

KAPITOLA 12

Grafická rozhraní pro Ruby

Není nic horšího, než ostré zobrazení neurčitého konceptu.

– Ansel Adams

Nejsou žádné pochybnosti o tom, že se nacházíme ve věku grafického uživatelského rozhraní (GUI). Je zřejmé, že v budoucnu bude preferovaným způsobem interakce s počítačem nějaká forma grafického rozhraní. Nemyslím si ovšem, že by v dalším desetiletí měl vymizet příkazový řádek – ten má jisté své místo ve světě. Ale dokonce i hackeři ze staré školy (kteří by raději používali příkaz `cp -R` než rozhraní drag-and-drop) někdy používají GUI, když je to vhodné.

S programováním grafiky se neodmyslitelně pojí různé významné problémy. První problém pochopitelně spočívá v návrhu smysluplného a použitelného prostředí programu. V návrhu uživatelského rozhraní nemá obrázek vždy cenu tisíce slov. Tato kniha rozhodně nemůže pojmut celou problematiku tvorby GUI. Naším cílem zde není řešit ergonomii, estetiku či psychologii.

Druhým obvyklým problémem je to, že programování grafiky je složitější. Musíme si totiž dělat starosti o velikost, tvary, umístění a chování všech ovládacích prvků, které mohou být zobrazeny na obrazovce, jež mohou být ovládány myší a/nebo klávesnicí.

Třetí potíž spočívá v tom, že různé počítačové kultury mají odlišné představy o tom, co je systém oken a jak by měl být implementován. Nesourodost mezi těmito systémy musí být prvně vyzkoušena, aby mohla být následně plně pochopena. Nejedna programátor se pokoušel vytvořit nějaký multiplatformní nástroj, aby nakonec zjistil, že tou nejtěžší částí je přizpůsobení GUI.

Tato kapitola vám s těmito problémy příliš pomoci nemůže. Maximum, co pro vás mohu udělat, je poskytnout úvod k několika populárním GUI systémům, a nabídnout několik cenných rad a postřehů. Převážná část této kapitoly je věnována systémům Tk, GTK+, FOX a Qt. Ačkoliv se jedná o nejvíce rozšířené systémy, je docela slušná šance, že se zeptáte: "Proč v této kapitole nebyl popsán (sem vložte název vašeho oblíbeného GUI)?"

Důvodů může být několik. Jedním důvodem je omezený prostor, protože tato kniha primárně není o grafickém rozhraní. Dalším důvodem může být to, že váš oblíbený systém nemá natolik vyspělé

Ruby rozhraní, aby se dal použít. A posledním důvodem může být fakt, že ne všechny systémy uživatelského rozhraní jsou si rovny. Tato kapitola se tudíž snaží pokrýt pouze ty, které jsou nejdůležitější a nejvíce vyspělé. O zbytku se zmíníme pouze krátce.

12.1 – Ruby/Tk

Kořeny Tk sahají až do roku 1988 (pokud počítáme různé předběžně uvolněné verze). Dlouho bylo považováno za společníka k programovacímu jazyku Tcl, nicméně Tk se začalo používat i s několika dalšími jazyky, včetně Perlu a Ruby. Pokud by Ruby někdy mělo nějaké nativní GUI, pravděpodobně by se jednalo o Tk. V době psaní této knihy je velmi široce používáno, přičemž některé verze Ruby lze stáhnout včetně Tk.

Předchozí zmínka o Perlu není úplně bezdůvodná. Tk pro Ruby a Perl je dost podobné na to, aby materiál pro Perl/Tk mohl být z velké části použitelný i pro Ruby/Tk. Doporučujeme si sehnat zajímavou knihu *Learning Perl/Tk*, ISBN 1565923146 od Nancy Walsh.

12.1.1 – Přehled

V roce 2001 bylo Tk pravděpodobně nejvíce používaným GUI pro Ruby. Bylo první, které bylo dostupné a dlouho bylo i součástí standardní instalace Ruby. I když v současnosti už není tak populární jako předtím, je stále široce používáno. Někteří vývojáři říkají, že na Tk je patrný jeho věk – pro ty, co mají rádi čisté, objektově orientované, rozhraní, může být rozhraní Tk trochu zklamání. Ale má výhodu v popularitě, přenositelnosti a stabilitě.

Jakákoliv aplikace Ruby/Tk musí použít příkaz `require` pro načtení rozšíření `tk`. Poté je aplikační rozhraní sestavováno postupně, počínaje nějakým druhem kontejneru a ovládacích prvků, kterými je tento kontejner naplněn. Nakonec je provedeno volání `Tk.mainloop` (tato metoda zachytává všechny události – jako pohyb myši a stisky tlačítek – a zpracovává je).

```
require "tk"
# Sestavení aplikace...
Tk.mainloop
```

Stejně jako ve většině (nebo rovnou ve všech) okenních systémech jsou i zde ovládací prvky Tk nazývány *widgety*. Tyto *widgety* jsou obvykle dohromady seskupeny v kontejnerech. Kontejner na nejvyšší úrovni je nazýván kořen. Ačkoliv tento kořen není nutné specifikovat explicitně, rozhodně je to vhodné.

Každá třída *widgetu* je pojmenována na základě svého názvu ve světě Tk (připojením slova Tk na začátek). Proto *widget* `Frame` odpovídá třídě `TkFrame`.

Widgety jsou instanciovány prostřednictvím metody `new`. První parametr specifikuje kontejner, do kterého je *widget* umístěn. Pokud je vynechán, předpokládá se kořen.

Nastavení, které je použito pro instanciaci widgetu, může být specifikováno dvěma způsoby. První způsob (jako v Perlu) spočívá v předání haše s vlastnostmi a hodnotami. (Připomínáme, že se jedná o trik syntaxe Ruby – haš, který je vložen jako poslední nebo jako jediný parametr, může mít závorky vynechány.)

```
my_widget = TkSomewidget.new( "borderwidth" => 2, "height" => 40 ,  
                               "justify" => "center" )
```

Další způsob spočívá v předání bloku do konstruktoru, který bude vyhodnocen pomocí `instance_eval`. Uvnitř tohoto bloku můžeme volat metody pro nastavení vlastností widgetu (prostřednictvím metod, které mají stejné názvy jako vlastnosti). Mějte na paměti, že blok kódu je vyhodnocován v kontextu objektu, nikoliv volajícího. To například znamená to, že proměnné instance volajícího nemohou být odkazovány uvnitř tohoto bloku.

```
my_widget = TkSomewidget.new do  
  borderwidth 2  
  height 40  
  justify "center"  
end
```

V Tk jsou dostupní celkem tři správci rozložení – všichni slouží pro účely řízení relativní velikosti a umístění widgetu na obrazovce. První (a nejčastěji používaný) je `pack`. Další dva jsou `grid` a `place`: správce `grid` je sofistikovaný, ale poněkud náchylný k chybám; správce `place` je ze všech nejjednodušší, protože vyžaduje absolutní hodnoty pro umístění widgetu. Ve všech příkladech této kapitoly budeme používat pouze správce `pack`.

12.1.2 – Jednoduchá aplikace s okny

V této sekci si předvedeme nejjednodušší možnou aplikaci – jednoduchou aplikaci kalendáře, která zobrazí aktuální datum. Abychom se dostali do programovací nálady, začneme explicitním vytvořením `root` a umístěním widgetu `Label` dovnitř něj.

```
require "tk"  
  
root = TkRoot.new() { title "Today's Date" }  
str = Time.now.strftime("Today is \n%B %d, %Y")  
lab = TkLabel.new(root) do  
  text str  
  pack("padx" => 15, "pady" => 10, "side" => "top")  
end  
Tk.mainloop
```

V předchozím kódu vytváříme kořen, nastavujeme řetězec s datem a vytváříme popisek (label). Při vytváření popisku nastavujeme text, který je hodnotou `str` a voláme `pack`, který tohle všechno

úhledně uspořádá. Správci pack sdělujeme, aby nastavil výplň 15 pixelů horizontálně a 10 pixelů vertikálně, a žádáme, aby text byl centrován na střed. Obrázek 12.1 ukazuje vzhled aplikace.



Obrázek 12.1. Jednoduchá Tk aplikace.

Jak jsme už zmínili výše, vytvoření popisu (label) může vypadat takto:

```
lab = TkLabel.new(root) do
  text str
  pack("padx" => 15, "pady" => 10, "side" => "top")
end
```

Použitými jednotkami, které jsou v tomto příkladě specifikovány pro `padx` a `pady`, jsou standardně pixely. Pochopitelně, můžeme pracovat i v jiných jednotkách – k hodnotě stačí připojit požadovanou jednotku. Taková hodnota se sice nyní stává řetězcem, ale protože to Ruby/Tk vůbec nevádí, nevádí to ani nám. Dostupné jednotky jsou centimetry (c), milimetry (m), palce (i) a body (p). Podívejte se na následující příklad:

```
pack("padx" => "80m")
pack("padx" => "8c")
pack("padx" => "3i")
pack("padx" => "12p")
```

Atribut `side` v tomto případě vůbec nic neovlivňuje, protože jsme ji nastavili na výchozí hodnotu. Pokud změníte velikost okna aplikace, povšimnete si, že text se "přilepí" do horní části oblasti, ve které je umístěn. Jak asi tušíte, další možné hodnoty jsou `right`, `left` a `bottom`.

Metoda `pack` poskytuje několik dalších voleb, které řídí umístění widgetu na obrazovce:

- Volba `fill` specifikuje, zdali widget vyplňuje svůj přidělený obdélník (v horizontálním a/ nebo vertikálním směru). Možné hodnoty jsou `x`, `y`, `both` a `none` (výchozí je `none`).
- Volba `anchor` bude ukotvovat widget uvnitř přiděleného obdélníku na základě "kompasové" notace. Výchozí je `center`; další možné hodnoty jsou `n`, `s`, `e`, `w`, `ne`, `nw`, `se` a `sw`.
- Volba `in` zabaluje widget s ohledem na nějaký jiný kontejner, než je rodičovský. Výchozí je samozřejmě rodič.
- Volby `before` a `after` mohou být použity pro změnu pořadí widgetu. To je občas užitečné, protože widgety nemusí být vytvořeny (na rozdíl od jejich umístění na obrazovce) v nějakém konkrétním pořadí.

Celkově můžeme říci, že Tk je docela flexibilní z hlediska umístění widgetů na obrazovce. Prostudujte si dokumentaci pro další informace.

12.1.3 – Práce s tlačítky

Jedním z nejběžnějších widgetů v jakémkoliv GUI je stisknutelné tlačítko (nebo jednoduše tlačítko). Jak asi očekáváte, v aplikacích Ruby/Tk umožňuje použití tlačítek třída `TkButton`. V případě složitějších aplikací obvykle vytváříme rámce, které obsahují různé widgety, jež poté umísťujeme na obrazovku. V těchto kontejnerech mohou být umístěny i widgety ve formě tlačítka.

Pro tlačítko musíme specifikovat nejméně tři následující vlastnosti:

- Text tlačítka.
- Příkaz spojený s tlačítkem. Tento příkaz bude proveden, když je tlačítko stisknuto.
- Pozice tlačítka uvnitř jeho kontejneru.

Tady je malá ukázka:

```
btn_OK = TkButton.new do
  text "OK"
  command (proc { puts "The user says OK." })
  pack("side" => "left")
end
```

V tomto fragmentu kódu vytváříme nové tlačítko a přiřazujeme nový objekt do proměnné `btn_OK`. Do konstruktoru předáváme blok, ačkoliv bychom mohli použít i haš. V tomto případě používáme víceřádkovou formu (kterou osobně preferuji, ačkoliv v praxi můžete do jednoho řádku nacpat tolik kódu, kolik jenom chcete. Zapamatujte si, že blok je prováděn pomocí `instance_eval`, což znamená, že je vyhodnocen v kontextu objektu (v tomto případě nového objektu `TkButton`).

Text, který je specifikován ve formě parametru v metodě `text`, bude umístěn na samotné tlačítko. Může to být několik slov nebo dokonce i několik řádků. Použití metody `pack` už jsme viděli. Není zajímavá, ačkoliv je nepostradatelná, pokud má být widget vůbec viditelný. Zajímavou částí kódu je metoda `command`, která přijímá objekt `Proc` a připojuje ho k tlačítku. V této kapitole budeme často používat metodu `lambdaproc` z modulu `Kernel`, která zkonvertuje blok na objekt `Proc`.

Akce, kterou zde provádíme, je ovšem dosti hloupá. Když uživatel stiskne tlačítko, bude provedeno negrafické `puts`. To znamená, že výstup půjde do okna příkazového řádku, ze kterého byl spuštěn program (nebo do pomocného okna konzoly).

Nyní vám nabídneme lepší příklad. Výpis 12.1 ukazuje aplikaci termostatu, která bude zvyšovat nebo snižovat zobrazenou hodnotu (čímž nám poskytne iluzi toho, že můžeme ovládat teplotu v místnosti). Vysvětlení následuje, jako vždy, až za výpisem.

Výpis 12.1. Simulovaný termostat.

```
require 'tk'

# Běžné nastavení...
Top = { 'side' => 'top', 'padx'=>5, 'pady'=>5 }
```

```

Left = { 'side' => 'left', 'padx'=>5, 'pady'=>5 }
Bottom = { 'side' => 'bottom', 'padx'=>5, 'pady'=>5 }

temp = 74 # Počáteční teplota...

root = TkRoot.new { title "Thermostat" }

top = TkFrame.new(root) { background "#606060" }
bottom = TkFrame.new(root)

tlab = TkLabel.new(top) do
  text temp.to_s
  font "{Arial} 54 {bold}"
  foreground "green"
  background "#606060"
  pack Left
end

TkLabel.new(top) do
  text "o"
  font "{Arial} 14 {bold}"
  foreground "green"
  background "#606060"
  # Přidat anchor-north do hash
  pack Left.update({ 'anchor' => 'n' })
end

TkButton.new(bottom) do
  text " Up "
  command proc { tlab.configure("text"=>(temp+=1).to_s) }
  pack Left
end

TkButton.new(bottom) do
  text "Down"
  command proc { tlab.configure("text"=>(temp-=1).to_s) }
  pack Left
end

top.pack Top
bottom.pack Bottom
Tk.mainloop

```

Pomocí toho kódu vytváříme dva rámce. Horní rámec ovládá pouze zobrazení. Teplotu ve Fahrenheitech zobrazujeme prostřednictvím velkého písma (pro znak stupně používáme malé, strategicky umístěné, písmenko "o"). Spodní rámec je pak použit pro tlačítka "nahoru" a "dolů".

Povšimněte si, že používáme několik nových vlastností pro objekt `TkLabel`. Metoda `font` specifikuje písmo a velikost textu. Hodnota řetězce je závislá na platformě. Ta, která je ukázána zde, je platná pro systém Windows. V systému Unix by se obvykle jednalo o dlouhý a těžkopádný název fontu ve stylu `-Adobe-Helvetica-Bold-R-Normal-*-*120-*-*-*-*-*`.

Metoda `foreground` nastavuje barvu textu. V našem příkladu vkládáme řetězec "green", který má v útrokách Tk předem definovaný význam. (Pokud chcete zjistit, zdali je v Tk předdefinována nějaká barva, nejsnazší cestou je to vyzkoušet). Barva pozadí se nastavuje pomocí metody `background`. V tomto případě specifikujeme barvu odlišně, protože ji chceme v typickém hexadecimálním formátu (řetězec `"#606060"` reprezentuje pěknou šedou barvu).

Abychom nekomplikovali pěkný jednoduchý design, nepřidali jsme do něj žádné tlačítko "Konec". Aplikaci můžete jako obvykle zavřít kliknutím na ikonu Close v pravém horním rohu okna.

V příkazech pro tlačítka je použita metoda `configure`, kterou se mění obsah horního popisku při zvyšování nebo snižování aktuální teploty. Jak už jsem zmínil dříve, tímto způsobem může být za běhu změněna jakákoliv vlastnost (příčemž její efekt se okamžitě projeví na monitoru).

Nyní zmíníme dva další triky, které můžete s textovými tlačítky provést. Metoda `justify` přijímá parametr (`left`, `right` nebo `center`), kterým můžete specifikovat umístění textu na tlačítku (výchozí hodnota je `center`). Tlačítko může obsahovat i několik řádků textu – metoda `wraplength` specifikuje sloupec, ve kterém dojde k zalomení textu.

Styl tlačítka může být změněn prostřednictvím metody `relief`, která mu dá 3D vzhled. Parametrem této metody musí být jeden z těchto řetězců: `flat`, `groove`, `raised`, `ridge` (výchozí), `sunken` nebo `solid`. Metody `width` a `height` explicitně nastavují velikost tlačítka. Je také dostupná metoda `borderwidth`. Pro další volby – kterých je opravdu mnoho – se podívejte do dokumentace.

Pojďme se nyní podívat na některé další příklady. Naše nová tlačítka na sobě nebudou mít text, ale pouze obrázek. Pro tyto účely jsem předem vytvořil dvojici obrázků ve formátu GIF, které obsahují šipky nahoru a dolů. (Tyto grafické soubory jsou pojmenovány vskutku intuitivně – `up.gif` a `down.gif`.) Pro získání reference na každý z nich můžeme použít třídu `TkPhotoImage`. Poté můžeme tyto reference použít při instanciaci tlačítek.

```
up_img = TkPhotoImage.new("file"=>"up.gif")
```

```
down_img = TkPhotoImage.new("file"=>"down.gif")
```

```
TkButton.new(bottom) do
```

```
  image up_img
```

```
  command proc { tlab.configure("text"=>(temp+=1).to_s) }
```

```
  pack Left
```

```
end
```

```

TkButton.new(bottom) do
  image down_img
  command proc { tlab.configure("text"=>(temp-=1).to_s) }
  pack Left
end

```

Tímto kódem jednoduše nahradíte odpovídající řádky v našem prvním příkladu s termostatem. S výjimkou změněných tlačítek je všechno stejné. Obrázek 12.2 ukazuje aplikaci termostatu.



Obrázek 12.2. Simulace termostatu (s grafickými tlačítky).

12.1.4 – Práce s textovými poli

Vstupní textové pole může být zobrazeno a spravováno použitím widgetu `TkEntry`. Jak asi očekáváte, pro ovládání velikosti, barvy a chování tohoto widgetu je dostupných mnoho voleb; my vám nabídneme jeden rozsáhlý příklad, který ilustruje několik z nich. Vstupní pole je užitečné pouze v případě, kdy z něj dokážeme získat hodnotu, která byla do něj vložena. Pole bude vázáno na proměnnou `TkVariable` (jak uvidíme později sami), ačkoliv může být použita i metoda `get`.

Předpokládejme, že chceme vytvořit telnet klienta, který bude přijímat čtyři informace – hosta, číslo portu (standardně 23), uživatelské ID a heslo. Přidáme také dvě tlačítka pro operaci "přihlášení" a "zrušení". Následující kus kódu provádí nějaké triky s rámci pro seřazení věcí a pro jejich lepší vzhled. Kód není napsán přenositelným způsobem a skutečný Tk guru by tento přístup odsoudil. Ale protože se jedná pouze o ilustrační ukázkou, tato skutečnost nás vůbec netrápí. Na obrázku 12.3 je zobrazena již dokončená aplikace. Její kód je pak uveden ve výpisu 12.2.



Obrázek 12.3. Simulovaný telnet klient.