CALCUL D'INTERSECTIONS

UN SEGMENT DE LIGNE ?

Un segment $[a\ b]$ est défini par ses deux extrémités a et b

Soit $\lambda \in [1,0]$. Tout point p d'un segment $[a\ b]$ est défini par :

$$x = \lambda * a.x + (1 - \lambda) * b.x \quad ||| \quad y = \lambda * a.y + (1 - \lambda) * b.y$$

Une intersection entre deux segments : Un point commun entre les deux

Intersection strictement à l'intérieur des deux segments : intersection propre

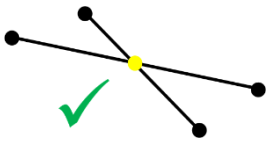
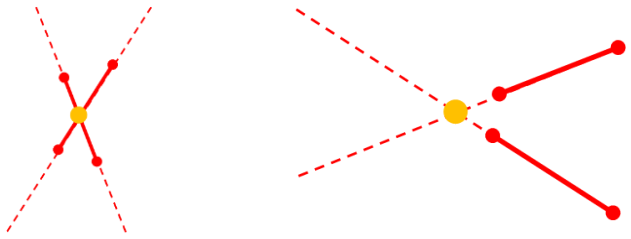


Figure 1 Intersection propre

INTERSECTION DE 2 SEGEMENTS

1. **Calcul de l'intersection des deux supports :**
   Système linéaire de deux équations à deux variables.
2. **Comparer la solution avec les deux extrémités :**
   i. A l'intérieur : Il existe une intersection
   ii. Sinon : Intersection vide



- Complexité : $O(1)$
- Cas dégénéré : L'intersection est une ligne ?

LE BALAYAGE GEOMETRIQUE

PRINCIPE DE L'ALGORITHME :

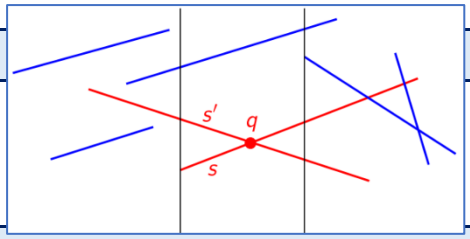
1. Une ligne verticale ($l_t : x = t$) traverse le plan de gauche à droite et affiche la scène :
   i. $State_t$ est l'ensemble des Segments en intersection avec l_t
   ii. *Les extrémités des segments sont triées par leurs ordonnées y suivant leurs intersections avec l_t*
2. A chaque évènement, correspond un traitement :
   i. *Un traitement pour les débuts de segments ;*
   ii. *Un traitement pour les fins de segments ;*
   iii. *Un traitement pour les intersections.*

DEFINITION DES CAS DEGENERES :

1. 2 segments collinaires
2. Deux points avec la même abscisse (x)
3. 3 segments qui se coupent dans le même point

CONSTATATION GEOMETRIQUE

S'il existe une intersection entre deux segments, ces deux segments sont adjacents dans la liste $State_t$ (L'état strictement antérieur à l'intersection)



FORMULATION DE L'ALGORITHME

1. *Sort the ends in Event* -------- $O(n \log(n))$ // n : Nombre de segments
2. *State* ← {} ---------- $O(1)$
3. ***While** Event **isNotEmplty*** -------- $O(n + m)$ // m : Nombre d'intersections
   i. $P \leftarrow MIN(Event)$ -------- $O(1)$
   ii. *Remove P from Event* -------- $O(1)$
   iii. *P is the start of S*
      ✓ $State \leftarrow State + S$ -------- $O(k)$ // k : Taille de State ; Maintien de l'ordre dans la liste
      ✓ $S \cap S_l = P_l$, $S \cap S_r = P_r$
      ✓ $Event \leftarrow Event + \{P_l , P_r\}$ -------- $O(n + m)$; Maintien de l'ordre dans la liste
   iv. *P is the end of S*
      ✓ $State \leftarrow State - \{S\}$ -------- $O(k)$
      ✓ $S_r \cap S_l = P_i$ -------- $O(1)$
      ✓ *si* $x(P_i) \geq x(P)$
         $Event \leftarrow Event + \{P_i\}$ -------- $O(n + m)$
   v. $P \leftarrow S_g \cap S_d$
      ✓ *Tag P* -------- $O(1)$
      ✓ *Reverse S_l and S_d in State* -------- $O(1)$
      ✓ $S_{lr} \cap S_r = P_1$, $S_{rl} \cap S_l = P_2$ -------- $O(1)$
         $Event \leftarrow Event + \{P_1 , P_2\}$ -------- $O(n + m)$
4. ***EndWhile***