Développement Windows avec l'environnement Lazarus – Séance 2
Manipuler du texte et des formes colorées

Créer une nouvelle application:

Dans **Lazarus**, utilise la commande **« Fichier | Fermer tous les fichiers de l'éditeur »**, l'environnement de travail est remis à zéro.
Génère une nouvelle application avec la commande **« Fichier | Nouveau »** puis « Application ».

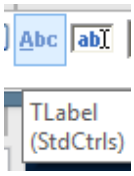
- Comme précédemment, crée un dossier **Perso▷Lazarus▷MonAppli2 .**
- Utilise la commande **Fichier|Enregistrer** sous de **Lazarus**, enregistre le projet (l'application) sous le nom **MonAppli2.lpi** dans le dossier **Perso▷Lazarus▷MonAppli2**
- Enregistre ensuite l'unité **unit1.pas** sous le nom **MainU.pas** dans le dossier **Perso▷Lazarus▷MonAppli2** ce fichier **MainU.pas** contiendra le code de l'application **MonAppli2.exe**

Nomme la fiche principale de ton projet **MainF**, comme précédemment.

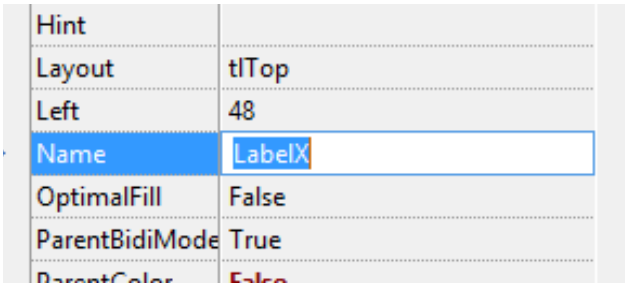
En utilisant l'inspecteur d'objets, donne à la nouvelle fiche les dimensions suivantes :
Largeur (propriété **Width** : 800 pixels) et Hauteur (propriété **Height** : 600 pixels).

A ce stade, tu peux compiler une première fois l'application (qui est une fiche vide) pour vérifier que tout fonctionne.

Manipuler du texte :

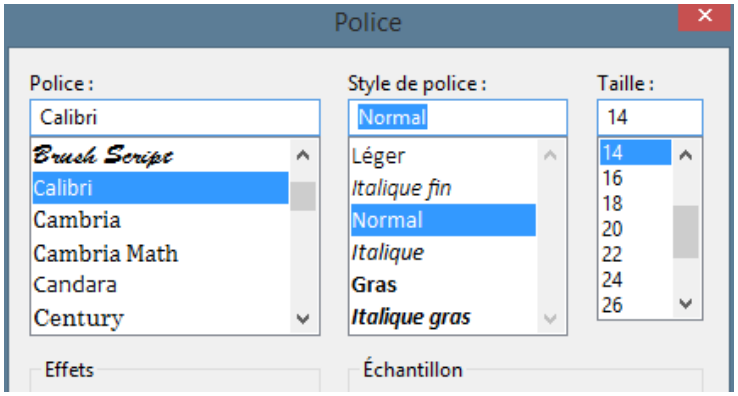


Place un composant **TLabel** (une étiquette en anglais) sur ta fiche.
Ce composant permet d'afficher toute sorte de texte.
Nomme ce TLabel **LabelX** en utilisant **l'inspecteur d'objets** (valeur **LabelX** affectée à la propriété **Name**).



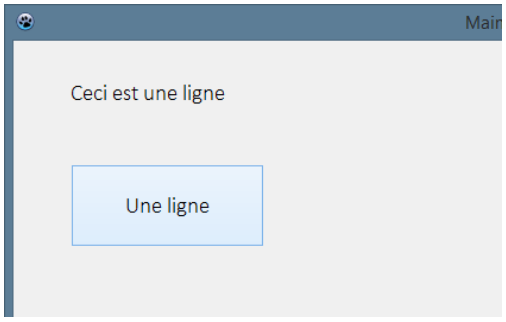
Pour l'instant le label affiche un texte par défaut (***LabelX***), tu peux modifier le texte en utilisant la propriété **Caption** du TLabel en utilisant l'inspecteur d'objets. Affecte par exemple, la valeur « ***Texte par défaut»*** à cette propriété **Caption**.

La police utilisée est un peu petite, en utilisant la propriété **Font** de **MainF** dans l'inspecteur d'objets, Affecte une police plus grande à la fiche **MainF** : Calibri 14 points
Tous les objets de cette fiche **MainF** utiliseront dorénavant cette police plus grande.



Crée maintenant un bouton **LigneBout** sur la fiche, son **Caption** affiche « Une ligne », lorsque l'on clique sur ce bouton **LabelX** affiche le texte « Ceci est une ligne ». Utilise pour cela la propriété **Caption** de **LabelX**

- Compile l'application, la fiche doit ressembler à ceci :



Effacer le texte d'un TLabel :

Pour effacer le texte d'un **TLabel**, il suffit d'affecter la valeur ' ' (espace vide) à son **Caption**, exemple :

```
begin
  LabelX.Caption:='';
end;
```

Crée un bouton **EffaceBout** dont le **Caption** affiche « Effacer » qui réalise l'effacement du texte de **LabelX**.

Afficher plusieurs lignes dans un TLabel

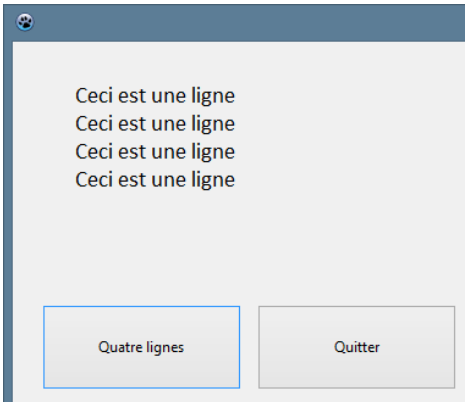
Il est possible d'afficher plusieurs lignes, l'une en dessous de l'autre dans un **TLabel**, il suffit de placer le code **+chr(13)+chr(10)+** entre le texte des lignes, voir le code ci-dessous.

```
procedure TMainF.Button1Click(Sender: TObject);
begin
  LabelX.Caption:='Ceci est une ligne'+chr(13)+chr(10)+
                  'Ceci est une ligne';
end;

end.
```

On peut passer des lignes dans le code ou placer des espaces entre les instructions, cela n'influe pas sur le fonctionnement du programme.

- Ajoute un bouton **QuitBout** « Quitter » à ta fiche, écris le code associé.
- Modifie ensuite le code précédent pour que la fiche affiche quatre lignes l'une en dessous de l'autre (utilise le copier/coller pour écrire le code).
- Le **Caption** du bouton **LigneBout** devient « Quatre lignes »
- Compile l'application, le résultat doit être celui qui est représenté par la fiche ci-contre à droite.



Automatisation d'une tâche, utilisation d'une boucle

Une boucle est une méthode pour répéter plusieurs fois une instruction ou une suite d'instructions.
Une boucle se présente de la manière suivante :

```
procedure TMainF.BoucleBoutClick(Sender: TObject);
var
  i : integer;
begin
  for i:=1 to 3 do
    begin

      // Ajoute ici le code à répéter

    end;
end;
```

Le code de boucle ci-contre va répéter trois fois une suite d'instructions (qui est vide ici).

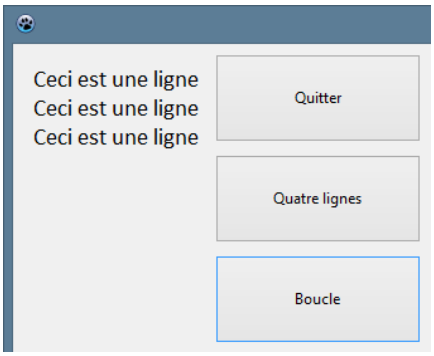
La boucle va être parcourue trois fois de **i :=1** à **i :=3** :
Une fois pour la variable **i:=1** ; Une fois pour la variable **i :=2** ; Une fois pour la variable **i :=3** ;

```
var
  i : integer;
```

Cette instruction déclare la variable i (de type **integer** qui est un entier, sans décimale donc) qui est utilisée pour le comptage des boucles de 1 à 3.

▪ Crée un bouton « **BoucleBout** » dont le **Caption** affiche 'Boucle' qui reprend l'exemple du code de boucle ci-dessus et qui permet d'afficher 3 fois 'Ceci est une ligne' comme dans la capture ci-contre à droite.

Modifie ensuite ton code pour afficher 5 lignes puis une deuxième fois pour afficher 10 lignes puis une troisième fois pour afficher 20 lignes.



Afficher la valeur d'une variable

▪ Crée un bouton **abBout**, affecte-lui le **Caption** 'a+b'
On peut utiliser librement une variable **integer**, comme dans ce code qui est exécuté lorsque l'on clique sur le bouton **abBout**. a et b sont des variables entières de type **integer** .

```
procedure TMainF.abBoutClick(Sender: TObject);
var
  a : integer;
  b : integer;
begin
  a:=3;
  b:=5;
end;
```

Les variables **a** et **b** sont déclarées. Dans le code ci-dessus, les valeurs 3 et 5 affectées aux variables **a** et **b** ne sont pas utilisées, ce code ne produit pas d'affichage.

Une valeur numérique ou une variable **integer** ne peut pas être affichée directement dans un **Caption**, elle doit d'abord être convertie en chaîne (type **string** ou chaîne).
La fonction qui permet la conversion est **IntToStr** (textuellement **IntegerToString** ou EntierVersChaine).

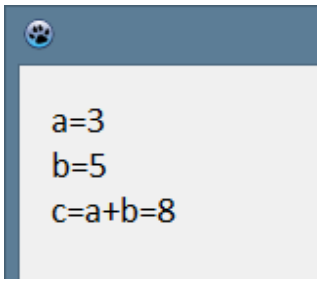
Par exemple, le code suivant convertit 1972 en une chaîne qui peut être affichée dans un **Caption**.

```
begin
  LabelX.Caption:=IntToStr(1972);
end;
```

Le code suivant affiche la valeur de la variable **integer b** dans **LabelX**, essaie ce code et compile l'application.

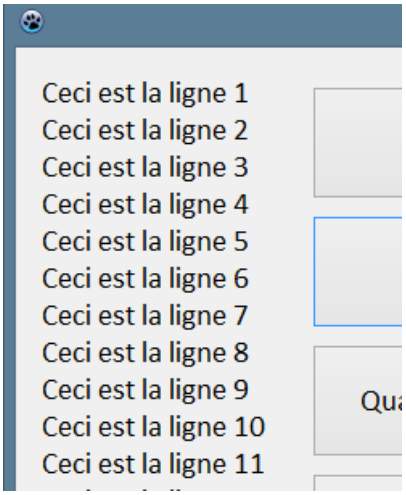
```
procedure TMainF.abBoutClick(Sender: TObject);
var
  a : integer;
  b : integer;
begin
  a:=3;
  b:=5;
  LabelX.Caption:=IntToStr(b);
end;
```

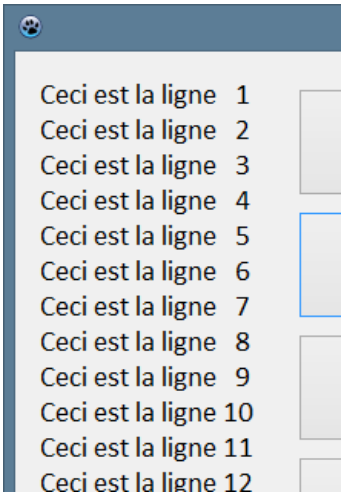
- Modifie ce code en ajoutant une variable **c** de type **integer**, cette variable est telle que c :=a+b
- Modifie le code de façon à ce que l'application calcule la somme c :=a+b
- Affiche en utilisant LabelX les valeurs de **a**, **b** et le résultat représenté par la variable **c** en utilisant la fonction **IntToStr**
- Résultat représenté par la capture ci-contre à droite.



<u>Afficher la valeur d'une variable dans la boucle</u>

Reprends le code de la boucle du bouton **BoucleBout** et modifie-le en utilisant l'instruction **IntToStr**, de manière à ce que chaque ligne affiche son numéro, comme dans la capture ci-contre à droite.





Les derniers chiffres sont décalés parce que '10' occupe davantage d'espaces que '9' par exemple.
En utilisant une condition « **if …then….** » décale de deux espaces vers la droite le nombre si il est inférieur à 10 comme dans la capture ci-contre à gauche. Ce nombre est représenté dans la boucle par la variable de type **integer i**.

Créer une nouvelle application:

Dans **Lazarus**, utilise la commande **« Fichier | Fermer tous les fichiers de l'éditeur »**, l'environnement de travail est remis à zéro.
Génère une nouvelle application avec la commande **« Fichier | Nouveau »** puis « Application ».

- Comme précédemment, crée un dossier **Perso▷Lazarus▷MonAppli3 .**
- Utilise la commande **Fichier|Enregistrer** sous de **Lazarus**, enregistre le projet (l'application) sous le nom **MonAppli2.lpi** dans le dossier **Perso▷Lazarus▷MonAppli3**

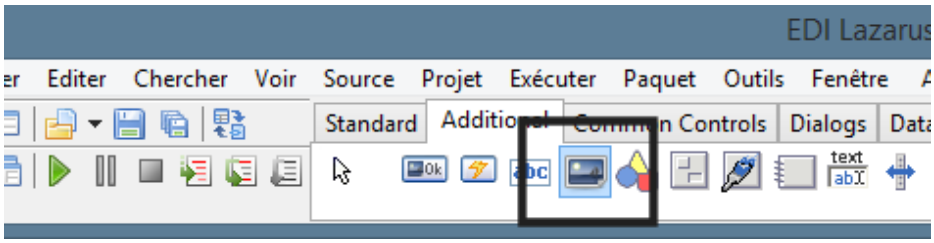
- Enregistre ensuite l'unité **unit1.pas** sous le nom **MainU.pas** dans le dossier **Perso▷Lazarus▷MonAppli3** ce fichier **MainU.pas** contiendra le code de l'application **MonAppli3.exe**

Nomme la fiche principale de ton projet **MainF**, comme précédemment.

En utilisant l'inspecteur d'objet, donne à la nouvelle fiche les dimensions suivantes :
Largeur (propriété **Width** : 800 pixels) et Hauteur (propriété **Height** : 600 pixels).
Utilise la même police Calibri 14 que précédemment, ajoute un bouton « Quitter ».

A ce stade, tu peux compiler une première fois l'application (qui est une fiche vide) pour vérifier que tout fonctionne.

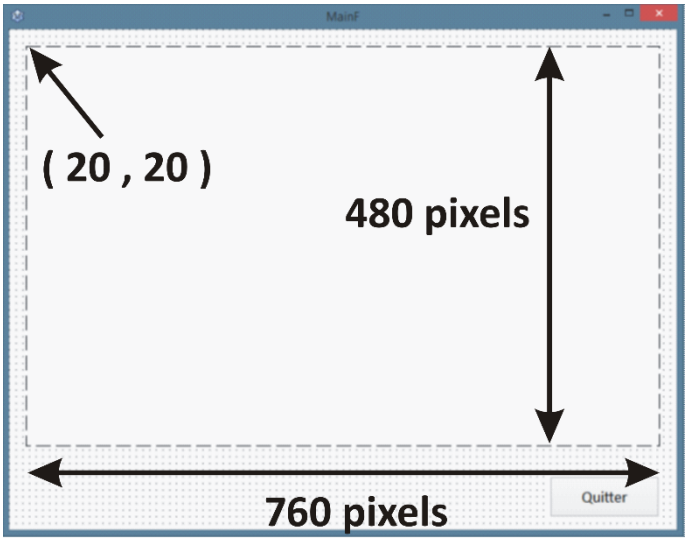
Ajouter une zone de dessin TImage sur la fiche:



Ajoute sur ta fiche un objet **TImage** (tu le trouves dans l'onglet **Additional**, voir encadré ci-dessus).
Appelle ce **TImage Dessin** (valeur **Dessin** effectée à sa propriété **Name**).

Affecte ensuite à ses propriétés les valeurs suivantes, **toutes les valeurs sont en pixels** :
Left : 20 , c'est son abscisse ou coordonnée X, mesurée à partir du bord gauche de l'écran
Top : 20 , c'est son ordonnée ou coordonnée Y, mesurée à partir du bord supérieur de la fiche **MainF**
(attention, pour les ordonnées Y, c'est l'inverse d'un repère orthonormé !)
Width : 760, c'est sa largeur
Height : 480, c'est sa hauteur

On obtient cette fiche, ci-contre à droite, si tu la compiles , **Dessin** n'apparaîtra pas, il est de la même couleur que le fond de la fiche.



Colorer la zone de dessin :

▪ Crée un bouton **FondBout**, affecte-lui le **Caption** 'Fond'

Le rôle de ce bouton est de colorer la fond de la zone de dessin en blanc (**clWhite**) et son contour en Noir (**clBlack**).

L'ensemble des procédures de dessin doivent être placées dans un bloc d'instructions Dessin.Canvas comme celui-ci :

```
procedure TMainF.FondBoutClick(Sender: TObject);
begin
  with Dessin.Canvas do
    begin

    end;
end;
```

Nous utiliserons trois instructions qui portent sur Dessin.Canvas (Canvas est la zone sur laquelle on peut dessiner, le « papier » en quelque sorte) :

▪ **Rectangle (x1,y1,x2,y2)** qui dessine un rectangle sur **Dessin**.Canvas du point de coordonnées (x1,y1) au point de coordonnées (x2,y2)

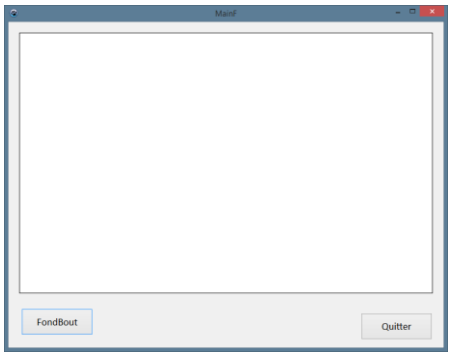
(attention, pour les ordonnées Y, c'est l'inverse d'un repère orthonormé !)

▪ **Pen.Color** qui définit la couleur du contour du rectangle
▪ **Brush.Color** qui définit la couleur du fond du rectangle

Donc pour dessiner un rectangle de contour noir et de fond blanc allant du point de coordonnées (0 , 0) au point de coordonnées (760, 480) le code adéquat est illustré ci-contre à droite :

```
begin
  with Dessin.Canvas do
    begin
      Pen.Color:=clBlack;
      Brush.Color:=clWhite;
      Rectangle (0,0,760,480);
    end;
end;
```

Pour le résultat ci-dessous :



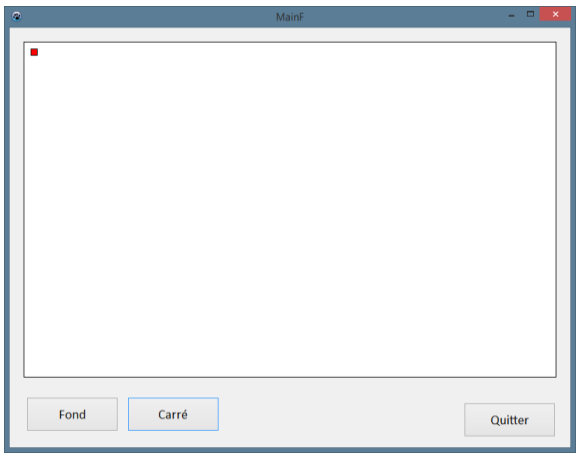
Insère ce code dans l'événement **OnClick** du bouton **FondBout**.

Pour que **Dessin** soit automatiquement dessiner de cette manière au démarrage, tu peux insérer ce code dans l'événement **OnShow** de **MainF** qui est exécuté au lancement de la fiche (To Show : montrer en anglais).

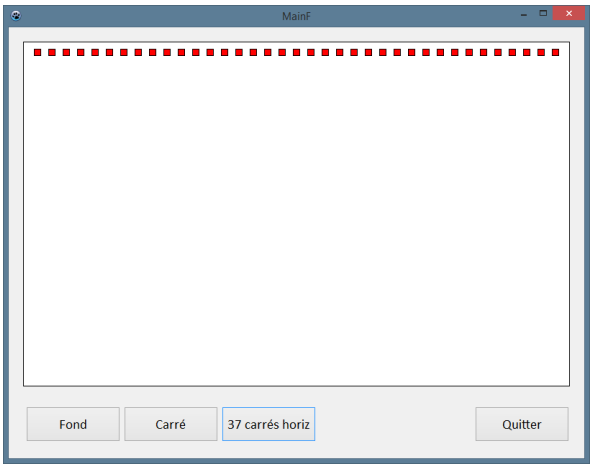
Dessiner un petit carré dans Dessin :

▪ Crée un bouton **CarreBout**, affecte-lui le **Caption** 'Carré'

Ecrit de code de l'événement **OnClick** de ce bouton de façon à ce qu'il affiche un carré de 10 pixels de côté au coordonnées (15 ,10) de **Dessin** .
Le carré a toujours un contour noir, mais cette fois-ci un fond rouge (**clRed**)
Résultat, ci-contre à droite.



Dessiner 37 carrés horizontaux dans Dessin :

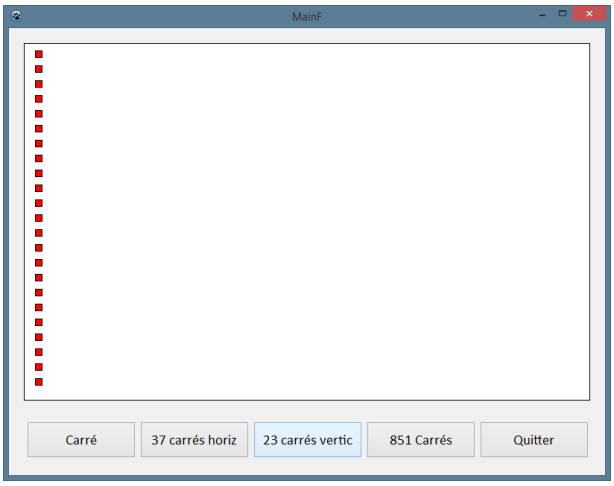


▪ Crée un bouton **CarresHBout**, affecte-lui le **Caption** '37 carrés horiz'

En utilisant une boucle portant sur une variable **i** de type **integer**, écris le code de l'événement **OnClick** de ce bouton de façon à ce qu'il affiche 37 carrés de 10 pixels de côté en commençant aux coordonnées (15 ,10) de **Dessin** .
Chaque carré est espacé horizontalement du précédent de dix pixels.

Résultat, ci-contre à droite.

Dessiner 23 carrés verticaux dans Dessin :



▪ Crée un bouton **CarresVBout**, affecte-lui le **Caption** '23 carrés vertic'

En utilisant une boucle portant sur une variable **j** de type **integer**, écris le code de l'événement **OnClick** de ce bouton de façon à ce qu'il affiche 23 carrés de 10 pixels de côté en commençant aux coordonnées (15 ,10) de **Dessin** .
Chaque carré est espacé verticalement du précédent de dix pixels.

Le carré a toujours un contour noir, mais cette fois-ci un fond rouge (**clRed**)
Résultat, ci-contre à droite.

Dessiner 851 carrés dans Dessin :

▪ Crée un bouton **CarresVHBout**, affecte-lui le **Caption** '851 carrés'

En combinant le code des deux boutons précédents, écris le code de l'événement **OnClick** de ce bouton de façon à ce qu'il affiche 851 carrés identiques aux précédents sur **Dessin** .

Utilise pour cela deux boucles l'une dans l'autre (on parle de boucles imbriquées), l'une portant sur la variable **i** de type **integer,** l'autre sur la variable **j** de type **integer .**

Les boucles imbriquées ont la structure suivante :

```
    .
130 for i:=0 to 10 do
    .  begin
    .    for j:=0 to 15 do
    .      begin
    .
135        end;
    .  end;
    .
```

Résultat, ci-contre à droite.