Spesifikasi:

Ukuran: 14x21 cm
Tebal: 367 hlm
Harga: Rp 41.800
Terbit pertama: Mei 2004
Sinopsis singkat:

Borland Delphi merupakan program aplikasi database yang berbasis Object Pascal dari Borland. Selain itu, Delphi juga memberikan fasilitas pembuatan aplikasi visual. Delphi merupakan pilihan dalam pembuatan aplikasi visual karena memberikan produktifitas yang tinggi. Delphi 7 memberikan fasilitas untuk dua platform, yaitu untuk platform Windows dan Linux. Delphi untuk Linux sebelumnya dikemas dalam sebuah aplikasi terpisah yang bernama Kylix, tetapi Delphi 7 menyatukannya dalam sebuah aplikasi. Library untuk Windows disebut VCL dan library untuk Linux disebut CLX.

Buku 36 Jam Belajar Komputer Delphi 7 ini dibuat agar Anda dapat mempelajari, memahami, mencoba dan melatih penggunaan fasilitas-fasilitas Delphi 7 secara mudah dan cepat sesuai dengan kebutuhan. Dalam penulisan buku ini penulis berusaha memberikan materi dan contoh yang sederhana dan dapat diaplikasikan. Tidaklah mungkin membahas semua fasilitas Delphi 7 dalam buku ini, tetapi penulis membahas hal-hal pokok sehingga setelah menyelesaikan buku ini, pembaca dapat dengan mudah mengembangkannya.

Modul 6

Menu dan Frame

Pokok Bahasan:

- ✓ **Memakai Menu**
- ✓ **Membuat Submenu**
- ✓ **Event OnContextPopup**
- ✓ **Membuat Item Menu Secara Dinamik**
- ✓ **Menu yang Owner-Draw**
- ✓ **Frame Sederhana**
- ✓ **Frame Bersarang**
- ✓ **Membuang Frame**
- ✓ **Frame Master Detil**

6.1 Memakai Menu

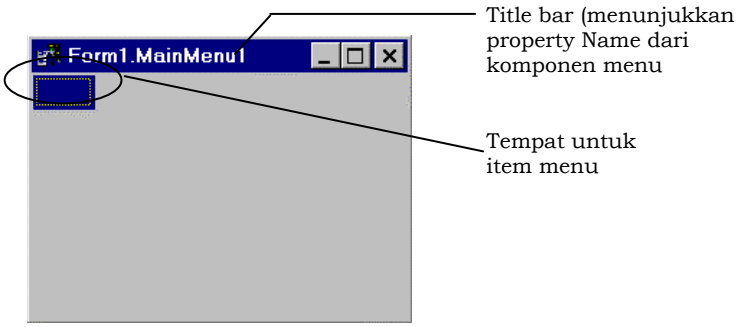
Menu memberikan cara yang mudah untuk menjalankan kelompok perintah-perintah. **Menu Designer** memungkinkan Anda untuk menambahkan sebuah menu ke dalam form. Anda hanya perlu menambahkan sebuah komponen menu ke dalam form, membuka **Menu Designer** dan mengetikkan item-item menu pada jendela **Menu Designer**. Pada saat perancangan, Anda dapat menambah, menghapus item-item menu, atau drag dan drop untuk mengatur ulang.

Ada dua komponen menu, yaitu **MainMenu** dan **PopUpMenu**. Komponen **MainMenu** membuat menu yang menempel pada title bar dari form. Sedangkan **PopUpMenu** membuat menu pada saat user mengklik kanan pada form atau pada sebuah komponen.

Kita akan membuat sebuah aplikasi yang memakai **MainMenu** dan **PopUpMenu**.

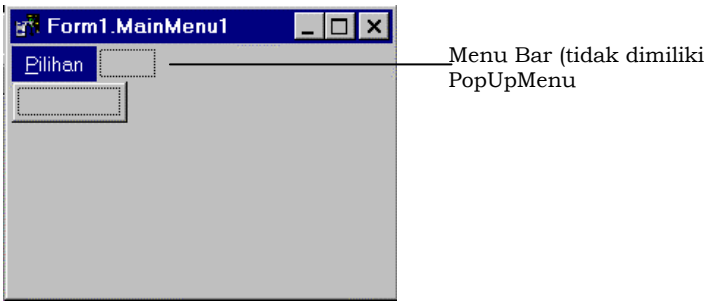
1. Buatlah aplikasi baru, ubah caption dari form menjadi '**Mencoba Menu**'.
2. Tambahkan **MainMenu**, **PopUpMenu** dan **Edit**. **MainMenu** dan **PopUpMenu** adalah komponen NonVisual, jadi Anda dapat menempatkannya di mana saja.
3. Pilih **MainMenu** dan kita akan membuka **Menu Designer**. Caranya:
   - Klik ganda pada komponen menu.
   - Dari page **Properties** Object Inspector, pilih **Items**, lalu klik ganda {Menu} pada kolom isian atau klik tanda (...).

   Menu Designer ditampilkan dengan item pertama yang kosong dan dipilih.



Gambar 6.1 Menu Designer untuk MainMenu

2. Ketikkan **&Pilihan**. Karakter & dipakai untuk menyatakan shortcut dari menu. Pada tampilan menu, karakter P akan bergaris bawah. Pada saat menjalankan aplikasi, Anda dapat memanggil menu dengan menunjuknya atau tekan Ctrl+P.

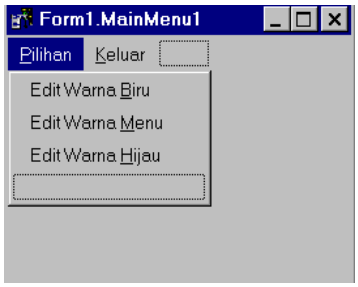


Gambar 6.2 Menu Designer dengan Satu Item

3. Lengkapi menu tersebut dengan beberapa item di bawahnya:

   Edit Warna &Biru
   Edit Warna &Menu
   Edit Warna &Hijau

4. Di samping kanan item **&Pilihan** tambahkan **&Keluar**. Menu lengkapnya sebagai berikut.



Gambar 6.3 Menu Lengkap

5. Sekarang kita akan mengisikan item pertama (**Edit Warna Biru**). Klik ganda item tersebut dan Anda akan masuk ke Code Editor dan isikan:

```
Edit1.Color := clBlue;
```

6. Selanjutnya untuk item kedua dan ketiga, masing-masing isikan:

```
Edit1.Color := clMenu ;
```
dan
```
Edit1.Color := clLime ;
```

7. Tekan tombol **Close** untuk mengakhiri main menu.
8. Kita akan mengisi **PopUpMenu.** Dari form, klik ganda pada **PopUpMenu**, sekali lagi Anda akan masuk ke **Menu Designer** dan isikan item-item menu berikut.

   Teks Warna K&uning
   Teks Warna Me&rah
   Teks Warna &Default

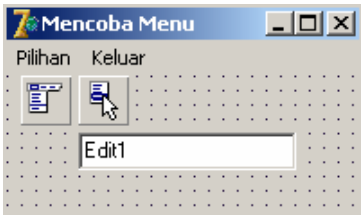
9. Untuk setiap item isikan:

```
Edit1.Font.Color := clYellow ;
Edit1.Font.Color := clRed ;
Edit1.Font.Color := clWindowText ;
```

10. Terakhir untuk item **Keluar**, isikan:

```
Application.Terminate ;
```

PopUpMenu1 akan ditampilkan jika Anda mengklik kanan pada komponen **Edit1**. Untuk menghubungkan kedua komponen tersebut, klik **Edit1**. Dari Object Inspector, pilih property **PopUpMenu**, Anda dapat mengklik anak panah bawah dan pilih **PopUpMenu1**. Rancangan form:



Gambar 6.4 Rancangan Form Aplikasi Menu

Listing lengkap:

```
unit UMenu1;

interface
```

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

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    PopupMenu1: TPopupMenu;
    Edit1: TEdit;
    Pilihan1: TMenuItem;
    Keluar1: TMenuItem;
    EditWarnaBiru1: TMenuItem;
    EditWarnaMerah1: TMenuItem;
    EditWarnaHijau1: TMenuItem;
    eksWarnaKuning1: TMenuItem;
    eksWarnaMErah1: TMenuItem;
    eksWarnaDefault1: TMenuItem;
    procedure EditWarnaBiru1Click(Sender: TObject);
    procedure EditWarnaMerah1Click(Sender: TObject);
    procedure EditWarnaHijau1Click(Sender: TObject);
    procedure eksWarnaKuning1Click(Sender: TObject);
    procedure eksWarnaMErah1Click(Sender: TObject);
    procedure eksWarnaDefault1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.EditWarnaBiru1Click(Sender: TObject);
begin
  Edit1.Color := clBlue ;
end;

procedure TForm1.EditWarnaMerah1Click(Sender: TObject);
begin
  Edit1.Color := clRed ;
end;

procedure TForm1.EditWarnaHijau1Click(Sender: TObject);
begin
  Edit1.Color := clGreen ;
end;

procedure TForm1.eksWarnaKuning1Click(Sender: TObject);
begin
  Edit1.Font.Color := clYellow
end;

procedure TForm1.eksWarnaMErah1Click(Sender: TObject);
begin
  Edit1.Font.Color := clRed ;
```

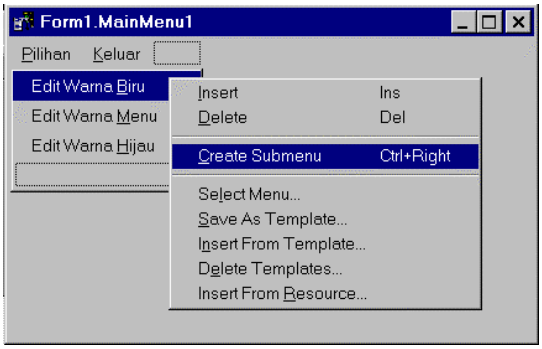
```
end;

procedure TForm1.eksWarnaDefault1Click(Sender: TObject);
begin
  Edit1.Font.Color := clWindowText ;
end;

end.
```

6.2 Membuat Submenu

Beberapa menu aplikasi mempunyai daftar drop-down yang muncul di samping sebuah item menu, dipakai untuk memberi pilihan tambahan (perintah-perintah yang berhubungan). Daftar tersebut ditunjukkan oleh sebuah anak panah di sebelah kanan item menu.

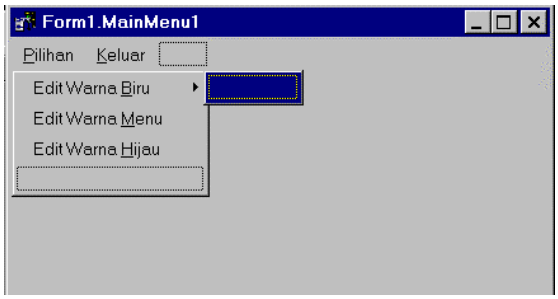
Cara demikian dapat menghemat tempat secara vertikal. Cara membuat submenu:

1. Buka kembali aplikasi menu sebelumnya, dan pilih sebuah item menu, misalnya **Edit Warna Biru**.
2. Klik kanan tombol mouse untuk menampilkan menu lokal (popup menu).



Gambar 6.5 Popup menu dari Sebuah Item Menu

3. Pilih **Create Submenu**, akan muncul anak panah kanan dan tempat mengisikan item menu.



Gambar 6.6 Mengisikan Submenu

4. Isikan item pertama, yaitu **Biru muda**. Tekan Enter atau panah bawah untuk item selanjutnya, yaitu **Biru tua**.

6.3 Event OnContextPopup

Komponen **PopupMenu** dapat dihubungkan ke sebuah komponen dengan mengatur properti **PopupMenu** dari komponen tersebut. Cara lain dengan memanggil **PopupMenu** menggunakan method **Popup**. Method ini memerlukan posisi layar untuk menampilkan menu. Salah satu cara menentukan posisinya adalah dengan mengubah posisi lokal dari komponen menjadi posisi layar. Untuk itu Anda memerlukan method **ClientToScreen**, yaitu untuk mengubah posisi lokal menjadi posisi layar.

Event **OnContextPopup** terjadi pada saat user mengklik kanan tombol mouse pada sebuah kontrol, atau memanggil menu popup dengan keyboard.

Untuk mencoba kemampuan-kemampuan tersebut, kita akan membuat sebuah aplikasi.

1. Tambahkan komponen **ColorDialog** (dari page **Dialogs**) dan tiga buah **Label.**
2. Tambahkan sebuah **PopupMenu** dengan sebuah menu, yaitu **Warna**.
3. Tambahkan sebuah **PopupMenu** lagi, dengan tiga buah menu, yaitu **Kiri, Tengah, Kanan.**

4. Misalnya jika Anda mengklik kanan tombol mouse pada **Label1**, kita akan menampilkan **PopupMenu** pertama dengan posisi mouse pada saat diklik. Jadi posisi menu dapat berubah-ubah bergantung pada tempat klik. Untuk itu kita harus mengubah posisi lokal pada **Label1** menjadi posisi di layar dengan method **ClientToScreen.** Buatlah event handler **OnMouseDown** dari **Label1.**

5. Event handler **OnContextPopup** dari **Label2** akan menampilkan menu Popup yang kedua dengan menambahkan dua menu, yaitu garis batas dan informasi waktu. Setelah selesai, kedua menu tambahan akan dihapus. Menu dihapus dengan urutan terbalik, dari yang terakhir dan berjalan mundur.

6. Event handler **OnContextPopup** tidak hanya untuk menampilkan menu popup, tetapi juga dapat melakukan operasi lain. Misalnya untuk menampilkan kotak dialog. Tambahkan **OnContextPopup** dari **Label3** untuk menampilkan kotak dialog.

Listing lengkap:

```
unit UMenu2;

interface

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

type
  TForm1 = class(TForm)
    ColorDialog1: TColorDialog;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    PopupMenu1: TPopupMenu;
    Warna1: TMenuItem;
    PopupMenu2: TPopupMenu;
    Kiri1: TMenuItem;
    Engah1: TMenuItem;
    Kanan1: TMenuItem;
    procedure Label1MouseDown(Sender: TObject; Button:
TMouseButton;
      Shift: TShiftState; X, Y: Integer);
```

```
    procedure  Label2ContextPopup(Sender:  TObject;  MousePos:
TPoint;
      var Handled: Boolean);
    procedure  Label3ContextPopup(Sender:  TObject;  MousePos:
TPoint;
      var Handled: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Label1MouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var PosisiLokal, PosisiLayar : TPoint ;
begin
  if Button = mbRight then
  begin
    PosisiLokal.X := X ; PosisiLokal.Y := Y ;
    PosisiLayar := Label1.ClientToScreen(PosisiLokal) ;
    PopupMenu1.Popup(PosisiLayar.X, PosisiLayar.Y);
  end ;
end;

procedure TForm1.Label2ContextPopup(Sender: TObject; MousePos:
TPoint;
  var Handled: Boolean);
var PosisiLayar : TPoint ;
begin
  // menambah menu secara dinamik
  PopupMenu2.Items.Add(NewLine) ;

PopupMenu2.Items.Add(NewItem(TimeToStr(Now),0,False,True,nil,0,
''));
  // menampilkan menu
  PosisiLayar := ClientToScreen(MousePos) ;
  PopupMenu2.Popup(PosisiLayar.X, PosisiLayar.Y) ;
  // menghapus menu tambahan
  PopupMenu2.Items[4].Free ;
  PopupMenu2.Items[3].Free ;
end;

procedure TForm1.Label3ContextPopup(Sender: TObject; MousePos:
TPoint;
  var Handled: Boolean);
begin
  ColorDialog1.Color := Label3.Color ;
  if ColorDialog1.Execute then Label3.Color :=
ColorDialog1.Color ;
end;

end.
```

6.4 Membuat Item Menu Secara Dinamik

Anda dapat membuat item-item menu secara dinamik pada saat run time. Cara ini diperlukan misalnya item-item menu bersifat pengulangan, atau item menu bergantung pada konfigurasi sistem dan user.

Dasarnya adalah pemakaian class **TMenuItem** yang dipakai pada komponen **MainMenu** dan **PopupMenu**. Setiap item menu mempunyai struktur sama dan bersifat rekursif, artinya sebuah item menu dapat berisi item-item menu lain sebagai submenunya. Ada dua properti yang penting, yaitu:

- **Items** yang berisi daftar item menu
- **Count** yang berisi banyaknya subitem
- **RadioItem** menyatakan apakah sebuah item menu bersifat *mutually exclusive* dengan item lain dalam group-nya (artinya, dalam sebuah group hanya satu item yang ditandai)

Kita akan membuat aplikasi yang akan membuat menu pull-down secara dinamik. Item-item menu tersebut menunjukkan ukuran font dari sebuah Label. Misalnya ukuran-ukuran font yang akan ditampilkan di menu adalah 8, 12, 16 dan seterusnya sampai 48. Oleh karena item-item menu menunjukkan pengulangan, kita akan membuatnya dalam sebuah loop.

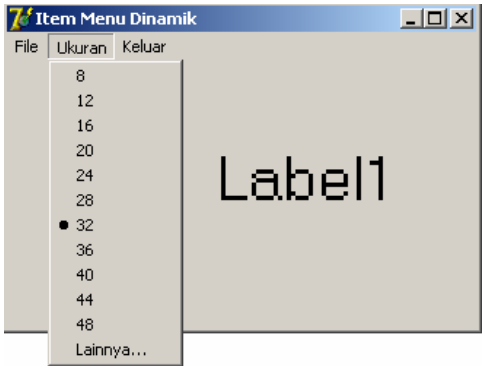
1. Buatlah aplikasi baru, tambahkan sebuah **Label** dan sebuah **MainMenu**.
2. Buatlah dua buah menu utama pada komponen **MainMenu** yaitu **File** dan **Keluar**.
3. Pada saat form diaktifkan (event **OnActivate**) akan ditambahkan menu utama **Ukuran** yang berada setelah menu **File**. Menu tersebut mempunyai item-item 8, 12, 16 sampai 48, oleh sebab itu dipakai sebuah loop. Setiap item menu mempunyai style **RadioItem** dan semua item masuk dalam group yang sama, yaitu 1. Event handler **OnClick** dari setiap item sama, yaitu **Proc1** yang dideklarasikan sebagai procedure **Private**. Pada akhir menu ditambahkan sebuah pilihan, yaitu **Lainnya**.

4. Deklarasikan **Proc1** di bagian **private** dari **Interface**.
5. Buatlah **Proc1** dengan parameter **Sender** yang bertipe **TObject**. Proc1 dipanggil pada saat user memilih menu **Ukuran** dan procedure ini akan mencari ukuran submenunya yang sesuai dengan ukuran font dari Label1 dan memberi tanda.
6. Deklarasikan **Proc2** di bagian **private** dari **Interface**.
7. Buatlah **Proc2** dengan parameter **Sender** yang bertipe **TObject**. Proc2 dipanggil pada saat user memakai salah satu ukuran dan mengubah ukuran font.

Listing lengkap:

```
unit Umenu3;

interface

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

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Label1: TLabel;
    Keluar1: TMenuItem;
    File1: TMenuItem;
    procedure FormActivate(Sender: TObject);
  private
    Procedure Proc1(Sender : TObject) ;
    Procedure Proc2(Sender : TObject) ;
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormActivate(Sender: TObject);
var PullDown, Item : TMenuItem ;
    Posisi, i : integer ;
begin
  // Membuat menu pulldown baru
  PullDown := TMenuItem.Create(Self) ;
  PullDown.AutoHotkeys := maManual ;
  PullDown.Caption := 'Ukuran' ;
```

```
  PullDown.OnClick := Proc1 ;
  // menghitung posisi dan menambah
  Posisi := MainMenu1.Items.IndexOf(File1) ;
  MainMenu1.Items.Insert(Posisi+1, PullDown) ;
  // membuat item-item menu
  I := 8 ;
  while I <= 48 do
  begin
    // membuat item baru
    Item := TMenuItem.Create(Self) ;
    Item.Caption := IntToStr(I) ;
    // Membuatnya sebagai radio item
    Item.GroupIndex := 1 ;
    Item.RadioItem := True ;
    // mengeset OnClick dan menambahkan item menu
    Item.OnClick := Proc2 ;
    PullDown.Insert(PullDown.Count,Item);
    I := I + 4 ;
  end ;
  // tambahkan item di akhir
  Item := TMenuItem.Create(Self) ;
  Item.Caption := '&Lainnya...';
  Item.GroupIndex := 1 ;
  Item.RadioItem := True ;
  PullDown.Insert(PullDown.Count,Item);
end;

procedure TForm1.Proc1(Sender : TObject);
var I : Integer ;
    Ketemu : Boolean ;
begin
  Ketemu := False ;
  with Sender as TMenuItem do
  begin
    // cara yang cocok, tidak termasuk yang terakhir
    for I := 0 to Count - 2 do
      if StrToInt(Items[i].Caption) = Label1.Font.Size
      then begin
             Items[i].Checked := True ;
             Ketemu := True ;
             System.Break ;; // berhenti
           end ;
    if not Ketemu then Items[Count-1].Checked := True ;
  end ;
end ;

procedure TForm1.Proc2(Sender : TObject);
begin
  with Sender as TMenuItem do
    Label1.Font.Size := StrToInt(Caption) ;
end ;

end.
```



Gambar 6.7 Menu Dinamik

6.5 Menu yang Owner-Draw

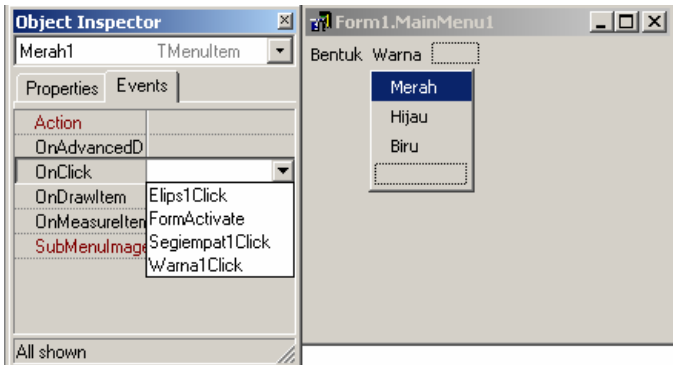
Teknik **owner-draw** adalah cara penggambaran sebuah kontrol yang dilakukan oleh pemilik (owner) dari kontrol tersebut. Biasanya ownernya adalah form. Pada komponen menu, Anda dapat menyatakan cara ini dengan mengatur properti **OwnerDraw** menjadi **True**. Ada dua event yang berhubungan:

- **OnMeasureItem** terjadi untuk setiap item menu pada saat menu pull-down ditampilkan.
- **OnDrawItem** terjadi pada saat item menu akan digambar ulang. Event ini terjadi pada saat Windows pertama kali menampilkan item dan setiap kali ada perubahan status, misalnya karena mouse bergerak ke sebuah item sehingga item tersebut akan disorot.

Untuk mencoba teknik ini, buatlah sebuah aplikasi.

1. Tambahkan komponen **MainMenu** dan ubah properti **OwnerDraw=True**.
2. Tambahkan menu utama **Bentuk** dengan item-item: **Lingkaran**, **Segiempat**, **Elips**.

3. Tambahkan menu utama **Warna** dengan item-item **Merah, Hijau, Biru**.
4. Tambahkan sebuah **Shape** dari page **Additional**.
5. Properti **Tag** dipakai untuk menyimpan nilai integer sebagai bagian dari sebuah komponen. Buatlah event handler **OnActivate** dari form untuk memberikan nilai integer bagi setiap menu warna.
6. Isikan event handler **OnMeasureItem** dari menu **Bentuk**. Procedure akan menentukan ukuran setiap item menu.
7. Event handler **OnDrawItem** dari menu **Segiempat, Elips** akan menentukan warna dan menggambar bentuk yang sesuai pada item menu.
8. Jika item menu dari **Bentuk** diklik, akan digambar bentuk yang sesuai pada form, bukan pada menu.
9. Jika menu **Warna** diklik, akan menentukan warna dari bentuk. Procedure ini akan dipanggil oleh menu **Merah, Biru, Hijau**. Pilih menu **Merah**, aktifkan tab **Events** dari Object Inspector, klik anak panah pada **OnClick**, ditampilkan procedure-procedure yang sudah ada.



Gambar 6.8 Menampilkan Event Handler yang Sudah Ada

10. Pilih **Warna1Click**. Lakukan cara yang sama untuk menu **Hijau** dan **Biru**.

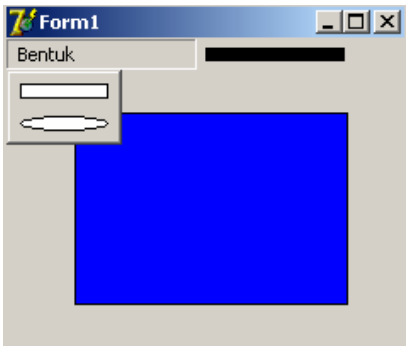
11. Event handler **OnDrawItem** dari menu **Warna** akan menentukan warna dan menggambar segiempat dengan warna tertentu pada item menu.
12. Event handler **OnDrawItem** dari menu **Merah, Biru, Hijau** akan mengacu ke procedure yang sudah ada, yaitu **Warna1DrawItem.** Pilihlah procedure tersebut dari yang sudah ada.
13. Isikan event handler **OnMeasureItem** dari menu **Warna**. Procedure akan menentukan ukuran setiap item menu.

Listing lengkap:

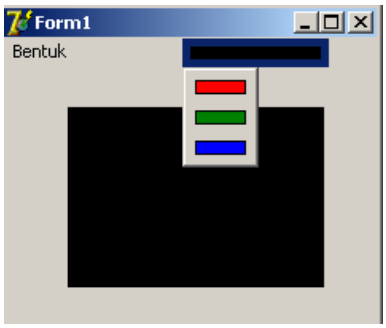
```
unit UODMenu;

interface

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

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    Bentuk1: TMenuItem;
    Segiempat1: TMenuItem;
    Elips1: TMenuItem;
    Warna1: TMenuItem;
    Merah1: TMenuItem;
    Hijau1: TMenuItem;
    Biru1: TMenuItem;
    Shape1: TShape;
    procedure FormActivate(Sender: TObject);
    procedure  Bentuk1MeasureItem(Sender:  TObject;  ACanvas:
TCanvas;
      var Width, Height: Integer);
    procedure  Segiempat1DrawItem(Sender:  TObject;  ACanvas:
TCanvas;
      ARect: TRect; Selected: Boolean);
    procedure Elips1DrawItem(Sender: TObject; ACanvas: TCanvas;
      ARect: TRect; Selected: Boolean);
    procedure Segiempat1Click(Sender: TObject);
    procedure Elips1Click(Sender: TObject);
    procedure Warna1Click(Sender: TObject);
    procedure Warna1DrawItem(Sender: TObject; ACanvas: TCanvas;
      ARect: TRect; Selected: Boolean);
    procedure  Warna1MeasureItem(Sender:  TObject;  ACanvas:
TCanvas;
      var Width, Height: Integer);
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormActivate(Sender: TObject);
begin
  Merah1.Tag := clRed ;
  Biru1.Tag  := clBlue ;
  Hijau1.Tag := clGreen ;
end;

procedure TForm1.Bentuk1MeasureItem(Sender: TObject; ACanvas:
TCanvas;
  var Width, Height: Integer);
begin
  Width := 100;
  Height:= 50 ;
end;

procedure  TForm1.Segiempat1DrawItem(Sender:  TObject;  ACanvas:
TCanvas;
  ARect: TRect; Selected: Boolean);
begin
  // mengeset warna background color dan membuat background
  if Selected then ACanvas.Brush.Color := clHighlight
  else ACanvas.Brush.Color := clMenu;
  ACanvas.FillRect (ARect);
  // menggambar bentuk
  ACanvas.Brush.Color := clWhite;
  InflateRect (ARect, -5, -5);
  ACanvas.Rectangle (ARect.Left, ARect.Top,ARect.Right,
ARect.Bottom);
end;

procedure TForm1.Elips1DrawItem(Sender: TObject; ACanvas:
TCanvas;
  ARect: TRect; Selected: Boolean);
begin
  // mengeset warna background color dan membuat background
  if Selected then ACanvas.Brush.Color := clHighlight
  else ACanvas.Brush.Color := clMenu;
  ACanvas.FillRect (ARect);
  // menggambar bentuk
  ACanvas.Brush.Color := clWhite;
  InflateRect (ARect, -5, -5);
  ACanvas.Ellipse(ARect.Left, ARect.Top,ARect.Right,
ARect.Bottom);
end;

procedure TForm1.Segiempat1Click(Sender: TObject);
begin
  Shape1.Shape := stRectangle ;
end;
```

```
procedure TForm1.Elips1Click(Sender: TObject);
begin
  Shape1.Shape := stEllipse ;
end;

procedure TForm1.Warna1Click(Sender: TObject);
begin
  Shape1.Brush.Color := (Sender as TComponent).Tag
end;

procedure TForm1.Warna1DrawItem(Sender: TObject; ACanvas:
TCanvas;
  ARect: TRect; Selected: Boolean);
begin
  // menentukan warna background dan menggambarnya
  if Selected then ACanvas.Brush.Color := clHighlight
  else ACanvas.Brush.Color := clMenu;
  ACanvas.FillRect (ARect);
  // menampilkan warna
  ACanvas.Brush.Color := (Sender as TComponent).Tag;
  InflateRect (ARect, -5, -5);
  ACanvas.Rectangle (ARect.Left, ARect.Top,ARect.Right,
ARect.Bottom);
end;

procedure TForm1.Warna1MeasureItem(Sender: TObject; ACanvas:
TCanvas;
  var Width, Height: Integer);
begin
  Width := 80 ;
  Height := 30 ;
end;

end.
```



Gambar 6.9 Menu Bentuk



Gambar 6.10 Menu Warna

6.6 Frame Sederhana

Frame adalah sejenis panel yang dapat dipakai pada saat perancangan aplikasi seperti pada form. Anda dapat membuat sebuah frame, menambahkan komponen-komponen ke dalamnya dan menambahkan event handler-nya. Perbedaan frame dengan form adalah frame hanya menempati sebagain area form dan frame tidak mempunyai batas tepi sehingga sebaiknya Anda menambahkan komponen **Bevel** dengan **Align=alClient.** Setelah frame selesai, Anda dapat membuka sebuah form, memilih komponen **Frame** dari page **Standard** dan memilih sebuah frame yang ada di project yang bersangkutan. Setelah menambahkan frame, Anda akan melihat komponen-komponen dari frame dikopikan ke form. Jika Anda mengubah frame aslinya, perubahan tersebut juga akan terjadi pada setiap frame yang dipakai pada form.

Sebagai contoh pertama kita akan membuat aplikasi yang memakai satu frame.

1. Buat aplikasi baru.
2. Tambahkan frame dengan memilih **File|New|Frame**, ditampilkan frame kosong.
3. Frame tidak berbingkai, oleh sebab itu sebaiknya ditambahkan komponen **Bevel** dari page **Additional**. Supaya ukuran Bevel=ukuran frame, ubah **Align=alClient**.

4. Ke dalam frame, tambahkan **Edit, ListBox,** dan dua **Button**, masing-masing dengan nama **BTambah, BHapus**.

Gambar 6.11 Contoh Frame

5. Tombol **Tambah** untuk menambahkan data yang diisikan pada kotak edit ke dalam ListBox. Isikan event handler **OnClick**.
6. Tombol **Hapus** untuk menghapus data pada ListBox. Isikan event handler **OnClick**.
7. Simpan unit frame tersebut dengan nama **Frame1**.

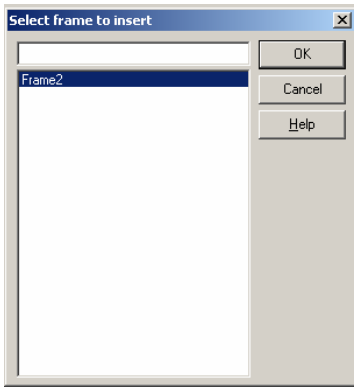
Listing program frame:

```
unit Frame1;

interface

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

type
  TFrame2 = class(TFrame)
    Bevel1: TBevel;
    Edit1: TEdit;
    ListBox1: TListBox;
    Btambah: TButton;
    BHapus: TButton;
    procedure BtambahClick(Sender: TObject);
    procedure BHapusClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
implementation

{$R *.dfm}

procedure TFrame2.BtambahClick(Sender: TObject);
begin
  If Edit1.Text <> '' then ListBox1.Items.Add(Edit1.Text) ;
end;

procedure TFrame2.BHapusClick(Sender: TObject);
begin
  If ListBox1.ItemIndex >=0 then
ListBox1.Items.Delete(ListBox1.ItemIndex);
end;

end.
```

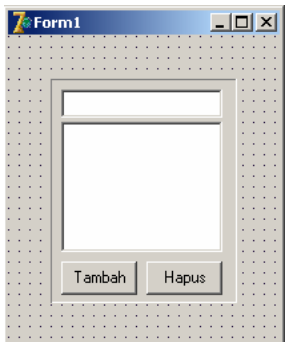
Langkah berikutnya melengkapi form utama.

1. Aktifkan form utama. Tambahkan komponen **Frame** (komponen pertama dari page **Standard**). Ditampilkan kotak dialog Select Frame to Insert.



Gambar 6.12 Kotak Dialog Select Frame to Insert

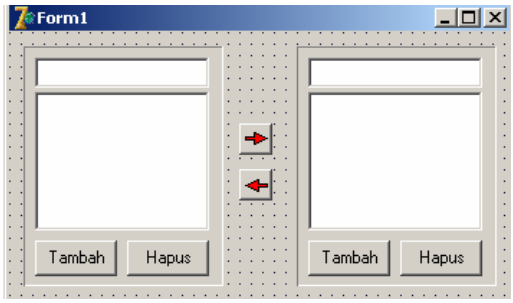
2. Oleh karena hanya ada satu frame, pilih frame tersebut dan klik **OK**. Di form utama ditampilkan frame beserta semua komponen di dalam frame.



Gambar 6.13 Frame di dalam Form Utama

Coba jalankan aplikasi tersebut. Jadi, apa gunanya frame? Dalam aplikasi tersebut frame hanya digunakan sekali. Frame akan sangat berguna jika kita gunakan berulang kali, karena kita tidak perlu mendefinisikan sebuah komponen beserta event handler-nya berulang kali. Ubahlah aplikasi tersebut.

1. Tambahkan sebuah frame lagi ke dalam aplikasi.
2. Tambahkan dua buah **SpeedButton** dari page **Additional**. Isikan properti **Glyph** dengan salah satu file gambar yang ada di direktori ..\Common Files\Borland Shared\Images\Button.

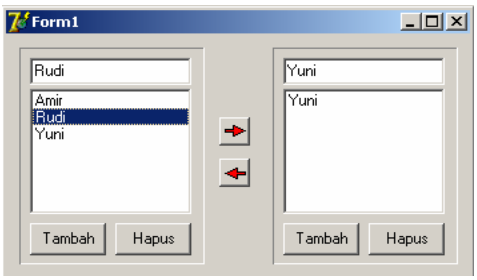


Gambar 6.14 Form dengan Dua Instance dari Frame

3. Tombol panah kiri untuk mengkopikan data dari frame pertama ke frame kedua. Sebaliknya, fungsi dari tombol panah kanan. Isikan event handler **OnClick** dari kedua tombol.

Listing form utama:

```
unit UFrame1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, Frame1, Buttons;

type
  TForm1 = class(TForm)
    Frame21: TFrame2;
    Frame22: TFrame2;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  Frame22.ListBox1.Items.AddStrings(Frame21.ListBox1.Items);
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  Frame21.ListBox1.Items.AddStrings(Frame22.ListBox1.Items);
end;

end.
```



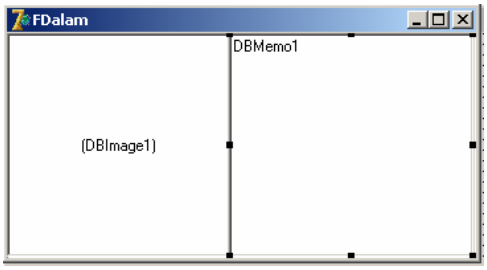
Gambar 6.15 Dua Frame dalam Sebuah Form

6.7 Frame Bersarang

Frame dapat bersarang, artinya frame di dalam frame lain.

Langkah-langkah membuat frame dalam:

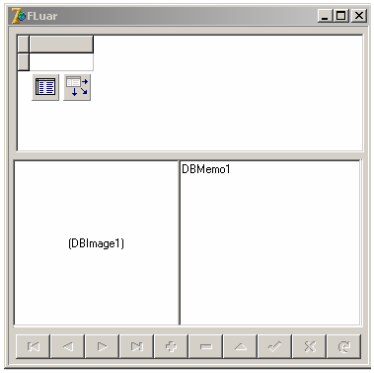
1. Buatlah aplikasi baru. Tambahkan sebuah frame dan beri nama **FDalam**.
2. Tambahkan **Bevcl** dengan **Align=alClient**.
3. Tambahkan **DBImage** dan **DBMemo** dari page **Data Controls**.
4. Simpan dengan nama **FDalam.pas**.



Gambar 6.16 Frame Dalam

Langkah selanjutnya membuat frame luar yang berisi frame dalam:

1. Tambahkan sebuah frame dan beri nama **FLuar**.
2. Tambahkan **Bevcl** dengan **Align=alClient**.
3. Tambahkan komponen **Table** dari page **BDE**. Isi **DatabaseName= DBDEMOS**.
4. Tambahkan **DataSource** dari page **DataAccess**. Isi **DataSet=Table1**.
5. Tambahkan **DBGrid** dan **DBNavigator** dari page **Data Controls**. Isi properti **DataSource** dari kedua komponen tersebut dengan **DataSource1**.
6. Tambahkan komponen **Frame** dan pilih **FDalam**. Atur posisinya sehingga seperti pada Gambar 6.17.
7. Simpan dengan nama **UFLuar.pas**.



Gambar 6.17 Frame Luar

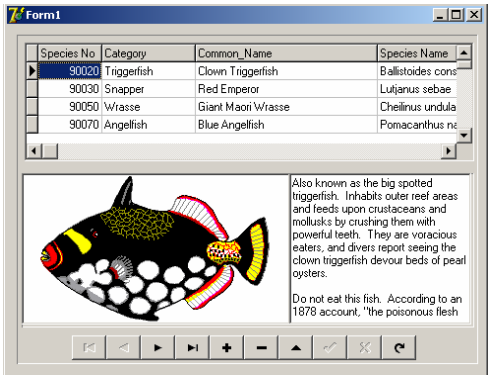
Tahap terakhir melengkapi form utama.

1. Tambahkan komponen **Frame** dan pilih **FLuar**.
2. Isikan event handler **OnCreate** dari form. Procedure akan mengisikan nama tabel pada komponen **Table** dan mengisikan properti **DataSource** dan **DataField** dari komponen **DBMemo** dan **DBImage**.

Listing form utama:

```
unit UFBersarang;

interface

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

type
  TForm1 = class(TForm)
    FLuar1: TFLuar;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
uses UFDalam;

{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
begin
  with FLuar1 do
  begin
    Table1.TableName := 'BioLife';
    with FDalam1 do
    begin
      DBMemo1.DataSource := Datasource1;
      DBMemo1.DataField := 'Notes';
      DBImage1.DataSource := Datasource1;
      DBImage1.DataField := 'Graphic';
    end;
    Table1.Open;
  end ;
end;

end.
```

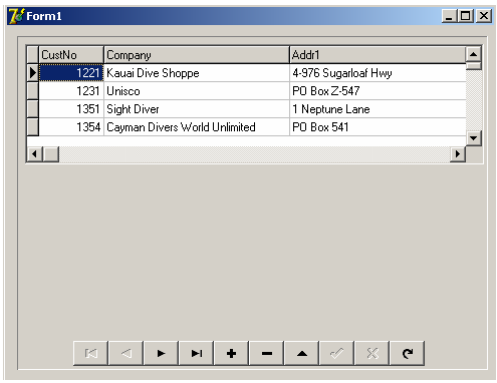


Gambar 6.18 Frame Bersarang

6.8 Membuang Frame

Pada contoh di atas kita memakai tabel **Biolife** yang mempunyai field bertipe **Graphic** dan **Memo**. Bagaimana jika tabel yang dipakai tidak mempunyai kedua field tersebut? Jadi, komponen **DBImage** dan **DBMemo** tidak diperlukan. Dalam hal ini, Anda dapat membuang frame dalam dengan method **Free**. Coba Anda ubah procedure **OnCreate** dari form utama menjadi seperti berikut.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  with FLuar1 do
  begin
    Table1.TableName := 'Customer';
    FDalam1.Free ;
    Table1.Open ;
  end ;
end;
```



Gambar 6.19 Frame Dalam Dibuang

6.9 Frame Master Detil

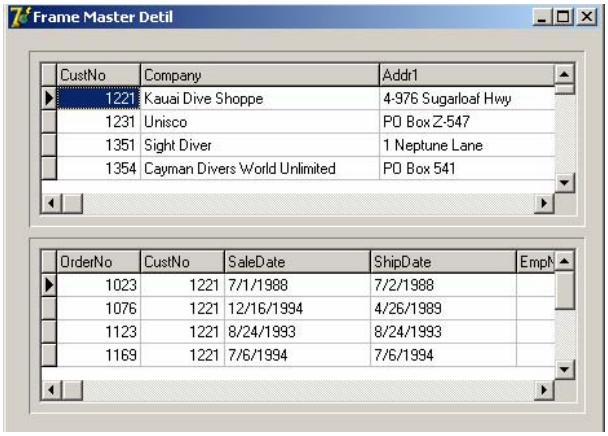
Hubungan master detil dalam database adalah satu record pada tabel induk berhubungan dengan beberapa record di dalam tabel anak. Untuk menampilkan hubungan tersebut, diperlukan dua **Table**, dua **DataSource** dan dua **DBGrid**. Untuk mempermudah, Anda dapat memakai frame.

1. Buatlah aplikasi baru dan buatlah frame dengan nama **FData**.
2. Ke dalam frame tambahkan **Bevel** yang **alClient**, **Table** dengan **DatabaseName=DBDEMOS**, **DataSource** dan hubungkan ke komponen **Table**, serta **DBGrid** yang dihubungkan ke **DataSource**.
3. Simpan dengan nama **UFData.pas**.

4. Kembali ke form utama. Tambahkan dua frame dari jenis yang sama.
5. Tambahkan event handler **OnCreate** dari form untuk menyatakan nama tabel di setiap frame dan meng-hubungkannya.

Listing lengkap form utama:

```
unit UFMasterDetil;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, UFData;

type
  TForm1 = class(TForm)
    FData1: TFData;
    FData2: TFData;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  with FData1 do
  begin
    Table1.TableName := 'Customer';
    Table1.Open ;
  end ;

  with FData2 do
  begin
    Table1.TableName := 'Orders';
    Table1.MasterSource := FData1.DataSource1 ;
    Table1.IndexName    := 'CustNo' ;
    Table1.MasterFields := 'CustNo' ;
    Table1.Open ;
  end ;
end;

end.
```



Gambar 6.20 Frame Master Detil