

Tvorba interaktivních dokumentů v Maple

Ing. Vladimír Žák

*Ústav matematiky, Fakulta strojního inženýrství, Vysoké učení technické v Brně,
Technická 2, 616 69 Brno
e-mail: zakyn@centrum.cz*

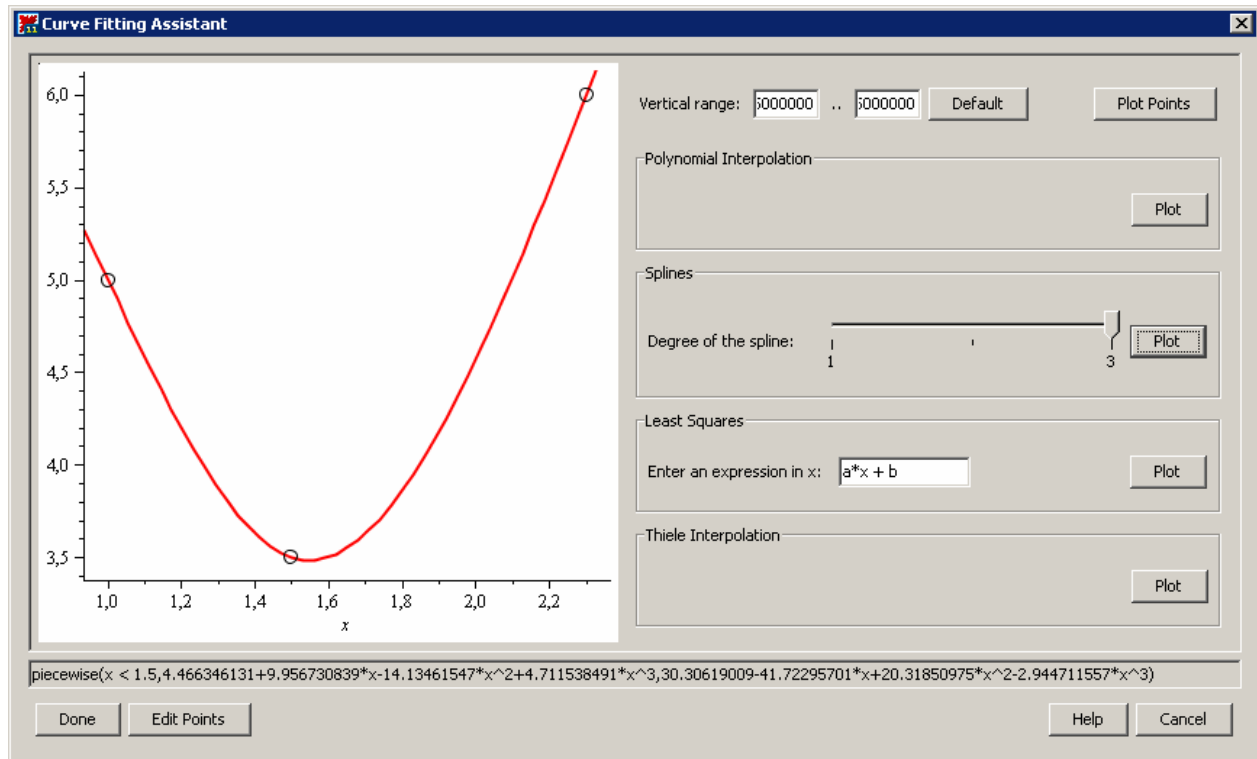
Abstrakt

Příspěvek se zabývá novými možnostmi tvorby interaktivních dokumentů v systému Maple. Využívá se tzv. *vložených komponent*, které umožňují vytvářet uživatelsky velmi přívětivé a plně interaktivní dokumenty. Takto vytvořené soubory lze užít nejen pro výuku, ale i pro tvorbu technologické dokumentace, či je použít jako simulace při modelování. Článek přehledně popisuje základní vložené komponenty a jejich možné využití na názorných příkladech.

1. Úvod

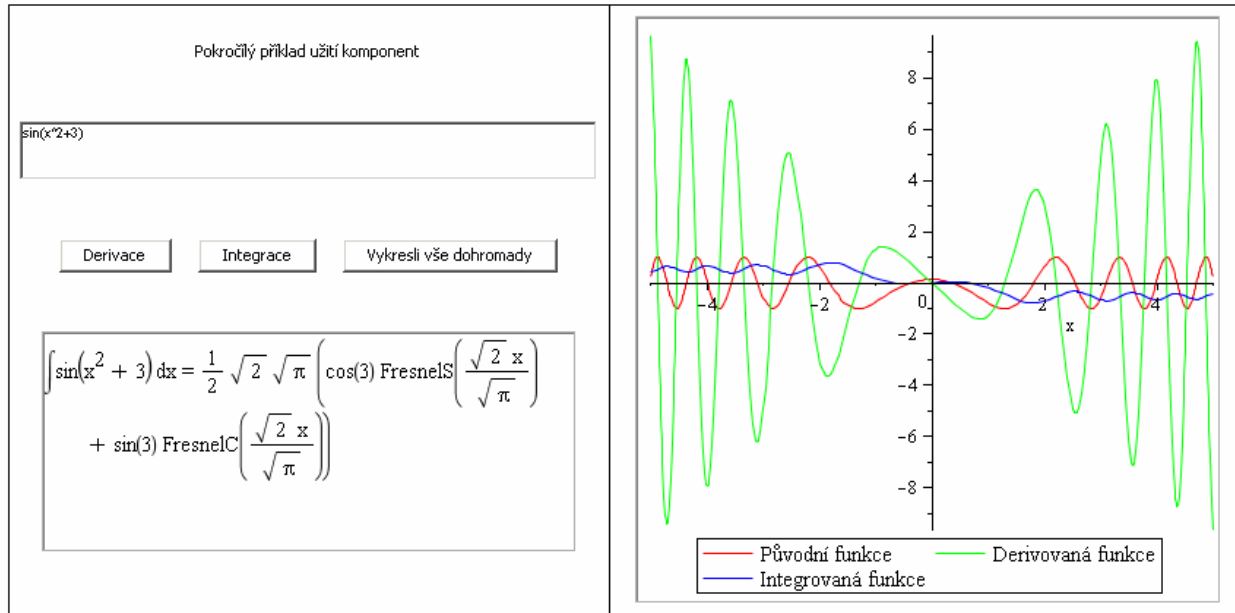
Matematický systém Maple nabízí ve své jedenácté verzi rozšíření možností tvorby interaktivních dokumentů, které se využívají zejména pro názorné ukázky řešení problémů. Verze Maple 10 poprvé představila tzv. vložené interaktivní komponenty, které se spolu s technologií Maplet podílejí právě na interaktivně systému Maple. Tyto dvě technologie jsou od sebe velmi odlišné, ale navzájem se doplňují v jeden celek, který umožňuje, aby byla práce v systému Maple pro uživatele velmi přívětivá a intuitivní.

Technologie Maplet je založena na jazyce Java, pomocí kterého je zobrazován dialog (aplikaci) s funkčními prvky. Tato aplikace může být spuštěna přímo z dokumentu systému Maple (v té chvíli je systém Maple nedostupný a čeká na ukončení Mapletu) a nebo jako samostatná aplikace. V druhém případě ke svému běhu potřebuje nainstalovaný Maple na daném počítači, protože výpočty stále provádí systém Maple a Maplet aplikace je jen uživatelským rozhraním pro výpočty. Příkladem Mapletu může být např. Curve Fitting Assistent vyobrazený na obrázku 1. Ještě poznamenejme, že technologii Mapletů využívá např. i MapleNet, který umožňuje distribuovat Maplety pomocí webového rozhraní a Java runtimeu klientského počítače.



Obrázek 1 - Curve Fitting Assistant

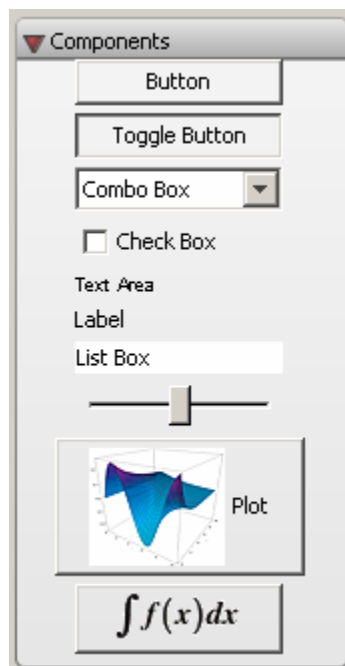
Technologie vložených komponent má trochu jinou filozofii. Umožňuje začlenit interaktivní prvky přímo do dokumentů systému Maple. Tento typ dokumentu se nazývá Rich Technical Document a umožňuje uživateli vytvářet plně interaktivní technické dokumenty ať už vytvořené tradičním způsobem a nebo pomocí vložených interaktivní komponent. Obrázek 2 ukazuje vložené interaktivní komponenty v dokumentu systému Maple.



Obrázek 2 - vložené interaktivní komponenty v dokumentu Maple



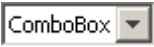


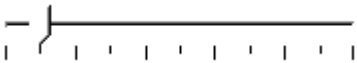
2. Přehled interaktivních komponent

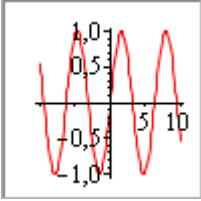

Interaktivní komponenty systému Maple se nacházejí na paletě nástrojů Components (obrázek 3). Použití je velmi jednoduché a intuitivní. Pomocí přetažení lze danou komponentu umístit na požadované místo v Maple dokumentu. V této chvíli je nutné poznamenat, že každá takto vytvořená komponenta po umístění do dokumentu získá své jedinečné jméno v rámci daného dokumentu (lze jej později změnit na jiné).



Obrázek 3 – paleta interaktivních komponent

Následující tabulka uvádí přehled vložených komponent spolu s jejich názvem, popisem, ukázkou a příkazem pro získání nápovědy k příslušné komponentě.

Název	Název komponenty	Komponenta	Nápověda
Tlačítko	Button		?buttonComponent
Výběrové tlačítko	Toggle Button		?togglebutton
Rozbalovací seznam	Combo Box		?ComboBox
Zaškrťovací políčko	Check Box	<input type="checkbox"/> CheckBox	?checkbox
Text	Text Area		?textarea
Popisek	Label	Label	?label
Seznam	List Box		?listbox
Posuvník	Slider		?slider
Graf	Plot		?plotComponent

			
Matematický zápis	Mathematical Expression		?MathExpressionComponent

Tabulka 1- přehled vložených komponent

Následující kapitoly se zabývají jednotlivými komponentami a jejich programováním.

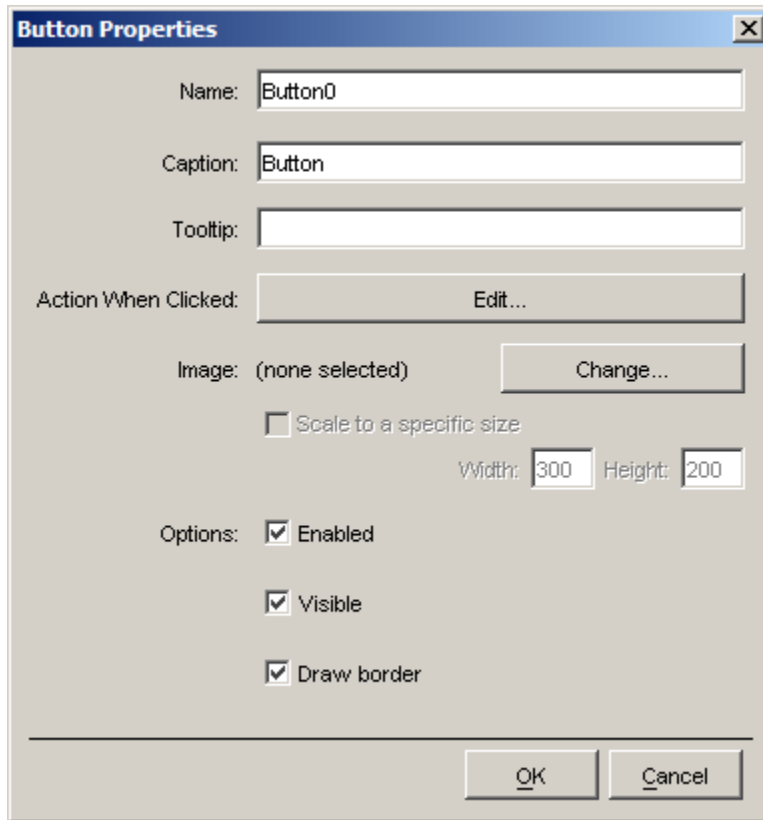
3. Tlačítko – Button

Jednou ze základních komponent je tlačítko. Slouží zejména k provedení určité operace.

Postup práce je následující:

- vložíme komponentu Button z palety Components
- klikneme pravým tlačítkem na komponentu

vybereme položku Component **Properties**



- všechny položky jsou zřejmé již z popisek


Pomocí tlačítka *Edit* lze vkládat zdrojový kód, který ovládá dané operace.

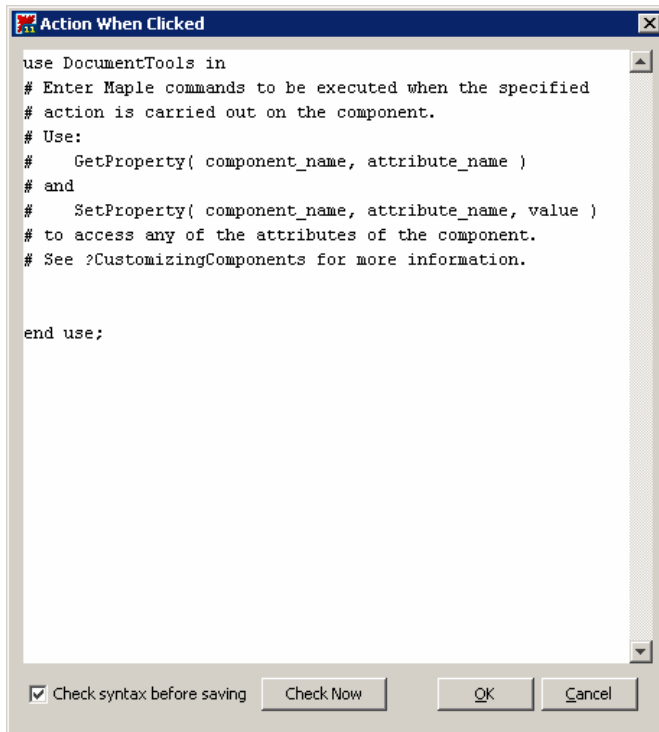
Následující příklad ukáže, jak je možné pracovat s komponentou *Button*. Vytvoříme tlačítko, pomocí kterého spočítáme primitivní funkci z výrazu obsaženém v proměnné *il*.

```
> il:=x^2;
```

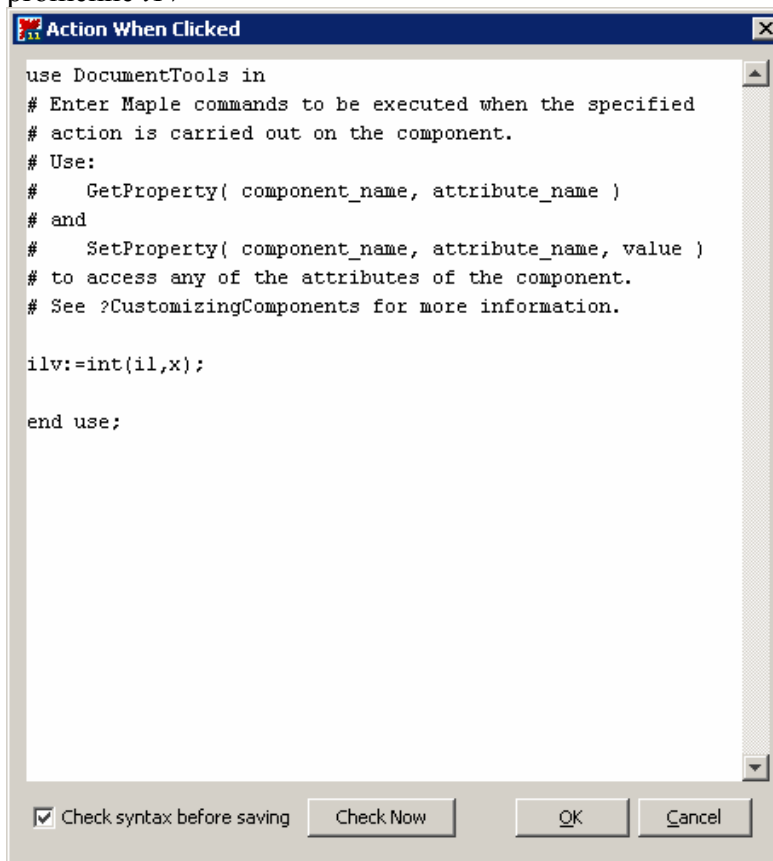
```
il :=x2
```

Postup:

- vložíme komponentu Button 
- kontextová nabídka -> Component Properties
- stiskneme Edit a obdržíme



- do dialogu vepíšeme zdrojový kód pro integraci výrazu il podle x a uložíme výsledek do proměnné ilv



- stiskneme OK

- stiskneme OK
- stiskneme tlačítko
- nakonec otestujeme, zda byla integrace provedena pomocí příkazu
`> ilv;`

$$\frac{1}{3} x^3$$

Dialog *Action When Clicked* obsahuje v poznámkách informace o možnosti práce s komponentami. Jde o volání funkcí *GetProperty* popř. *SetProperty* z knihovny *DocumentTools* pro získání popř. nastavení vlastnosti komponenty. Více informací lze získat v nápovědě pomocí:

```
> ?DocumentTools
> ?DocumentTools,GetProperty
> ?DocumentTools,SetProperty
```

Syntaxe je následující:

```
prom = GetProperty('identifikátor','vlastnost');
SetProperty('identifikátor','vlastnost',hodnota);
```

Pro úplnost je třeba uvést, že Maple 11 podporuje ještě funkci *Do* z téže knihovny, pro nastavování vlastností jednotlivých komponent. Syntaxe je následující a odpovídá předchozím funkcím:

```
prom = Do( %identifikátor(vlasnost) )
Do( %identifikátor(vlasnost) = hodnota )
```

Každá komponenta má určité vlastnosti. Pro názornost uvedme vlastnosti komponenty Button.

Název vlastnosti = hodnota	Popis
caption = string	Popisek, který je zobrazen na tlačítku
enabled = true or false	Určuje, zda je dostupný a nebo ne
image = name or string	Určuje zobrazení obrázku na tlačítku
pixelHeight = posint	Výška obrázku v pixelech
pixelWidth = posint	Šířka obrázku v pixelech
showBorders = true or false	Určuje, zda jsou zobrazeny okraje
tooltip = string	Zobrazuje tooltip u tlačítka
useSpecifiedSize = true or false	Určuje, zda je použita specifická velikost (viz help)
visible = true or false	Určuje, zda je komponenta viditelná a nebo ne

Tabulka 2 - vlastností komponenty Button

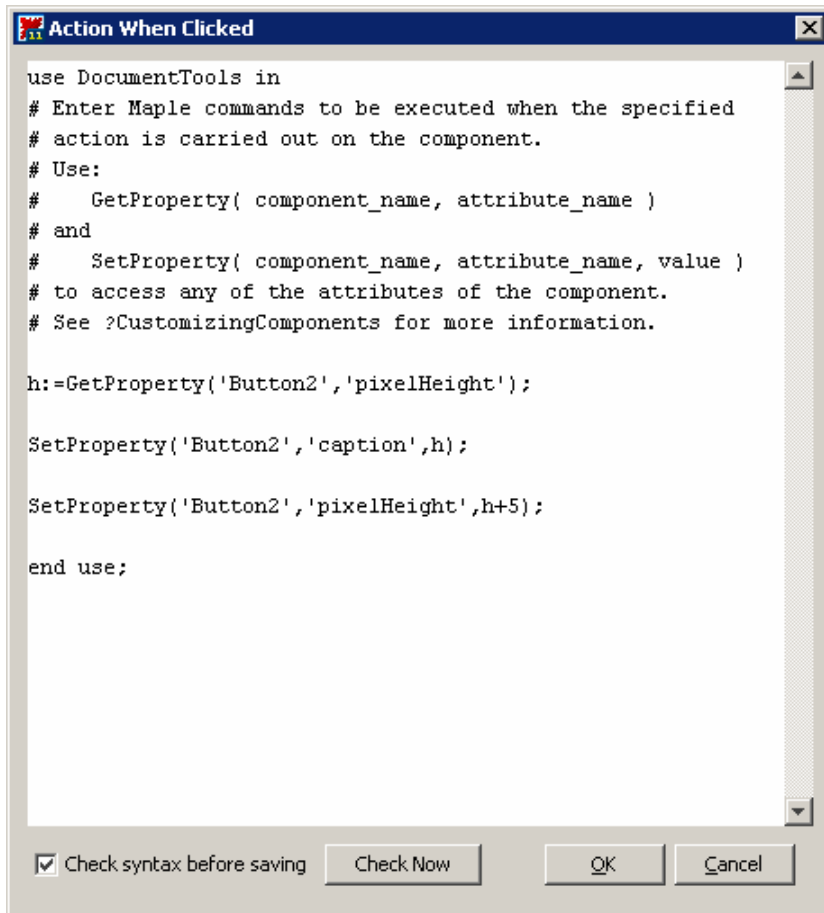
Na dalším příkladu ukážeme, jakým způsobem lze změnit komponentu Button. Změníme velikost tlačítka při jeho stisknutí.

Postup:

- vložíme tlačítko s obrázkem



- napíšeme kód pro změnu velikosti tlačítka



- stiskneme několikrát tlačítko a výška obrázku se bude vždy o 5 pixelů zvětšovat.

4. Rozbalovací seznam ComboBox a ListBox

Komponenty *ComboBox* a *ListBox* jsou si velmi podobné. Rozdíl je jen v zobrazení jednotlivých položek. Uveďme seznam vlastností.

caption = string	Nadpis komponenty
enabled = true or false	Udává, zda je komponenta dostupná
itemlist = symbol (combobox) itemList = list (listbox)	Obsahuje seznam všech položek v množině a nebo seznamu.
tooltip = string	Nastavuje tooltip
value = string	Udává vybranou hodnotu
visible = true or false	Udává, zda je komponenta viditelná

Tabulka 3 - Přehled vlastností obou komponent

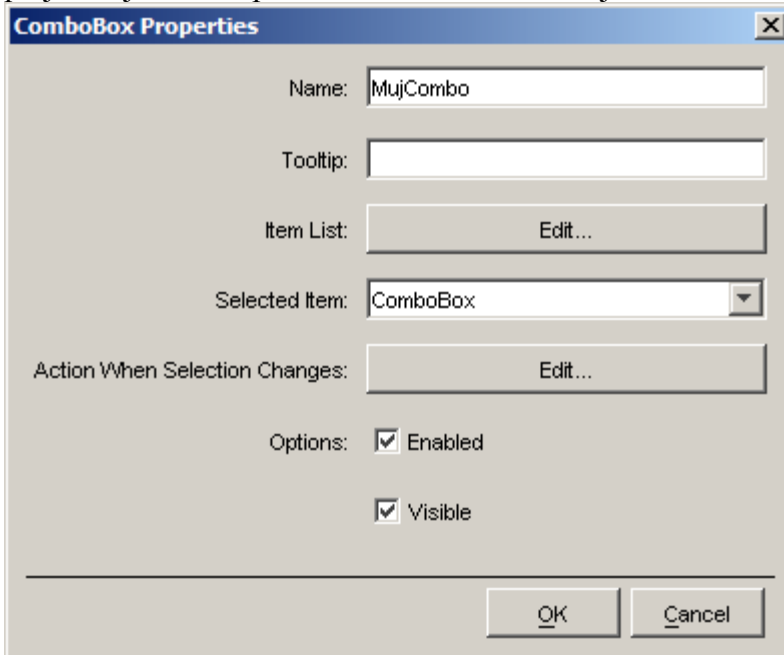
V ukázkovém příkladu budeme přesouvat hodnoty vybrané položky z komboboxu a seznamu a spojíme je do jednoho řetězce v textovém poli.

Postup:

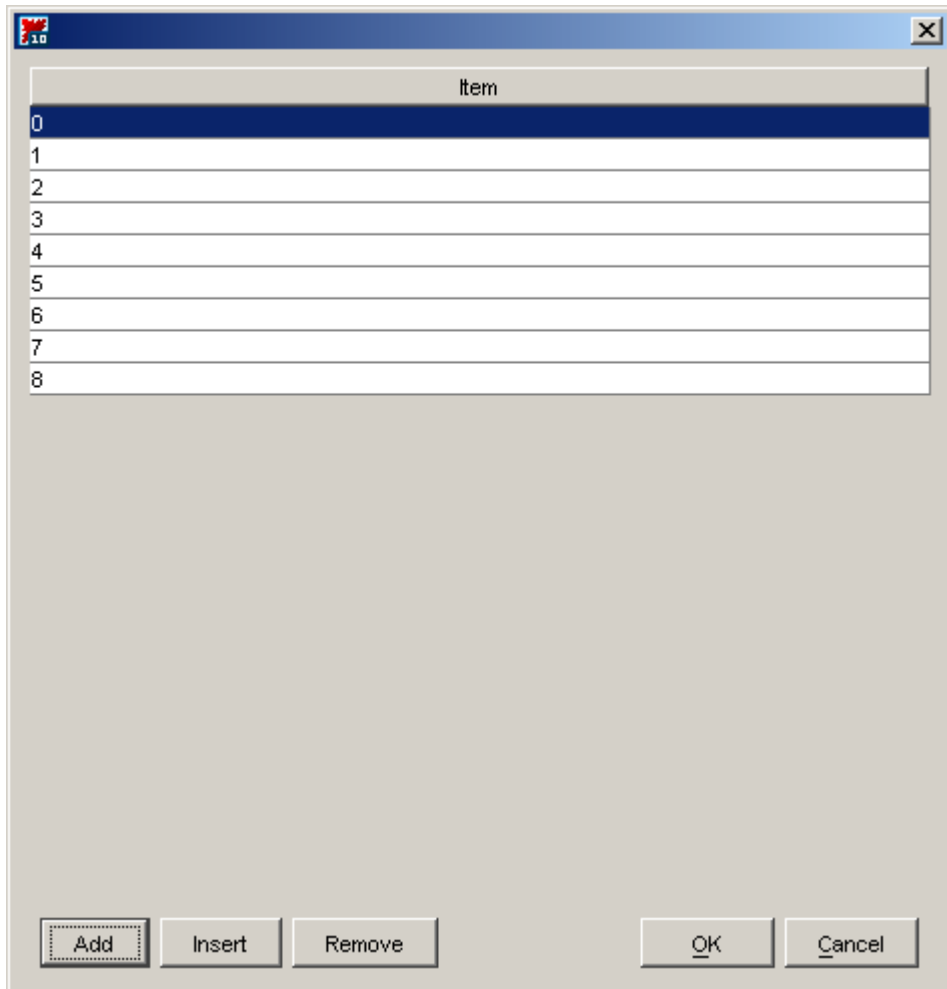
- vložíme ComboBox, ListBox, TextArea a Button



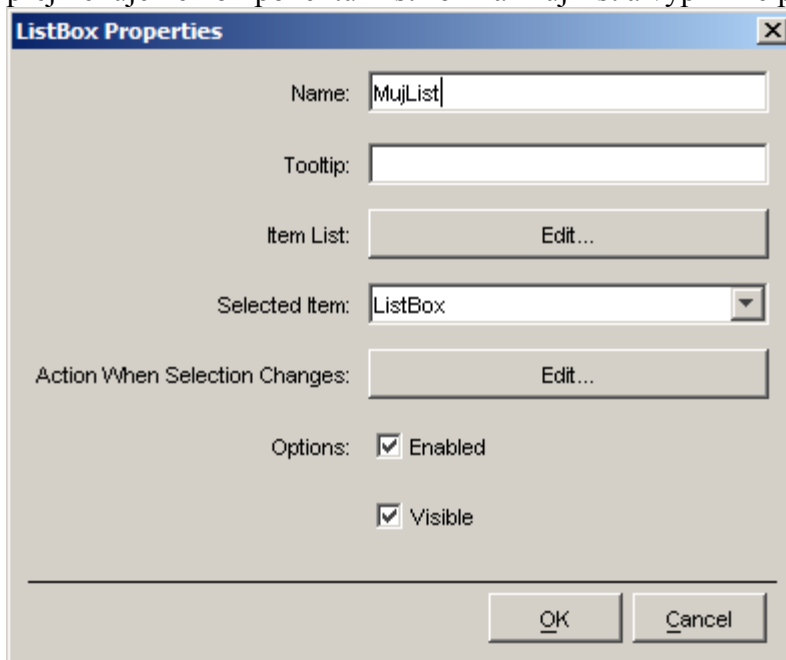
- přejmenujeme komponentu ComboBox na MujCombo



- přidáme do MujBox několik položek pomocí tlačítka Edit u položky Item List

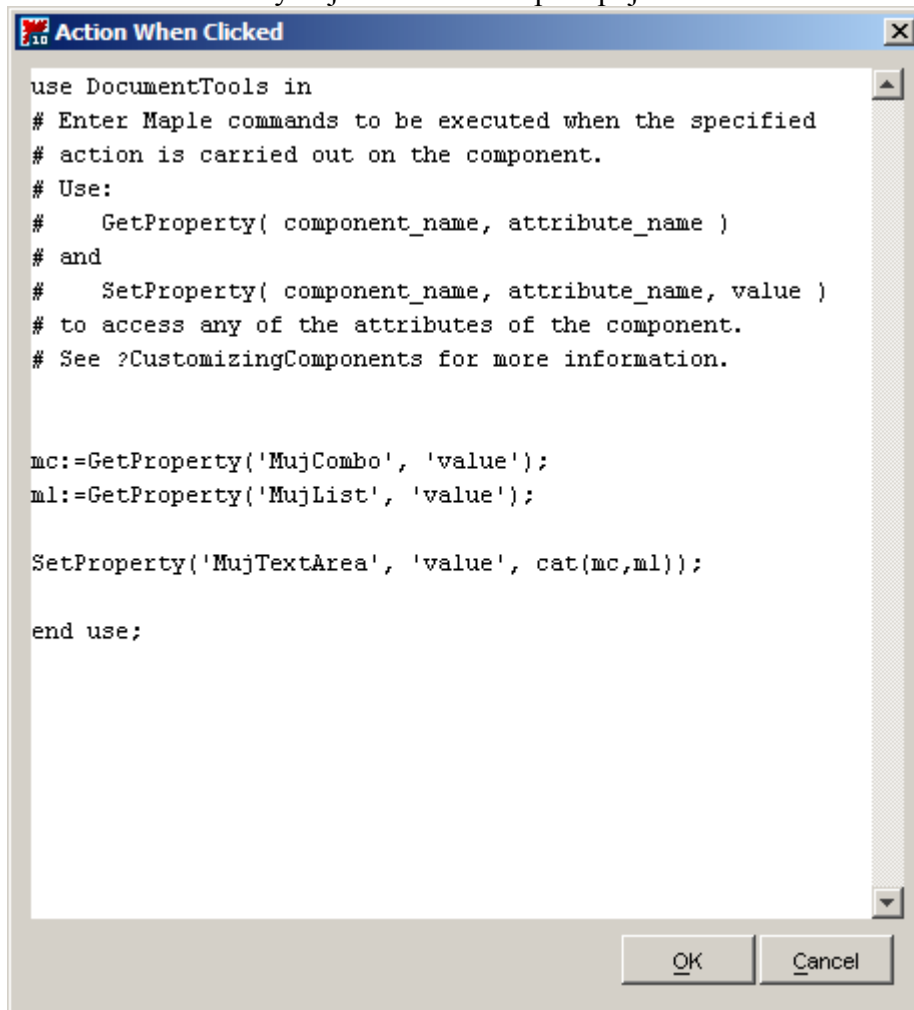


- přejmenujeme komponentu ListBox na MujList a vyplníme položky seznamu



- pomocí tlačítka budeme přesouvat hodnoty vybrané položky z MujCombo a z MujList do

TextArea – k tomu využijeme funkci `cat` pro spojení obou řetězců dohromady



```
use DocumentTools in
# Enter Maple commands to be executed when the specified
# action is carried out on the component.
# Use:
#   GetProperty( component_name, attribute_name )
# and
#   SetProperty( component_name, attribute_name, value )
# to access any of the attributes of the component.
# See ?CustomizingComponents for more information.

mc:=GetProperty('MujCombo', 'value');
ml:=GetProperty('MujList', 'value');

SetProperty('MujTextArea', 'value', cat(mc,ml));

end use;
```

- vyzkoušíme funkčnost

Ještě poznamenejme, že v tomto případě lze užít zkráceného zápisu pomocí funkce `Do` (dostupné od Maple 11) následovně:

```
> with(DocumentTools);
                               [Do, GetProperty, Retrieve, SetProperty]

> mc:=Do(%MujCombo);
   ml:=Do(%MujList);
                               mc :=8
                               ml :=a

> Do( %MujTextArea = cat(mc,ml) );
                               8a
```

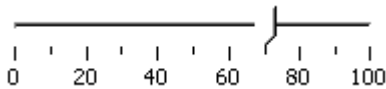
Jak je vidět z předchozích několika příkazů, je možné komponenty a jejich vlastnosti měnit také přímo v dokumentu systému Maple.

5. Posuvník a graf – Slider a Plot

Vysvětlení těchto dvou komponent spojíme do jedné kapitoly, neboť jejich spojení velmi efektní a efektivní. Podívejme se nejprve na jednotlivé komponenty.

Slider

Posuvník se s výhodou použije při stanovování hodnoty nějakého parametru.



Následující tabulka uvádí dostupné vlastnosti dané komponenty

caption = string	Jen pro čtení - Slider
enabled = true or false	Udává, zda je komponenta aktivní.
filled = true or false	Udává, zda jsou zobrazeny body. Defaultní je true
lower = int	Nejnižší hodnota. Defaultní je 0.
majorTicks = posint	Interval mezi hlavními čárkami. Defaultní je 20.
minorTicks = posint	Interval mezi vedlejšími čárkami. Defaultní je 10.
showLabels = true or false	Indikuje, zda jsou zobrazeny popisky.
showTicks = true or false	Zobrazuje čárky.
snapToTicks = true or false	Indikuje, zda má jezdec přiskakovat k hodnotám značeným čárkami.
tooltip = string	Nápověda ke komponentě.
upper = int	Horní hranice.
value = posint	Aktuální hodnota.
vertical = true or false	Indikuje, zda je posuvník zobrazen svisle a nebo vodorovně.
visible = true or false	Indikuje, zda je komponenta viditelná.

Tabulka 4 - Přehled vlastností komponenty Slider

Je nutné poznamenat, že některé vlastnosti jsou jen pro čtení a nebo pro zápis. Informace jsou dostupné v nápovědě k dané komponentě.

Plot

Komponenta Plot je určena pro zobrazování grafických výstupů. Následující tabulka uvádí dostupné vlastnosti. Informace lze nalézt v nápovědě.

continuous = true or false	Indikuje, zda je animace přehrávána kontinuálně a nebo po snímcích.
delay = posint	Časová mezera v milisekundách mezi jednotlivými snímky. Defaultní hodnota je 100.
frame = posint	Aktuálně zobrazený snímek.
frameCount = posint	Počet zobrazených snímků.
frameBackwards = true or false	Pokud je true, je zobrazen předchozí snímek sekvence. Pokud je animace přehrávána, tak se zastaví.
frameForwards = true or false	Pokud je true, je zobrazen následující snímek sekvence. Pokud je animace přehrávána, tak se zastaví.
pause = true or false	Nastavením true se animace zastaví.
pixelHeight = posint	Výška obrázku v pixelech. Defaultní je 400px.
pixelWidth = posint	Šířka obrázku v pixelech. Defaultní je 400px.
play = true or false	Nastavení na true začne přehrávat animaci nebo ji pozastaví (pauza).
`stop` = true or false	Nastavením true je animace zastavena. Protože (stop) je také klíčové slovo, je nutné uzavřít ho do uvozovek (``)
toEnd = true or false	Nastavením na true se animace přesune na konec. V případě přehrávání animace se zastaví.
toStart = true or false	Nastavením na true se animace přesune na začátek. V případě přehrávání animace se zastaví.
value = plot command	Hodnota, která má být zobrazena (musí to být PLOT struktura), tj. příkazy plot, plot3d nebo nějaká z vykreslovacích funkcí popř. uživatelem definovaná struktura PLOT nebo PLOT3D
visible = true or false	Udává, zda je komponenta viditelná.

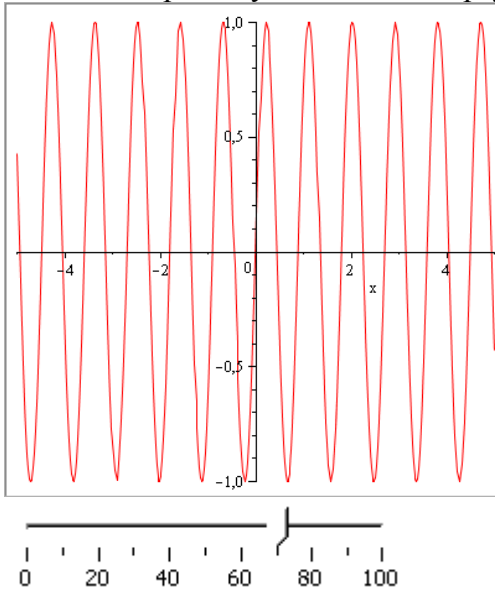
Tabulka 5 - Přehled vlastností komponenty Plot

Nyní se uvedeme velmi efektní aplikaci těchto dvou komponent. Aplikace bude vykreslovat

křivku v závislosti na daném parametru. Příklad poté rozšíříme pomocí komponent Text Area a MathExpression.

Postup:

- vložíme komponenty Slider a Plot a pojmenujme je jako Slider_plot a Plot_slider



- pomocí tlačítka Edit z dialogu komponenty Slider_plot vepíšeme požadovanou funkci

```
10 Action When Value Changes
use DocumentTools in
# Enter Maple commands to be executed when the specified
# action is carried out on the component.
# Use:
#   SetProperty( component_name, attribute_name )
# and
#   SetProperty( component_name, attribute_name, value )
# to access any of the attributes of the component.
# See ?CustomizingComponents for more information.

h:=GetProperty('Slider_plot','value')/10;

SetProperty('Plot_slider','value',plot(sin(h*x),x=-5..5));

end use;
```

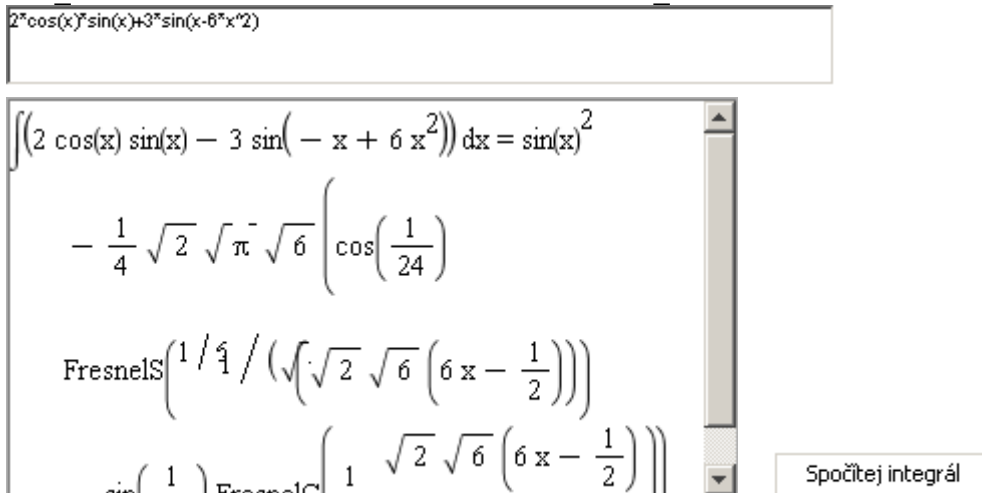
- pomocí jezdcy se nyní mění grafika v komponentě Plot_slider

6. Label, TextArea a Mathematical Expression

Všechny tyto tři komponenty jsou určeny pro zobrazování určitých textů. Komponenta Label se hodí k velmi jednoduchým aplikacím a zobrazování popisek apod. Komponenta TextArea je určena k zadávání vstupů od uživatele. Poslední komponenta MathExpression je určena pro zobrazování matematických výstupů, které jsou vnitřně uloženy ve formátu MathML.

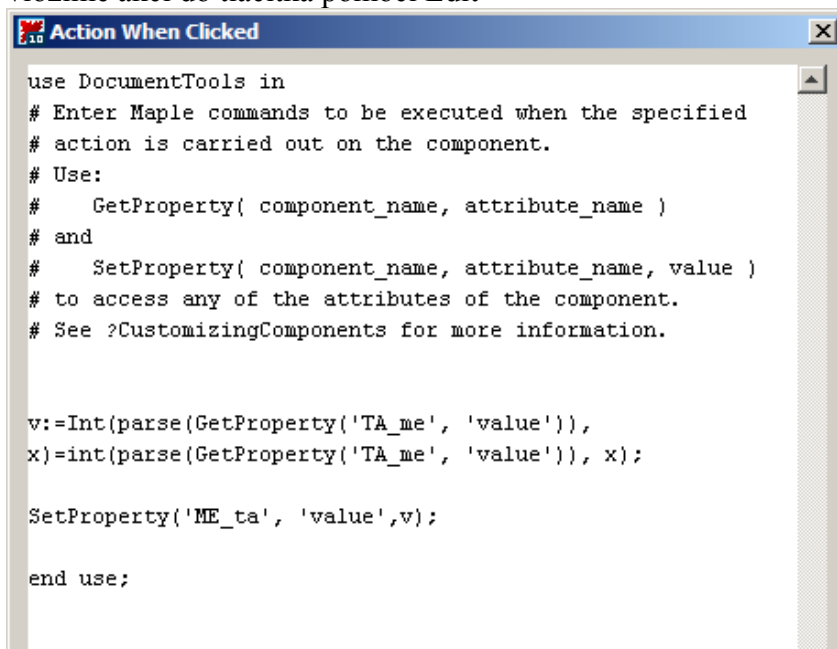
Postup:

- vložíme komponenty TextArea a MathExpression a přejmenujeme je na TA_me a ME_ta. Nakonec vložíme tlačítko s názvem Button_TAME.



The screenshot shows a Maple worksheet with a text input field containing the expression $2^2 \cos(x)^2 \sin(x) + 3 \sin(x - 6x^2)$. Below it, a MathExpression component displays the integral result: $\int (2 \cos(x) \sin(x) - 3 \sin(-x + 6x^2)) dx = \sin(x)^2 - \frac{1}{4} \sqrt{2} \sqrt{\pi} \sqrt{6} \left(\cos\left(\frac{1}{24}\right) \text{FresnelS}\left(\frac{1}{\sqrt{6}} / \left(\sqrt{\sqrt{2} \sqrt{6} \left(6x - \frac{1}{2}\right)}\right)\right) - \sin\left(\frac{1}{24}\right) \text{FresnelC}\left[\frac{1}{\sqrt{2} \sqrt{6} \left(6x - \frac{1}{2}\right)}\right]\right)$. A button labeled "Spočítej integrál" is located to the right of the MathExpression component.

- změníme názvy komponent
TextArea ~ TA_me
Button ~ Button_TAME
MathContainer ~ ME_ta
- vložíme akci do tlačítka pomocí Edit



```
use DocumentTools in
# Enter Maple commands to be executed when the specified
# action is carried out on the component.
# Use:
#   GetProperty( component_name, attribute_name )
# and
#   SetProperty( component_name, attribute_name, value )
# to access any of the attributes of the component.
# See ?CustomizingComponents for more information.

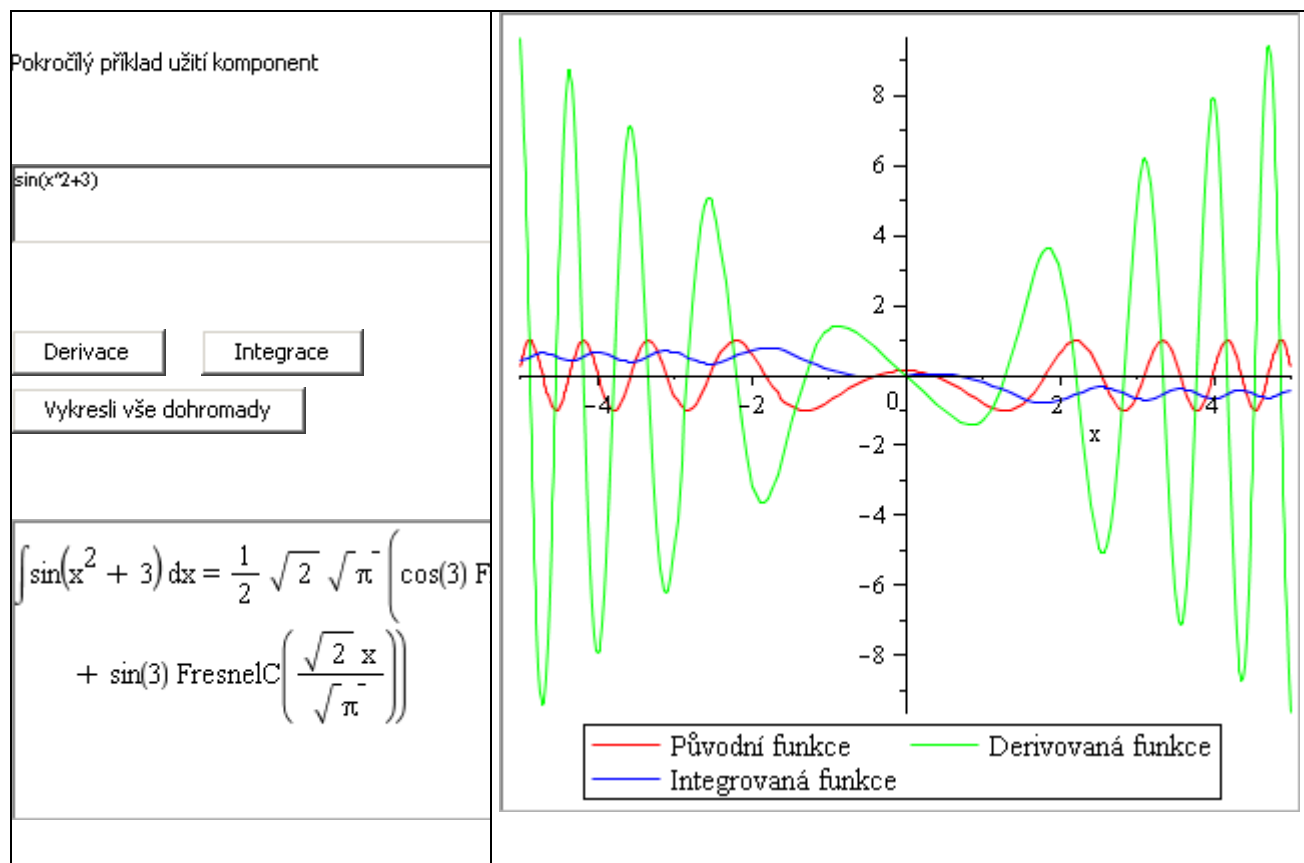
v:=Int(parse(GetProperty('TA_me', 'value')),
x)=int(parse(GetProperty('TA_me', 'value')), x);

SetProperty('ME_ta', 'value',v);

end use;
```


Pokud jde o ostatní komponenty, práce s nimi a jejich vlastnosti jsou velmi podobné s těmi probranými.

Následující příklad je ukázkou užití výše probraných komponent jako celku. Není zde vysvětlen postup práce, neboť je shodný s výše uvedenými postupy. Na konkrétní zdrojový kód jednotlivých komponent se podívejte pomocí Dialogu vlastností.



7. Závěr

V článku jsme na jednoduchých příkladech ukázali možnosti technologie vložených komponent v systému Maple. Je zřejmé, že s velmi malou námahou je možné vytvořit plně interaktivní a uživatelsky velmi přívětivé aplikace, jejichž výpočtová síla je schována v pozadí těchto komponent. Výhoda tohoto přístupu, na rozdíl od technologie Mapletů, je v tom, že je interaktivita obsažena přímo v dokumentu systému Maple a je tedy možné velmi jednoduše dané aplikace rozšiřovat.

Reference

- [1] Nápověda systému Maple 11
- [2] www.vladimirzak.com/maple
- [3] www.mapleprimes.com