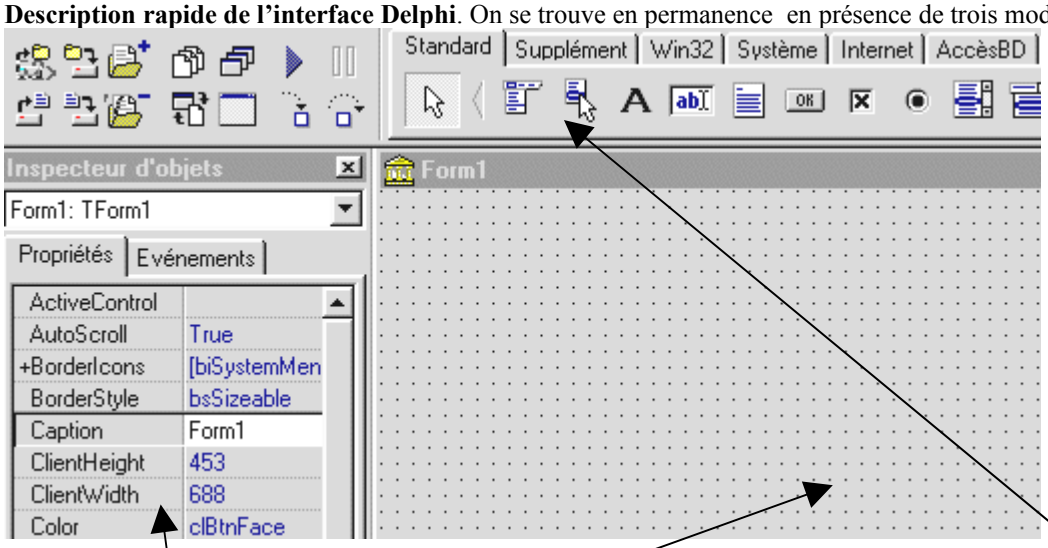
Création d'un outil de dessin avec DELPHI (pascal de BORLAND)

Création du répertoire de TRAVAIL ;
Commencer par créer un répertoire dans le répertoire DELPHI. Lancer l'Explorateur, Activer le répertoire DELPHI puis dans le menu ***Fichiers*** *de l'explorateur demander* ***Nouveau Dossier****. L'explorateur créé immédiatement un sous dossier de DELPHI que l'on nomme* ***DESSIN01*** *(il suffit en fait de remplacer* ***Nouveau dossier*** *par* ***DESSIN01****). Ce répertoire contiendra le fichier exécutable que nous allons créer.*

Description rapide de l'interface Delphi. On se trouve en permanence en présence de trois modules :



- Le module principal **Delphi 3** (en haut) contenant le **menu déroulant** et toutes les **barrettes d'outils**. C'est lui qui permet de décider des composants à placer sur la fiche et de **compiler** ou **exécuter** l'application.
  - La **fiche de travail** (fiche vierge au départ nommée **Form1**) qui est double :
    - Partie visuelle telle qu'elle apparaît au yeux de l'utilisateur
    - Partie texte dans lequel Delphi écrit lui-même le code pascal et nous laisse la possibilité de travailler sur ce code.
- **L'inspecteur d'objet**, propre a chaque objet créé sur la fiche (bouton par exemple) et contient deux onglets :
  - Les **propriétés** de l'objet nous permettent de modifier sa taille, sa couleur, etc.
  - Les **événements** qui permettent d'écrire les procédures qui doivent s'exécuter lors de tel ou tel événement (clic, touche, fermeture de fenêtre etc.)

Nous allons apprendre à travailler avec ces trois modules et voir avec quelle facilité il devient possible de développer une véritable application Windows. La programmation avec Delphi est **orientée objets** c'est à dire que le programme n'a pas de début (ni de fin) mais est constitué d'un ensemble d'objets qui réagissent aux événements (clic de souris, appui sur une touche, déplacement de souris etc.)

a- Ouverture et nom de la fiche de travail

Pour créer une interface utilisateur nous partons d'une fiche vierge nommée **Form1**.
Dans le menu **Fichiers** demandez **Nouvelle application**. Une fiche (**Form1**) et une unité (**Unit1.pas**) s'ouvrent.
Nous allons commencer par personnaliser le nom de cette fiche :

- Dans les **propriétés** de **l'inspecteur d'objets** remplacer dans le champ **caption Form1** par **Dessin**. La fiche vierge se nomme désormais **Dessin**.
- Régler la couleur (**color**) de la fiche sur **ClWhite** ;

Utiliser le menu **Fichiers-Enregistrer le projet sous,** placez vous dans le répertoire **\DELPHI\DESSIN01,** enregistrer l'unité1 (**unit1.pas**) en lui laissant son nom et enregistrer le projet (**project1.dpr**) sous le nom de **DESSIN.DPR** par exemple. *Lorsque l'on relance DELPHI c'est ce fichier projet qu'il faut ouvrir.*
Le fait d'enregistrer le projet immédiatement permet de définir le répertoire **\DELPHI\DESSIN01** comme répertoire par défaut.

b- Cadre de travail

Dans la **barrette Standard** cliquez sur le dernier composant **Panel**. Cliquez alors sur la fiche vierge pour placer un **fond pour les boutons de l'application**.
Dans les **propriétés** de **l'inspecteur d'objets** réglez le champ **Align** sur **AlLeft**. Le cadre de travail occupe alors toute la fiche en hauteur et se cale sur la gauche. C'est sur ce cadre que nous allons placez les boutons de l'application.
Dans les **propriétés** de **l'inspecteur d'objets** réglez le champ **width** sur **50** et effacez le champ **Caption** (qui doit contenir **Panel1**) sinon le mot **Panel1** reste affiché même pendant l'exécution.
Dans la **barrette Supplément** cliquez sur le composant **Image**. Cliquez alors sur la fiche vierge pour placer un composant **Timage** qui sera le fond de dessin.
Dans les **propriétés** de **l'inspecteur d'objets** réglez le champ **Align** sur **AlRight**. L'image occupe alors toute la fiche en hauteur et se cale sur la droite.
Dans les **propriétés** de **l'inspecteur d'objets** réglez le champ **width** sur **638** et effacer le champ **Name** (qui doit contenir **Image1**) pour le remplacer parle mot **Fond**.

Il nous reste maintenant à placer sur notre **Panel1** tous les boutons qui vont nous permettre de dessiner.

c- Le dessin sur la fiche
Si vous exécutez votre application dans l'état (en appuyant sur **F9**) rien ne se passe. Nous n'avons rien prévu pour gérer les évènements de la souris !
Refermez l'application (si vous l'avez lancée !).
Nous allons gérer maintenant le déplacement de la souris.
Dans les évènements de **Fond** (le composant **Timage**) Double-cliquez dans la case qui se trouve à droite de l'événement **OnMouseMove**.
Delphi fabrique alors le squelette de la procédure associée au déplacement de la souris. Il nous reste à remplir cette procédure pour indiquer à l'application ce qu'elle doit faire quand la souris se déplace sur le fond de dessin.
Le squelette de la procédure **OnMouseMove** à la forme suivante

```
procedure TForm1.FondMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin

end;
```

Les variable X et Y contiennent les coordonnées de la souris. Nous allons pouvoir les utiliser pour dessiner sur la fiche.
Dans la procédure **FondMouseMove** ajoutez l'instruction **Fond.Canvas.LineTo(X,Y) ;**
Canvas représente la surface associée à **Fond** sur laquelle on va pouvoir dessiner. La procédure **LineTo** permet de relier un point au précédent par un segment.
Votre procédure devient alors :

```
procedure TForm1.FondMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
 Fond.Canvas.LineTo(X,Y);
end;
```

REMARQUE : la touche F12 permet de basculer entre la fiche sur laquelle vous avez placé les composants (panel, boutons etc.) et l'unité dans laquelle vous écrivez du code pascal (essayez).

Chaque mouvement de la souris va donc relier à l'emplacement précédent par un petit segment.
*Vérifier en lançant l'application (**F9**).*
Il serait préférable de dessiner sur le fond seulement quand le bouton gauche de la souris est enfoncé. Nous allons rajouter une variable (ENFONCE) qui aura la valeur VRAI quand le bouton sera enfoncé.
Dans la partie **Public** de **Form1** ajoutez la déclaration de variable **ENFONCE : BOOLEAN ;**
(Un booléen peut prendre seulement deux valeurs : VRAI ou FAUX).
Il faut commencer par **initialiser** la variable ENFONCE à FALSE ;

*Dans **l'inspecteur d'objets** déroulez la liste pour activer **Form1**.*
Dans les **événements** de **Form1** définissez la procédure **OnActivate** par un Double-Clic. Cette procédure s'exécutera systématiquement à l'ouverture de la fiche.
Garnissez la procédure **OnActivate** en ajoutant la ligne : **ENFONCE := FALSE ;** vous obtenez alors la procédure :

```
procedure TForm1.FormActivate(Sender: TObject);
begin
 ENFONCE := FALSE;
```

```
end;
```

Transformez alors la procédure **FondMouseMove** pour ne dessiner que lorsque ENFONCE est VRAI :

```
procedure TForm1.FondMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  IF ENFONCE THEN  Fond.Canvas.LineTo(X,Y);
end;
```

Si vous lancez l'application en l'état il ne se passera plus rien. ENFONCE prend la valeur FALSE lorsque la Form1 est activée et les déplacements de la souris ne laissent une trace que si ENFONCE est TRUE ! ! ! !
Il nous reste à mettre ENFONCE à TRUE quand on enfonce le bouton gauche (**OnMouseDown**) et à FAUX quand on le relâche (**OnMouseUp**).
Dans la **liste** de **l'inspecteur d'objets** activez **Fond** et fabriquez les deux procédures :

```
procedure TForm1.FondMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 ENFONCE := TRUE;
end;
procedure TForm1.FondMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 ENFONCE := FALSE;
end;
```

*Lancez l'application (**F9**) et dessinez* : tout fonctionne bien : quand on clique on peut dessiner et le dessin s'arrête dés que l'on relâche. Seul petit problème le **LineTo** relie au dernier point dessiné ce qui provoque des segments parasites à chaque clic de souris.
Pour corriger ce problème il faut replacer le pointeur de dessin dans la procédure **FondMouseDown** en rajoutant l'instruction **Fond.Canvas.MoveTo(X,Y) ;** Le pointeur de dessin se placera alors à l'emplacement où s'est produit le clic avant de commencer à dessiner.

Remarque : pour **ouvrir** votre projet faites **Fichier – Ouvrir \DESSIN01\DESSIN.DPR**.

d- Changer de couleur

Nous allons fabriquer un bouton qui va permettre de choisir la couleur de tracé. Placez vous sur la fiche (**F12**).
Dans la barrette **Dialogues** cliquez sur le composant **ColorDialog** et placez en un **n'importe où** sur votre fiche. Un tel composant demeure **invisible** lors de l'exécution.
Nous allons maintenant **créer un bouton** qui va **appeler** le composant **ColorDialog1** que l'on vient d'ajouter à la fiche.
Dans la barrette **Supplément** cliquez sur **SpeedButton** et poser un bouton sur le **Panel1** (en haut par exemple).
Réglez ses dimensions à : **Height** : 42 ; **Left** : 4 ; **Top** : 6 ; **Width** : 42.
Choisissez alors un **dessin** pour le bouton en cliquant sur la propriété **Glyph…** Dans le répertoire **\Images\Buttons** on pourra prendre le dessin **Brush.BMP**.
Votre bouton doit alors avoir une véritable allure professionnelle.
Il s'agit maintenant de décider de l'action à faire lorsque l'utilisateur va cliquer sur le bouton.
Dans les événements de votre bouton Double-cliquez sur l'événement **Onclick** et remplissez la procédure **SpeedButton1Click** en ajoutant l'instruction : **colordialog1.Execute;** Le clic sur ce bouton provoquera l'exécution du composant **ColorDialog1** qui ouvrira une boîte de dialogue dans laquelle vous pourrez choisir la couleur.
Remarque : un **SpeedButton** est un bouton qui n'est jamais actif ou inactif mais qui déclenche une opération (ici ouvrir une fenêtre de dialogue pour la couleur).

Exécuter votre application (F9) pour vérifier le bon fonctionnement du bouton.

Le bouton fonctionne bien mais la couleur du tracé n'est pas changée. Il faut rajouter l'instruction **Fond.Canvas.Pen.color:=ColorDialog1.Color;** dans la procédure **FondMouseDown**. Cette instruction doit être ajoutée **avant** l'instruction **Fond.Canvas.MoveTo(X,Y)** ;

Exécuter alors votre application et dessinez ! ! !

N'oubliez pas de faire **Fichier - Tout enregistrer** régulièrement.

e- Dessiner des formes prédéfinies

Il faut maintenant rajouter des boutons permettant de dessiner des formes prédéfinies (rectangle, cercle etc.) .
Commencer par ajouter dans **Unit1.pas** dans la partie **Public** de **Form1** trois variables :
FORME : INTEGER ;
Old_X, Old_Y : INTEGER ;
Suivant la valeur de la variable FORME on dessinera un objet différent : si FORME est égale à zéro on dessinera une ligne (c'est le cas actuellement) si elle est égale à 1 on dessinera un rectangle et si elle est égale à 2 une ellipse.
Les variables **Old_X** et **Old_Y** permettront de mémoriser les valeurs de X et Y lors de l'appui sur le bouton gauche. On pourra ainsi les réutiliser pour définir le coin en haut à gauche du rectangle.
Commencez par ajouter la ligne **FORME := 0 ;** dans la procédure **FormActivate** de façon à être sûr que cette valeur soit la valeur par défaut lors du lancement de l'application.
Ajoutez les deux instructions **Old_X := X ;** et **Old_Y := Y ;** dans la procédure **FondMouseDown** pour mémoriser les valeurs de X et Y lors de l'appui sur le bouton gauche de la souris.
Il faut maintenant construire deux boutons : le premier pour dessiner une ligne quelconque et le second pour dessiner un rectangle. Réactivez la fiche (**F12**) .
Dans la barrette **Supplément** cliquez sur le bouton **BitBtn** et cliquez sur **panel1** pour placer le bouton.
Réglez ses dimensions à : **Height** : 42 ; **Left** : 4 ; **Top** : 56 ; **Width** : 42.
Mettez le champ **Name** à **Ligne**.
Charger le **Glyph** avec **\images\buttons\Pencil.BMP**.
Mettez le **Layout** à **BlGlyphTop**.
Faites un Double-clic sur le bouton pour créer la procédure **LigneClick**. Remplir cette procédure avec l'instruction **FORME := 0 ;** . Lors du clic sur ce bouton la variable FORME prendra la valeur Zéro.
Il faut maintenant créer un autre bouton pour mettre FORME à 1 et faire des rectangles.
Procédez comme vous venez de le faire pour le bouton Ligne.
On règlera le champ **TOP** à **106** pour un espacement régulier.
Prendre le même **Glyph** et nommez le bouton **Rect** (champ **Name**).
N'oubliez pas de définir la procédure **RectClick** (en double-cliquant sur le bouton) et de lui ajouter l'instruction **FORME :=1 ;** .
Il faut maintenant ajouter les instructions qui permettent de dessiner des formes différentes suivant la valeur de la variable FORME.
Il suffit d'intervenir sur la procédure **FondMouseMove** de la façon suivante :

```
procedure TForm1.FondMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
 IF ENFONCE THEN
  BEGIN
     IF FORME = 0  THEN Fond.Canvas.LineTo(X,Y) ;
     IF FORME = 1  THEN Fond.Canvas.rectangle(Old_X,Old_Y,X,Y);
  END;
end;
```

Lancez votre programme et essayer.

Premier problème les formes Rectangles sont pleines, on préfère qu'elles soient vides bien entendu !. Il suffit de rajouter la commande **Fond.Canvas.Brush.Style:=BsClear;** dans la procedure **FormActivate**.
Réessayez votre programme.
Second problème (plus compliqué celui-là) les formes s'affichent toutes mais ne s'effacent pas ! !
Première étape : il faut se mettre en mode crayon-fond : ce mode permet de mélanger les couleurs du crayon et du fond de manière à ne jamais effacer le fond. Pour ce faire ajoutez l'instruction **Fond.Canvas.Pen.mode:=PmNotXor;** dans la procédure **FondMouseDown.** Dans le mode **NotXor** un second tracé replace le fond et donc efface le premier tracé.
Seconde étape : il faut mémoriser les valeurs de X et Y pour retracer le rectangle à sa précédente position de façon à ce qu'il s'efface. Nous allons utiliser deux variables, X_prec et Y_prec pour mémoriser les valeurs de X et Y dans la procédure **MouseMove** de façon à pouvoir les réutiliser pour effacer l'ancien rectangle avant d'en retracer un nouveau.
Dans la partie **Public** de **Form1** ajoutez les deux variables **X_prec, Y_prec : INTEGER ;**
Troisième étape : modifier la procédure **FormMouveMove** : il faut mémoriser les valeurs de X et Y à la fin de la procédure et il faut afficher le rectangle deux fois : une première fois avec X_prec et Y_prec pour l'effacer et une seconde fois avec X et Y pour le tracer. La procédure devient alors :

```
procedure TForm1.FondMouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
begin
 IF ENFONCE THEN
  BEGIN
   IF FORME=0 THEN Fond.Canvas.LineTo(X,Y);
   IF FORME=1 THEN
     BEGIN
      Fond.Canvas.rectangle(Old_X,Old_Y,X_prec,Y_prec);
      Fond.Canvas.rectangle(Old_X,Old_Y,X,Y);
     END;
END;
  X_prec:=X;
  Y_prec:=Y;
end;
```

Quatrième étape : il faut **initialiser** les variables **X_prec** et **Y_prec**. Dans la procedure **FondMouseDown** ajoutez les instructions **X_prec :=X ;** et **Y_prec :=Y ;** . Si cette initialisation n'est pas faite les variables X_prec et Y_prec ont une valeur non définie au **premier passage** dans **FondMouseMove** ; Ce qui peut provoquer une erreur d'effaçage dans certains cas.
Cinquième étape : il faut retracer le rectangle en mode normal (**PmCopy**) dans **fondMouseUp**. Ajouter les instructions suivantes dans **fondMouseUp** :

```
Fond.Canvas.Pen.mode:=PmCopy;
IF FORME = 1 THEN Fond.Canvas.Rectangle(Old_X,Old_Y,X,Y) ;
```

Relancez votre application et essayez.

f- Essayez seul d'ajouter un bouton pour tracer des ellipses

En procédant comme pour le bouton **Rect** ajouter un **nouveau bouton** pour tracer des ellipses.
Seule indication : la fonction **Ellipse** de Delphi fonctionne comme la fonction **Rectangle**, il faut lui mettre quatre entiers : **Fond.Canvas.Pen.Ellipse (X0, Y0, X1, Y1)** ; L'ellipse sera tracée dans un rectangle (invisible bien sûr !) dont les coordonnées du coin haut-gauche sont (X0, Y0) et les coordonnées du coin bas-droite sont (X1, Y1).

Vous pouvez également ajouter un bouton **rectangle aux coins arrondis** : l'instruction est **Fond.Canvas.Pen.RoundRect (X0, Y0, X1, Y1, a, b) ;** Les deux entiers **a** et **b** sont là pour donner la grandeur de l'arrondi (a sur les X et b sur les Y). On pourra prendre 20 pour **a** et **b** pour faire un essai.

g- Pot de peinture

Ajouter un bouton **Peinture** (choisissez lui un **Glyph** et appelez le **Peinture**). Le clic sur ce bouton donnera la valeur **4** à la variable **FORME**.
Ajouter alors le bloc suivant dans la procédure **FondMouseDown** :

```
IF FORME=4 THEN
   BEGIN
    Fond.Canvas.Brush.Style:=BsSolid;
    Fond.Canvas.Brush.color:=ColorDialog1.Color;
    Fond.Canvas.FloodFill(X,Y,Fond.Canvas.pixels[X,Y],FsSurface);
    Fond.Canvas.Brush.Style:=BsClear;
   END;
```

On remet la **brosse** du canvas à **Solid** pour remplir les formes. On lui attribue la **couleur** en cours.
La procédure **Fond.Canvas.FloodFill** permet de remplir une surface dont la couleur est **pixels[X,Y]** (c'est à dire la couleur du pixel sur lequel on a cliqué) à partir du point (X,Y) .
On remet ensuite la **brosse** du Canvas à **Clear** pour pouvoir tracer les formes vides (rectangles, etc.).

h- Imprimer
Ajouter un bouton **SpeedButton** (qui déclenche une action) et surcharger sa méthode **OnClick** avec les instructions suivantes :

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
  begin
    Printer.BeginDoc;
    Printer.Canvas.CopyRect(Fond.Canvas.ClipRect,Fond.Canvas,Fond.Canvas.ClipRect);
    Printer.EndDoc;
  end;
```

Attention ! pour que l'objet **Printer** puisse fonctionner il faut ajouter **Printers** dans les **unités** utilisées. Dans la clause **Uses** de l'unité **Unit1.pas** ajoutez l'unité **Printers**.
BeginDoc ouvre l'impression ou met en attente si une autre impression est déjà lancée.
EndDoc ferme l'impression pour libérer l'imprimante.
Printer.Canvas correspond à la feuille de l'imprimante. Vous pouvez dessinez directement dessus. L'instruction Par exemple **Printer.Fond.Canvas.Rectangle**(500,500,1000,1000) dessine un rectangle sur la feuille de l'imprimante. Facile non ?
Dans le cas présent nous utilisons la méthode **CopyRect** (R1, Canv, R2) ;
R1 doit contenir le rectangle de délimitation de la feuille imprimante.
R2 doit contenir le rectangle de notre Canvas écran.
Canv doit indiquer le canvas écran.
En essayant le programme nous remarquons que l'impression de notre écran n'occupe qu'une petite zone de la feuille (suivant la résolution de l'imprimante).
Nous pouvons facilement agrandir cette zone : voici les éléments pour le faire.
Fond.Canvas.ClipRect.Top contient la marge haut du canvas (0 en général)
Fond.Canvas.ClipRect.Left contient la marge gauche du canvas (0 en général)
Fond.Canvas.ClipRect.Right contient la marge droite du canvas (640 en général)
Fond.Canvas.ClipRect.Bottom contient la marge bas du canvas (480 en général).
Pour multiplier les dimensions de votre image sur la feuille il faut donc ajouter quelques instructions :

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
VAR R1 : Trect;
      k   : INTEGER;
begin
  k:=2;
  R1.Top    := k*Fond.Canvas.ClipRect.Top;
  R1.Left   := k*Fond.Canvas.ClipRect.Left;
  R1.Right  := k*Fond.Canvas.ClipRect.Right;
  R1.Bottom := k*Fond.Canvas.ClipRect.Bottom;
  Printer.BeginDoc;
  Printer.Canvas.CopyRect(R1,Fond.Canvas,Fond.Canvas.ClipRect);
  Printer.EndDoc;
end;
```

Vous trouverez les dimensions de la feuille imprimante dans le rectangle : **Printer.Canvas.ClipRect**.

i- Sauvegarder le dessin BMP

Nous allons ajouter un bouton permettant de sauvegarder le dessin au format **BMP**. Le format BMP est le format standard **BitMap** utilisé par Windows et notamment par PaintBrush. C'est un format d'image non compressé.
Fond.Picture contient la Bitmap associée à notre dessin. Il va nous suffire de la sauvegarder.
Dans la **barrette Dialogues** cliquez sur **SavePictureDialog** et posez un composant sur la fiche. Il se nomme **SavePictureDialog1** et contient la fenêtre de dialogue des fichiers images. Réglez le champ **DefaultExt** sur ***.BMP et FileName** sur ***.BMP et Filter** sur **Bitmap (*.BMP)**
Ajouter un bouton **SpeedButton** et mettez lui le **Glyph FileSave.Bmp.** Double cliquez sur ce bouton et remplissez la méthode click :

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  IF SavePictureDialog1.Execute THEN Fond.Picture.saveToFile(SavePictureDialog1.FileName);
end
```

On exécute SavePictureDialog1 et si la réponse est affirmative on sauve la Bitmap Fond.Picture.
Faites un essai.

j- Ouvrir un fichier BMP.

Nous allons ajouter un dernier bouton permettant d'ouvrir un fichier BMP.
Dans la barrette Dialogues cliquez sur **OpenPictureDialog** et posez un composant sur la fiche. Il se nomme **OpenPictureDialog1** et contient la fenêtre de dialogue des fichiers images. Réglez le champ **DefaultExt** sur ***.BMP et FileName** sur ***.BMP et Filter** sur **Bitmap (*.BMP)**
Ajouter un bouton **SpeedButton** et mettez lui le **Glyph Fileload.Bmp.** Double cliquez sur ce bouton et remplissez la méthode click :

```
procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  IF OpenPictureDialog1.Execute THEN
   BEGIN
    Fond.Picture.loadFromFile(OpenPictureDialog1.FileName);
    Form1.Activate;
   END;
end;
```

On exécute **OpenPictureDialog1** et, si la réponse est affirmative, on charge la **Bitmap Fond.Picture**. L'instruction **Form1.Activate** permet de réinitialiser les variables pour pouvoir dessiner sur le dessin que l'on vient de charger.

k- Les fichiers nécessaires à l'exécution du programme

A l'aide de l'**Explorateur** de Windows aller dans le répertoire **\DELPHI\DESSIN01**. Dans ce répertoire on y trouve les fichiers suivants :

- *.PAS fichiers pascal dans lesquels sont toutes les procédures
- *.DCU Unités compilées utiles au projet
- *.DPR fichier de projet. C'est le fichier principal, il contient toutes les informations relatives au projet.
- *.RES fichier de ressources (contient le menu déroulant par exemple…)
- *.EXE fichier exécutable. Ce fichiers est indépendant et peut être exécuter sans l'aide des autres fichiers. Seuls les fichiers propre à la base de donnée lui sont nécessaires.
- *.DFM fichier contenant le code source (pascal) des fiches.

Essayons de lancer notre application en cliquant sur le fichier **DESSIN.EXE**. Cela fonctionne parfaitement. Pour installer l'application sur une autre machine il suffit de copier le fichier **DESSIN.EXE** Même si la machine ne contient pas Delphi elle pourra exécuter l'application.

n- Taille du fichier exécutable

Regarder dans l'explorateur la taille du fichier *.EXE. Ce fichier a une taille importante car il contient toutes les informations nécessaires au débogage de l'application. Il contient en fait les adresses de chacune des procédures pour pouvoir indiquer au développeur d'où vient l'erreur en cas de plantage. Une fois la mise au point du programme faite il n'est plus nécessaire de garder ces informations qui surchargent inutilement le code.

Retourner sous Delphi. Dans le menu **Projet-Options** cliquez sur l'onglet **Lieur** et désactiver **Inclure info de débogage TD32**. Après avoir refermé la fenêtre de dialogue des options demander **Tout construire** dans le menu **Projet**. Revenir alors dans le répertoire **DELPHI\DESSIN01** avec l'**explorateur** Windows et regarder la nouvelle taille du fichier exécutable : elle a été divisée par trois !