

# Práca s grafickým rozhraním 2

RNDr. Radovan Palík

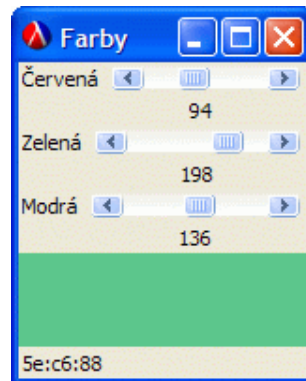
13. februára 2007

## Abstrakt

Použitie pokročilejších metód pri tvorbe grafického používateľského rozhrania v PLT-Scheme pomocou knižnice *PLT MrEd: Graphical Toolbox* a práca s farbami.

## 1 Úvod

V tejto časti si ukážeme použitie triedy *slider%* a prácu s farbami — triedou *color%*. Výsledným produktom by mala byť aplikácia na „miešanie“ farieb vrátane zobrazenia kódu výslednej farby v šesťnástkovom tvare.



## 2 Trieda *color%*

Trieda *color%* reprezentuje R-G-B kombináciu primárnych farieb. Každá jej zložka (červená, zelená, modrá) môže nadobúdať ľubovoľnú celočíselnú hodnotu z intervalu 0..255. Napríklad (0, 0, 0) predstavuje čiernu farbu, (255, 255, 255) bielu a (255, 0, 0) červenú. Vytvoríme teraz objekt *farba* s nulovou hodnotou všetkých zložiek.

```
(define farba (make-object color% 0 0 0))
```

Tento objekt bude slúžiť na uloženie výslednej farby miešania a touto farbou bude vyplnená plocha pod trojicou posuvníkov.

### 3 Trieda *slider%*

Posuvníky slúžia na rýchle nastavenie hodnoty alebo požadovanej pozície v inom ovládacom prvku. My ich v našej aplikácii využijeme na nastavenie hodnôt jednotlivých farebných zložiek. Najprv si však vytvoríme hlavné aplikačné okno:

```
(define frame (instantiate frame% () (label "Farby") (x 300) (y 300)))
```

Teraz pridajme prvý posuvník na nastavenie červenej farebnej zložky:

```
(define cervena (instantiate slider% ()  
  (label "Cervena")  
  (min-value 0)  
  (max-value 255)  
  (parent frame)  
  (style '(horizontal))  
  (callback zmen-farbu/cb)))
```

Konštruktor triedy *slider%* má štyri povinné argumenty: titulok, minimálnu hodnotu, maximálnu hodnotu a rodičovský kontajner. My sme pridali ďalšie dva — štýl (určuje tvar posuvníka) a procedúru, ktorá sa má vyvolať v prípade, že manipulujeme s posuvníkom. O tom, čo všetko obsahuje si povieme neskôr. Teraz pridajme ďalšie dva posuvníky. Keďže už poznáme povinné argumenty a ich poradie, môžeme výsledný kód skrátiť:

```
(define zelena (instantiate slider% ("Zelena" 0 255 frame)  
  (style '(horizontal))  
  (callback zmen-farbu/cb)))  
(define modra (instantiate slider% ("Modra" 0 255 frame)  
  (style '(horizontal))  
  (callback zmen-farbu/cb)))
```

Farebná plocha, ktorá bude meniť farbu podľa nastavenia posuvníkov, bude inštanciou triedy *canvas%*. Jej konštruktor má len jeden povinný argument — rodičovský kontajner:

```
(define plocha (instantiate canvas% (frame)  
  (min-width 50) (min-height 50)))
```

My nastavíme aj jej minimálne rozmery 50×50. Reálna veľkosť sa však bude pri zmene veľkosti okna meniť, nikdy ale nebude menšia než 50 pixelov.

Ako posledný ovládací prvok vložíme text predstavujúci šestnástkovú reprezentáciu čísla farby:

```
(define hexa (instantiate message% ("0:0:0" frame) (stretchable-width #t)))
```

Dvojicu povinných argumentov konštruktora tvoria text a rodičovský kontajner. Nakoniec nastavíme, aby sa šírka tohto ovládacieho prvku automaticky prispôbovala jeho obsahu.

Vytvoríme teraz callback procedúru, ktorá sa spustí pri manipulácii s posuvníkmi a zmení obsah objektu *farba*. Výslednou kombináciou jej zložiek potom zafarbí skúšobnú plochu a tiež vypočíta šestnástkový kód tejto farby.

```
(define (zmen-farbu/cb s e)
  (begin (send farba set
            (send cervena get-value)
            (send zelena get-value)
            (send modra get-value))
         (send plocha set-canvas-background farba)
         (send plocha refresh)
         (send hexa set-label (hex-farba farba))))
```

Má dva argumenty. Prvým je objekt, ktorý vyvolal callback procedúru a druhým samotná udalosť. Keďže v našom príklade ani jeden z argumentov nevyužívame, nebudeme sa nimi momentálne zaoberať.

Hodnoty posuvníkov zistíme zaslaním správy *get-value*. Tie použijeme ako argumenty metódy *set*, ktorá nastaví jednotlivé zložky objektu *farba*. Farbu plochy zmeníme metódou *set-canvas-background*. Jej jediným argumentom je objekt predstavujúci farbu. Aby sa zmena prejavila, musíme zavolať metódu *refresh* zabezpečujúcu prekreslenie plochy. Posledným krokom je výpočet kódu farby. To zabezpečí nasledujúca procedúra:

```
(define (hex-farba a-color)
  (string-append
    (number->string (send a-color red) 16) ":"
    (number->string (send a-color green) 16) ":"
    (number->string (send a-color blue) 16)))
```

Samotné spustenie programu zabezpečíme zaslaním správy *show* formuláru s parametrom *#t*. Pri prvom spustení si všimneme, že farba skúšobnej plochy je biela, kým všetky posuvníky majú hodnotu 0 a teda predstavujú čiernu farbu. Vyriešime to tak, že túto plochu ešte pred samotným zobrazením formulára zafarbíme na čierne:

```
(send plocha set-canvas-background farba)
```

## 4 Výpis programu

```
::objekt farba
(define farba (make-object color% 0 0 0))

::callback procedura
(define (zmen-farbu/cb s e)
```

```

(begin (send farba set
        (send cervena get-value)
        (send zelena get-value)
        (send modra get-value))
        (send plocha set-canvas-background farba)
        (send plocha refresh)
        (send hexa set-label (hex-farba farba))))

;;vypocet hexa-kodu farby
(define (hex-farba a-color)
  (string-append
    (number->string (send a-color red) 16) ":"
    (number->string (send a-color green) 16) ":"
    (number->string (send a-color blue) 16)))

;;definicia objektov
(define frame (instantiate frame% () (label "Farby") (x 300) (y 300)))
(define cervena (instantiate slider% ()
  (label "Cervena")
  (min-value 0)
  (max-value 255)
  (parent frame)
  (style '(horizontal))
  (callback zmen-farbu/cb)))
(define zelena (instantiate slider% ("Zelena" 0 255 frame)
  (style '(horizontal))
  (callback zmen-farbu/cb)))
(define modra (instantiate slider% ("Modra" 0 255 frame)
  (style '(horizontal))
  (callback zmen-farbu/cb)))
(define plocha (instantiate canvas% (frame)
  (min-width 50) (min-height 50)))
(define hexa (instantiate message% ("0:0:0" frame) (stretchable-width #t)))

;;spustenie aplikacie
(send plocha set-canvas-background farba)
(send frame show #t)

```