



Rémy Malgouyres,  
Algorithmes pour la synthèse d'images  
et l'animation 3D,  
DUNOD, 2002

## TP n° 2 : Remplissage de polygones convexes

### Objectifs :

Le but de ce TP est d'implémenter l'algorithme de remplissage de polygones convexes vu en cours et TD.

### 1) Travail à réaliser :

On souhaite rajouter au programme réalisé au TP 1 (qui portait sur le tracé de droites) une commande *Insere/polygone plein* qui permet à l'utilisateur de définir les sommets d'un polygone à la souris, puis qui affiche les points intérieurs au polygone avec une certaine couleur.

### 2) Classes nécessaires au remplissage :

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

a) On retrouve la classe *Arete* :

```
class Arete
{
    friend class TAconv;
    friend class TAAconv;
    friend class Polygone2d;

    int xbas, ybas, xhaut, yhaut; // ybas doit toujours être inférieur à yhaut
public:
    // Constructeur initialisant les données :
    Arete(int x1, int y1, int x2, int y2);
    Arete(void) // Constructeur par défaut

    // Dessin du segment dans un device context :
    void dessiner(CDC* pDC, COLORREF couleur);
};
```

Par rapport au TP 1, on a simplement rajouté des classes amies (qui seront définies plus loin), pour que celles-ci puissent accéder aux données membres privées de la classe.

b) La classe *EdgeData* :

```

class EdgeData
{
    friend class TAconv;
    friend class TAAconv;
    friend class Polygone2d;

    int yhaut;
    int x;
    int numerat;
    int denomin;
};

```

Correspond au codage d'une arête du polygone dans la table des arêtes globale et dans la table des arêtes active. Cette classe n'a pas de fonction membre et 3 classes sont déclarées amies pour qu'elles puissent accéder aux données membres privées de la classe EdgeData.

c) La classe TAconv (table des arêtes globale) :

```

class TAconv
{
    friend class Polygone2d;

    int taille; // maxybas - minybas + 1
    int * nombres; // pour chaque i, nombre d'arêtes ayant ybas = i
    EdgeData **tab;

    // Constructeur (initialisation) :
    TAconv(Arete * tabarettes, int nbarettes, int minybas, int maxybas);
    ~TAconv(); // destructeur

    void trier(void); // fonction utilisée dans le constructeur
};

```

La taille de la table des arêtes globale est égale à 1 plus la plus grande valeur de *ybas* moins la plus petite valeur de *ybas*. Cela correspond à toutes les hauteurs où une arête est susceptible de commencer (c.a.d. d'avoir son *ybas*).

Le tableau *nombres* représente, pour chaque hauteur *i* avec  $0 \leq i < \text{taille}$ , le nombre d'arêtes ayant son *ybas* égal à *i*. Le polygone étant convexe, pour chaque *i*, la valeur *nombres*[*i*] doit être inférieure ou égale à 2 car il y a au plus deux arêtes sur une même horizontale.

Le tableau *tab* est un tableau d'EdgeData de dimensions *taille* \* 2 qui contient au plus deux arêtes pour chaque horizontale. L'arête *tab*[*i*][*j*], pour  $i < \text{taille}$  et  $0 \leq j \leq \text{nombres}[i]$ , code la *j<sup>ième</sup>* arête qui a son *ybas* égal à *i*. Lors de la construction du tableau *tab*, il faut prendre garde de ne pas introduire les arêtes horizontales du polygone.

Les tableaux *nombres* et *tab* doivent évidemment être alloués dans le constructeur, et libérés dans le destructeur.

Pour un *i* fixé, si *nombres*[*i*] est égal à 2, les deux arêtes *tab*[*i*][0] et *tab*[*i*][1] doivent être rangées de gauche à droite; c'est le but de la fonction membre *trier*, qui trie les (au plus) deux arêtes sur chaque horizontale.

d) La classe TAAconv (table des arêtes actives) :

```

class TAAconv
{
    friend class Polygone2d;

    EdgeData table[2];

    void dessinspixels(int y, CDC *pDC, COLORREF couleur);
    void miseajour(int increment[2]);
};

```

Dans le cas de polygones convexes, on a deux arêtes sur chaque horizontale. Ces deux arêtes, ainsi que les variables nécessaires à leur parcours, sont stockées dans un tableau de 2 `EdgeData` nommé `table`.

La fonction membre `dessinspixels` permet d'afficher les pixels à une hauteur  $y$  qui sont compris entre `table[0].x` et `table[1].x`, qui sont les deux points des arêtes actives à la hauteur  $y$ .

Le paramètre `pDC` est un pointeur sur le *device context* et le paramètre `couleur` donne la couleur de l'affichage.

La fonction `miseajour` réalise la mise à jour des données `x` des deux `EdgeData` de la table des arêtes actives, ainsi que des deux variables entières `incrément` nécessaires au parcours des deux arêtes actives.

e) La classe `Polygone2d` (qui code la donnée d'un polygone) :

```

class Polygone2d
{
    int nbaretes; // nombre d'arêtes du polygone
    Arete * tabaretes; // tableau des arêtes du polygone

    int calcminy(void); // calcul de la plus petite valeur de ybas
    int calcmaxybas(void); // calcul de la plus grande valeur de ybas
    int calcmaxyhaut(void); // calcul de la plus grande valeur de yhaut

public :
    Polygone2d(void); // constructeur par défaut
    Polygone2d(Pixel * tabsommets, int nb); // constructeur
    ~Polygone2d(); // destructeur

    void remplissage(CDC*pDC, COLORREF couleur); // fonction de remplissage
};

```

Le constructeur ayant en paramètre un nombre d'arêtes et un tableau de pixels doit créer le tableau `tabaretes` des arêtes du polygone (après allocation de ce dernier). Le tableau `tabaretes` doit être libéré dans le destructeur.