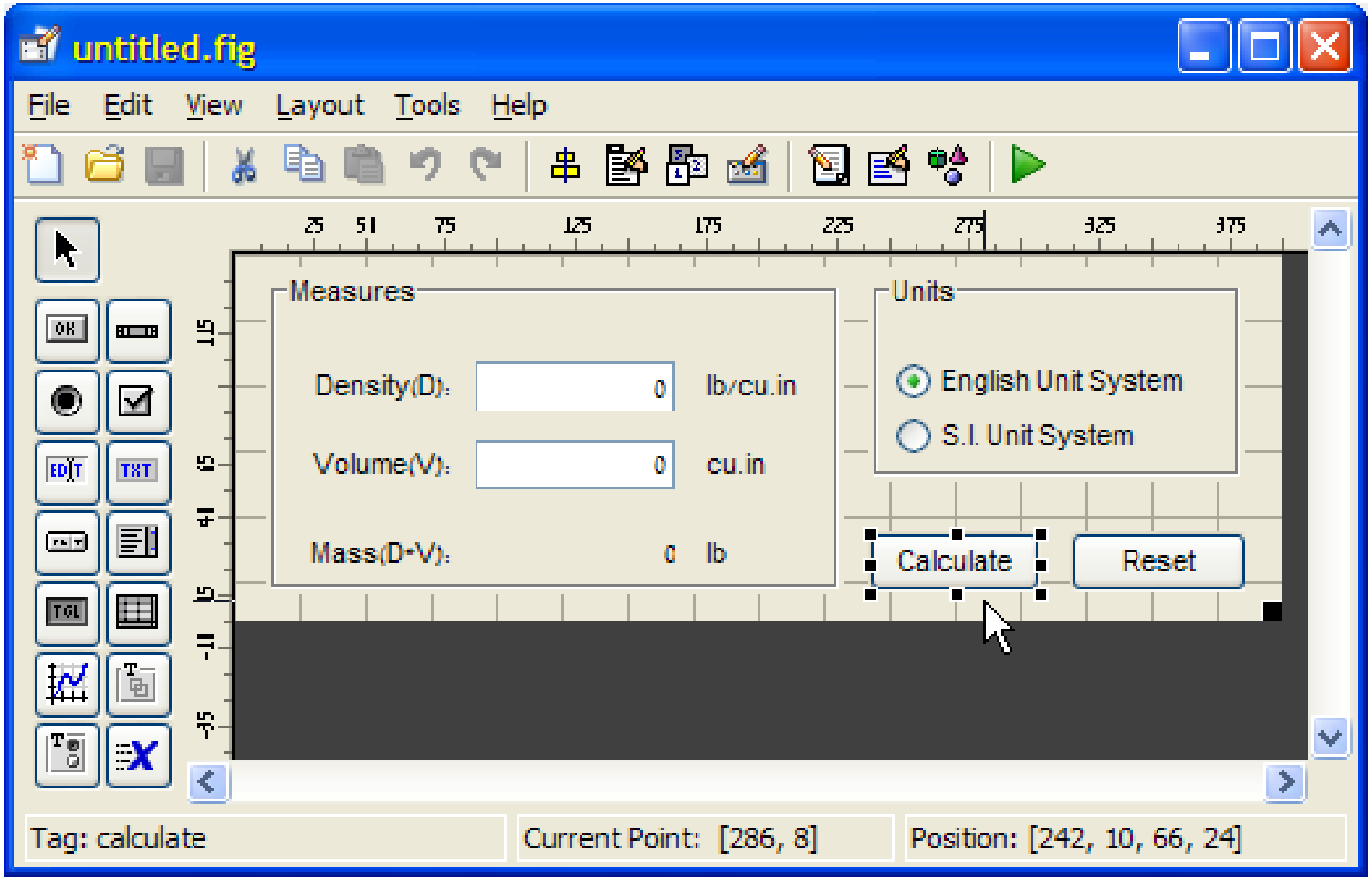
Advanced Matlab GUI

Open Day – Jan 8, 2013

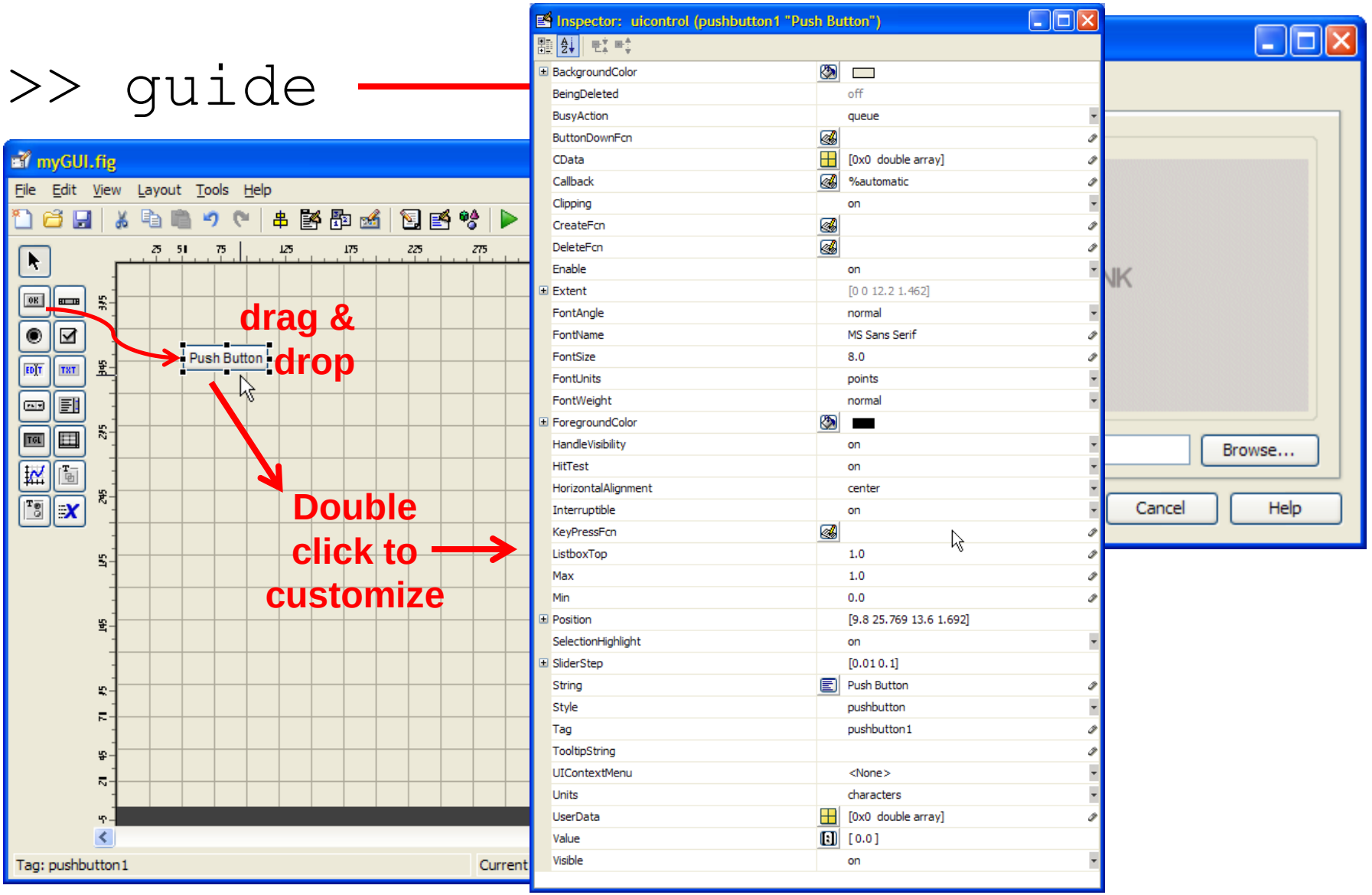
http://UndocumentedMatlab.com/files/OpenDay.zip

GUIDE: Graphical User Interface Design Env



GUIDE: Creating a simple GUI

```
>> guide
```



Available uicontrols

- Push-button
- Slider (scrollbar)
- Radio button
- Checkbox
- Editbox
- Text label
- Popup menu (combo-box, drop-down)
- Listbox
- Toggle button
- Uitable (R2008a+)
- Plot axes
- Panel
- Button group
- ActiveX control

Advanced GUI topics

- The GUIDE-generated file duo
- OpeningFcn vs. OutputFcn
- Setting and using callbacks
- Callback event handling, reentrancy, dynamic data
- The `hObject` parameter
- The `eventdata` parameter
- The `handles` struct
- `guihandles` vs. `guidata`
- Controlling handles visibility

Using HTML

- All Matlab GUI is based on Java Swing
  - Java Swing supports HTML labels + extensive CSS subset
- HTML processing is CPU intensive – bad for performance
  - Especially bad if recurring multiple times (table/listbox cells)
    - Use only if <20-30 elements
  - Alternatives: built-in Font* properties; Java cell renderers
- Case-insensitive HTML tags, attributes
- Required: <html> prefix at the string's beginning
- Not required: closing tags (</html>)
- Note: HTML processing may modify some display aspects
  - Font, background color, label margins (visible in tooltips)

Using HTML – some examples

- Listbox

```
uicontrol('Style','list', 'Position',[10,10,70,70], 'String', ...
{'<HTML><FONT color="red">Hello</Font></html>', 'world', ...
'<html><font style="font-family:impact;color:green"><i>What a', ...
'<Html><FONT color="blue" face="Comic Sans MS">nice day!</font>'});
```

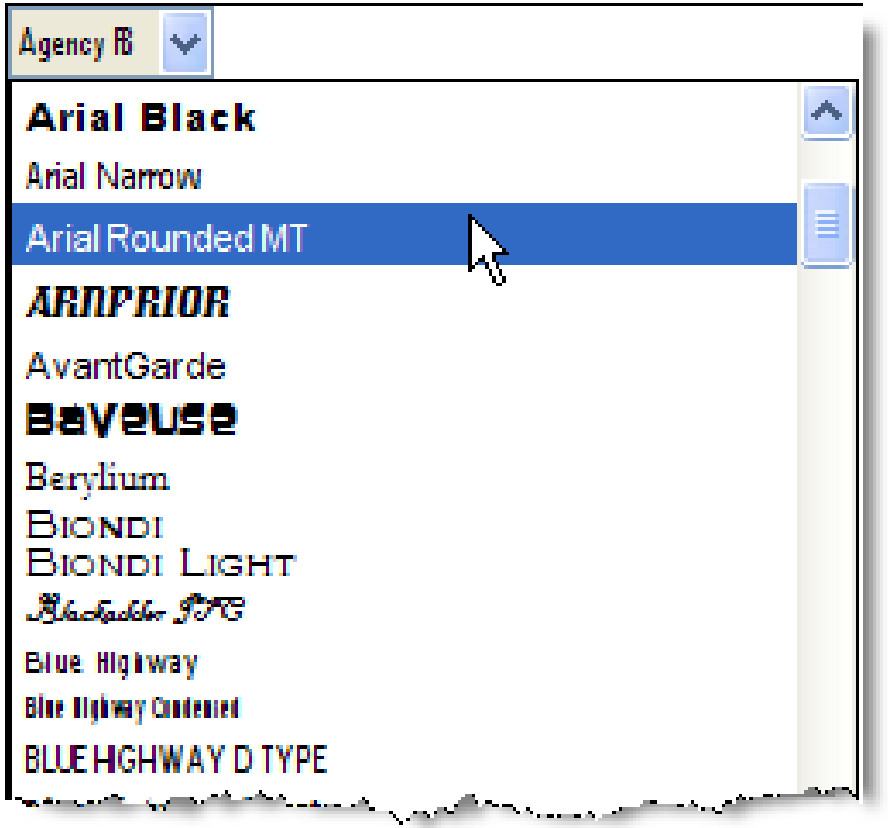
- Combo-box

```
uicontrol('Style','popup', 'Position',[10,10,150,100], 'String', ...
{'<HTML><BODY bgcolor="green">green background</BODY></HTML>', ...
'<HTML><FONT color="red" size="+2">Large red font</FONT></HTML>', ...
'<HTML><BODY bgcolor="#FF00FF"><PRE>fixed-width font'});
```

Using HTML – some examples

- Fonts in combo-box

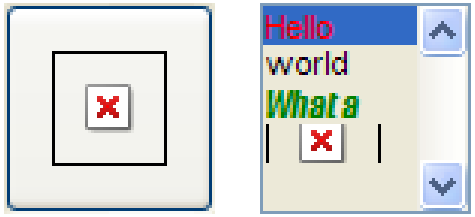
```
fontStr = @(font) ['<html><font face="' font '">' font '</font></html>'];
htmlStr = cellfun(fontStr,listfonts,'uniform',false);
uicontrol('style','popupmenu', 'string',htmlStr, 'pos',[20,350,60,20]);
```



Using HTML images

- HTML img src must have full pathname in URI format:

```
>> uicontrol('Position',..., 'String','<Html><img src="img.gif">');
>> uicontrol('Style','list', ... '<Html><img src="img.gif"/>'});
```

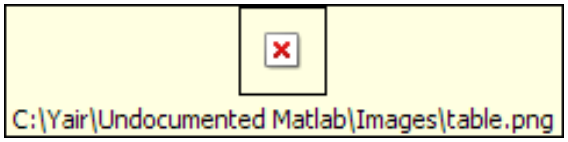



```
>> iconsFolder = fullfile(matlabroot,'/toolbox/matlab/icons/');
>> iconUrl = strrep(['file:/' iconsFolder 'matlabicon.gif'],'\','/');
>> str = ['<Html><img src="' iconUrl '">']
str = <Html><img src="file:/C:/Program Files/MATLAB/.../icons/matlabicon.gif">
>> uicontrol('Position',..., 'String',str);
>> uicontrol('Style','list', ... str});
```

Using HTML images

- HTML images can also be placed in tooltips:

```
filePath = 'C:\Yair\Undocumented Matlab\Images\table.png';
str = ['<html><center><img src="' filePath '"><br>' filePath];
set(hButton,'tooltipString',str);
```




```
filePath = 'C:\Yair\Undocumented Matlab\Images\family.jpg';
filePath = strrep(['file:/' filePath],'\','/');
str = ['<html><center><img src="' filePath '"><br>' ...
       '<b><font color="blue">' filePath];
set(hButton,'tooltipString',str);
```

Hidden properties

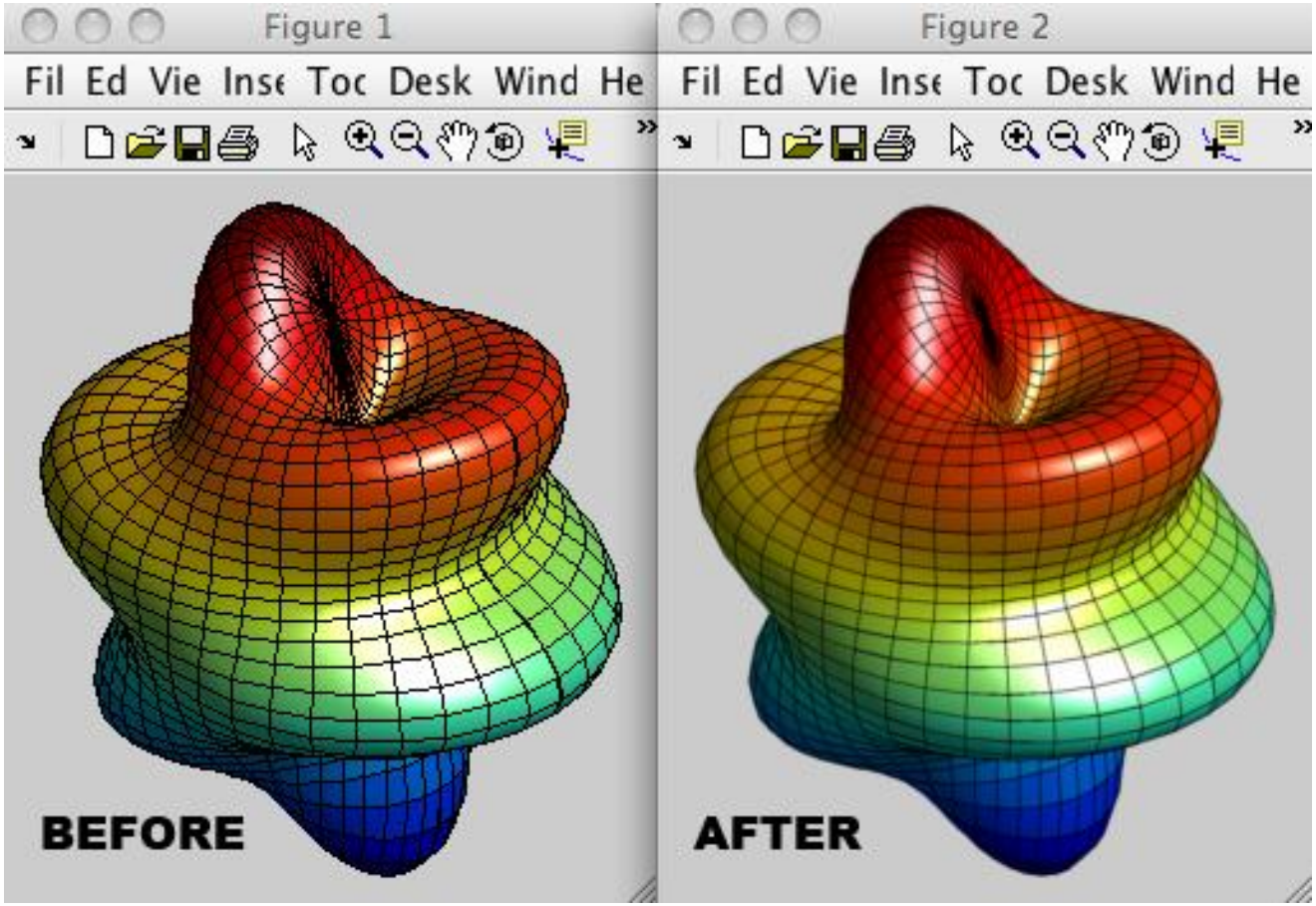
- Most Matlab graphics components (GUI, plots, …) have hidden unsupported/undocumented properties
- These properties behave exactly like regular properties, except that they are not displayed by default in `get`/`set`
- To display these properties:
  - `set(0,'HideUndocumented','off');  % default='on'`
  - HideUndocumented is itself a hidden property…
  - Or use the FEX: ***getundoc*** utility (32934)
- Some hidden properties have no effect, some are useful
  - scatter-plot Jitter, plot LineSmoothing, axes LooseInset, figure JavaFrame

!! Hidden properties are unsupported and may be removed in any future Matlab release without prior notice

Hidden property example – LineSmoothing

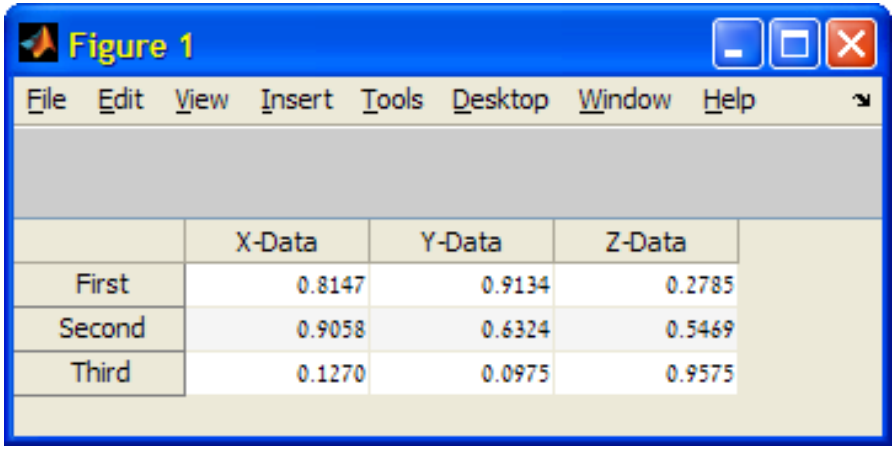
undocumentedmatlab.com/blog/plot-linesmoothing-property/

www.mathworks.com/matlabcentral/fileexchange/20979-Myaa



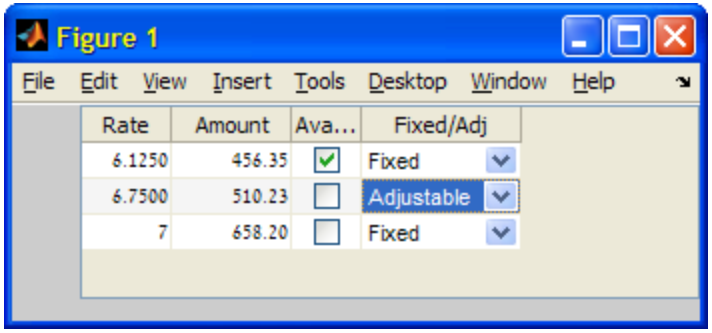
Displaying data in a uitable

```
>> data = rand(3);
>> headers = {'X-Data', 'Y-Data', 'Z-Data'};
>> rowNames = {'First', 'Second', 'Third'};
>> hTable = uitable('Parent',gcf, ...
                    'Data',data, ...
                    'Pos',[20,0,360,90], ...
                    'ColumnName',headers, ...
                    'RowName',rowNames);
```

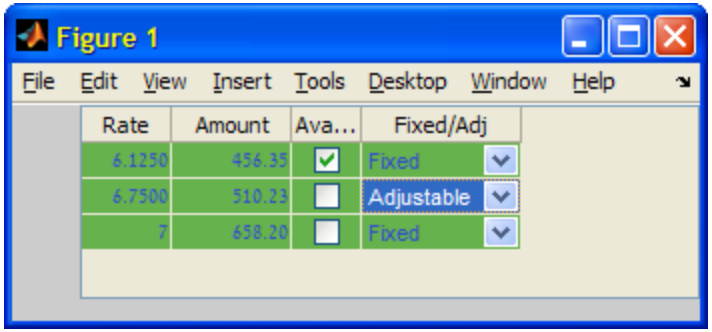


Customizing uitable appearance

```
>> set(h,'ColumnWidth',{25})
>> set(h,'ColumnWidth',{45,60,35,80})
```



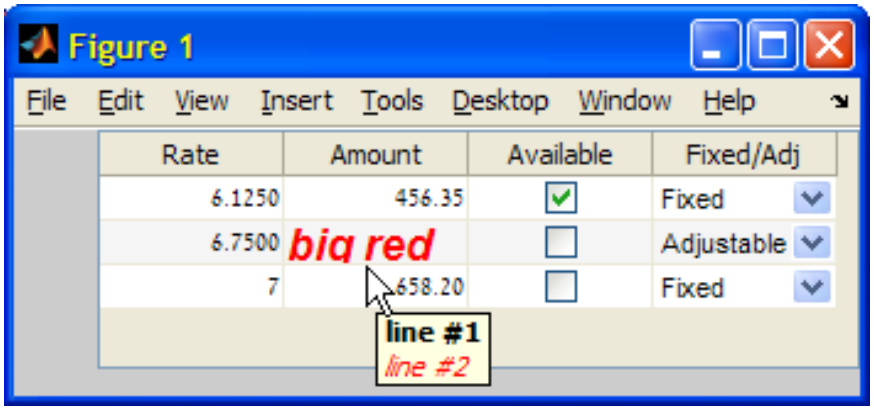

```
>> set(h,'BackgroundColor',[.4,.7,.3])
>> set(h,'ForegroundColor',[.2,.3,.8])
```




```
>> set(h,'RowStriping','off')
>> set(h,'RearrangeableColumns','on')
```

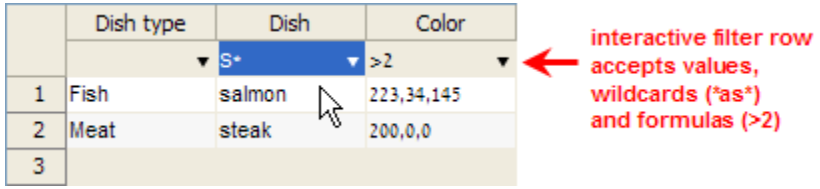
HTML support

```
>> data{2,2} = '<html><b><i><font color="red" size=+1>big red'
data =
  [6.125]  [                                          456.3457]  [1]  'Fixed'
  [ 6.75]  '<html><b><i><font color="red" size=+1>big red'       [0]  'Adjustable'
  [    7]  [                                            658.2]  [0]  'Fixed'
>> set(h,'Data',data)

>> tooltipStr = '<html><b>line #1</b><br><font color="red"><i>line #2';
>> set(h,'TooltipString',tooltipStr)
```

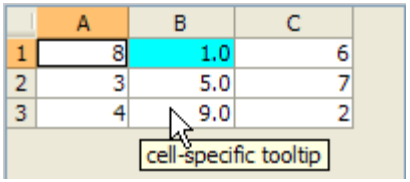


Advanced customizations using Java

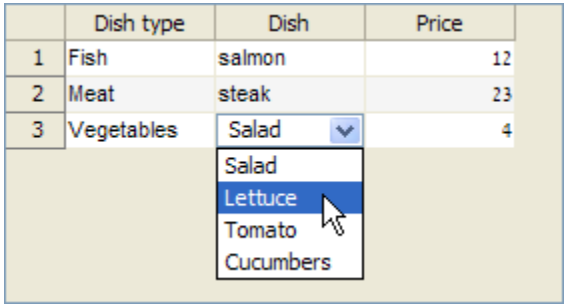
- Sorting and filtering

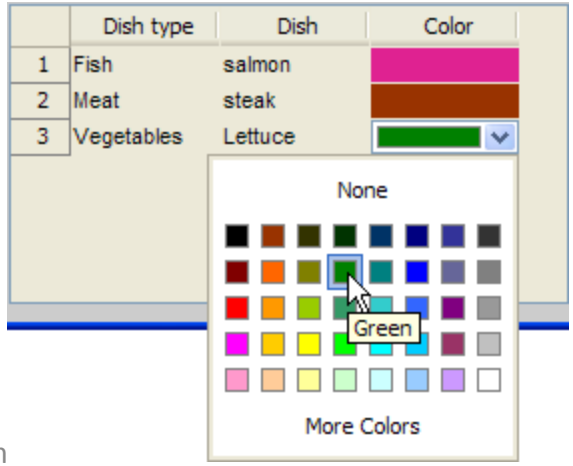


- Customized cell renderer

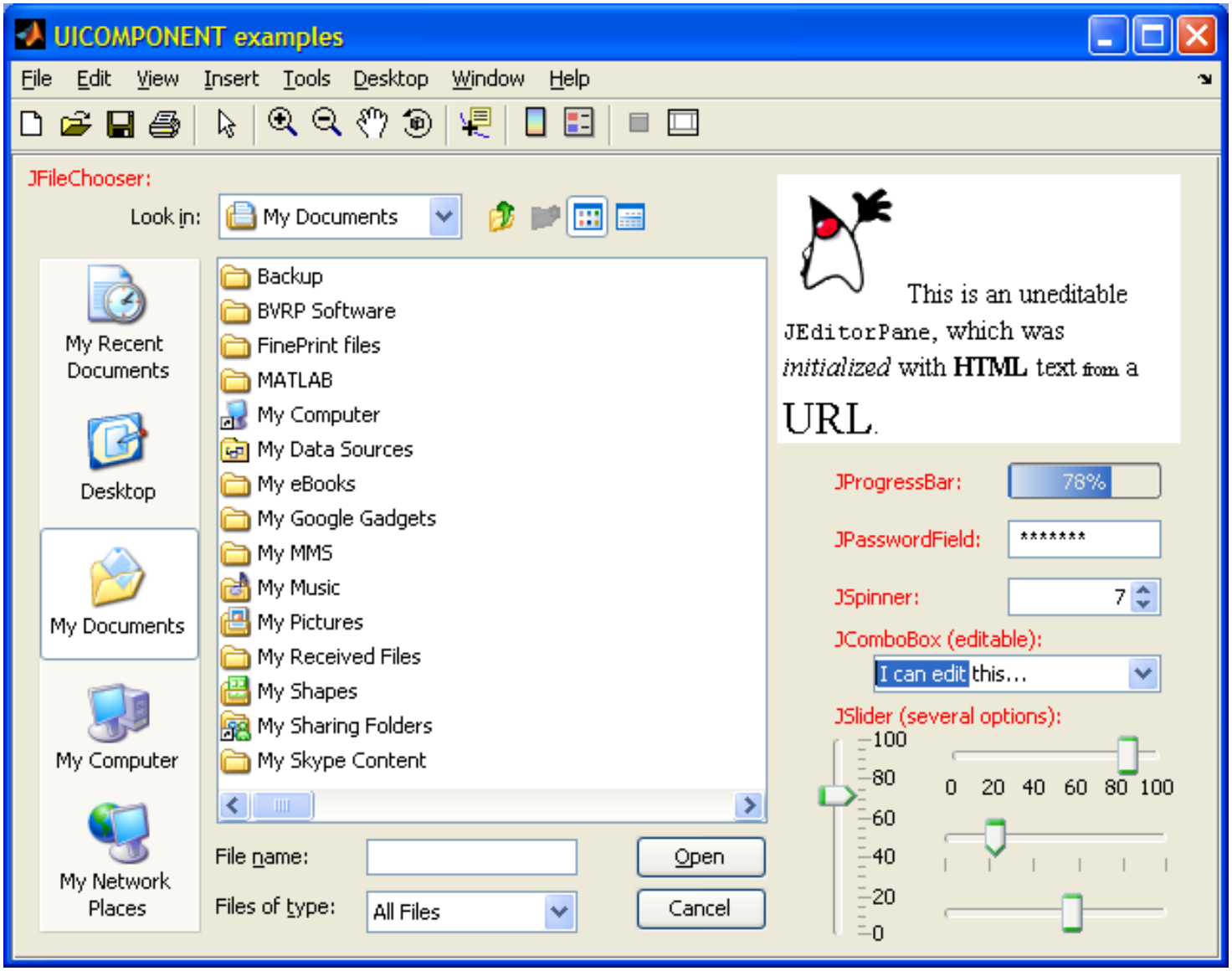


- Customized cell editor

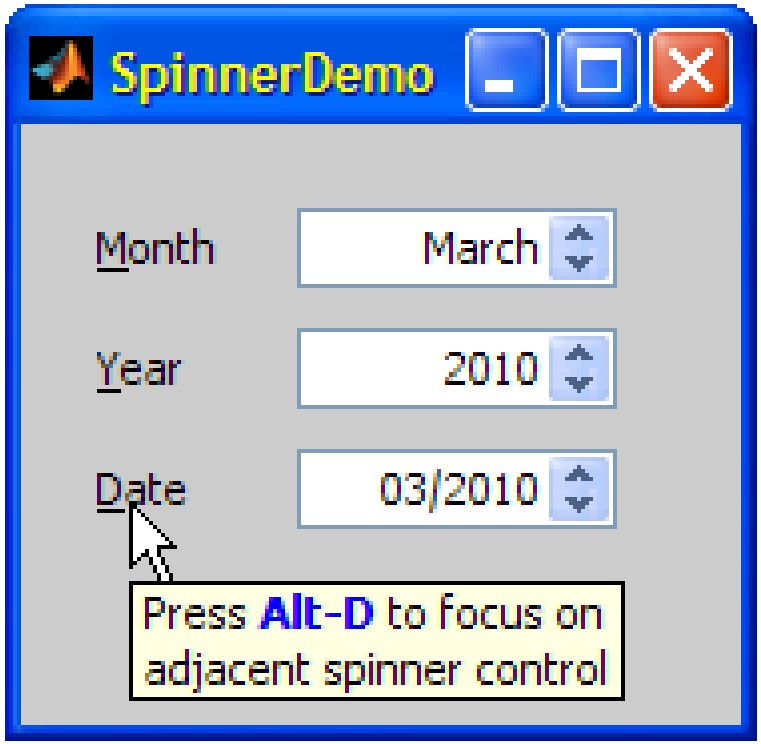




Java components in Matlab figure



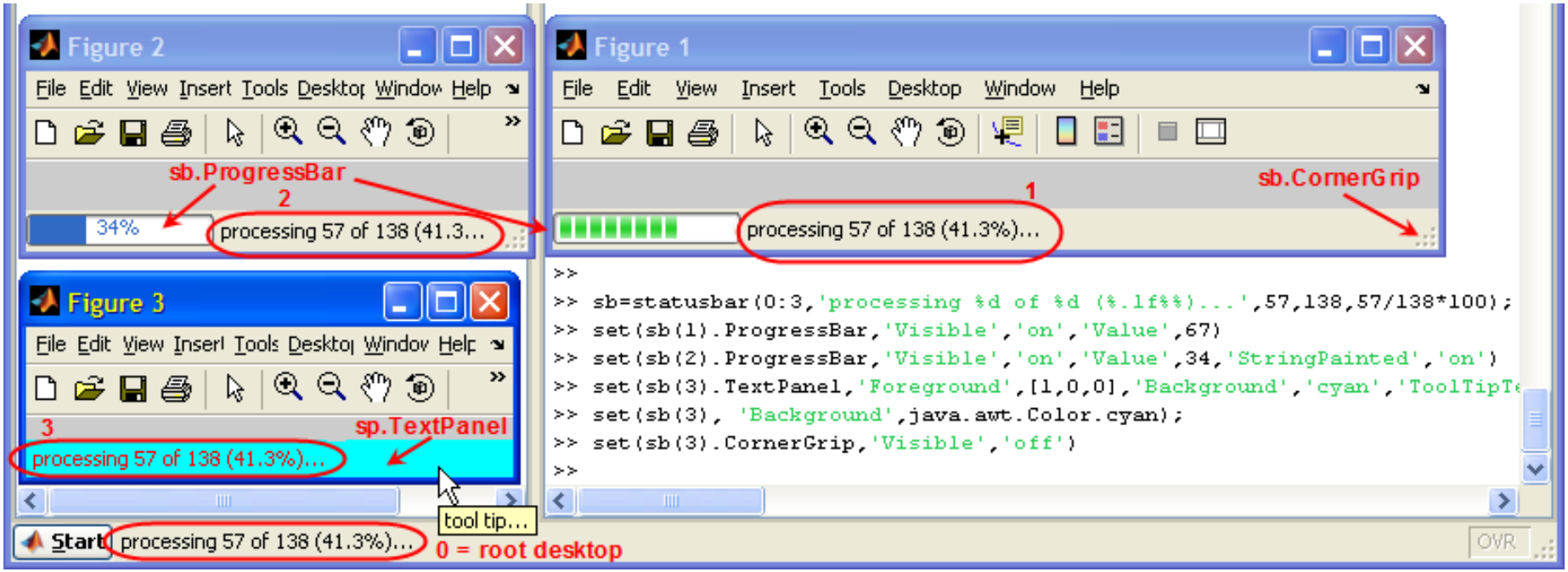
Java components in Matlab figure



FEX: ***SpinnerDemo*** (#26970)

Feedback for long-duration tasks

- FEX: ***Statusbar*** (#14773):



- An indeterminate progress bar (`JProgressBar`):

Matlab's slider uicontrol

- Continuous dragging callback possible via `handle.listener`

```
hListener = handle.listener(hSlider,'ActionEvent',@myCbFcn);
```

- The standard Matlab slider uicontrol is actually a scrollbar with Windows-95 look-and-feel:

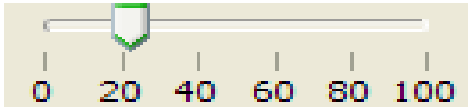
```
jScrollbar = javaObjectEDT(javax.swing.JScrollBar);
jScrollbar.setOrientation(jScrollbar.HORIZONTAL);
javacomponent(jScrollbar,[10,40,200,20]);
uicontrol('style','slider', 'position',[10,10,200,20]);
```

JScrollBar

uicontrol('slider')

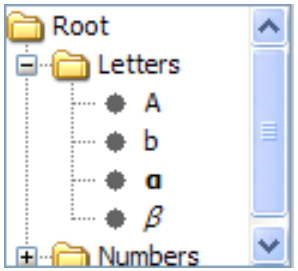
- An actual Java slider:

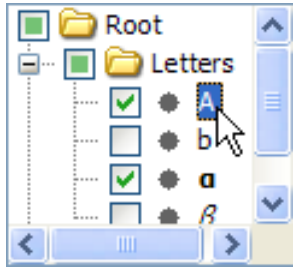
```
jSlider = javaObjectEDT(javax.swing.JSlider);
set(jSlider, 'Value',22, 'MajorTickSpacing',20, 'PaintTicks',true);
jSlider.setPaintLabels(true);     % or: jSlider.setPaintLabels(1);
```



Example: mwswing CheckBoxTree

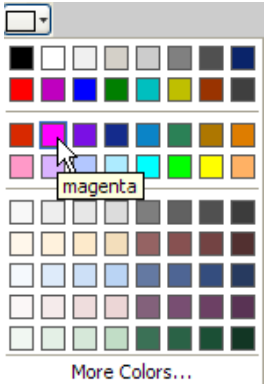
```
import com.mathworks.mwswing.checkboxtree.*
jRoot = DefaultCheckBoxNode('Root');
l1a = DefaultCheckBoxNode('Letters'); jRoot.add(l1a);
l1b = DefaultCheckBoxNode('Numbers'); jRoot.add(l1b);
l2a = DefaultCheckBoxNode('A'); l1a.add(l2a);
l2b = DefaultCheckBoxNode('b'); l1a.add(l2b);
l2c = DefaultCheckBoxNode('<html><b>&alpha;'); l1a.add(l2c);
l2d = DefaultCheckBoxNode('<html><i>&beta;'); l1a.add(l2d);
l2e = DefaultCheckBoxNode('3.1415'); l1b.add(l2e);

% Present the standard MJTree:
jTree = com.mathworks.mwswing.MJTree(jRoot);
jScrollPane = com.mathworks.mwswing.MJScrollPane(jTree);
[jComp,hc]=javacomponent(jScrollPane,[10,10,120,110],gcf);

% Now present the CheckBoxTree:
jCheckBoxTree = CheckBoxTree(jTree.getModel);
jScrollPane = com.mathworks.mwswing.MJScrollPane(jCheckBoxTree);
[jComp,hc]=javacomponent(jScrollPane,[150,10,120,110],gcf);
```





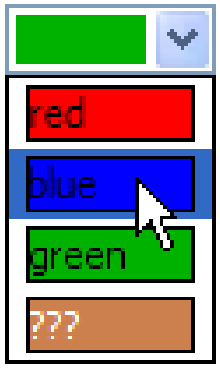
Color-selection combo-boxes

- `com.mathworks.mlwidgets.graphics.ColorPicker`:

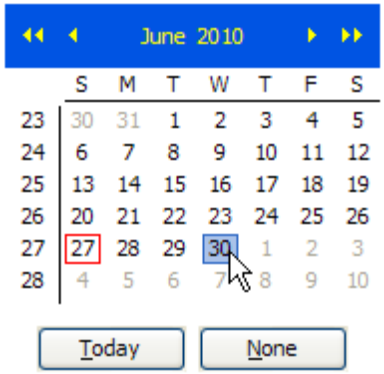


- `com.jidesoft.combobox.ColorComboBox`:

- `com.mathworks.mwswing.MJColorComboBox` (R2010a-):

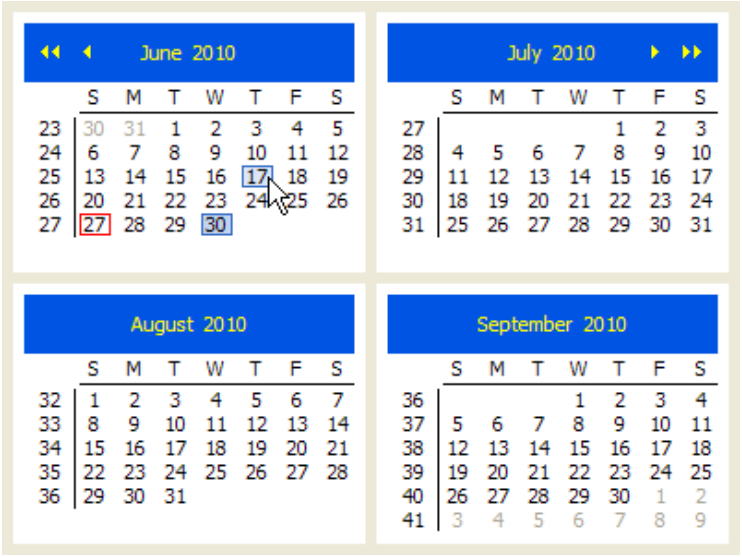


JIDE date-chooser components

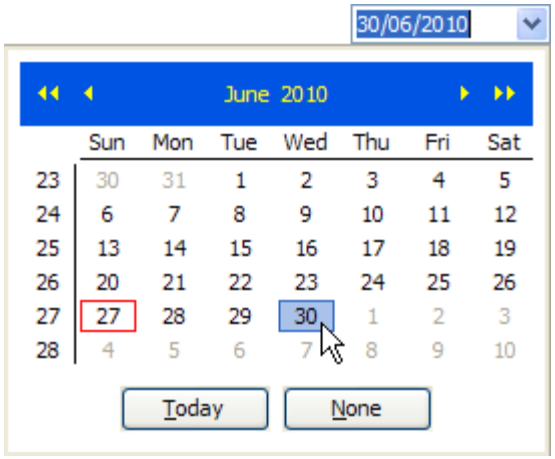


DateChooserPanel

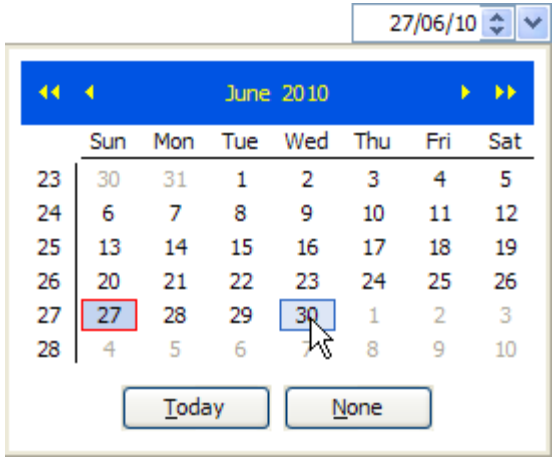
MonthChooserPanel



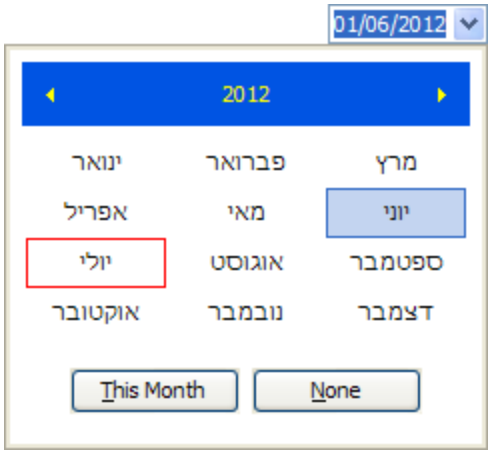
CalendarViewer



DateComboBox

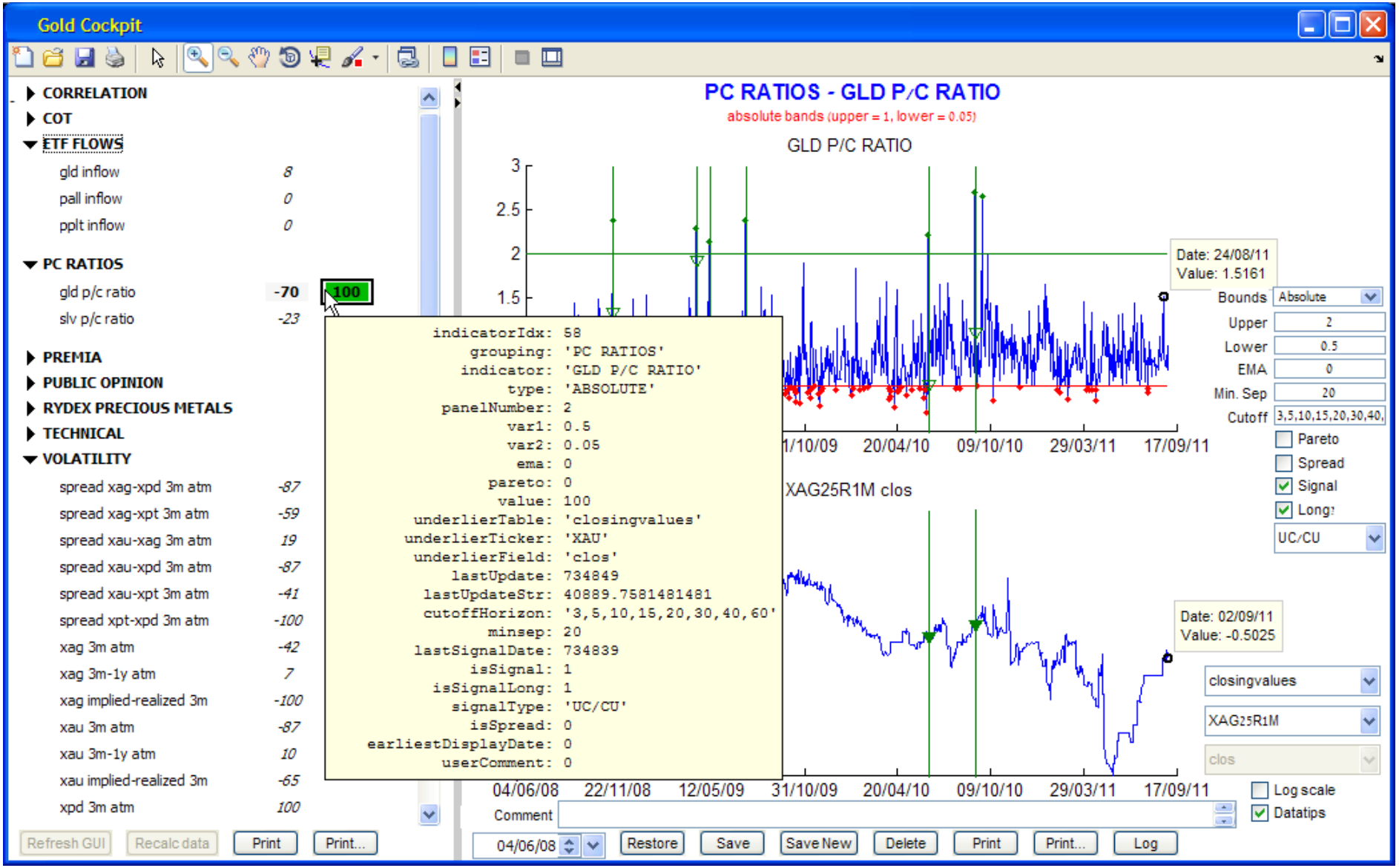


DateSpinnerComboBox

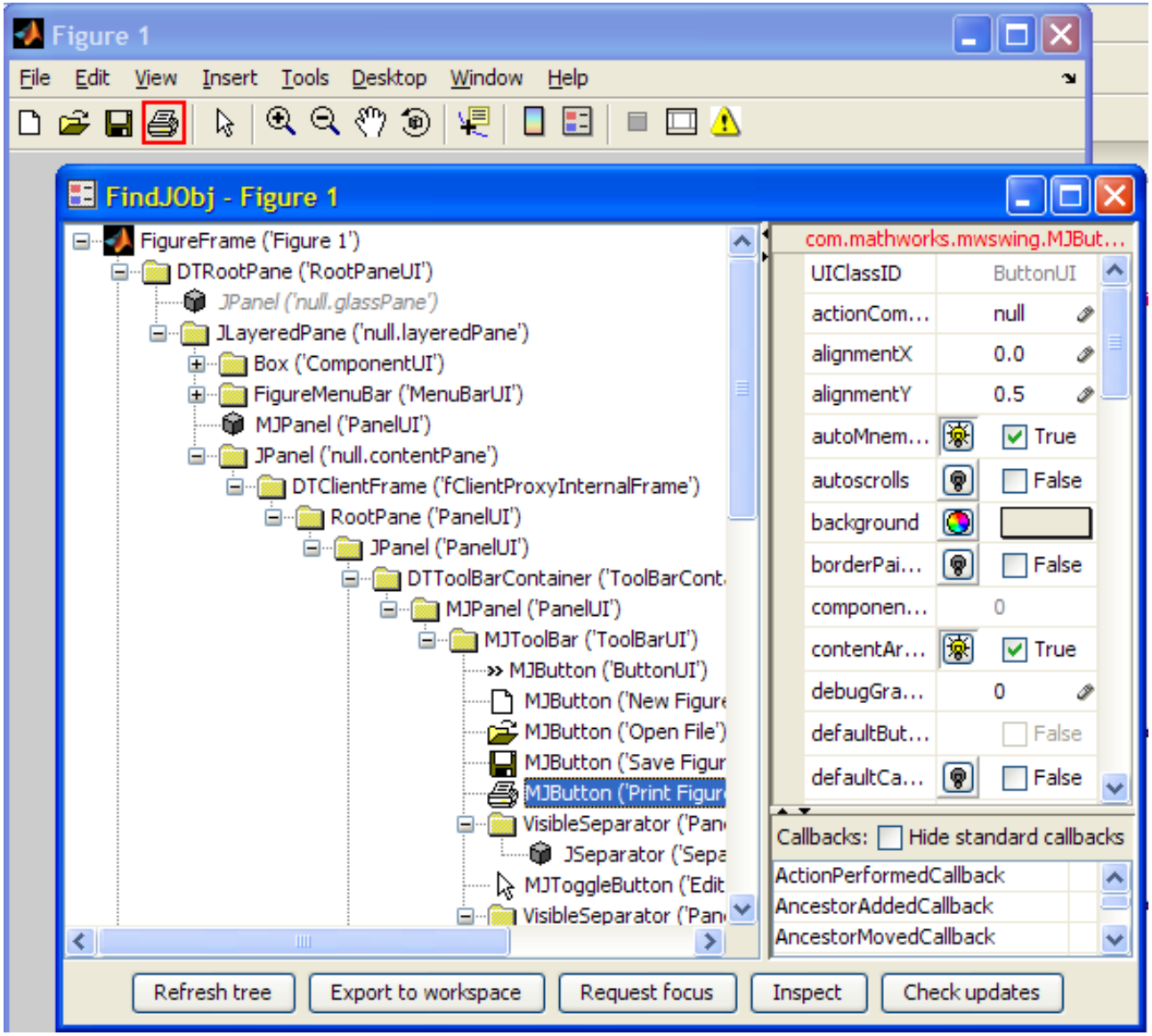


MonthComboBox

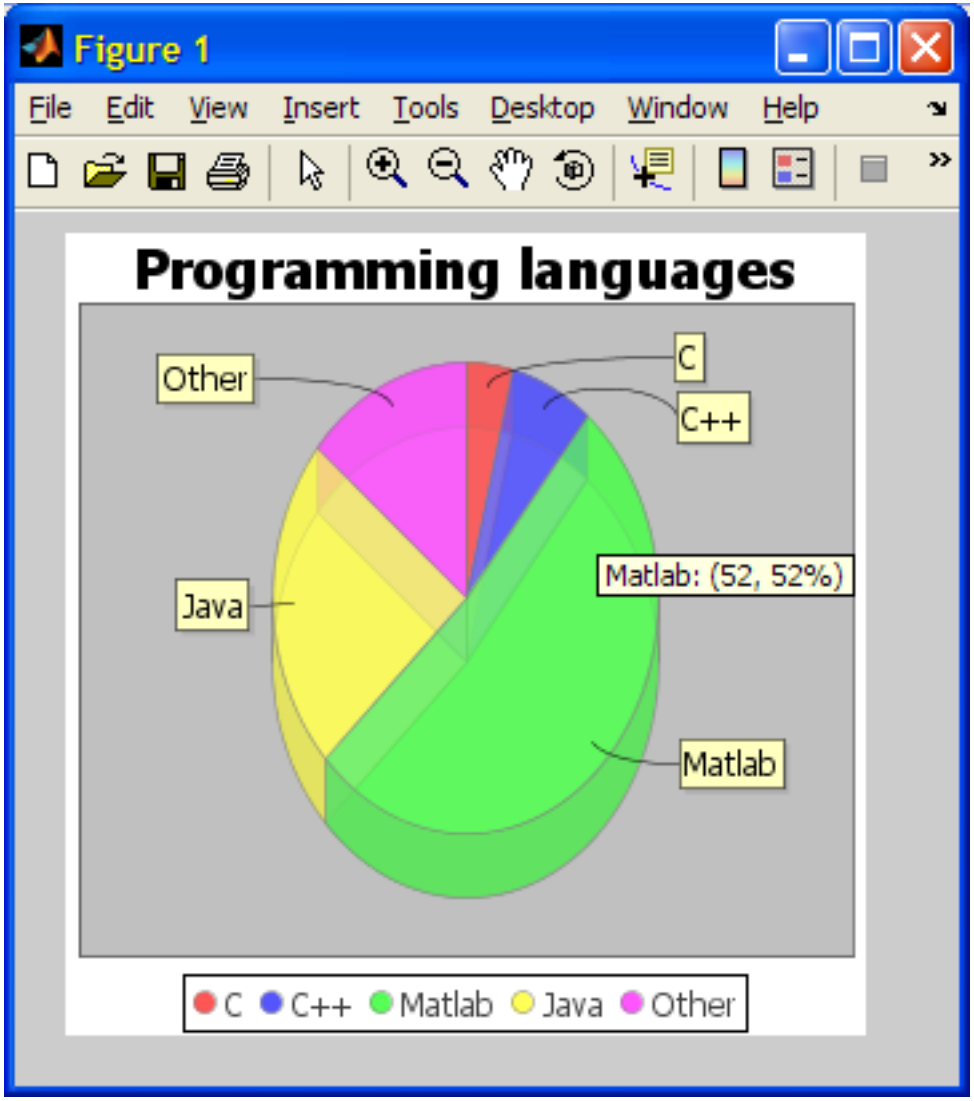
Sample Java integrated in Matlab GUI



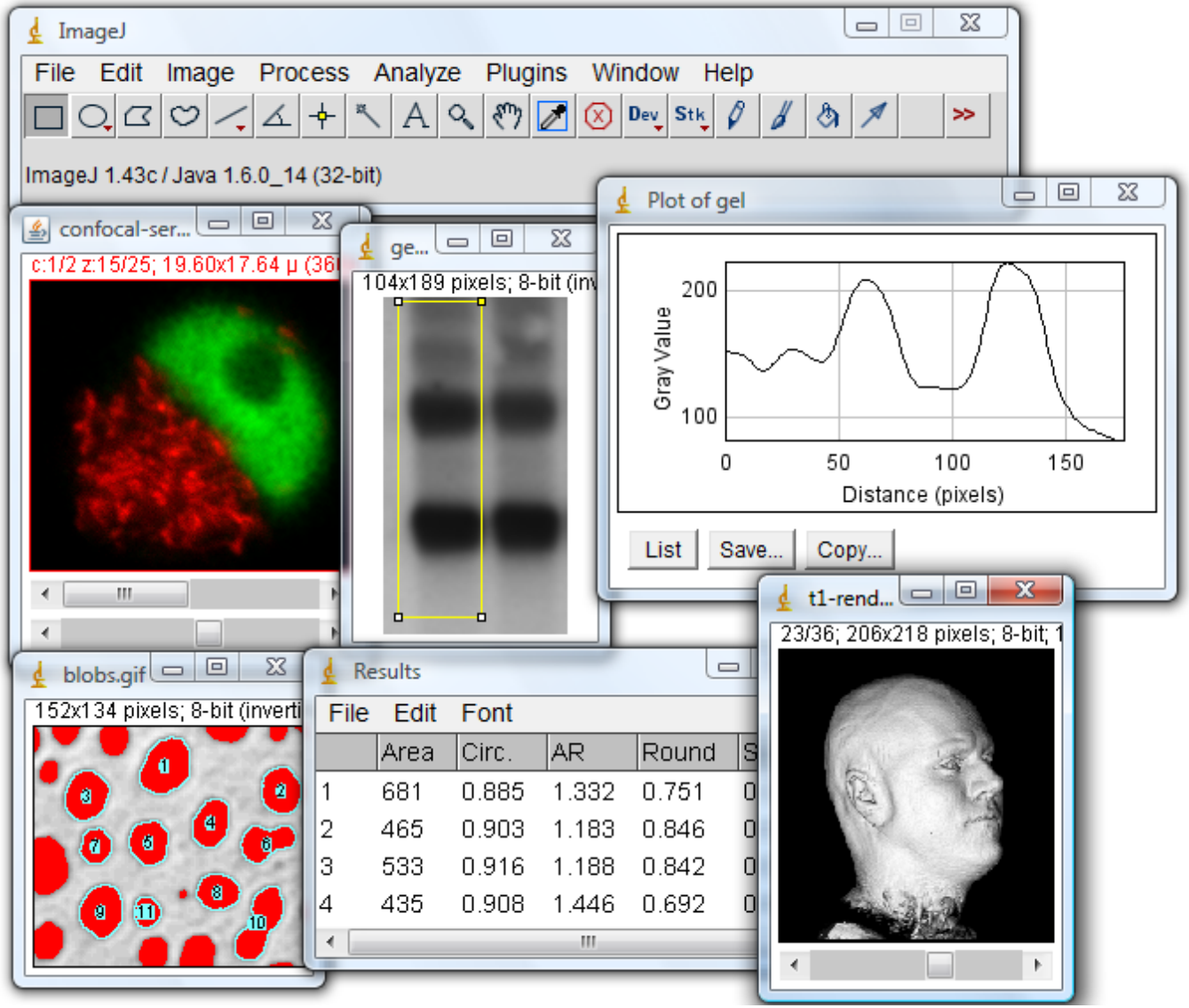
FEX: *findjobj* (#14317)



Integrating 3rd-party libs: JFreeChart



Integrating 3rd-party libs: ImageJ



Conclusion

- Standard Matlab GUI is easy to set-up but limited
- However, extremely easy to customize & extend
- Multiple undocumented properties, functions enable nice customizations, even without Java
- Entire power of Java is also directly available, and can be combined with regular Matlab GUI
- Matlab GUI is limited by our design imagination more than by actual technical limitations