

# PRESEK

List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652

Letnik 24 (1996/1997)

Številka 4

Strani 210-214

Jože Marinček:

## NAREDI SVOJ PAINTBRUSH

Ključne besede: računalništvo, okna, windows.

Elektronska verzija: <http://www.presek.si/24/1301-Marincek.pdf>

© 1997 Društvo matematikov, fizikov in astronomov Slovenije

© 2010 DMFA - založništvo

Vse pravice pridržane. Razmnoževanje ali reproduciranje celote ali posameznih delov brez poprejšnjega dovoljenja založnika ni dovoljeno.

## NAREDI SVOJ PAINTBRUSH

### Zamisel

Program `PaintBrush` je med začetniki zanesljivo eden najbolj priljubljenih dodatkov k Oknom. Razlog je na dlani: je enostaven za uporabo, risba pa je sploh eno človekovih najstarejših izraznih sredstev, v kar se lahko prepričamo, denimo, v španski Altamiri.

Uporabljeni program pa ni niti zdaleč tako zanimivo, kot napisati ga. Zakaj ne bi torej napisali svojega programa `PaintBrush`? Poskusimo to storiti v Visual BASICu.

Naš prvi poizkus bo seveda kar se da enostaven. Ko bomo premaknili miš, hočemo na zaslonu videti ustrezno črto. Kadarkoli se miš premakne, Okna pošljejo ustrezno sporočilo.

Program, napisan v Visual BASICu, se na to sporočilo odzove tako, da pokliče podprogram `MouseMove`. Torej moramo samo napisati ustrezni podprogram:

```
Sub Form_MouseMove (Button As Integer,
                    Shift As Integer,
                    X As Single,
                    Y As Single)
    Line -(X, Y)
End Sub
```

Program napišemo tako, da dvakrat kliknemo na okno z napisom `Form1`. Pokaže se urejevalnik, v katerem izberemo podprogram `MouseMove` in dopišemo manjkajočo vrstico.

Ukaz `Line (x1, y1)-(x2, y2)` nariše črto med točkama  $(x_1, y_1)$  in  $(x_2, y_2)$ . Če prvo točko izpustimo, nariše črto od zadnje točke risanja do točke  $(x_2, y_2)$ . Zadnja točka risanja se spreminja ob ukazih za risanje (`Line`, `Circle`, `PSet`, ...). Koordinati zadnje točke risanja sta shranjeni v spremenljivkah `CurrentX` in `CurrentY`. Ko program poženemo, velja `CurrentX = CurrentY = 0`, točka  $(0, 0)$  pa je v levem zgornjem kotu okna.

Naš program za risanje že deluje. Ima pa še nekaj pomanjkljivosti, ki jih hitro opazimo. Tako ne moremo premakniti miške, ne da bi se za njo poznala sled. Naš vzornik (program `Paintbrush`) to rešuje tako, da se sled pozna le, če med premikanjem miške njen levi gumb držimo pritisnjen. Druga večja hiba je naslednja: Če miškin kazalec zapusti naše okno, se risanje prekine (miš pa je na voljo drugim programom). Ko pa se z mišjo vrnemo v okno (denimo na drugem koncu), program potegne ravno črto od točke, kjer smo okno zapustili, do točke, kjer smo se v okno vrnili.

Najprej odpravimo prvo pomanjkljivost (pokazalo se bo, da bomo tako nehoti popravili tudi drugo). Parameter `Button` podprograma `MouseMove` nam pove, kateri miškin gumb smo držali, ko smo jo premaknili. Vrednost 0 pomeni, da ni bil pritisnjen noben gumb. Če je bil pritisnjen levi gumb, se prišteje 1. Če je bil pritisnjen desni gumb, se prišteje 2, in če je bil pritisnjen srednji gumb, se prišteje 4. Tako lahko prepoznamo tudi različne kombinacije. Vrednost `Button=3` na primer pomeni, da smo hkrati pritisnili tako levi kot desni gumb. Seveda črto vlečemo le, če je pritisnjen levi gumb – to pa je natanko tedaj, ko je vrednost spremenljivke `Button` liha. Naš dopolnjeni podprogram se sedaj glasi:

```
Sub Form_MouseMove (Button As Integer,  
                    Shift As Integer,  
                    X As Single,  
                    Y As Single)  
    If Button Mod 2 = 1 Then  
        Line -(X, Y)  
    EndIf  
End Sub
```

Žal program še ne deluje povsem tako, kot smo hoteli. Dokler ne držimo nobenega gumba, program res ne vleče črte. Čim pa pritisnemo gumb, program poveže konec narisane črte s trenutnim položajem. Seveda, ukaz `Line -(x,y)` potegne črto med točkama (`CurrentX`, `CurrentY`) in (`x,y`), od zadnjega risanja pa se vrednosti `CurrentX` in `CurrentY` nista nič spremenili. Sedaj, ko poznamo problem, hitro odkrijemo tudi rešitev: ko črte ne vlečemo, enostavno popravimo vrednosti spremenljivkama `CurrentX` in `CurrentY` na trenutni položaj. Tega izvemo iz argumentov `X` in `Y` podprograma `MouseMove`.

```
Sub Form_MouseMove (Button As Integer,  
                    Shift As Integer,  
                    X As Single,  
                    Y As Single)  
    If Button Mod 2 = 1 Then  
        Line -(X, Y)  
    Else  
        CurrentX = X  
        CurrentY = Y  
    EndIf  
End Sub
```

Tako spremenjen program pa že omogoča prostoročno risanje, kot smo želeli. Mimogrede preverimo še, kaj se zgodi, če v programu z miško zapustimo zaslon na enem koncu in se vrnemo v okno nekje drugje.

## Nastavitev parametrov risalnega okna

Naš program pa še vedno ni popolnoma zrel za uporabo. Če ne verjamete, narišite enostavno sliko, potem pa program pomanjšajte in za tem vrnite na prvotno velikost. Slika je izginila!


Na srečo moramo samo nastaviti parameter ali dva in stvar bo urejena. Program ustavimo (če teče), potem pa v okolju Visual BASIC izberimo okno z napisom `Form1`. V oknu **Properties** (*lastnosti*) lahko sedaj nastavimo nekatere lastnosti tega okna (če okna **Properties** ne najdete, ga lahko priključite s pomočjo tipke **F4**). Za nas bosta ta hip pomembni le dve: `AutoRedraw` in `Caption`.

Lastnost `AutoRedraw` ima lahko vrednost `False` ali `True`. Če je njena vrednost `False`, je programer (torej mi) odgovoren za prikazovanje slike, ko se iz tega ali onega razloga del slike izgubi. Najpogostejši razlog, da izgubimo del slike, je, da je bilo okno zakrito. Vrednost `True` pa pomeni, da bo Visual BASIC sam poskrbel za obnavljanje slike. Uporabnost te izbire je očitna, saj nam ni treba storiti praktično ničesar, slika pa bo obnovljena. Žal pa program pri tej izbiri porabi več pomnilnika in tudi risanje je nekoliko počasnejše. No, v našem primeru pravzaprav nimamo druge možnosti, saj bi bilo sicer prostoročno risbo zelo težko obnoviti. Zato postavimo lastnost `AutoRedraw` na `True`.

Lastnost `Caption` določa napis na oknu. Ker napis `Form1` ni najbolj primeren, ga spremenimo, denimo `Risanje` (ali `Moj Paintbrush` ali kaj tretjega). Lastnost `Caption` ne vpliva na delovanje programa, pač pa le na njegov videz.

## Pomembni dodatki

Slabost našega programa je nedvomno tudi ta, da črte, ki smo jo potegnili, ne moremo izbrisati na noben drug način, kot da program ustavimo in znova poženemo. Rešitev je tudi tokrat preprosta: podprogram `Cls` (`Clear Screen`) pobriše zaslon. Odločiti se moramo le, kdaj bomo ta podprogram poklicali.

Takoj pomislimo na ukazni gumb. Vendar ga nimamo kam postaviti, saj je celotna površina okna naša risalna deska, torej bi bil gumb napoti. No, nič ne de – risalno površino bomo nekoliko omejili. Uporabili bomo gradnik "slika" (`picture`). V orodjarni izberemo gumb  in na okno

narišemo pravokotnik. Z malo truda bomo dosegli, da bo risanje omejeno na notranjost pravokotnika, zunaj pravokotnika pa bomo pustili dovolj prostora za naš gumb.

Če sedaj poženemo program, ugotovimo, da lahko rišemo le po zunanosti pravokotnika – prav nasprotno od zelenega! Ko se miš premakne, o tem obvesti objekt, ki je pod njo. Če je ta objekt okno, se izvede podprogram `MouseMove`. Ko pa je ta objekt slika, je ustrezní podprogram prazen (v kar se hitro prepričamo, če v Visual BASICu dvakrat pritisnemo na sliko in poiščemo podprogram `MouseMove`). Če želimo risati le znotraj slike, moramo izbrisati podprogram `MouseMove`, ki smo ga napisali za okno `Form1`, in podobnega napisati za sliko `Picture1`. Ukaz `Line` in spremenljivki `CurrentX` ter `CurrentY` se nanašajo na okno, ne na sliko. Če hočemo risati po sliki `Picture1`, moramo dodati predpono `Picture1`, na primer `Picture1.Line` ali `Picture1.CurrentX`. Dobimo torej podprogram

```
Sub Picture1.MouseMove (Button As Integer,
                        Shift As Integer,
                        X As Single,
                        Y As Single)
    If Button Mod 2 = 1 Then
        Picture1.Line -(X, Y)
    Else
        Picture1.CurrentX = X
        Picture1.CurrentY = Y
    EndIf
End Sub
```

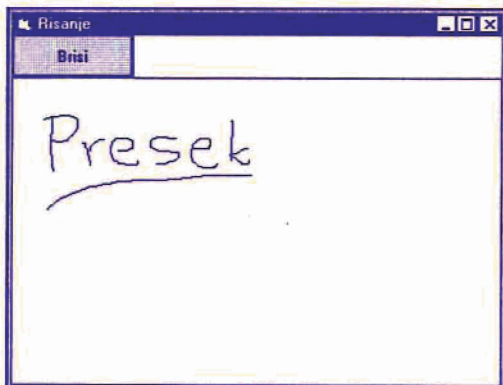
Seveda moramo sliki popraviti tudi lastnost `AutoRedraw`, podobno, kot smo to storili z oknom. Če sliki spremenimo še lastnost `Align` in jo postavimo na `Align Bottom`, bomo dosegli, da bo risalna površina zasedala skoraj vse okno razen ozkega pasu na vrhu. Lastnost `Align` določa, kako se gradnik postavi na okno. Običajno je njena vrednost `0 - None`. Če jo postavimo na `1 - Align Top`, gradnik zavzame vso površino okna nad seboj, in če jo postavimo na `2 - Align Bottom`, zavzame vso površino okna pod seboj.

Dolžni smo še gumb za brisanje risalne površine. Gumb poberemo v orodjarni in ga postavimo v nepokriti pas na vrhu okna. Njegovo lastnost `Caption` popravimo na `Brisi`. Sedaj dvakrat pritisnemo na gumb in Visual BASIC nas postavi v telo podprograma `Command1.Click`. Program bo poklical ta podprogram, kadar bomo pritisnili na gumb. Telo podprograma je kar se da enostavno:

```
Sub Command1.Click ()  
    Picture1.Cls  
End Sub
```

Paziti moramo, da ne pozabimo na predpono `Picture1`, drugače bo ukaz `Cls` pobrisal okno, ne pa tudi slike. Poprej pa smo trudoma dosegli, da je risba omejena samo na sliko.

Tako, program je sestavljen. Ker smo z njim zadovoljni, ga shranimo na disk (`File|Save Project`). Program se bo shranil v dve datoteki. V prvi, s končnico `FRM`, bo zapisan program skupaj z opisom okna in gradnikov na njem. V drugo datoteko, ta ima končnico `MAK`, bo prišel seznam uporabljenih komponent in razne nastavitve. Obema datotekama damo enako



ime, denimo `Risanje`. Za vsakdanjo uporabo pa si lahko naredimo še prevedeni program, `Risanje.EXE`. Uporabimo Ukaz `File|Make EXE file...` Tako preveden program za delovanje ne potrebuje okolja Visual BASIC, zadošča mu že datoteka `VBRUN300.DLL` (pri različici 3.0).

*Jože Marinček*