

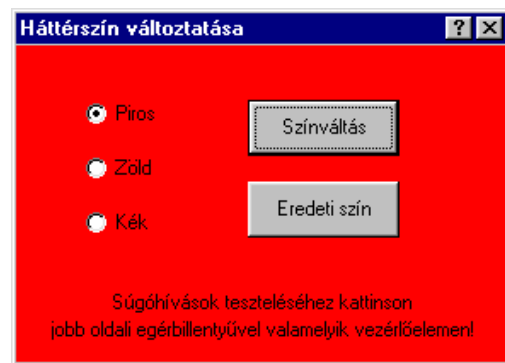
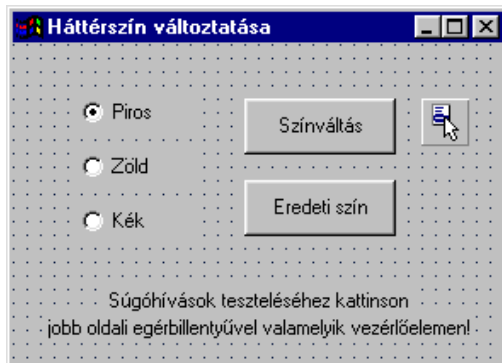
## 8.5. Sógórendszer készítése Delphi alkalmazásokhoz

1. Alkalmazás környezet-érzékeny súgóval [HatterSzin](#)
2. Animációs alkalmazás környezet-érzékeny súgóval [Forgasok](#)
3. Játékprogram névjeggyel [Lovolde](#)



Készítsünk egy egyszerű súgófájlt, illetve egy alkalmazást, a környezet-érékeny súgó hívásainak bemutatására! (*HatterSzin*)

Annak érdekében, hogy a súgóhívási lehetőségekre tudjunk koncentrálni, az alkalmazásunk működése legyen nagyon egyszerű! Helyezzünk a formra három választógombot (**RadioButton**) a különböző színek kiválasztására, és két nyomógombot (**Button**): az egyikük felirata legyen *Színváltás*, a másiké pedig „*Eredeti szín*”! Ha a felhasználó rákattint a *Színváltás* gombra, az alkalmazás háttere színessé változik, ha pedig az „*Eredeti szín*” gombra kattint, a háttérszín visszaváltozik világosszürkére.

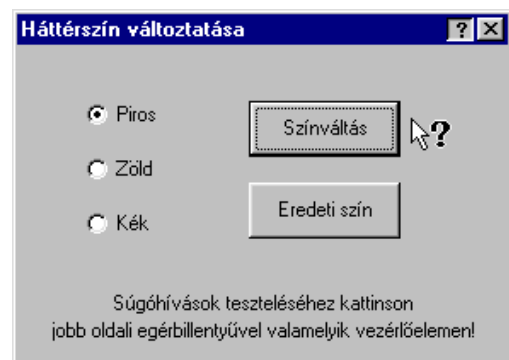


```
procedure TForm1.Button1Click(Sender: TObject);
begin
    if RadioButton1.Checked then Form1.Color:=clRed           // színváltás pirosra
    else if RadioButton2.Checked then Form1.Color:=clGreen    // színváltás zöldre
    else Form1.Color:=clBlue;                                // színváltás kékre
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    Form1.Color:=clBtnFace; // színváltás világosszürkére
end;
```

Az alkalmazás célja a kontext-érékeny súgóhívási lehetőségek bemutatása, amelyekből kettőt valósítunk meg. Az egyik súgóhívási mód a „*Mi ez?*” felbukkanó menü („*nyomógomb*”), a másik pedig az át nem méretezhető ablakok – általában párbeszédablakok – fejlécsorában található kérdőjel alakú gomb megjelenítése, illetve alkalmazása súgóhívásokhoz.

A két megoldás közül a második módszer az egyszerűbb. Állítsuk át a **Form** objektum **BorderStyle** tulajdonságának értékét *bsDialog*-ra, a **BorderIcons** tulajdonság *biHelp* mezőjét pedig *true*-ra! Az alkalmazás elindítása után - a kérdőjelogomb csak futási időben jelenik meg, és csak a nem átméretezhető ablakokban - az ablak fejlécében láthatóvá válik a súgóhívó gomb, amelyre kattintva az egérmutatató kérdőjel alakúra változik, jelezve, hogy az alkalmazásunk súgóhívási állapotba került.

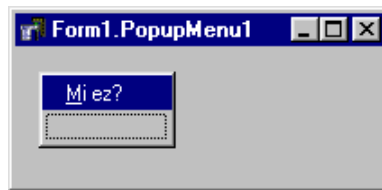


Ahhoz, hogy a súgóhívási állapotban valamelyik vezérlőelemre kattintva valóban megjelenjen a vezérlőelem rendeltetését magyarázó súgófejezet, három dologra van szükségünk. Egyrészt létre kell hozni a súgófejezeteket tartalmazó súgófájlt, másrészt be kell állítani az alkalmazásunk **HelpFile** tulajdonságát a súgófájl nevére, valamint a vezérlőelemek **HelpContext** tulajdonsága értékeként meg kell adni a megfelelő súgófejezet kontextszámát.

Az esetek többségében kényelmesebb fordított sorrendben dolgozni, először beállítjuk a vezérlőelemek kontextszámait, és utána hozzuk létre a súgófájlt, mely súgófejezeteihez hozzárendeljük a programban használt számozósítókat. A példában a vezérlőelemek a következő kontextszámokat kapták meg:

<i>Name</i>	<i>Caption</i>	<i>HelpContext</i>
<i>Button1</i>	<i>Színváltás</i>	1
<i>Button2</i>	<i>Eredeti szín</i>	2
<i>RadioButton1</i>	<i>Piros</i>	3
<i>RadioButton2</i>	<i>Zöld</i>	4
<i>RadioButton3</i>	<i>Kék</i>	5

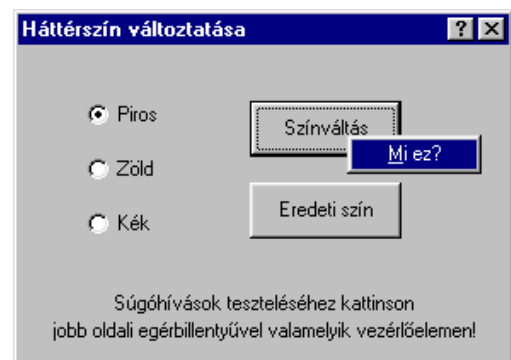
Mielőtt hozzákezdenénk az alkalmazásunk súgófájljának elkészítéséhez, valósítsuk meg a „*Mi ez?*” felbukkanó menüpontot is! Ehhez helyezzünk el a formon egy **PopupMenu** vezérlőelemet, melynek **Items** tulajdonságán keresztül adjuk meg a „&Mi ez?” feliratú menüpontot:



Kétszer kattintva a létrehozott menüponton, adjuk meg a következő eseménykezelő eljárást:

```
procedure TForm1.MieziClick(Sender: TObject);
begin
    Application.HelpCommand(HELP_CONTEXTPOPUP,
        (PopupMenu1.PopupComponent as TWinControl).HelpContext);
end;
```

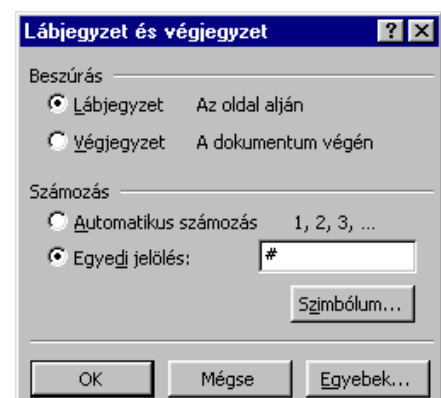
A fenti eljárásban az **Application** objektum egyik súgóhívási metódusa, a **HelpCommand** segítségével egy kisebb keretben való megjelenítést kérjük (**HELP\_CONTEXTPOPUP**) annak a súgófejezetnek, amely kontextszámát beállítottuk a felbukkanó menüt kiváltó vezérlőelem (**PopupMenu1.PopupComponent**) **HelpContext** tulajdonságában. A kontextmenü akkor bukkan fel, amikor a felhasználó jobb oldali egérgombbal rákattint a vezérlőelemre, ha a vezérlőelem **PopupMenu** tulajdonsága értékeként megadtuk a kontextmenü nevét (példánkban ez **PopupMenu1**).

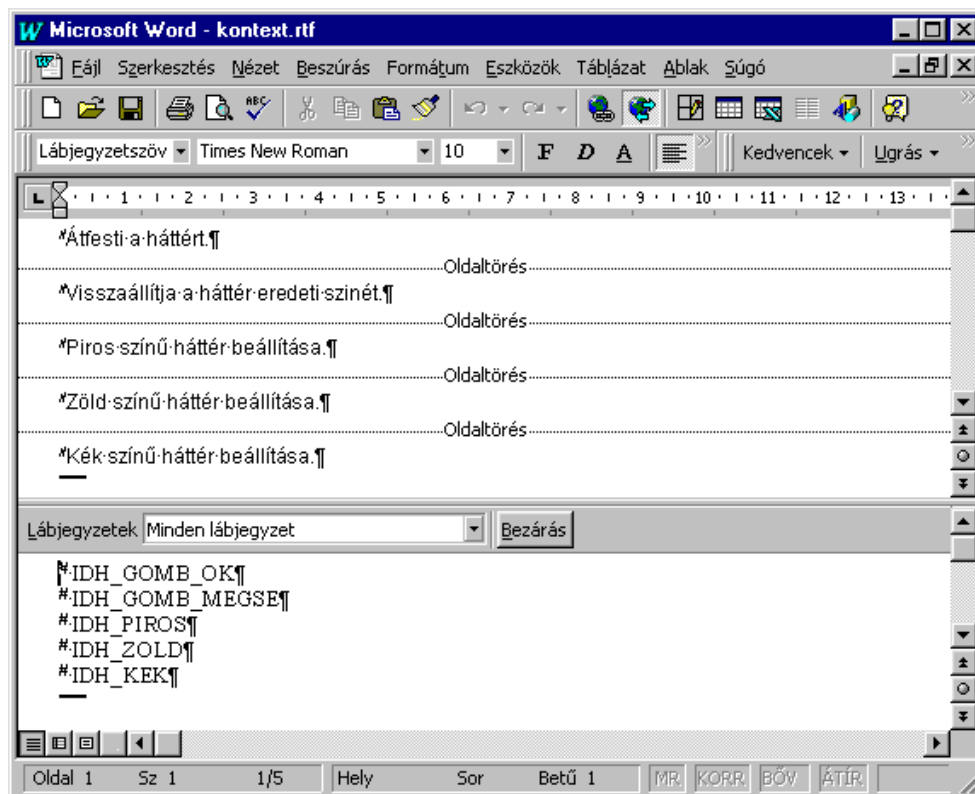


Most pedig kezdjük hozzá a súgófájl készítéséhez!

A **Word** szövegszerkesztőben gépeljük be az alkalmazásunk vezérlő-elemeinek használatát magyarázó szövegét. A különböző vezérlőelemekre vonatkozó szövegeket – az ún. súgófejezeteket - oldaltörési jellel választjuk el egymástól (**Beszűrés** | **Törés** | **Oldaltörés**)!

Minden súgófejezetnek adjunk meg egy ún. fejezetazonosítót, amely első betűit a **IDH\_** karaktersorozat adja! Ehhez a szövegkurzort állítsuk be a súgófejezet elejére, és **Beszűrés** | **Lábjegyzet** menüpont kiválasztása után adjuk meg először a # jelet, mint lábjegyzetjelet, utána pedig a lábjegyzet ablakában gépeljük be a súgófejezet **IDH\_**xxx alakú fejezetazonosítóját.



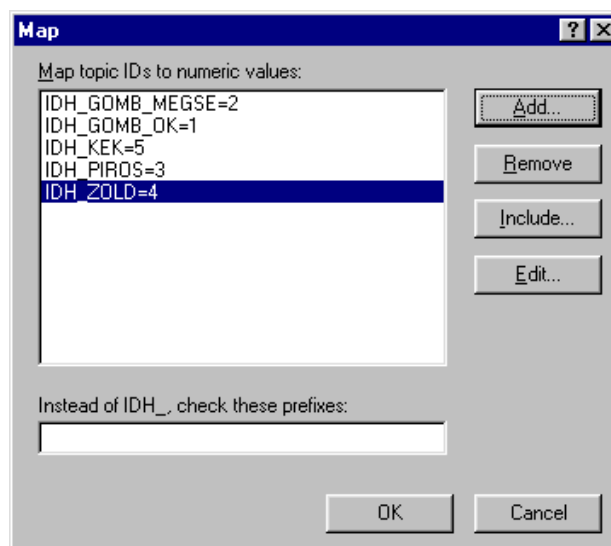


A kész tematikus fájlt mentjük el .RTF formátumban (**Fájl | Mentés másként**, és a **Fájltípus** listából válasszuk ki a *Rich Text Formátum* bejegyzést)! Jelen példában a *kontext.rtf* nevet használtuk.

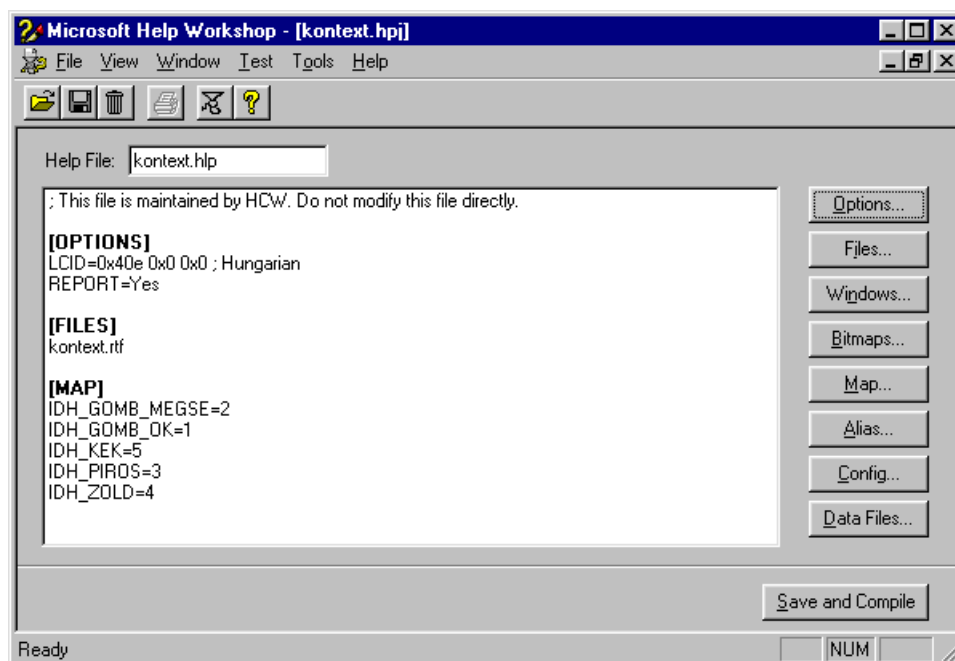
Indítsuk el a **Help Workshop** programot (amit megtalálunk a telepített Delphi rendszer *C:\Program Files\Borland\Delphi5\Help\Tools* könyvtárban *HCW.EXE* néven), és válasszuk ki az új sűgőprojekt készítését (**File | New | Help Project**)! Kényelmi szempontból ajánlatos a projektfájlt ugyanabba a könyvtárba menteni, ahová a tematikus állományt (.RTF).

A **Help Workshop** ablakában kattintsunk a **Files** nyomógombon, és a megjelenő párbeszédablakban az **Add** nyomógombon való kattintás után adjuk meg a tematikus fájlnk nevét (*kontext.rtf*)!

Ezek után a **Help Workshop** ablakában kattintsunk a **Map** nyomógombon, és a megjelenő párbeszédablakban – kattintgatva az **Add** nyomógombon - egymás után adjuk meg a tematikus fájlban szereplő fejezetazonosítókat (**Topic ID**) - és a Delphi programban a vezérlők **HelpContext** tulajdonságában beírt kontextszámok (**Mapped numeric value**) párosítását.



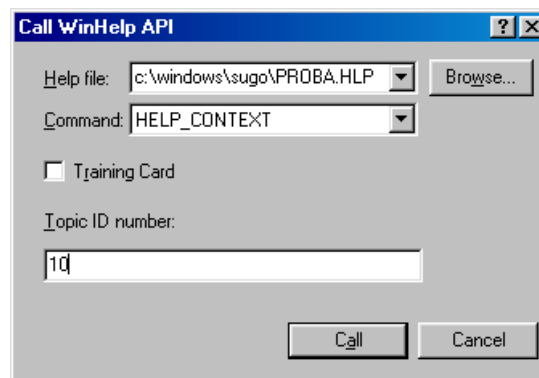
Miután az **OK** nyomógomb segítségével lezártuk a **Map** párbeszédablakot, a súgófájl fordítását a „**Save and Compile**” nyomógombon való kattintással indíthatjuk el (a fordítási üzeneteket tartalmazó ablakból a szerkesztési ablakba a **Window** menü segítségével léphetünk vissza).



A fordítás során létrejött súgófájl (.HLP) teszteléséhez válasszuk ki a **Test** menüből a „**WinHelp API**” menüpontot, a megjelenő párbeszédablakban pedig a „**Help file**” mezőben adjuk meg a tesztelni kívánt súgófájl nevét! A **Command** listából válasszuk ki a „**HELP\_CONTEXTPOPUP**” parancsot, a „**Topic ID number**” mezőbe pedig írjuk be valamelyik kontextszámot (a példánkban az 1-től 5-ig terjedő számokat használtuk)! Ezek után kattintsunk a **Call** nyomógombon!

Hosszabb tematikus fájlok esetén kényelmesebb, ha a fejezetazonosítók és kontextszámok párosítását nem egyenként adjuk meg, hanem létrehozunk egy külön fejlécfájlt („**Csak szöveg**” típusú és .H kiterjesztéssel), amelyben következő példához hasonló módon adjuk meg a párokat:

```
#define IDH_GOMB_OK      1
#define IDH_GOMB_MEGSE  2
#define IDH_PIROS        3
#define IDH_ZOLD         4
#define IDH_KEK          5
```



Ebben az esetben a **Help Workshop** ablakában a **Map** gombon kattintva egyszerűen meg kell adnunk a fejlécfájl nevét, miután rákattintottunk az **Include-**, utána pedig a **Browse** nyomógombra. A párosításokat tartalmazó fejlécfájl súgóprojekthez való hozzákapcsolása után csak a súgóállomány fordítása marad (**Save and Compile**).

Az egyszerű, de az alkalmazások kontext-érzékeny súgója megvalósításához teljes mértékben elegendő, súgófájl létrehozása után a súgóhívásokat már tartalmazó programunkban még egy fontos lépést kell megtennünk, be kell állítanunk a súgófájl nevét. Ezt az alkalmazásunk projektbeállításai között (**Project | Options, Application** lap, „**Help file**” mező) megadva, vagy pedig futási közben, a form betöltésekor is megtehetjük:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    Application.HelpFile:='kontext.hlp';
end;
```

Lehetséges még egy olyan beállítási módszer is, amikor a súgófájlt csak bizonyos formokhoz rendeljük hozzá (többablakú alkalmazásokban):

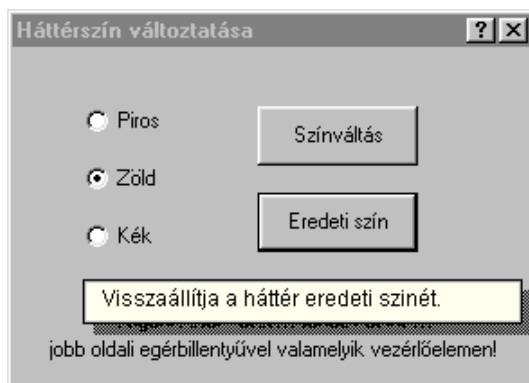
```
Form1.HelpFile := "KONTEXT.HLP";
```

### ***Végül egy megjegyzés:***

Ha az alkalmazásunkban több olyan eseménykezelőt definiálunk, amelyek egyszerre több vezérlő működését is leírják, érdemes használni az **ActionList** vezérlőelemet (**Standard** lap). Ilyenkor a felbukkanó menü(k) **OnClick** eseményének nem a saját, alapértelmezett eseménykezelő nevét kell beállítani, hanem az **ActionList** elem segítségével meghatározott akció nevét.

Az akció létrehozásához kattintsunk kétszer az **ActionList** vezérlőelemen, majd egyszer, jobb oldali egérbillentyűvel a megjelenő szerkesztési ablakban! Válasszuk ki a felbukkanó menüből a „**New Action**” menüpontot, és kattintsunk kétszer a létrehozott akciónéven! Maga az akciókód teljesen megegyezik a fentiekben látott **TForm1.Miez1Click** eseménykezelő eljárással:

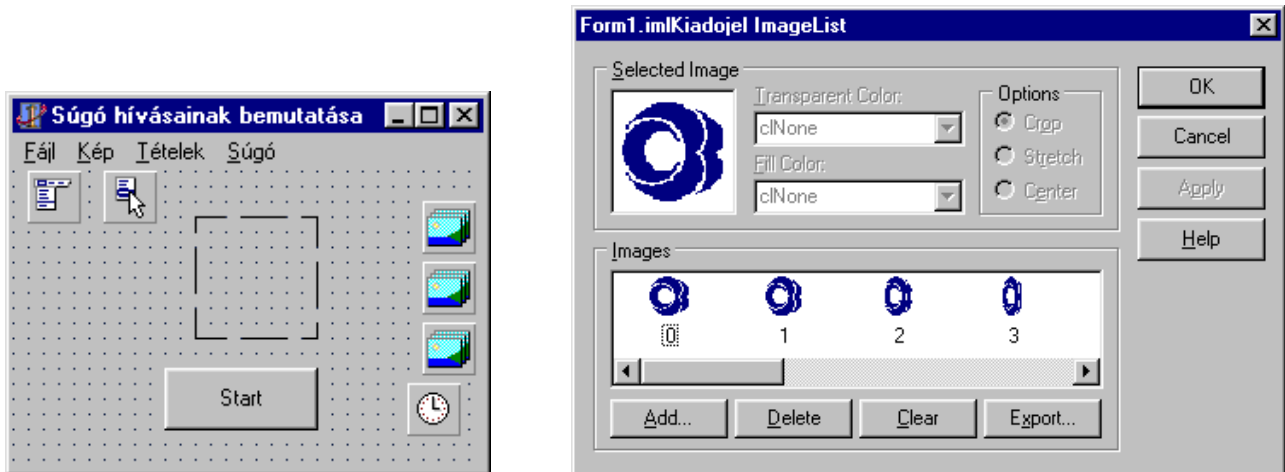
```
procedure TForm1.Action1Execute(Sender: TObject);  
begin  
    Application.HelpCommand(HELP_CONTEXTPOPUP,  
        (PopupMenu1.PopupComponent as TWinControl).HelpContext);  
end;
```





Készítsünk animációs alkalmazást a környezet-érzékeny és a menün keresztül hívható sűrű megvalósításának bemutatására! Készítsük el a feladat sűrűhívásaihoz szükséges három sűrűállományt is! (*Forgások*)

Az alkalmazás űrlapján helyezzük el a következő vezérlőelemeket: egy képmezőt (*Image*), egy *Start* feliratú nyomógombot (*Button*), egy főmenüt (*MainMenu*), egy felbukkanó menüt (*PopupMenu*), egy időzítőt (*Timer* az *Interval* tulajdonságot állítsuk be 0-ra), illetve három képsorozatot (*ImageList*), amelyekre kétszer kattintva adjuk meg az animációhoz szükséges képeket (*Add* gomb)!



A főmenü a következő *Caption* tulajdonságú menüpontokat tartalmazza:

<i>&amp;Fájl</i>	<i>&amp;Kép</i>	<i>&amp;Tételek</i>	<i>&amp;Sűrű</i>
<i>&amp;Kilépés</i>	<i>&amp;Rózsa</i>	<i>&amp;1. tétel</i>	<i>&amp;Témakörök és index</i>
	<i>K&amp;utyus</i>	<i>&amp;2. tétel</i>	-
	<i>&amp;Kiadójel</i>	<i>&amp;Több tétel...</i>	<i>&amp;Mi ez?</i>

Az alkalmazásban megvalósuló animáció lényege, hogy miután a felhasználó rákattintott a *Kép* menüben valamelyik menüpontra, az ennek megfelelő képsorozat első képkockája betöltődik a képkeretbe (*Image* vezérlőelem). A *Start* gombra való kattintás után - a *Timer* vezérlő által generált jelre - elkezdődik a képek *Image* objektumban való váltása – ezzel elérjük az animációs hatást. A *Start* gomb felirata a kattintás után változik *Stop*-ra (és fordítva), így az animációt meg tudjuk állítani, illetve újraindítani.

A *Start* gomb eljáráskezelőjét kicsit később, az időzítőjét pedig most tárgyaljuk:

```

procedure TForm1.tmrIdozitoTimer(Sender: TObject);
begin
  Kocka[Forgo] := Kocka[Forgo] + ElojelesEgy[Forgo]; // számláló növelése/csökkenése

  case Forgo of
    2: Kocka[Forgo] := Kocka[Forgo] mod Forgando.Count; // kutyuskép
    // rózsa, illetve kiadójel:
    1, 3: // Ha elértük a képsorozat első, illetve utolsó képkockáját
          // (az i számláló túllépte
          // a sorozat első, illetve felső indexhatárát):
    if (Kocka[Forgo] = -1) or (Kocka[Forgo] = Forgando.Count) then
      begin
        ElojelesEgy[Forgo] := ElojelesEgy[Forgo]*(-1); // forgásirány váltása
        // Visszalépés a második, illetve az utolsó előtti képkockára:
        Kocka[Forgo] := Kocka[Forgo]+2*ElojelesEgy[Forgo];
      end;
    end;
  KepMegjelenites(Kocka[Forgo], Forgando); // a Kocka indexű képkocka megjelenítése
end;

```

A fenti eseménykezelő eljárásban szereplő *Forgo*, *Forgando*, *Kocka* és *ElojelesEgy* globális változók, amelyek a program elindításakor kapják meg a kezdőértékeiket:

```
implementation
const
  KepSzam = 3;
var
  Kocka: array [1..KepSzam] of integer;
  ElojelesEgy: array [1..KepSzam] of integer;
  Forgando: TImageList;
  Forgo: Longint;
...

procedure TForm1.FormCreate(Sender: TObject);
var i: integer;
begin
  for i:=1 to KepSzam do
    begin
      Kocka[i] := 0;           // az elsőként megjelenő képkocka száma
      ElojelesEgy[i] := 1;    // képváltás iránya (növekvő vagy csökkenő sorrend)
    end;
    Forgo:=1;                 // elsőként a rózsakép jelenik meg az Image keretben
    KepMegjelenites(0,imlRozsa); // a rózsakép megjelenítése az Image keretben
  end;
```

A *KepMegjelenites* lokális eljárás, amelyet mind az időzítő eseménykezelő eljárásából, mind pedig a *FormCreate* eljárásból meghívunk.

```
procedure KepMegjelenites(Kocka:integer; Lista: TImageList);
begin
  // a Kocka-dik képkocka megjelenítése az Image keretben:
  Lista.GetBitmap(Kocka, Form1.imgKep.Picture.Bitmap);
  Form1.imgKep.Refresh;      // a kép frissítése
  Forgando:= Lista;         // az aktuálisan animált képsorozat beállítása
end;
```

A képsorozat váltásához rá kell kattintanunk valamelyik menüpontra a *Kép* menüből. A menü mindegyik menüpontjának **OnClick** eseménykezelője azonos lehet, ha statikusan beállítjuk a menüpontok **Tag** tulajdonságában a következő eljárásban is látható értékeket (sorban 1, 2, illetve 3):

```
procedure TForm1.mnuKepsorozatClick(Sender: TObject);
begin
  // az eseménykezelő eljárást kiváltó menüpont sorszáma:
  Forgo:=(Sender as TMenuItem).Tag;
  // megfelelő kép megjelenítése az Image keretben:
  case Forgo of
    1: KepMegjelenites(0,imlRozsa);
    2: KepMegjelenites(0,imlKutyus);
    3: KepMegjelenites(0,imlKiadojel);
  end;
end;
```

Az eddig ismertett eljárások csupán az alkalmazásunk „háttértevékenységét”, az animációt hivatottak elvégezni (ahol a „háttértevékenység” jelző természetesen csak a jelenleg megtanulandó anyag szempontjából igaz). Az alkalmazás pillanatnyilag számunkra fontosabb része a sűgőhívások megvalósítása (a szükséges sűgőfájlok elkészítését később részletezzük).

Az alkalmazásunkban összesen három különböző sűgőállományt használunk: a vezérlőelemek rendeltetését magyarázó *kontext.hlp* fájlt - kontext-érzékeny sűgőhívásokhoz, a *Tételek* menün keresztül hívható vicces tételeket tartalmazó *tetelek.hlp*-t, illetve a *Sűgő* menün keresztül hívható *sugo\_pld.hlp* állományt, amely az alkalmazásunk használatára vonatkozó információkat tartalmazza. Ezeken kívül megvalósítjuk az ún. sűgőhívási állapot bekapcsolását is a *Sűgő* menü „Mi ez?” menüpontja segítségével.



Kezdjük ezzel az utolsóval! Ahhoz, hogy a „Mi ez?” menüpontra való kattintás után az egérmutató kérdőjeles nyíl alakját vegye fel (ha eddig nem volt pipával bejelölve a menüpont), a következő lépésekre lesz szükségünk:

```
procedure TForm1.mnuMiezClick(Sender: TObject);
begin
  with Screen do
    if Cursor=crDefault           // ha a kurzor alakja - egyszerű nyíl
    then begin
      Cursor:=crHelp;             // kurzoralak megváltoztatása kérdőjeles nyíllá
      mnuMiEz.Checked:=true;      // menüpont "kipipálása"
    end
    else begin
      Cursor:=crDefault;          // kurzoralak megváltoztatása egyszerű nyíllá
      mnuMiEz.Checked:=false;     // menüpont melletti pipának eltüntetése
    end;
end;
```

Ahhoz, hogy a „Mi ez?” állapot kiváltása esetén az alkalmazás vezérlőelemein való kattintás után megjelenjen a kontext-érzékeny súgó, statikusan be kell állítani a vezérlőelemek **HelpContext** tulajdonságának értékét a megfelelő súgófejezetet jelölő kontextszámra, illetve az **Application** objektum **HelpFile** tulajdonságát a fejezeteket tartalmazó súgóállomány nevére. Jelen megoldásban a *kontext.hlp* a megfelelő súgófájl neve, a benne meghatározott kontextszámok pedig a következők: **Start** gomb: 1, **Image** vezérlőelem: 6. Mivel azonban az **Image** objektum nem rendelkezik külön **HelpContext** tulajdonsággal, a kontextszámot a **Tag** tulajdonságában adhatjuk meg, így a súgóhívás következőképpen néz ki:

```
procedure TForm1.imgKepClick(Sender: TObject);
begin
  if Screen.Cursor=crHelp then    // ha az egérmutató alakja - kérdőjeles nyíl
  begin
    Application.HelpFile := 'kontext.hlp'; // súgófájl nevének megadása
    Application.HelpCommand(HELP_CONTEXTPOPUP, imgKep.Tag); // súgóhívás
  end;
end;
```

A **Start** gomb esetén a szokásos módon, a **HelpContext** tulajdonságon keresztül, érjük el a megfelelő kontextszámot:

```
procedure TForm1.cmdStartClick(Sender: TObject);
const
  szStartStop: array [0..1] of string = ('Start', 'Stop');
  i: integer = 0;
begin
  if Screen.Cursor=crHelp then    // ha a kurzor alakja - kérdőjeles nyíl
  begin
    Application.HelpFile := 'kontext.hlp'; // súgófájl nevének megadása
    Application.HelpCommand(HELP_CONTEXTPOPUP, cmdStart.HelpContext); // súgóhívás
    MiEzAllapotKikapcsolasa;
  end
  else
  begin
    // A Start/Stop gomb feliratának megfelelően elindítjuk, illetve megállítjuk
    // (Interval = 0) az animációt, és átírjuk a gomb feliratát:
    i:=1-i;
    tmrIdozito.Interval:= i*100;
    cmdStart.Caption:= szStartStop[i];
  end;
end;
```

A fenti eljárásban látható *MiEzAllapotKikapcsolasa* lokális eljárás hívására azért van szükség, hogy a súgómegjelenítést követően az egérmutató alakja visszaváltozzon egyszerű nyíllá. Csak ezzel a mutatóalakkal tudjuk ki- és bekapcsolni az időzítőt, illetve az animációt (lásd a fenti eljárás **else** ágát):

```

procedure MiEzAllapotKikapcsolasa;
begin
  if Screen.Cursor=crHelp then
    begin
      Screen.Cursor:=crDefault;
      Form1.mnuMiEz.Checked:= false;
    end;
end;

```

Ahogy láttuk, mind az *imgKepClick*, mind pedig a *cmdStartClick* eljárásban dinamikusan állítjuk be az *Application* objektum *HelpFile* tulajdonságának értékét. Az ilyen megoldás elkerülhetetlen abban az esetben, amikor egy alkalmazásban több súgófájltra van szükség. Jelen esetben a *kontext.hlp* állományon kívül két további fájlt is alkalmazunk. A *tetelek.hlp* beállítására akkor van szükség, ha a felhasználó a *Tételek* menüből választ egy menüpontot:

```

procedure TForm1.mnuTetelClick(Sender: TObject);
begin
  MiEzAllapotKikapcsolasa;
  Application.HelpFile := 'tetelek.hlp';    // súgófájl neve
  // Kontextszámnak megfelelő tétel megjelenítése:
  Application.HelpCommand(HELP_CONTEXT, (Sender as TMenuItem).HelpContext);
end;

```

Láthatjuk, hogy ebben az esetben az *Application* objektum *HelpCommand* metódusának hívását nem a kontextérzékeny súgó hívásainál alkalmazott *HELP\_CONTEXTPOPUP* értékkel, hanem a *HELP\_CONTEXT* értékkel végezzük. Ennek hatására egy külön súgóablakban jelenik meg a metódus második paraméterében meghatározott kontextszámú súgófejezet. Természetesen ehhez a menüpontok *HelpContext* tulajdonságát előbb megfelelő értékekre kell beállítani (a példában ez sorban 1, 2 és 3 az *1. tétel*, a *2. tétel* és a *Több tétel...* menüpont esetén).

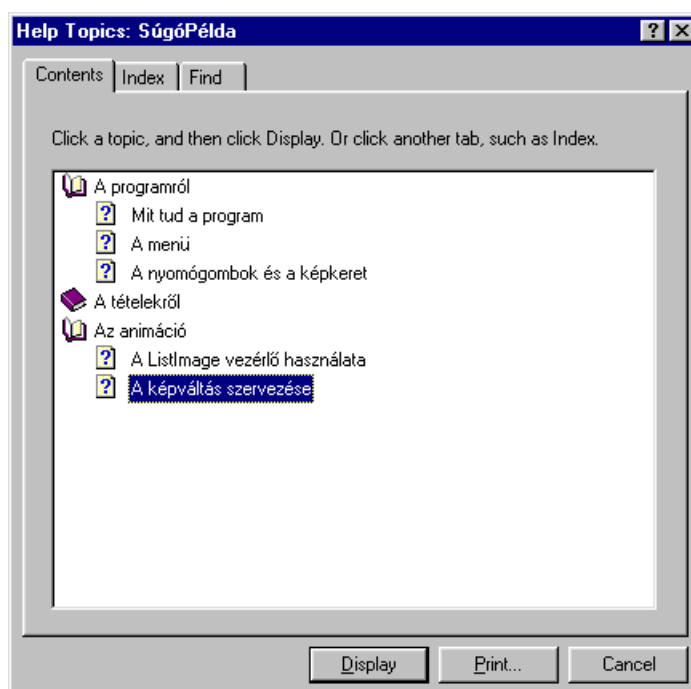
Az alkalmazásunkkal használt harmadik - a *sugo\_pld.hlp* - fájl beállítására akkor van szükség, ha a felhasználó kiválasztja a *Súgó* menü „*Témakörök és index*” menüpontját:

```

procedure TForm1.mnuTemakorokClick(Sender: TObject);
begin
  MiEzAllapotKikapcsolasa;
  Application.HelpFile := 'sugo_pld.hlp';    // súgófájl neve
  // A súgóablak Tartalom, illetve Tárgymutató lapján való megnyitása:
  Application.HelpCommand(HELP_FINDER, 0);
end;

```

Az *Application.HelpCommand* eljárás első paraméterének értéke ebben az esetben a *HELP\_FINDER*, amelynek hatására a súgó hárompaneles ablakát jeleníti meg a rendszer. Ennek első lapján a súgófájl fordításakor megadott *.CNT* állomány (jelen példában ez *sugo\_pld.cnt*) tartalma lesz látható:



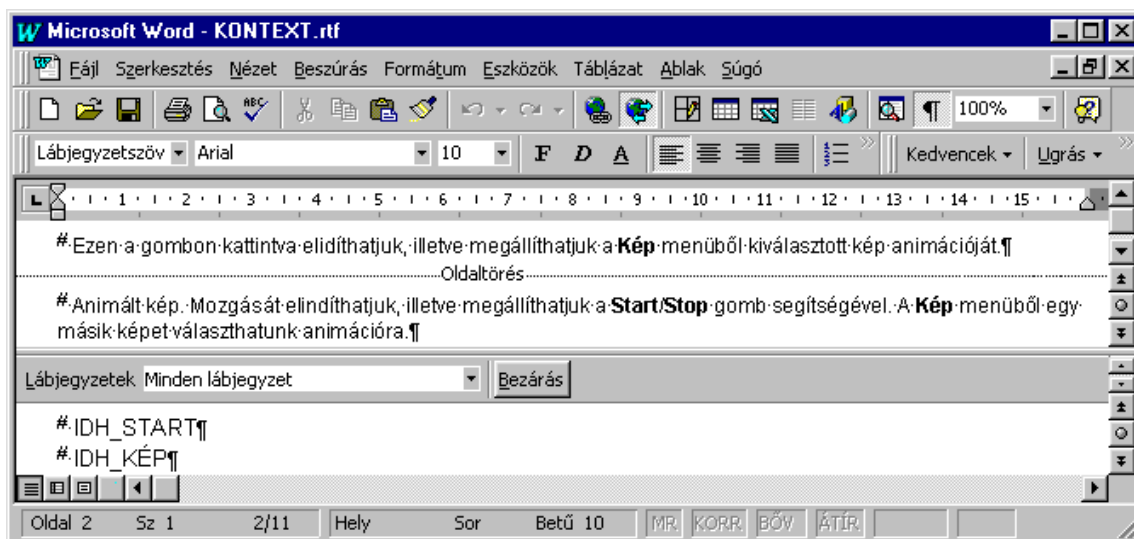
Az alkalmazásunkban a Windows rendszerben megszokott módon, a jobb oldali egérgombbal való kattintás hatására megjelenő „**Mi ez?**” súgóhívási „gombot” is szeretnénk használni. A gomb valójában egyetlen-egy menüpontot tartalmazó felbukkanó menü, ezért kattintsunk kétszer a formon elhelyezett **PopupMenu** vezérlőelemen, és alakítsunk ki egy „**Mi ez?**” feliratú menüpontot! A menüpont eseménykezelő eljárása a következőképpen alakul:

```
procedure TForm1.pmnuMiezClick(Sender: TObject);
begin
    // A megjelenítendő súgófájl nevének megadása:
    Application.HelpFile := 'kontext.hlp';
    // Súgóhívás:
    with PopupMenu1 do
        if PopupComponent is TButton
            then Application.HelpCommand(HELP_CONTEXTPOPUP, ~
                (PopupComponent as TWinControl).HelpContext)
        else
            if PopupComponent is TImage
                then Application.HelpCommand(HELP_CONTEXTPOPUP, (PopupComponent as TComponent).Tag);
    end;
```

Mivel Windows alatt a „**Mi ez?**” gomb használata környezet-érzékeny súgóhívásokat jelent, súgófájlként a *kontext.hlp* állományt kell beállítanunk. A *pmnuMiezClick* eljárás hívását kiváltó vezérlőelem tesztelése pedig azért különbözik a **Button** és az **Image** vezérlőelem esetén, mert ahogy ezt már megemlítettünk, az **Image** objektumok nem rendelkeznek **HelpContext** tulajdonsággal, így csak a **Tag** tulajdonságban tudjuk megadni a kontextszámot.

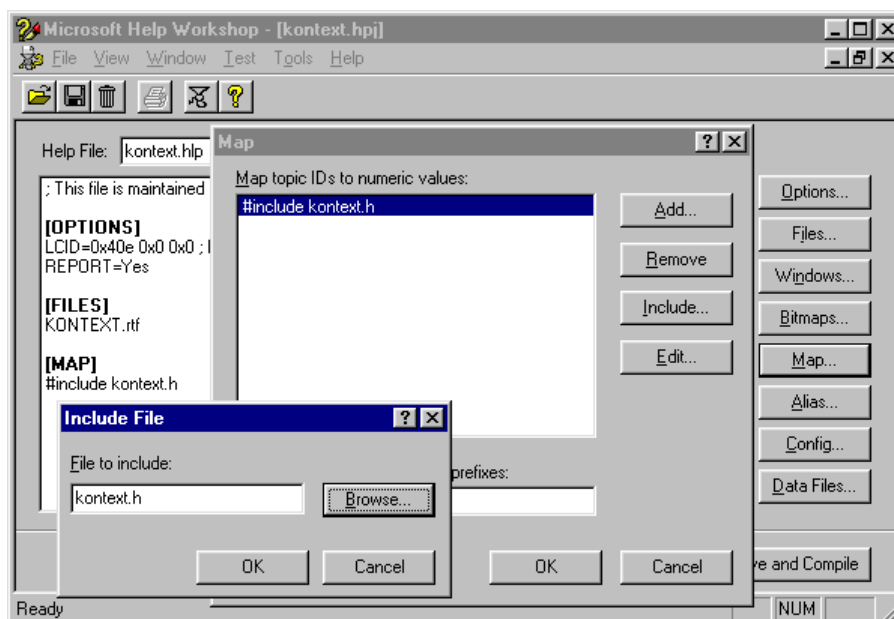
Végezetül adjuk meg a **Button**, illetve az **Image** vezérlőelem súgóbuborékjainak szövegét (például *Animáció* és *Animált kép*) a **Hint** tulajdonságukban!

Az alkalmazás környezet-érzékeny súgóját (*kontext.rtf*) az előző feladat megoldásában ismertetett módon készítsük el úgy, hogy a kontextszámokat és a fejezetazonosítókat egy fejlécfájl (*kontext.h*, amely egy egyszerű szöveges fájl .H kiterjesztéssel) segítségével adjuk hozzá a súgó projektfájljához!

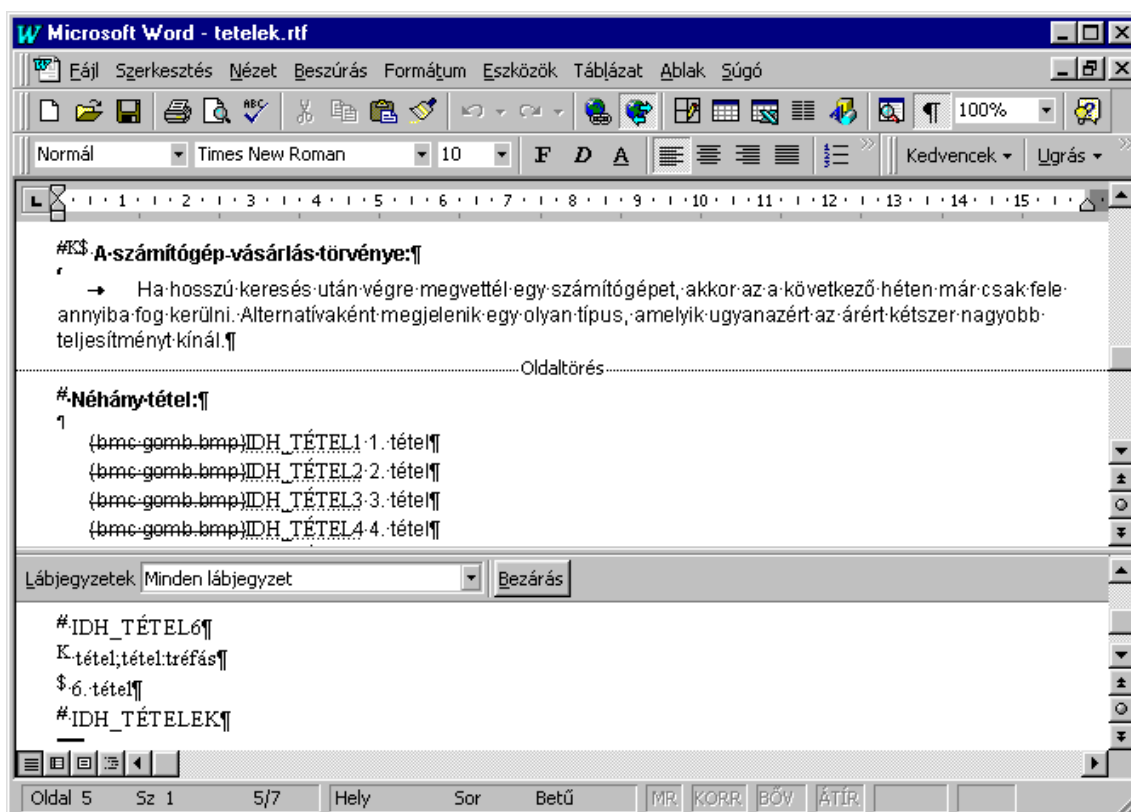


A *kontext.h*: tartalma

```
#define IDH_START 1
#define IDH_KÉP 2
```



A vicces tételeket tartalmazó, menü keresztül hívható *tetelek.hlp* súgófájl elkészítését a környezet-érzékeny súgóhoz hasonló lépésekkel kezdjük (szöveg begépelése, oldalra való tördelés, a # jelt követő fejezetazonosítók megadása lábjegyzetben, RTF-állományként való mentés).



Ahhoz, hogy a „Több tétel” súgófejezet megjelenítésekor a súgóablakban „forró pontként” működő gombok is legyenek, hozzuk létre a gombot tartalmazó BMP fájlt (*gomb.bmp*)! Ezek után megfelelő módon adjuk meg a súgófejezet szövegében a hivatkozásokat a képfájlokra, illetve azokra a súgófejezetekre, amelyek megjelennek a gombon való kattintás után. Például:

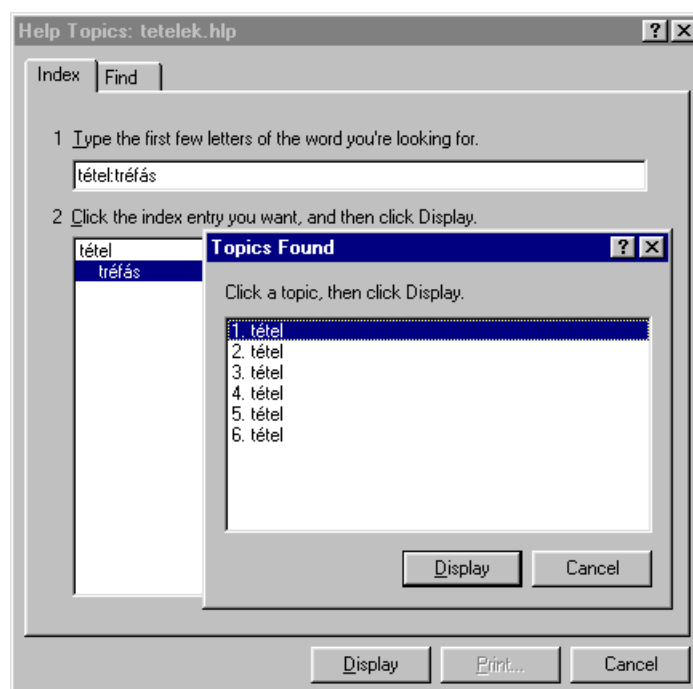
{bmc-gomb.bmp} 1. tétel

A gomb-képfájljára való hivatkozást áthúzott betűvel, a hivatkozott súgófejezet fejezetazonosítóját rejtett betűvel, a megjelenő szöveget pedig mindenféle speciális formázások nélküli betűvel adjuk meg.



Ahhoz, hogy a különböző tételeket a súgó hárompaneles párbeszédablakának *Index* lapjáról is meg lehessen keresni, adjuk meg a súgófejezet *K* jelű lábjegyzeteit is, amelyek az ún. kulcsszavakat (keresési címszavakat) jelölik! Példánkban az összes súgófejezetnek – kivéve az előbb tárgyaltat – a következő *K*-lábjegyzetet adtuk:

*K* tétel;tétel:tréfás



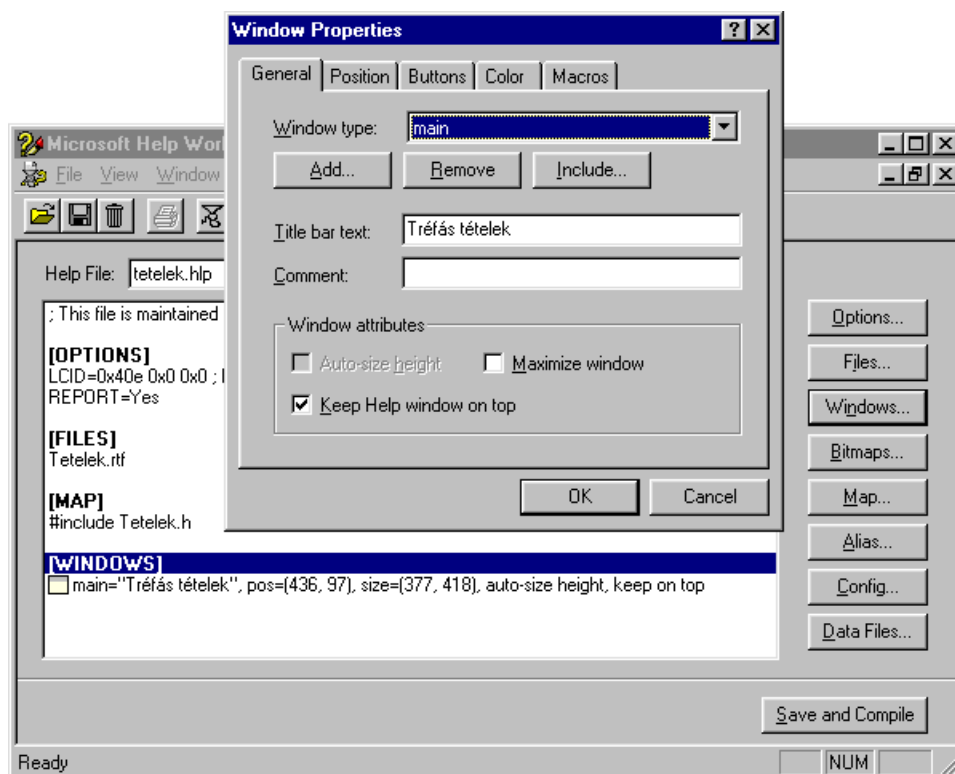
A *K*-lábjegyzettel rendelkező súgófejezetek a *\$* jelű lábjegyzetben megadott címszavakkal (keresési címkével) is rendelkeznek, például:

*\$* 6. tétel

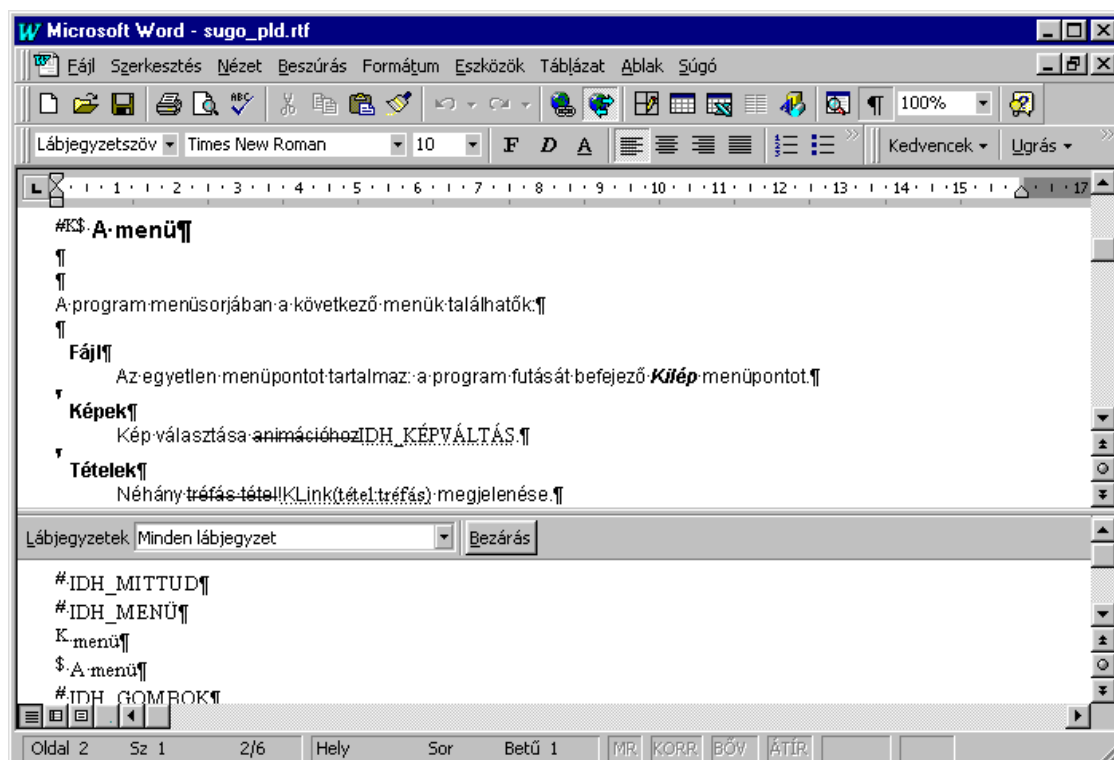
Ezek a címszavak a fenti ábrán is látható, megtalált súgófejezeteket tartalmazó, párbeszédablakban jelennek meg, de csak akkor, ha több súgófejezetnek ugyanazt a szót adtunk meg a *K*-lábjegyzetében.

A Microsoft *Help Workshop*-ben az új súgóprojekt elindítása után adjuk meg a létrehozott tematikus állomány nevét (a **Files** gomb)! Ha azt akarjuk, hogy a tételek a fentiekben is látsz, sárga háttérű, kisebb méretű súgóablakban jelenjenek meg, meg kell szerkesztenünk ezt a külön ablakot. Kattintsunk a **Windows** gombon, és a megjelenő párbeszédablak paneljain adjuk meg a következő ablaktulajdonságokat: **General** – általános információk (az ablak típusa, neve, hogyan jelenjen meg), **Position** – az ablak bal felső sarkának koordinátái, illetve szélessége és magassága, **Buttons** – az ablak fejléce alatt megjelenő gombok be-, illetve kikapcsolása, **Color** – az ablak álló és görgetéssel mozgatható részeinek színei, **Macros** – az ablak megjelenítésekor végrehajtandó speciális súgómakrók.

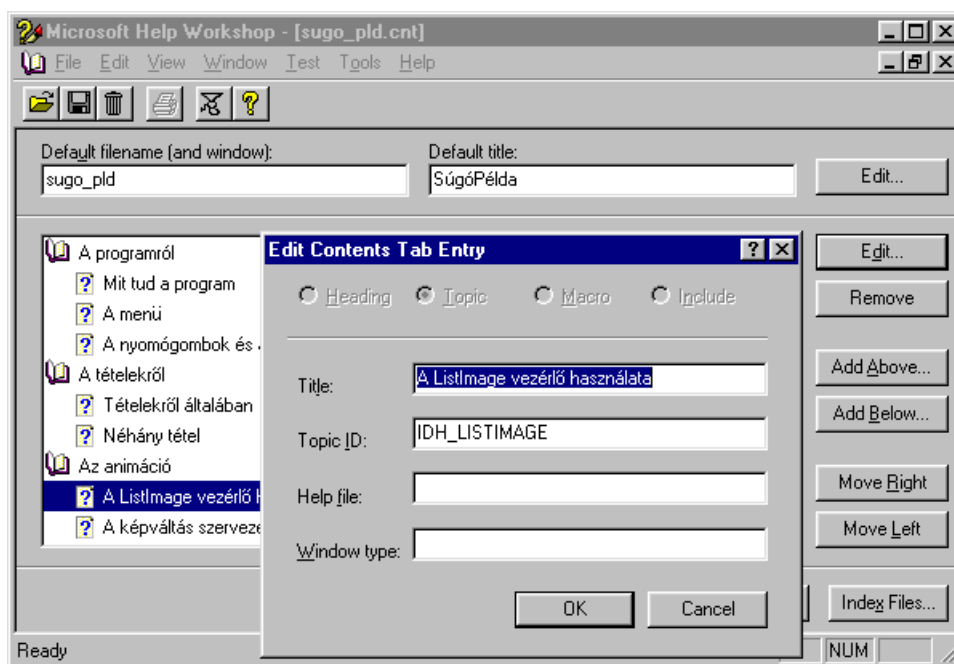
Ezek után a „*Save and Compile*” nyomógombra kattintva végezzük el a kész sűgóállomány fordítását!



Az alkalmazás harmadik, a *sugo\_pld.hlp* állomány készítését szintén tematikus fájl írásával kezdjük!

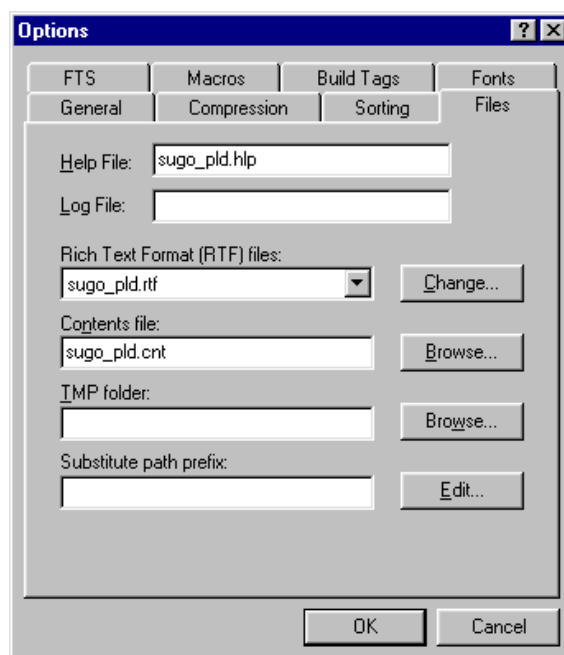


Az állomány *Help Workshop*-ban való fordítása előtt azonban létre kell hoznunk egy tartalomfájlt (.CNT) is. Ehhez válasszuk ki a *Help Workshop*-ban az új tartalomállomány készítését (**File | New | Help Contents**), és adjuk meg a sűgó hárompaneles párbeszédablakának első lapján megjelenő címsorokat, illetve az általuk hivatkozott sűgófejezeteket! (**Heading** – könyv alakú ikonnal jelzett témacsoport neve, **Topic** – hivatkozásként működő sűgófejezet-cím, amely a **Topic ID** mezőben megadott azonosítóval rendelkező sűgófejezetre mutat.)



Előtte azonban az **Edit** nyomógombon kattintva adjuk meg a súgófejezeteket tartalmazó súgófájl nevét, kiterjesztés nélkül (jelen példában ez a *sugo\_pld*)!

Miután elmentettünk a CNT állományt, az **Options** nyomógombon kattintva adjuk meg nevét a súgó projekt fájljához tartozó állományoknak!



Végezetül pedig fordítsuk le a súgófájlt (**Save and Compile**)!



Készítsünk animációs játékot, amelynek lényege a szaladgáló gombok „kilövése”! A játék állását jelezzük ki az állapotsorban, a *Névjegy* menün keresztül jelenítsük meg a szerzőkre vonatkozó információkat! (*Lövölde*)

Az alkalmazás játékrészének készítését ismerjük a multimédiával foglalkozó fejezetből! Ugyanott találunk információt a **StatusBar** vezérlőelem használatáról is.

A következőkben az alkalmazás *Névjegy* menüpontjának megvalósításáról lesz szó. A *Névjegy* menüpont a Windows alkalmazások „szabványos” kelléke, melynek megjelenítése egy külön form jelenletét igényli. Az új űrlap hozzáadásához válasszuk a Delphi **File** menüjéből a „**New form**” menüpontot. Megfelelő átméretezés után állítsuk be a form **BorderStyle** tulajdonságának értékét *bsDialog*-ra, és adjunk hozzá minden szükséges programleíró, szerzői jogokra vonatkozó stb. információt, használva a **Label**, az **Image**, az **Animation**, a **Button** és más vezérlőelemeket. Ezek közül a „legfontosabb” a párbeszédablakot lezáró **OK** nyomógomb:

```
procedure TForm2.btnOKClick(Sender: TObject);
begin
    close;
end;
```

Ha egy **Súgó** feliratú nyomógombra is lenne szükség, ezt is megvalósíthatjuk, megadva a gomb eseménykezelőjében az **Application** objektum **HelpCommand** metódusának megfelelően paraméterezett hívását.

A *Microsoft Word* névjegye megjelenítésekor egy WAV hangállomány is kerül lejátszásra („háttérzene”), amit vagy a **PlaySound** függvény, vagy pedig a **MediaPlayer** vezérlőelem segítségével végezhetjük el (bővebb információt lásd a multimédiáról szóló fejezetben).

A Delphi rendszer **About** párbeszédablaka pedig egy hivatkozást is tartalmaz a Borland cég webhelyére:

Ezt mi úgy tudjuk megvalósítani a legkisebb erőfeszítések árán, hogy miután megformáltuk a hivatkozás szövegét tartalmazó **Label** vezérlőelem **Font** tulajdonságának mezőértékeit (**Color**: *clBlue*, **Style**: *fsUnderline: true*), a feliratobjektum eseménykezelőjében megadjuk a következő hívást:

```
procedure TForm2.Label5Click(Sender: TObject);
begin
    WinExec(PChar('c:\program files\internet explorer\
        iexplore.exe ' + Label5.Caption), SW_SHOWNORMAL);
end;
```

A fentiekben látott **WinExec** hívással elindítjuk az *Internet Explorer* programot, paramétersorban átadva a betöltendő weboldal címét (amelyet a **Label** vezérlőelem **Caption** tulajdonsága határozza meg):

