

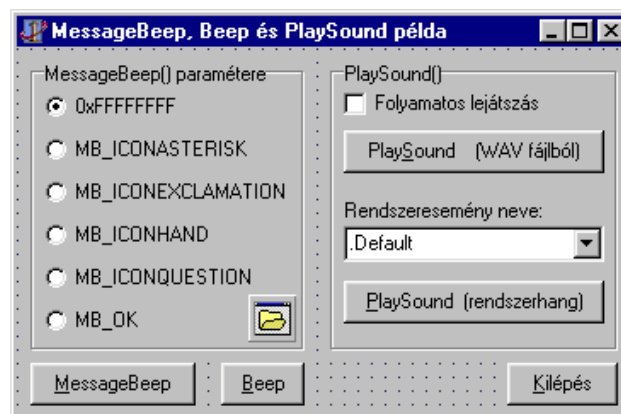
8.1. Multimédiás alkalmazások készítése

1. Alkalmazás a *MessageBeep*, *Beep* és *PlaySound* használatával [MessageBeep](#)
2. Az *mciExecute* hívása [MCIPLAY](#)
3. Az *Animation* vezérlőelem használata [Animation](#)
4. Animáció *Timer*-rel [AnimaTimer](#)
5. Kép mozgatás zenével [Panorama](#)
6. Mediafájl lejátszása *MediaPlayer*-rel [MFajlLejatszo](#)
7. CD lejátszó *MediaPlayer* vezérlővel [CDLejatszo](#)
8. Hangfelvétel *MediaPlayer* vezérlővel [Hangfelvetel](#)
9. Szönyegszöveg [Szonyeg](#)
10. Szaladgáló gombok „kilövése” [Lovolde](#)



Készítsünk alkalmazást a *MessageBeep*, a *Beep*, illetve a *PlaySound* függvény működésének bemutatására! (*MessageBeep*)

Készítsük el az alkalmazás felhasználói felületét! Helyezzünk a formra egy *RadioGroup* vezérlőelemet, 6 darab (*Items*) választógombbal (feliratuk a *MessageBeep* függvény lehetséges értékei), két nyomógombot a *MessageBeep* és a *Beep* függvény teszteléséhez, illetve egy további gombot a *PlaySound* függvény teszteléséhez! A „Rendszeresemény neve” *ComboBox* vezérlőelem *Items* tulajdonságát futási időben töltjük majd fel elemekkel, a képen látható *.Default* érték a kombinált lista statikusan beállított *Text* tulajdonságában tárolódik.



A *MessageBeep* feliratú nyomógomb alapértelmezés szerinti eseménykezelő eljárása:

```
procedure TForm1.btnMsgBeepClick(Sender: TObject);
var
  param: integer;
begin
  PlaySound(nil, 0, SND_PURGE); // hanglejátszás leállítása
  param:=0;                    // változó-inicializálás
  case MsgBeepParam.ItemIndex of // ha a bejelölt választógomb sorszáma egyenlő a(z)...
    0: param:=0;
    1: param:=MB_ICONASTERISK;
    2: param:=MB_ICONEXCLAMATION;
    3: param:=MB_ICONHAND;
    4: param:=MB_ICONQUESTION;
    5: param:=MB_OK;
  end;
  MessageBeep(param);          // hanglejátszás
end;
```

A fenti eljárásban először leállítottuk a rendszerben esetleg még végbemenő hanglejátszást, utána pedig attól függően, hogy melyik választógombot jelölte be a felhasználó, más és más paraméterértékkel hívjuk a *MessageBeep* függvényt.

A *Beep* feliratú nyomógomb működtetésekor a következő eljárás hajtódik végre:

```
procedure TForm1.btnBeepClick(Sender: TObject);
begin
  PlaySound(nil, 0, SND_PURGE); // hanglejátszás leállítása
  Beep;                         // hanglejátszás, ua., mint a MessageBeep(0);
end;
```

A *Beep* Delphi függvény működése megegyezik a *MessageBeep(0)* MCI függvény hívásával.

A „*PlaySound (WAV fájlból)*” nyomógomb funkciójának megvalósításához szükség lesz egy **OpenDialog** vezérlőelem (**Dialogs** palettalap) formon való elhelyezésére. Ebben az esetben az eseménykezelő eljárás következőképpen alakul:

```
procedure TForm1.btnPlayWAVSoundClick(Sender: TObject);
begin
    if OpenDialog1.Execute then           // ha kiválasztottunk egy fájlt
    begin
        PlaySound(nil,0,SND_PURGE);      // hanglejátszás leállítása
        if chkLoop.State = cbChecked then
            // ha be van jelölve a "Folyamatos lejátszás"
            PlaySound(PChar(OpenDialog1.FileName),0,SND_FILENAME or SND_ASYNC or SND_LOOP)
        else
            // ha nincs bejelölve a "Folyamatos lejátszás" jelölőnégyzet
            PlaySound(PChar(OpenDialog1.FileName),0,SND_FILENAME or SND_ASYNC);
        end;
    end;
end;
```

A .WAV fájl kiválasztása után a lejátszást végző **PlaySound** függvényt különbözőképpen hívjuk, attól függően, hogy a felhasználó kéri-e a fájl folyamatos lejátszását vagy sem. A lejátszás folyamatosságát a **PlaySound** harmadik paraméterében beállítható **SND_LOOP** érték jelzi. A **SND_FILE** arra utal, hogy a lejátszandó hangot egy fájlból (*OpenDialog1.FileName*) kell olvasnia a rendszernek, a **SND_ASYNC** pedig a lejátszás aszinkron voltát állítja be (ilyenkor a vezérlés rögtön a függvényhívás után visszatér a programhoz, ellentétben szinkron lejátszással, amikor meg kell várni a lejátszás végét).

A „*PlaySound (rendszerhang)*” nyomógomb alapértelmezett eseménykezelő eljárása nagyon egyszerű:

```
procedure TForm1.btnPlaySystemSoundClick(Sender: TObject);
begin
    PlaySound(nil, 0, SND_PURGE);          // hanglejátszás leállítása
    PlaySound(PChar(cmbSystemSound.Text), 0, SND_ALIAS or SND_ASYNC);    // hanglejátszás
end;
```

A fenti példában csak a **SND_ALIAS** paraméterérték az új, ami azt jelzi, hogy a lejátszandó hangot jelölő szöveg (mutató a *cmbSystemSound.Text* karakterláncra) megegyezik a **Registry**-ben, vagy a WIN.INI állományban megadott rendszeresemények ún. *alias* nevével.

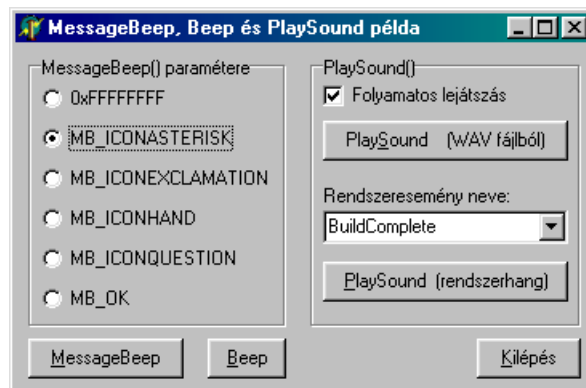
A program hanglejátszó része ezzel be is fejeződik. Mivel azonban az elején üresen hagytuk a *cmbSystemSound* kombinált listát – azzal, hogy a program maga „nézzen körül” a rendszerben – a kitöltéshez be kell nyúlnunk a **Registry**-be, és kiolvasni onnan az *AppEvents\EventLabels* kulcs alatti értékeket. A kitöltést természetesen az alkalmazás indításakor kell elvégeznünk, hogy a form megjelenítésekor az összes érték már a helyén legyen. Ehhez a *FormCreate* eljárásban – biztonság kedvéért - kitöröljük a **ComboBox** elemeit, utána pedig meghívunk egy saját magunk által írt eljárást:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    OpenDialog1.InitialDir:=GetCurrentDir;
    cmbSystemSound.Items.Clear;
    Read_Registry;
end;
```

A **Registry**-olvasó eljárást a következőképpen alakíthatjuk ki:

```
procedure TForm1.Read_Registry;
var
  Reg: TRegIniFile;
begin
  Reg:=TRegIniFile.Create;
  try
    Reg.RootKey:=HKEY_CURRENT_USER;
    if Reg.OpenKey('AppEvents\EventLabels', False) then
      Reg.ReadSections(cmbSystemSound.Items);
  finally
    Reg.Free;
  end;
end;
```

A **TRegIniFile** (*registry* modul) típusú objektum egyszerű felületet biztosít a **Registry** kezeléséhez. A **RootKey** tulajdonságában meg kell adni az ún. gyökerkulcsot (az aktuális keresés csak ez alatt a kulcs alatt folyhat), az **OpenKey** függvény beállítja az aktuális kulcsot (sikertelen próbálkozás esetén a gyökerkulcs lesz az aktuális kulcs), a **ReadSections** eljárással pedig a szekció összes alkulcsát beolvassuk a **ComboBox** vezérlőelemünkbe. Sikeres működés végén fel kell szabadítanunk a **Reg** objektum számára lefoglalt memóriát (**Free**).

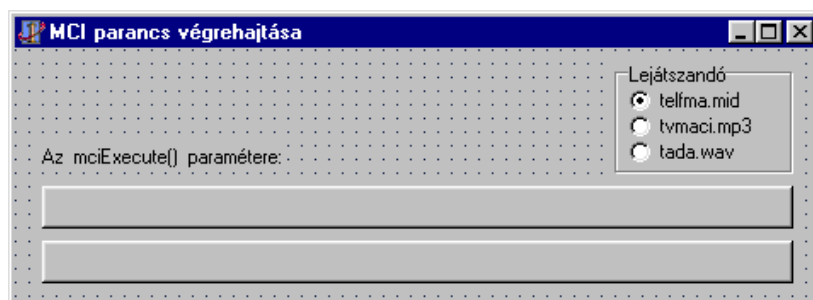




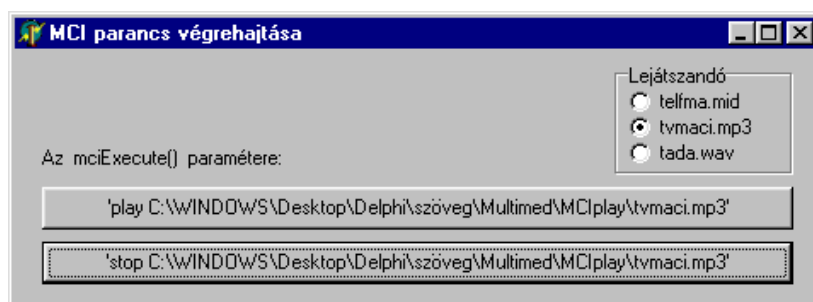
Az *mciExecute* függvény hívásához szükségünk van a *MMSystem* modulra:

```
uses MMSystem;
```

Az alkalmazás űrlapján elhelyezzük a következő ábrán látható elemeket (a *RadioGroup* vezérlőelem *Items* tulajdonságában megadott fájlnevek a program könyvtárába másolt hangállományokat jelölik):



A nyomógombok a feliratukat futási időben kapják meg, melynek szövege attól függ, melyik választógombot jelölte be a felhasználó. Például:



A gombok feliratozását a *RadioGroup* vezérlőelem *Click* eseménykezelőjében kell elvégezni:

```
procedure TMCIForm. RadioGroup1Click (Sender: TObject);
begin
    if lejatszاسMegy then Button2Click(Sender); // hanglejátszás leállítása

    // a lejátszandó zenefájl neve:
    fajlnev:=RadioGroup1.Items[RadioGroup1.ItemIndex];
    teljesnev:=GetCurrentDir+'\'+fajlnev;

    // a gombok feliratának létrehozása:
    Button1.Caption := 'play ' + teljesnev + ' ';
    Button2.Caption := 'stop ' + teljesnev + ' ';
end;
```

Az eseménykezelőben először megállítjuk az előzőleg elindított lejátszást azzal, hogy a *lejatszاسMegy* logikai változó *true* értéke esetén a programból „rákattintunk” a *stop ...* feliratú gombra (amelynek a *Caption* tulajdonsága az eljárás elején még az előbb elindított hangállomány adatait tartalmazza).

Az *mciExecute* az függvény paraméterként egy parancsnévből és a lejátszandó hangállomány nevéből összeállított sztringet vár. A *teljesnev* globális változóban tárolt teljes fájlnevet a példában a *GetCurrentDir* függvény segítségével lekérdezett aktuális könyvtárnevéből és a *RadioGroup* vezérlőelem bejelölt választógombjának feliratából fűzzük össze.

A *lejatszasaMegy* logikai változó az alkalmazás indításakor kap kezdőértéket:

```
procedure TMCIForm.FormCreate(Sender: TObject);
begin
    lejatszasaMegy := false;    // nincs elindított hanglejátszás
    RadioGroup1Click(Sender);  // nyomógombok feliratozása
end;
```

A *FormCreate* eljárás második sorában a **RadioGroup** már tárgyalt eseménykezelőjét hívjuk annak érdekében, hogy a form már feliratozott gombokkal jelenjen meg. Ehhez azonban a fejlesztés során be kell jelölnünk valamelyik választógombot, beállítva a vezérlőelem **ItemIndex** tulajdonságának értékét (például 0-ra, ami az első választógomb bejelölését jelenti).

A nyomógombok eseménykezelő eljárásai:

```
procedure TMCIForm.Button1Click(Sender: TObject);
begin
    parancs:='play '+teljesnev;
    mciExecute(PChar(parancs));    // a lejátszás elindítása
    ActiveControl := Button2;      // a fókusz átállítása
    lejatszasaMegy := true;
end;

procedure TMCIForm.Button2Click(Sender: TObject);
begin
    parancs:='stop '+teljesnev;
    mciExecute(PChar(parancs));    // a lejátszás megállítása}
    ActiveControl := Button1;      // a fókusz átállítása
    lejatszasaMegy := false;
end;
```

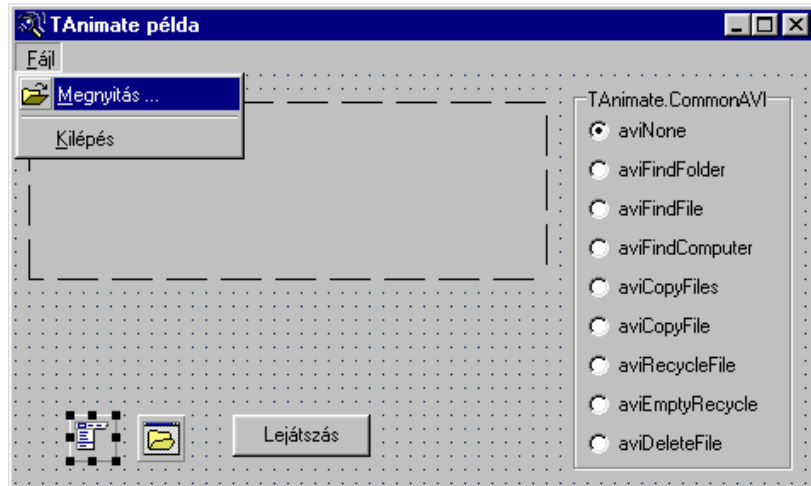
A hanglejátszást elindító alkalmazásokban fontos lépés a befejezésekor végrehajtott **FormClose** eljárás helyes programozása, ahol le kell állítanunk az elindított lejátszást. Ha ezt nem tesszük meg, a program lezárása után a gépünk továbbra is „zenélni” fog, ami roppant zavaró tud lenni, különösen hosszú hangállományok esetén:

```
procedure TMCIForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    parancs:='stop '+teljesnev;
    mciExecute(PChar(parancs));
end;
```



Készítsünk alkalmazást az **Animation** vezérlőelem használatára, amely a Windows rendszer szokványos, fájlműveleteket kísérő animációin túlmenően tetszőleges AVI fájl lejátszására is alkalmas! (*Animation*)

Helyezzük el az alkalmazás űrlapján az alábbi ábrán is látható vezérlőelemeket, azaz egy **Animate**, egy **MainMenu**, egy **OpenDialog**, egy **Button** és egy **RadioGroup** objektumot:



A **RadioGroup** vezérlőelem **Items** tulajdonságában meg kell adnunk az ábrán is látható választógomb-feliratokat, amelyek megfelelnek a **TAnimate.CommonAVI** tulajdonság lehetséges értékeinek:

```
type TCommonAVI = (aviNone, aviFindFolder, aviFindFile, aviFindComputer, aviCopyFiles,
                   aviCopyFile, aviRecycleFile, aviEmptyRecycle, aviDeleteFile);
property CommonAVI: TCommonAVI;
```

A fentiekben felsorolt tulajdonságértékek a Windows szokványos AVI állományait jelölik, amelyek a fájlműveleteket kísérő animációkat tartalmazzák.

A létrehozott **RadioGroup** vezérlőelem valamelyik választógombját be kell állítani (a **RadioGroup** vezérlőelem **ItemIndex** tulajdonságán keresztül, ahol 0 érték az első választógombot jelöli).

A **Fájl** menü **Megnyitás** menüpontja arra szolgál, hogy a felhasználó egy fájlnyitási párbeszédablakból maga válassza ki a lejátszandó AVI állományt:

```
procedure TForm1.mnuNyitasClick(Sender: TObject);
begin
    if OpenDialog1.Execute then
        Animate1.FileName := OpenDialog1.FileName;
end;
```

Ha már beállítottuk az **Animation** vezérlő **FileName** tulajdonságának értéket, az animációt a **Lejátszás** feliratú nyomógomb megnyomásával indíthatjuk el. A indítást programból az alábbi eseménykezelő eljárásban látható módon, illetve az **Animation** vezérlő **Play** metódusának segítségével egyaránt elvégezhethetjük:

```
procedure TForm1.btnLejatszasClick(Sender: TObject);
begin
    if (Animate1.CommonAVI <> aviNone) or (Animate1.FileName <> '') then
        Animate1.Active := true;    // lejátszás elindítása
end;
```

Ha nincs beállítva a **FileName** tulajdonság értéke, akkor az **Animation** vezérlőelem a **CommonAVI** tulajdonságában megadott szokványos animációt játssza le (*aviNone* érték „lenullázza” ezt a tulajdonságot). A megfelelő érték beállítása akkor történik, amikor a felhasználó bejelöl egy másik választógombot a **RadioGroup** vezérlőn:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
    Animatel.CommonAVI := TCommonAVI(RadioGroup1.ItemIndex);
    Animatel.Open := true;    // az első képkocka megjelenítése
end;
```



Ha a Delphi alapértelmezés szerinti beállításai (az AVI állomány összes képkockájának folyamatos lejátszása) valamilyen ok miatt nem felelnek meg az igényeinknek, akkor módosíthatjuk azokat. A lejátszás ismétlési számát az **Animation** vezérlőelem **Repetitions** tulajdonságában állíthatjuk át (0 érték folyamatos lejátszást jelent), a kezdő és az utolsó lejátszandó képkocka sorszámát pedig a **StartFrame**, illetve a **StopFrame** tulajdonságban adhatjuk meg.



Írjunk alkalmazást a „hagyományos” (az *Animation* és a *MediaPlayer* vezérlőelem nélküli), időzítővel megvalósított mozgató és képváltási animációk készítésére! Képváltás esetén használjuk az *ImageList* vezérlőelemet, az egyik szöveges vezérlőelem felírata pedig működjön forgószöveggént! (*AnimaTimer*)

Az alkalmazásunkban a következő animációkat valósítjuk meg: az *ImageList* vezérlőelemben tárolt képsorozat animálása (azaz képek folyamatos váltása) egy *Image* vezérlőelemben, egy *Shape* alakzat méretváltoztatása (növekedés - összezsugorodás), illetve mind az *Image*, mind pedig a *Shape* vezérlőelem vízszintes és függőleges mozgatása. Az egyik felirat szövege pedig forgószöveggént („futóreklámként”) jelenik meg. Az *Image* vezérlőelemben megjelenő képsorozatot, illetve a *Shape* vezérlő geometriai alakzatát menün keresztül változathatjuk.



Az alkalmazás űrlapján a következő vezérlőelemeket kell elhelyezni: egy *Image*, három-három *ImageList* és *Timer*, egy *MainMenu*, egy *Button* és egy *GroupBox* vezérlőelemet, benne három jelölőnégyzettel (*CheckBox*):

A három időzítő közül az első az *Image* objektumban megjelenő képek váltását, illetve a forgószöveg „futtatását” végzi, továbbá figyeli, hogy be kell-e kapcsolni az *Image* vezérlőelem mozgatásáért felelős második időzítőt:



```
procedure TForm1.Timer1Timer(Sender: TObject);
var
  s:string;
begin
  // A menüben pipával jelölt képsorozat (imlAktualis) következő
  // képkockájának megjelenítése:
  kepkocka:=(kepkocka+1) mod imlAktualis.Count;           // léptetés
  imlAktualis.GetBitmap(kepkocka, imgKep.Picture.Bitmap);  // kép beolvasása
  imgKep.Repaint;                                          // az Image objektum átrajzolása

  // Ha be van jelölve valamelyik a két jelölőnégyzet közül, bekapcsoljuk
  // a második időzítőt is, ellenkező esetben pedig kikapcsoljuk:
  Timer2.Enabled := chkHor.Checked or chkVer.Checked;

  // forgószöveg:
  if chkForgoSzoveg.Checked then // ha bejelöltük a jelölőnégyzetet
  begin
    // az első betű beillesztése a szöveg végére:
    s:=concat(chkForgoSzoveg.Caption,chkForgoSzoveg.Caption[1]);
    delete(s,1,1); // az első betű kivágása
    chkForgoSzoveg.Caption:=s;
  end;
end;
```

A fenti eljárásban használt globális változók közül a *kepkocka* (*Integer*) a *FormCreate* eljárásban kapja meg 0 kezdőértékét, az *imlAktualis* (*TImageList*) pedig akkor változik, amikor a felhasználó kiválaszt egy képet az alkalmazás *Kép* menüjéből.

Ahhoz, hogy az első időzítő működésbe lépjen, a felhasználónak ki kell választania egy képsorozatot a *Kép* menüből, és rá kell kattintania a *Start* gombra:

```
procedure TForm1.btnStartClick(Sender: TObject);
const
  felirat: array [0..1] of string = ('Start', 'Stop'); // a gomb lehetséges feliratai
  melyik: integer = 1; // a gomb aktuális felirata
begin
  // Az időzítők be-, illetve kikapcsolása:
  Timer1.Enabled:= not Timer1.Enabled;
  Timer2.Enabled:= (chkHor.Checked or chkVer.Checked) and (melyik=1);
  Timer3.Enabled:= not Timer3.Enabled;

  // A Start/Stop gomb feliratának átírása:
  btnStart.Caption:= felirat[melyik];
  melyik:=abs(melyik - 1);
end;
```

A fent látható eljárás akkor fog megfelelően működni, ha mind a három időzítő, mind pedig a *Start* nyomógomb **Enabled** tulajdonságát akár statikusan, akár a *FormCreate* eljárásban *false*-ra állítjuk. Ami az időzítők által generált órajel gyakoriságát illeti, első teszteléskor az **Interval** tulajdonság értékét *Timer1* vezérlőelem esetén 150-re, a *Timer2* és a *Timer3* esetén pedig 100-ra érdemes beállítani.

Az első időzítő által elindítható második időzítőnek az a feladata, hogy az **Image** objektum vízszintes, illetve függőleges mozgását végezze. Az időzítő a mozgatást a jelölőnégyzetek állásától függően végzi, növelve, illetve csökkentve bizonyos eltolási értékkel a vezérlőelem **Left**, illetve **Top** tulajdonságát. A mozgatás csak a form kliens területén belül történik, azaz a határok elérésekor az eltolási érték előjele mehgváltozik, így az objektum „visszafelé” mozog:

```
procedure TForm1.Timer2Timer(Sender: TObject);
const
  dX:integer=-2;
  dY:integer=2;
begin
  if chkHor.Checked then // vízszintes mozgás
  begin
    if (imgKep.Left < abs(dX)) or (imgKep.Left+imgKep.Width > Form1.ClientWidth-abs(dX))
    then dX:=-dX;
    imgKep.Left:=imgKep.Left+dX;
  end;

  if chkVer.Checked then // függőleges mozgás
  begin
    if (imgKep.Top < abs(dY)) or (imgKep.Top+imgKep.Height > Form1.ClientHeight-abs(dY))
    then dY:=-dY;
    imgKep.Top:=imgKep.Top+dY;
  end;
end;
```

Az utolsó időzítő a *Shape* objektum mozgatását, illetve folyamatos átméretezését végzi. A geometriai alakzatok mozgatási algoritmusai megegyeznek az *Image* vezérlőelem esetén alkalmazott megoldással, azzal a különbséggel, hogy az alakzat egy minimális érték és a form határa között változó méretét is figyelembe kell vennünk határelérés ellenőrzése során:

```

procedure TForm1.Timer3Timer(Sender: TObject);
const
    dR:integer = 2;
    dX:integer = 2;
    dY:integer = 1;
    minR:integer = 10;
var
    absDXR, absDYR: integer;
begin
    absDXR:= abs(dX)+abs(dR);
    absDYR:= abs(dY)+abs(dR);

    with Shape1 do
        begin
            if chkHor.Checked then           // vízszintes mozgás
                begin
                    if (Left < absDXR) or (Left+Width > Form1.ClientWidth-absDXR)
                        then dX:=-dX;
                    Left:=Left+dX;
                end;

            if chkVer.Checked then         // függőleges mozgás
                begin
                    if (Top < absDYR) or (Top+Height > Form1.ClientHeight-absDYR)
                        then dY:=-dY;
                    Top:=Top+dY;
                end;

            // átméretezés, határelérés ellenőrzésével:
            if (Left+Width>Form1.ClientWidth-absDXR) or (Top+Height > Form1.ClientHeight-absDYR) or
                (Height < minR) or (Width < minR)
                then dR:=-dR;
                Width:= Width+dR;
                Height:=Height+dR;
            end;
        end;
end;

```

Ahhoz, hogy a képeket és az alakzatokat menükön keresztül tudjuk váltani, szükségünk lesz a menüpontok *Click* eseményeit kezelő eljárásokra. Ezekben az eljárásokban az aktuálisan kiválasztott menüpont kap egy jelölő pipát, és ennek megfelelően megtörténik a megjelenítendő képsorozat, illetve alakzat váltása. Mivel azonban mind a *Kép*, mind az *Alakzat* menü esetén az általuk tartalmazott menüpontok kiválasztására lefutó eljárások csak a menüpontok sorszámában különböznek, menünként csak egy eljárást fogunk megírni (amelyet majd hozzárendelünk a menü összes menüpontjához), megfelelő módon lekérdezve benne az aktuális menüpont sorszámát.

Így a *Kép* menü mindegyik menüpontjának az *OnClick* eseménykezelője a következő lesz:

```

procedure TForm1.aktKepvaltasExecute(Sender: TObject);
var
    i:integer;
begin
    // a Kép menü összes jelölőpipájának kikapcsolása:
    for i:=0 to mnuKep.Count-1 do mnuKep.Items[i].Checked:= false;
    // jelölőpipa megjelenítése az aktuálisan kiválasztott menüpont előtt:
    (Sender as TMenuItem).Checked:= true;

    // az aktuális képsorozat beállítása
    imlAktualis:=kepek[(Sender as TMenuItem).MenuIndex]; kepkocka:=0;
    imlAktualis.GetBitmap(kepkocka, imgKep.Picture.Bitmap);
    imgKep.Repaint;

    btnStart.Enabled:=true; // a Start gomb működőképesé tétele
end;

```

Az *Alakzat* menü elemeihez pedig a következő eseménykezelő eljárást kell hozzárendelnünk:

```
procedure TForm1.AlakzatValtas(Sender: TObject);
var
  i: integer;
begin
  // az Alakzat menü összes jelölőpipájának kikapcsolása:
  for i:=0 to mnuAlakzat.Count-1 do mnuAlakzat.Items[i].Checked:= false;
  // jelölőpipa megjelenítése az aktuálisan kiválasztott menüpont előtt:
  (Sender as TMenuItem).Checked:= true;

  Shape1.Shape:=alakzatok[(Sender as TMenuItem).MenuIndex]; // alakzatváltás
  Shape1.Pen.Color:=random($00FFFFFF); // az alakzat színe
  Timer3.Enabled:=Timer1.Enabled; // ha megy a képváltás, változzon az alakzat is
end;
```

A fenti eljárás érdekessége, hogy az alakzat színét véletlenszerűen állítjuk be a *random* függvény segítségével. A véletlenszerű számok generátorát pedig az alkalmazás elindításakor lefutó eljárásban inicializáljuk a *randomize* eljárással:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  kepkocka:=0;
  randomize;
  Shape1.Pen.Color:=random($00FFFFFF);
  // az Alakzat menü egyik menüpontjának „kipipálása”
  mnuNegyszog.Checked:=true;

  // lehetséges képsorozatok beállítása:
  kepek[0]:=imlRozsa;
  kepek[1]:=imlKutyus;
  kepek[2]:=imlKiadojel;
end;
```

Az *AlakzatValtas* és a *FormCreate* eljárásban szereplő *alakzatok*, illetve *kepek* tömbazonosítóival már találkoztunk az előbb tárgyalt eljárásokban is. E két globális tömbazonosító közül az első egy olyan tömbkonstans, amely a *Shape* objektum *Shape* tulajdonságában beállítható értékeket tartalmazza, a második pedig egy tömbváltozó, amelynek elemei az *Image* vezérlőelem *Picture* tulajdonságában beállítható képsorozatokat jelölik:

```
implementation
const
  // kiválasztható geometriai alakzatok (Shape objektum):
  alakzatok: array [0..2] of TShapeType=(stRectangle, stCircle, stEllipse);
var
  // képsorozaton belül a következő megjelenítendő képkockát jelölő számláló:
  kepkocka: Integer;
  imlAktualis: TImageList; // aktuális képsorozat
  kepek: array [0..2] of TImageList; // lehetséges képsorozatok
```

Mivel a *kepek* tömböt a már létrejött form képlistáinak tárolására szánjuk, a kezdőértékeit a tömb csak az alkalmazás elindítása után kaphatja meg (lásd a *FormCreate* eljárást).

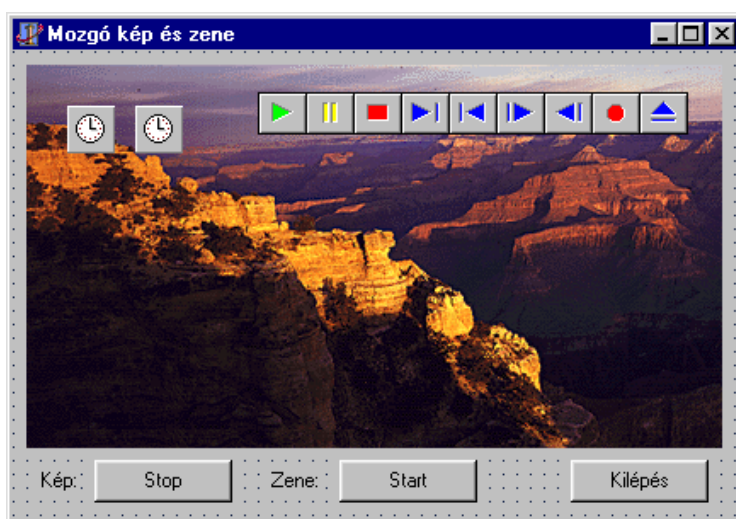


Készítsünk alkalmazást, amely egy BMP fájlal megadott kép mozgását végzi, zenei aláfestéssel! A kép mozgását, illetve a zenét külön-külön lehessen elindítani, illetve leállítani! (*Panorama*)

Nagy, panorámás képek animálásának érdekessége abban rejlik, hogy a kép méretei meghaladják a megjelenítési „keret”-objektum nagyságát, így a képet tartalmazó vezérlőelem (például *Image*) *Left* és *Top* tulajdonságainak értéke a kerethez képest negatívak lesznek.

A készülő alkalmazásunkban a keretobjektum egy *Panel* vezérlőelem, a képtároló pedig a panel fölé helyezett *Image* vezérlő. Az *Image* vezérlőelembe töltünk be - a *Picture* tulajdonságon keresztül – egy olyan képet (BMP formátumban), amelynek vízszintes mérete meghaladja a panel szélességét.

Zenei aláfestést a *PlaySound* MCI függvény segítségével is megvalósíthatjuk, azonban a megoldásban erre a célra egy *MediaPlayer* komponenst használtunk. A felsorolt vezérlőelemeken kívül szükségünk van még két időzítőre (*Timer*), illetve az ábrán is látható nyomógombokra (*Button*) és az előttük elhelyezkedő feliratokra (*Label*):



Állítsuk be az időzítők *Enabled* tulajdonságát *false*-ra, az *Interval* tulajdonság értékét pedig 56-ra, illetve 1000-re!

A kisebb *Interval*-értékkel rendelkező időzítőt a kép animálására használjuk. Ehhez az időzítő eseménykezelő eljárásában elvégezzük a képet tartalmazó *Image* objektum vízszintes eltolását (*dX*-szel növelve a *Left* tulajdonságának értékét):

```
procedure TPanoramaFrm.Timer1Timer(Sender: TObject);
const
  dX : integer=5;
begin
  if (Image1.Left > -abs(dx)) or (Image1.Left < Panel1.Width+abs(dx)-Image1.Width)
  then dX:=-dX;
  Image1.Left := Image1.Left + dX;
end;
```

A fenti eljárás első sorában azt teszteljük, hogy a képtároló objektum bal-, illetve jobb oldali határa kívül esik-e a keret szerepét játszó *Panel* vezérlő határain. Ha a kép „beszaladt” a keretbe, megváltoztatjuk a növekmény (*dX*) előjelét, ezzel a mozgás iránya is megváltozik.

Az időzítő bekapcsolását a *Kép*: felirattal megjelölt nyomógomb segítségével végezzük:

```
procedure TPanoramaFrm.StopGombClick(Sender: TObject);
begin
    if Timer1.Enabled                // ha működik az időzítő
    then StopGomb.Caption := szStart // feliratváltás
    else StopGomb.Caption := szStop;
    Timer1.Enabled := not Timer1.Enabled; // az időzítő be-, illetve kikapcsolása
end;
```

Az eljárásban használt *szStart* és *szStop* azonosítók a gomb lehetséges feliratait tartalmazó globális konstansok:

```
const
    szStart = 'Start'; // a gombok lehetséges feliratai
    szStop  = 'Stop';
```

Ugyanezeket a karakterláncokat használjuk a zene be- és kikapcsolását vezérlő nyomógomb feliratozásához is:

```
procedure TPanoramaFrm.ZeneGombClick(Sender: TObject);
begin
    if ZeneGomb.Caption = szStart then
        // kattintáskor a gomb „Start” felirattal rendelkezett
        begin
            MediaPlayer1.Open;           // médialejátszó megnyitása
            MediaPlayer1.Play;           // zenelejátszás elindítása
            ZeneGomb.Caption := szStop;  // feliratváltás
        end
    else
        begin
            MediaPlayer1.Stop;           // zenelejátszás megállítása
            MediaPlayer1.Close;          // médialejátszó lezárása
            ZeneGomb.Caption := szStart; // feliratváltás
        end
    end;
    Timer2.Enabled := not Timer2.Enabled; // a második időzítő be-, illetve kikapcsolása
end;
```

A *MediaPlayer* vezérlőelem *Play* metódusának segítségével elindíthatjuk, a *Stop* metódus hívásával pedig leállíthatjuk a médialejátszó *FileName* tulajdonságában beállított hangfájl lejátszását. A médiaeszközt lejátszás előtt meg kell nyitni (*Open* metódus), a lejátszás befejeztével pedig fel kell szabadítani a lefoglalt erőforrásokat (*Close* metódus). A hangfájl nevét fejlesztés alatt, illetve a program futáskor egyaránt megadhatjuk. Ha ezt a második módszert szeretnénk használni, ügyelnünk kell arra, hogy az állomány teljes elérési útvonalát kell megadnunk:

```
procedure TPanoramaFrm.FormCreate(Sender: TObject);
const
    szZeneFajl = 'Alafest.wav'; // a lejátszandó hangfájl neve
begin
    // a program aktuális könyvtára + a hangfájl neve:
    MediaPlayer1.FileName:=GetCurrentDir+'\'+szZeneFajl;
end;
```

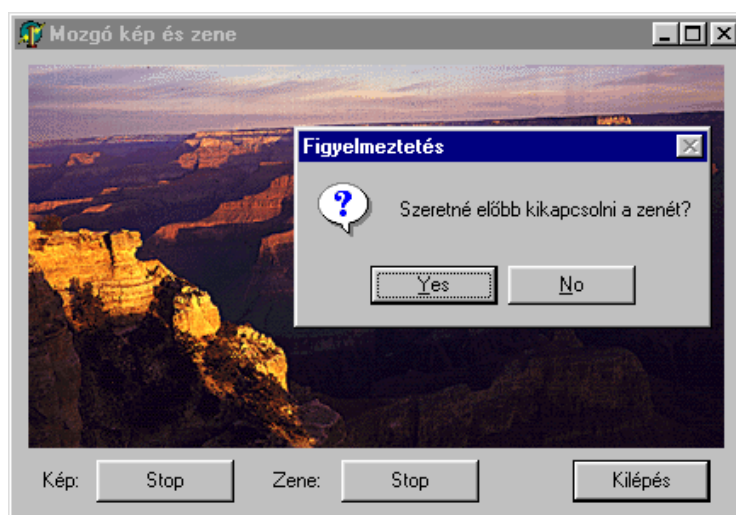
A *GetCurrentDir* függvény az alkalmazásunkat tartalmazó könyvtár nevével tér vissza, amely könyvtár a fenti példában a lejátszandó hangfájlt is tartalmazza.

A lejátszás folyamatossá tételéhez a nagyobb **Interval** értékkel rendelkező időzítőt használjuk, melynek be-, illetve kikapcsolása a hanglejátszással együtt, a *ZeneGombClick* eljárásban történik. Az időzítő eseménykezelőjében csak annyit teszünk, hogy teszteljük a hangállomány végének elérését (azaz egyenlő-e már a lejátszó **Position** tulajdonságában tárolt érték, a betöltött fájl méretét tároló **Length** tulajdonság értékével), és ebben az esetben visszatekerjük és újraindítjuk a lejátszót:

```
procedure TPanoramaFrm.Timer2Timer(Sender: TObject);
begin
    with MediaPlayer1 do
        if Position >= Length then // hangfájl vége
            begin
                Rewind; // visszatekerés
                Play; // hanglejátszás elindítása
            end;
end;
```

Ha a médialejátszónk **AutoRewind** (automatikus visszatekerés) tulajdonságát beállítjuk **true**-ra, akkor a fenti eljárásban látható **Rewind** metódus hívása feleslegessé válik.

A számítógép médiaeszközeit használó programok esetén nagyon fontos lépés az eszközök megfelelő és időben történő felszabadítása. Erre a legjobb módszer az, ha a *FormClose* eljárásban többek között meghívjuk a médialejátszó **Close** metódusát is. A jelen példában azonban a felhasználóra bízunk a médialejátszó lezárását azáltal, hogy az alkalmazás leállításakor – ha még szól a zene – felajánljuk neki, hogy előbb kikapcsolja a zenét:

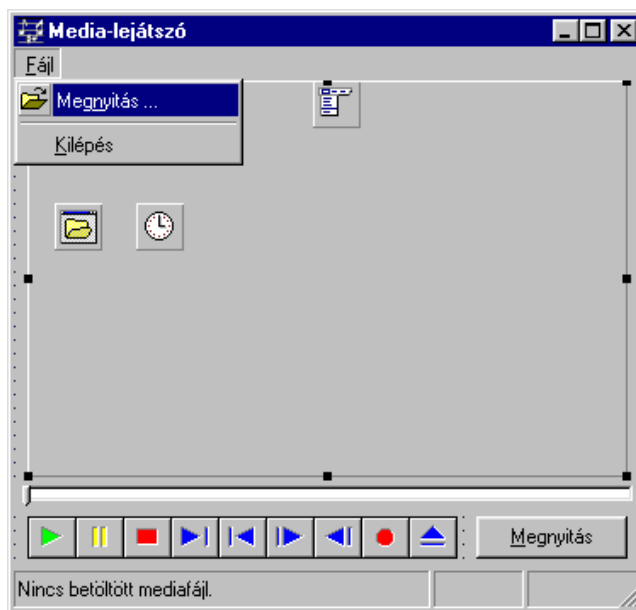


```
procedure TPanoramaFrm.KilepGombClick(Sender: TObject);
begin
    if ZeneGomb.Caption = szStop then
        if Application.MessageBox('Szeretné előbb kikapcsolni a zenét?',
            'Figyelmeztetés', MB_YESNO + MB_ICONQUESTION) = IDYES
        then MediaPlayer1.Close; // médialejátszó eszköz felszabadítása
    Close;
end;
```

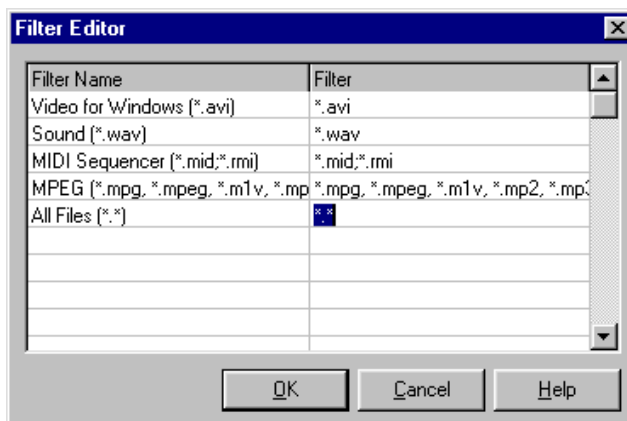



Készítsünk alkalmazást, amely médiafájlok lejátszását végzi a **MediaPlayer** vezérlőelem segítségével! (*MFajlLejatszo*)

A készülő alkalmazásunk űrlapján a **MediaPlayer** vezérlőelemen kívül helyezzük el a következő komponenseket is: egy főmenüt (**MainMenu**), amelyben egy *Megnyitás*, egy elválasztó vonal és egy *Kilépés* menüpontot alakítunk ki. A *Megnyitás* menüpont párjaként egy *Megnyitás* feliratú nyomógombot (**Button**), egy fájlnyitási párbeszédablakot (**OpenDialog**), egy időzítőt (**Timer**), az ablak aljára pedig egy állapotsort, vagyis egy **StatusBar** vezérlőelemet is felhelyezünk az űrlapra. A lejátszási folyamat szemléltetéséhez tegyünk a formra csúszkát (**TrackBar**) is. Mozgóképeket tároló állományok (AVI, MPEG-fájlok) tartalmának megjelenítéséhez érdemes kialakítani egy „képernyőt” is, például egy **Panel** vezérlőelem segítségével.



A fájlnyitási párbeszédablak (**OpenDialog**) **Filter** tulajdonságán keresztül adjuk meg a programunkkal lejátszható médiaformátumokat, vagyis a fájlnyitási párbeszédablak **Fájltípus** mezőjében megjelenő ún. szűrőneveket, illetve az általuk jelzett fájlnévmaszkokat (szűrőket):



A párbeszédablak megjelenítésekor a **Fájltípus:** mezőben látható szűrőnév sorszámát a **FilterIndex** tulajdonságon keresztül adhatjuk meg (a szűrők számozása 1-gyel kezdődik). Az **InitialDir** tulajdonság pedig a kezdőkönyvtár nevének megadására szolgál, amelynek tartalma megjelenik a megnyitott párbeszédablakban. Ha nem hozunk létre saját, médiafájlokat tartalmazó könyvtárat, akkor érdemes a tulajdonságot például a sok hangfájl tartalmazó Windows könyvtárra, a C:\WINDOWS\MEDIA-ra állítani. Ellenkező esetben, a párbeszédablakban a futó alkalmazást tartalmazó aktuális könyvtár lesz látható.

A lejátszással kapcsolatos információk megjelenítéséhez a form aljára helyezett állapotjelző sorban (**SatusBar**) alakítsuk ki a következő három panelt (**Panels[]**): egy hosszabbat a lejátszás alatt álló állomány nevének megjelenítéséhez (statikusan adjuk meg a panel kiinduló feliratát is, például „*Nincs betöltött médiafájl.*”) és két rövidebbet, üresen. Közülük az első a lejátszás elejétől eltelt időt, illetve - képkockás médiaformátumok, vagyis animációs és videófájlok esetén - a már bemutatott képkockák számát fogja jelezni. A második panelben pedig a lejátszás alatt álló médiaállomány teljes hossza (idő vagy képkockaszám formájában) jelenik meg.

Az utolsó statikus beállítás az időzítőre (**Timer**) vonatkozik. Az időzítőt mindenféleképpen kapcsoljuk ki az **Enabled** tulajdonság **false** értékre való állításával, mivel a működésére csak egy médiafájl betöltése után, pontosabban a lejátszás alatt lesz szükség. Az időzítőt arra fogjuk használni, hogy bizonyos időközönként frissítsük az állapotsor feliratát (eltelt idő vagy lejátszott képkockák száma), és változtassuk a pozíciómutató szerepét játszó csúszka (**TrackBar**) állását, illetve – ha elértük a médiafájl végét –, hogy leállítsuk az időzítőt és „visszatekerjük” a lejátszót a médiaállomány elejére:

```
procedure TMediaFrm.Timer1Timer(Sender: TObject);
var
    perc, mperc : Longint;
begin
    if TrackBar1.Position=TrackBar1.Max // lejátszás végének (csúszkaállásának) figyelése
    then
        begin
            felirat; // állapotsor frissítése (saját eljárás)
            Timer1.Enabled:=false; // az időzítő leállítása
            MediaPlayer1.Rewind; // a lejátszó visszatekerése
            TrackBar1.Position:=0; // a csúszka visszaállítása
        end
    else
        case MediaPlayer1.TimeFormat of
            // pozíciót jelző tulajdonságok értékei ezredmásodpercekben értendők
            tfMilliseconds: begin
                TrackBar1.Position := MediaPlayer1.Position; // a csúszka átállítása
                mperc:=MediaPlayer1.Position div 1000;
                perc:=mperc div 60;
                // az eltelt időt jelző panel szövegének a frissítése:
                AllapotSor.Panels.Items[1].Text:=Format('%2d:%2d:%2d', -
                    [(perc div 60), (perc mod 60), (mperc mod 60)]);
            end;
            tfFrames: // pozíciót jelző tulajdonságok értékei képkocka-számot tartalmaznak
            begin
                TrackBar1.Position := MediaPlayer1.Position;
                AllapotSor.Panels.Items[1].Text:=IntToStr(TrackBar1.Position) + ' keret';
            end;
            tfHMS:
                MediaPlayer1.TimeFormat:=tfMilliseconds
            else
                begin
                    TrackBar1.Position := MediaPlayer1.Position;
                    AllapotSor.Panels.Items[1].Text:=IntToStr(MediaPlayer1.Position);
                end;
            end;
        end;
    end;
```

Az időzítő eseménykezelő eljárásában - abban az esetben, ha befejeződött a lejátszás, és ezzel a pozíciómutató csúszka végállásba került (az aktuális pozíciót jelző **Position** tulajdonság értéke egyenlő a **Max** tulajdonságban megadott végértékkel) -, kikapcsoljuk az időzítőt, visszatekerjük a lejátszót a **Rewind** metódus segítségével, és átállítjuk a csúszkát a kezdő, 0-ás pozícióra.

Ha azonban még tart a lejátszás, el kell végeznünk az állapotsor egyik paneljének frissítését. A panelben megjelenő szöveg – hányadik percnél és másodpercnél, illetve hányadik képkockánál tart a lejátszás – formázása eltérő módon történik attól függően, milyen érték van eltárolva a médialejátszó **TimeFormat** tulajdonságában:

```

type TmpTimeFormats = (tfMilliseconds, tfHMS, tfMSF, tfFrames, tfSMPTE24,
                        tfSMPTE25, tfSMPTE30, tfSMPTE30Drop, tfBytes, tfSamples, tfTMSF);

property TimeFormat: TmpTimeFormats;

```

Itt ugyanis az jelenti a legnagyobb gondot, hogy a pozíciót jelző tulajdonságok értékének formátuma médiafüggő, azaz különböző médiaformátumok esetén eltérő módon kell értelmezni a médialejátszó **Position**, **Length**, **StartPos**, **Start**, és **EndPos** tulajdonságain keresztül elérhető 4 bájtos adatot. Például ha ún. keretes (*tfFrames*), azaz képek sorozatának tárolására alkalmas formátumról van szó, akkor a felsorolt tulajdonságok 4 bájta szám, a **Position** pedig az aktuális képkocka sorszámát jelöli. Vannak azonban olyan formátumok is, ahol a négy bájtt külön-külön órákat, percek, másodpercek, illetve valamilyen más értéket (például hangsávszámot, keretszámot, vagy „semmit”, ha nem használják a bájtot) tárol.

A jelen programban az időzítő eseménykezelőjében a médiafájlok esetén csak a két leggyakrabban előforduló – és egyben a legegyszerűbb - esetet dolgozzuk fel: a *tfMilliseconds* és a *tfFrames* értékeket. Mindkét esetben a kapott pozícióérték szám, és ezredmásodpercek, illetve képkocka-számot tartalmaz. Így a csúszkát – feltéve, hogy a lejátszás előtt átállítottuk a **Max** tulajdonságának értékét a médialejátszóba betöltött állomány hosszát jelző **Length** értékre – mindenféle átszámítás nélkül egyenesen az aktuális **Position** értékre állíthatjuk. A **TimeFormat** *tfMilliseconds* értéke esetén – mivel az időt az *XX:YY:ZZ* formában szeretnénk kiírni (a Delphi *Sysutils* modulja **Format** függvénye segítségével), ki kell számítani, hogy az ezredmásodpercekben kapott érték hány órát, percet és másodpercet jelent:

```

...
mperc:=MediaPlayer1.Position div 1000;
perc:=mperc div 60;
// az eltelt időt jelző panel szövegének a frissítése:
AllapotSor.Panels.Items[1].Text:=Format('%2d:%2d:%2d', -
                                         [(perc div 60), (perc mod 60), (mperc mod 60)]);
...

```

Abban az esetben, ha a **TimeFormat** értéke *tfHMS* (azaz a legalacsonyabb helyiértékű bájttól kezdve következő a bájtok felosztása: órák, percek, másodpercek, nem használt bájtt), amely formátum használata gyakori a MIDI állományoknál, a jelen programban egyszerűség kedvéért átállítjuk a **TimeFormat** értékét *tfMilliseconds*-ra. Hosszabb médiafájlok lejátszása esetén azonban érdemes az eredeti bájttformátum „szétválogatásánál” maradni, amit például az *shl*, *shr* operátorok segítségével, vagy egy megfelelően kialakított 4-bájtos rekordtípus (**record**) felhasználásával végezhetünk el. Ugyanez vonatkozik a többi, a példaprogramunkban nem kezelt formátumra is, amelyek bájttjainak leírását megtaláljuk a Delphi súgójában a **TMediaPlayer.TimeFormat** címszó alatt.

Az időzítő eseménykezelőjében hívott *felirat* eljárás az alkalmazásunk **private** metódusa, amely az állapot-jelzősor utolsó, azaz a betöltött médiaállomány hosszát jelző panel feliratának beállítását végzi:

```

procedure TMediaFrm.felirat();
var
  perc, mperc : Longint;
begin
  TrackBar1.Max := MediaPlayer1.Length; // a síkcsuszka végállási értékének a megadása
  case MediaPlayer1.TimeFormat of
    tfMilliseconds:
      begin
        mperc:=MediaPlayer1.Length div 1000;
        perc:=mperc div 60;
        // a médiafájl hosszának kiírása:
        AllapotSor.Panels.Items[2].Text:=Format('%2d:%2d:%2d', -
                                                  [(perc div 60), (perc mod 60), (mperc mod 60)]);
      end;
    tfFrames: // képek
      AllapotSor.Panels.Items[2].Text:=IntToStr(MediaPlayer1.Length) + ' keret';
    tfHMS:
      MediaPlayer1.TimeFormat:=tfMilliseconds
    else
      AllapotSor.Panels.Items[2].Text:=IntToStr(MediaPlayer1.Length);
  end;
end;

```

A *felirat* eljárásban a csúszka végértékét (**Max**) is a médiafájl hosszának megfelelő értékre állítjuk. Ezt az eljárást akkor is meg kell hívunk, amikor a felhasználó a médialejátszó valamelyik gombjára kattint:

```

procedure TMediaFrm.MediaPlayer1Click(Sender: TObject; Button: TMPBtnType;
  var DoDefault: Boolean);
begin
  felirat; // az állapotsor frissítése

  case Button of
    btPlay: // ha a Play gombra kattintottunk
      begin
        // az időmutató frissítésének gyakorisága:
        Timer1.Interval:=(TrackBar1.Max div TrackBar1.Width)+1;
        Timer1.Enabled:=true; // az időmutató bekapcsolása
        with MediaPlayer1 do
          begin
            Display:=Panel1; // képernyőterület beállítása
            DisplayRect:=Rect(2,2,0,0);
            if (DisplayRect.Bottom-DisplayRect.Top)>(Panel1.Height-4)
              then DisplayRect:=Rect(2,2,Panel1.Width*((Panel1.Height-4) div
                (DisplayRect.Bottom-DisplayRect.Top)),Panel1.Height-2)
            else
              if (DisplayRect.Right-DisplayRect.Left)>(Panel1.Width-4)
                then DisplayRect:=Rect(2,2,Panel1.Width-2,Panel1.Height*((Panel1.Width-4)
                  div (DisplayRect.Right-DisplayRect.Left)));
            Panel1.Repaint;
          end;
        end;
        btStop:
          if TrackBar1.Position=0 then MediaPlayer1.Position:=0;
        end;
      end;

```

A fenti eseménykezelőben az állapotjelző sor átírását követően attól függően, hogy a felhasználó a **Play** vagy a **Stop** gombra kattintott, elvégezzük a lejátszás elindítását, illetve megállítást kísérő beállításokat. A **Stop** gomb lenyomása esetén csak akkor tekerjük vissza a lejátszót, ha a lejátszás már befejeződött.

A **Play** esetén egyrészt beállítjuk a felirat frissítéséhez használt időzítő **Interval** tulajdonságát, és bekapcsoljuk az időzítőt. Ezek után pedig a médialejátszó **Display** tulajdonságán keresztül megadjuk a mozgóképek megjelenítéséhez szükséges „képernyőobjektumot” (a példában *Panel1*), illetve az azon belül képek megjelenítésére használt terület méreteit a **Rect** függvény segítségével:

Alkalmazásunkban ezt a függvény a 2, 2, 0, 0 paraméterértékekkel hívjuk, ezzel beállítva a kép megjelenítését a (2, 2) koordinátával rendelkező bal felső saroktól kezdve a képkocka alapértelmezés szerinti méretében (amit az utolsó két 0 érték jelez). Ha azonban a képkockák valamelyik mérete nagyobb a megjelenítésükre alkalmazott **Panel** vezérlőelem megfelelő méreteinél, arányosan átállítjuk a képkocka megjelenítési méretét (átméretezzük) legfeljebb a (2, 2, *Panel1.Width-2*, *Panel1.Height-2*) területre. Ezek után pedig megjelenítjük a betöltött médiaállomány első képkockáját a **Panel** vezérlőelem **Repaint** metódusa segítségével.



Ugyanezekre a műveletekre akkor is szükség van, ha a *Megnyitás* feliratú menüpont, illetve az azonos nevű nyomógomb segítségével megnyitunk egy médiaállományt:

```
procedure TMediaFrm.btnMegnyitasClick(Sender: TObject);
var
    ext, hibaStr: string;
begin
    // ha kiválasztottunk egy fájlt az Open párbeszédablakban
    if OpenFileDialog1.Execute then
        begin
            if MediaPlayer1.DeviceID<>0 then // ha van megnyitott médiaeszköz
                begin
                    MediaPlayer1.Stop; // lejátszás leállítása
                    MediaPlayer1.Close;
                end;

            ext:= AnsiUpperCase(ExtractFileExt(OpenDialog1.FileName));

            // csak az Open párbeszédablakra megadott szűrőinek megfelelő médiafájlokkal
            // dolgozunk tovább:
            if not (StrPos(StrLower(PChar(OpenDialog1.Filter)), StrLower(PChar(ext))) = nil)
            then
                with MediaPlayer1 do
                    begin
                        if (ext='.MID') or (ext='.RMI') then // MIDI
                            begin
                                TimeFormat:=tfMilliseconds;
                                DeviceType:=dtSequencer;
                            end
                        else DeviceType:=dtAutoSelect; // automatikus csatornatípus-kiválasztás

                        FileName:= OpenDialog1.FileName;
                        try
                            Open; // a csatorna megnyitása
                        except
                            hibaStr := 'Hibakód: ' + IntToStr(Error) + #13#10;
                            MessageDlg(hibaStr + ErrorMessage, mtError, [mbOk], 0);
                        end;

                        Display:=Panell1; // képernyőterület beállítása
                        DisplayRect:=Rect(2,2,0,0);
                        if (DisplayRect.Bottom-DisplayRect.Top)>(Panell1.Height-4) then
                            DisplayRect:=Rect(2, 2, Panell1.Width*((Panell1.Height-4) div
                                (DisplayRect.Bottom-DisplayRect.Top)),Panell1.Height-2)
                        else
                            if (DisplayRect.Right-DisplayRect.Left)>(Panell1.Width-4) then
                                DisplayRect:=Rect(2, 2, Panell1.Width-2,
                                    Panell1.Height*((Panell1.Width-4) div (DisplayRect.Right-DisplayRect.Left)));
                                Panell1.Repaint;
                                // az állapotsor felirata:
                                AllapotSor.Panels.Items[0].Text := OpenDialog1.FileName + ' lejátszása';
                            end
                        else // ha más kiterjesztésű fájlt választottunk ki:
                            begin
                                AllapotSor.Panels.Items[0].Text := 'Nincs betöltött médiafájl.';
                                AllapotSor.Panels.Items[1].Text := '';
                                AllapotSor.Panels.Items[2].Text := '';
                            end;
                    end;
                end;
            end;
        end;
end;
```

A fenti eseménykezelőben, a médiaállomány párbeszédablakból való kiválasztása után megvizsgáljuk, befejeződött-e már az előző lejátszás. Ha nem, leállítjuk azt és felszabadítjuk a médiaeszközt. Ezek után megvizsgáljuk az *ExtractFileExt* függvény segítségével lekérdezett médiafájl-kiterjesztést. Ha a fájlnyitási párbeszédablak *Filter* tulajdonságában megadott karakterlánc tartalmazza a beolvasott kiterjesztésnek megfelelő szűrőt, megkezdődhet a médiafájl megnyitása a médialejátszóban. Ellenkező esetben csak az állapotjelző sor panelfeliratait állítjuk át a kiinduló értékekre ('Nincs betöltött médiafájl.', '', '').

A médiaállomány megnyitása előtt be kell állítani a médiaeszköz típusát jelző **DeviceType** tulajdonság értékét. A **DeviceType** tulajdonság *dtAutoSelect* értékre való beállításával a médialejátszóra bizzuk a megfelelő médiaeszköz felderítését, bizonyos esetekben azonban nem kerülhetjük el „kézi” beállítást, különben a program hibázni fog. Az alkalmazásunk által lejátszható médiaformátumok közül a MIDI formátumú fájlok követelik az ilyen speciális bánásmódot (*dtSequencer*), a többivel nincs gond.

Ezek után be kell állítani a médialejátszó **FileName** tulajdonságának értékét az **OpenDialog** által visszaadott állománynévre, és megnyitni a médiaeszközt. Ha a megnyitás nem sikerült, a hiba kódját és leírását a **MediaPlayer** vezérlőelem **Error**, illetve **ErrorMessage** tulajdonságaiba írt értékek segítségével jeleníthetjük meg, egy figyelmeztető párbeszédablakban. Ahhoz pedig, hogy a mozgóképeket tartalmazó fájlok esetén még a lejátszás elindítása előtt megjelenjen az első képkocka a **Panel** vezérlőelemen, el kell végeznünk a korábbiakban már tárgyalt képernyőbeállításokat.

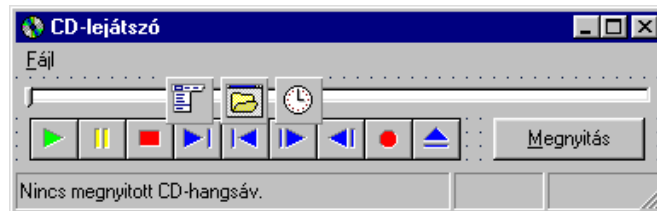
Mivel az alkalmazásunk futása során rendszereszközöket foglalunk le, az alkalmazás befejezése előtt gondoskodnunk kell azoknak felszabadításáról:

```
procedure TMediaFrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if MediaPlayer1.DeviceID<>0 then
  begin
    MediaPlayer1.Stop;
    MediaPlayer1.Close;
  end;
end;
```

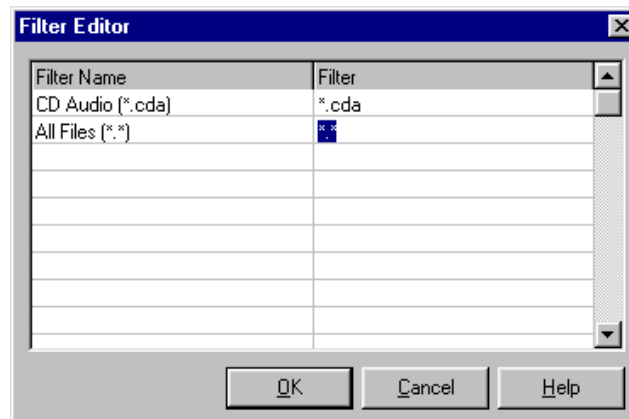


Készítsünk alkalmazást, amely bemutatja a **MediaPlayer** vezérlőelem CD-lejátszási lehetőségeit! (CDLejatszo)

A számítógépünk CD- vagy DVD-meghajtójába tett zenei CD lejátszására alkalmas program írását kezdjük el azzal, hogy a formon elhelyezzük a következő vezérlőelemeket: egy médialejátszót (**MediaPlayer**), egy főmenüt (**MainMenu**), egy *Megnyitás* feliratú nyomógombot (**Button**), egy fájlnyitási párbeszédablakot (**OpenDialog**), egy időzítőt (**Timer**), egy csúszkát (**TrackBar**) és egy állapotjelző sort (**StatusBar**).



Az állapotjelző sorban alakítsunk ki három panelt: az első, hosszabb panel felirata legyen '*Nincs megnyitott CD-hangsáv.*', a másik kettő ne kapjon statikus feliratot. A főmenüben hozzunk létre egy *Megnyitás*, egy elválasztó vonal és egy *Kilépés* menüpontot. Az **OpenDialog** vezérlőelem **Filter** tulajdonságán keresztül adjuk meg a következő ábrán is látható szűrőneveket és fájlnévmaszkokat (ún. szűrőket), a **FilterIndex** tulajdonság értékét pedig állítsuk be 1-re.



Az időzítőt (**Timer**) kapcsoljuk ki, beállítva az **Enabled** tulajdonságában a **false** értéket. Az időzítő eseménykezelőjében pedig végezzük el a csúszka pozíciójának, illetve az állapotjelző sor középső panelszövegének átállítását a lejátszás pozíciójának megfelelően:

```
procedure TMediaFrm.Timer1Timer(Sender: TObject);
var
    perc, mperc : Longint;
begin
    case MediaPlayer1.TimeFormat of
        tfTMSF: // CD
            begin
                perc:= (MediaPlayer1.Position shl 16) shr 24;
                mperc:= (MediaPlayer1.Position shl 8) shr 24;
                TrackBar1.Position:= mperc + perc*60;
                AllapotSor.Panels.Items[1].Text:=Format('%2d:%2d', [perc,mperc]);
                if (perc=0) and (mperc<1) then felirat;
            end;
        else
            begin
                TrackBar1.Position := MediaPlayer1.Position;
                AllapotSor.Panels.Items[1].Text:=IntToStr(MediaPlayer1.Position);
            end;
    end;
    if AllapotSor.Panels.Items[1].Text = AllapotSor.Panels.Items[2].Text
    then TrackBar1.Position:=0;
end;
```

A CD-n levő hangsávokon belüli pozíciókat jelző **MediaPlayer**-tulajdonságok időformátuma a **TimeFormat** tulajdonság *dtTMSF* értékének felel meg. A formátum szerint a pozíciót jelző 4 bájt közül a legalacsonyabb helyiértékű bájtban az aktuális hangsáv sorszámát, a következőben percek, a harmadikban másodperceket, a legmagasabb helyiértékű bájtban pedig keretszámot (*frames*) kapjuk. Így az idő kijelzéséhez szükségünk a megfelelő értékeket ki kell számítanunk:

```
perc:= (MediaPlayer1.Position shl 16) shr 24;
mperc:= (MediaPlayer1.Position shl 8) shr 24;
...
AllapotSor.Panels.Items[1].Text:=Format('%2d:%2d', [perc,mperc]);
```

A csúszka pozícióját másodpercekben adjuk meg:

```
TrackBar1.Position:= mperc + perc*60;
```

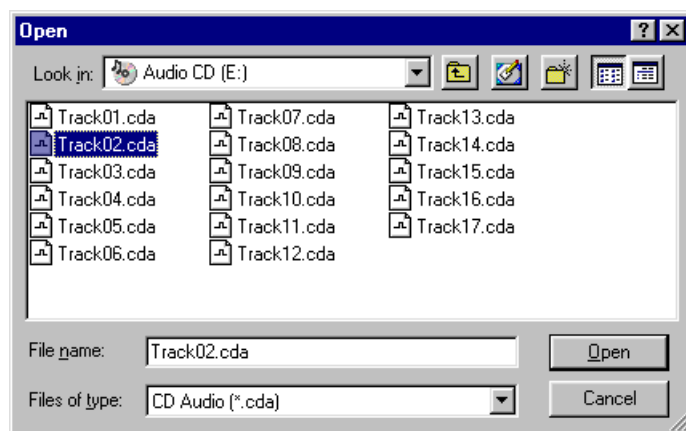
Az időzítő eseménykezelő eljárásában azt is figyeljük, elértük-e már az aktuális hangsáv végét, és ha igen, visszaállítjuk a csúszka pozícióját 0-ra:

```
if AllapotSor.Panels.Items[1].Text = AllapotSor.Panels.Items[2].Text
then TrackBar1.Position:=0;
```

Az időzítő a *Megnyitás* nyomógombhoz, illetve a *Megnyitás* menüponthoz rendelt eseménykezelő eljárásban indítjuk el:

```
procedure TMediaFrm.btnMegnyitasClick(Sender: TObject);
var
  ext, hibaStr: string;
  sav: longint;
begin
  if OpenFileDialog1.Execute then // ha kiválasztottunk egy fájlt
  begin
    if MediaPlayer1.DeviceID<>0 then // ha van megnyitott médiaeszköz
    begin
      MediaPlayer1.Stop; // lejátsszás leállítása
      MediaPlayer1.Close;
    end;
    // az állapotsor felirata:
    AllapotSor.Panels.Items[0].Text := OpenFileDialog1.FileName + ' lejátszása';
    ext:= AnsiUpperCase(ExtractFileExt(OpenDialog1.FileName));
    if ext='.CDA' // a fájl - egy hangsáv ("zeneszám") a zenei CD-n
    then
    begin
      MediaPlayer1.DeviceType := dtCDAudio; //a lejátszó típusának beállítása
      try
        MediaPlayer1.Open; //a csatorna nyitása
      except
        hibaStr := 'Hibakód: ' + IntToStr(Error) + #13#10;
        MessageDlg(hibaStr + MediaPlayer1.ErrorMessage, mtError, [mbOk], 0);
      end;
      // a kiválasztott hangsáv sorszáma:
      sav:= StrToInt(Copy(OpenDialog1.FileName,
        Length(OpenDialog1.FileName)-5, 2));
      MediaPlayer1.Position := MediaPlayer1.TrackPosition[sav]; // pozicionálás
      AllapotSor.Panels.Items[0].Text:= 'Zenei CD lejátszása, ' +
        IntToStr(sav) + '. sáv'; // feliratozás
    end
  else // ha nem egy CD-n levő hangsávot választottunk ki a párbeszédablakból
  begin
    AllapotSor.Panels.Items[0].Text := 'Nincs megnyitott CD-hangsáv.';
    AllapotSor.Panels.Items[1].Text := '';
    AllapotSor.Panels.Items[2].Text := '';
  end;
end;
end;
```


Mivel a CD-n levő hangsávok nem fájlok, a jelen alkalmazásban nincs értelme annak, hogy beállítsuk az **OpenDialog** által visszaadott **FileName** tulajdonság értékét, a **MediaPlayer** objektum **FileName** tulajdonságában. Ehelyett a médiaeszköz nyitása előtt először be kell állítani a lejátszó **DeviceType** tulajdonságának értékét *dtCDAudio*-ra, és a megnyitás után el kell végezni a lejátszó pozicionálását arra a hangsávra, amelyet a felhasználó kiválasztott a párbeszédablakból:



A pozicionáláshoz először kimásoljuk az **OpenDialog** által visszaadott karakterláncból a hangsáv sorszámát tartalmazó két karaktert. A karaktereket számmá alakítva beállíthatjuk a kívánt sávpozíciót a **TrackPosition** tulajdonság segítségével:

```
sav:= StrToInt(Copy(OpenDialog1.FileName, Length(OpenDialog1.FileName)-5, 2));
MediaPlayer1.Position := MediaPlayer1.TrackPosition[sav]; // pozicionálás
AllapotSor.Panels.Items[0].Text:= 'Zenei CD lejátszása, ' + IntToStr(sav) + '. sáv';
```



Amennyiben az **OpenDialog** vezérlőelem **FileName** tulajdonságában nem egy CDA kiterjesztésű karakterláncot kapunk, semmi sem történik, csak az állapotsor feliratai jelzik a kiválasztott hangsáv hiányát.

A *felirat* **private** metódust mindegyik hangsáv esetén csak a lejátszás első másodpercében hívjuk, hogy egyrészt beállítsuk a csúszka **Max** tulajdonságának értékét az aktuális hangsáv hosszára, másrészt pedig, hogy aktualizáljuk a hangsáv számát, illetve hosszát kijelző állapotsor-panelet:

```
procedure TMediaFrm.felirat();
var
  sav, perc, mperc : Longint;
begin
  case MediaPlayer1.TimeFormat of
    tftMSF: // CD
      begin
        sav:= (MediaPlayer1.Position shl 24) shr 24;
        perc:= (MediaPlayer1.TrackLength[sav] shl 24) shr 24;
        mperc:= (MediaPlayer1.TrackLength[sav] shl 16) shr 24;
        TrackBar1.Max := mperc + perc*60; // sínscuszka végértékének megadása
        AllapotSor.Panels.Items[0].Text:= 'Zenei CD lejátszása, '+IntToStr(sav)+' . sáv';
        AllapotSor.Panels.Items[2].Text:=Format('%2d:%2d', [perc,mperc]); // időhossz
      end;
    else
      begin
        TrackBar1.Max := MediaPlayer1.Length; // sínscuszka végértékének megadása
        AllapotSor.Panels.Items[2].Text:=IntToStr(MediaPlayer1.Length);
      end;
  end;
end;
```


Az időzítő eseménykezelő eljárásának hívásgyakoriságát (***Interval***) akkor állítjuk be, amikor a felhasználó rákattint a médialejátszó ***Play*** gombjára, ezután pedig be is kapcsoljuk az időzítőt:

```
procedure TMediaFrm.MediaPlayer1Click(Sender: TObject; Button: TMPBtnType;
  var DoDefault: Boolean);
begin
  case Button of
    btPlay: // ha a Play gombra kattintottunk
      begin
        Timer1.Interval:=(TrackBar1.Max div TrackBar1.Width)+1;
        Timer1.Enabled:=true; // az időzítő belapcsolása
        felirat(); // állapotsor feliratozása
      end;
    end;
end;
```

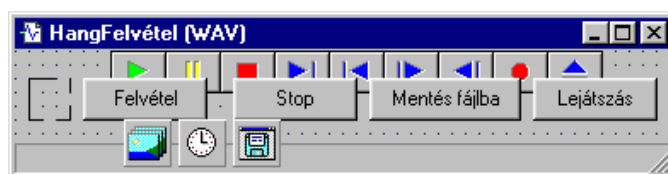
Az alkalmazásból való kilépés előtt fel kell szabadítani a lefoglalt médiaeszközt:



```
procedure TMediaFrm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if MediaPlayer1.DeviceID<>0 then
    begin
      MediaPlayer1.Stop;
      MediaPlayer1.Close;
    end;
end;
```



Készítsünk alkalmazást, amely hangfelvételt készít WAV fájlokba a **MediaPlayer** vezérlőelem segítségével! (*Hangfelvétel*)

Az alkalmazásunk készítését kezdjük a szükséges vezérlőelemek formon való elhelyezésével! A hangfelvételt egy médialejátszó (**MediaPlayer**) segítségével végezzük, melynek **Visible** tulajdonságát állítsuk be **false**-ra! Ahhoz, hogy egy láthatatlan médialejátszót vezérelni tudjunk, szükségünk lesz a következő négy nyomógombra (**Button**): egy *Felvétel*, egy *Stop*, egy „*Mentés fájlba*” és egy *Lejátszás* feliratúra. A felvett anyag fájlba való mentéséhez helyezzünk az űrlapra egy **SaveDialog** vezérlőelemet, melynek **Filter** tulajdonságán keresztül állítsuk be a „**Filter Name**” mező értékét „*Hangfájl (WAV)*” szövegre, a **Filter** (fájlnév-maszk, illetve szűrő) mezőben pedig adjuk meg a '*.wav' karakterláncot!



A fenti ábrán látható többi vezérlőelem a felhasználó tájékoztatását szolgálja a hangfelvétel, illetve a felvett hanganyag lejátszása alatt. Az **Image** és az **ImageList** objektumok segítségével egy egyszerű, a hangfelvételi folyamatot kísérő animációt valósíthatunk meg (az **ImageList** objektumon kétszer kattintva töltünk be a képsorozatba két-három kicsit eltérő képet, például  és , amelyek váltogatásával villogó hatást érünk el).

Szöveges tájékoztatásra egy állapotjelző sort (**StatusBar**) használunk, amelynek a **SimplePanel** tulajdonságát **true**-ra állítjuk. Az animációs képváltás, illetve a felvétel kezdete óta eltelt idő megjelenítése mindig egy időzítő (**Timer**) által generált órajelre történik. Az órajel gyakorisága (az időzítő **Interval** tulajdonságában beállított érték) a programunkban legyen 500 ezredmásodperc.

A **MediaPlayer** segítségével elkészített hangfelvételek sajátossága, hogy a hangfelvétel előtt a **DeviceType** tulajdonság **dtWaveAudio**-ra való beállításán kívül feltétlenül be kell állítani egy érvényes WAV-állomány nevét is a **FileName** tulajdonságban. Az újonnan felvett anyag az „alapfájl” hanganyagába a **StartPos** tulajdonságban megadott ezredmásodperctől ékelődik be. Ha **StartPos** értéket 0-ra, vagy pedig a médialejátszó **Length** tulajdonságban tárolt érték, akkor a betöltött médiaanyag hosszára állítjuk, az új felvételrész a fájl elején lesz, illetve hozzáadódik a már meglévő anyaghoz. A **StartPos** megadásával az új felvétel teljes egészében hozzáadódik az alapfájlhoz. Ha azonban meg szeretnénk szabni az új rész hosszát, a **StartPos** tulajdonságon kívül az **EndPos** tulajdonságban is meg kell adni az ezredmásodpercekben kifejezett értéket.

A médialejátszó **StartRecording** metódusa segítségével felvett anyag először egy átmeneti memóriaterületre kerül (így már lejátszható), de addig nem írja felül az alapállományt, amíg meg nem hívjuk a lejátszó **Save** metódusát. Ha el szeretnénk kerülni a hangalapot képező állomány felülírását, a **Save** metódus hívása előtt át kell állítanunk a médialejátszó **FileName** tulajdonságát más fájlnévre. Éppen ezt a megoldást fogjuk alkalmazni a példaprogramban is. Ahhoz, hogy a felvett hanganyag mentes legyen minden idegen zenétől, érdemes létrehozni egy kb. 1 perces „csendes” WAV-állományt (a következő példában **_tmp.wav**), amelyet alapfájlként fogunk használni. Az alapállományt helyezzük például az alkalmazásunk könyvtárába, és a program elindítása után állítsuk be a nevét egy globális változóban:

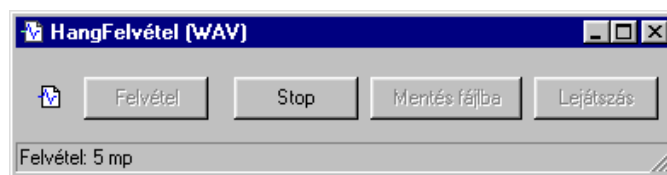
```
var
    alapWAVFajl:string;
...
procedure TForm1.FormCreate(Sender: TObject);
begin
    alapWAVFajl:=GetCurrentDir+'\_tmp.wav';
    SaveDialog1.InitialDir:=GetCurrentDir+'\wav';
    // nyomógombok (nem-) engedélyezése:
    btnFelvetel.Enabled:=true;
    btnStop.Enabled:=false;
    btnMentes.Enabled:=false;
    btnLejatszas.Enabled:=false;
end;
```

A fenti eseménykezelő eljárás második sorában beállítottunk a fájlmentési párbeszédablak kiinduló könyvtárát is (előtte természetesen létre kell hozni egy *wav* nevű könyvtárat, az alkalmazásunk aktuális könyvtárában). Az eljárás további soraiban „kikapcsoltuk” az összes nyomógombot, kivéve a *Felvétel* feliratút.

A *Felvétel* nyomógombon való kattintásra a következő eseménykezelő eljárás fut le:

```
procedure TForm1.btnFelvetelClick(Sender: TObject);
var
  hibaStr:string;
begin
  with MediaPlayer1 do
  begin
    if DeviceID<>0 then // ha van megnyitott médiaeszköz
    begin
      Stop; // lejátszás leállítása
      Close;
    end;
    DeviceType := dtWaveAudio; // médiaeszköz típusa
    FileName := alapWAVFajl; // alaphanganyagot tartalmazó fájl nevének beállítása
    try
      Open; // a csatorna megnyitása
      // az új hanganyag beékelődési pontja:
      StartPos:=1; // 1 ezredmásodperc az alapfájl elejétől
      StartRecording; // hangfelvétel
      // nyomógombok (nem-) engedélyezése:
      btnStop.Enabled:=true;
      btnFelvetel.Enabled:=false;
      btnMentes.Enabled:=false;
      btnLejatszas.Enabled:=false;
      Timer1.Enabled:=true; // időzítő bekapcsolása
    except
      hibaStr := 'Hibakód: ' + IntToStr(Error) + #13#10;
      MessageDlg(hibaStr + ErrorMessage, mtError, [mbOk], 0);
    end;
  end;
end;
```

A felvételi folyamat előtt ki kell kapcsolni az előzőleg elindított lejátszást, be kell állítani a médiaeszköz típusát, illetve a hangfelvételhez használt alapállomány nevét, meg kell nyitni az eszközt, és meg kell adni a kiindulási pozíciót. Ha mindent beállítottunk, meghívhatjuk a hangfelvételt végző ***StartRecording*** médialejátszó-metódust. Végezetül szabályozzuk a nyomógombok elérhetőségét, illetve elindítjuk a felvételt kísérő animációhoz szükséges időzítőt.



Az időzítő eseménykezelőjében nem csak a ***Image*** vezérlőelemben megjelenő képek váltását, hanem az állapotsor feliratának frissítését is elvégezzük:

```
procedure TForm1.Timer1Timer(Sender: TObject);
const
  melyik:integer=0;
  eltelt:Cardinal=0;
begin
  // képváltás (animáció):
  ImageList1.GetBitmap(melyik, Image1.Picture.Bitmap);
  Image1.Repaint;
  melyik:=(melyik+1) mod ImageList1.Count;
  // állapotsor feliratának frissítése:
  eltelt:=eltelt+Timer1.Interval;
  StatusBar1.SimpleText:='Felvétel: ' + IntToStr(eltelt div 1000) + ' mp';
end;
```

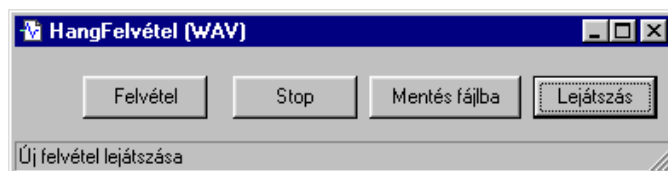
A *Stop* gombra való kattintással leállíthatjuk a médialejátszó által éppen végzett műveletet (a jelen alkalmazásban ez hangfelvétel, vagy hanglejátszás lehet), illetve kikapcsoljuk az időzítőt, és ezzel együtt az animációt is:

```
procedure TForm1.btnStopClick(Sender: TObject);
begin
    MediaPlayer1.Stop;           // médialejátszó leállítása
    Timer1.Enabled:=false;       // az időzítő kikapcsolása
    Image1.Picture.Bitmap:=nil;  // az animált kép "eltüntetése"
    // nyomógombok (nem-) engedélyezése:
    btnStop.Enabled:=false;
    btnFelvetel.Enabled:=true;
    btnMentes.Enabled:=true;
    btnLejatszas.Enabled:=true;
    StatusBar1.SimpleText:='';    // az állapotsor feliratának törlése
end;
```

Mivel rögtön a felvétel után az új, még el nem mentett hanganyag már lejátszható, a *Lejátszás* nyomógomb eseménykezelőjében csak vissza kell tekerni a lejátszót a **Rewind** metódus segítségével, és elindíthatjuk a lejátszást a **Play** metódus hívásával:

```
procedure TForm1.btnLejatszasClick(Sender: TObject);
begin
    MediaPlayer1.Rewind; // médialejátszó visszatekerése
    MediaPlayer1.Play;   // lejátszás elindítása
    btnStop.Enabled:=true; // a Stop gomb engedélyezése

    // az állapotjelző sor feliratozása:
    if MediaPlayer1.FileName = alapWAVFajl
    then StatusBar1.SimpleText:='Új felvétel lejátszása'
    // ha már elmentettük az új felvételt egy WAV fájlba:
    else StatusBar1.SimpleText:=MediaPlayer1.FileName+' lejátszása';
end;
```



A fenti eseménykezelő eljárás állapotsor-átírásra használt **if ... then ... else** struktúrájában azt teszteljük, hogy a médialejátszó **FileName** tulajdonsága még mindig az alapállomány nevét tartalmazza-e (az alapbeállítást a *Felvétel* gomb lenyomásakor adtuk meg). A fájlnev átállítását a „*Mentés fájlba*” feliratú nyomógomb eseménykezelője végzi:

A hangfelvétel mentésekor arra kell figyelni, hogy a fájlmentési párbeszédablak által visszaadott állománynév ne legyen az alapfájlként használt hangnélküli állományunk neve (ellenkező esetben más fájlnev kiválasztását javasoljuk a felhasználónak), illetve a fájl mindenképpen *.wav* kiterjesztéssel rendelkezzen. Miután beállítottuk a médialejátszó **FileName** tulajdonságában a mentéshez használt állomány nevét, már hívhatjuk is a mentést végző **Save** metódust.

```

procedure TForm1.btnMentesClick(Sender: TObject);
var
    s, ujfajl, hibaStr:string;
begin
    if SaveDialog1.Execute then      // ha megadtunk egy fájlnevet a Save párbeszédablakban
    begin
        ujfajl:= SaveDialog1.FileName;
        if (ujfajl <> alapWAVFajl) then // ha a mentési fájlnev nem az alapfájlt jelöli
        begin
            StatusBar1.SimpleText:='Felvétel mentése...';
            // a mentési fájlnevnek .wav kiterjesztéssel kell végződnie:
            s:=AnsiUpperCase(ExtractFileExt(SaveDialog1.FileName));
            if (s <> '.WAV') then ujfajl:=SaveDialog1.FileName+'.wav';
            with MediaPlayer1 do
            begin
                DeviceType := dtWaveAudio;
                FileName := ujfajl;
                try
                    Save;           // a felvétel mentése a FileName-ben beállított nevű állományba
                    StatusBar1.SimpleText:='Felvétel mentése O.K.';
                except
                    hibaStr := 'Hibakód: ' + IntToStr(Error) + #13#10;
                    MessageDlg(hibaStr + ErrorMessage, mtError, [mbOk], 0);
                    StatusBar1.SimpleText:='Felvétel mentése nem sikerült';
                end;
            end;
        end
        else begin           // ha a mentési fájlnev az alapfájlt jelöli:
            MessageBeep(0);
            StatusBar1.SimpleText:='Válasszon más fájlnevet!';
        end;
    end;
end;

```

Mint minden olyan objektum esetén, amely rendszer-erőforrásokat foglal, az alkalmazás befejezése előtt fel kell szabadítani a lefoglalt erőforrásokat:

```

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    with MediaPlayer1 do
        if DeviceID<>0 then // ha van megnyitott médiaeszköz
        begin
            Stop; // lejátszás leállítása
            Close; // médialejátszó lezárása
        end;
end;

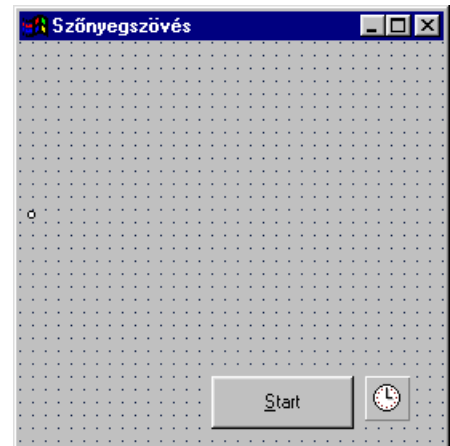
```



Az alkalmazásunk formján helyezük el a következő vezérlőelemeket: egy *Start* feliratú nyomógombot (**Button**), egy időzítőt (**Timer**) és egy alakzat-objektumot (**Shape**). Az időzítőt kapcsoljuk ki, beállítva az **Enabled** tulajdonságának értékét **false**-ra, a **Shape** vezérlőelem **Visible** tulajdonságát szintén állítsuk át **false**-ra.

A *Start* gomb lenyomására a következő eseménykezelő eljárás fut le:

```
procedure TForm1.btnStartClick(Sender: TObject);
begin
    // a Start gomb láthatatlanná tétele
    btnStart.Visible := false;
    keret();           // színes keret rajzolása
    Timer1.Interval := 50; // az időzítő eseményeinek
                        // g ezredmásodpercben
end;
```



Az összes rajzolási műveletet a **Form** objektum **Canvas** tulajdonságán keresztül végezzük el. A szőnyeg szövésszíneit megadó színes keretet a **private keret** metódus segítségével rajzoljuk meg:

```
procedure TForm1.Keret(); // szivárványszínű keret rajzolása
var
    i: Longint;
const
    szelesseg = 8; // a keret szélessége
begin
    // Rajzolási szélesség beállítása:
    Form1.canvas.Pen.Width:= 1;

    // A keret felső és alsó részének rajzolása:
    for i:=0 to Form1.ClientWidth do
    begin
        // A keret felső léccének rajzolása:
        Form1.canvas.Pen.Color:= i*256*256;           // színbeállítás
        Form1.canvas.MoveTo(i + Szelesseg, 0);
        Form1.canvas.LineTo(i + Szelesseg, Szelesseg);
        // A keret alsó léccének rajzolása:
        Form1.canvas.Pen.Color:= i * 256 * 256+i;
        Form1.canvas.MoveTo(i + Szelesseg, Form1.ClientHeight - Szelesseg);
        Form1.canvas.LineTo(i + Szelesseg, Form1.ClientHeight);
    end;

    // A keret oldalsó részeinek rajzolása:
    for i:= 0 to Form1.ClientHeight do
    begin
        // A keret bal léccének rajzolása:
        Form1.canvas.Pen.Color:= i;
        Form1.canvas.MoveTo(1, i);
        Form1.canvas.LineTo(Szelesseg+1, i);
        // A keret jobb léccének rajzolása:
        Form1.canvas.Pen.Color:= i * 256;
        Form1.canvas.MoveTo(Form1.ClientWidth-1, i);
        Form1.canvas.LineTo(Form1.ClientWidth - Szelesseg-1, i);
    end;
end;
```

Ezek után az elindított időzítő a *private* szöves metódust hívja, az *Interval* tulajdonságában beállított időközönként (a *golyo* után rajzolendő nyomvonal színe a kerettel való ütközés helyén felvett szín lesz):

```

procedure TForm1.Timer1Timer(Sender: TObject);
begin
    szoves();
end;

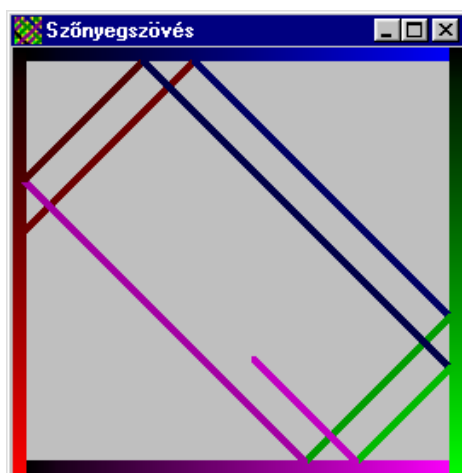
procedure TForm1.szoves();
const
    dX: Integer=-5;
    dY: Integer=-5;
    szin: Longint=256*256;
    Hatar = 5;
var
    CurrentX, CurrentY :Longint;
begin
    // a golyó (Shape) aktuális középpont-koordinátának kiszámítása:
    CurrentX := Golyo.Left + Golyo.Width div 2;
    CurrentY := Golyo.Top + Golyo.Height div 2;

    // A golyó kerettel való ütközése esetén átállítjuk
    // a nyomvonal színét, illetve a mozgás irányát
    if (Golyo.Left <= Hatar) or (Golyo.Left+Golyo.Width >= Form1.ClientWidth-Hatar) then
        begin
            dX := -dX;
            // a rajzolási szín a golyó ütközési helyén fellelt keretszín lesz:
            szin := Form1.canvas.Pixels[CurrentX - dX, CurrentY];
        end;
    if (Golyo.Top <= Hatar) or (Golyo.Top+Golyo.Height >= Form1.ClientHeight-Hatar) then
        begin
            dY := -dY;
            // a rajzolási szín a golyó ütközési helyén fellelt keretszín lesz:
            szin := Form1.canvas.Pixels[CurrentX, CurrentY - dY];
        end;

    // A golyó X és Y irányú eltolása:
    Golyo.Left := Golyo.Left + dX;
    Golyo.Top := Golyo.Top + dY;

    Form1.canvas.Pen.Width:= 3; // rajzolási szélesség (a golyó nyoma) beállítása
    // A golyó nyomának kirajzolása:
    Form1.canvas.Pen.Color:= szin; // a kiszámított rajzolási szín beállítása
    Form1.canvas.MoveTo(CurrentX, CurrentY);
    Form1.canvas.LineTo(CurrentX + dX, CurrentY + dY);
end;

```



A program indításakor a **Form** vezérlőelem **Canvas** tulajdonságán keresztül kijelöljük a rajzterületet (**ClipRect**).

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  with Form1.Canvas.ClipRect do
  begin
    Left:=0;
    Top:=0;
    Right:=Form1.ClientWidth;
    Bottom:=Form1.ClientHeight;
  end;
end;
```

Ha eléggé türelmesek vagyunk, egy idő múlva az elindított alkalmazásunk a következőhöz hasonló „szőnyeget” készíti:





A játékprogramunk formján helyezzük el a következő ábrán is látható vezérlőelemeket: egy főmenüt (**MainMenu**), egy időzítőt (**Timer**), egy *start* feliratú nyomógombot (**Button**) és egy állapotjező sort (**StatusBar**) három panellel! Az időzítőt kapcsoljuk ki, az **Enabled** tulajdonságának értékét **false**-ra állítva.



A játék lényege, hogy a felhasználó minél gyorsabban „lője ki” az össze-vissza szaladgáló nyomógombokat a gombon való kattintás segítségével:



Az alkalmazásunknak szüksége van néhány globális azonosítóra is:

```
const
  n=15;
  . . .
private
  futok      : array [1..n] of TButton;
  idozitok   : array [1..n] of TTimer;
  . . .

implementation
{$R *.DFM}
var
  lathatok: integer;
  eltelt: longint;
  dX: array [1..n] of integer;
  dY: array [1..n] of integer;
```

Az n - n db gombot (*futok*) és a „futtatásukat” végző időzítőkett (*idozitok*) dinamikusan hozzuk létre a nyomógombon való kattintáskor:

```
procedure TForm1.btnStartClick(Sender: TObject);
var i:integer;
begin
    eltelt:=0;    // a játék keletjétől eltelt idő
    lathatok:=n;  // a formon még látható (ki nem lőtt) gombok száma

    with StatusBar1.Panels do    // állapotjelző sor feliratozása
    begin
        Items[0].Text:='Összesen: '+IntToStr(lathatok);
        Items[1].Text:='Maradt még: '+IntToStr(lathatok);
        Items[2].Text:='Eltelt: 0 mp';
    end;

    btnStart.visible:=false;    // a Start nyomógomb elrejtése

    randomize;

    for i:=1 to n do    // a gombokra vonatkozó X és Y irányú eltolási lépések megadása
    begin
        dX[i]:=-3+random(6);
        dY[i]:=-3+random(6);
        // az eltolási lépés nem lehet nulla, különben a domb nem mozdul el:
        if dX[i]=0 then dX[i]:=1;
        if dY[i]=0 then dY[i]:=1;

        // a gombok létrehozása a Form1 vezérlőelemen:
        futok[i]:=TButton.Create(self);
        with futok[i] do
        begin
            left:=random(Form1.ClientWidth-100)+40;    // véletlenszerű bal koordináta
            top:=i*10;
            width:=35;
            height:=20;
            parent:=self;
            cursor:=crHandPoint;    // a gombon az egérkurzor kéz alakú lesz
            visible:=true;
            tag:=i;    // a Tag tulajdonság a gomb sorszámát tartalmazza
            // a gombon való kattintás estén a FutoProc eseménykezelő eljárás fog lefutni:
            onclick:=FutoProc;
        end;
    end;

    // az időzítők létrehozása a Form1 vezérlőelemen:
    for i:=1 to n do
    begin
        idozitok[i]:=TTimer.Create(self);
        with idozitok[i] do
        begin
            Interval:=50;
            enabled:=true;    // az időzítő bekapcsolása
            tag:=i;    // a Tag tulajdonság az időzítő sorszámát tartalmazza
            // az időzítő eseménykezelő eljárásának a megadása:
            ontimer:=IdozitoProc;
        end;
    end;
    Timer1.enabled:=true;    // a főidőzítő bekapcsolása
end;
```

A főidőzítő feladata a játék elejétől eltelt idő kiírása az állapotjelző sor egyik paneljében:

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
    eltelt:=eltelt+Timer1.Interval;
    StatusBar1.Panels.Items[2].Text:='Eltelt: '+IntToStr(eltelt div 1000)+' mp';
end;
```

A gombok eseménykezelő eljárása a következőképpen néz ki:

```
procedure TForm1.FutoProc(Sender: TObject);
begin
    // az aktuális gombot (amelyre rákattintottak) el kell tüntetni
    // (láthatatlanná tétellel),
    // a megfelelő időzítőt pedig ki kell kapcsolni:
    idozitok[(sender as TButton).tag].enabled:=false;
    futok[(sender as TButton).tag].visible:=false;

    lathatok:=lathatok-1; // eggyel kevesebb gomb látszik az alkalmazás ablakában
    // felirat frissítése:
    StatusBar1.Panels.Items[1].Text:='Maradt még: '+IntToStr(lathatok);

    if lathatok=0 then // ha az összes gombot kilőttünk
        with StatusBar1.Panels do
            begin
                Timer1.enabled:=false; // a főidőzítő kikapcsolása
                Application.MessageBox('Játék vége!', PChar(Items[2].Text), MB_OK); // üzenetablak
                Items[1].Text:=''; // felirat törlése
                Items[2].Text:='';
                btnStart.visible:=true; // a Start gomb megjelenítése
            end;
end;
```

Az időzítők eseménykezelő eljárása pedig a következő lesz:

```
procedure TForm1.IdozitoProc(Sender: TObject);
var
    melyik: integer;
begin
    melyik:=(sender as TTimer).tag; // milyen sorszámú gombnak felel meg az időzítő
    with futok[melyik] do
        begin
            // a gomb koordinátáinak eltolása:
            left:=left+dX[melyik];
            top:=top+dY[melyik];

            // formhatárok elérése esetén változik a gomb mozgásiránya, a gomb „visszapattan”:
            if (left<6) or (left>(Form1.ClientWidth-width-6))
                then dX[melyik]:=-dX[melyik];
            if (top<6) or (top>(Form1.ClientHeight-StatusBar1.Height-25))
                then dY[melyik]:=-dY[melyik];
        end;
end;
```

Az utolsó tennivalónk – az alkalmazás befejeztével - megsemmisíteni a dinamikusan létrehozott objektumokat:

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var i: integer;
begin
    for i:=1 to n do
        begin
            futok[i].Free;
            idozitok[i].Free;
        end;
    Application.Terminate;
end;
```