Design of DataGUI3

An approach to increase the code quality of MATLAB GUIs

Sascha Meykopff, MPY, DESY (Hamburg)
24.09.2019, Zeuthen

HELMHOLTZ ASSOCIATION

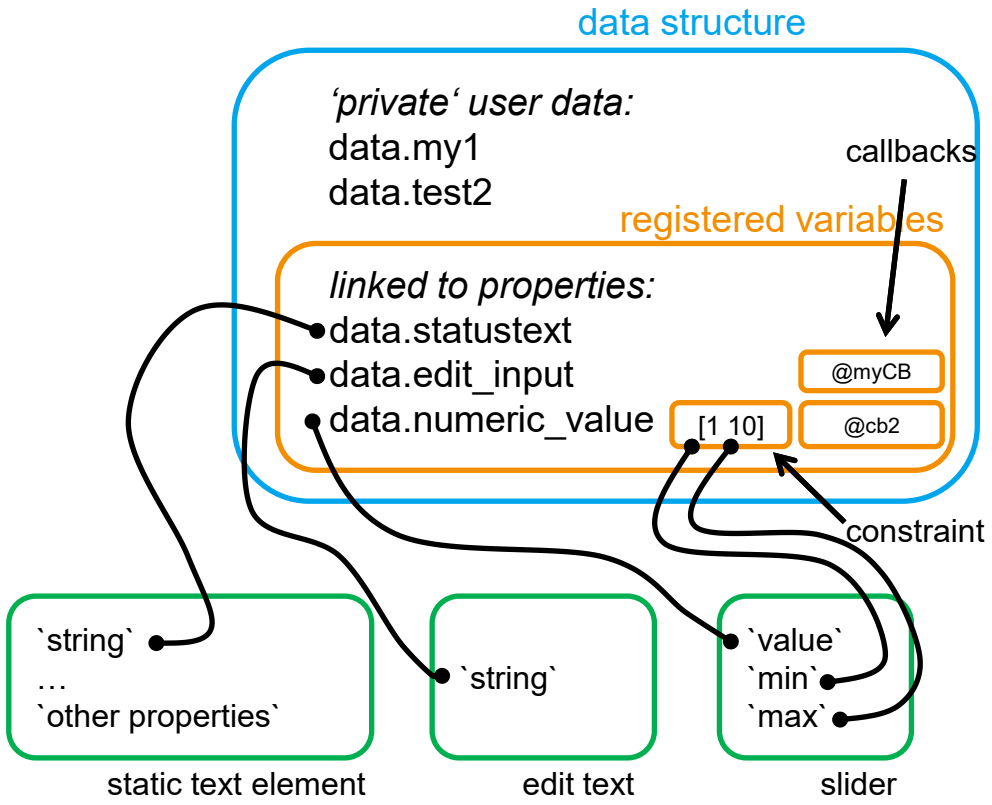
DESY

Motivation

- GUI development with Matlab is inconsistent
- Creating GUIs programmatically or with GUIDE
- GUIDE GUIs need much coding work
- Get / set values with handles
- Search Matlab help for relevant properties
- Parameter check / conversion by hand
- How to store ‘private‘ data ? Globals ? `Guidata` call ? Userdata property ?
- Callbacks are not uniform (search help)
- No copy and paste
- Difficult to read and modify foreign code

Design of DataGUI3

- The GUI is completely callback driven
- Callbacks are serialized
- Common callback interface
- A central data structure stores your variables
- GUI properties will be updated from registered variables
- Updated GUI properties will be mirrored to registered variables
- Callbacks are linked to registered variables (not to GUI objects)!
- Automatic type conversion of data and check of constraints
- A layout engine handles resize of GUI elements



Code example 1

```
function data_gui3_demo_quick_1
    data_gui3(@demoinit);

function [data,redraw]= demoinit( data, redraw )
    data.myValue = rand();
    i = cell( {
        {
        {
    });
    data = data.CALL.addData(data, i); % add to data structure

g = cell( {
        {
        {
        } );
    data = data.CALL.addGui(data, g);
```

call DataGUI3 API

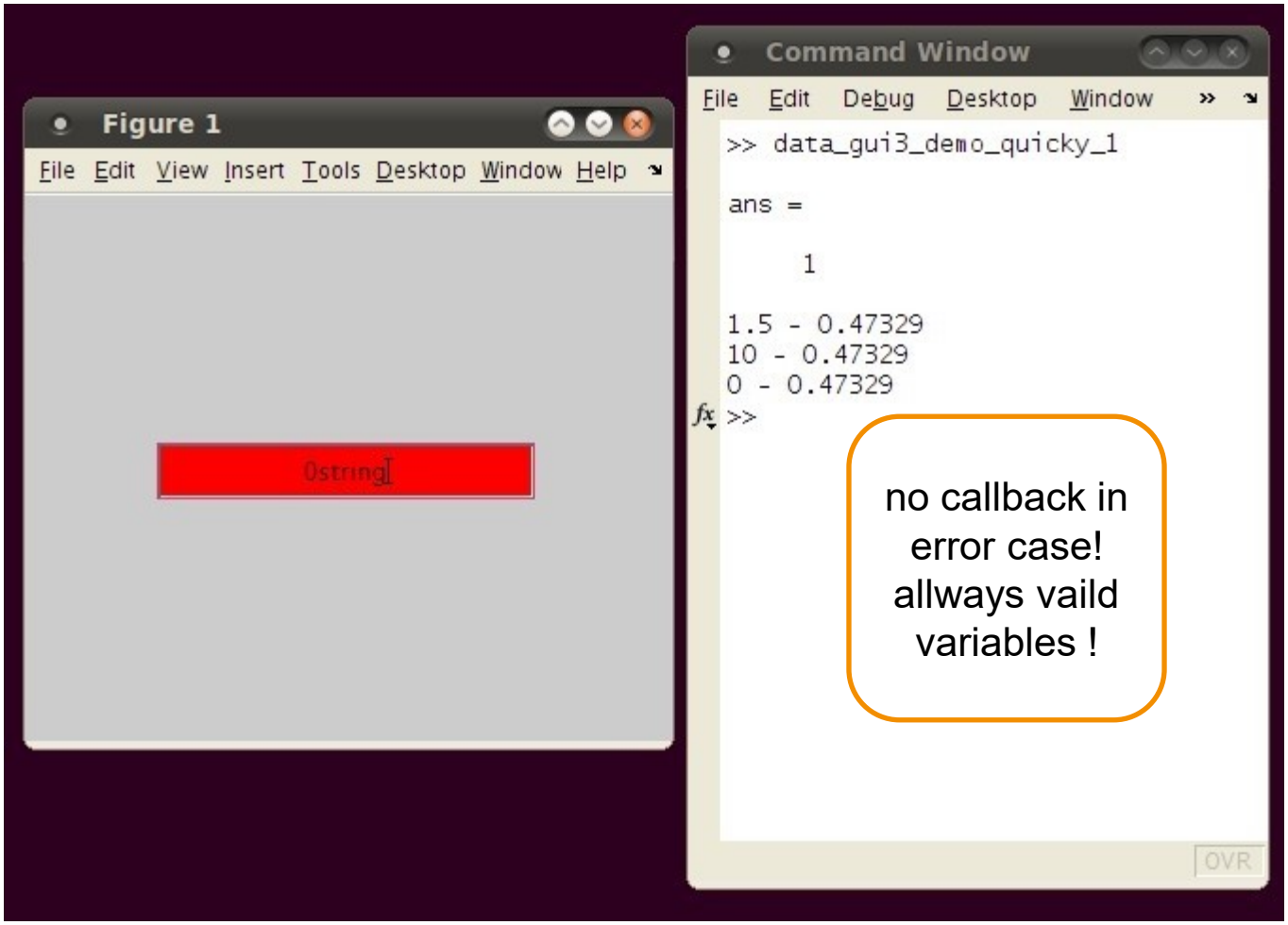
variable names	default values	constraints	callbacks
'data.editvalue'	1	[0 10]	@editCallback }
'data.figName'	'Figure name'		}

types	tag names	parent tags	required parameters	Optional Parameters – linked to gui element properties
'figure'	'demo1fig'			'Name' 'data.figName' }
'edittext'	'editfield'	'demo1fig'	'data.editvalue'	'FontSize' 10 'Enable' 'on' }

```
function [data,redraw] = editCallback( data, redraw, event )
    display([ num2str(data.editvalue) ' - ' num2str(data.myValue) ] )
```

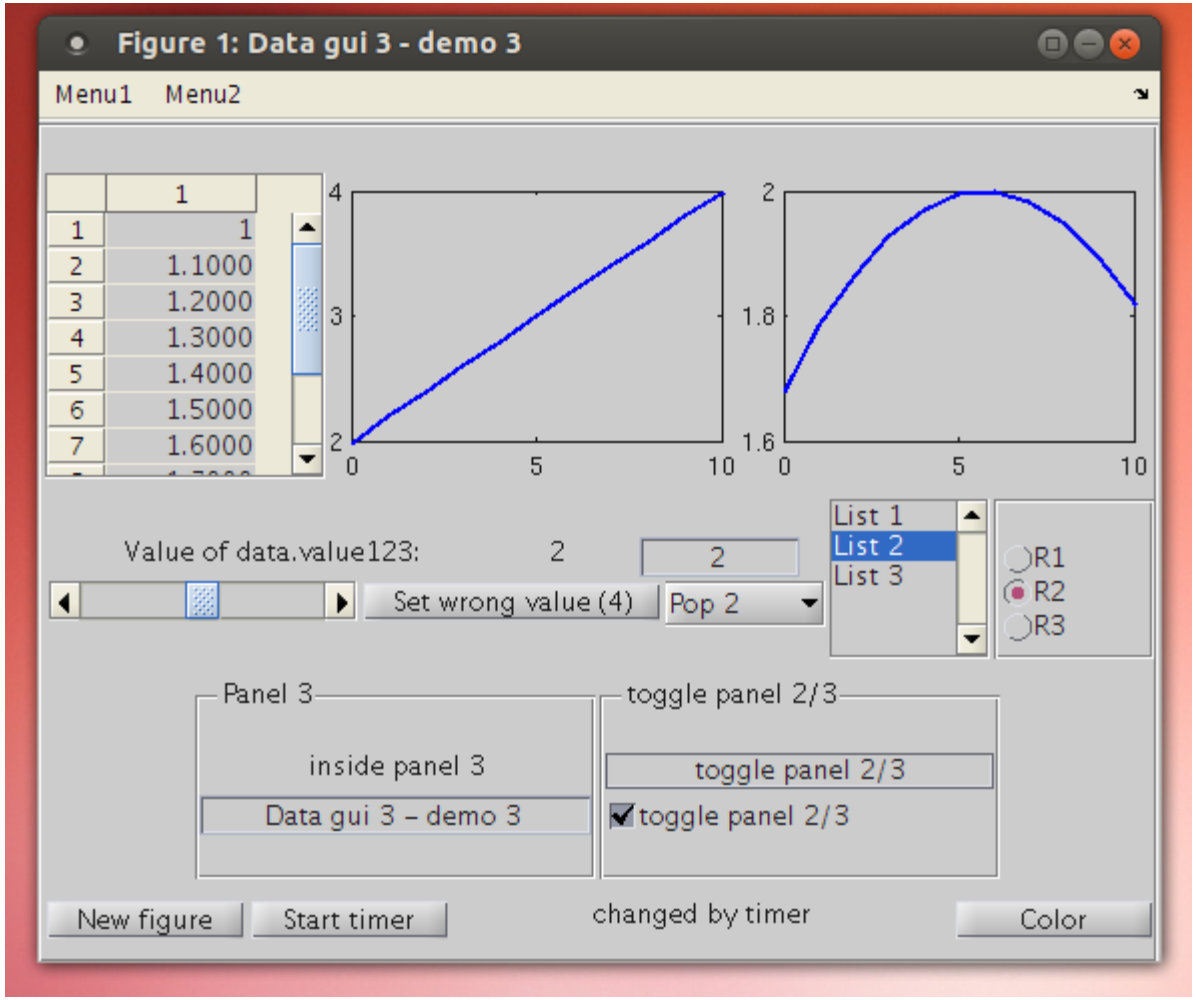

DESY

Code example 1 – screen shots



Supported Matlab GUI standard elements

- axes
- checkbox
- edittext
- figure
- listbox
- menuitem
- panel
- popupbox
- pushbutton
- radiogroup
- slider
- statictext
- table
- togglebutton

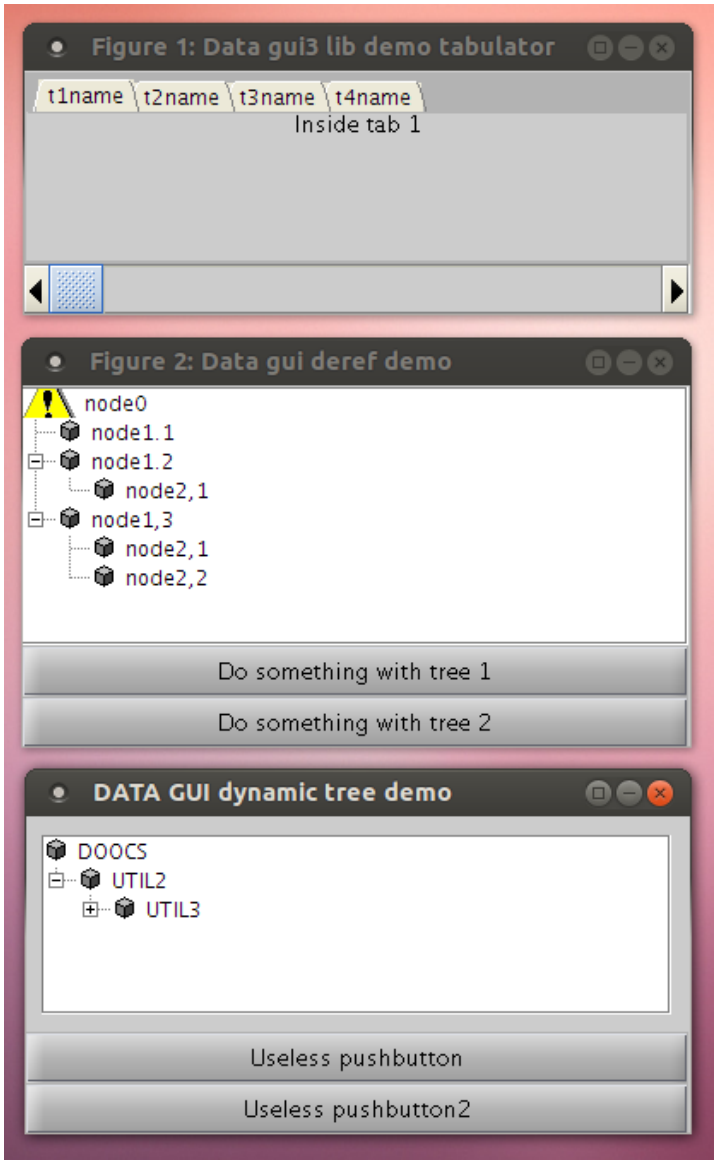


Supported special elements

- deref
- timer
- tabgroup
- tree
- treedyn

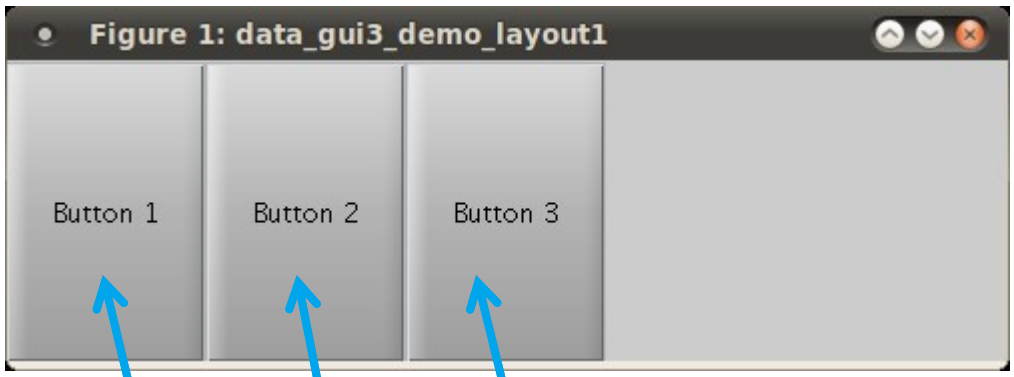
Special GUI support:
- Continuouse slider
- Mouse wheel /
- Mouse button up/down
- Enter/Leave focus events

etc.

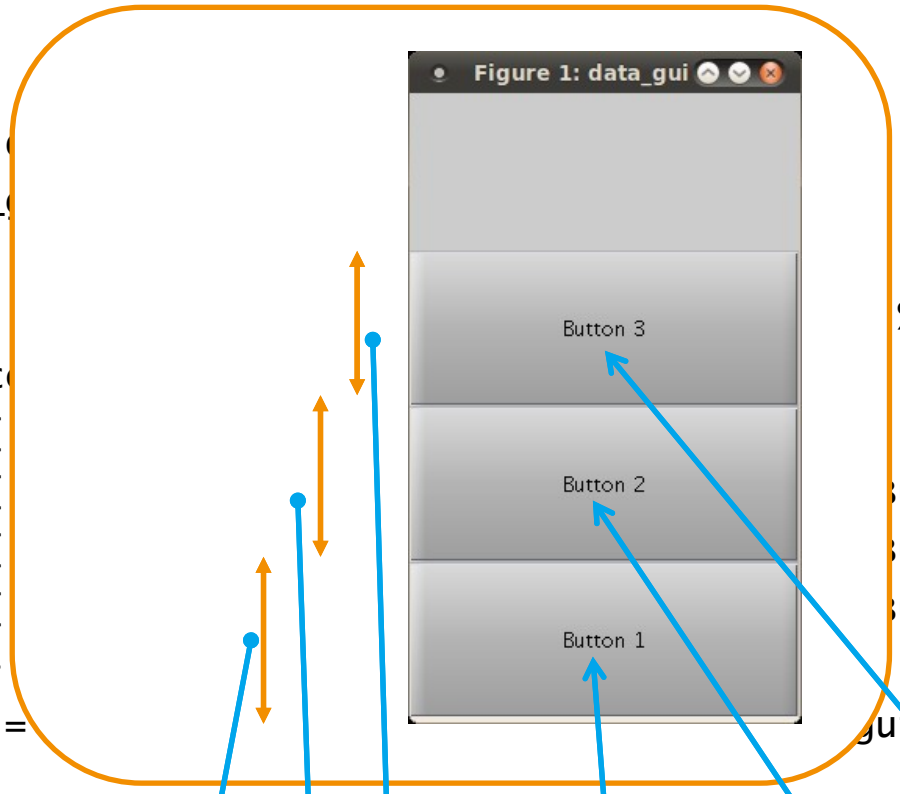


Horizontal layout

```
function data_gui3_dem
    data_gui3(@demoi

function [data,redra
    g = cell( {
        { 'figure'
        { 'pushbutto
        { 'pushbutto
        { 'pushbutto
        } );
    data = data.CALL.addGui(data, g); % create gui elements

    layout = { { 100 100 100} { 'push1' 'push2' 'push3'} }; % hor layout
    data = data.CALL.addLayout( data, 'demo1fig', layout ); % use layout
```



Vertical layout



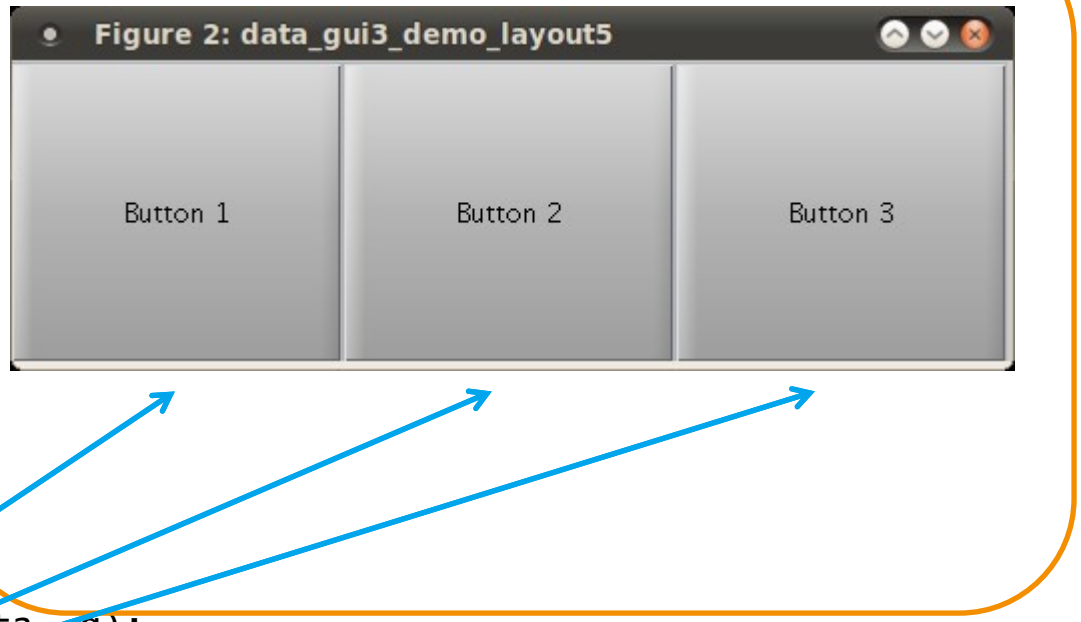

```
function (
    data_(

function                                          % init callback
    g = c(
        {                                                  'MenuBar' 'none'}
        {                                          utton 1'                }
        {                                          utton 2'                }
        {                                          utton 3'                }
        }
    data =                                         ui elements

    layout = { { 100 100 100} ; { 'push1' 'push2' 'push3'} }; % vert layout
    data = data.CALL.addLayout( data, 'demo1fig', layout ); % use layout
```

Layout with place holder

```
function data_gui3_demo_
    data_gui3(@demoin

function [data,redraw

    g = cell( {
        { 'figure'
        { 'pushbutton
        { 'pushbutton
        { 'pushbutton
        } );
    data = data.CALL.addGui(data, g); % create gui elements

    layout = { { 100 50 100 50 100} { 'push1' [] 'push2' [] 'push3'} };
    data = data.CALL.addLayout( data, 'demo1fig', layout ); % use layout
```

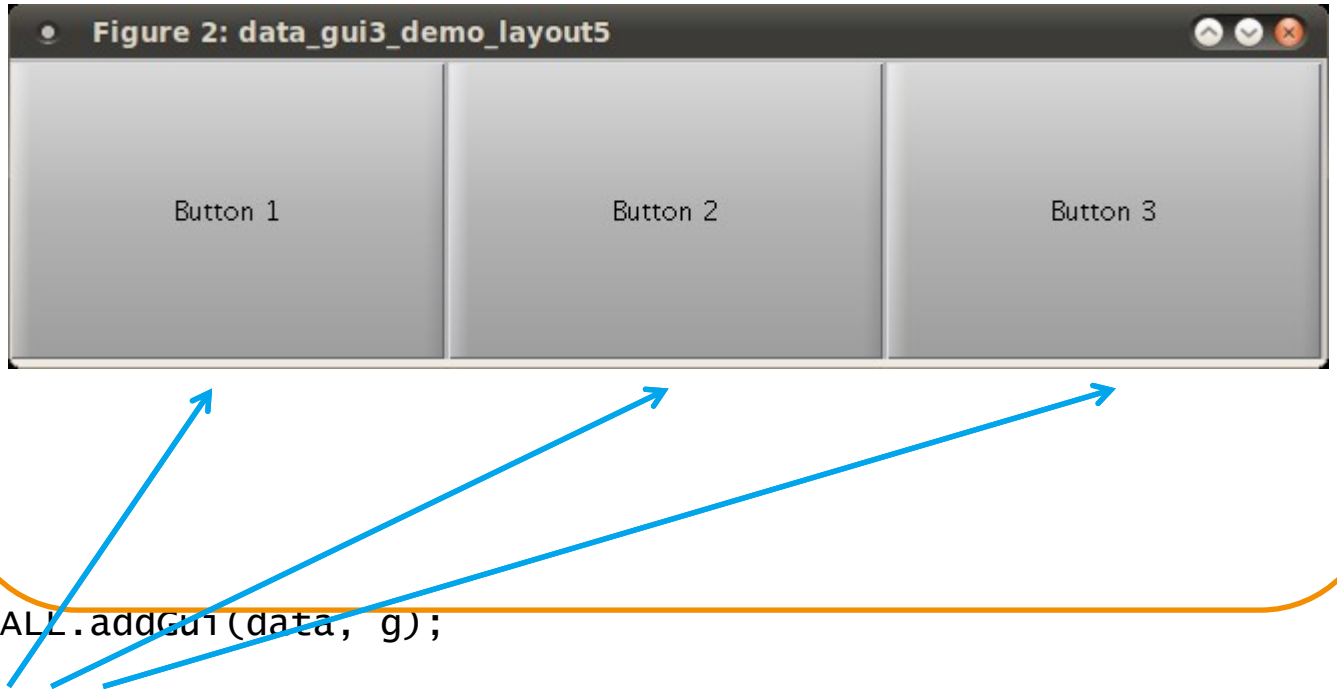
Layout with equal width

```
function data_gui3_demo_layou
    data_gui3(@demoinit);

function [data,redraw]= demo

    g = cell( {
        { 'figure'       'de
        { 'pushbutton'   'pu
        { 'pushbutton'   'pu
        { 'pushbutton'   'pu
        } );
    data = data.CALL.addGui(data, g);

    layout = { { 0 0 0} { 'push1' 'push2' 'push3'} };
    data = data.CALL.addLayout( data, 'demo1fig', layout );
```



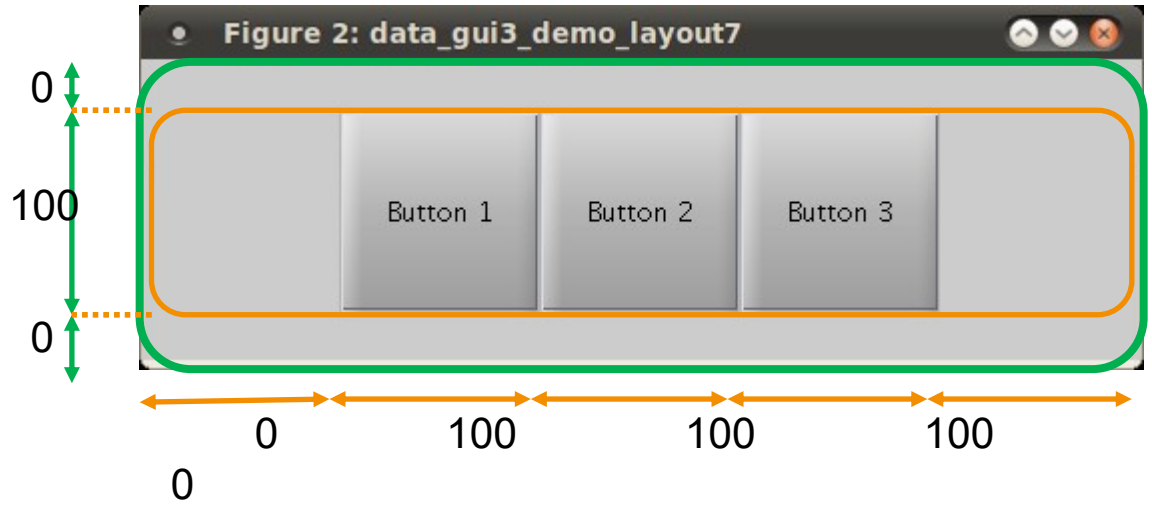
Layout with equal width

```
function data_gu
    data_gui3(@

function [data,

    g = cell( {
        { 'figu
        { 'pushl
        { 'pushl
        { 'pushl
        } );
    data = data.CALL.addGui(data, g);

    layout = { { 0 0 0} { 'push1' 'push2' 'push3'} };
    data = data.CALL.addLayout( data, 'demo1fig', layout );
```



Layout hierarchy (1)

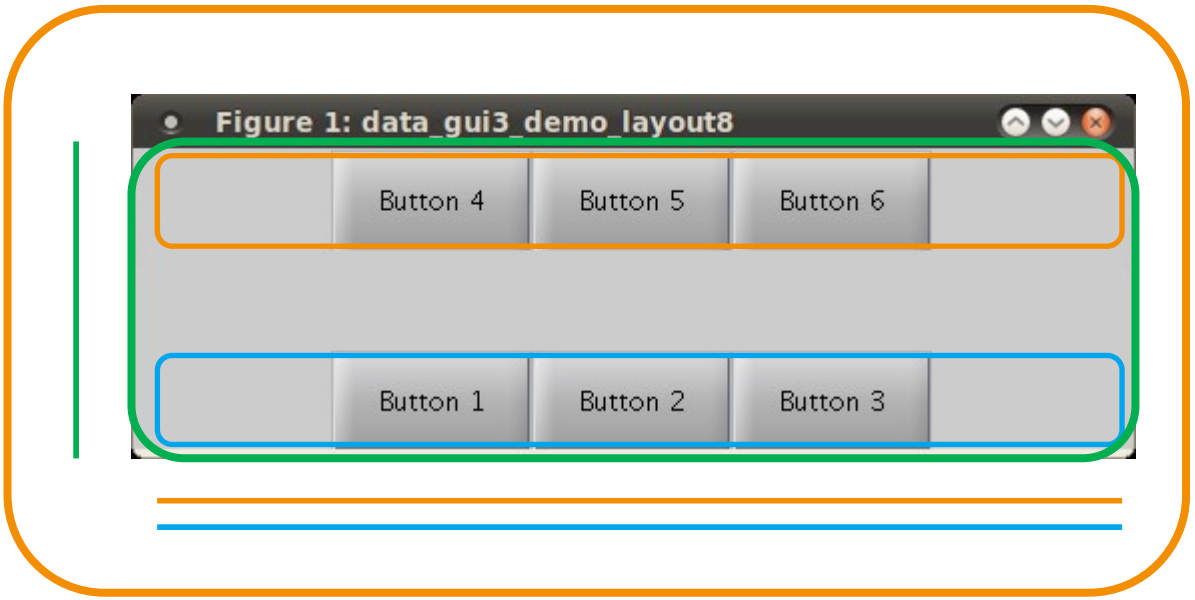
```
function data_gui3_d
    data_gui3(@demo

function [data,redr
    g = cell( {
        { 'figure'
        { 'pushbutt
        { 'pushbutt
        { 'pushbutt
        } );
    data = data.CAL
```




```
    layout_h = { {0 100 100 100 0}  { [] 'push1' 'push2' 'push3' [] } };
    layout_v = {      {0 100 0}    ; { [] layout_h [] }                 };

    data = data.CALL.addLayout( data, 'demo1fig', layout_v );
```

Layout hierarchy (2)

```
function data_gui3_demo_layout8
    data_gui3(@demoinit);

function [data,redraw]
    g = cell( {
        { 'figure'
        { 'pushbutton'
        { 'pushbutton'
        { 'pushbutton'
        { 'pushbutton'
        { 'pushbutton'
        { 'pushbutton'
        } );
    data = data.CALL.addGui(data, g);

    layout_h1 = { {0 100 100 100 0}   {[] 'push1' 'push2' 'push3' []} };
    layout_h2 = { {0 100 100 100 0}   {[] 'push4' 'push5' 'push6' []} };
    layout_v  = {      {50 0 50}    ; { layout_h1  [] layout_h2 }     };
    data = data.CALL.addLayout( data, 'demo1fig', layout_v ); % use layout
```



Layout summary

Layout size meanings:

- Size > 1 : pixel
- Size == 0 : remaining space
- 0 < Size < 1 : fraction

Layout modes:

- horizontal / vertical layout
- layout hierarchy
- panel layout
- stacked layout (panels and axes)
- dynamic layout

Summary

- Using DataGUI3 speeds up development
- Uniform API
- Copy and paste possible
- Robust GUIs with parameter check and conversion
- GUIs with small code size
- Uncommon Matlab GUI features (tabs / trees / wheel …)
- Fully resizable layout
- Ideas are welcome

Code under http://www.desy.de/xfel-beam/

Thank you for you attention