



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

## TP n° 1 : Tracé de droites dans le plan

### Objectifs :

Le but de ce TP est d'implémenter l'algorithme dit "du point milieu" permettant d'afficher une droite sur un écran composé de pixels. On verra aussi comment utiliser des bitmaps sous Visual C++, et comment construire une petite application graphique interactive basée sur des événements du type "click de souris" en utilisant les classes MFC de Visual C++.

### 1) L'application à réaliser :

On souhaite réaliser une application permettant à un utilisateur de définir et de visualiser dans une fenêtre graphique des segments de droites et des polygones.

Pour définir une droite, l'utilisateur dispose d'un menu *Insère* avec une commande *Segment*. Après avoir activé cette commande, l'utilisateur clique en deux points de la fenêtre avec le bouton gauche de la souris, et le segment entre ces deux points est affiché avec l'algorithme du point milieu.

Pour définir un polygone, l'utilisateur dispose d'une commande *Polygone vide* dans le menu *Insère*. Après avoir activé cette commande, l'utilisateur clique sur un certain nombre de points avec le bouton gauche de la souris, puis clique sur le dernier point avec le bouton droit. Le polygone tracé est affiché au fur et à mesure dans la fenêtre.

### 2) Création de l'application :

Lancer Visual C++ et faire *Fichier, Nouveau...* puis sélectionner *Projet MFC AppWizard (exe)*. Donner le nom "tp2d" au projet et le placer à l'emplacement désiré dans votre répertoire. Sélectionner l'option *Single Document* et prendre toutes les autres options par défaut.

### 3) L'utilisation de la classe CBitmap :

L'utilisation de la classe `CBitmap` permet de construire une image en mémoire par différents algorithmes, puis de l'appliquer instantanément dans une fenêtre graphique au moment voulu en utilisant la fonction `BitBlt`. Cette technique est très utilisée en synthèse d'images.

Pour cela, définir dans la classe `Ctp2dView` (classe vue du projet dans le fichier `tp2dView.h`) deux données membres :

```
CBitmap mybitmap; // bitmap GDI  
CDC* mypdc; // pointeur sur un device context
```

Utiliser ensuite *Class Wizard* pour ajouter à la classe `Ctp2dView` l'événement *OnInitialUpdate*. Remplir ensuite le code de la fonction `OnInitialUpdate` dans le fichier *tp2dView.cpp* avec le code suivant :

```
void Ctp2dView::OnInitialUpdate()
{
    CView::OnInitialUpdate();

    // TODO: Add your specialized code here and/or call the base class

    mypdc = new CDC; // allocation du pointeur sur CDC
    CClientDC clientdc(this); // définition d'un device context compatible
    mypdc->CreateCompatibleDC(&clientdc); // Création d'un espace mémoire
    mybitmap.CreateCompatibleBitmap(&clientdc,1000,700); // Création du bitmap
    mypdc->SelectObject(mybitmap); // Sélection du bitmap dans le device context
    for (int i = 0 ; i < 1000 ; i++) // initialisation
        for (int j = 0 ; j < 700 ; j++) // du bitmap avec
            mypdc->SetPixelV(i,j,RGB(255,255,0)); // un fond jaune
}
```

Enfin, dans la fonction `OnDraw` de la classe vue, ajouter la ligne suivante qui permet d'appliquer le bitmap dans la fenêtre graphique :

```
pDC->BitBlt(0,0, 1000, 700, mypdc,0,0,SRCCOPY);
```

Exécutez maintenant le programme; la fenêtre graphique doit maintenant afficher un fond jaune. Ces préliminaires étant achevés, nous pouvons maintenant passer au tracé de segments proprement dit.

#### 4) Implémentation de l'algorithme de tracé de droites :

Nous devons tout d'abord définir une classe C++ pour représenter un segment de droite. La structure que nous proposons pour cette classe, que nous nommerons "Arete", est la suivante :

```
class Arete
{
    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);
};
```

Le segment de droite est représenté par ses deux extrémités :  $(x_{bas}, y_{bas})$  et  $(x_{haut}, y_{haut})$ , le point  $(x_{haut}, y_{haut})$  ayant une plus grande deuxième coordonnée que  $(x_{bas}, y_{bas})$ . Dans l'implémentation du constructeur ayant quatre paramètres, un test doit donc être réalisé pour savoir si  $y_1$  est inférieur à  $y_2$  ou le contraire.

C'est dans l'implémentation de la fonction membre `dessiner` que l'on programme l'algorithme du point milieu permettant de dessiner le segment. Dans la fonction `dessiner`, pour afficher un pixel  $(x, y)$  avec la couleur `couleur`, on utilise l'instruction suivante :

```
pDC->SetPixelV(x, y, couleur);
```

## 5) Définir un segment de manière interactive :

Pour définir de manière interactive un segment de droite en spécifiant ses extrémités comme indiqué au 1), procéder comme suit.

a) Ajouter les données membres suivantes à la classe `Ctp2dView` :

```
CPoint pointpreced;
bool segmentenconstruction;
bool premierpoint;
Arete segment;
```

- Le booléen `premierpoint` indique si un premier point a déjà été défini.
- Le booléen `segmentenconstruction` indique si un segment est en cours de construction suite à l'activation de la commande *Insère/Segment*. Ce booléen doit être initialisé à `false` dans le constructeur de la classe vue.
- Le point `pointpreced` est égal au premier point qui a été défini lorsque le booléen `premierpoint` est à `false`.
- L'élément `segment` de la classe `Arete` représente le segment défini et est initialisé avec le constructeur d'arêtes sitôt les deux extrémités du segment connues.

b) Ajouter aux ressources de l'application un menu *Insère* avec une commande *Segment* ayant `ID_INSERTE_SEGMENT` pour *ID*. Utiliser *Class Wizard* pour ajouter à la classe `Ctp2dView` une fonction membre `OnInsereSegment` correspondant à la commande `ID_INSERTE_SEGMENT`.

c) Dans la fonction `OnInsereSegment` de la classe vue, on doit simplement initialiser les variable `premierpoint` et `segmentenconstruction`.

d) En utilisant *Class Wizard*, ajouter à la classe vue une fonction pour gérer l'événement `WM_LBUTTONDOWN` (click sur le bouton gauche de la souris). Dans cette fonction `OnLButtonDown`, on doit tester si un segment est en cours de construction et si un premier point a déjà été défini. Si on en est au deuxième point, on initialise l'arête `segment`, puis on le dessine en utilisant la fonction membre de la classe `Arete` définie au 4). Il faut ensuite rafraîchir la fenêtre graphique en appelant la fonction `OnDraw` de la classe vue.

## 6) Définir un polygone de manière interactive :

Il faut procéder de manière analogue au cas d'un segment, en ajoutant cette fois à la classe vue les données membres suivantes :

```
Pixel inputpolygon[100]; // tous les sommets du polygone
int nbaretes; // nombre d'arêtes définies
bool polygonevideenconstruction;
```

et en ajoutant aussi à la classe vue une fonction pour gérer le click sur le bouton droit de la souris.