Université Lumière – Lyon 2, Faculté de Sciences Économiques et de Gestion
Maîtrise d'Économétrie – Année 1999-2000
Delphi – TD n° 4 : Menus, boîtes de dialogue et fiches
J. Darmont (*jerome.darmont@univ-lyon2.fr*), 05/01/2000

Exercice 1

Créer une nouvelle application dont l'interface comporte une liste (*ListBox*) nommée « Elements » qui contiendra des nombres entiers.

Ajouter un menu principal (*MainMenu*) à l'application. Ce menu contiendra deux choix, eux-mêmes divisés en sous-choix : Fichier/Quitter, Eléments/Ajouter/Supprimer. Pour constituer le menu, double-cliquer sur son icône. NB : l'icône menu est un composant invisible.

Associer le code approprié aux éléments du menu (Quitter, Ajouter et Supprimer) en cliquant dessus en mode édition. Les entiers à ajouter dans la liste augmentent de 1 à chaque ajout. C'est le dernier élément entré qui est supprimé.

Ajouter le caractère '&' devant les éléments « Fichier » et « Eléments » du menu, afin de permettre l'accès aux sous-menus par les raccourcis ALT+F et ALT+E, respectivement. Associer les éléments « Quitter », « Ajouter » et « Supprimer » aux raccourcis clavier CTRL+Q, CTRL+A et CTRL+S, respectivement (attribut *ShortCut*).

Ajouter un composant invisible boîte de dialogue polices de caractères (*FontDialog*) nommé « ChoixPolice ». Ajouter un élément de menu « Police » dans le menu « Elément » et lui associer le code permettant de changer la police de caractères de la liste « Elements ». Utiliser les attributs *Font* de la liste et de la boîte de dialogue. Une boîte de dialogue s'active par la méthode *execute*.

De même, ajouter un composant invisible boîte de dialogue couleur (*ColorDialog*) nommé « ChoixCouleur ». Ajouter un élément de menu « Couleur du fond » dans le menu « Elément » et lui associer le code permettant de changer la couleur du fond de la liste « Elements ». Utiliser les attributs *Color* de la liste et de la boîte de dialogue.

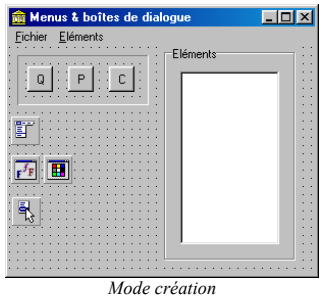
Ajouter une boîte de dialogue standard au projet grâce à la galerie de modèles de fiches (Fichier/Nouveau/Dialogues). Modifier le menu « Quitter » pour qu'il affiche le message de confirmation de la boîte de dialogue. Le bouton « OK » permet de quitter, « Cancel » de revenir à l'application. Utiliser l'attribut *Visible* pour afficher et cacher la fiche-boîte de dialogue.

Modifier le menu « Supprimer » en lui ajoutant un sous-menu Un élément/Tous les éléments. Associer le menu « Un élément » au raccourci clavier CTRL+S à la place de « Supprimer ». Déplacer le code associé à « Supprimer » vers « Un élément » et associer le code adéquat à « Tous les éléments ».

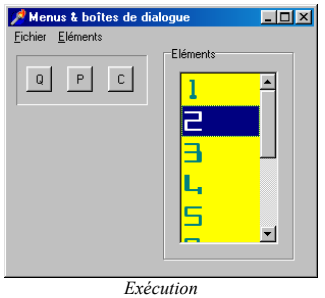
Ajouter un menu surgissant (*PopupMenu*) et le lier à la feuille (attribut *PopupMenu*). Insérer les éléments « Ajouter », « Supprimer » et « Supprimer tous » à ce menu. Lier ces éléments au code déjà défini pour ces actions (événement *OnClick* de chaque élément de menu).

Ajouter trois turbo boutons (*SpeedButton*) de légende « Q », « P » et « C » pour constituer une mini barre d'icônes. Associer ces boutons au code « Quitter », « Choix de police » et « Couleur du fond », respectivement.

Apparence finale de l'application :



Mode création



Exécution

Code :

```
unit menus_boites;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Menus, Buttons, ExtCtrls;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    GroupBox1: TGroupBox;
    Elements: TListBox;
    Fichier1: TMenuItem;
    Quitter1: TMenuItem;
    Elments1: TMenuItem;
    Ajouter1: TMenuItem;
    Supprimer1: TMenuItem;
    Unlment1: TMenuItem;
    Tousleslments1: TMenuItem;
    ChoixPolice: TFontDialog;
    Police1: TMenuItem;
    PopupMenu1: TPopupMenu;
    Supprimer2: TMenuItem;
    Supprimertous1: TMenuItem;
    Ajouter2: TMenuItem;
    Bevel1: TBevel;
    ChoixCouleur: TColorDialog;
    Couleur1: TMenuItem;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton1: TSpeedButton;
    procedure Ajouter1Click(Sender: TObject);
    procedure Police1Click(Sender: TObject);
    procedure Quitter1Click(Sender: TObject);
    procedure Tousleslments1Click(Sender: TObject);
    procedure Unlment1Click(Sender: TObject);
    procedure Couleur1Click(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;
```

```
var
  Form1: TForm1;
  elt_courant: word;

implementation

uses dialogue, diagsaisie;

{$R *.DFM}

procedure TForm1.Ajouter1Click(Sender: TObject);
begin
     elt_courant:=elt_courant+1;
     Elements.Items.Add(IntToStr(elt_courant));
end;

procedure TForm1.Police1Click(Sender: TObject);
begin
     ChoixPolice.Execute;
     Elements.Font:=ChoixPolice.Font;
end;

procedure TForm1.Quitter1Click(Sender: TObject);
begin
     Confirmation.Visible:=True;
end;

procedure TForm1.Tousleslments1Click(Sender: TObject);
begin
     Elements.Clear;
end;

procedure TForm1.Unlment1Click(Sender: TObject);
begin
     Elements.items.delete(Elements.items.count-1);
end;

procedure TForm1.Couleur1Click(Sender: TObject);
begin
      ChoixCouleur.Execute;
      Elements.Color:=ChoixCouleur.Color;
end;

end.
```

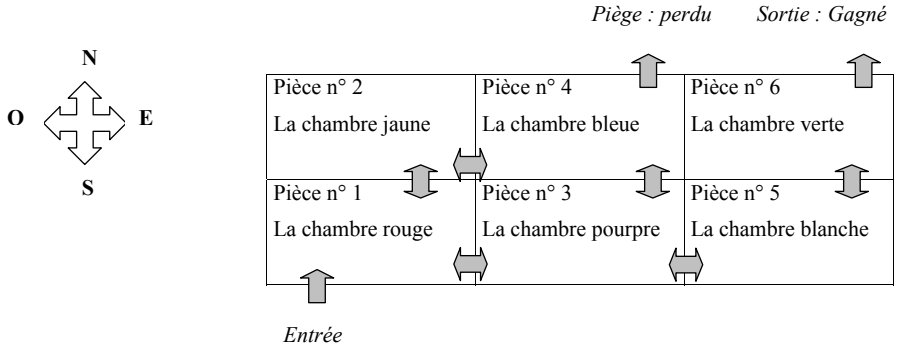
```
unit dialogue;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls;

type
  TConfirmation = class(TForm)
    OKBtn: TButton;
    CancelBtn: TButton;
    Bevel1: TBevel;
    Label1: TLabel;
    procedure OKBtnClick(Sender: TObject);
    procedure CancelBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```



```
  end;

var
  Confirmation: TConfirmation;

implementation

{$R *.DFM}

procedure TConfirmation.OKBtnClick(Sender: TObject);
begin
     halt;
end;

procedure TConfirmation.CancelBtnClick(Sender: TObject);
begin
     Confirmation.Visible:=false;
end;

end.
```

Exercice 2

On souhaite programmer un petit jeu dans lequel le joueur doit s'orienter dans le « labyrinthe » suivant. Les flèches indiquent les passages possibles d'une pièce à l'autre.



1) Nommer la fiche par défaut du projet « Piece1 » (propriété *Name*). Modifier ses attributs *Color* et *Caption* pour que la couleur et la légende de la feuille correspondent à ceux de la pièce n° 1 du labyrinthe.

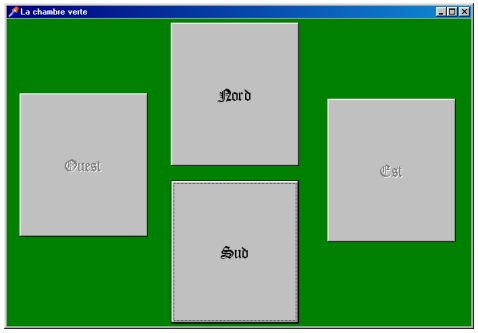
2) Ajouter 5 autres fiches au projet (menu Fichier/Nouvelle fiche). Les nommer « Piece2 » à « Piece6 » et les modifier comme la première fiche de façon à reproduire les pièces du labyrinthe.

3) Ajouter sur chaque feuille 4 boutons de commande de nom (*Name*) et de légende (*Caption*) identiques : « Nord », « Sud », « Est » et « Ouest ». Il est possible d'insérer ces boutons sur une feuille et de les copier/coller sur les autres. Ces boutons permettront de passer d'une feuille à l'autre (c'est-à-dire d'une pièce à l'autre, pour le joueur, la pièce de départ étant la pièce n° 1).

4) Désactiver les boutons de commande inutiles en mettant leur propriété *Enabled* à False. Par exemple, le bouton « Ouest » de la feuille « Piece1 » ou le bouton « Sud » de la feuille « Piece3 » doivent être désactivés car il n'existe pas de passage à ces endroits.

5) Associer aux autres boutons de commande le code permettant de passer aux pièces appropriées. Le piège de la pièce n° 4 renvoie dans la pièce n° 1 après un message (procédure `ShowMessage`). Lorsque le joueur gagne, il sort du programme après un message de félicitations.

Exemple : apparence de la feuille « Piece6 »



Question subsidiaire : Compter le nombre de « coups » nécessaire au joueur pour sortir du labyrinthe et l'afficher au moment où il gagne. Pour cela, utiliser une variable commune stockée dans une unité indépendante des fiches (menu Fichier/Nouveau/Unité).

Code :

```
unit upiece1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TPiece1 = class(TForm)
    Nord: TButton;
    Sud: TButton;
    Ouest: TButton;
    Est: TButton;
    procedure NordClick(Sender: TObject);
    procedure EstClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  Piece1: TPiece1;

implementation
```



```
uses upiece2, upiece3, ucompte;

{$R *.DFM}

procedure TPiece1.NordClick(Sender: TObject);
begin
     Piece1.Visible:=False;
     Piece2.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece1.EstClick(Sender: TObject);
begin
     Piece1.Visible:=False;
     Piece3.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece1.FormCreate(Sender: TObject);
begin
     coups:=0;
end;

end.
```

```
unit upiece2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TPiece2 = class(TForm)
    Sud: TButton;
    Ouest: TButton;
    Nord: TButton;
    Est: TButton;
    procedure SudClick(Sender: TObject);
    procedure EstClick(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  Piece2: TPiece2;

implementation

uses upiece1, upiece4, ucompte;

{$R *.DFM}

procedure TPiece2.SudClick(Sender: TObject);
begin
     Piece2.Visible:=False;
     Piece1.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece2.EstClick(Sender: TObject);
```

```
begin
     Piece2.Visible:=False;
     Piece4.Visible:=True;
     coups:=coups+1;
end;

end.
```

```
unit upiece3;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TPiece3 = class(TForm)
    Sud: TButton;
    Ouest: TButton;
    Nord: TButton;
    Est: TButton;
    procedure OuestClick(Sender: TObject);
    procedure NordClick(Sender: TObject);
    procedure EstClick(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  Piece3: TPiece3;

implementation

uses upiece1, upiece4, upiece5, ucompte;

{$R *.DFM}

procedure TPiece3.OuestClick(Sender: TObject);
begin
     Piece3.Visible:=False;
     Piece1.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece3.NordClick(Sender: TObject);
begin
     Piece3.Visible:=False;
     Piece4.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece3.EstClick(Sender: TObject);
begin
     Piece3.Visible:=False;
     Piece5.Visible:=True;
     coups:=coups+1;
end;

end.
```



```
unit upiece4;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TPiece4 = class(TForm)
    Sud: TButton;
    Ouest: TButton;
    Nord: TButton;
    Est: TButton;
    procedure OuestClick(Sender: TObject);
    procedure SudClick(Sender: TObject);
    procedure NordClick(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  Piece4: TPiece4;

implementation

uses upiece2, upiece3, upiece1, ucompte;

{$R *.DFM}

procedure TPiece4.OuestClick(Sender: TObject);
begin
     Piece4.Visible:=False;
     Piece2.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece4.SudClick(Sender: TObject);
begin
     Piece4.Visible:=False;
     Piece3.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece4.NordClick(Sender: TObject);
begin
     ShowMessage('Caramba ! C''était un piège...');
     Piece4.Visible:=False;
     Piece1.Visible:=True;
     coups:=coups+1;
end;

end.
```

```
unit upiece5;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
```

```
type
  TPiece5 = class(TForm)
    Sud: TButton;
    Ouest: TButton;
    Nord: TButton;
    Est: TButton;
    procedure OuestClick(Sender: TObject);
    procedure NordClick(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  Piece5: TPiece5;

implementation

uses upiece3, upiece6, ucompte;

{$R *.DFM}

procedure TPiece5.OuestClick(Sender: TObject);
begin
     Piece5.Visible:=False;
     Piece3.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece5.NordClick(Sender: TObject);
begin
     Piece5.Visible:=False;
     Piece6.Visible:=True;
     coups:=coups+1;
end;

end.
```

unit upiece6;

```
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TPiece6 = class(TForm)
    Sud: TButton;
    Ouest: TButton;
    Nord: TButton;
    Est: TButton;
    procedure SudClick(Sender: TObject);
    procedure NordClick(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  Piece6: TPiece6;
```



```
implementation

uses upiece5, ucompte;

{$R *.DFM}

procedure TPiece6.SudClick(Sender: TObject);
begin
     Piece6.Visible:=False;
     Piece5.Visible:=True;
     coups:=coups+1;
end;

procedure TPiece6.NordClick(Sender: TObject);
begin
     coups:=coups+1;
     ShowMessage('Bravo ! Vous avez gagné en '+IntToStr(coups)+' coups.');
     Halt;
end;

end.
```

unit ucompte;

```
interface

         var coups: byte;

implementation

end.
```