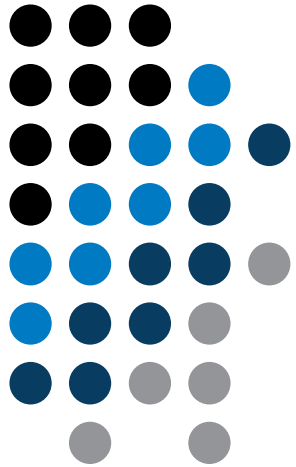
A0B17MTB – Matlab

Část #10



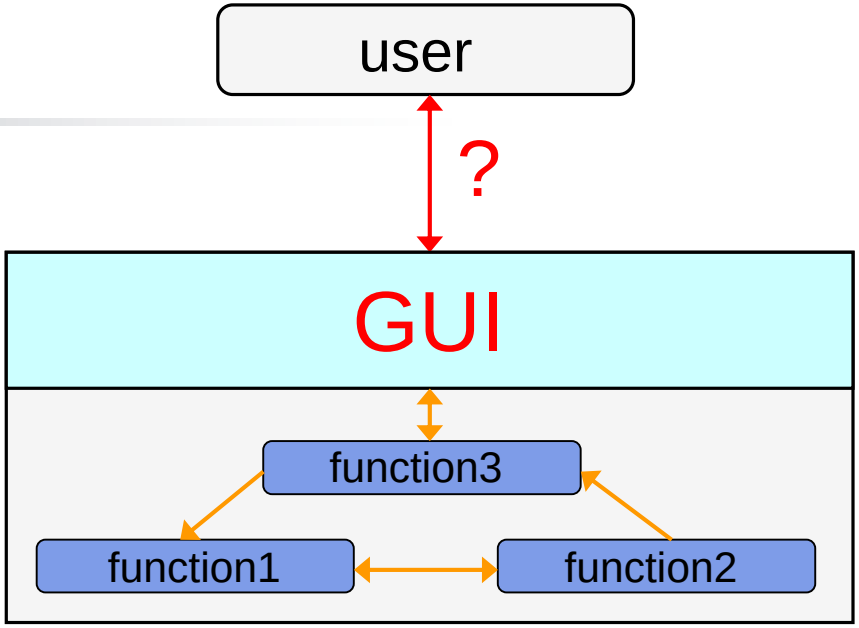
Miloslav Čapek
miloslav.capek@fel.cvut.cz

Filip Kozák a Viktor Adler

Katedra elektromagnetického pole
B2-626, Dejvice

Naučíte se …

GUI #2



!!! **Pozor:** OD MATLABu R2014b ZMĚNY V GRAFICE !!!

Techniky návrhu GUI - rozdělení

- existuje několik přístupů (metodik) na tvorbu GUI
  - návrh pomocí nástroje GUIDE

```
>> guide
```

    - nedoporučuje se
  - switch-board technika
    - nedoporučuje se
  - využití vedlejších a vnořených (*nested*) funkcí jako callback funkcí
    - standardní
  - plně OOP přístup (vč. OO funkční části kódů)
    - ideální

Vyhodnocení callback funkce

- callback funkce je vyhodnocena jako handle funkce

```
hb = uicontrol('Style', 'pushbutton', 'String', 'Plot line')

% Volání funkce pomocí handle funkce
set(hb, 'Callback', @myFunc)
```

```
function myFunc(hObject, callbackdata)
% Callback funkce vždy přidává dva základní vstupy

hObject        % reference na objekt, který vyvolal callback
callbackdata   % struktura různých událostí (může být i prázdná)
```

Vyhodnocení callback funkce

- callback funkce je vyhodnocena jako handle funkce

```
hb = uicontrol('Style', 'pushbutton', 'String', 'Plot line')

% Cell array, kdy první prvek je handle funkce
set(hb, 'Callback', {@myFunc, inp1, ..., inpN})
```

```
function myFunc(hObject, callbackdata, inp1, ..., inpN)
% Opět přidává na první pozice základní vstupy

hObject        % reference na objekt, který vyvolal callback
callbackdata   % struktura různých událostí (může být i prázdná)

inp1, ..., inpN % ostatní vstupy
```

Vyhodnocení callback funkce

- callback funkce je vyhodnocena jako řetězec (funkcí) `eval`

```
hb = uicontrol('Style', 'pushbutton', 'String', 'Plot line')

% 3. způsob volání callback funkce
set(hb,'Callback', 'plot(rand(20,3))')
```

  - v řetězci mohou být i proměnné
    - správně se vyhodnotí jen ty, které jsou v `base Workspace`

Vyhodnocení callback funkce

- callback funkce je vyhodnocena jako anonymní funkce

```
hb = uicontrol('Style', 'pushbutton', 'String', 'Plot line')

% TIP – v případě volání funkce, která nepodporuje základní
vstupy Callback funkce, je možnost použít anonymní funkci
set(hb, 'Callback', @(a, b)myFunc(inp))
```

```
function myFunc(inp)

inp % vstupem pouze proměnné uživatele
```

Callback funkce – seznam

`Callback`	context menu, uiobjects
`CellEditCallback`	uitable
`CellSelectionCallback`	uitable
`ButtonDownFcn`	axes, figure, button group, panel, uiobjects
`ClickedCallback`	push tool, toggle tool
`CreateFcn, DeleteFcn`	axes, button group, context menu, figure, menu, panel, uiobjects, …
`OffCallback, OnCallback`	toggle tool
`ResizeFcn (<R2014b)`	figure, panel, button group
`SelectionChangeFcn`	button group
`KeyPressFcn`	figure, uiobjects
`KeyReleaseFcn`	figure
`WindowButtonDownFcn`	figure
`WindowButtonMotionFcn`	figure
`WindowButtonUpFcn`	figure
`WindowKeyPressFcn`	figure
`WindowKeyReleaseFcn`	figure
`WindowScrollWheelFcn`	figure
`CloseRequestFcn`	figure

Callback funkce – seznam

`Callback`	context menu, uiobjects
`CellEditCallback`	uitable
`CellSelectionCallback`	uitable
`ButtonDownFcn`	axes, figure, button group, panel, uiobjects
`ClickedCallback`	push tool, toggle tool
`CreateFcn, DeleteFcn`	axes, button group, context menu, figure, menu, panel, uiobjects, …
`OffCallback, OnCallback`	toggle tool
`SizeChangedFcn (>=R2014b)`	figure, panel, button group
`SelectionChangeFcn`	button group
`KeyPressFcn`	figure, uiobjects
`KeyReleaseFcn`	figure
`WindowButtonDownFcn`	figure
`WindowButtonMotionFcn`	figure
`WindowButtonUpFcn`	figure
`WindowKeyPressFcn`	figure
`WindowKeyReleaseFcn`	figure
`WindowScrollWheelFcn`	figure
`CloseRequestFcn`	figure

Funkce gcf, gca a gco

- slouží k jednoduchému přístupu k identifikátorům objektů, které jsou právě aktivní, konkrétně jde o:
  - `gcf` – vrací identifikátor aktuálního objektu `figure`
  - `gca` – vrací identifikátor aktuálního objektu `axes`
  - `gco` – vrací identifikátor objektu, na který se naposledy kliklo myší (tolerance u čar je 5 px)

```
figure
figRef = gcf
```

- tyto funkce můžeme využívat jako vstupní identifikátor pro jiné funkce vyžadující referenci objektu `figure` nebo `axes`

```
set(gcf,'color',[0 0 0])
```

Cvičení – pozice myši

600 s ↑

- vytvořte pouze textové pole, které ukazuje pozici myši nad automaticky otevřeným oknem `figure`.
  - referenci na okno můžete získat pomocí funkce `gcf`
  - informace o pozici naleznete v některé vlastnosti okna

Funkce findobj

- najde objekt(y) s požadovanou vlastností
- vrací referenci na objekt (event. pole referencí)

```
>> figHndl = gcf     % figHndl = figure;
>> axsHndl = gca     % axsHndl = figure;
>> htx1 = uicontrol('style','text','string','hello','tag','tx');
>> htx2 = uicontrol('style','text','string','test1','tag','tx2');
```

```
>> h = findobj('Style','text','-and','Tag','tx')

h =

  UIControl (tx) with properties:

              Style: 'text'
             String: 'hello'
    BackgroundColor: [0.9400 0.9400 0.9400]
           Callback: ''
              Value: 0
           Position: [20 20 60 20]
              Units: 'pixels'

  Show all properties
```

```
>> h = findobj('Style','text')
h =

  2x1 UIControl array:

  UIControl     (tx2)
  UIControl     (tx)
```

Cvičení – snímání klávesnice

600 s ↑

- vytvořte textové pole, které zobrazuje poslední stisknutou klávesu
  - informaci o stisknuté klávese najdete ve vstupu `callbackdata`
  - referenci na textové pole vyhledejte pomocí `findobj`

Funkce `findall`, `allchild`

- `findall` najde všechny grafické objekty (včetně skrytých), `handle_list` může být např. `gcf`
- `allchild` najde všechny potomky vybraného objektu (včetně skrytých)
- pokud je `handle_list` vektor identifikátorů, vrací Matlab pole buněk (`cell`)

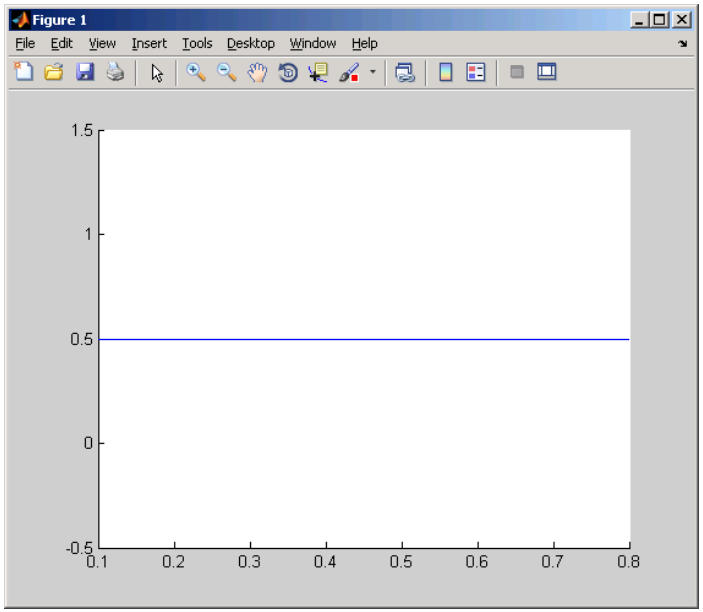
Funkce copyobj

- tato funkce umožňuje ovlivnit životní cyklus objektu
  - zkopíruje objekt i jeho potomky

- více >> doc copyobj

```
>> hf  = figure
>> ha  = axes
>> hl1 = line([0.1 0.8], [0.5 0.5])
>> hl2 = copyobj(hl1,ha)
>> ishandle(hl1) && ishandle(hl2)

ans =
     1
```



Funkce `delete`, `reset`

- tyto funkce umožňují ovlivnit životní cyklus objektu
- `delete` odstraní soubor(y) nebo grafický objekt(y) i s jeho potomky

```
>> delete(hf) % hf viz predesly priklad
>> ishandle(hl1) && ishandle(hl2)
ans =
     0
```

- `reset` nastaví všechny hodnoty objektu zpět na implicitní

```
reset(h)
```

Pokročilá vizualizace v Matlabu

- funkce předdefinování proměnné `gobjects`

```
% preallocation
h = gobjects(3,1);

h(1) = figure;
h(2) = plot(1:10);
h(3) = gca;
class(h)
```

- funkce `isgraphics()`

```
x = 1:10; y = sin(x);

p = plot(x,y);
ax = gca;

isgraphics([p, ax])
```

- chceme-li zjistit, zda je proměnná handle, užijeme funkci `ishandle`

```
>> figHandle = figure;
>> ishandle(figHandle)
```

  - `>> doc Graphics Object Identification`

Uchovávání dat v GUI

- jak uchovávat data v GUI?
  - globální proměnné (krajní možnost, příkaz `global`)
    - nepřijatelné
  - použití vlastnosti `UserData` (záleží na velikosti aplikace)
    - únosné
  - použití funkcí `guidata` nebo `setappdata` a `getappdata`
    - vhodné
  - plně OOP přístup (vč. OO funkční části kódů)
    - ideální

Funkce `guidata`

- funkce umožňuje uložit nebo získat data
- postup je následující:
  - získáme kopii dat: `data = guidata(object_handle)`
  - provedeme změnu dat / potřebný výpočet
  - pokud se data změnila, uložíme `guidata(object_handle,data)`
- data tedy svazujeme s určitým handlem, který existuje po celou dobu existence GUI

Funkce `guidata`

```
>> fhndl = figure('Toolbar', 'none');
>> allFigHndl = guihandles(fhndl);
>> guidata(fhndl, allFigHndl);
```

funkce `guihandles` vrací reference všech viditelných objektů ve `figure`

```
function myCallback()
% ...
myAllFigHndl = guidata(gcbo);
myAllFigHndl.time = clock;
guidata(gcbo, myAllFigHndl);
```

funkce `gcbo` vrací referenci objektu, jehož callback je právě vyhodnocován

Funkce `setappdata`, `getappdata`

- `setappdata`: umožní definovat nová data (dvojice jméno-hodnota) pro danou aplikaci

```
setappdata(hndl, 'speedA', value)
```

- `getappdata`: umožní získat dříve definovaná data z vybraného objektu

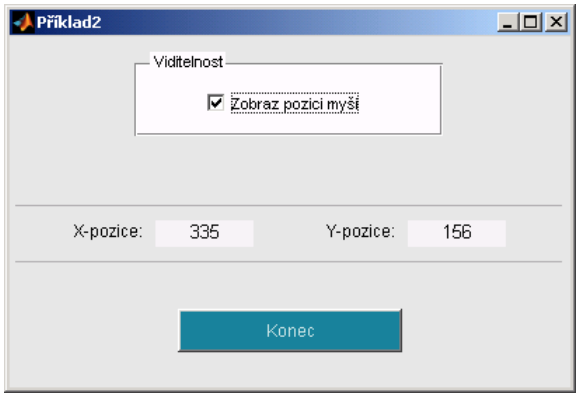
```
value  = getappdata(hndl, 'speedA')

% values je struktura
values = getappdata(hndl)
```

Cvičení – pohyb myši + tlačítka

600 s ↑

- vytvořte aplikaci podle obrázku
  - tlačítko „Konec“ ukončuje aplikaci
  - levé, resp. pravé, tlačítko myši mění řez písma „X-pozice“, resp. „Y-pozice“, z normálního na tučný a zpět
  - pokud je aktivován checkbox, vykresluje program pozici kurzoru myši



Cvičení – pohyb myši + tlačítka

Cvičení – pohyb myši + tlačítka

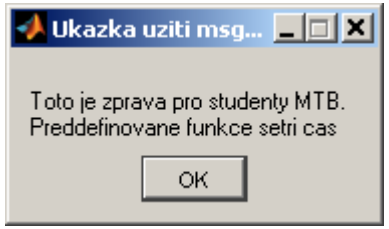
Předdefinovaná dialogová okna

- nejčastější operace uživatel ↔ GUI jsou předdefinovány
- ty nejužívanější jsou zobrazeny níže (většina):

Funkce msgbox

- zobrazí zprávu pro uživatele

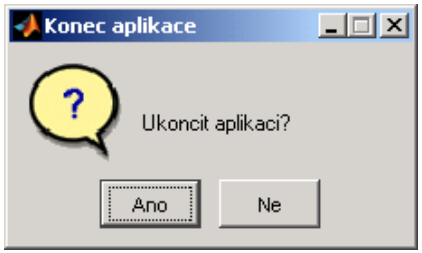
```
>> h = msgbox({'Toto je zprava pro studenty MTB.', ...
               'Preddefinovane funkce setri cas'}, ...
               'Ukazka uziti msgbox.')
```



Funkce `questdlg`

- zobrazí otázku, vrátí odpověď

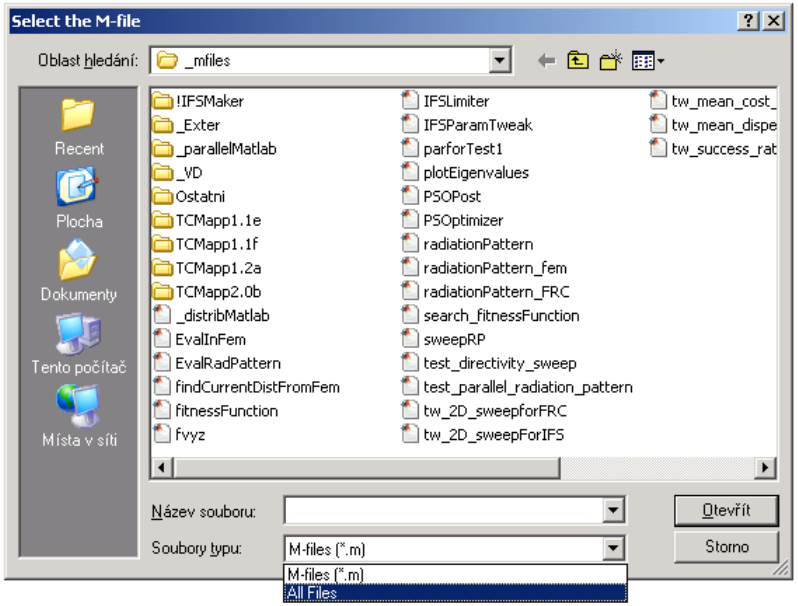
```
>> query = questdlg('Ukoncit aplikaci?', ...
                    'Konec aplikace','Ano','Ne','Ano')
```



Funkce `uigetfile`

- uživatel může vybrat soubor(y) ze souborového systému
  - soubory lze filtrovat podle jejich přípony

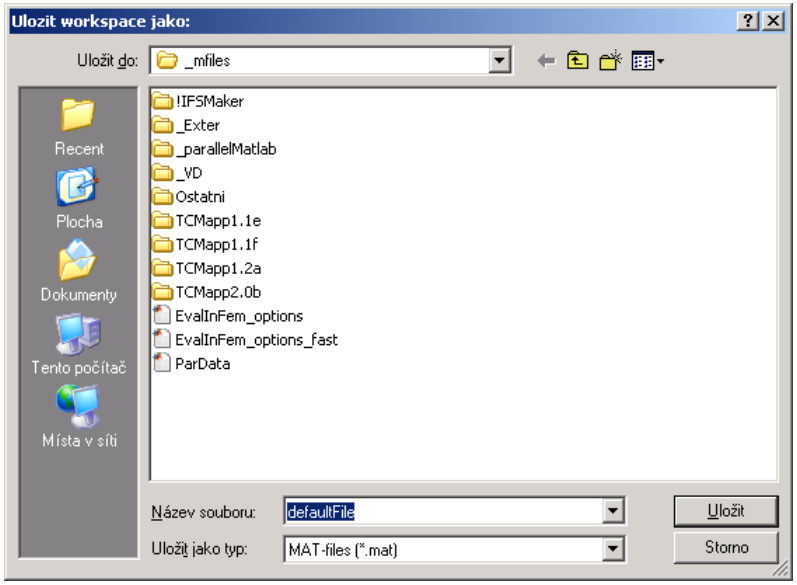
```
>> [FileName,PathName] = uigetfile('*.m', 'Select the M-file');
```



Funkce `uiputfile`

- otevře dialog pro ukládání soubor(ů)
  - soubory lze filtrovat podle přípony

```
>> [file,path] = uiputfile('*.mat', 'Ulozit workspace jako:', ...
    'defaultFile.mat')
```



Cvičení – ukládání do souboru

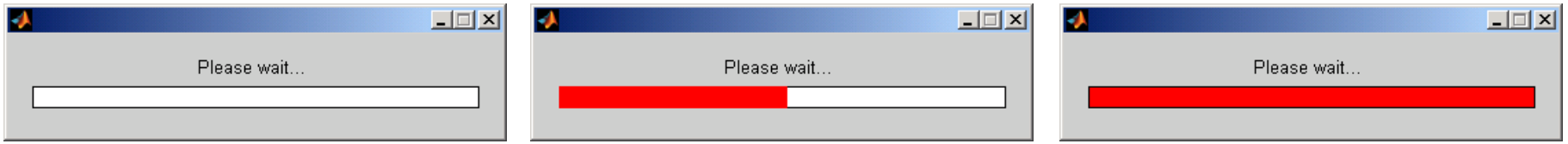
400 s ↑

- s pomocí dialogu uložte `data` z Workspace do souboru

Funkce `waitbar`

- zobrazuje stav procesu

```
>> h = waitbar(0, 'Please wait...');
   kroku = 1000;
   for k = 1:kroku
       waitbar(k/kroku);
   end
   close(h);
```



Návrh jednoduchého GUI #1

- co by měl umět (podrobný popis funkcionality)
- co chceme ovlivňovat na vstupu (uživatelské vstupy)
- požadované výstupy

- použité objekty (náčrt GUI, seznam prvků, návrh tagů, vlastností)
- callback funkce, dynamické prvky

- uložení identifikátorů a dat v GUI
- styl programování

- programování jednotlivých částí

- zprovoznění, testování

Probrané funkce

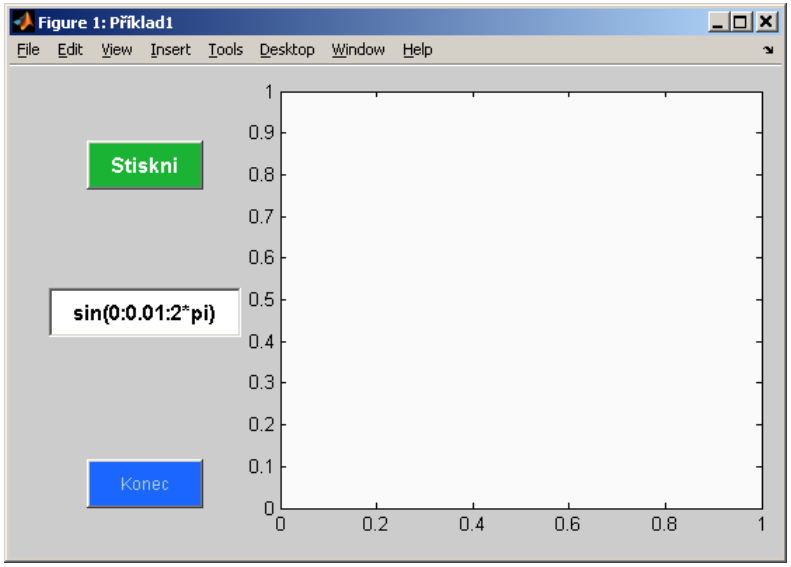
`gcf, gca, gco`	
`findobj, findall, allchild`	
`copyobj`	
`delete, reset`	
`gobjects, ishandle, isgraphics`	
`helpdlg, msgbox, warndlg, errordlg`	•
`inputdlg, listdlg, questdlg`	•
`uigetdir, uigetfile, uiopen`	•
`uiputfile, uisave`	•
`waitbar`	
`guide`	
`guidata, setappdata, getappdata`	

Cvičení – zobrazení funkce

600 s ↑

- rozšiřte předchozí funkci tak, aby umožňovala vykreslovat funkci, která bude zadávána uživatelem
  - využijte `try-catch` pro eliminaci chybných zadání
  - využijte funkce `reset` pro vyčištění grafu před dalším vykreslováním
  - jakou funkci využijete pro vyhodnocení textového řetězce?

```
>> MTB_GUI1edit
```

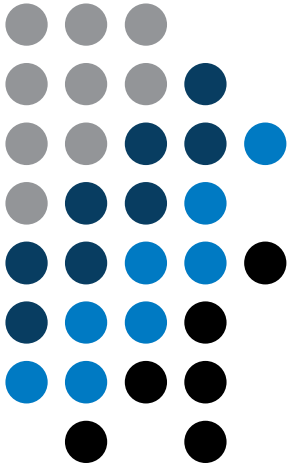


Cvičení – zobrazení funkce

Cvičení – zobrazení funkce

Děkuji!

ver. 3.1 (27/04/2015)
Miloslav Čapek
miloslav.capek@fel.cvut.cz



Materiál vytvořen v rámci předmětu A0B17MTB.