7. tétel

Self kulcsszó. Grafikus lehetőségek a Delphiben.

A, Self kulcsszó: egy form kódjában soha nem szabad közvetlenül a Delphi által hozzárendelt változóval az illető formra hivatkoznunk. Például ha a TForm3 kódjában a Form3.Caption-ra (vagyis a form címsorára) hivatkozunk, akkor ha a formból (vagyis a TForm3 osztályból) második példányt hozunk létre, akkor a Form3.Captino kifejezés a Form3 változóval azonosított form objektumra fog hivatkozni, ami nem biztos, hogy ugyanaz az objektum, mint amelyik végrehajtja a kódot. Ennek elkerülésére a Caption tulandonságra a formok tagfüggvényén belül hivatkozzunk, és használjuk a Self kulcsszót, ha az adott form valamelyik objektumára hivatkozunk. Tehát, ha egy formról több másolatot készítünk, a problémák elkerülése végett ajánlatos eltávolítanunk a globális form objektumot a formot leíró egység `interface` részéből, ugyanis ez a globális változó csak az automatikus létrehozáshoz szükséges.
Ha a Self-et úgy kezeljük, mint az eseménykezelő tagfüggvény meghívásának paraméterét, biztosak lehetünk abban, hogy a tagfüggvény meghívásakor annak Sender paraétere valóban az eseményt elindító objektumra hivtkozik, pontosan úgy, ahogyan azt komponens felhasználóként elvárjuk.
Minta a létrehozásra: a kód hatására létrejön egy gomb a kattintás helyén, a gomb felirata pedig 'Ez egy gomb' lesz.

```
Var btn: Tbutton;
Begin
        btn:=Tbutton.create ( Self );
        btn.Parent := Self;
        btn.Left := 30;
        btn.Top := 20;
        btn.Width := btn.width+50;
        btn.caption := 'Ez egy gomb';
End;
```

B, Grafikus lehetőségek a Delphiben a grafikus megjelenítést egy grafikus alprogramrenszer végzi, melynek neve GDI (Graphics Device Interface). Többféle módon készíthetünk grafikus ábrát:

- használhatjuk a grafikus vezérlőket, melyeknek ábráit az alkalmazás tervezésekor, vagy a program futása során hozzuk létre,
- bizonyos komponensek felületére – csak a futó programból – rajzolhatunk
- A képernyő mellett a nyomtatón is megjeleníthetünk információkat.

Alapvető grafikus eszközök: a rajzolás során speciális tulajdonságokkal rendelkező alapeszközöket (toll, ecset, betűtípus és bitkép) használhatunk.

1. Színek=> a TColor típussal színek tárolására alkalmas változókat definiálhatunk. A színdefiníciókban a piros, a zöld és a kék értékeket, 0-tól $FF-ig terjedően, az rr,gg,bb számok jellemzik A leggyakrabban használt színeket konstansként is elérhetjük (clBackground: háttérszín, clMenu: menüszín, clWindowFrame: ablak keretének a színe)

2. Rajzolási módok=> a színezéshez hasonlóan azt is megadhatjuk, hogy a különböző rajzeszközök hogyan rajzoljanak. A grafikus objektumok felületére tollakkal (TPen) vonalakat lehet húzni. Ha azt szeretnénk szabályozni, hogyan fogjon a toll, annak PenMode által meghatározott logikai műveletekkel összekapcsolva

kapjuk meg a húzott vonal színét. A TPenMode típus lehetséges értékei pld.: fekete vonal – pmBlack, fehér vonal – pmWhite

Alapeszközök:

- Toll (Pen): vonalas ábrák készítéekor használjuk. Jellemzője a rajzolási módja (Mode), a színe (Color), a vastagsága (Width), és a húzott vonal típusa (Style)
- Ecset (Brush): grafikus elemek kifestéséhez használjuk az ecsetet (TBrush). Jellemző tulajdonságai a szín (Color) és a kifestés módja (Style)
- Betűtípus (Font): a kiíráshoz használt betűtípust adhatjuk meg. Betű a tulajdonságai: Color – karakter színe, a karaktert befogadó cella magassága – Height, a betűtípus mérete – Size, a betűtípus családja – Name. A betűk a Windowsban megszokott formátumok lehetnek: normál, félkövér, dőlt, áthúzott (fsStrikeOut).
- Bitkép láthatóvá tétele (Bitmap)

3. A képek=>: kétféle leírása létezik:
   - Raszter típusú: a pontokból álló kép színeit definiáljuk,
   - Vektoros : az ábra rajzolásának olyan lépéseit tároljuk, mint vonalhúzás, kifestés stb.

   Sokszor találkozunk ikonokkal (ico), melyek speciális méretű- és szerkezetű bitképek. A raszteres képleirás speciális módszerekkel történő tömörítésre alkalmasak a .jpg és a .jpeg típusú állományok.
   A képek méreteit – képpontban - meghatározhatjuk (integer típusú) height és width tulajdonságokkal.

4. Geometriai alakzatok megjelenítése=> a komponenspaletta Additional lapján lévő Shape komponenst haszáljuk egyszerű geometriai alakzatok megjelenítésére.
5. Képek megjelenítése=>a következő vezérlőkkel képeket lehet megjeleníteni oly módon, mintha az bekeretezett rajzvásznon lenne:
   - PaintBox: a komponenspaletta System lapján található PaintBox (festőmező) rajzvásznán csak rajzeszközökkel rajzolhatunk.
   - Image: ez a vezérlő is egy rajzvászon. Az Image vezérlő Picture tulajdonsága meghatározza, hogy mit látunk a 'falra akasztott képen'. A keret lehet képhez igazodó (AutoSize), azonban a képet ki is feszíthetjük a keretre a Stretch tulajdonság tru-ra állításával.
6. Képlisták=> gyakran használunk több azonos méretű képet. A – TimageList típusú – ImagiList komponens egyetlen nagy bitképben, a képernyőhöz illeszkedő formában tárolja a képeket. Ezek méretét a Height és Width tulajdonsággal, számát a Count jellemző határozza meg. Mindegyik képhez tartozik egy index, melynek alapján elérhető.

A Canvas tulajdonság használata: a Canvas objektumot úgy képzelhetjük el, mint egy festőkészletet. Négy alapvető rajzeszköze: toll, ecset, betűtípus, bitkép.

Néhány alapfogalom:

1. Objetum kirajzolásának határai (ClipRect)=> a Trect egy téglalaprekord, melynek mezői kijelölik a bal- és jobb sarkot.

```
Trect = record
    Case integer of
        0: (left, top, right, bottom: integer);
```

1: (TopLeft, BottomRight: Tpoint);
end;

A (Trect típusú) Canvas.ClipRect csak olvasható tulajdonság meghatározza a rajz (vászon) határait.

2. Koordináta-rendszer=> alapértelmezésben a 0,0 pont a képernyő bal felső sarka. Ha ezen változtatni akarunk, akkor a SetMapMode fgv.-t használhatjuk.

Rajzolás a Canvas objektum felhasználásával: a vásznon a pontok színének kezelésére a (TColor típusú elemeket tartalmazó) Pixels tulajdonságot használhatjuk.

Property Pixels(x,y:integer):TColor;

1. rajzolás tollakkal=>a toll aktuális pozícióját a PenPos tulajdonság tárolja. A toll x,y pozíciójára a MoveTo (x,y:integer) metódus használatával ugorhatunk. Vonal rajzolására a LineTo(x,y:integer), vonallánc rajzolására a PolyLine (const points: array of Tpoint) metódussal rajzolhatunk.
2. Festés a rajzvásznon=>rajzterületek kifestésére az ecsetet használjuk. A Canvas objektum úgy kezeli az ecsetet, mint a tollat. Pld.: téglalapot – Rectangle (x1,y1,x2,y2) metódussal, lekerekített téglalap – RoundRect (x1,y1,x2,y2) metódussal, kifestett poligont – Polygon (const points:array of Tpoint) metódussal, ellipszist – ellipse (x1,y1,x2,y2:integer) metódussal rajzolhatunk,stb.
3. Írás a rajzvászonra=> TextOut (x,y:integer; const text:string)
4. Bitképek megjelenítése=>mérettorzítás nélkül a Draw (x,y:inteeger;graphic:Tgraphic); metódussal. (a Graphic paraméter definiálja a grafikus objektumot, az x,y pedig a bal felső sarkot.) Torzítva a StrechtDraw (const Rect:Trect; Graphic:Tgrapic); metódussal, ahol a Rect azt a téglalapot definiálja, ahova a Graphic kép kerül.

Kirajzolás, újrafestés: az ablakokban megjelenő grafikus objektumot gyakran eltakarja egy másik ablak, majd amikor megszűnik a takarás, újra ki kell rajzolni a képet. Ez úgy történik, hogy az újrarajzoláshoz a Windows automatikusan érvényteleníti a takart területet és meghívja az objektumok OnPaint eseményének kezelőjét. Ha tehát azt szeretnénk, hogy a rajz a takarásból előbukkanó objektumokon frissítésre kerüljön, akkor a Canvas rajzoló metódusait az OnPaint esemény kezelőjében kell elhelyezni.