

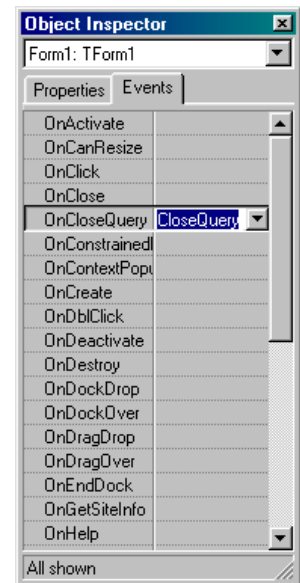
### 3.2. Az alkalmazások ablaka

1. Alkalmazást kilépési jóváhagyással [CanClose](#)
2. Akciólista használata. [Akcio](#)
3. Egéresemények használata [Egeresemeny](#)
4. Billentyűzet események kezelése [Billesemeny](#)
5. Keretek használata [Frame](#)

A feladat elvégzéséhez nincs más dolgunk, mint a form **OnCloseQuery** eseményét kikeresni az **Object Inspector Events** lapján. Ha kétszer rákattintunk az eseményre, bekerül a TForm1 objektumba a **FormCloseQuery** metódus.

```
type
  TForm1 = class(TForm)
    procedure FormCloseQuery(Sender: TObject;
                           var CanClose: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

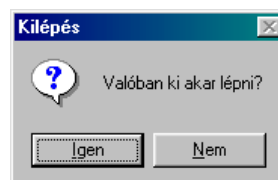
A felhasználói környezet szövegszerkesztőre vált és a szövegszerkesztő tartalma éppen az eseménykezelő. Az **OnCloseQuery** esemény az ablak zárásakor aktivizálódik és jellegzetessége, hogy a *CanClose* cím szerinti paraméter segítségével értéket ad vissza.



Használjuk az **TApplication** osztály **MessageBox** metódusát arra, hogy párbeszédablak segítségével rákérdezhessünk, valóban be szeretnénk-e zárni az ablakot. A metódus első paramétere a kérdés, amelyik megjelenik a párbeszédablakban, a második az ablak címsora. A harmadik paraméter szabályozza azt, hogy milyen gombok, milyen ikonok jelenjenek meg, hogyan működjön a párbeszédablak.

A függvény visszatérési értéke attól függ, hogy melyik gombot nyomta meg a felhasználó.


```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  // Ha a felhasználó a Yes gombot nyomja kiléphet
  if Application.MessageBox('Valóban ki akar lépni?', 'Kilépés', mb_YesNo) = IDYES then
    CanClose:=true
  else
    CanClose:=false;
end;
```

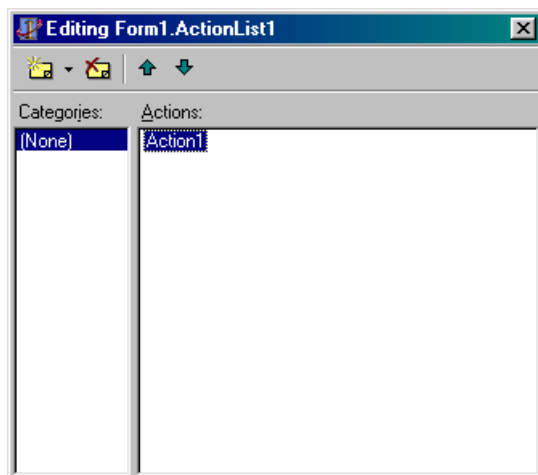




Tervezzünk alkalmazást, melyből a kilépést menüből, vagy gyorsítógombról kezdeményezhetjük! Használjuk az akciólistát a tevékenységek összehangolására! (*Akcio*)



Helyezzük először az űrlapra az **ActionList** komponenst! (Ezzel az **ActnList** modul automatikusan bekerül a programba.) Kattintsunk kétszer a komponensen. A megjelenő **ActionList** párbeszédablakban a **NewAction** nyomógommbal  vagy az <Ins> billentyű megnyomásával létrehozhatjuk az **Action1** akciót.



A objektumkezelőben írjuk át a **Caption** tulajdonságot '**Kilépés**'-re! Ha kétszer kattintunk az **Actions** lista **Action1** elemén, akkor megírhatjuk a futtatandó akciót.

```
procedure TForm1.Action1Execute(Sender: TObject);
begin
    // Közös akció az ablak zárása
    Form1.Close;
end;
```

Helyezzük el a formon a menüt és a **SpeedButton** komponenst. A **MainMenu1** elemnek legyen - a szokásoknak megfelelően **Fájl** főmenüje, melynek egyetlen legördülő menüeleme a **Kilépés**. Az automatikusan elkészülő objektumok neve **Fjl1** és **Kilps1** lesznek. Az utóbbi **Bitmap** tulajdonságához - az objektumkezelőben - hozzárendeljük a **Kilep1.bmp** állományt. Ugyanezt a bitképet rendeljük a **SpeedButton1.Glyph** tulajdonsághoz.

Mivel azt szeretnénk, hogy a **Kilépés** menü és a **SpeedButton1** ugyanazt a tulajdonságot kezelje, mindkét elem **Action** tulajdonságát **Action1**-re állítjuk. Érdekes megjegyezni, hogy ha menü **Caption** tulajdonsága nem lett volna '**Kilépés**', az akció hozzárendelés hatására ez – mint a gomb esetén - automatikusan megtörténik.

Azt szeretnénk, ha programunk indulásakor nem engedélyezze a **Kilépés** menü és a **SpeedButton1** használatát, azonban az ablakon történő egérekattintáskor a kilépés engedélyezettsége változzon. A form létrehozásakor ezért nem engedélyezzük az akciót.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    // Létrehozáskor tiltjuk az akciót
    Action1.enabled:=false;
end;
```

Ennek hatására a menü és a gomb egyaránt letiltott lesz.

Az ablakon való kattintás ellenkezőjére váltja a kilépés engedélyezést.

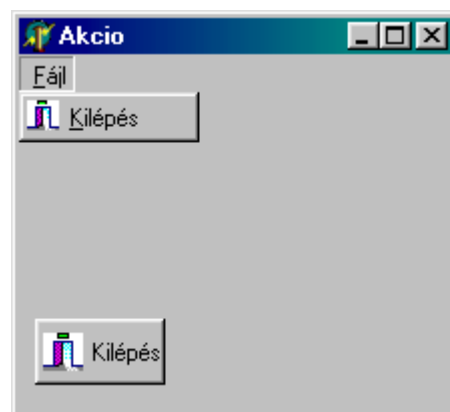
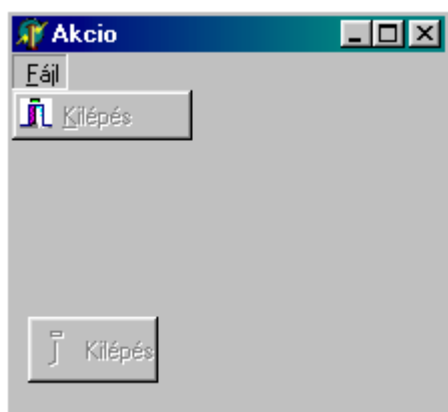
```
procedure TForm1.FormClick(Sender: TObject);
begin
    // Váltjuk a kilépésengedélyezést
    Action1.enabled:=not (Action1.enabled);
end;
```

A [CanClose](#) példából ismert kilépés-jóváhagyó párbeszédablakot használjuk a *OnCloseQuery* esemény kezelésére.

Érdemes áttekinteni az *TForm1* típust is.

```
type
    TForm1 = class(TForm)
        MainMenu1: TMainMenu;
        Fj11: TMenuItem;
        Kilps1: TMenuItem;
        SpeedButton1: TSpeedButton;
        ActionList1: TActionList;
        Action1: TAction;
        procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
        procedure Action1Execute(Sender: TObject);
        procedure FormClick(Sender: TObject);
        procedure FormCreate(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;
```

Az alábbi ábrák a program futását mutatják egérekattintás előtt és után.





A Delphi felirat megjelenítéséhez a **Label** komponenst használjuk. Helyezzük el a **Label1** komponenst a formon! Állítsuk a **Label1.Caption** tulajdonságot 'Delphi'-re! Beállíthatjuk azt is, hogy a **Label1.AutoSize** igaz legyen. Állítsuk be betűtípust. (**Label1.Font.Name**='Times New Roman', **Label1.Font.Color**=clBlue, **Label1.Font.Size**=19)!

Úgy fogunk dolgozni, hogy az egérgomb lenyomásakor megjegyezzük az egér pozícióját és az egér mozgásakor – ha a bal gomb lenyomott - az aktuális pozíció és az előző pozíció különbségével változtatjuk meg a címke pozícióját.

Az előző egérpozíció tárolásához szükségünk lesz a Form moduljában globálisan deklarált változóra.

```
var  
    ponte:TPoint;
```

Az egérgomb lenyomásakor küldő objektum lokális koordináarendszerében értelmezett *X* és *Y* koordinátákat átalakítjuk képernyő koordinátává. Ez azért szükséges, mert így egyféleképpen kezelhetjük majd a form és a címke felületét. Az átalakításhoz a **TControl** osztály **ClientToScreen** metódusát használhatjuk. A függvény paramétere az átalakítandó pont visszatérési értéke a képernyőpont.

```
function ClientToScreen(const Point: TPoint): TPoint;
```

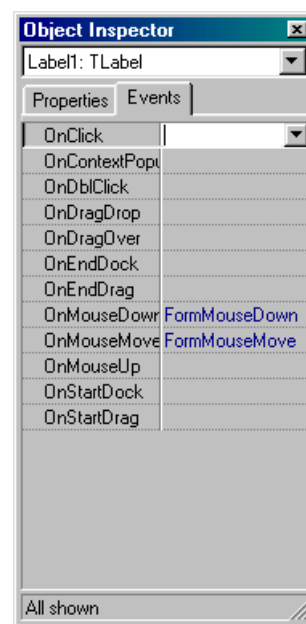
A képernyő koordinátát a *ponte* változóban elmentjük.

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;  
    Shift: TShiftState; X, Y: Integer);  
    var pont:TPoint;  
begin  
    // Az egérgomb lenyomásakor megjegyezzük az egérkoordinátát  
    pont.x:=x;  
    pont.y:=y;  
    // Minden adatot képernyőkoordinátában tárolunk  
    ponte:=TControl(Sender).ClientToScreen(pont);  
end;
```

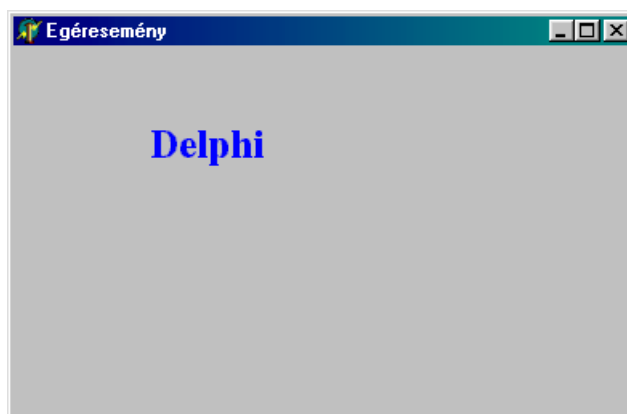
Egérmozgatáskor, ha a bal egérgomb le van nyomva. akkor elmozdítjuk a címkét és az új pozíciót, mint előzőt megjegyezzük.

```
procedure TForm1.FormMouseMove(Sender: TObject;  
    Shift: TShiftState; X,Y: Integer);  
    var pont:TPoint;  
begin  
    // Egérmozgatáskor a balgomb lenyomott,  
    // mozgatjuk a feliratot  
    if ssLeft in Shift then  
        begin  
            pont.x:=x;  
            pont.y:=y;  
            // Képernyőkoordináta  
            pont:=TControl(Sender).ClientToScreen(pont);  
            Label1.left:= Label1.left+pont.x-ponte.x;  
            Label1.top:= Label1.top+pont.y-ponte.y;  
            // Megjegyezzük az előzőt  
            ponte:=pont;  
        end;  
end;
```

Végül intézkednünk kell arról, hogy a címkén ugyanazok az egéresemények aktivizálódjanak, mint a formon. Ezt legegyszerűbben az objektum-felügyelőben tehetjük meg a **Label1 OnMouseDown** és **OnMouseMove** eseménykezelők beállításával.



A program futási képe:




A tervezés alatt állítsuk be az ablak címsorát úgy, hogy az üres **string** legyen. Ahhoz, hogy minden egyes betű megjelenjen a címsorban az **OnKeyPress** eseményt kell kezelnünk. Az esemény *Key* - karakter típusú - bemenő paramétere tartalmazza a lenyomott gomb kódját. Az abc betűinek kódja 47-nél nagyobb, a szóköz kódja 32, elegendő ezeket a karaktereket megjeleníteni, ha ezt a megszorítást nem tesszük, akkor a Windows a nem látható karaktereket egy téglalappal jeleníti meg. A billentyűzet eseménykezelője tehát minden gombnyomás esetén egy karaktert illeszt a címsorhoz.

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
    // Gombnyomáskor, ha a billentyű kódja A-nál nagyobb (betű), vagy szóköz
    // a címbe kerül
    if (Ord(key)>47) or (Ord(key)=32) then
        Form1.Caption:=Form1.Caption+Key;
end;
```

Ha azt szeretnénk elérni, hogy a **<Backspace>** gombbal az utolsó karakter eltűnjön a címsorból, akkor használhatjuk az **OnKeyDown** eseményt. A **<Backspace>** billentyű virtuális kódja *VK\_BACK*.

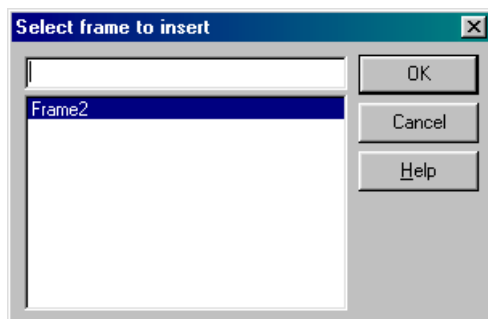
```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
begin
    // visszatörlésnél rövidítjük a címet
    if (key=VK_BACK) and (length(Caption)>0) then
        begin
            caption:=copy(caption,1,length(caption)-1);
        end;
end;
```




 Tervezzünk alkalmazást, mely ugyanazt a keretbe helyezett címkét jeleníti meg kétszer és a billentyűzés a két címkében fordított irányban jelenik meg! (*Frame*)

A billentyűzetüzenetek kezelésével nem lesz gondunk, hiszen ugyanúgy kell eljárni, mint a [Billesemeny](#) példánál. Példánk inkább a keretek lehetőségeit szándékozik bemutatni.

Első lépésben válasszuk ki a **File|New frame** menüpontot. Ennek hatására megjelenik a *Frame* modellje, mintha forma lenne. Erre a keretre helyezzük a címkét és beállítjuk, hogy a felső szélhez igazodjon. Állítsuk a **Wordwrap** tulajdonságot igazra és az **Autosize** tulajdonságot hamisra, hogy a címke többsoros legyen, és húzzuk az alját a keret aljáig. Ezzel elkészítettük a keretet, visszatérhetünk a form programozásához.



Ha visszatérünk a formhoz, akkor a **Frames** komponenst  kiválasztva és a formon kattintva a megjelenő párbeszédablakból kiválasztjuk a keretet és kétszer a formra tesszük. Az egyiket **Frame21** az ablak felső széléhez igazítjuk, a másikat **Frame22** az alsóhoz.

A form címsorát állítsuk be az '*Oda-vissza ír*' stringnek megfelelően!

A keret típusa *Unit2*-ben

```
type
  TFrame2 = class(TFrame)
    Label1: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

A kereteket megjelenítő form típusa

```
type
  TForm1 = class(TForm)
    Frame21: TFrame2;
    Frame22: TFrame2;
  procedure FormKeyPress(Sender: TObject; var Key: Char);
  procedure FormCreate(Sender: TObject);
  procedure FormKeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

Az **OnCreate** eseményben töröljük le a keretek címkéit!

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Frame21.Label1.caption:='';
  Frame22.Label1.caption:='';
end;
```



A billentyűzetkezelés majdnem megegyezik a [Billesemeny](#) programmal, mindössze az a különbség, hogy egyszerre kell kezelni a két keret címkéjét.

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
    // Gombnyomáskor, ha a billentyű kódja A nál nagyobb (betű), vagy szóköz
    // a két címkébe kerül ellentétes sorrendben
    if (Ord(Key)>47) or (Ord(Key)=32) then
        begin
            Frame21.Label1.caption:=Frame21.Label1.caption+Key;
            Frame22.Label1.caption:=Key+Frame22.Label1.caption;
        end
    end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
begin
    // visszatörlésnél rövidítjük mindkét feliratot
    if (Key=VK_BACK) and (length(Caption)>0) then
        begin
            Frame21.Label1.caption:=copy(Frame21.Label1.caption,1,
                length(Frame21.Label1.caption)-1);
            Frame22.Label1.caption:=copy(Frame22.Label1.caption,2,
                length(Frame22.Label1.caption)-1);
        end
    end;
end;
```

A program futási képe jól szemlélteti a működést.

