



R my Malgouyres,
Algorithmes pour la synth se d'images
et l'animation 3D,
DUNOD, 2002

TP n  3 : Fen trage de polygones 2D

Objectifs :

Le but de ce TP est d'impl menter l'algorithme de fen trage de polygones vu en cours et en TD.

1) Travail   r aliser :

On souhaite rajouter au programme r alis  au TP 1 et TP2 une commande *Ins re/R gion de fen trage convexe* qui permet   l'utilisateur de d finir les sommets d'une r gion convexe   la souris. On rajoutera aussi une commande *Ins re/Polygone   intersecter* qui permet   l'utilisateur de d finir un polygone convexe. Ensuite, le programme affiche seulement les ar tes du polygone, et remplit l'intersection du polygone avec la r gion pr alablement d finie.

2) Classes n cessaires   l'algorithme :

Nous donnons ici les classes qui doivent  tre mises dans un fichier *Fenetrage.h*. Le but du TP est d'effectuer l'impl mentation des fonction, qui doit  tre mise dans un fichier *Fenetrage.cpp*, puis de tester ces fonctions.

a) La classe *AreteFenetre* :

```
class AreteFenetre
{
    double xa, ya, xb, yb;

public:
    AreteFenetre();
    AreteFenetre(double x1, double y1, double x2, double y2);

    bool interieur(double x, double y);
    void intersection(double xp, double yp, double xq, double yq,
                     double & xi, double & yi);
    void cliparete(double * InputX, double * InputY, int nbsommin,
                  double * OutputX, double * OutputY, int & nbsommout);
};
```

Remarquons que,   la diff rence des ar tes d finies dans les pr c dents TP, nous travaillons cette fois en virgule flottante (variables de type double).

La fonction `interieur` permet de déterminer si un point (x, y) se trouve dans le demi-plan de gauche de l'arête.

La fonction `intersection` permet de calculer les coordonnées x_i et y_i de l'intersection d'un segment $[P, Q]$ avec une arête $[A, B]$.

La fonction `cliparete` permet de calculer en sortie (Output) l'intersection d'un polygone en entrée (Input) avec le demi-plan gauche de l'arête de région.

b) La classe `RegionFenetre` :

```
class RegionFenetre
{
    AreteFenetre *tabarettes;
    int nbarettes;

public:
    // constructeurs :
    RegionFenetre();
    RegionFenetre(RegionFenetre & R);
    RegionFenetre(Pixel *t, int nb);

    ~RegionFenetre(); // destructeur

    // opérateur d'affectation
    RegionFenetre & operator=(RegionFenetre & R);

    // fonction de fenêtrage
    void clipregion(double * InputX, double * InputY, int nbsommin,
                   double * OutputX, double * OutputY, int & nbsommout);
};
```

Cette classe permet de définir une région de fenêtrage comme un polygone convexe dont les arêtes sont représentées comme des éléments de la classe `AreteFenetre`.

La fonction `clipregion` permet de calculer en sortie (Output) l'intersection d'un polygone en entrée (Input) avec la région.

c) Cas d'une fenêtre rectangulaire :

On implémentera la fonction suivante :

```
void cliprectangle(double * InputX, double * InputY, int nbsommin,
                  double * OutputX, double * OutputY, int & nbsommout,
                  double xmin, double xmax, double ymin, double ymax);
```

qui permet de calculer en sortie (Output) l'intersection d'un polygone en entrée (Input) avec une région rectangulaire définie par $x_{min} \leq x \leq x_{max}$ et $y_{min} \leq y \leq y_{max}$.