

Práca s grafickým rozhraním 3

RNDr. Radovan Palík

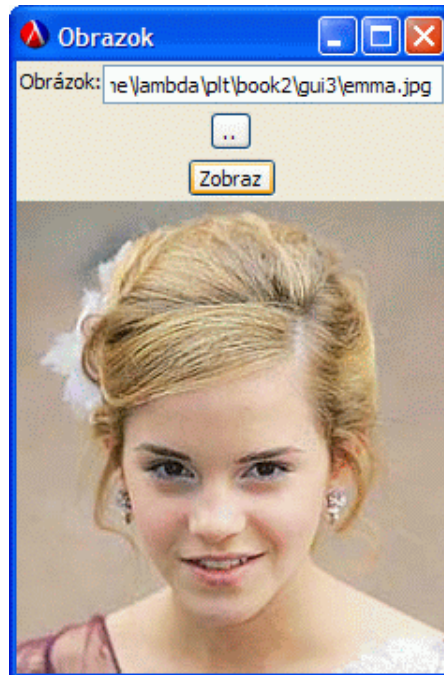
16. decembra 2007

Abstrakt

Práca s bitovými mapami a obrazovými súborami v PLT-Scheme pomocou knižnice *PLT MrEd: Graphical Toolbox*.

1 Úvod

V tejto časti si ukážeme použitie triedy *bitmap%*, ktorá slúži na prácu s bitovými mapami. Vytvoríme si aplikáciu slúžiacu ako jednoduchý prehliadač obrázkov.



2 Trieda *bitmap%*

Trieda *bitmap%* reprezentuje, ako už hovorí jej názov, bitovú mapu. Pod týmto pojmom si pre zjednodušenie môžeme predstaviť obrázok, ktorý je určený tromi parametrami: počtom použitých farieb (bitová hĺbka), šírkou a výškou. Objekt triedy *bitmap%* vytvárame volaním funkcie *make-bitmap*. Ňou vytvorená bitmapa môže byť:

- prázdna, určená len rozmermi a určením, či ide o monochromatickú
- zadaná počtom bitových hodnôt a určením, či ide o monochromatickú
- načítaná zo súboru určením cesty, formátu a farbou pozadia

My použijeme prvý spôsob a na začiatku vytvoríme prázdny bitmapový objekt (**define** *b* (*make-object bitmap% 1 1 #f*)) minimálnych rozmerov. Neskôr si ukážeme tretí spôsob — ako načítať obrázok zo súboru.

3 Trieda *canvas%*

Na to, aby sme mohli bitovú mapu zobraziť, potrebujeme vytvoriť inštanciu triedy, ktorá to podporuje. Najčastejšie používanou je *canvas%* — plátno. Na tento objekt môžeme „kresliť“ priamo, no stretneme sa s problémom známym z iných jazykov a grafických rozhraní. Pri prekrytí plátna, napríklad iným oknom, sa jeho obsah vymaže. My musíme takéto prekrytia kontrolovať a príslušné poškodené časti prekresľovať, čo si vyžaduje značné úsilie. Jednoduchšie je nechať to na knižnicu a okenný manažér použitím *callback* procedúry. Tá je automaticky volaná vždy, keď je potrebné prekresliť plátno. Má dva parametre, *canvas* — plátno, ktoré je treba prekresliť a jeho *dc* — „device context“ — všeobecné označenie akéhokoľvek výstupného zariadenia, ktoré dokáže vykresľovať grafiku a text (plátno *canvas*, tlačiareň, postscriptový súbor, ...).

Zobrazme teda plátno o minimálnej veľkosti 100×100 bodov. Nastavme aj *callback* procedúru, ktorá vykreslí bitmapu *b* na našom plátne.

```
(define plocha (instantiate canvas% (frame)
  (min-width 100)
  (min-height 100)
  (paint-callback
   (lambda (canvas dc)
     (send dc draw-bitmap b 0 0 'solid black #f))))))
```

Samotné zobrazenie bitmapy je realizované zaslaním správy *draw-bitmap* s parametrami:

- bitmapa *b*
- súradnice ľavého horného rohu plátna, kde začneme vykresľovať bitmapu [0, 0]

- štýl vykreslenia (`'solid`, `'opaque`, `'xor`) a farba sú pre farebné obrázky ignorované, preto ich nastavíme na štandardné hodnoty
- maska — nastavíme na štandardnú hodnotu `#f`

Obrázok väčší ako 100×100 bodov by sa nám nezobrazil celý. Preto pred každým vykreslením zistíme jeho rozmery a prípadne zväčšíme plátno nastavením minimálnych rozmerov:

```
(send canvas min-client-width (send b get-width))
(send canvas min-client-height (send b get-height))
```

Poslednou úlohou je načítanie obrázka zo súboru a donútenie plátna, aby sa prekreslilo. Načítanie realizujeme zaslaním metódy `load-file` objektu typu `bitmap`. Parametrami sú cesta k súboru, typ súboru a farba pozadia. Cestu k súboru získame z textového poľa (`text-field%`) — `ifile`. Typ súboru nastavíme na `'jpg`, keďže chceme zobrazovať súbory typu `JPEG`. S farbou pozadia nebudeme pracovať, preto ju nastavíme na `#f`. Nakoniec plochu prekreslíme zaslaním správy (`send plocha refresh`).

Našu pozornosť si zaslúži aj dialóg na výber súboru, konkrétne parameter `filters`. Ten je tvaru (`list (list filter-name filter-glob)`), teda zoznam zoznamov, z ktorých každý obsahuje názov filtra a masku.

```
(define fdialog (instantiate path-dialog% () (label "Dialog")
                    (existing? #t)
                    (filters '((Obrazky "*.jpg")))))
```

4 Výpis programu

```
(require (lib "path-dialog.ss" "mrlib"))
```

```
::zakladne objekty
```

```
(define black (make-object color% 0 0 0))
(define b (make-object bitmap% 1 1 #f))
```

```
::Dialog na nacitanie suboru
```

```
(define fdialog (instantiate path-dialog% () (label "Dialog")
                    (existing? #t)
                    (filters '((Images "*.jpg")))))
```

```
::Aplikacia
```

```
(define frame (instantiate frame% () (label "Obrazok") (x 300) (y 200)))
(define ifile (instantiate text-field% ("Obrazok:" frame) (min-width 200)))
(define ffind (instantiate button%
                    (".." frame)
                    (callback (lambda (button event)
                                (let ([p (send fdialog run)]))
```

```

                                (if (not (equal? p #f))
                                    (send ifile set-value (path->string p))))))
(define view (instantiate button%
  ("Zobraz" frame)
  (callback (lambda (button event)
              (send b load-file (send ifile get-value) 'jpeg #f)
              (send plocha refresh))))))
;;Plocha na zobrazenie obrazka
(define plocha (instantiate canvas% (frame)
  (min-width 100)
  (min-height 100)
  (paint-callback
   (lambda (canvas dc)
     (send canvas min-client-width (send b get-width))
     (send canvas min-client-height (send b get-height))
     (send dc draw-bitmap b 0 0 'solid black #f))))))

(send frame show #t)

```