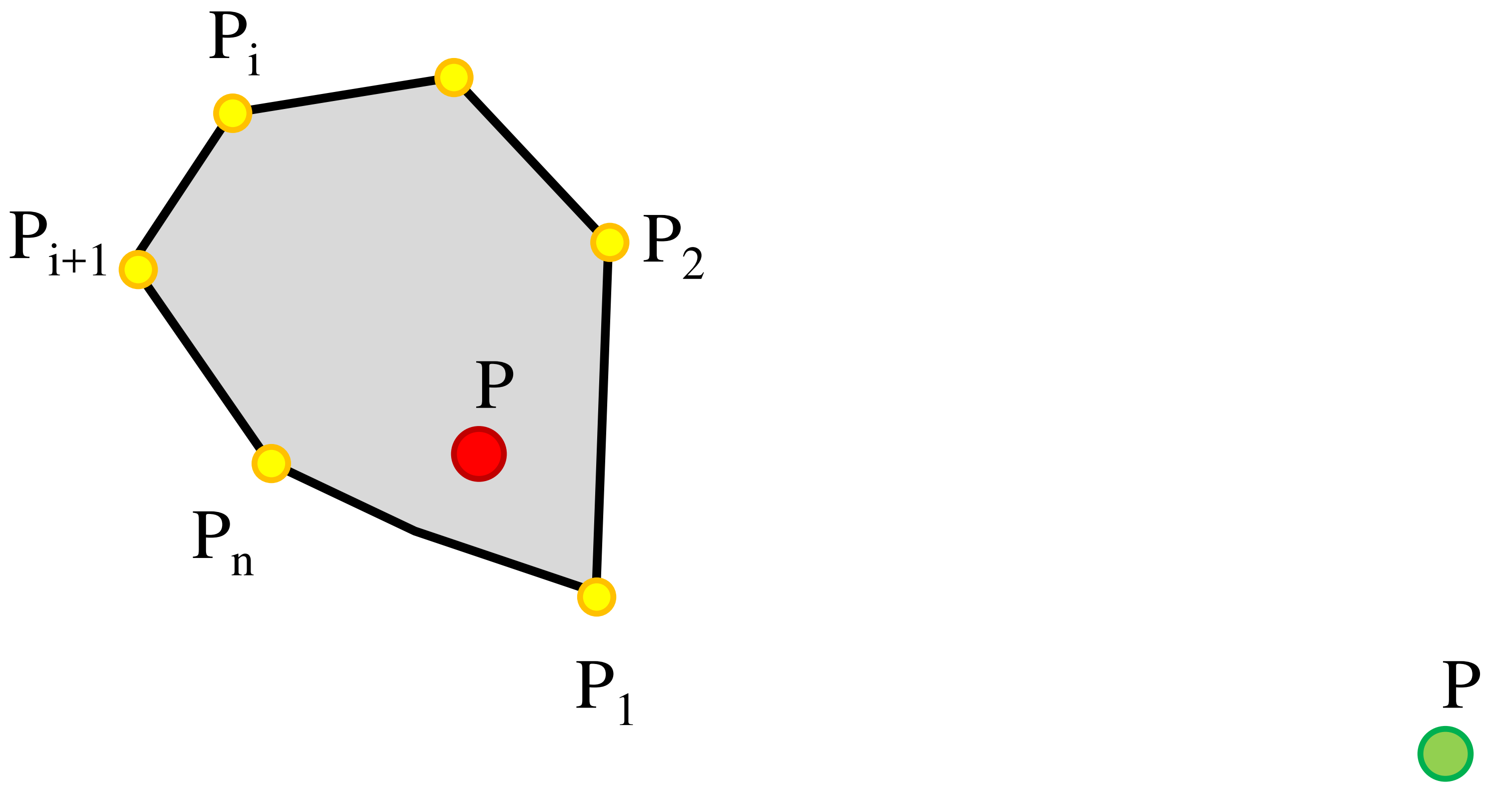
Point dans un triangle ?



$$sign(\overrightarrow{P_1P_2} \wedge \overrightarrow{P_2P}) = sign\left(\overrightarrow{P_2P_3} \wedge \overrightarrow{P_3P}\right) = sign(\overrightarrow{P_3P_1} \wedge \overrightarrow{P_3P})$$

Et si on ne connait pas l'orientation du triangle ?

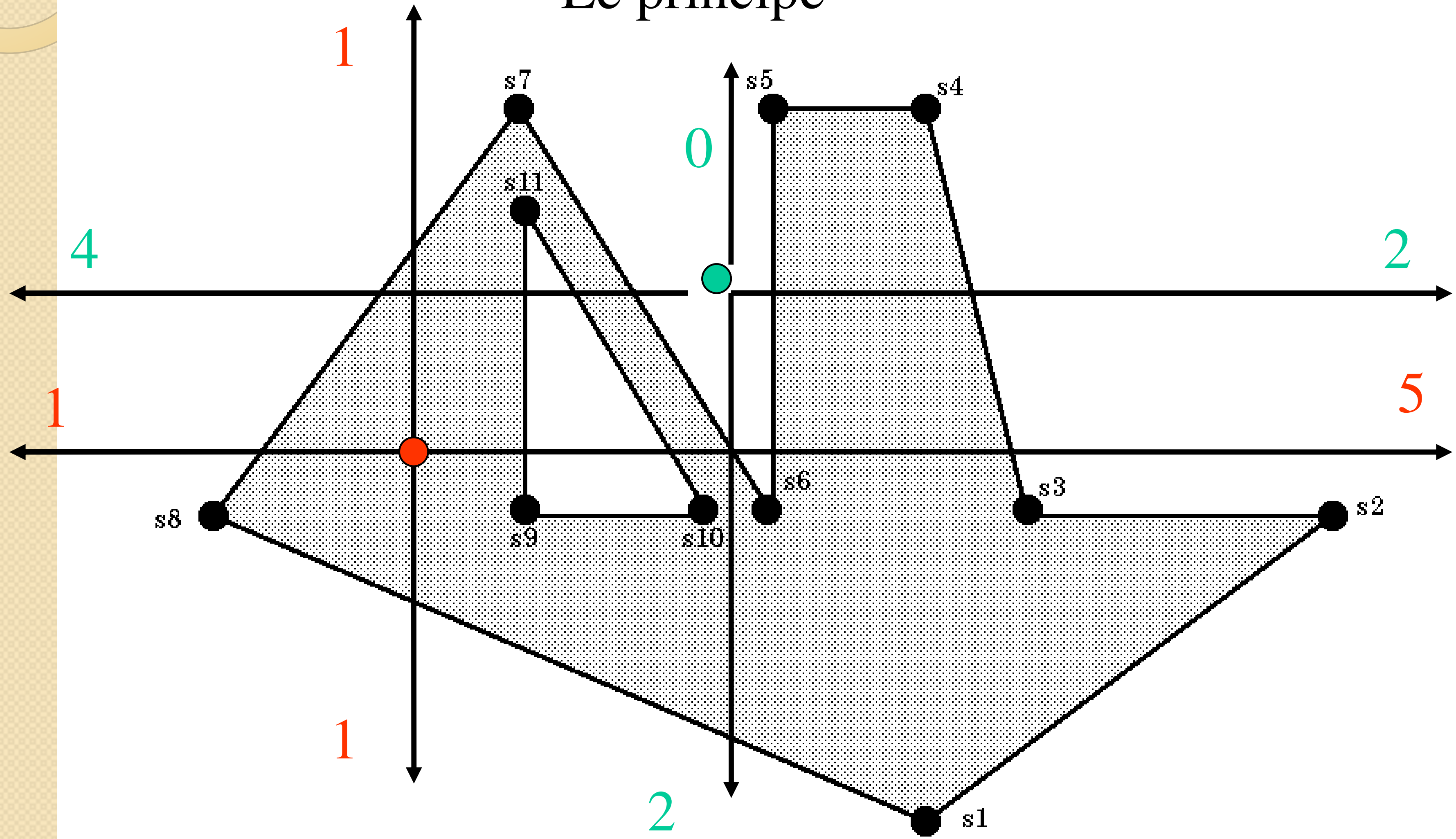
Point dans convexe ?



Complexité de l'algorithme ?

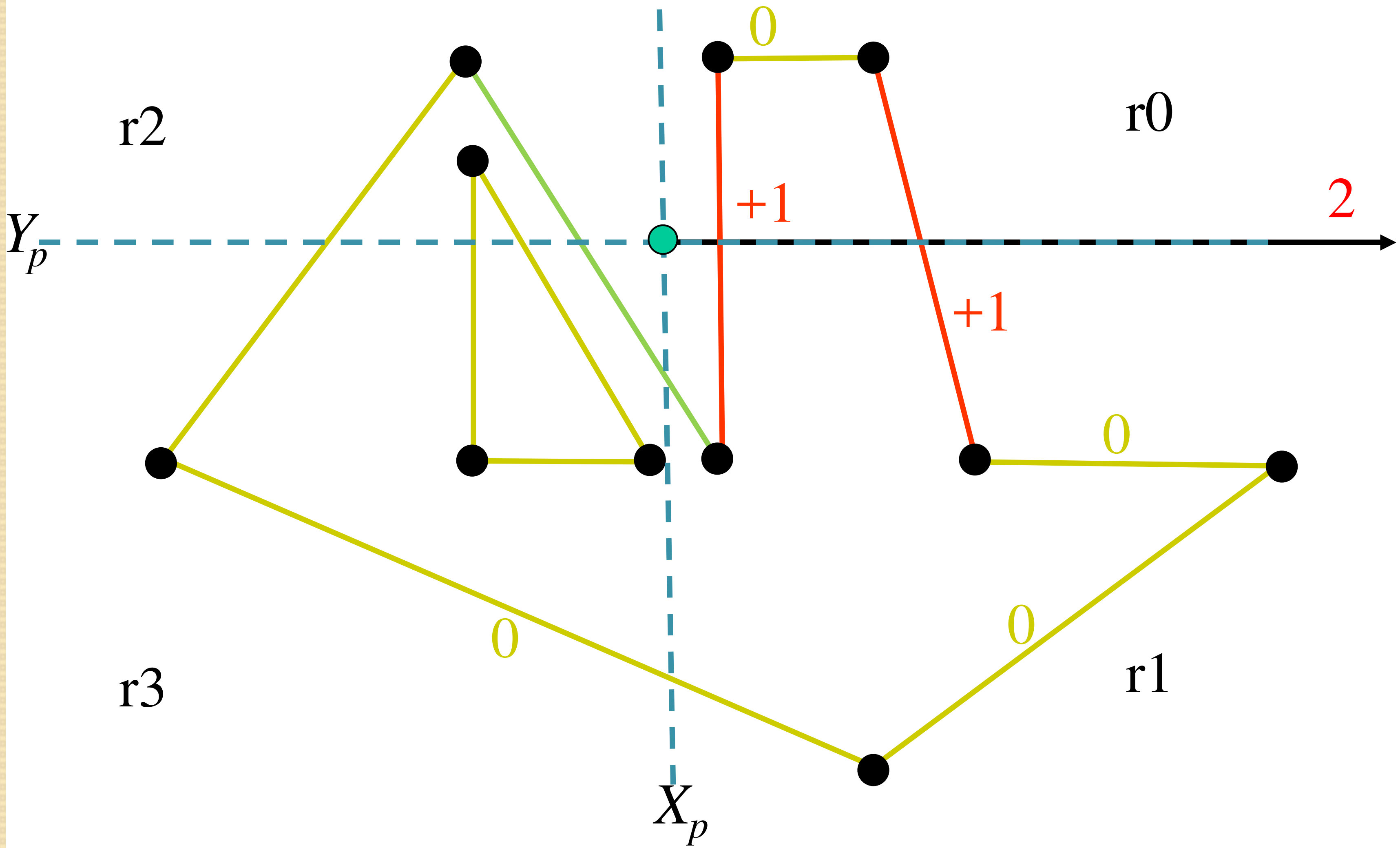
Point dans polygone ?

Le principe



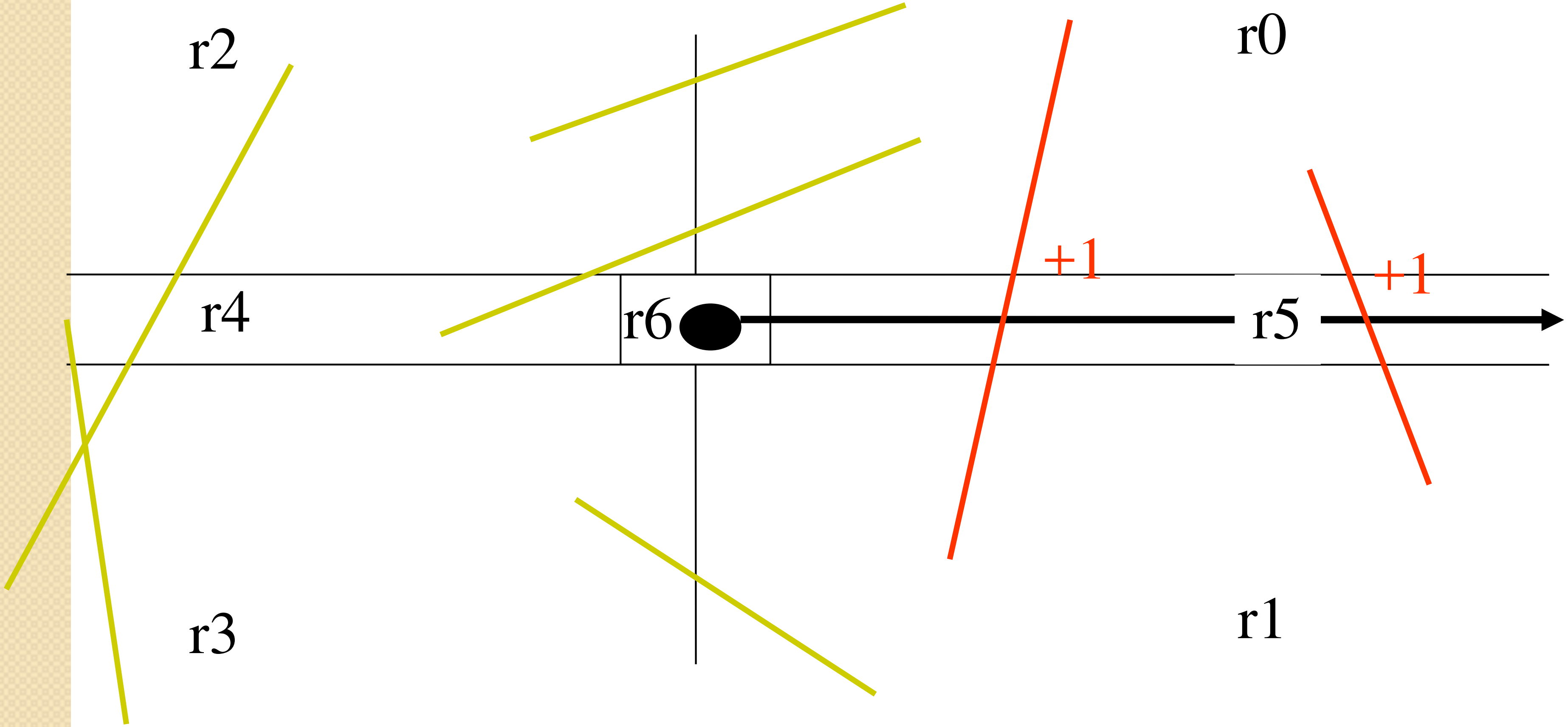
Point dans polygone ?

Le principe



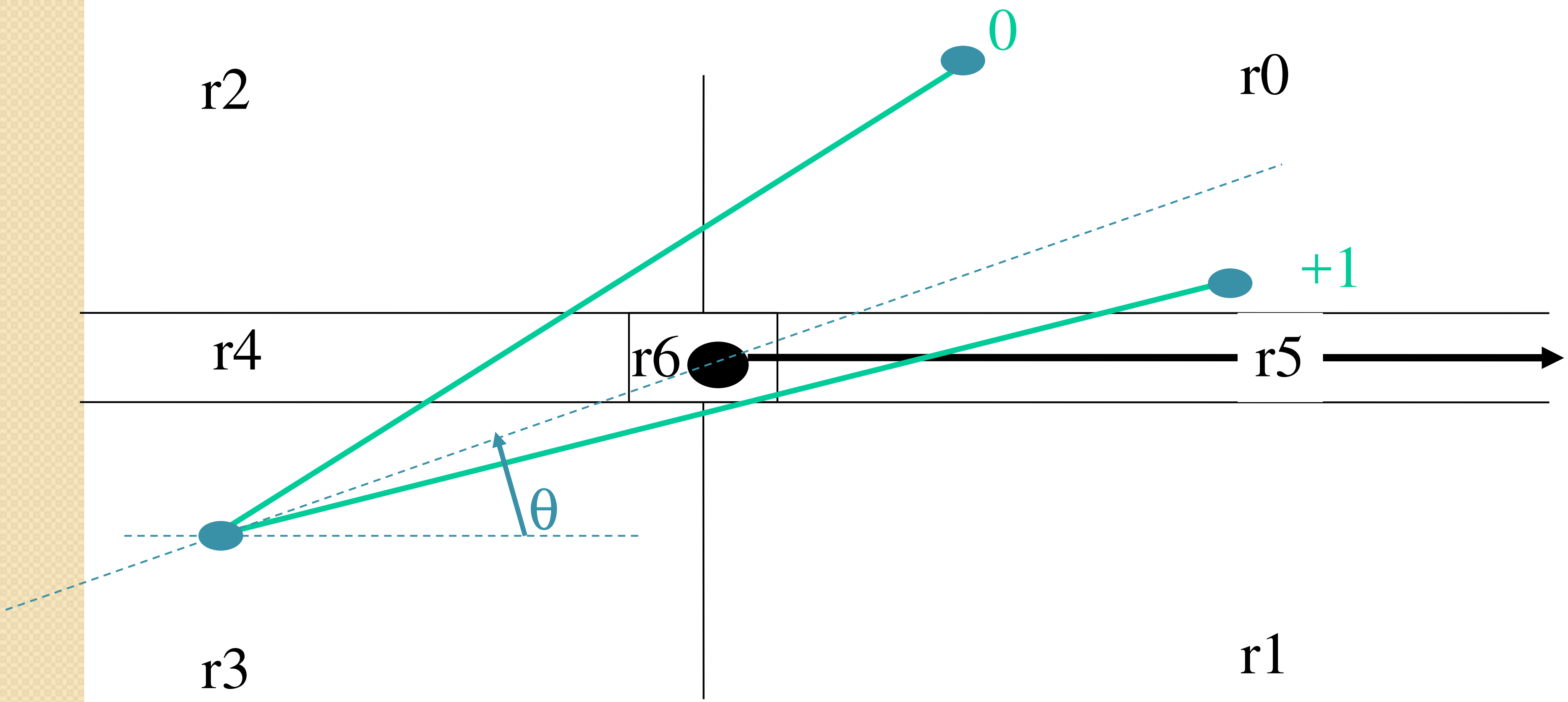
Compter les intersections (1)

Les cas simples



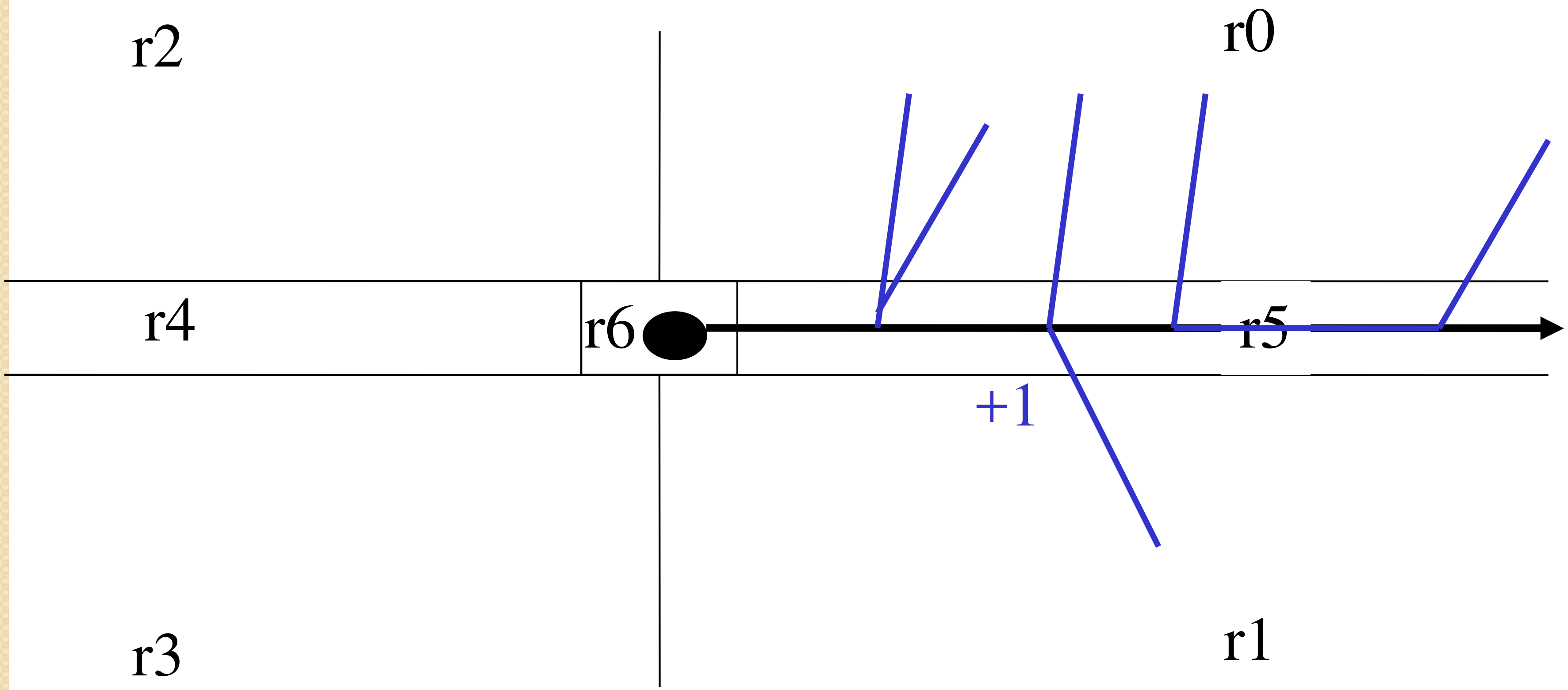
Compter les intersections (2)

Singularité simples



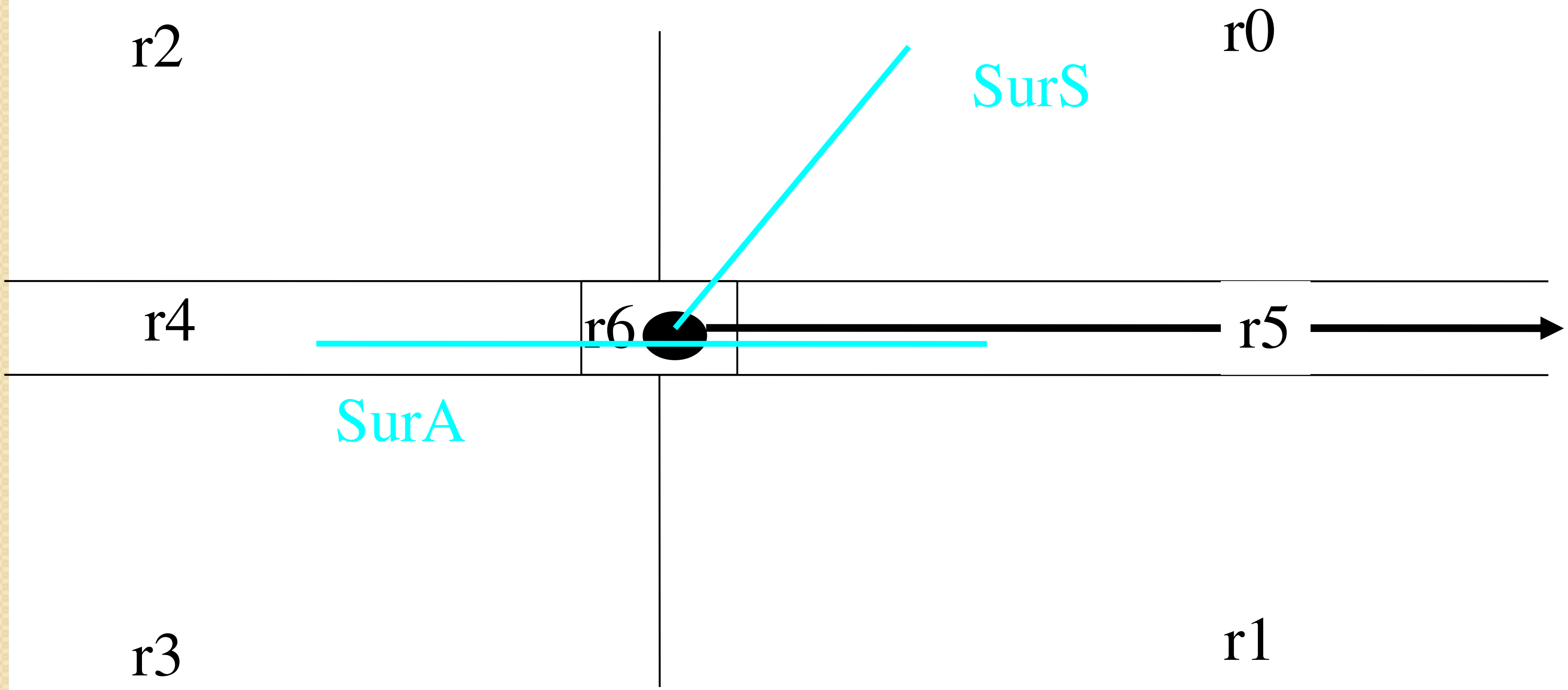
Compter les intersections (3)

Singularité compliquée



Compter les intersections (4)

Cas terminaux



Point dans polygone ?

L 'algorithme

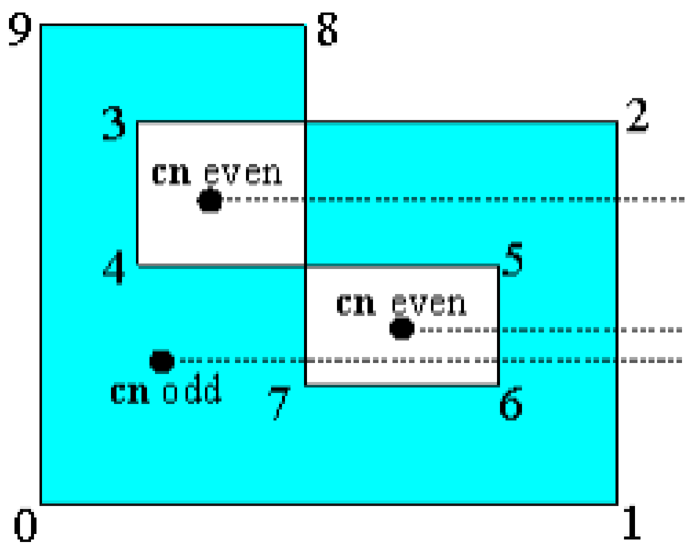
	r0	r1	r2	r3	r4	r5	r6
r0	0	+1	0		0		SurS
r1	+1	0		0	0		SurS
r2	0		0	0	0		SurS
r3		0	0	0	0		SurS
r4	0	0	0	0	0	SurA	SurS
r5					SurA		SurS
r6	SurS	SurS	SurS	SurS	SurS	SurS	SurS

Test de l 'angle

Parcours structure

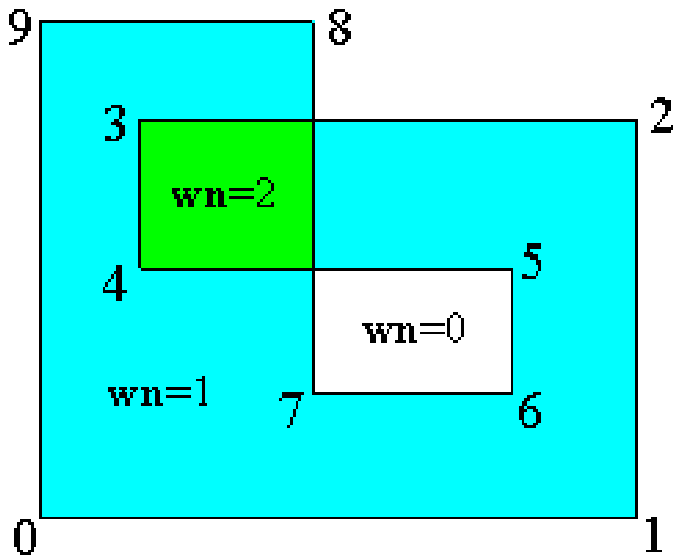
Autres algorithmes

Crossing Number Method



Vertices are numbered: 0 1 2 3 4 5 6 7 8 9

Winding Number Method



$$\mathbf{wn}(P,\mathbf{C}) = \frac{1}{2\pi}\sum_{i=0}^{n-1}\theta_i$$

$$= \frac{1}{2\pi}\sum_{i=0}^{n-1}\arccos\left(\frac{(V_i - P)\bullet(V_{i+1} - P)}{|(V_i - P)||(V_{i+1} - P)|}\right)$$

From Dan Sunday http://geomalgorithms.com/

Exercice : find in CGAL doc.

Dans la documentation CGAL (www.gcal.org) retrouvez l'équivalent de la fonction ***point_dans_ polygone().*** Vérifiez que l'algorithme repose sur le même principe que celui présenté en cours, justifiez (extraire les lignes correspondantes).

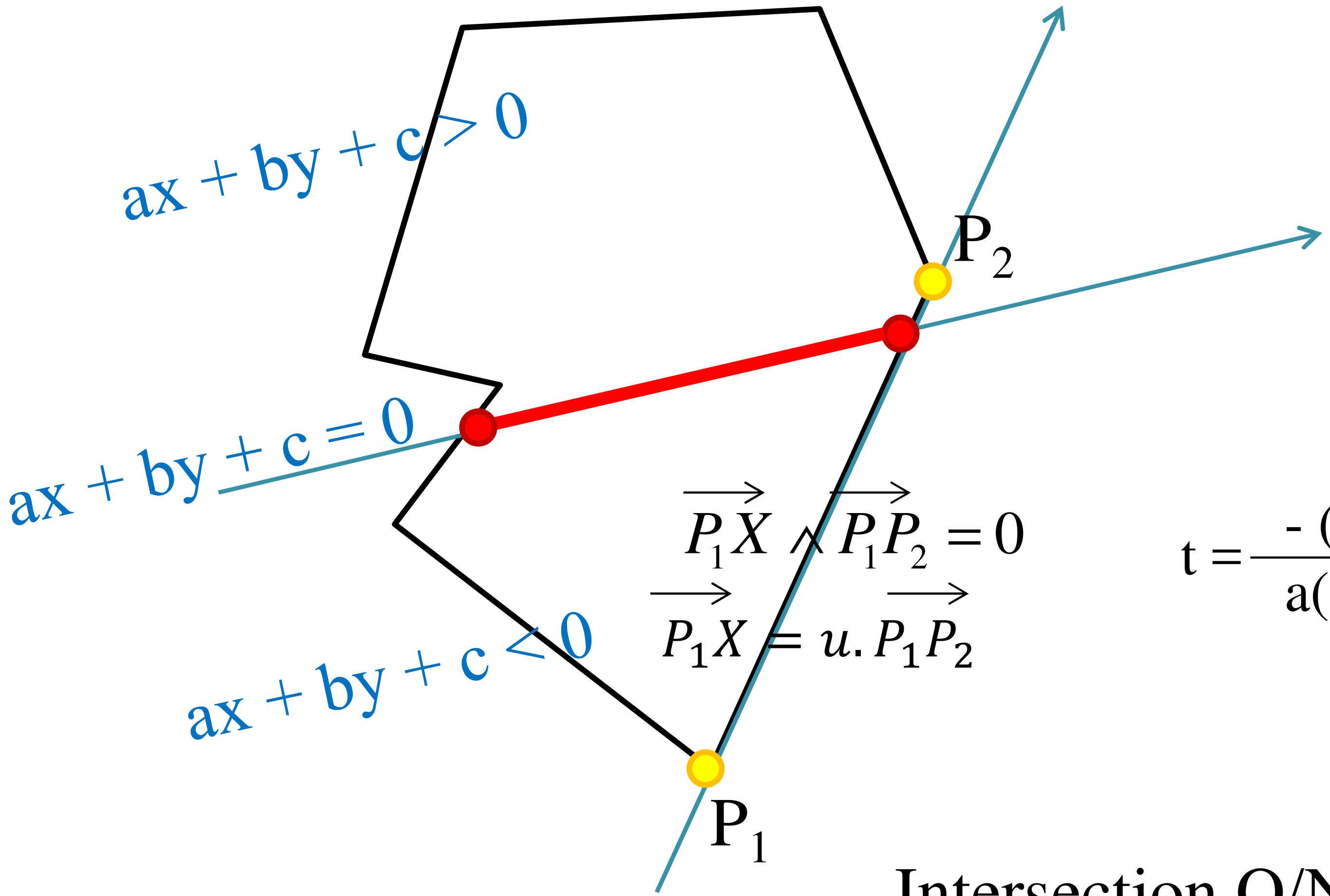
4. Intersection et collisions

Skip >

End >

Intersection avec une droite 2D ?

Le polygone = $\{P_i\,(x_i,y_i)\}$



$$\overrightarrow{P_1X} \wedge \overrightarrow{P_1P_2} = 0$$

$$\overrightarrow{P_1X} = u.\overrightarrow{P_1P_2}$$

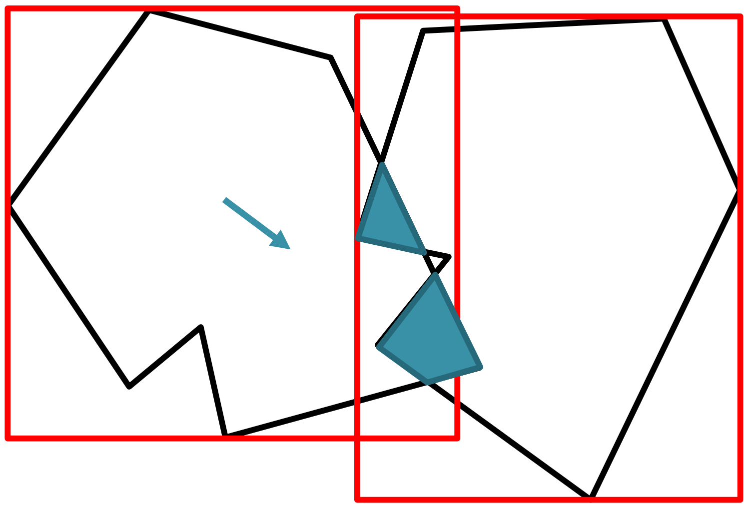
$$t = \frac{-\,(c+ax_1+by_1)}{a(x_2-x_1)+b(y_2-y_1)}$$

Intersection O/N ?

Coordonnées du Point d'intersection ?

Intersection de 2 polygones

1. **Détecter** : englobants & structures
2. **Calculer**/estimer : C= AnB
3. **Éviter** (trajectoire) : somme de Minkowski

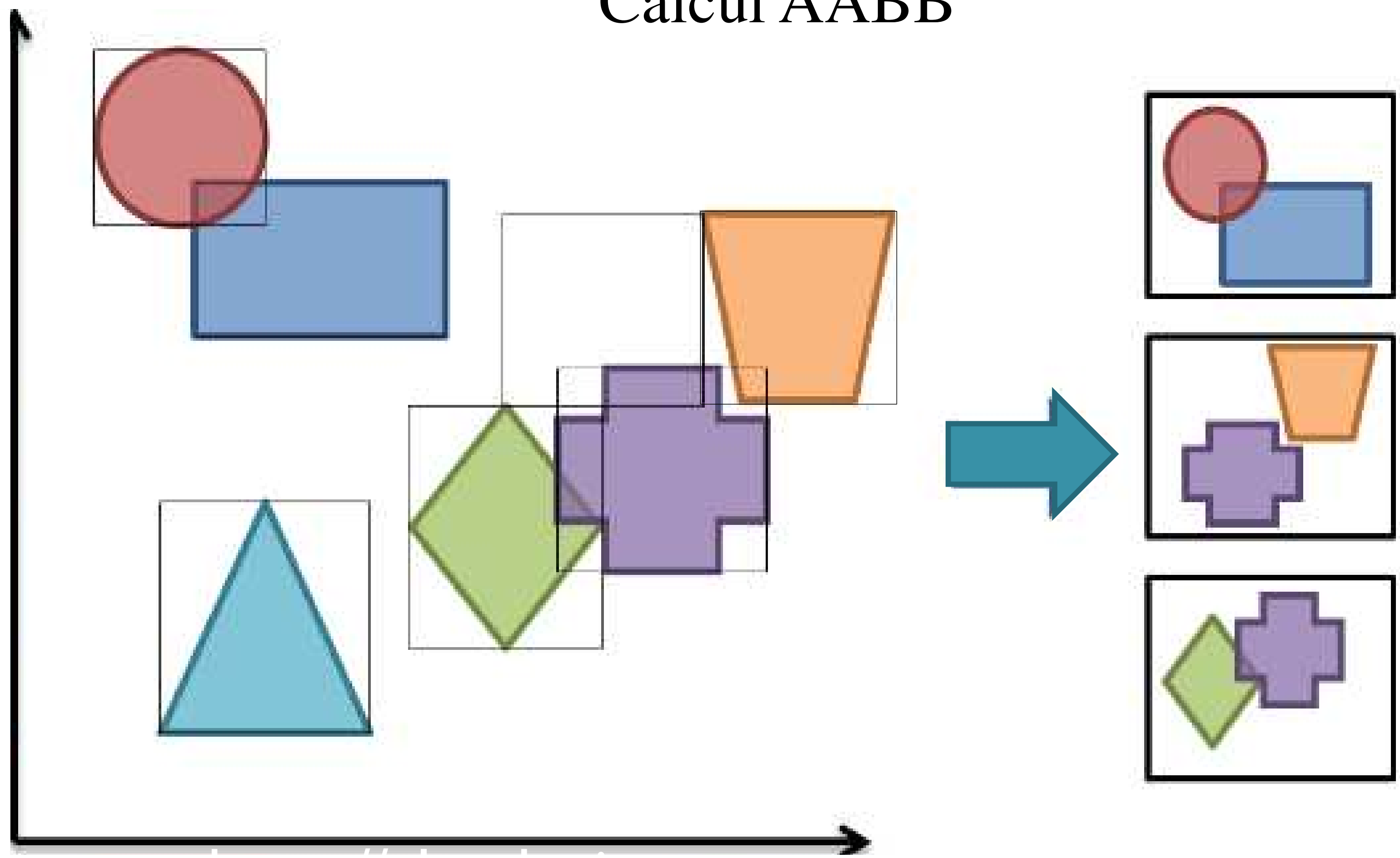


Détecter une collision

Définir la zone accessible

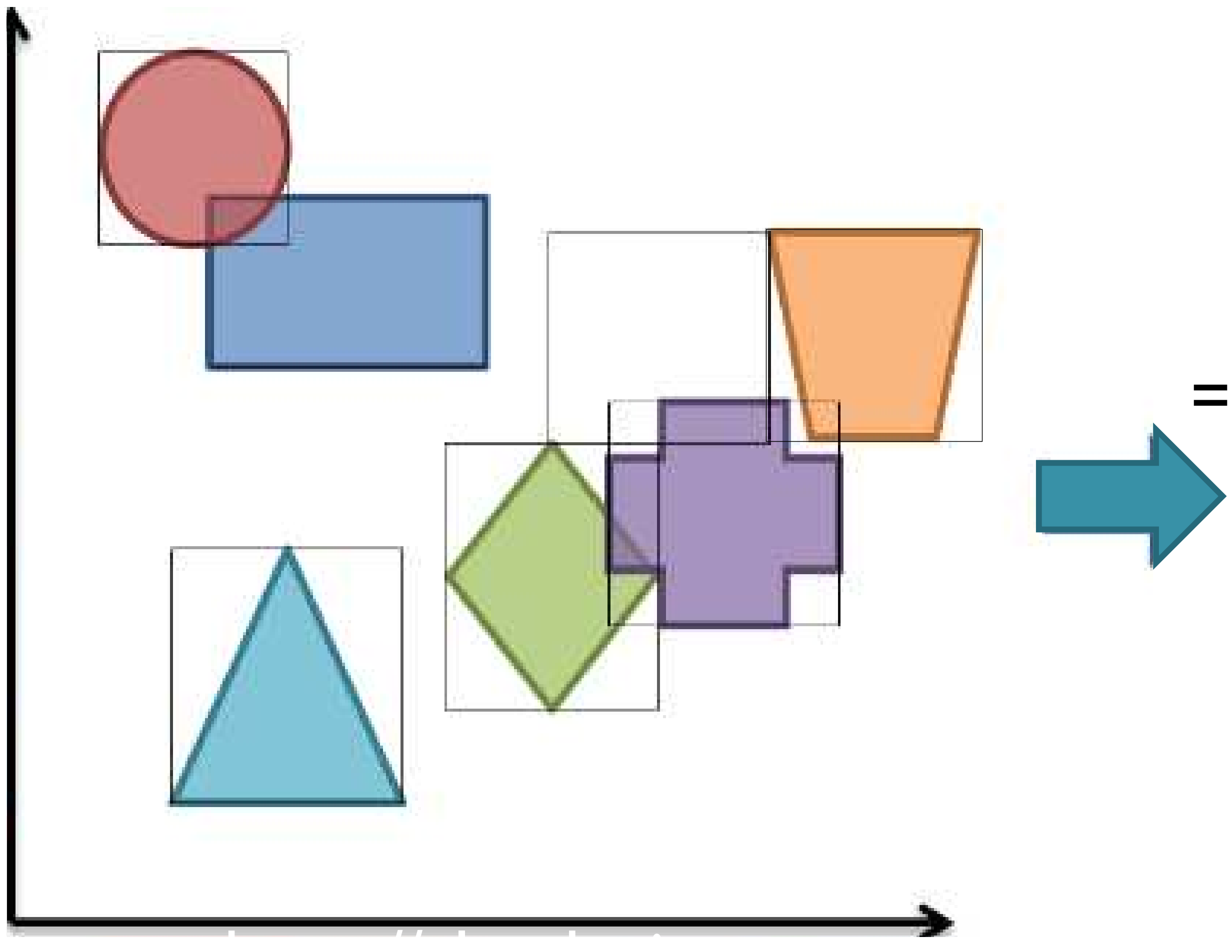
Détection : le volume englobant

Calcul AABB



Complexité?

Détection : le volume englobant

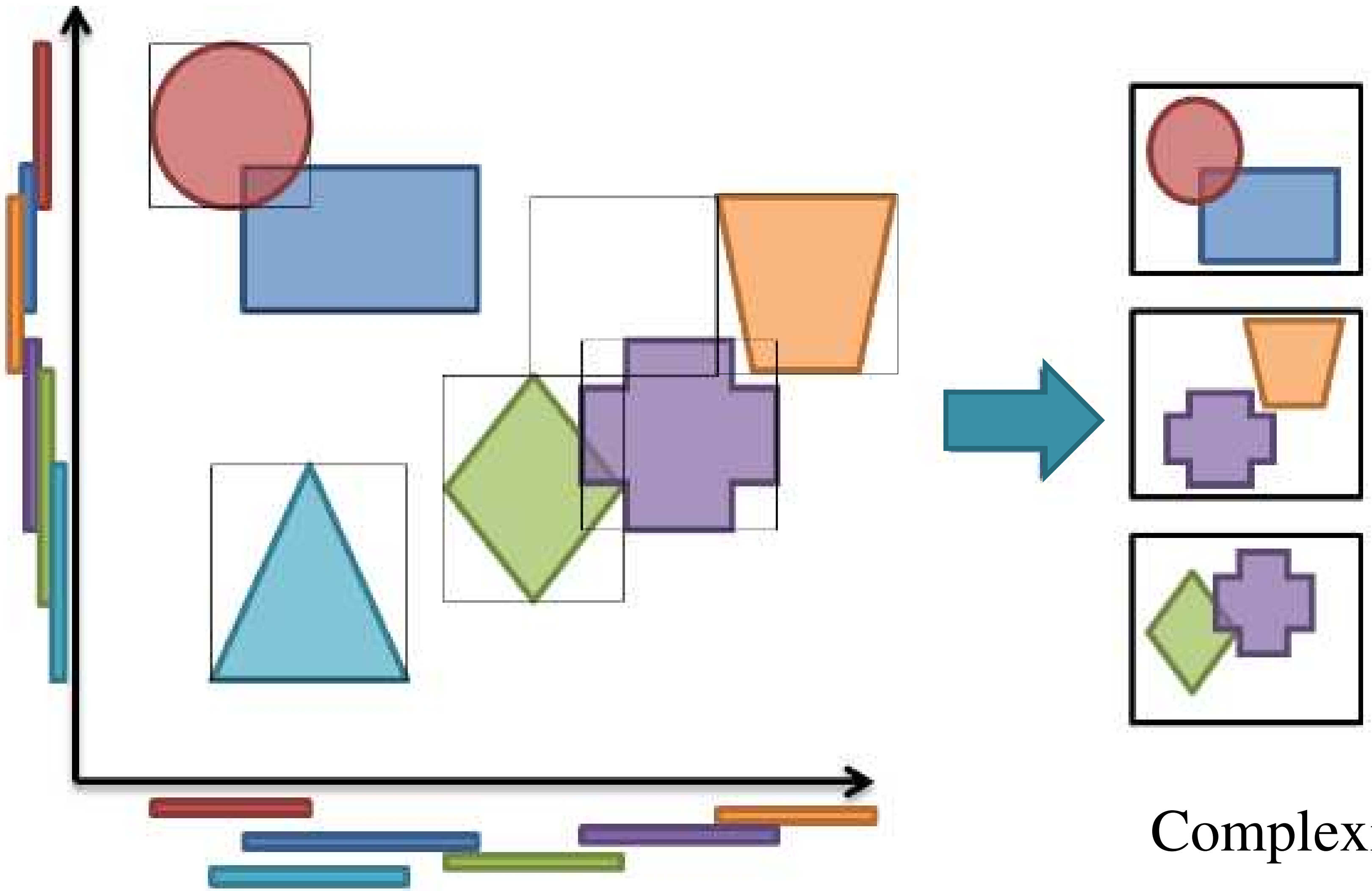


$= (n-1) + (n-2) + (n-3) + (n-4) + \ldots \rightarrow O(n^2)$

Complexité?

Détection : le balayage

$$(B_1 \cap B_2 \neq \emptyset) \Leftrightarrow (Bx_1 \cap Bx_2 \neq \emptyset) et (By_1 \cap By_2 \neq \emptyset)$$

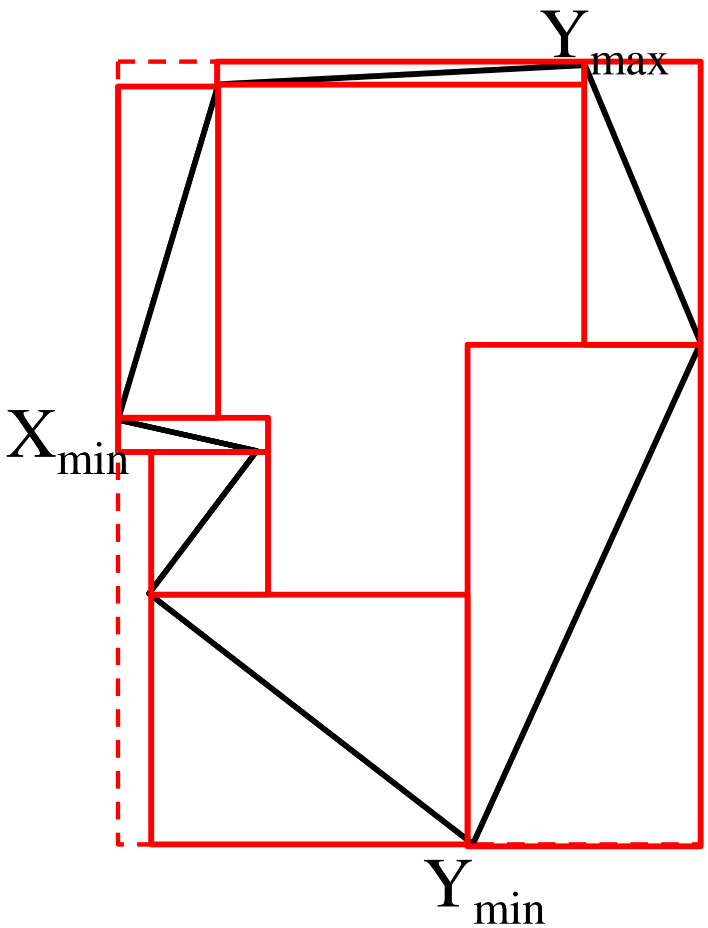


Complexité?

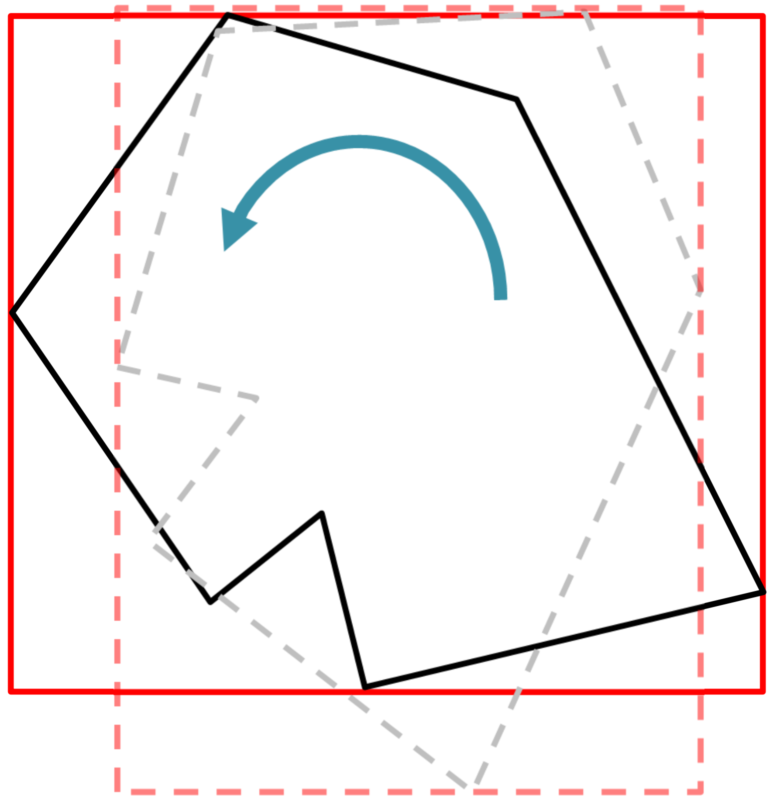
Balayage (activation): pas de tests inutiles

Boites d'encombrement

AABB : Axis-Aligned Bounding Box

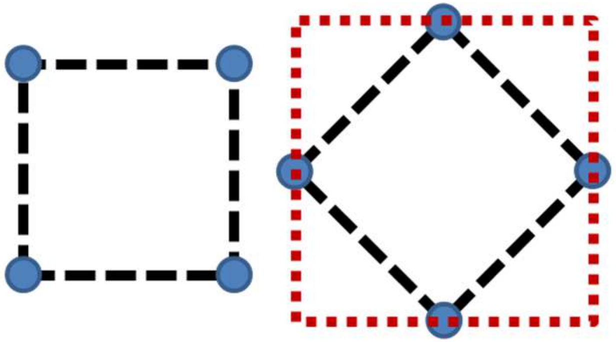


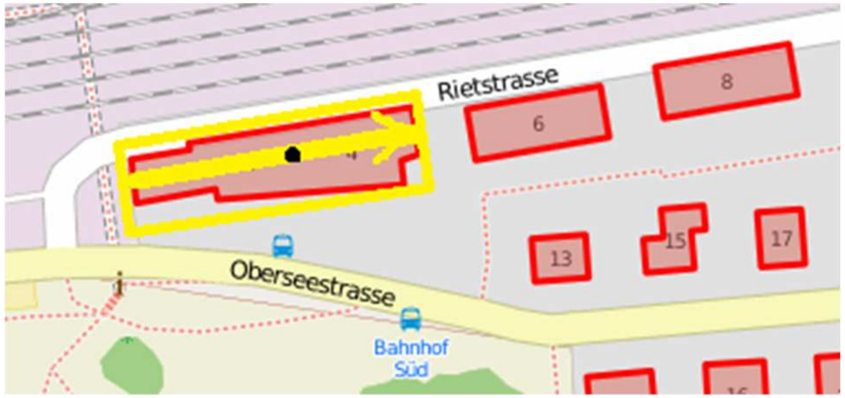
Boites hiérarchiques

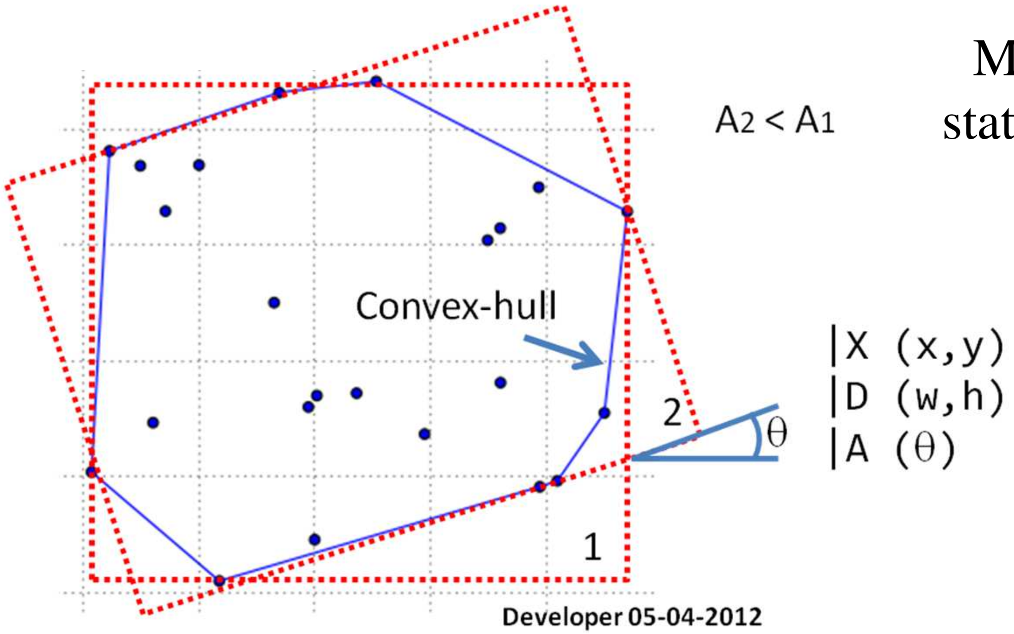


Rotation AABB !
(vs OBB, cercle)

Oriented Bounding Box

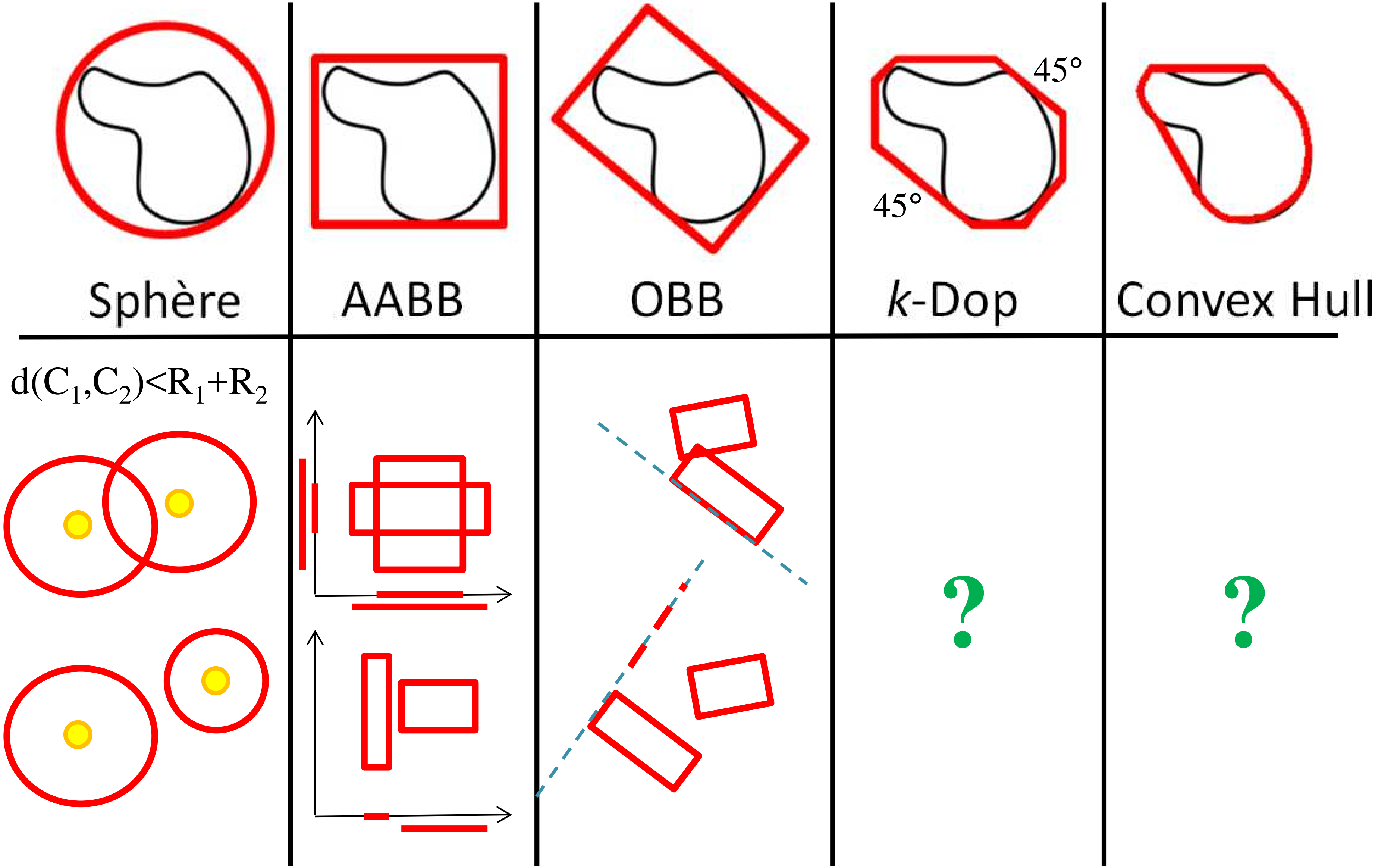






Même dans des environnements statiques on peut préférer des OBB

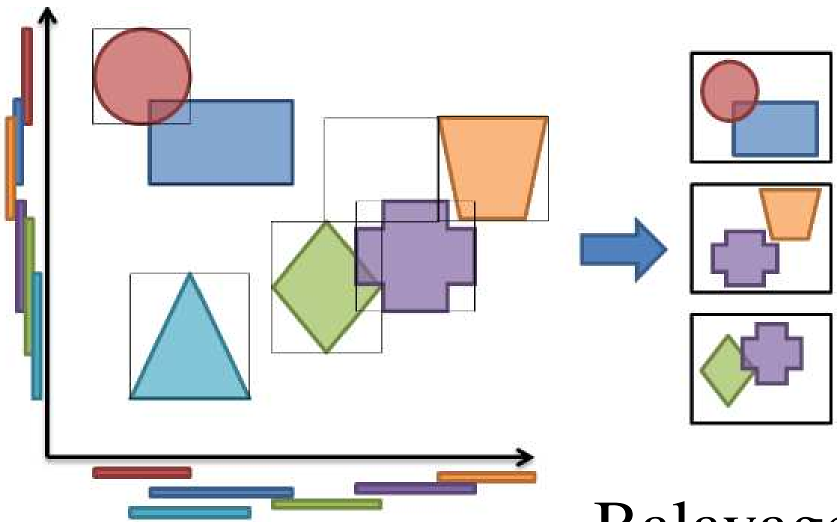
Volumes englobant



Principe : 2 convexes non intersectant peuvent être séparé par une droite D // à un côté d'un polygone
Algo : Trouver D_i, les projetés des 2 polygones sur la droite D_i sont disjoints

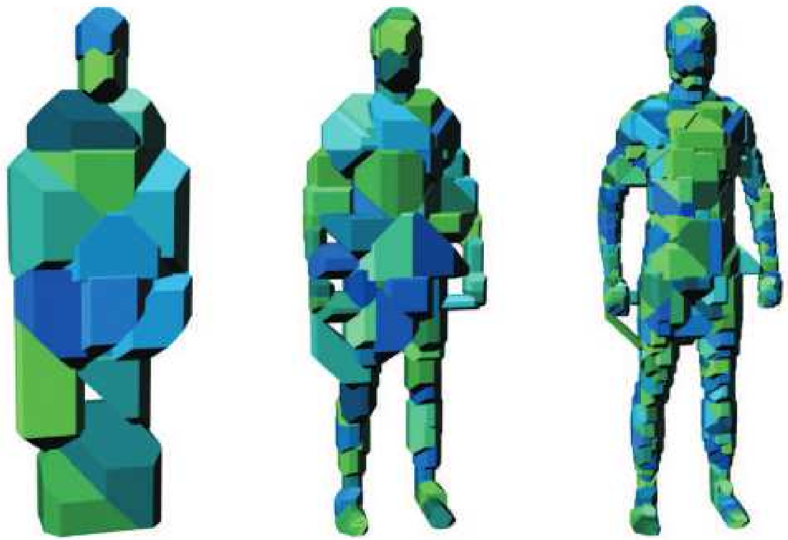
Détecter l'intersection

Volume englobant

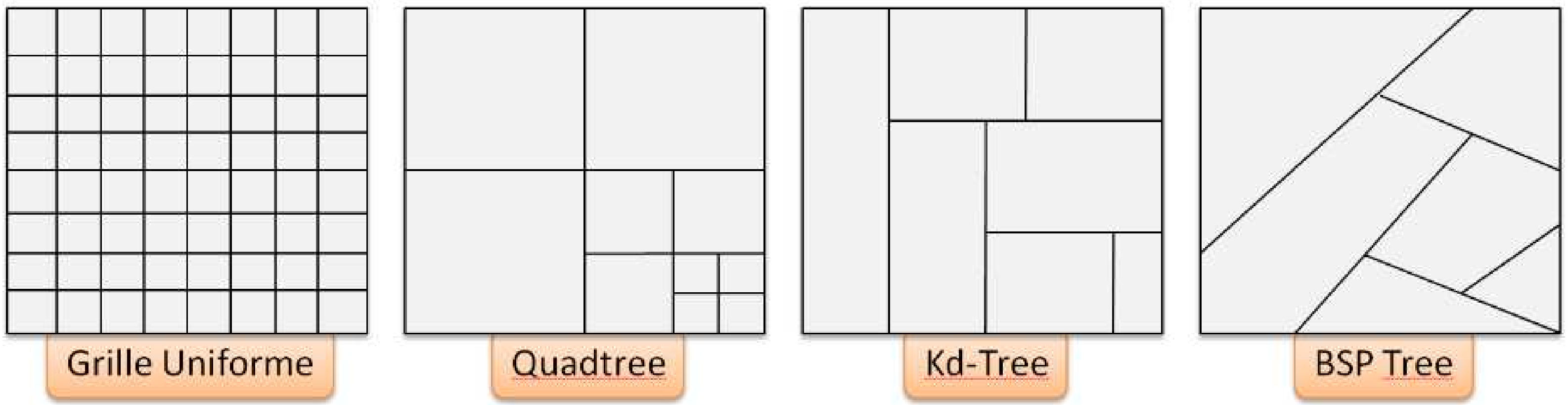


Balayage

Hiérarchie

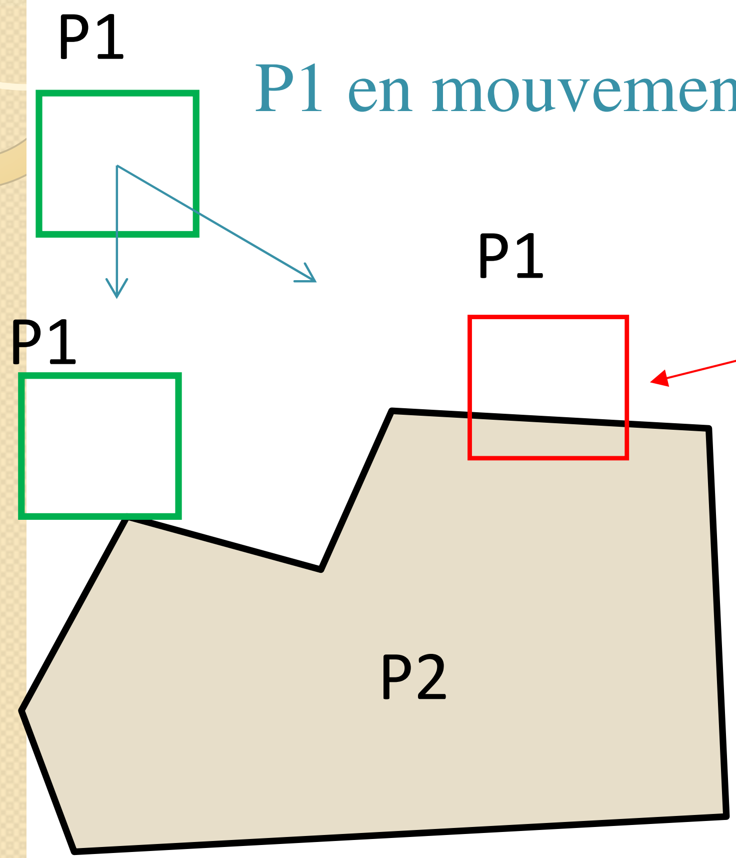


Structures d'accès / subdivision spatiale



Éviter les « Collisions »

P1 en mouvement (de translation)

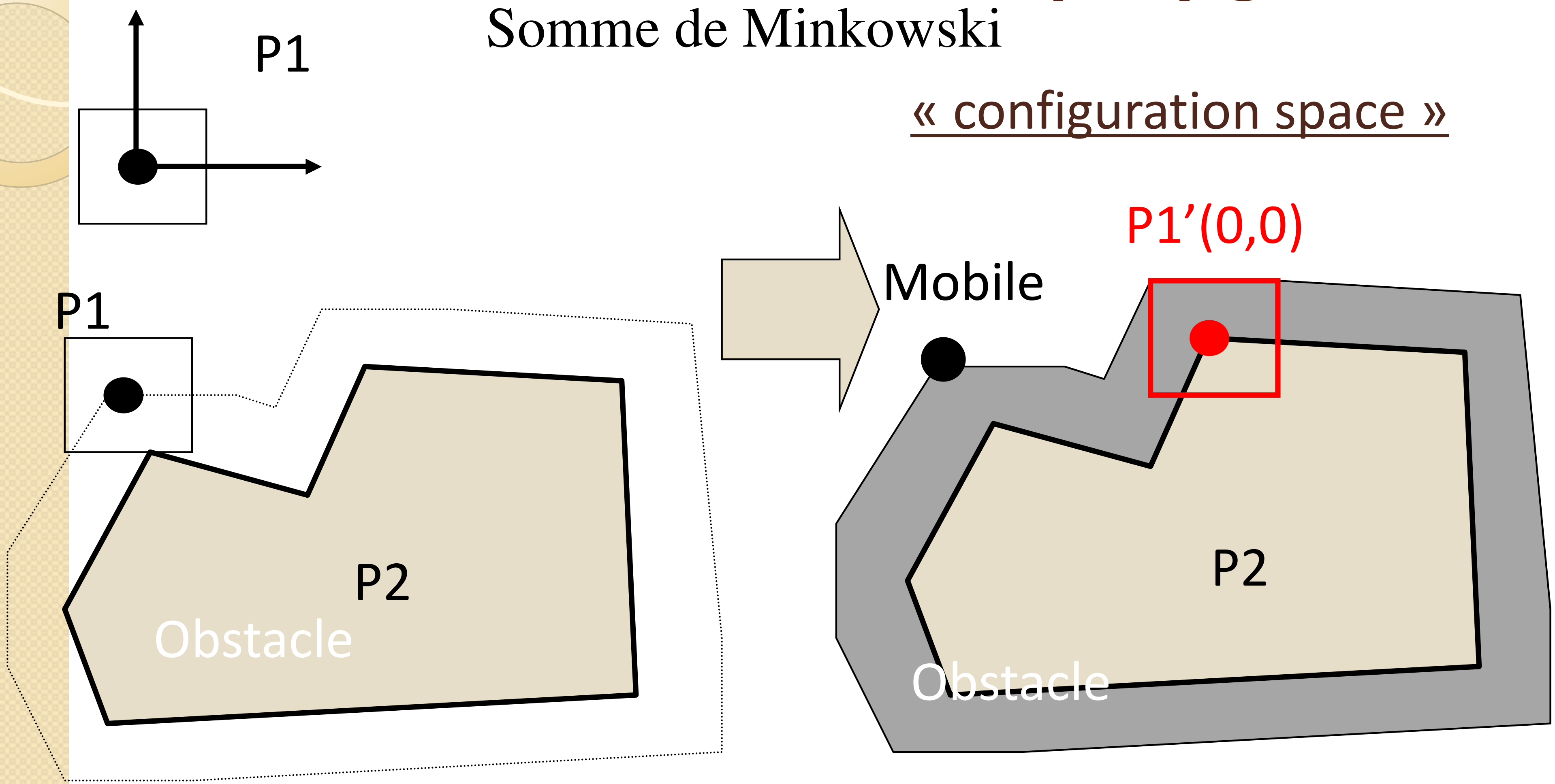


On veut éviter l'intersection

P2 immobile = obstacle

« Collision » entre 2 polygones

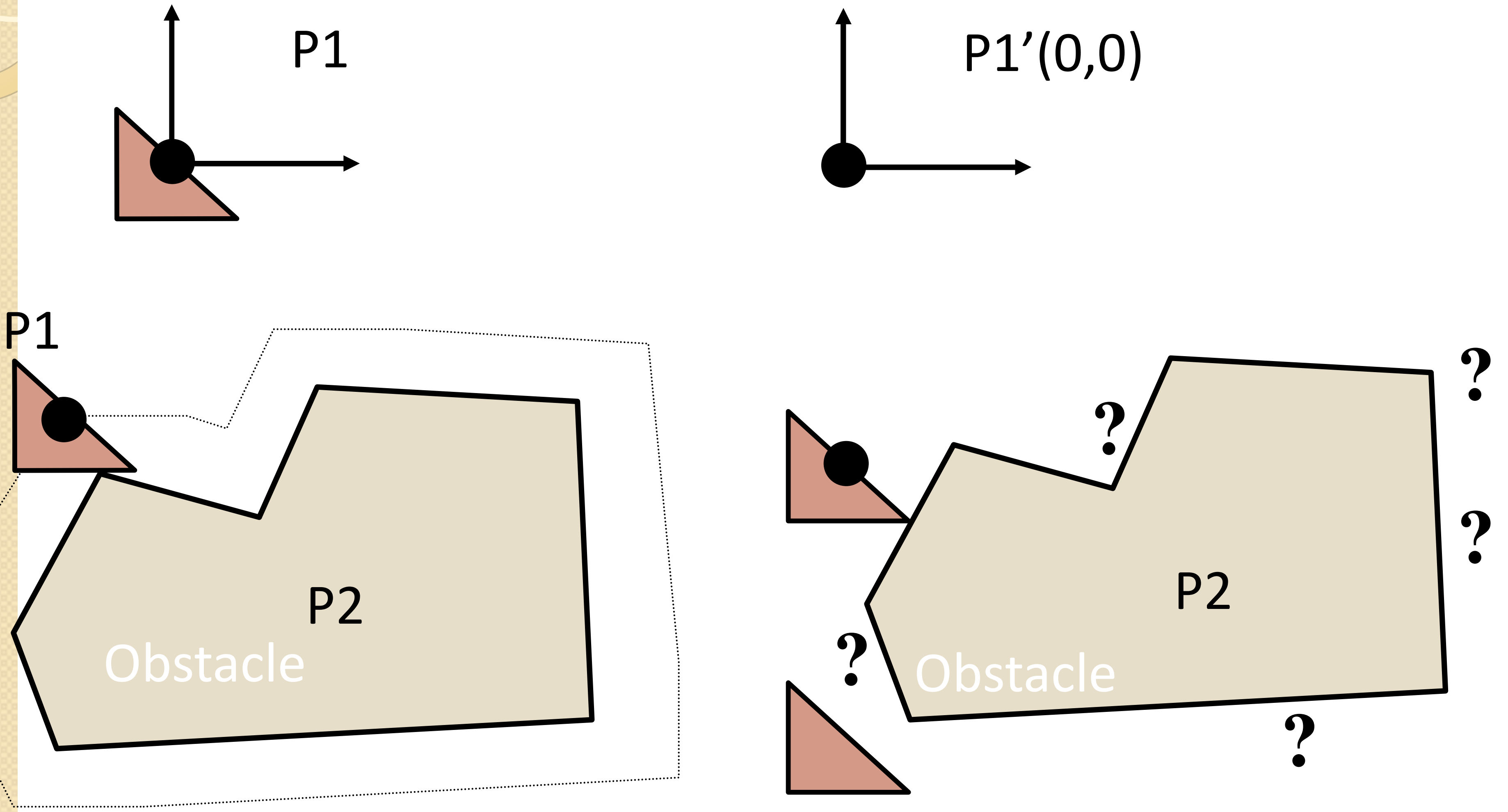
Somme de Minkowski



FS(P2,P1) = P2 + P1'(0,0)

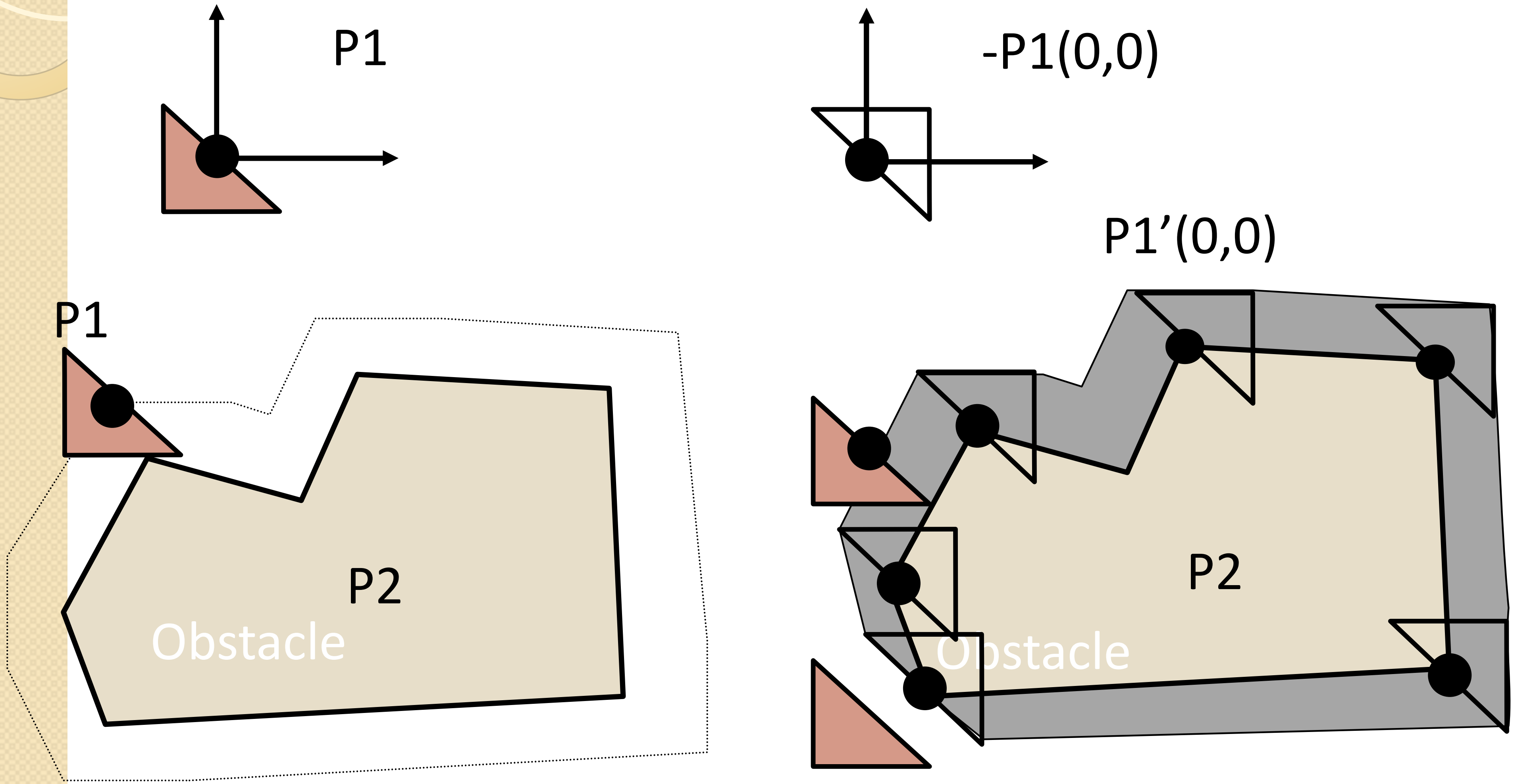
Intersection P1 n P2 =0 ? ⇔ Point « c1 » hors FS(P2,P1) ?

« Collision » entre 2 polygones



FS(P2,P1) = P2 + P1'(0,0)

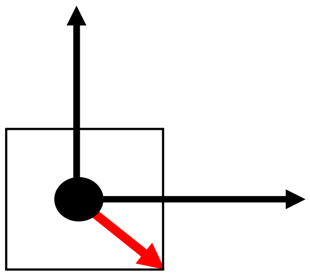
« Collision » entre 2 polygones

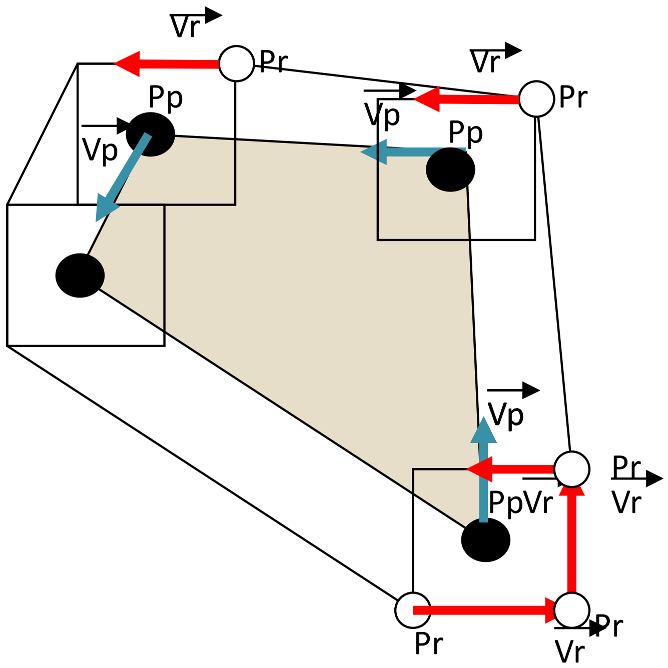


FS(P2,P1) = P2 *+* -P1(0,0)

Sommes de Minkowski

Polygones convexes





$\mathrm{Vr} = \overrightarrow{\mathrm{Pr\ SUIV[Pr]}}$

$\mathrm{Vp} = \overrightarrow{\mathrm{Pp\ SUIV[Pp]}}$

Si(PrV(Vr, Vp)> 0)Alors

Pr = SUIV[Pr]
FS = FS u { Pp + Pr }

Sinon Pp = SUIV[Pp]

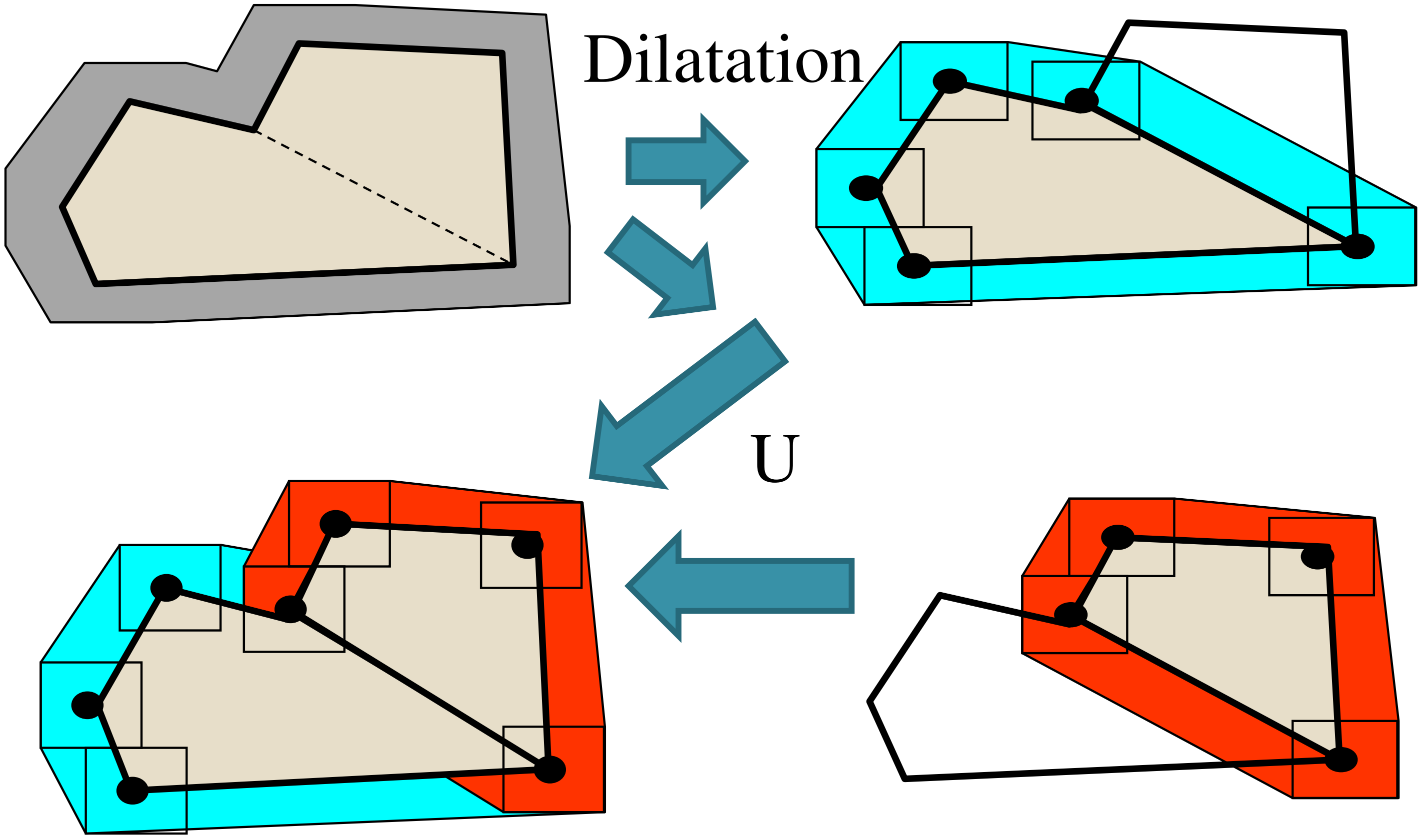
FS = FS u { Pp + Pr }

Les sommets du polygone P1⊕P2 = sommets de P2 + VSi

Sommes de Minkowski

Polygones non convexes

Décomposition en convexes



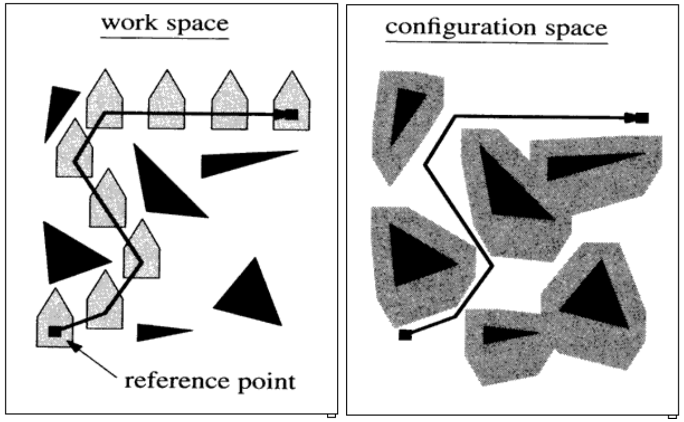
Opération booléenne d'union entre les 2 polygones dilatés

Motion planning…

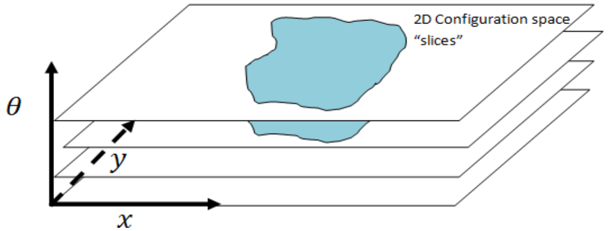
Somme de Minkowski

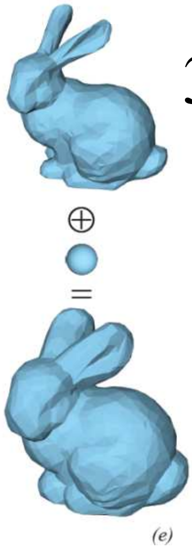
translation

2D : jeux vidéo

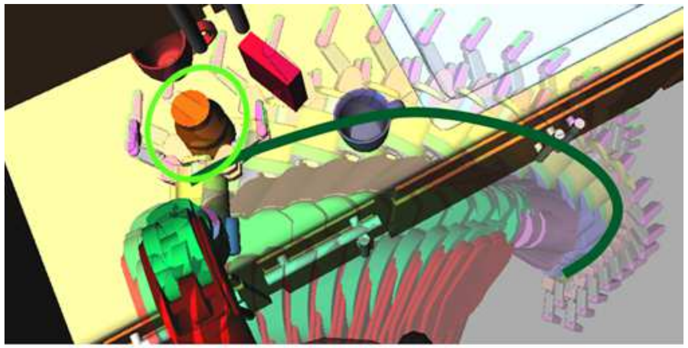


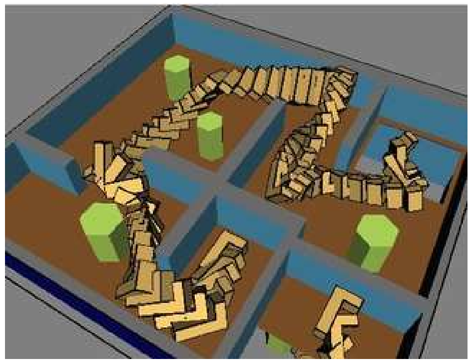
rotation



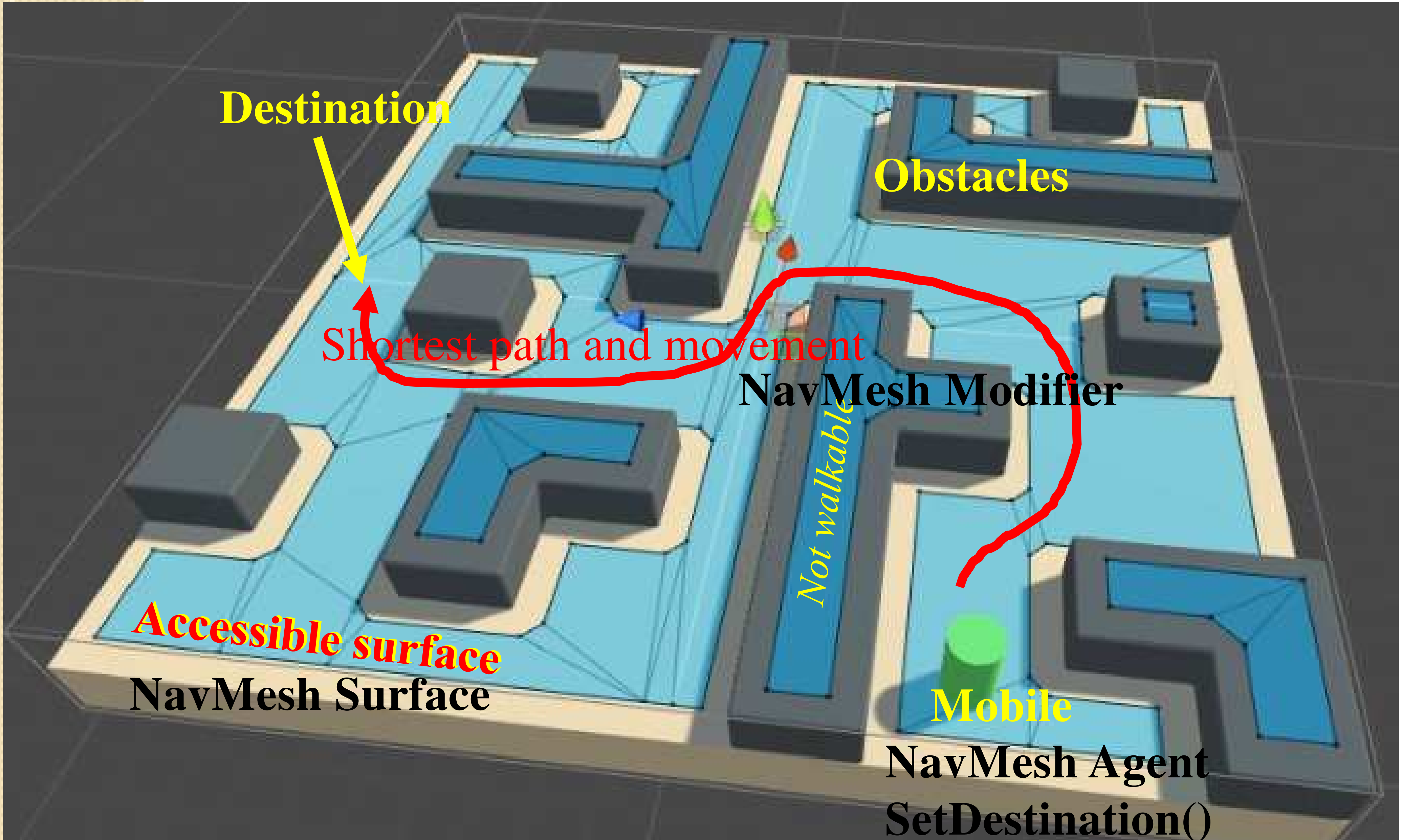


3D : Robotique



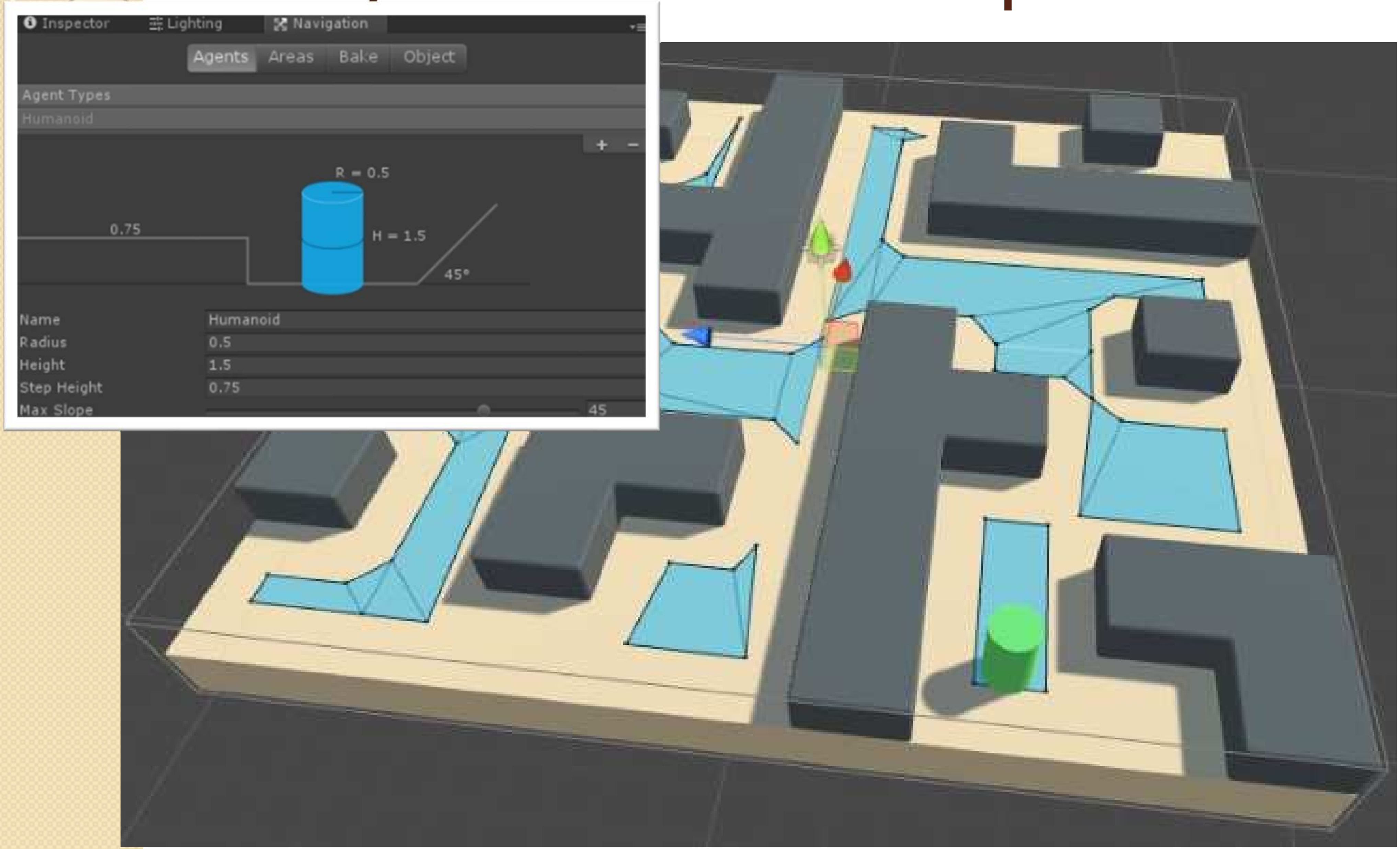


Unity NavMesh



Brackeys : https://www.youtube.com/watch?v=CHV1ymlw-P8 (12min)

Unity NavMeshSurface specification



5. Opérations booléennes

Skip >

End >

Opérations booléennes

Fr(A), Fr(B)

Fr(AnB) =

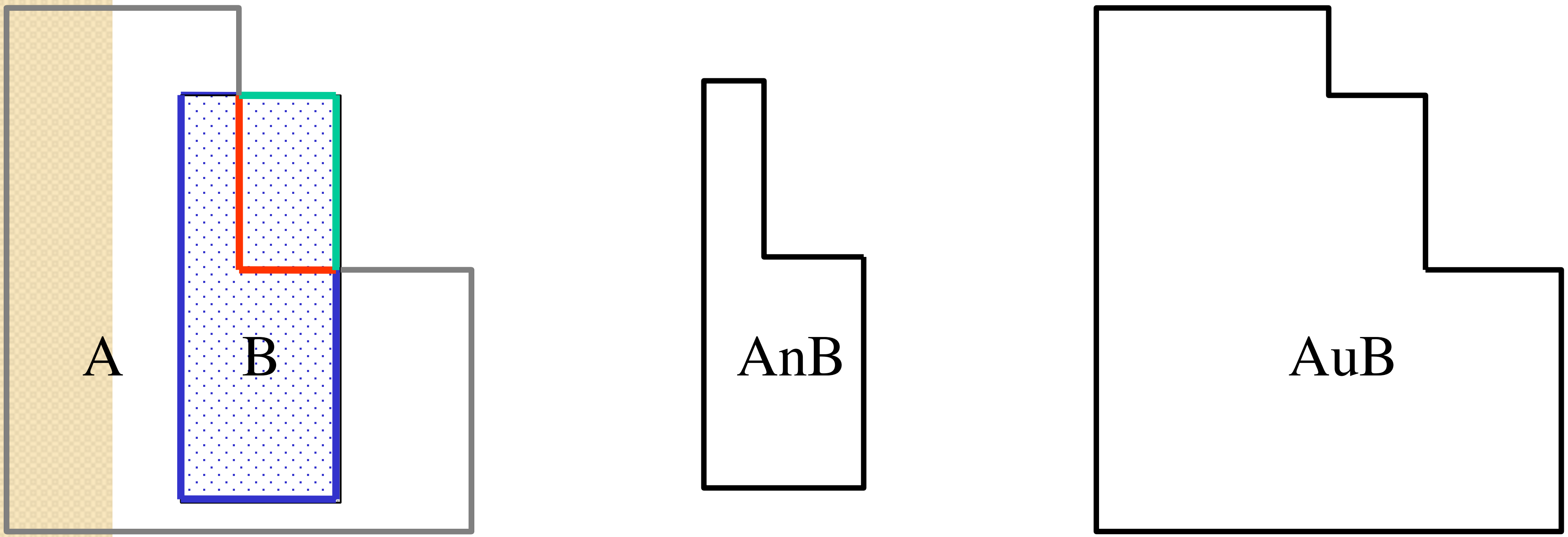
Fr(B)dans A

Fr(A)dans B

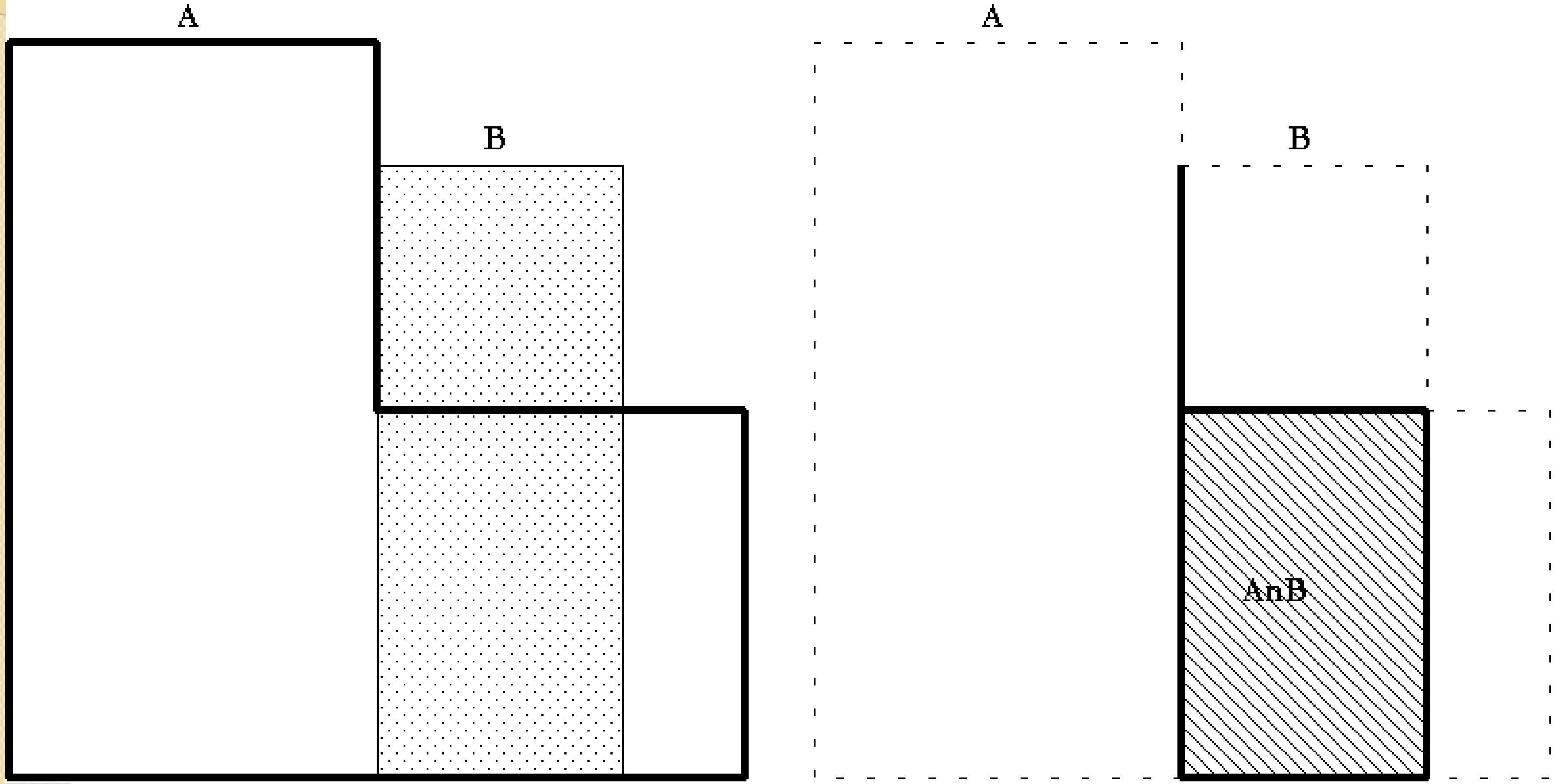
Fr(AuB) =

Fr(A) hors B

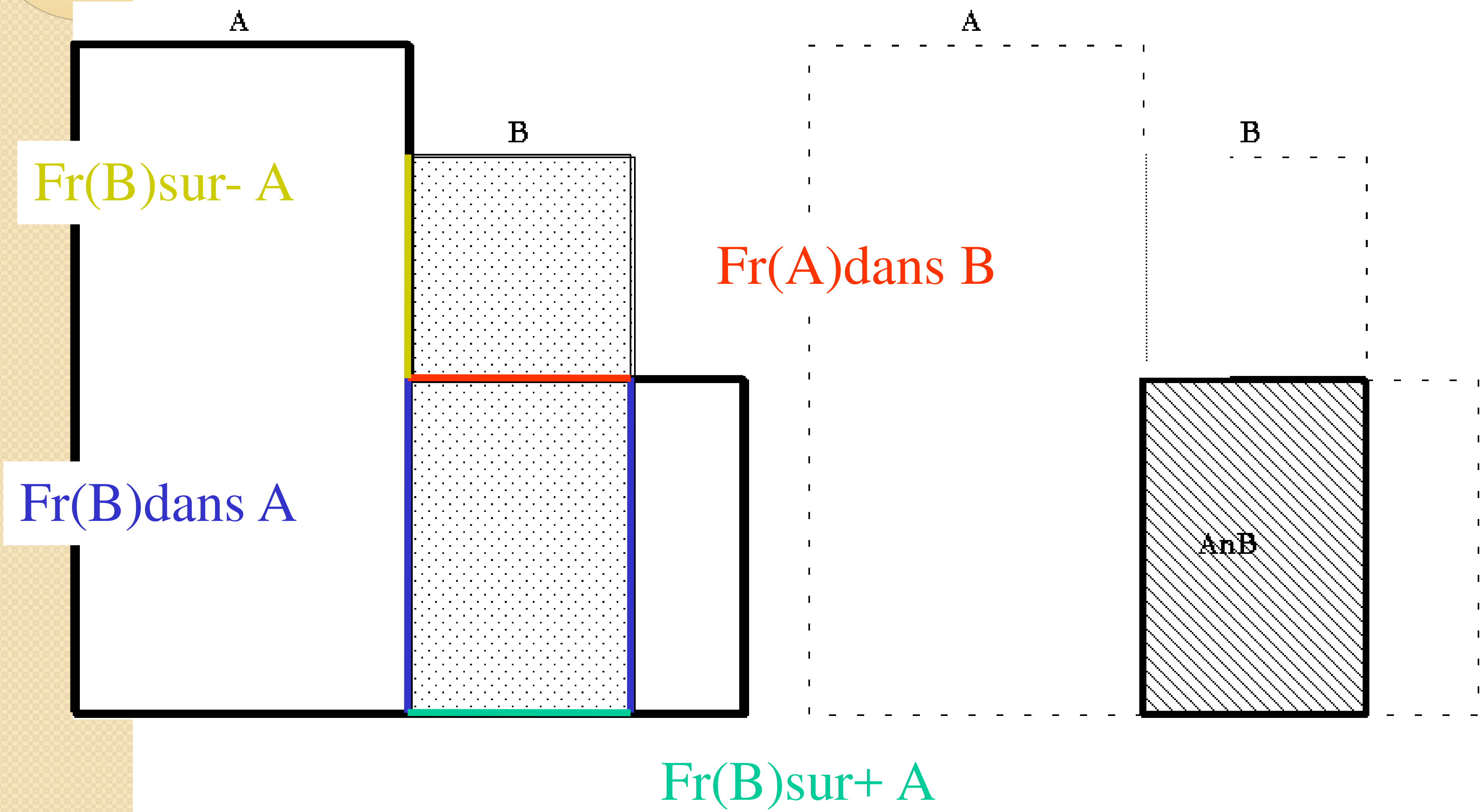
Fr(B)hors A



Opérations booléennes régularisées



Opérations booléennes régularisées



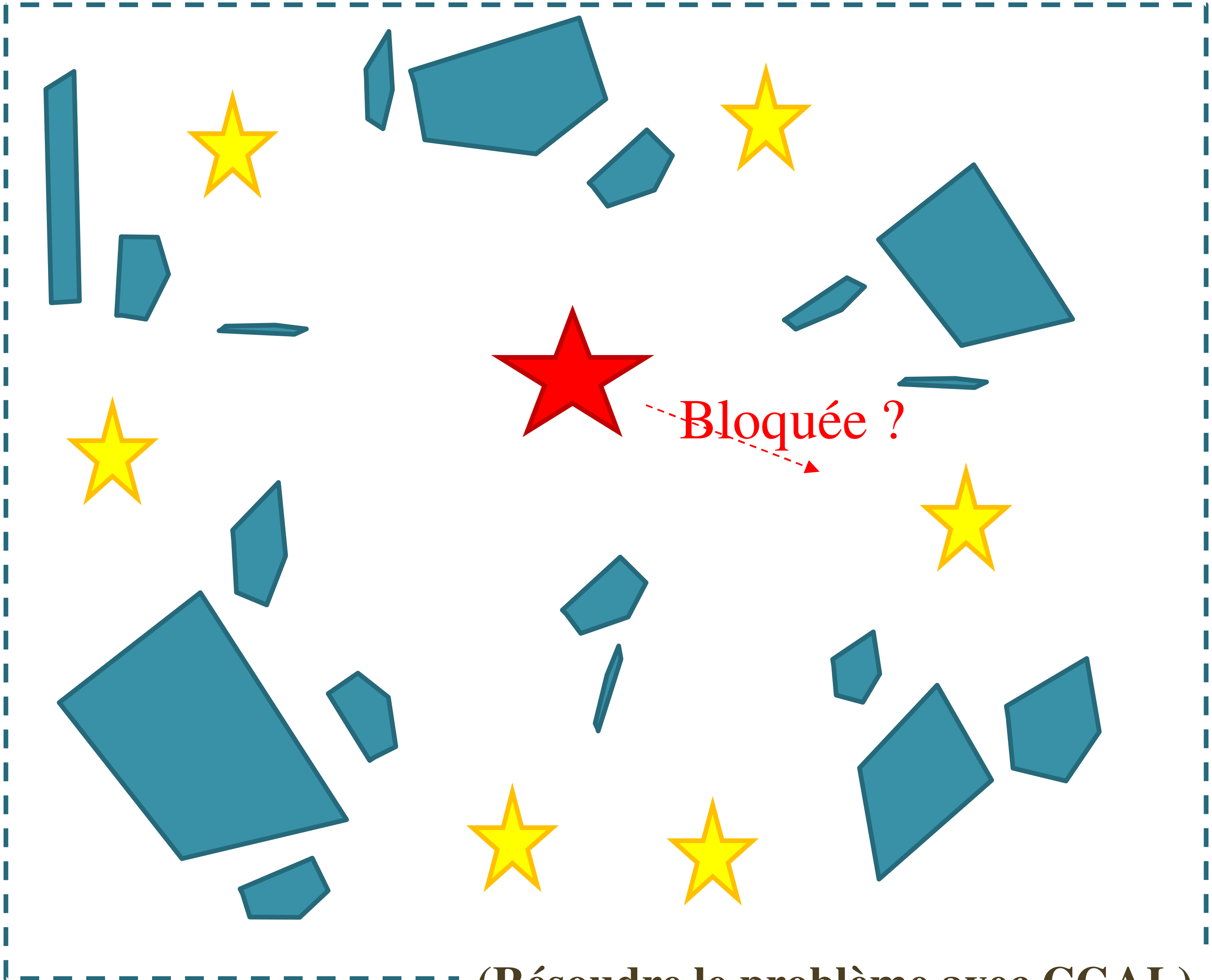
Opérations booléennes régularisées

Fr(AnB) = Fr(A) dans B + Fr(B) dans A + Fr(B) sur + A

Fr(AuB) = Fr(A) hors B + Fr(B) hors A + Fr(B) sur +A

Fr(A-B) = Fr(A) hors B + Fr(B) dans A + Fr(B) sur - A

Exercice : Jeu de la prison



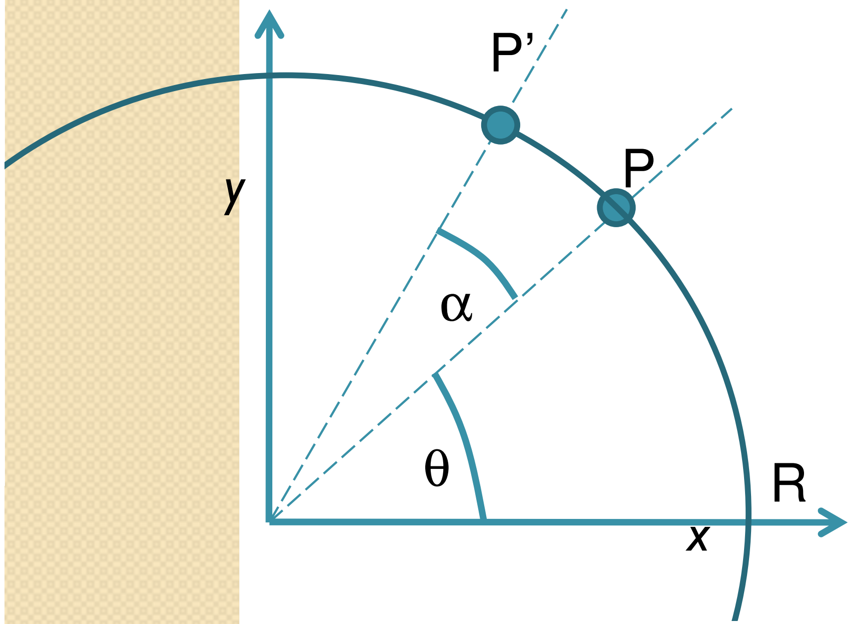
(Résoudre le problème avec CGAL)

6. Transformations & coordonnées Homogènes

Transformations : la rotation

Rappels de géométrie (1)

$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \qquad P' = R_z(\alpha)(P) = \begin{pmatrix} x' = x\cos(\alpha) - y\sin(\alpha) \\ y' = x\sin(\alpha) + y\cos(\alpha) \end{pmatrix}$$



En coordonnées polaires

$$P = \begin{pmatrix} x = r\cos(\theta) \\ y = r\sin(\theta) \end{pmatrix}$$

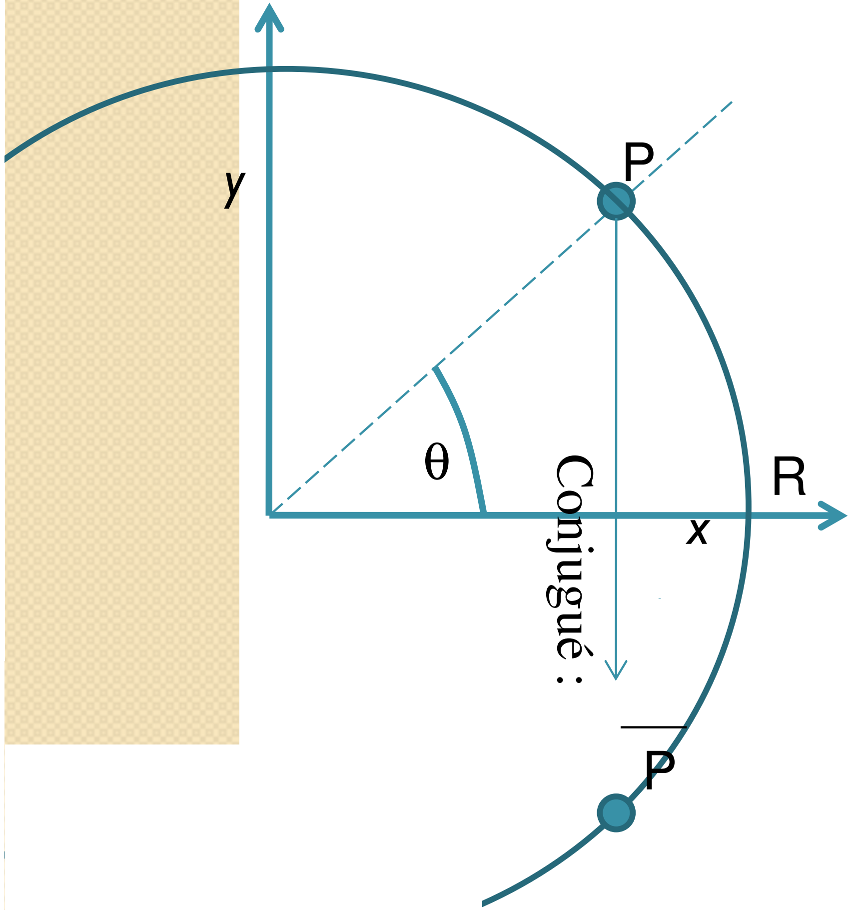
$$P' = \begin{pmatrix} x' = r\cos(\alpha+\theta) \\ y' = r\sin(\alpha+\theta) \end{pmatrix}$$

Composition : $R_{oz}(\alpha) \circ R_{oz}(\beta) = R_{oz}(\alpha+\beta)$

Rotations et complexes

Représentation complexe : $P\,(\,x + i\,y\,)$

$P\,(\,r\,(\,cos(\theta) + i\,sin(\theta))\,)$



Conjugué : $\overline{P}\,(\,x - i\,y\,)$ $\qquad \|P\| = \sqrt{P.\overline{P}}$

<u>Multiplication :</u>

$P_1.P_2 = x_1.x_2 - y_1.y_2 + i\,(x_1.y_2 + y_1.x_2)$

$$\boxed{P_1.P_2 = r_1.r_2\,(\,cos(\alpha_1+\alpha_2) + i\,sin(\alpha_1+\alpha_2)\,)}$$

rotation d'angle α

⇔

multiplication par un complexe
d'arg α et de norme 1

Transformations en Coordonnées Homogènes

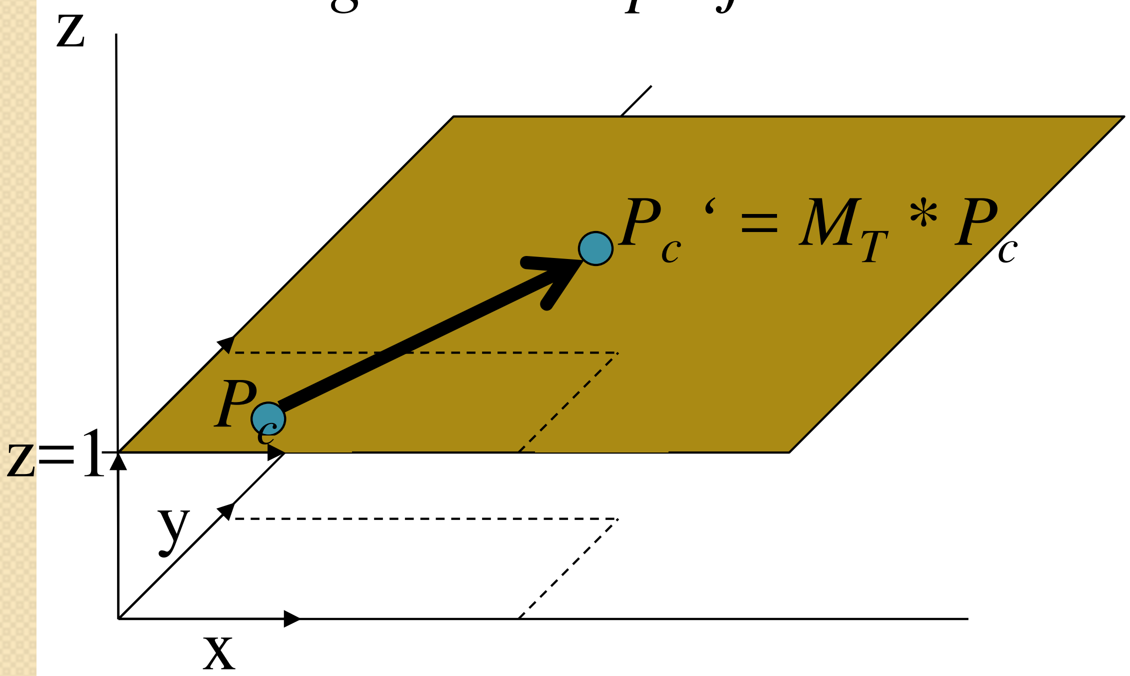
Matrice de transformation dans R^{d+1}

d=2

$$Pc = \begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow Pc = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \longrightarrow Pc = \begin{pmatrix} x/1 \\ y/1 \end{pmatrix}$$

Coordonnées Homogènes
géométrie projective

$$R_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



$$M_T = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Bibliographie

- Introduction à la modélisation et à l'algorithmique géométrique
  Olivier STAB
  - *http://people.mines-paristech.fr/olivier.stab/geomodeling.pdf*
- Geometric Modeling
  Michael MORTENSON -*Wiley*
- Computer Graphics, principles and practice
  Foley, Van Dam, Feiner, Hughes - *Addison Wesley*
- Advanced Animation and Rendering Techniques,
  Alan & Mark Watt, *Addison Wesley*
- Géométrie discrète en analyse d 'images
  J.M Chassery & A. Montanvert – *Hermes*

Extension :

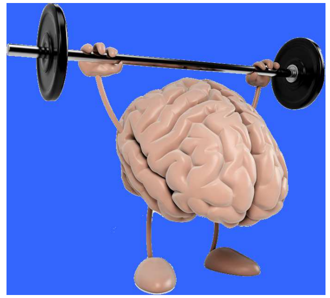
- Modélisation et construction de surfaces pour la CFAO,
  J.C. Léon, *Hermes*

Programing - Libraries

Dan Sunday http://geomalgorithms.com/

- CGAL (The Computational Geometry Algorithms Library)
- VCG (The Visualization and Computer Graphics Library)
- Geometric-tools (www.geometrictools.com)
- GLM (OpenGL Mathematics)
- Boost (polygon)

Ce qu'il faut retenir



Pour résoudre des problèmes géométriques il faut savoir **reconnaitre, retrouver, (choisir) et utiliser** :

- Le produit vectoriel/scalaire :
  - Pour calculer le sens de rotation, la visibilité, la convexité, les modèles d'éclairage, la localisation…
- Les 3 représentations d'une courbe :
  - Pour la discrétisation, le calcul d'intersection, l'animation sur une courbe , la localisation…
- Les propriétés des polygones (connexité, convexité, ensembles réguliers)
  - Savoir reconnaitre, corriger ou adapter…
- Les structures de données & algorithmes
  - Choisir la structure en fonction des propriétés, écrire une structure de données, un algo de parcours…
- Les collisions
  - Choisir le type de volume englobant ou le type de traitement en fonction de l'application, du problème posé…
- Les opérations booléennes
  - Quand et comment elles s'appliquent…
- Transformations, coordonnées homogènes :
  - Savoir reconnaitre et retrouver une transformation (matrice ou complexes), dans un système cartésien ou en coordonnées homogène…

Exercices

1. Calcul de l'aire d'un polygone courbe
2. Algo : is_convex()
3. Algo : is_connex()
4. Find « point dans polygon() » in CGAL doc
5. Jeu du prisonnier (++)
6. Faire le TP n°1 : Point dans polygone sous Scilab

Organisation des travaux pratiques sous Scilab (et Python)

http://people.mines-paristech.fr/olivier.stab/TP_scilab_MG91

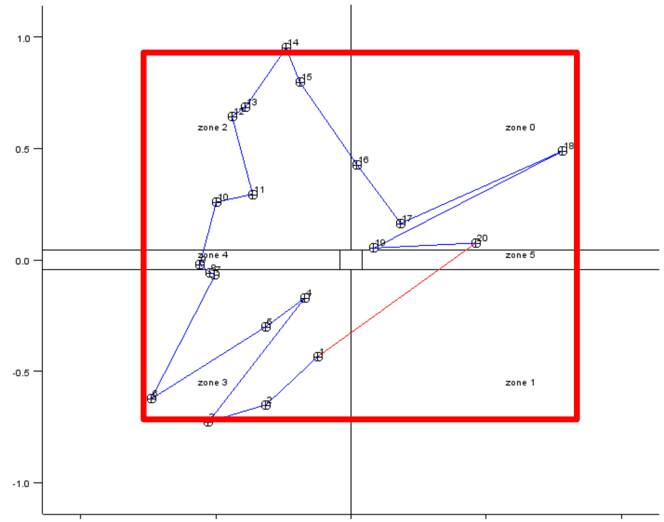
- Données :
  - fichier d'énoncé (sujet_TPx.sci), scripts et exemples.
- Travail :
  - répondre aux questions dans le fichier, sauver les données de test et les figures d'exemple…
- Me rendre :
  - le travail : le répertoire avec les fichiers à la fin de la séance.

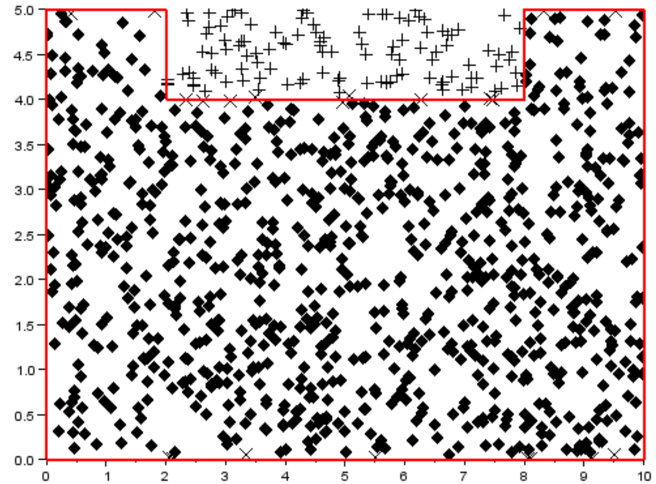
TP 0 : introduction à Scilab (ou à Python)

- Premiers pas
- Vecteurs & matrices
- Graphisme
- IO fichiers
- Scripts & fonctions
- Exercices d'application

TP 1 : Boundary Representation

1. Polygones : MyPoly=[1,2;1,3;...;1,2];
   - Manipulations simples
   - Fonctions élémentaires :
     - Aire, boite d'encombrement
2. Test de point_dans_poly
   - PIP_Demo(XYcoord)
   - [pos]=PIP_PointLocalization(XYcoord, XYP, eps)
     avec pos =(DANS/HORS/SURA/SURS)





Estimation de l'Aire du polygone à partir de point_dans_poly() ?