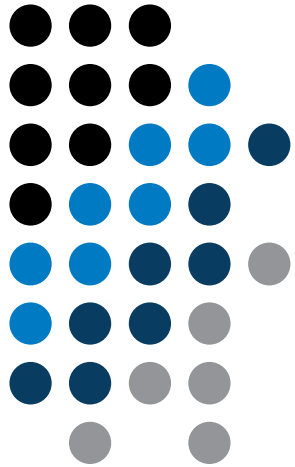
A0B17MTB – Matlab

Část #9

Miloslav Čapek
miloslav.capek@fel.cvut.cz

Filip Kozák a Viktor Adler

Katedra elektromagnetického pole
B2-626, Dejvice



Naučíte se …

Vizualizace v Matlabu #2

GUI #1

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

Pokročilá vizualizace v Matlabu

- základní možnosti vizualizace známe z 5. části přednášek
  - `figure` a základní zobrazení (`plot`, `stem`, …)
  - nastavení křivek grafu `LineSpec (doc LineSpec)`
  - funkce pro popis grafů, mřížka a legenda
- vlastnosti grafů
  - graf jako handle/objekt (změna od verze R2014b)
  - způsob nastavení hodnot vlastností grafických „objektu“
- Vybrané rozšířené možnosti vizualizace
  - vkládání více grafů do jednoho `figure`
  - desítky typů grafů (viz dokumentace)
  - zobrazení 3D grafů
  - `view`, `colormap`

Identifikátory objektů (do R2014b)

- každý samostatný objekt má svůj identifikátor (řečí Matlabu handle)
- tyto handely jsou v podstatě referencí na existující objekt
  - handle Matlab vytvoří vždy, je na uživateli jeho uchování
  - složité grafy (vrstevnice) mohou mít více identifikátorů
- `root` má vždy handle = 0 (více o `root` později), `figure` zpravidla celé kladné číslo, ostatní objekty mají za handle kladné reálné číslo (třídy `double`)

handely

```
>> figHandle = figure;
>> axHandle  = axes;
```

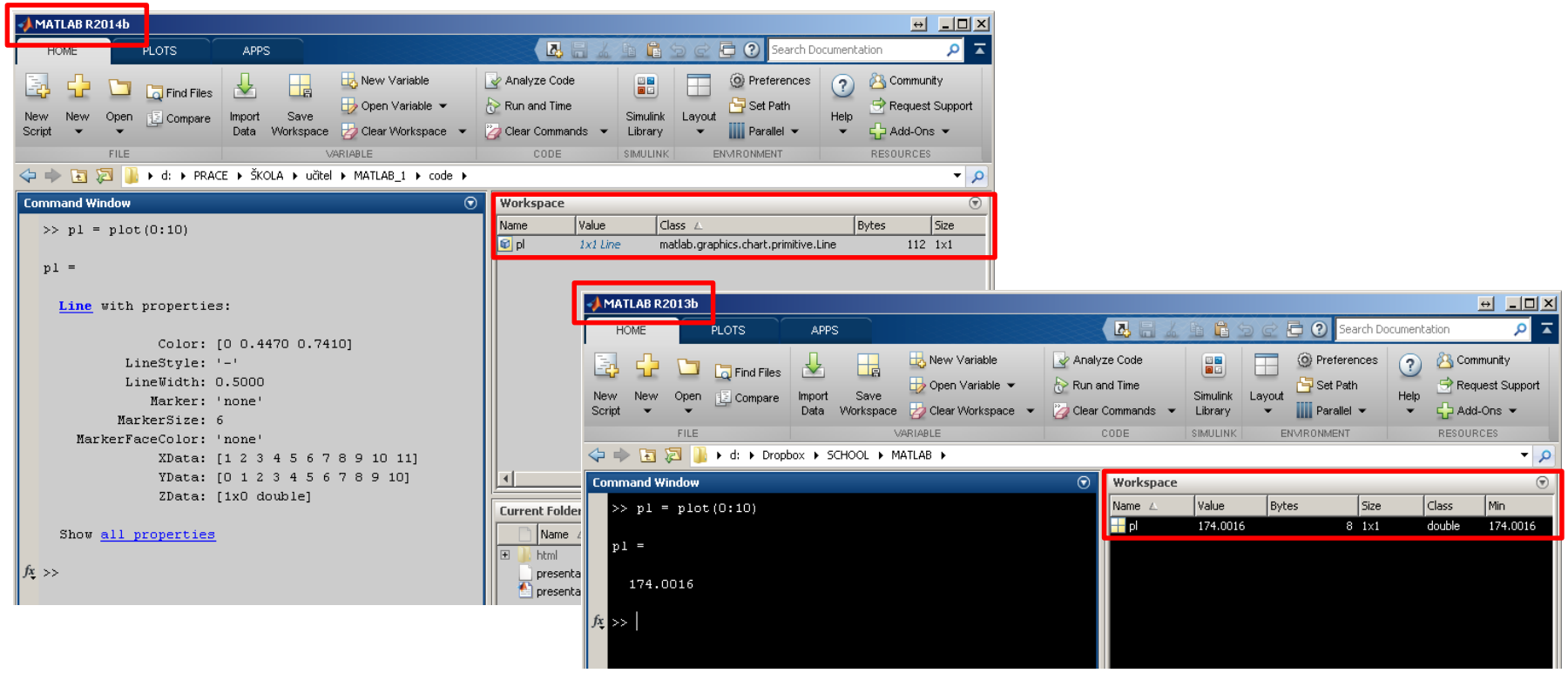
- číslo uložené v proměnné `figHandle` existuje i po uzavření okna, již se ale nejedná o handle

Identifikátory objektů (od R2014b)

- každý grafický objekt už je v pracovním prostoru značen jako objekt
  - objekt je definován třídou se svými vlastnostmi a metodami
- do `root` se dá stále přistupovat přes funkci `get()` s parametrem 0
  - `root` je nově `groot` objekt
  - (více v části GUI #1)
- po zničení objektu (zavření `figure`)
  - ve Workspace objekt stále existuje (jeví se jako reference na smazaný objekt)

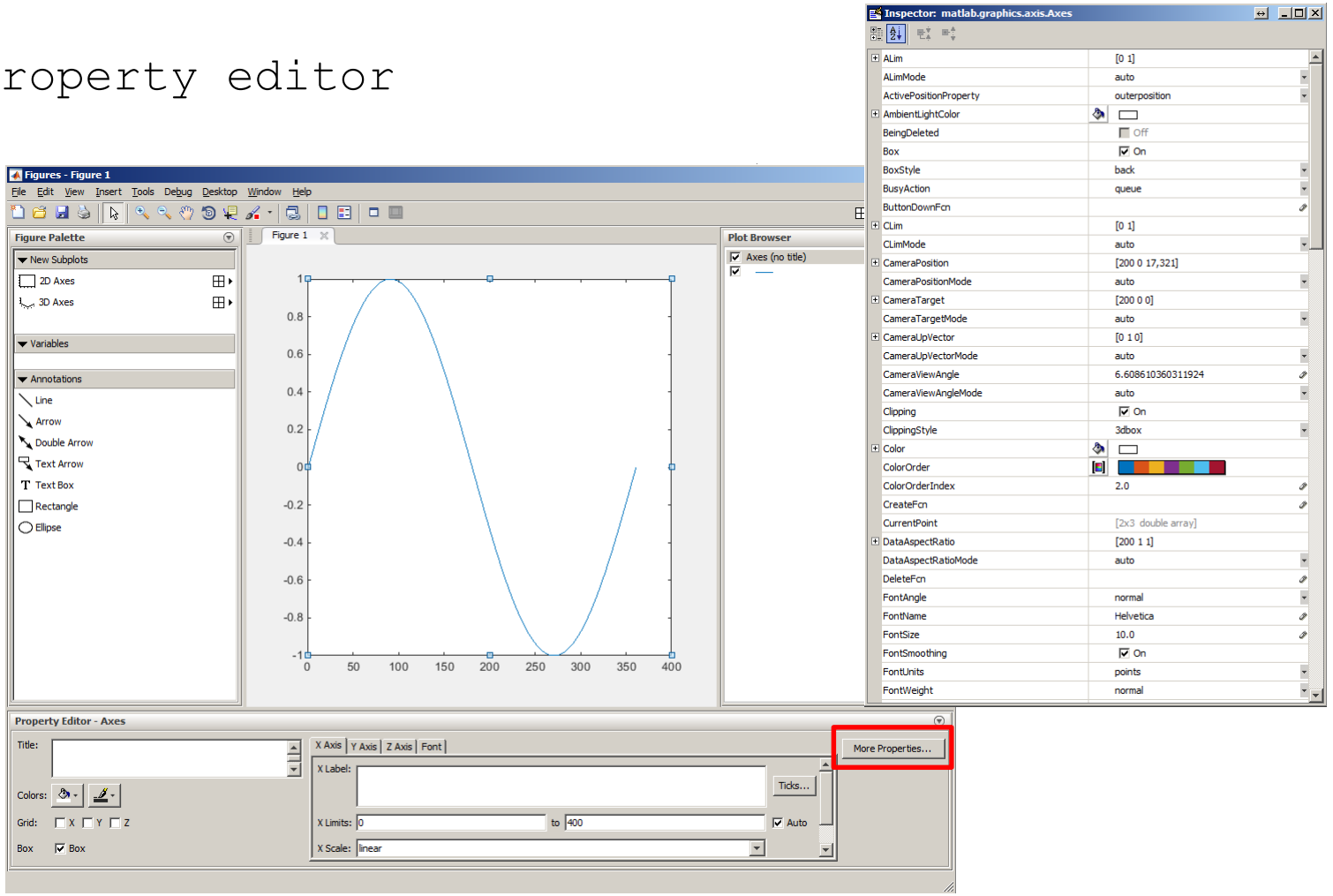
Pokročilá vizualizace v Matlabu

- graf jako handle číslo (verze < R2014b)
- graf jako objekt (od verze R2014b)
  - pozn. dále budeme mluvit o grafech jako o handle objektech



Pokročilá vizualizace v Matlabu

- Property editor



Pokročilá vizualizace v Matlabu

- způsob nastavování vlastností handle objektů
  - u obou verzí je možnost využít funkcí `set` a `get`
    - není case sensitive

```
>> myPlotObj = plot(1:10);
>> get(myPlotObj, 'color')
```

```
>> set(myPlotObj, 'color', 'r')
>> get(myPlotObj, 'color')
```

  - tečková notace (pouze od verze R2014b)
    - je cAsE sEnSiTiVe

```
>> myPlotObj = plot(1:10);
>> myPlotObj.Color
```

```
>> myPlotObj.Color = 'r';
>> myPlotObj.Color
```

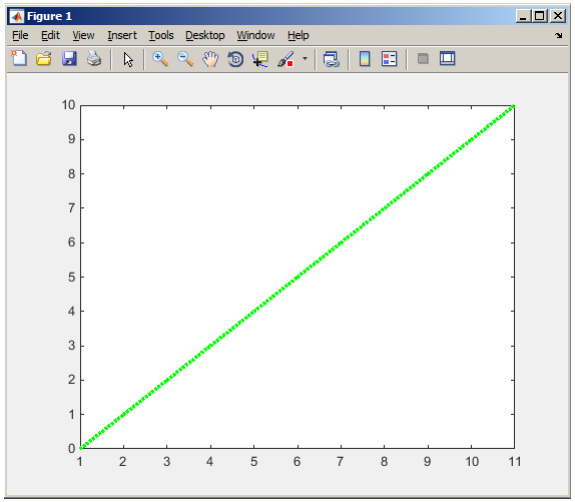
Funkce `get` a `set`

60 s ↑

- Vytvořte grafický objekt podle předpisu. Pak s využitím funkcí `get` a `set` udělejte následující úkoly.

```
myPlotObj = plot(0:10);
```

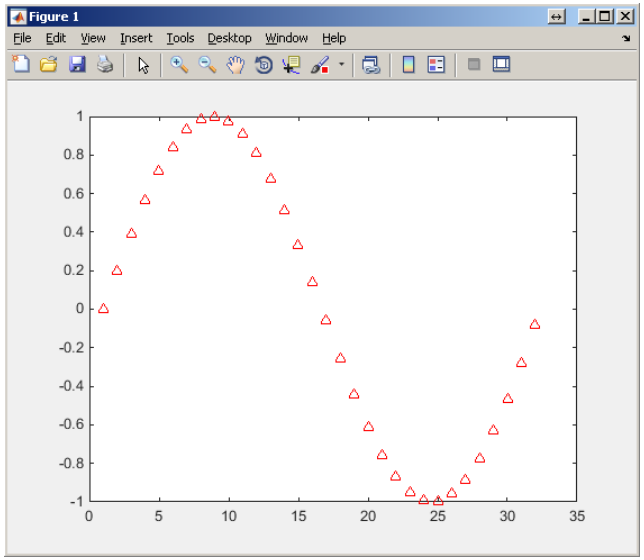
  - zjistěte základní šířku čáry, ke které přičtěte 1.5
  - nastavte barvu čáry na zelenou
  - nastavte styl čáry na tečkovanou



Použití tečkové notace

60 s ↑

- Pomoci tečkové notace změňte základní nastavení předepsané funkce tak, aby jste dosáhli stejného zobrazení.

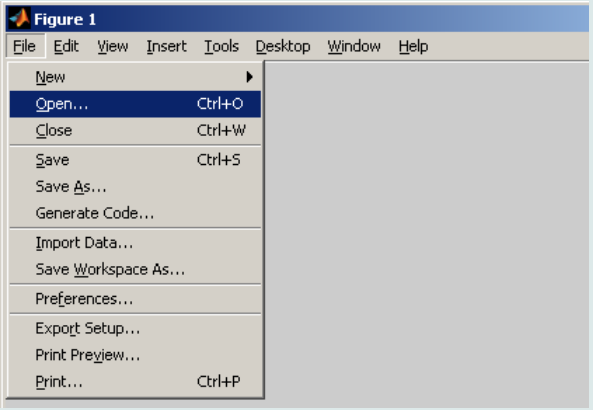


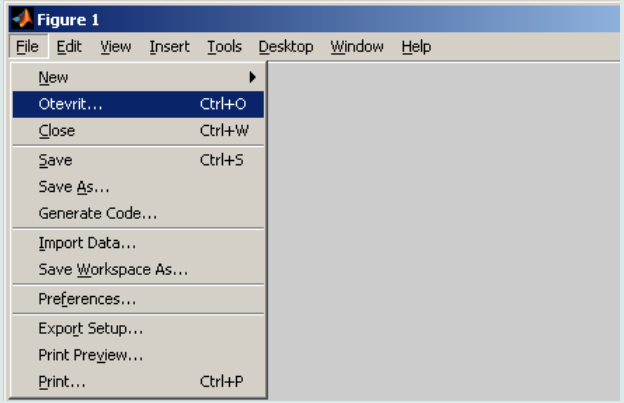

```
myPlotObj = plot(sin(0:0.2:2*pi));
```

K čemu lze využít handle?

- máme-li handle, zcela vládneme daným objektem
- příklad níže vrátí všechny identifikátory existující v okně `figure`
- můžeme tak změnit např. nápis Open… na Otevrit…
  - případně cokoliv jiného (např. callback otevírání souboru na callback zavření okna ☺)

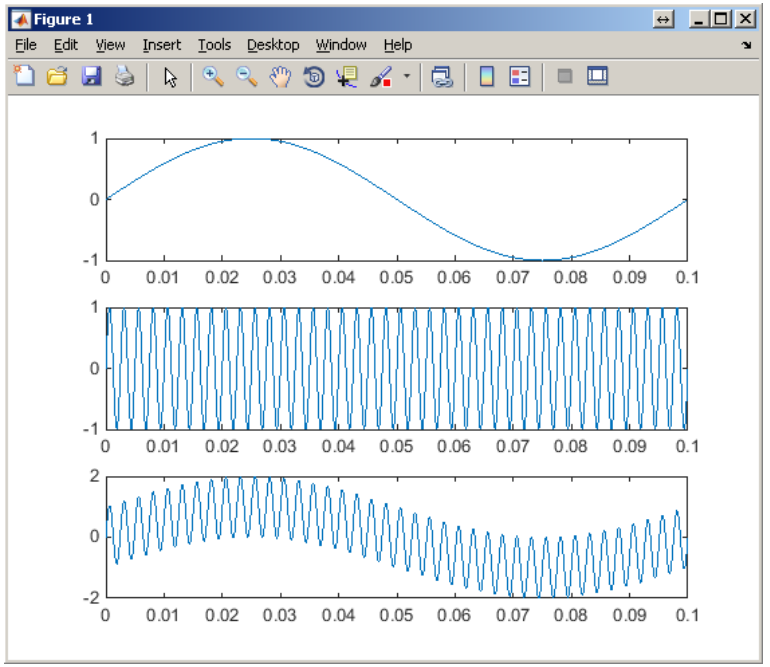
```
fhndl = figure('Toolbar','none');
allFigHndl = guihandles(fhndl);
set(allFigHndl.figMenuOpen,'Label','Otevrit...')
```





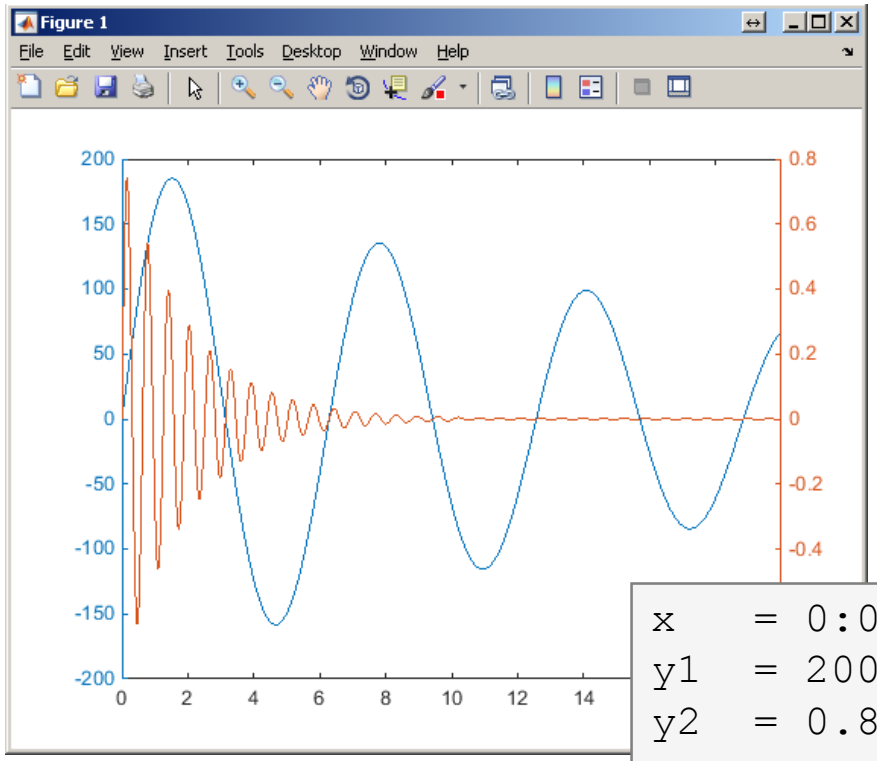
Více grafů v okně – `subplot`

- vkládání více různých grafů do jednoho okna `figure`
  - funkce `subplot(m,n,p)`
  - `m` – počet řádků
  - `n` – počet sloupců
  - `p` – umístění




```
t   = linspace(0, 0.1, 0.1*10e3);
f1  = 10;   f2  = 400;

y1  = sin(2*pi*f1*t);
y2  = sin(2*pi*f2*t);
y   = sin(2*pi*f1*t) + sin(2*pi*f2*t);

figure('color', 'w')
subplot(3, 1, 1); plot(t, y1);
subplot(3, 1, 2); plot(t, y2);
subplot(3, 1, 3); plot(t, y);
```

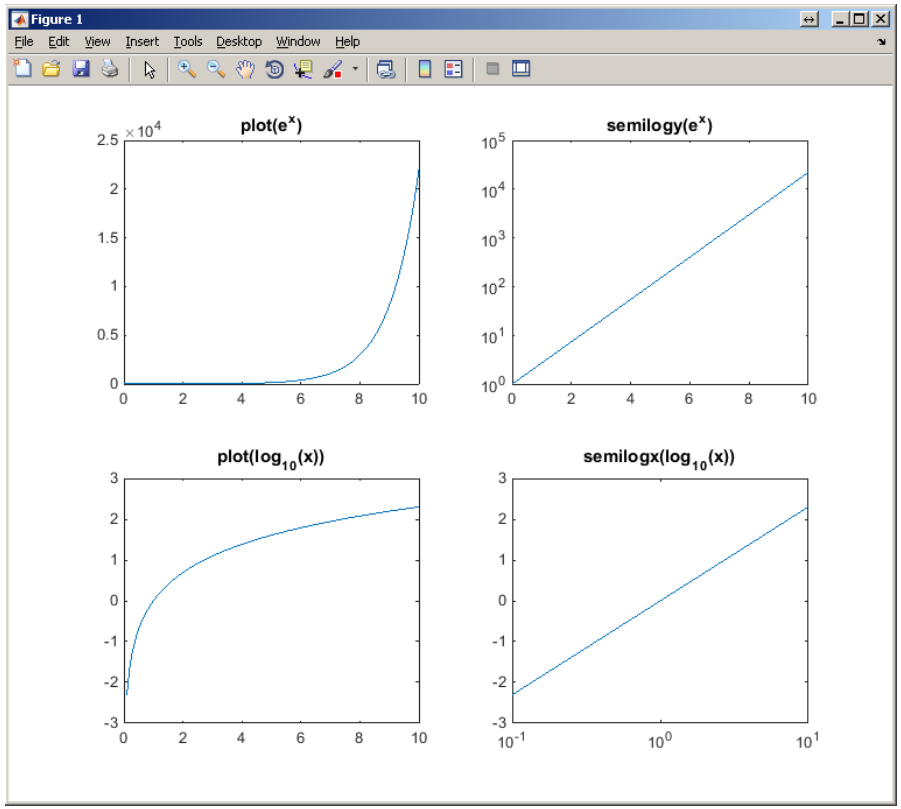
Dvě osy y – plotyy




```
x    = 0:0.01:20;
y1   = 200 * exp(-0.05*x) .* sin(x);
y2   = 0.8 * exp(-0.5*x) .* sin(10*x);

figure('color', 'w');
plotyy(x, y1, x, y2);
```

Logaritmické měřítko

- funkce `semilogy`, `semilogx`, `loglog`



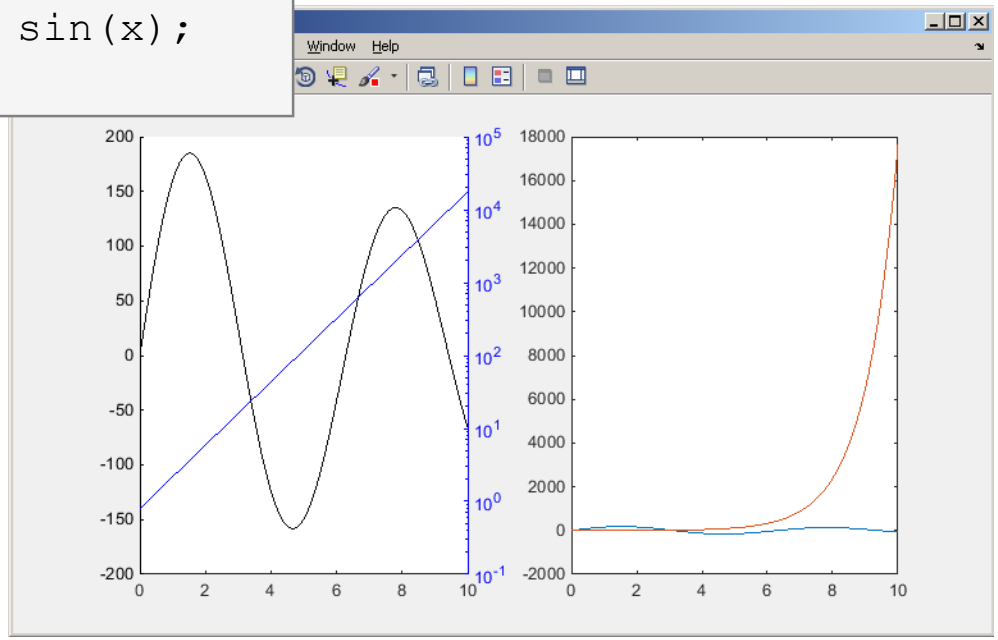

```
x    = 0:0.1:10;
y1   = exp(x);
y2   = log(x);

figure('color', 'w')
subplot(2, 2, 1); plot(x, y1);
title('plot(e^x)');

subplot(2, 2, 2); semilogy(x, y1);
title('semilogy(e^x)')

subplot(2, 2, 3); plot(x, y2);
title('plot(log_1_0(x))')

subplot(2, 2, 4); semilogx(x, y2);
title('semilogx(log_1_0(x))')
```

Příklad

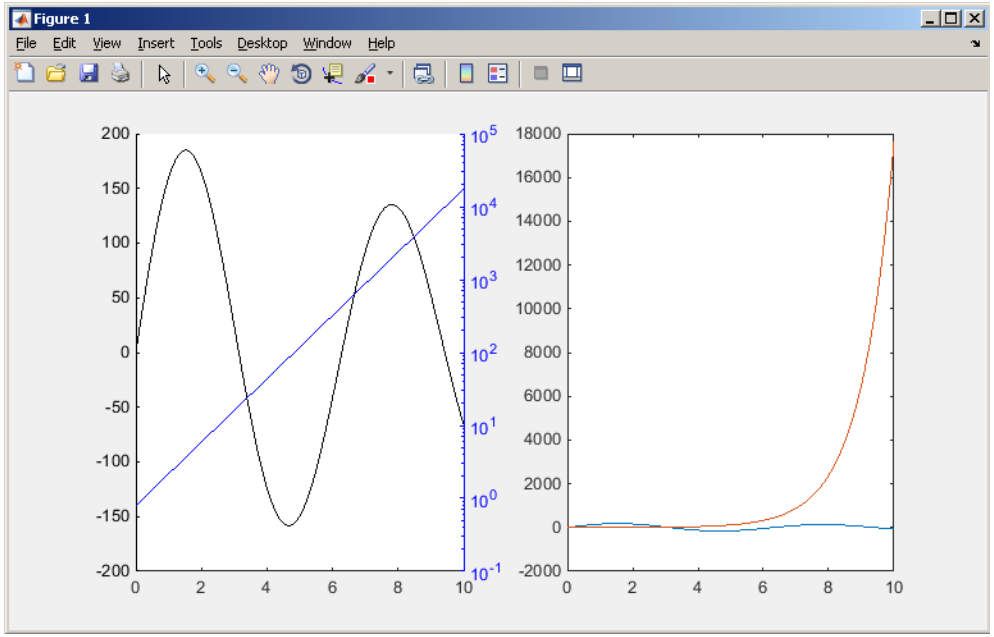
600 s ↑

- porovnejte funkce `plot` a `plotyy` v jednom objektu `figure` (s využitím `subplot`) na následujícím předpisu
  - v grafickém objektu vytvořeného `plotyy` změňte defaultní barvy jednotlivých čar na modrou a černou (nezapomeňte i na osy)

```
x  = 0:0.1:10;
y1 = 200 * exp(-0.05*x) .* sin(x);
y2 = 0.8 * exp(x);
```

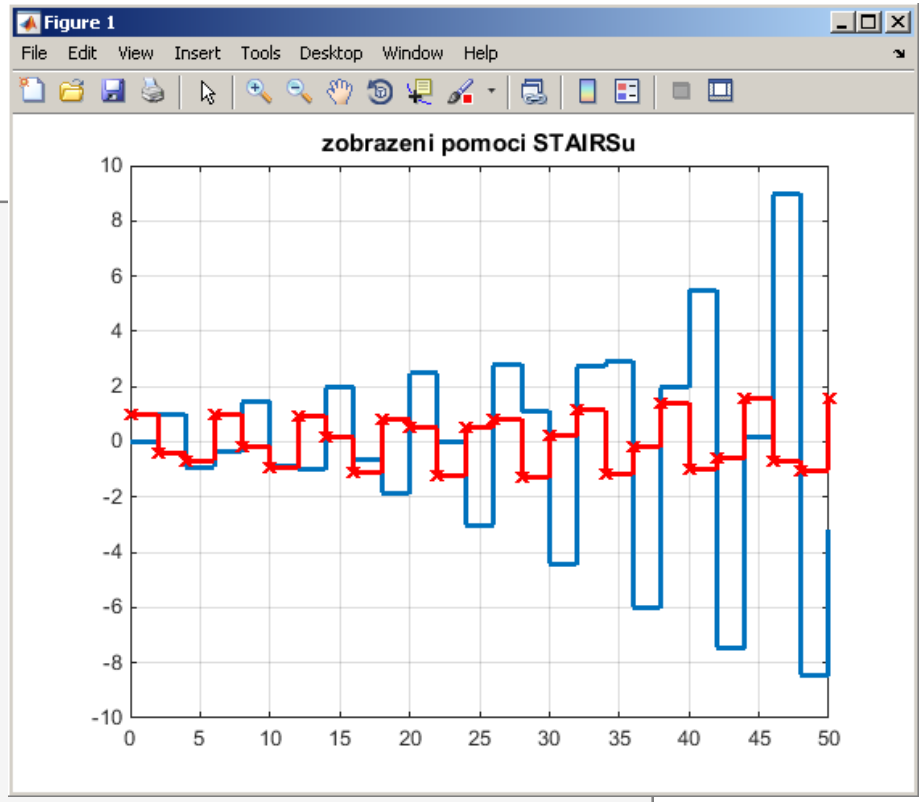


Příklad - řešení



stairs

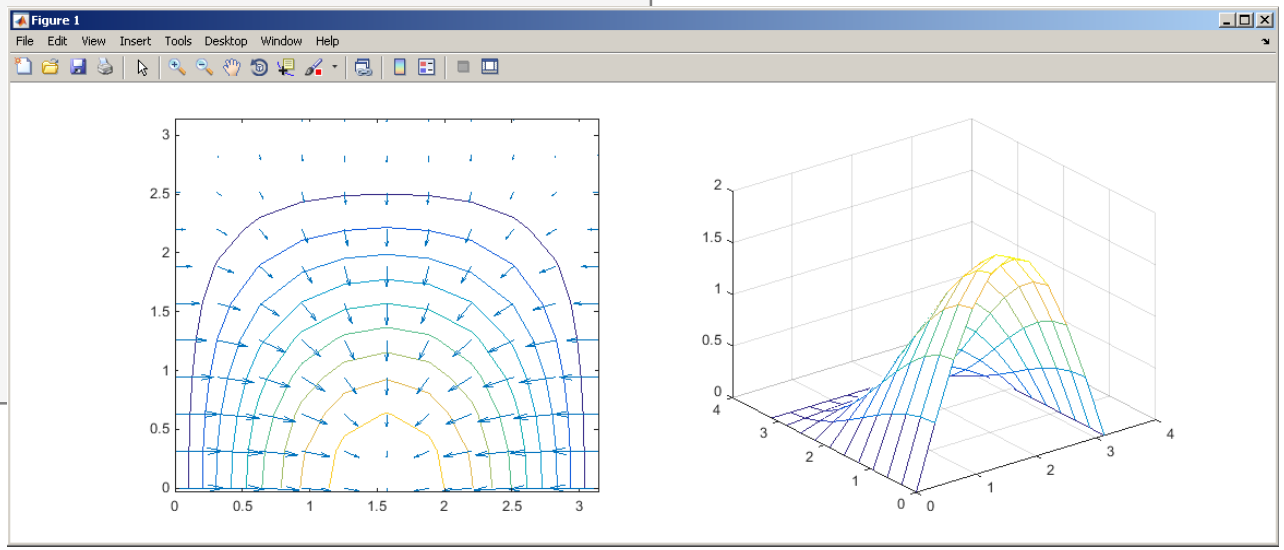
```
x  = 0:2:50;
y1 = exp(0.05*x) .* sin(x);
y2 = exp(0.01*x) .* cos(x);

figure('color', 'w');
stairs(x, y1, 'LineWidth', 2);
hold on; grid on;
stairs(x, y2, ...
    'Color', 'r', ...
    'Marker', 'x', ...
    'LineWidth', 2);

title('zobrazeni pomoci STAIRSu');
```



Zobrazení funkcí 2 proměnných

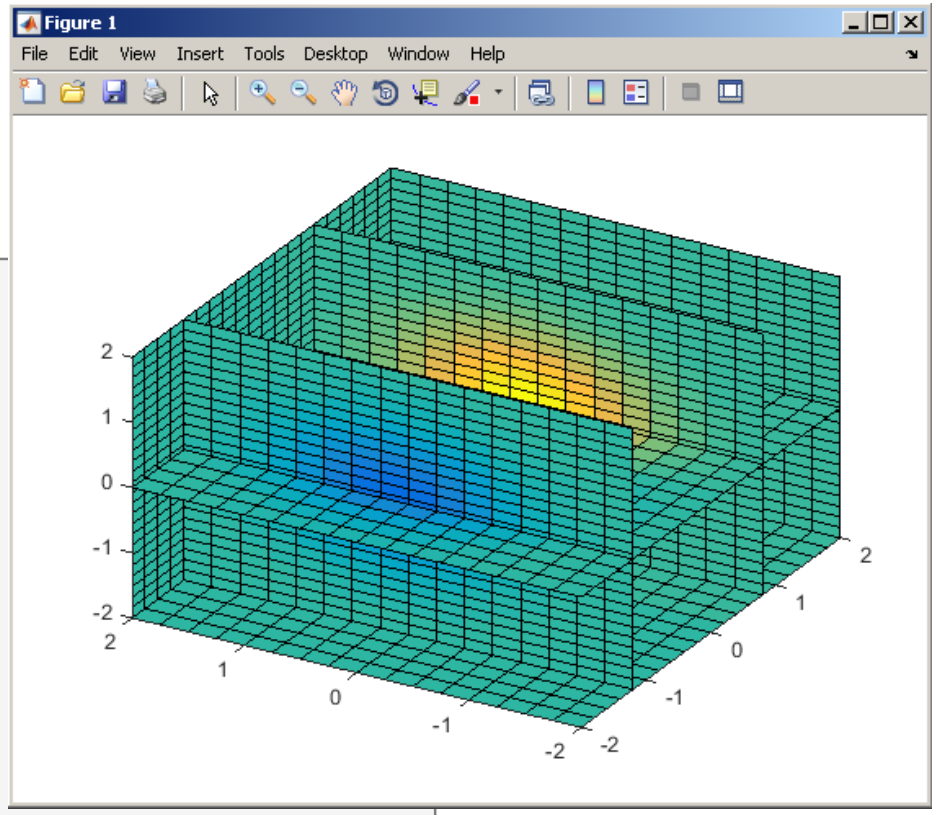
- `contour`, `quiver`, `mesh`

```
t          = 0:pi/10:pi;
[x, y]     = meshgrid(t);
z          = sin(x) + cos(y) .* sin(x);
[gx, gy]   = gradient(z);

figure('color','w');

subplot(1, 2, 1);
contour(x, y, z);
hold on;
quiver(t, t, gx, gy);

subplot(1, 2, 2);
mesh(x, y, z);
```



Pokročilá vizualizace v Matlabu

- funkce `slice`
- funkce `view`

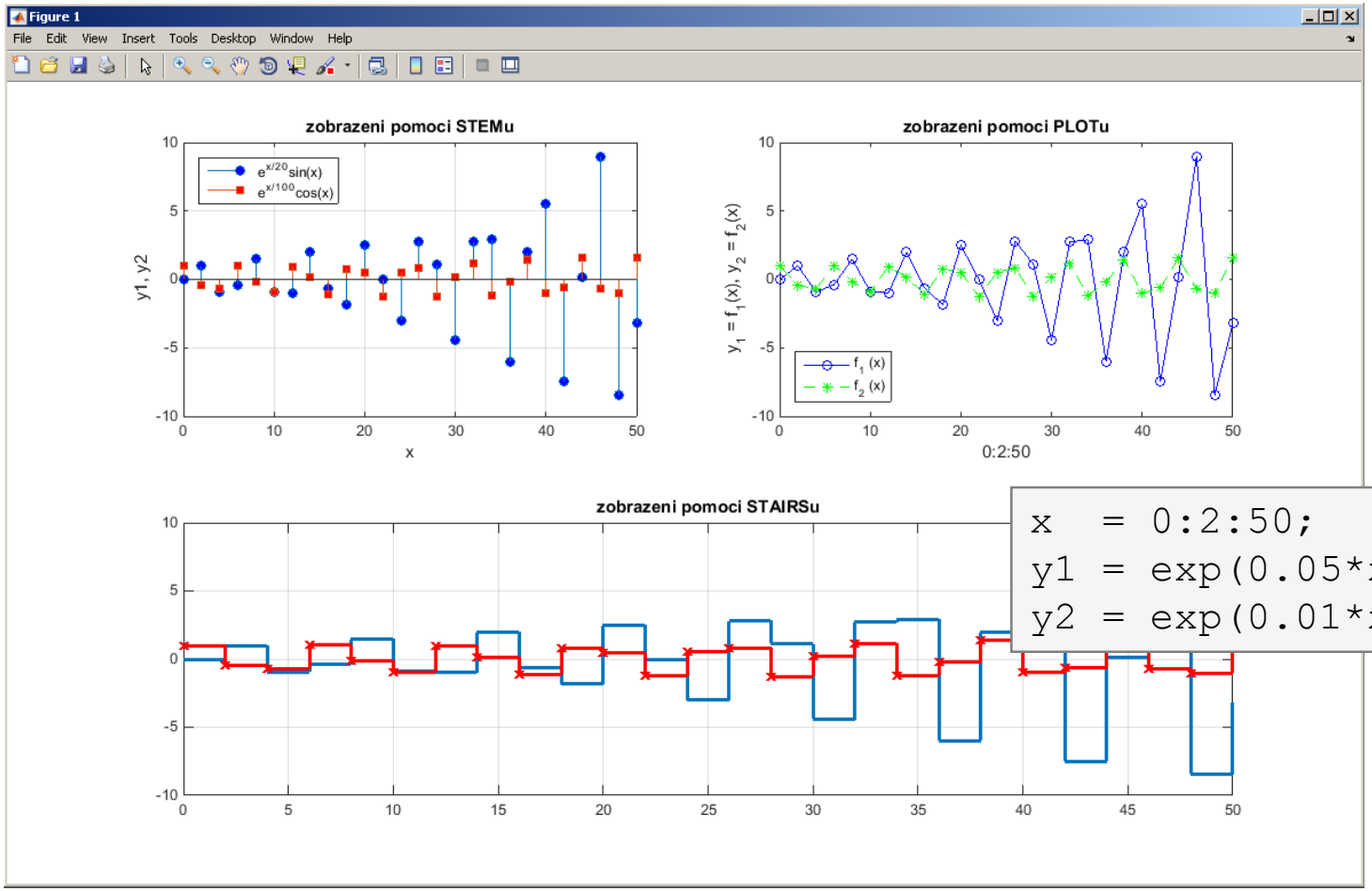



```
[x, y, z] = meshgrid(-2:0.2:2, ...
                     -2:0.25:2, ...
                     -2:0.16:2);

v = x .* exp(-x.^2 - y.^2 - z.^2);

xslice = [-1.2, 0.8, 2];
yslice = 2;
zslice = [-2, 0];

figure('color', 'w');
slice(x, y, z, v, xslice, yslice, zslice);
% view(azimuth, elevation)
view(-60, 40);
```

Cvičení #1 zadání

600 s ↑

- napodobte obrázek, jsou-li funkce `y1` a `y2` definovány jako:

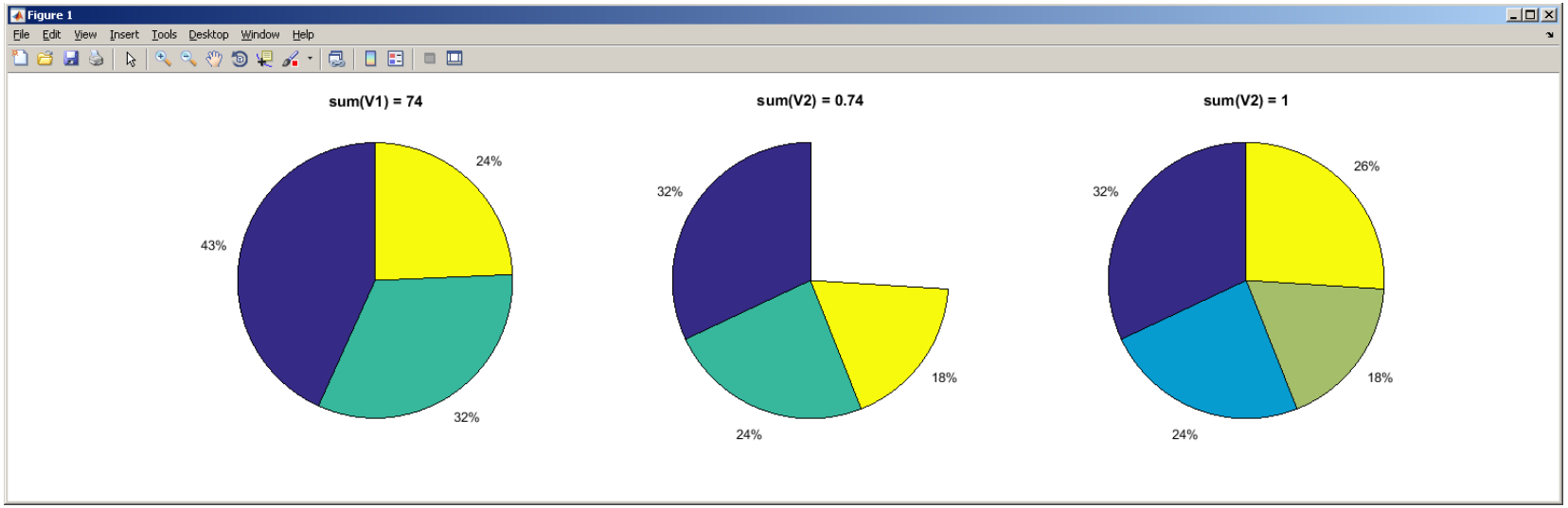



```
x  = 0:2:50;
y1 = exp(0.05*x) .* sin(x);
y2 = exp(0.01*x) .* cos(x);
```

Cvičení #1 řešení

Koláčový graf – pie, pie3

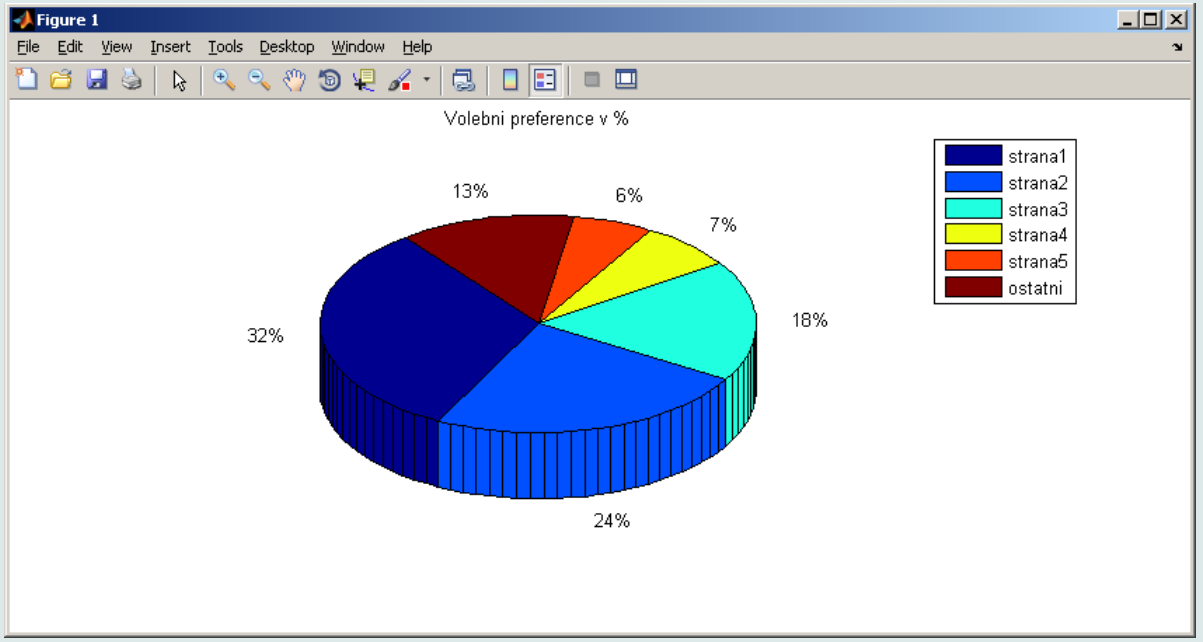
```
V1 = [32 24 18];          % sum(V1) = 74
V2 = V1/100;              % sum(V2) = 0.74
V3 = [V2 1-sum(V2)];      % sum(V3) = 1

figure('color', 'w');
subplot(1, 3, 1); pie(V1); title('sum(V1) = 74');
subplot(1, 3, 2); pie(V2); title('sum(V2) = 0.74');
subplot(1, 3, 3); pie(V3); title('sum(V2) = 1');
```



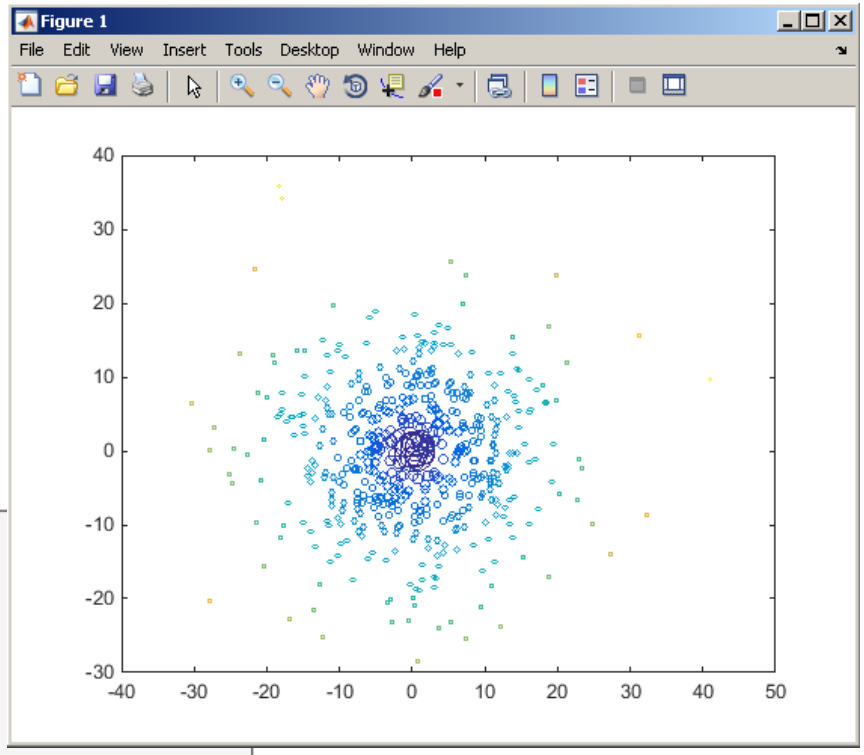
Cvičení

600 s ↑

- volební strany dosáhly následujících preferencí:
- zobrazte tyto preference pomocí klasického volebního koláče vč. položky „ostatní“
  - 1. strana: 32%
  - 2. strana: 24%
  - 3. strana: 18%
  - 4. strana: 7%
  - 5. strana: 6%



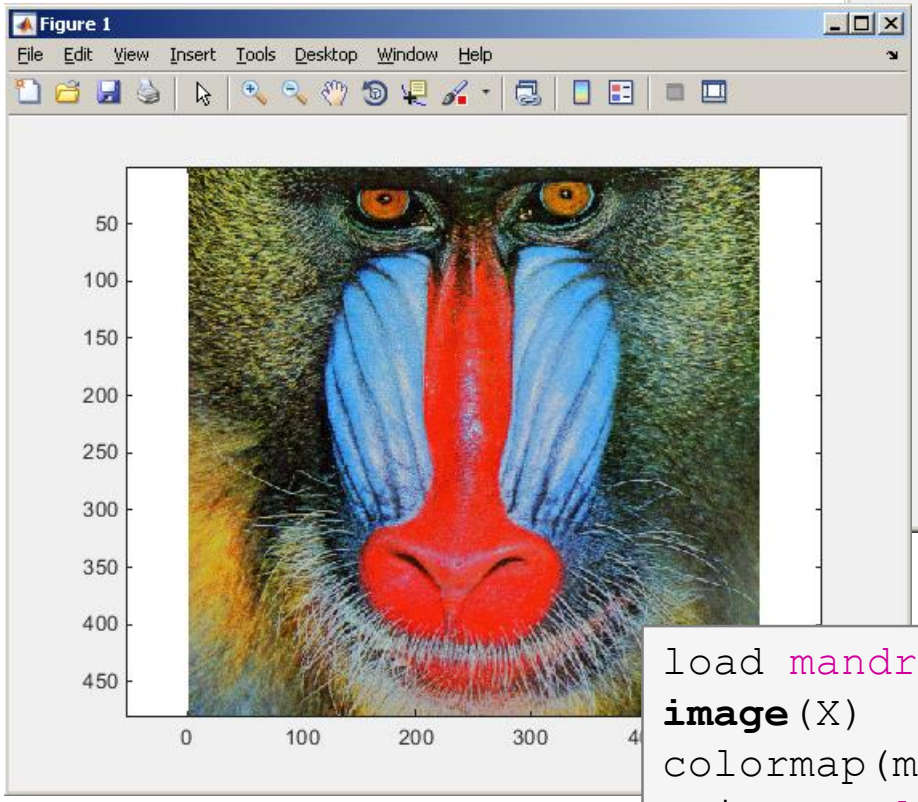
scatter



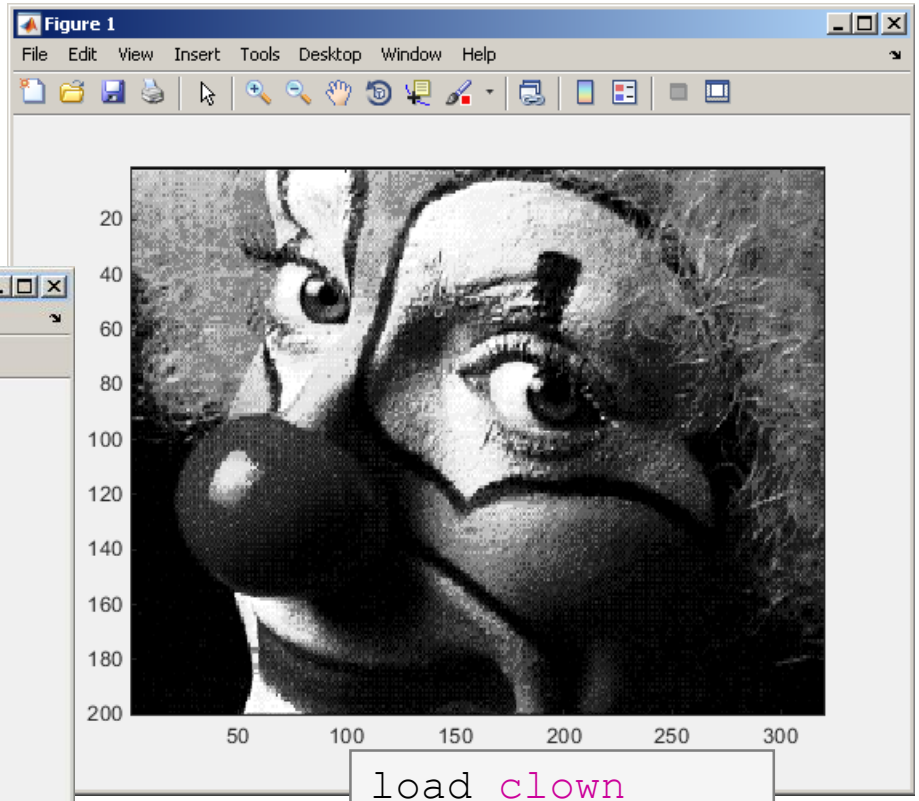

```
x = 10 * randn(500, 1);
y = 10 * randn(500, 1);
c = hypot(x, y);

figure('color', 'w');
scatter(x, y, 100./c, c);
box on;
```

Zobrazení obrázků

- funkce `image`, `imagesc`
- funkce `colormap`



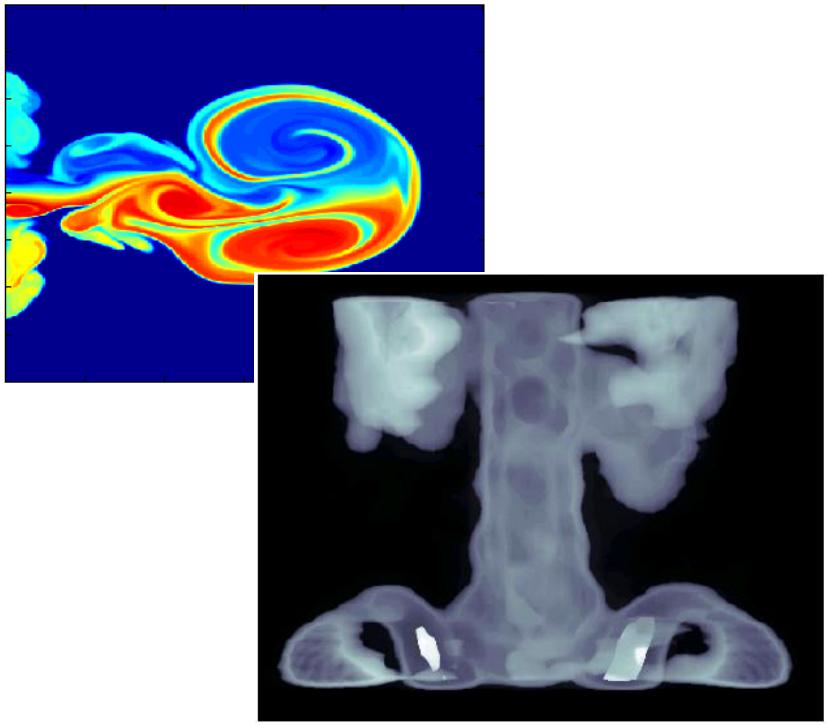

```
load mandrill
image(X)
colormap(map)
axis equal
```




```
load clown
imagesc(X)
colormap(gray)
```

colormap

- označuje barevnou škálu, na kterou jsou mapovány obrázky
- lze vytvořit / použít vlastní: `colormapeditor`

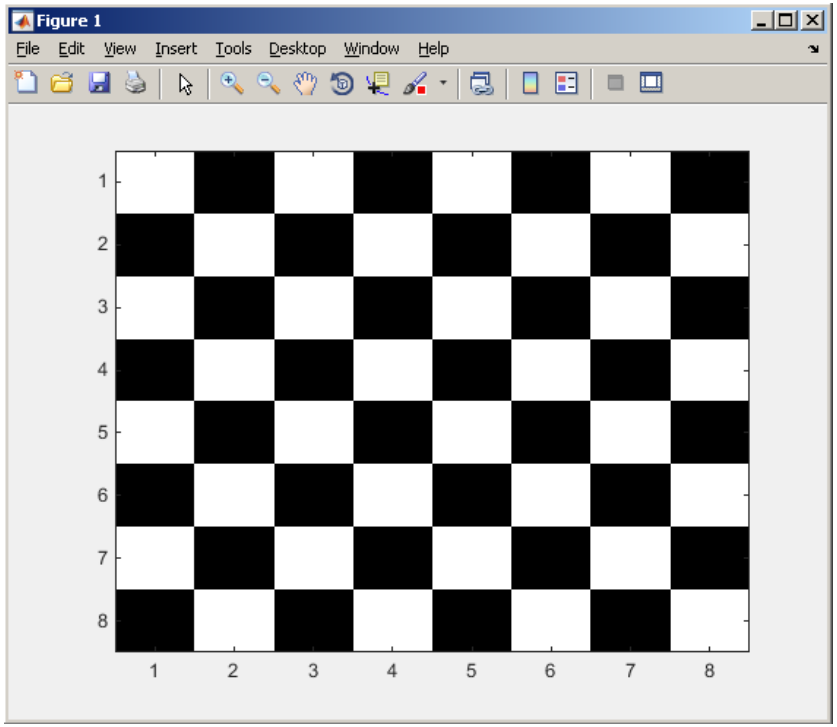


Colormap Name	Color Scale
parula	
jet	
hsv	
hot	
cool	
spring	
summer	
autumn	
winter	
gray	
bone	
copper	
pink	
lines	
colorcube	
prism	
flag	
white	

Cvičení

600 s ↑

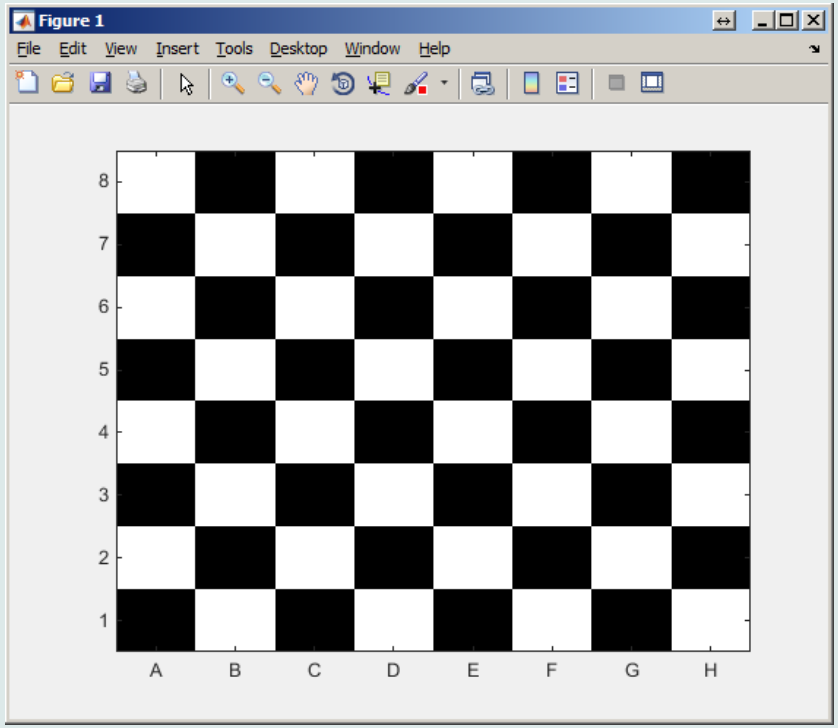
- vytvořte šachovnici podle obrázku:
  - obrázek lze vykreslit pomocí funkce `imagesc`
  - zvažte nastavení `colormap`



Cvičení

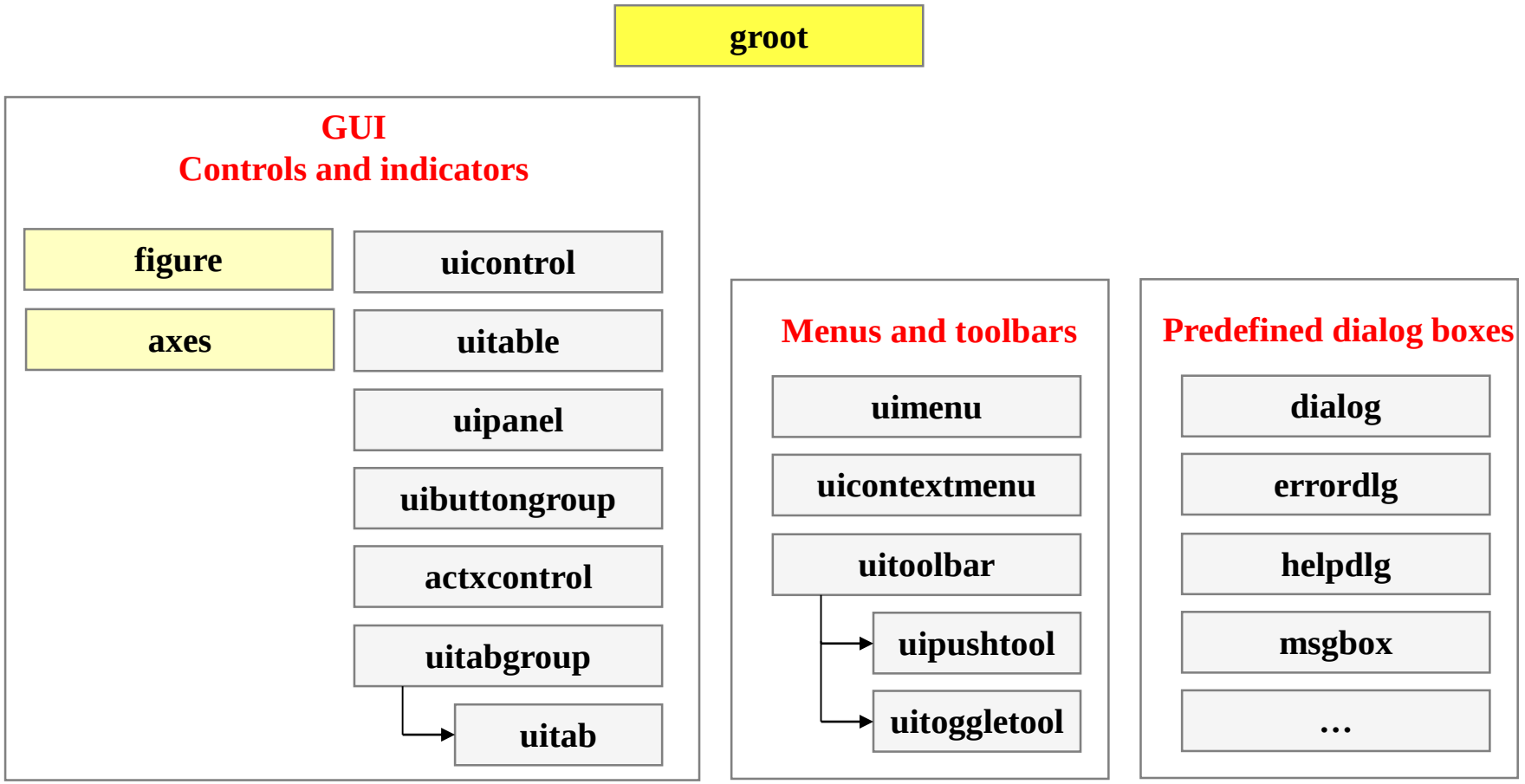
600 s ↑

- Upravte osy šachovnice tak, aby popis odpovídal skutečnosti:



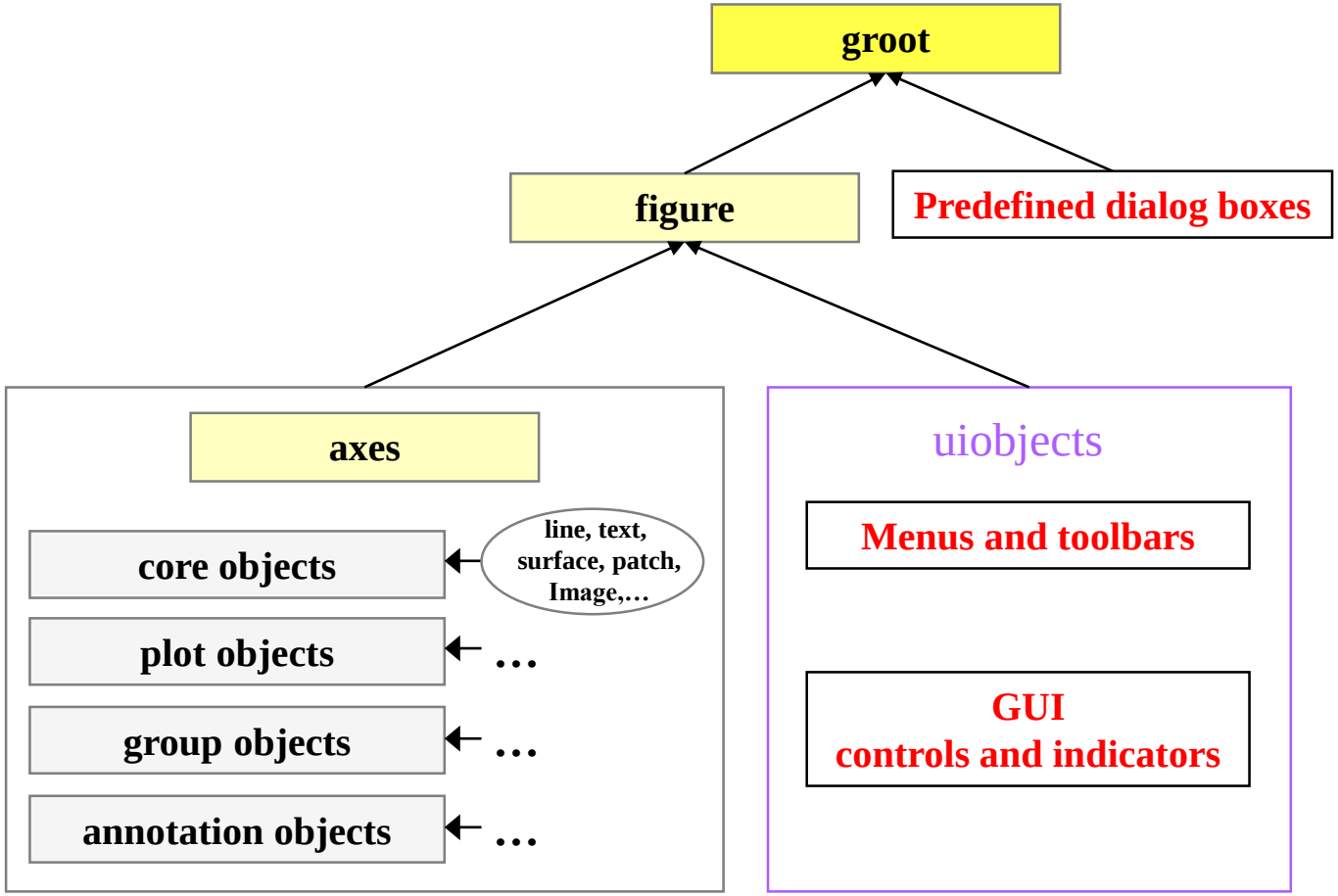
Struktura GUI

- objekty jsou rozděleny podle logické závislosti

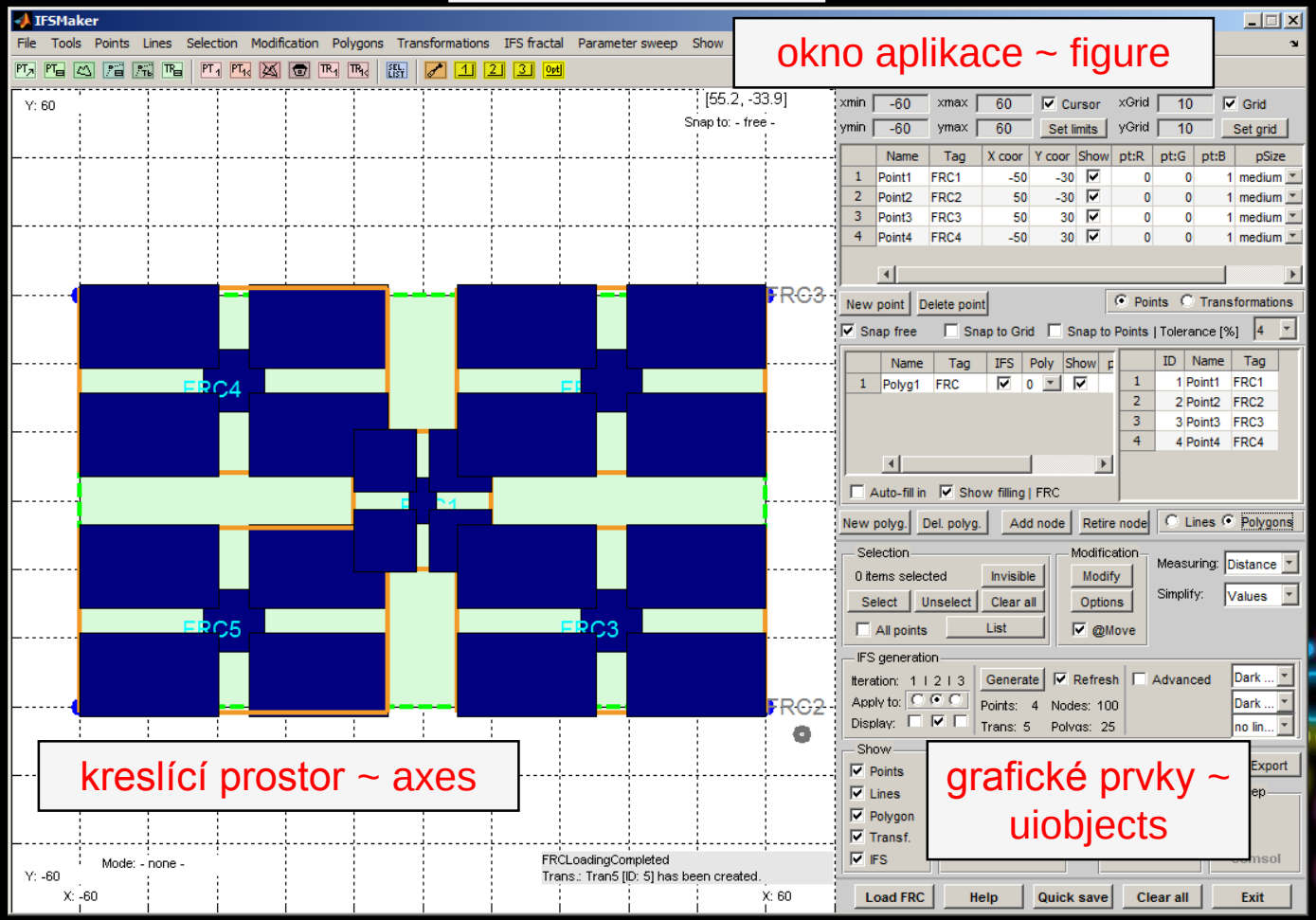


Struktura GUI

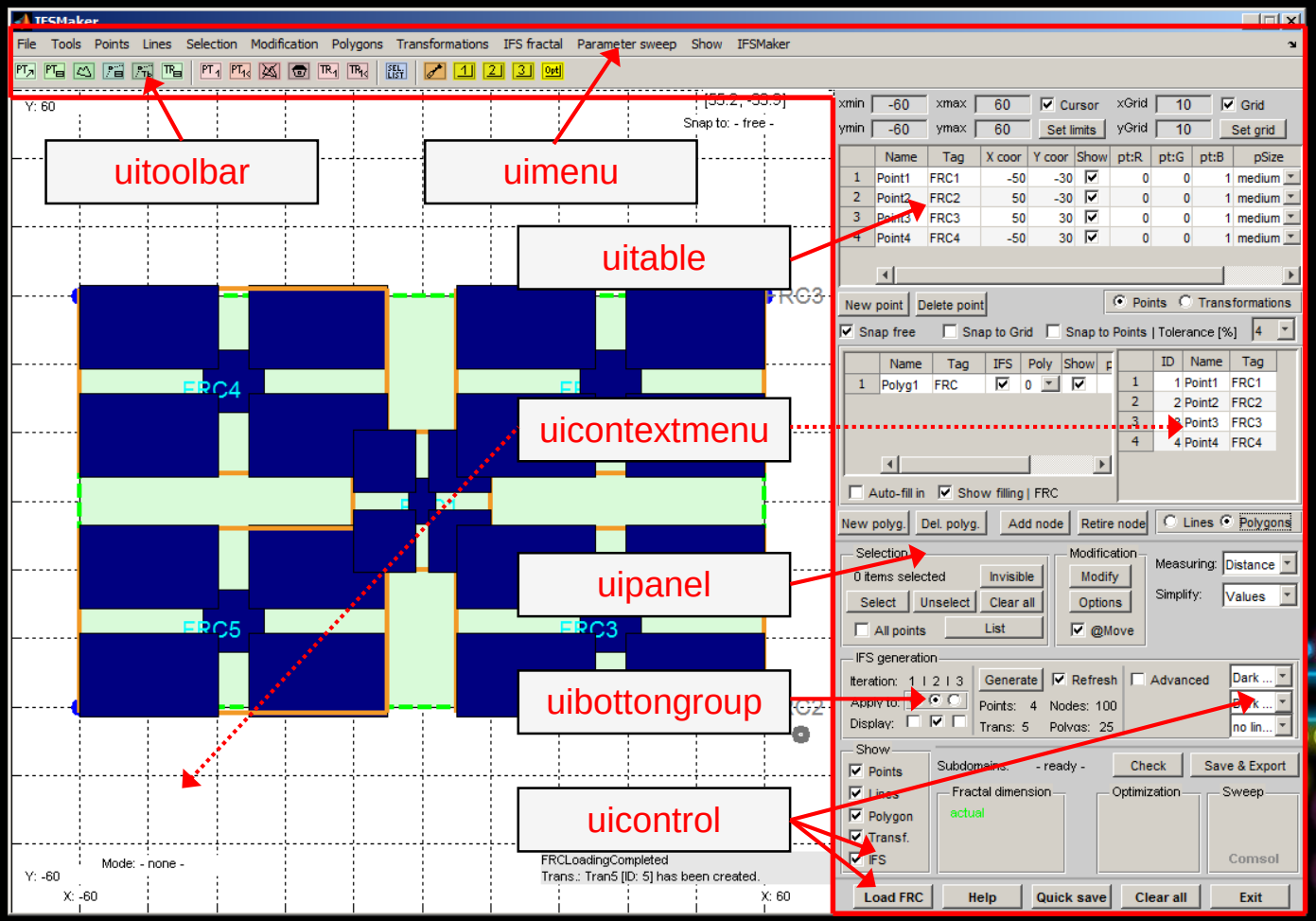
- hierarchie objektů



Struktura GUI #1

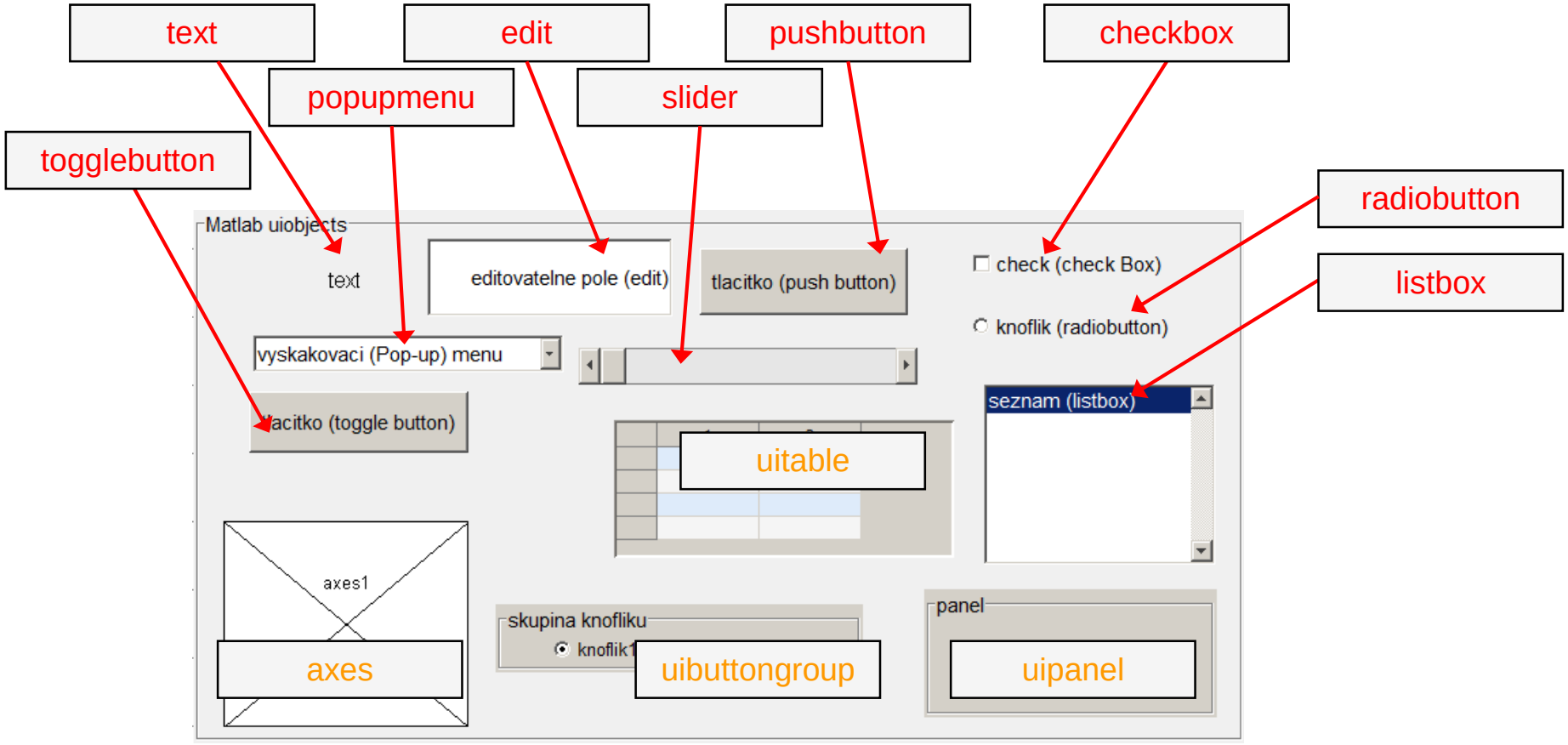


Struktura GUI #2



Katedra elektromagnetického pole, FEL ČVUT , miloslav.capek@fel.cvut.cz

Struktura GUI #3



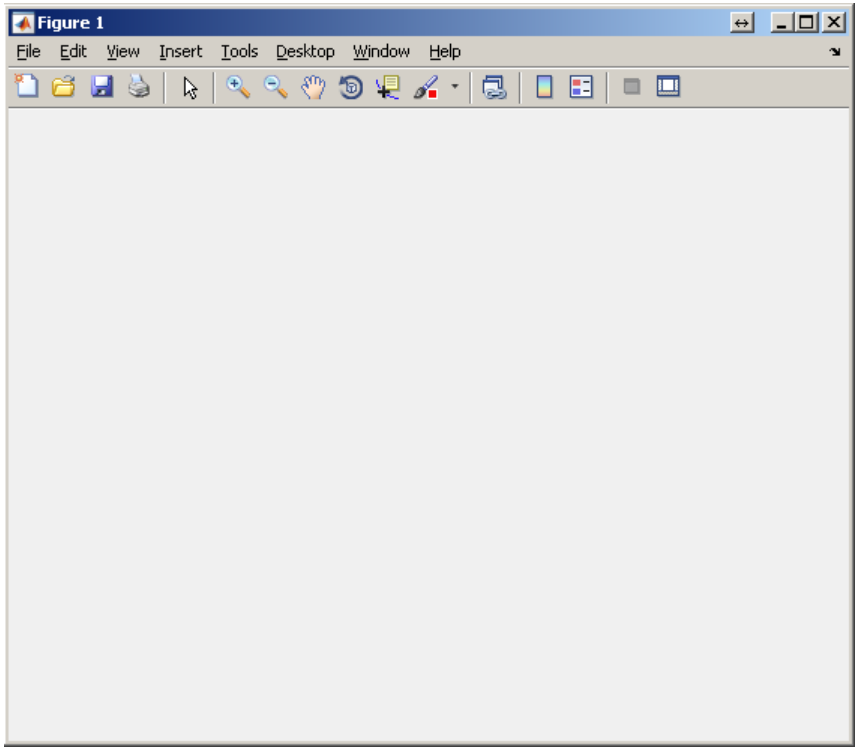
Vlastnosti monitoru, `groot`

- v Matlabu odpovídá obrazovce počítače
- je unikátní a lze volat pomocí funkce
  - `get(0)`
    - ve workspace – datová struktura
  - `groot`
    - ve workspace – handle object
- všechny další objekty jsou potomci

```
>> groot

ans =

  Graphics Root with properties:

          CurrentFigure: []
    ScreenPixelsPerInch: 96
             ScreenSize: [1 1 1920 1200]
       MonitorPositions: [2x4 double]
                  Units: 'pixels'

  Show all properties

          CallbackObject: []
                Children: []
          CurrentFigure: []
     FixedWidthFontName: 'Courier New'
       HandleVisibility: 'on'
       MonitorPositions: [2x4 double]
                 Parent: []
        PointerLocation: [2855 778]
            ScreenDepth: 32
    ScreenPixelsPerInch: 96
             ScreenSize: [1 1 1920 1200]
      ShowHiddenHandles: 'off'
                    Tag: ''
                   Type: 'root'
                  Units: 'pixels'
               UserData: []

>>
```

Grafické okno, `figure`

- objekt `figure` vytváří samostatné grafické okno
  - voláme-li podružnou funkci bez existence okna, vytvoří se nové
  - všechny event. okna jsou potomkem objektu `groot`
  - všechny podružné grafické objekty jsou potomkem objektu `figure` a jsou v daném okně zobrazeny
  - `figure` má mnoho vlastností
    - viz `get(figure)`
    - `a = figure` (bez středníku)



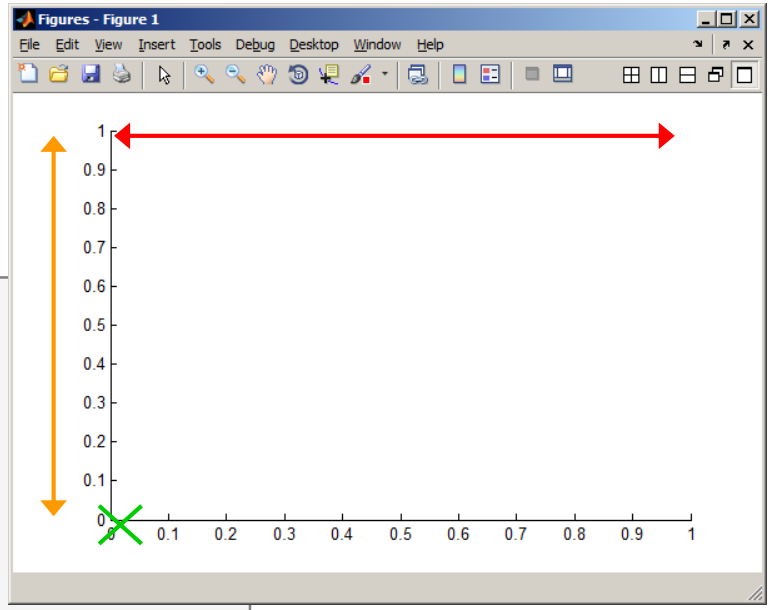
Vlastnost `position`

- Matlab slučuje velikost objektu a jeho umístění do jedné matice
- existují dva způsoby zadání

  (A) absolutně v pixelech

  (B) normalizovaně vzhledem k rozměru nadřazeného objektu

```
[left bottom width height]
```

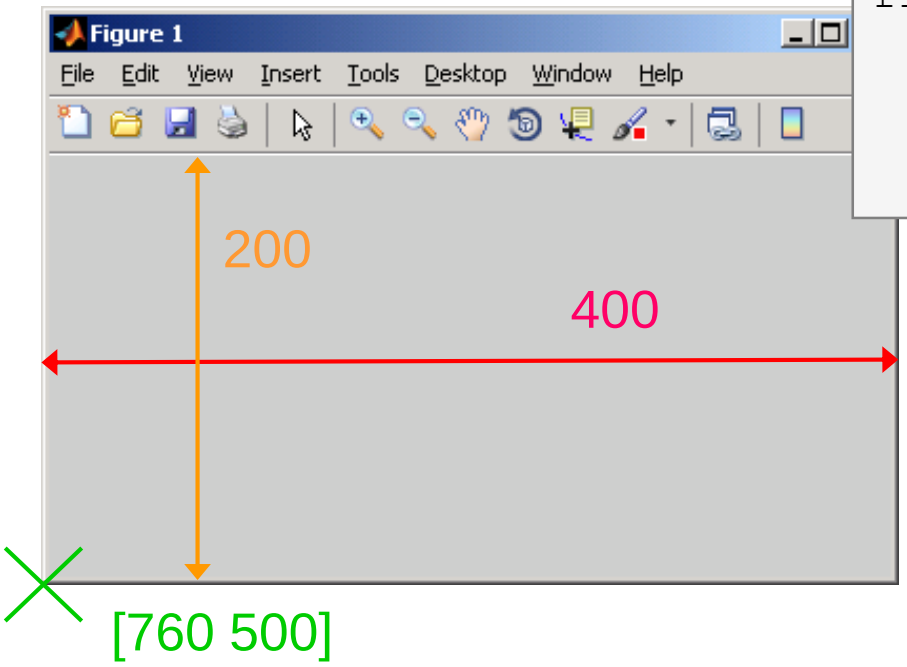



```
%% A)
uicontrol('Units','Pixels',...
          'Style','Text',...
          'Position',[50 150 75 25],...
%% B)
uicontrol('Units','Normalized',...
          'Style','Text',...
          'Position',[0.05 0.12 0.075 0.02],...
```

Tvorba okna

- využijeme, chceme-li např. vytvořit `figure` uprostřed monitoru
  - šířka okna: 400px, výška okna: 200px

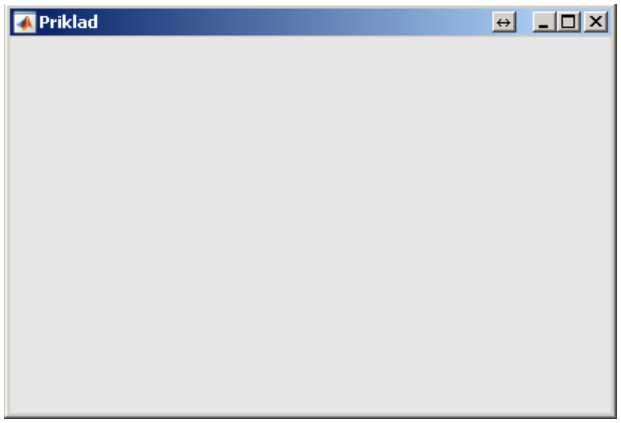
```
dispSize = get(0, 'ScreenSize');
figSize = [400 200];
figHndl = figure('pos', ...
     [(dispSize(3)-figSize(1))/2 ...
      (dispSize(4)-figSize(2))/2 ...
       figSize(1)  figSize(2)]);
```



Cvičení – tvorba okna GUI

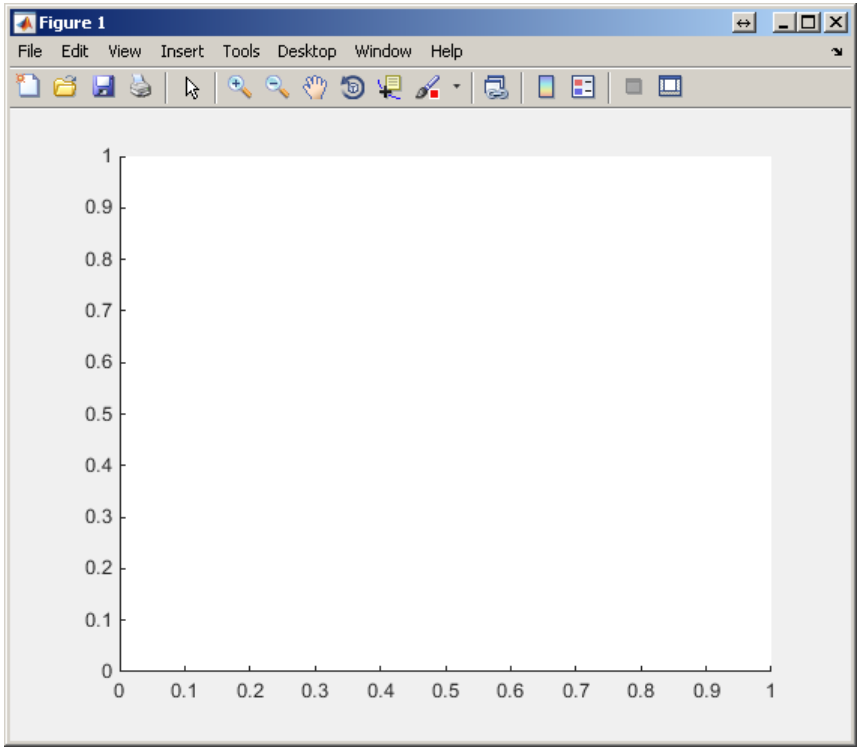
400 s ↑

- v **novém skriptu**, který budeme po zbytek přednášky rozšiřovat, vytvořte okno `figure`, které se bude otevírat vždy svým středem uprostřed monitoru se šířkou 400 pixelů a výškou 250 pixelů
  - zajistěte, aby se okno jmenovalo „Priklad“ a nezobrazovalo se číslo `figure 1`
  - vyžijte vlastnosti `Tag` pro pojmenování (např. `fig_Priklad`)
  - změňte barvu okna (dle vaší volby)



Oblast grafů, `axes`

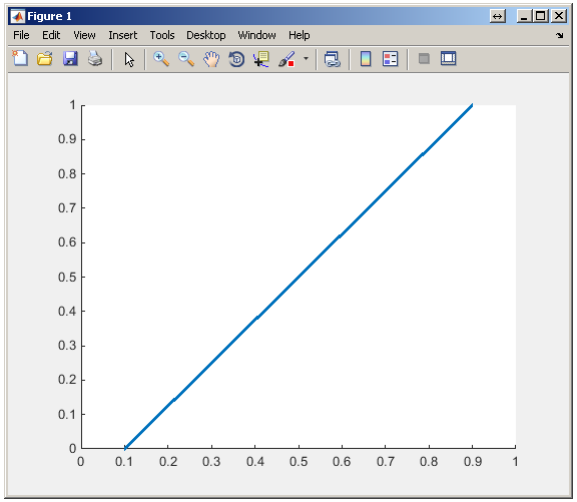
- definuje oblast v grafickém okně, kam jsou umísťovány potomci objektu `axes`
- všechny podružné objekty k objektu `axes` generují osy i pokud zatím neexistují (podobně jako `figure`)
- `axes` má mnoho vlastností
  - viz `get(axes)`

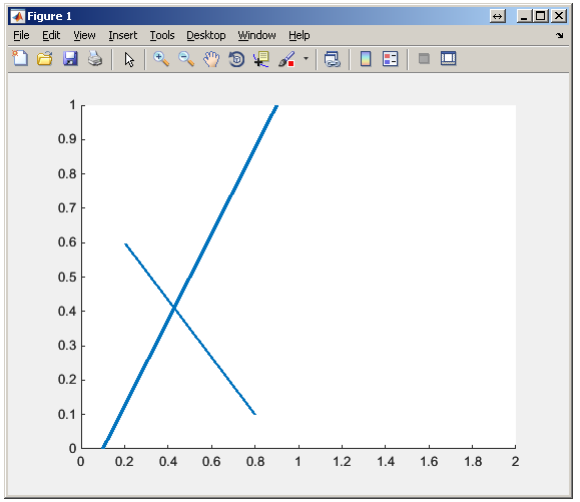


Funkce `axis`

- `axis` nastavuje rozsah osy `axes`
  - formát (2D): `[x_od x_do y_od y_do]`
  - formát (3D): `[x_od x_do y_od y_do z_od z_do]`

```
line([0.1 0.9], [0 1], 'LineWidth', 3)
axis([0 1 0 1])
```



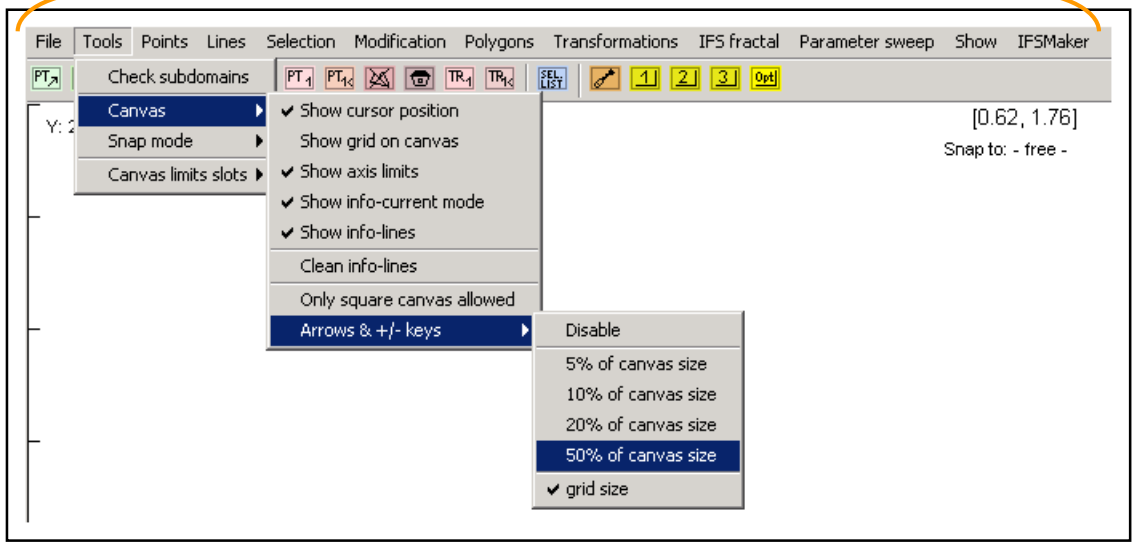



```
line([0.8 0.2], [0.1 0.6], 'LineWidth', 2)
axis([0 2 0 1])
```

Skupina uiobjects: `uimenu`

- lze definovat klávesové zkratky (např. CTRL+L)
- v menu se lze pohybovat pomocí ALT+písmeno
- lze přiřadit `callback` funkci

490 řádek kódu

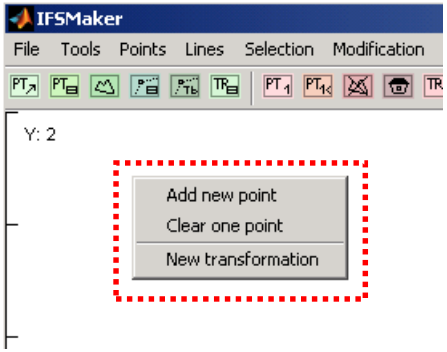


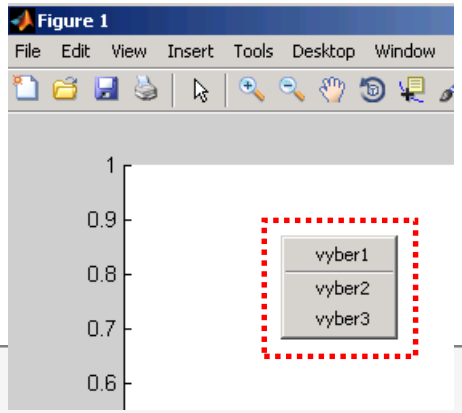
- více viz `help uimenu`

uiobjects
uimenu
uicontextmenu
uitoolbar
uipanel
uitabgroup
uitable
uibuttongroup
actxcontrol
uicontrol

Skupina uiobjects: `uicontextmenu`

- vytvoří kontextové menu
  - lze ho vyvolat kliknutím pravého tlačítka myši
  - výběr položky z menu aktivuje zadaný callback





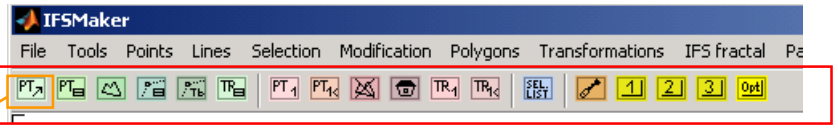
uiobjects
- uimenu
- **uicontextmenu**
- uitoolbar
- uipanel
- uitabgroup
- uitable
- uibuttongroup
- actxcontrol
- uicontrol

```
figHndl = figure;
cMenu   = uicontextmenu;
axsHndl = axes('Parent',figHndl,'UIContextMenu',cMenu);
uimenu(cMenu,'Label','vyber1','Callback',@callbackFcn1);
uimenu(cMenu,'Label','vyber2','Callback',@callbackFcn2,...
    'Separator','on');
uimenu(cMenu,'Label','vyber3','Callback',@callbackFcn3);
```

Skupina uiobjects: `uitoolbar`

- v Matlabu lze vytvářet vlastní ikony do menu
  - není složité ale mimo rámec kurzu
  - zájemci si mohou projít >> `doc uitoolbar`

takto je ikona ‘nakreslena‘ v m-souboru



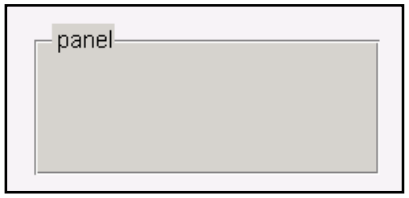

```
pics = [1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1;...
        1 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 1;...
        1 .6  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 .6 1;...
        1 .6  1  0  0  0  1  0  0  0  0  0  1  1  1  1  1  1 .6 1;...
        1 .6  1  0  1  1  0  1  1  0  1  1  1  1  1  1  1  1 .6 1;...
        1 .6  1  0  1  1  0  1  1  0  1  1  1  1  1  1  1  1 .6 1;...
        1 .6  1  0  0  0  1  1  1  0  1  1  1  1  1  1  1  1 .6 1;...
        1 .6  1  0  1  1  1  1  1  0  1  1  1  0  0  0  0  1 .6 1;...
        1 .6  1  0  1  1  1  1  1  0  1  1  1  1  1  0  0  1 .6 1;...
        1 .6  1  0  1  1  1  1  1  0  1  1  1  1  0  1  0  1 .6 1;...
        1 .6  1  1  1  1  1  1  1  1  1  1  1  1  0  1  1  0  1 .6 1;...
        1 .6  1  1  1  1  1  1  1  1  1  1  1  0  1  1  1  1  1 .6 1;...
        1 .6  1  1  1  1  1  1  1  1  1  1  0  1  1  1  1  1  1 .6 1;...
        1 .6  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 .6 1;...
        1 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 .6 1;...
        1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1];
icon(:,:,1) = .85. * pics;
icon(:,:,2) = .98. * pics;
icon(:,:,3) = .85. * pics;
```

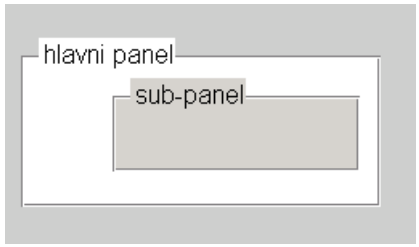
- uiobjects
  - uimenu
  - uicontextmenu
  - uitoolbar
  - uipanel
  - uitabgroup
  - uitable
  - uibuttongroup
  - actxcontrol
  - uicontrol

- způsoby vkládání ikon
  - >> `doc uipushtool`
  - >> `doc uitoggletool`

Skupina uiobjects: `uipanel`

- vytvoří panel, který je rodičem dalším objektům
- objekty uvnitř jsou orientovány vzhledem k panelu
- lze nastavovat mnoho vlastností (viz >> `doc uipanel`)





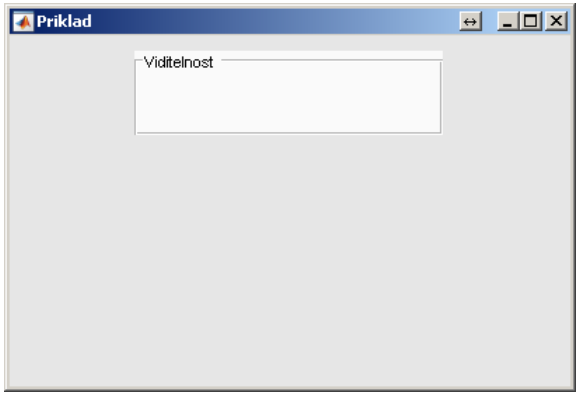

```
fgHnd = figure;
h1p   = uipanel('Title', 'hlavni panel', ...
        'FontSize', 12, 'BackgroundColor', ...
        'white', 'Position', [0.25 0.25 0.4 0.25]);
h2p   = uipanel('Parent', h1p, ...
        'Title', 'sub-panel', 'FontSize', 12, ...
        'Position', [0.25 0.25 0.7 0.7]);
```

uiobjects
uimenu
uicontextmenu
uitoolbar
uipanel
uitabgroup
uitable
uibuttongroup
actxcontrol
uicontrol

Cvičení – `panel`

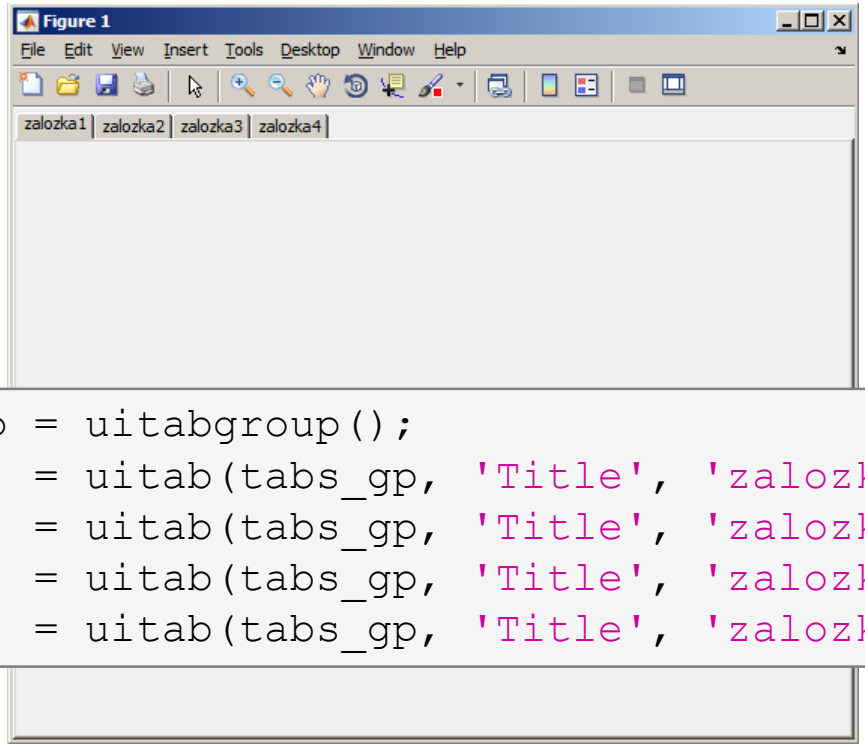
400 s ↑

- vytvořte panel, který bude na pozici `[90 180 220 60]` px
- pojmenujte si svůj panel „Viditelnost“, nastavte tag na „panelViditelnost“
- určete jeho barvu a tu uložte do proměnné, kterou dále budeme používat pro sjednocení barev ostatních objektů v rámci panelu



Skupina uiobjects: `uitab`

- vytvoří záložku, která bude rodičem pro ostatní objekty (stejně jako u panelu)
- více >> `doc uitabgroup`



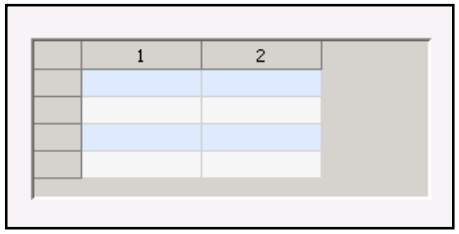

```
tabs_gp = uitabgroup();
tabs_1  = uitab(tabs_gp, 'Title', 'zalozka1');
tabs_2  = uitab(tabs_gp, 'Title', 'zalozka2');
tabs_3  = uitab(tabs_gp, 'Title', 'zalozka3');
tabs_4  = uitab(tabs_gp, 'Title', 'zalozka4');
```

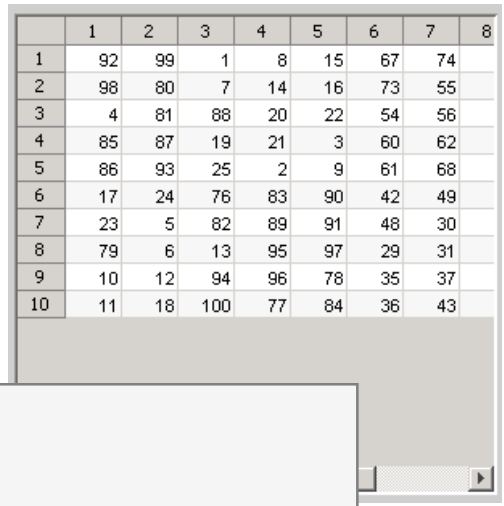
uiobjects

- uimenu
- uicontextmenu
- uitoolbar
- uipanel
- **uitabgroup**
- uitable
- uibuttongroup
- actxcontrol
- uicontrol

Skupina uiobjects: `uitable`

- vytvoří 2D tabulku
  - může být umístěna kdekoli v okně `figure`
  - má celou řadu vlastností i prvků (check, popup)
- viz `>> doc uitable`





	1	2	3	4	5	6	7	8
1	92	99	1	8	15	67	74	
2	98	80	7	14	16	73	55	
3	4	81	88	20	22	54	56	
4	85	87	19	21	3	60	62	
5	86	93	25	2	9	61	68	
6	17	24	76	83	90	42	49	
7	23	5	82	89	91	48	30	
8	79	6	13	95	97	29	31	
9	10	12	94	96	78	35	37	
10	11	18	100	77	84	36	43	

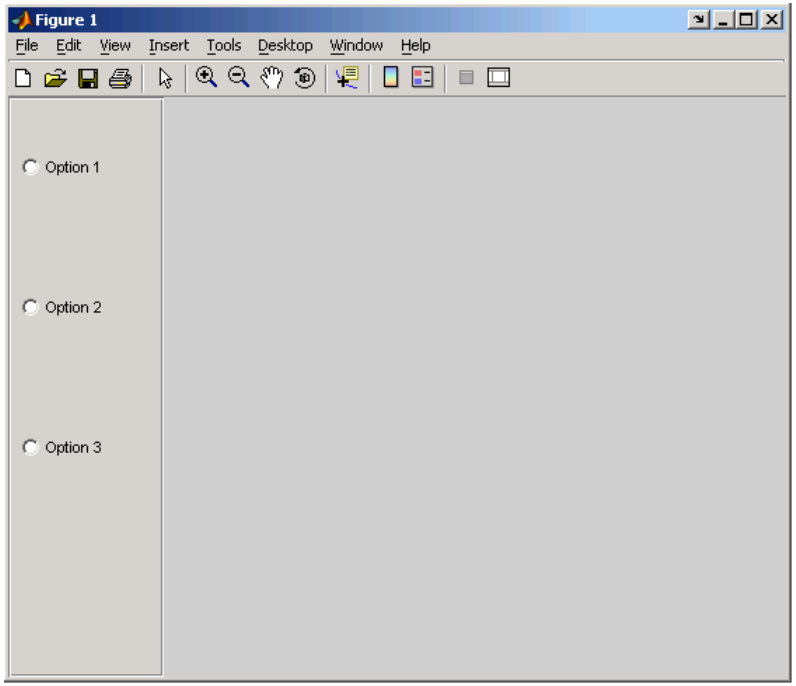
```
>> figure
>> t = uitable;
>> set(t, 'Data', magic(10));
>> set(t, 'ColumnWidth', {35})
```

uiobjects

- uimenu
- uicontextmenu
- uitoolbar
- uipanel
- uitabgroup
- **uitable**
- uibuttongroup
- actxcontrol
- uicontrol

Skupina uiobjects: `uibuttongroup`

- blok se skupinou tlačítek
- více >> `doc uibuttongroup`



uiobjects
uimenu
uicontextmenu
uitoolbar
uipanel
uitabgroup
uitable
uibuttongroup
actxcontrol
uicontrol

Skupina uiobjects: `actxcontrol`

- umožní vytvořit Microsoft ActiveX control ve `figure` okně
  - seznam podporovaných Microsoft ActiveX control

```
>> list = actxcontrollist
>> h    = actxcontrolselect
```

  - příklady
    - webový prohlížec

```
>> h = actxcontrol('AcroPDF.PDF.1',...
```

    - PDF reader

```
>> h = actxcontrol('Shell.Explorer.2',...
```

  - více informací

```
>> doc getting started with COM
```

Skupina uiobjects: uicontrol

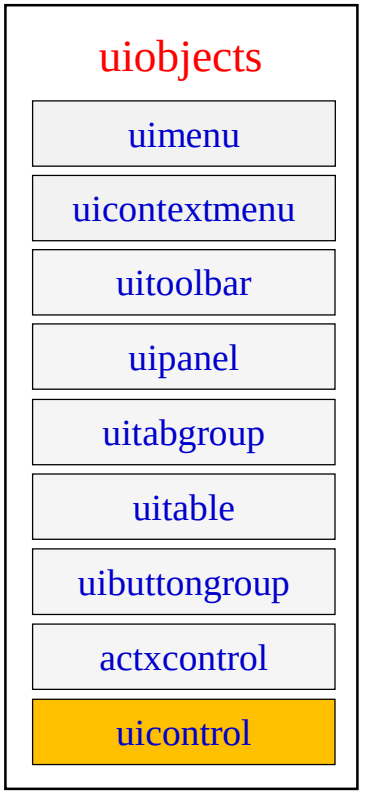
- uicontrol vytváří základní funkční prvky GUI
- ke změně stylu uicontrol použijeme vlastnost style

```
>> t = uicontrol;
>> set(t, 'Style', 'text');
```

- pro zjištění vlastností uicontrol využijeme

```
>> get(t);
```

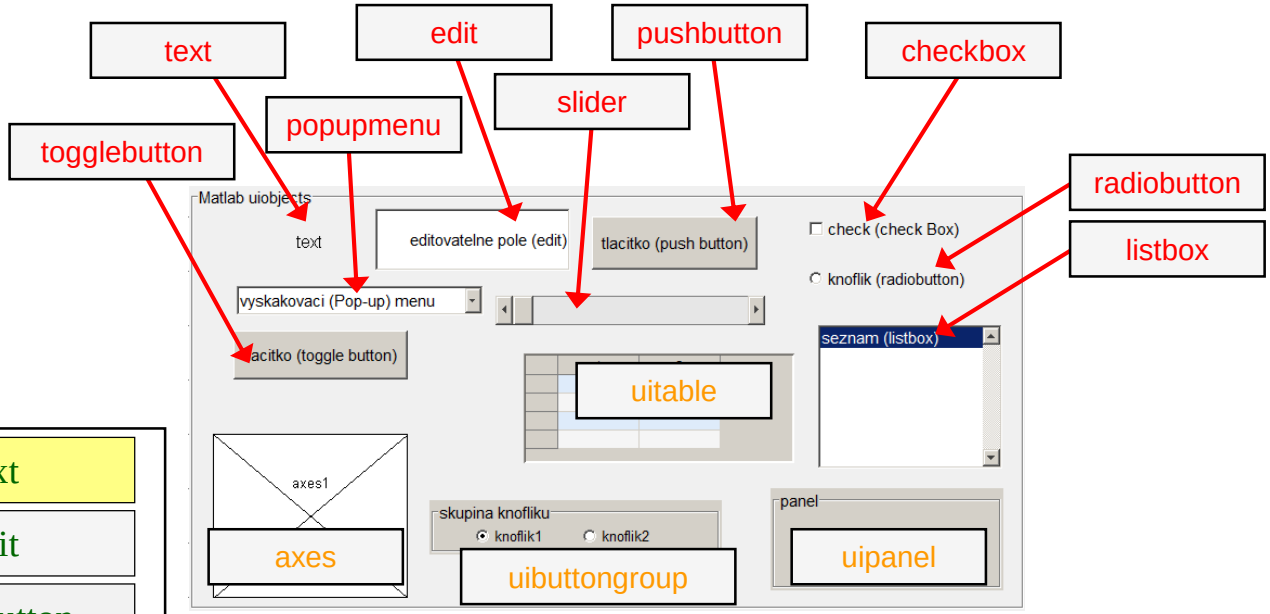
- více viz >> doc uicontrol



Skupina uiobjects: uicontrol

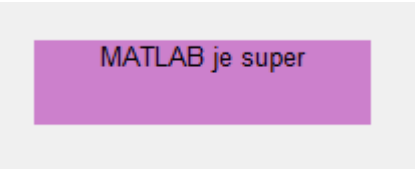
uiobjects
- uimenu
- uicontextmenu
- uitoolbar
- uipanel
- uitabgroup
- uitable
- uibuttongroup
- actxcontrol
- uicontrol

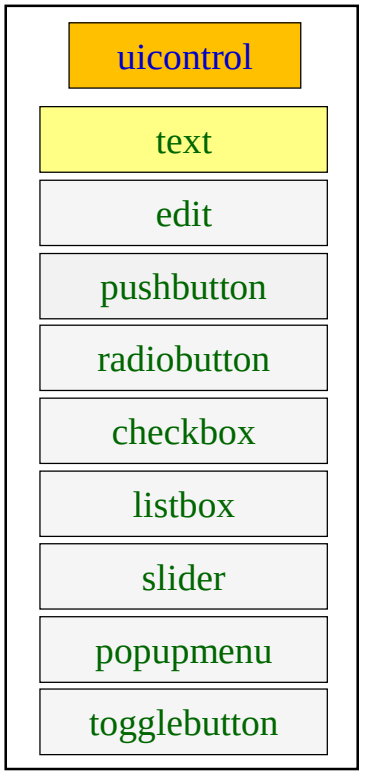
- text
- edit
- pushbutton
- radiobutton
- checkbox
- listbox
- slider
- popupmenu
- togglebutton



Skupina uicontrol: `text`

- umístí na zvolené místo text
- zpravidla se využívá
  - jako popiska dalších prvků
  - informační text pro uživatele



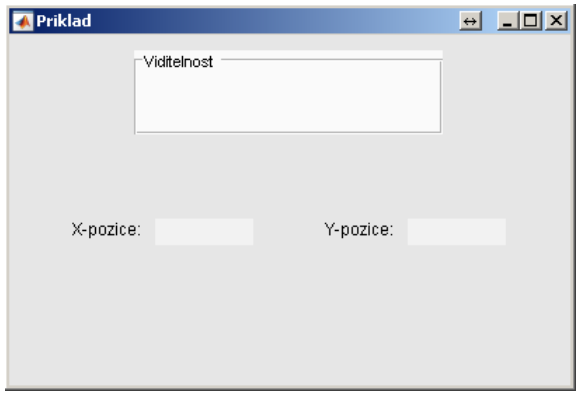



```
>> figure
>> text1 = uicontrol(...
    'Units', 'Normalized', ...
    'Style', 'Text', ...
    'Position', [0.35 0.45 0.3 0.1], ...
    'Tag', 'MTB', ...
    'FontSize', 10, ...
    'BackgroundColor', [0.8 0.5 0.8], ...
    'HorizontalAlignment', 'center', ...
    'String', 'MATLAB je super');
```

Cvičení – `text`, zadání

400 s ↑

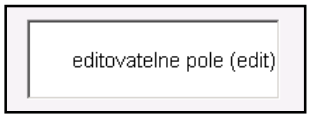
- vytvořte 4 textové pole, které budou pomocí normovaných hodnot umístěny na pozice s vlastnosti
  - `[0.1  0.4 0.15  0.075]  font  9  barva 1`
  - `[0.26 0.4 0.175 0.075]  font 10  barva 2`
  - `[0.55 0.4 0.15  0.075]  font  9  barva 1`
  - `[0.71 0.4 0.175 0.075]  font 10  barva 2`
- textová pole s `barva 1` zobrazte s textem `X-pozice/Y-pozice` ostatní nechte bez textu
- každému textovému poli přiřaďte svůj handle

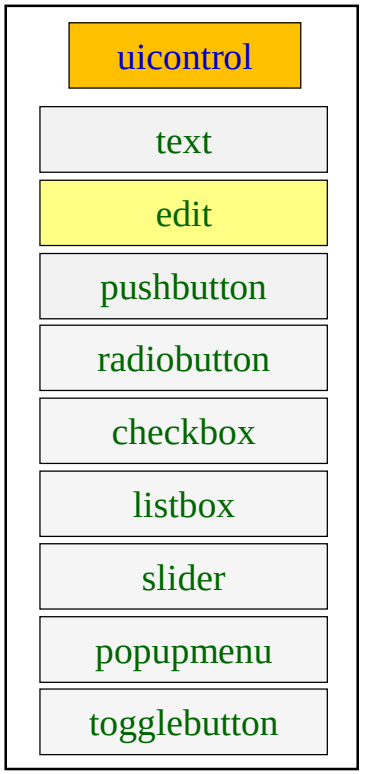


Cvičení – `text`, řešení

skupina uicontrol: `edit`

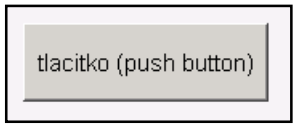
- umožňuje načíst řetězec
  - řetězec je typu `string`
  - je třeba řetězec zpracovat (`str2num`, `str2double`,...)
- uživatel může pracovat i pomocí CTRL+C,+V,+X,+A,+H
- pomocí `edit` lze vytvořit v Matlabu např. konzole

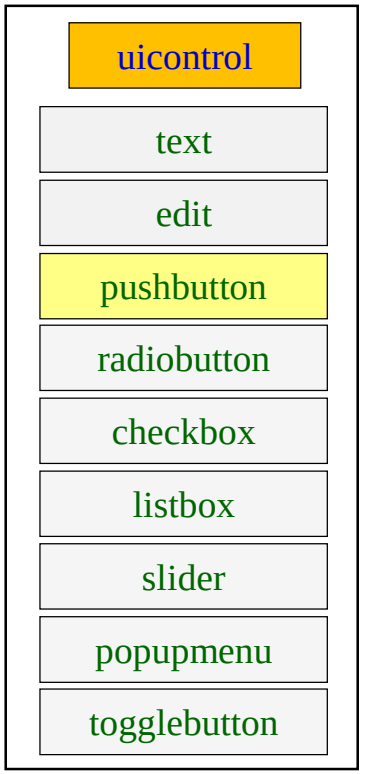




skupina uicontrol: `pushbutton`

- jednostavové tlačítko
- jeho stisknutím je vyvolána `callback` funkce
- nastavení vzhledu je podobné jako u objektu `text`

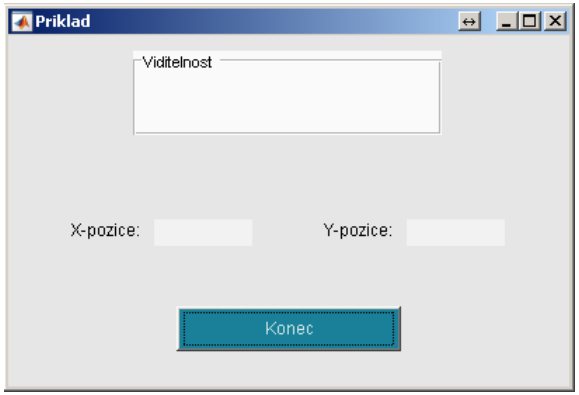




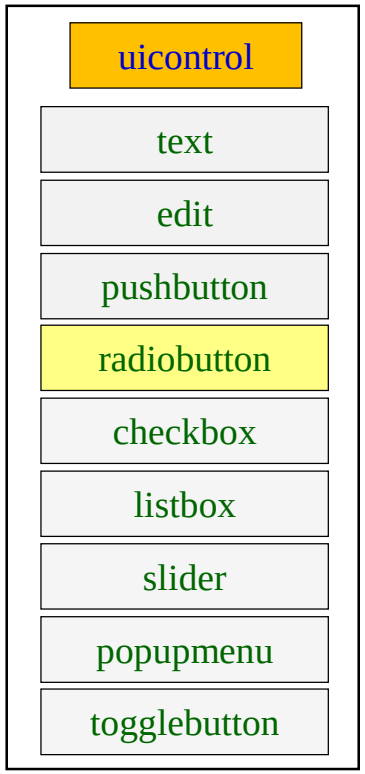
Cvičení – tlačítko

400 s ↑

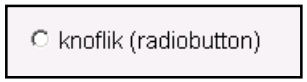
- vytvořte tlačítko s nápisem „konec“
  - vložte ho na (normovanou) pozici `[0.3 0.1 0.4 0.125]`
  - velikost písma zvolte 9
  - barva pozadí: `[0.1 0.5 0.6]`
  - barva textu: `[0.8 0.9 0.9]`



Skupina uicontrol: `radiobutton`

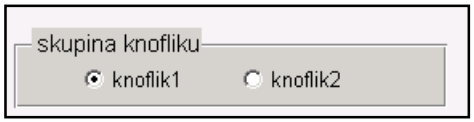


- stavy typu (vybrán – nevybrán)

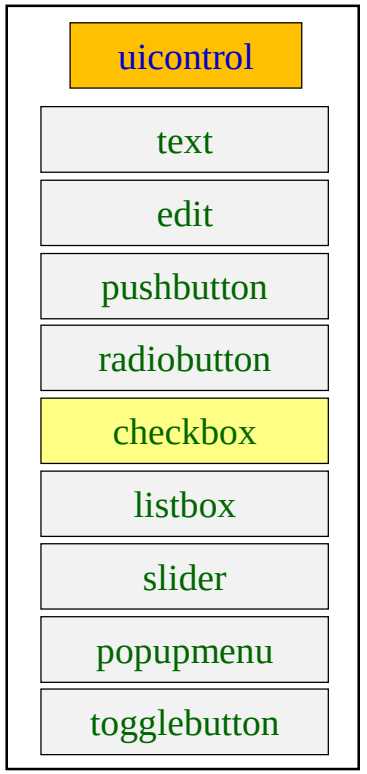


- tyto prvky lze sdružovat do větších skupin
  - button group (objekt `uibuttongroup`)

- `callback` funkce dokáže zjistit dokonce přechod prvků



Skupina uicontrol: checkbox



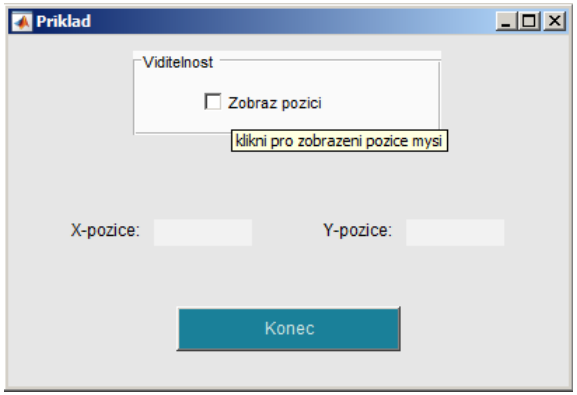
- podobné jako `radiobutton`
- zaškrtávací políčko (s doprovodným textem)
- `callback` vyvolán změnou stavu


```
function checkboxFcn(hObj) % osetreno
%% zjisteni, zda je check vybran
if (get(hObject,'Value') % vybran
  % ...
else % nevybran
  % ...
end
```

Cvičení – zatržítko

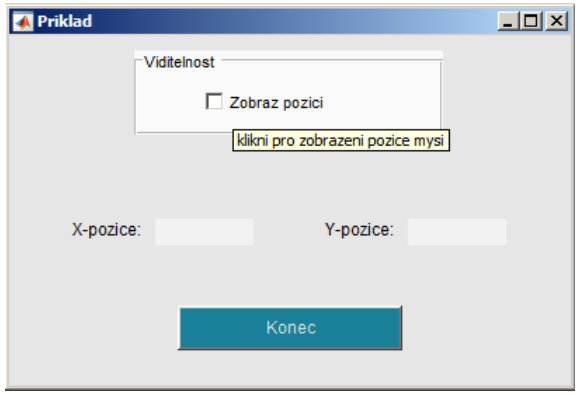
400 s ↑

- vytvořte checkbox , bude umístěn uvnitř panelu `panel1`
- popisek bude „Zobraz pozici “
  - zajistěte, aby ze zobrazovala nápověda (anglický ekvivalent popisku) bude-li kurzor myši poblíž grafického prvku
- přiřaďte prvku vlastní tag
- barvu pozadí nastavte stejnou jako v případě panelu



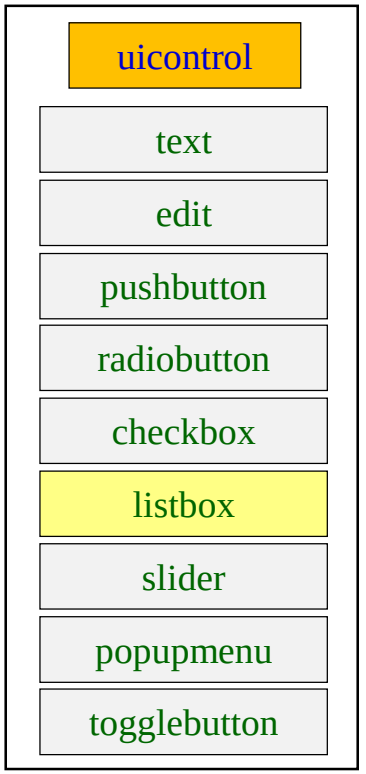
Cvičení

- soubor s GUI si někam uložte (budeme ho potřebovat příští hodinu)

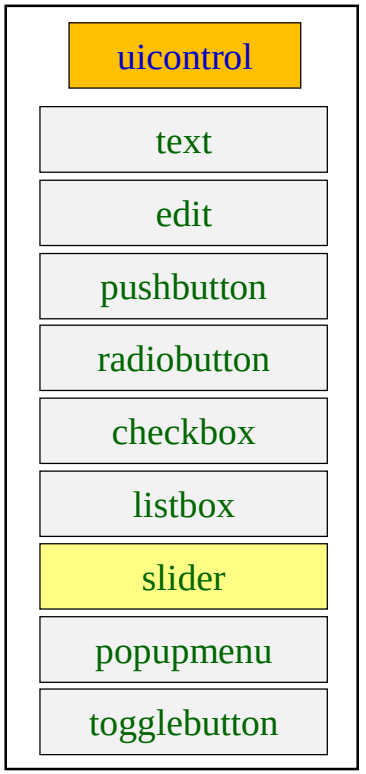


Skupina uicontrol: `listbox`

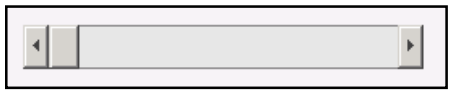
- zobrazuje seznam, lze vybrat jednu nebo i více položek
- vlastnost `string` obsahuje seznam řetězců (položek)
- vlastnost `value` obsahuje matici vybraných položek
- hodnoty `max` a `min` ovlivňují výběr



Skupina uicontrol: `slider`

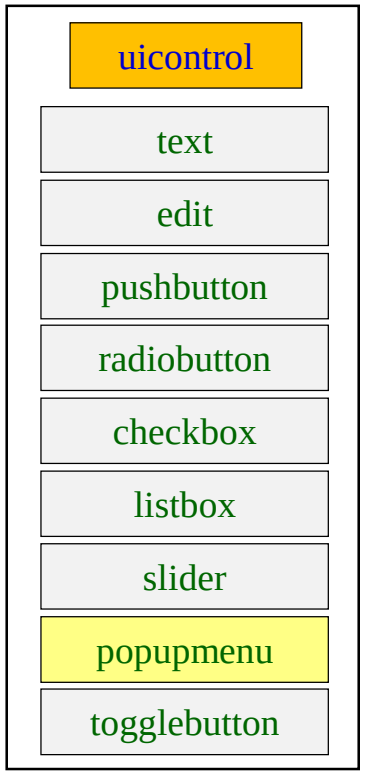


- na vstupu numerický rozsah (`max` a `min`)
- uživatel se pohybuje po skocích (`sliderstep`) jezdcem
- požaduje
  - rozsah
  - krok tažením
  - krok kliknutím
  - počáteční hodnota

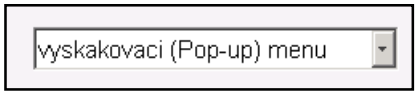


```
slider_step(1) = 0.4/(10-2);
slider_step(2) = 1/(10-2);
set(sliderHndl,'sliderstep',...
    slider_step,'max',10,...
    'min',2,'Value',6.5);
```

Skupina uicontrol: `popupmenu`



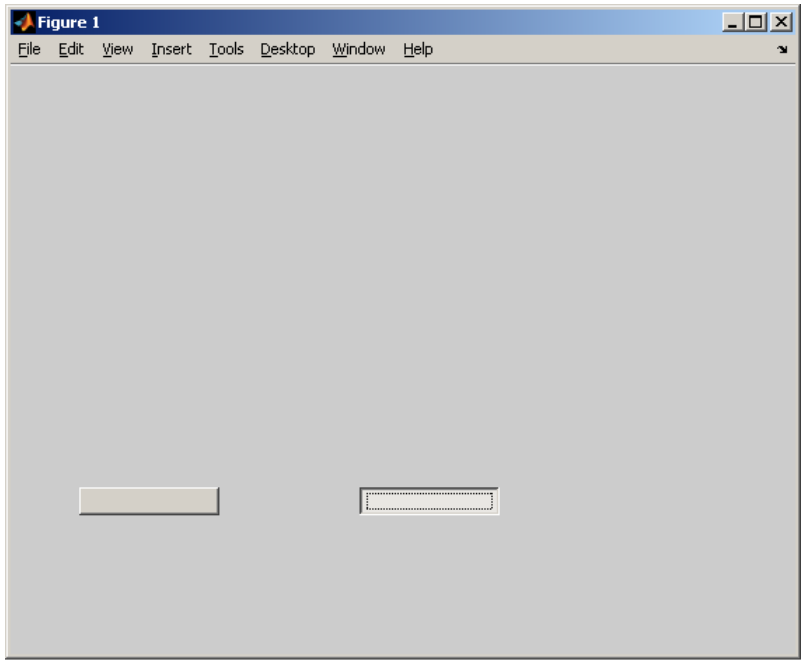
- klik na šipku zobrazí seznam, z něj lze vybírat položku
  - `string` obsahuje seznam řetězců
  - `value` obsahuje index vybrané položky
- více informací >> `doc uicontrol`

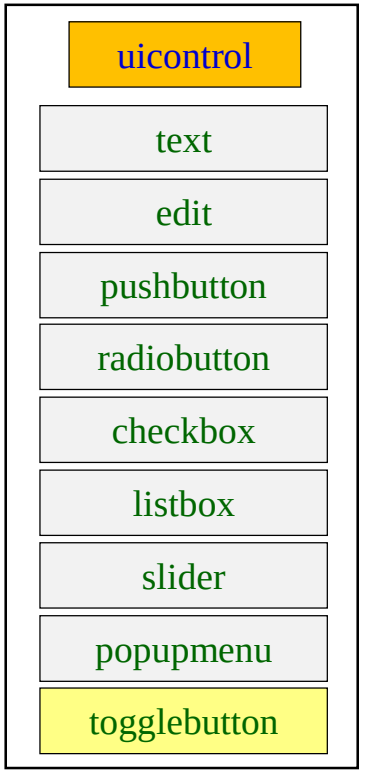



```
function popupFcn(hObj) % osetreno
val = get(hObj,'Value');

string_list     = get(hObj,'String');
selected_string = string_list{val};
% ...
```

Skupina uicontrol: `togglebutton`

- přepínací tlačítko
  - po kliknutí zůstane zakliknuté
- více informací >> `doc uicontrol`





Callback funkce

- nad každým objektem jsou definované operace, které může uživatel využít (klik na tlačítko, výběr z nabídky, …)
- tyto operace jsou obsluhovány pomocí tzv. callback funkcí
  - jinak řečeno, pokud uživatel klikne na tlačítko, aktivuje se callback funkce této události (je-li definována).
- pokud nemá být GUI statický, musí vždy obsahovat alespoň jednu callback funkci
- callback funkce je uložena jako vlastnost objektu – lze je měnit, mazat, kopírovat atp.

callback

- ukazuje na funkci, která je provedena, je-li `uicontrol` aktivován
  - je několik možností, jak `callback` zapisovat – nejobecnější je pomocí „handle funkce“

```
function GUI
uicontrol('Units', 'Normalized', 'Style', 'Push', 'String', ...
    'Stiskni', 'Callback', @pressButton, 'ForegroundColor', ...
    'white', 'BackgroundColor', [0.7 0.2 0], 'Fontweight', 'bold', ...
    'Fontsize', 11, 'Position', [0.1 0.65 0.15 0.1]);
end
function pressButton(scr,event)
% scr a event jsou defaultní parametry vracené funkcemi callback
% scr   - zdroj callbacku (v tomto případě handle objekt tlačítka)
% event - info o vzniklé události, pro některé objekty (radiobutton)

  disp(scr);   % kontrolní výpis - handle objektu
  disp(event); % žádný event není Matlabem vypsán
  set(scr, 'String', 'Done', 'Fontsize', 9, ...
  'BackgroundColor', [0.2 0.7 0]);
end
```

Probrané funkce

`get, set`	zjišťuje či mění vlastnosti objektů	•
`subplot`	vkládání více grafů do jednoho okna	•
`plotyy, semilogy, semilogx, loglog,`	2D grafy s upravenou osou/osami	•
`pie, stairs, contour, quiver`	2D grafy	•
`image, imagesc`	vykreslení matice jako obrázek	•
`pie3, mesh, slice, scatter`	3D grafy	•
`colormap`	změní colormap obrázku	•
`view`	definuje pohled pro 3D graf	•
`axis`	nastavuje pevný rozsah os	•

Cvičení 1 – callback tlačítka

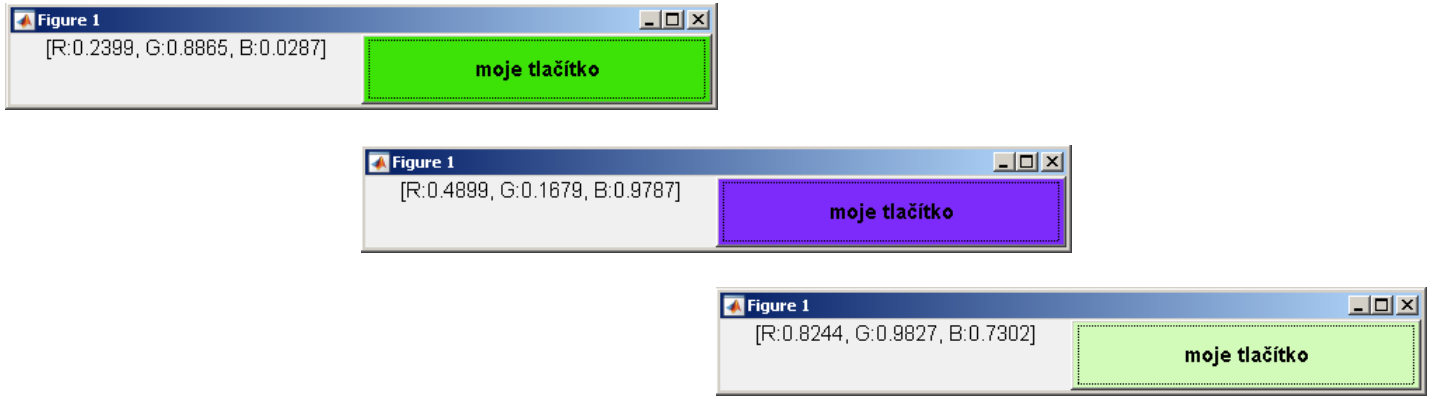
600 s ↑

- vytvořte tlačítko, které mění barvu po kliknutí

Cvičení 2 – tlačítko

400 s ↑

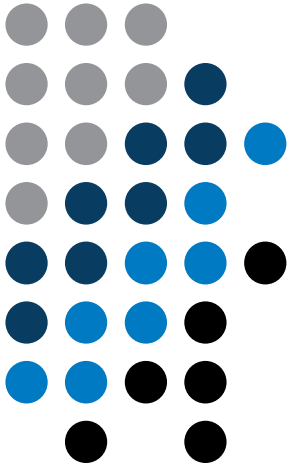
- rozšiřte Callback funkci tak, aby
  - ta vygeneruje náhodný vektor RGB barvy
  - v textovém poli vypíše jednotlivé složky RGB
  - změní pozadí tlačítka na onu vygenerovanou barvu



Cvičení 2 – tlačítko, řešení

Děkuji!

ver. 3.1 (21/04/2015)
Miloslav Čapek
miloslav.capek@fel.cvut.cz



Materiál vytvořen v rámci předmětu A0B17MTB.