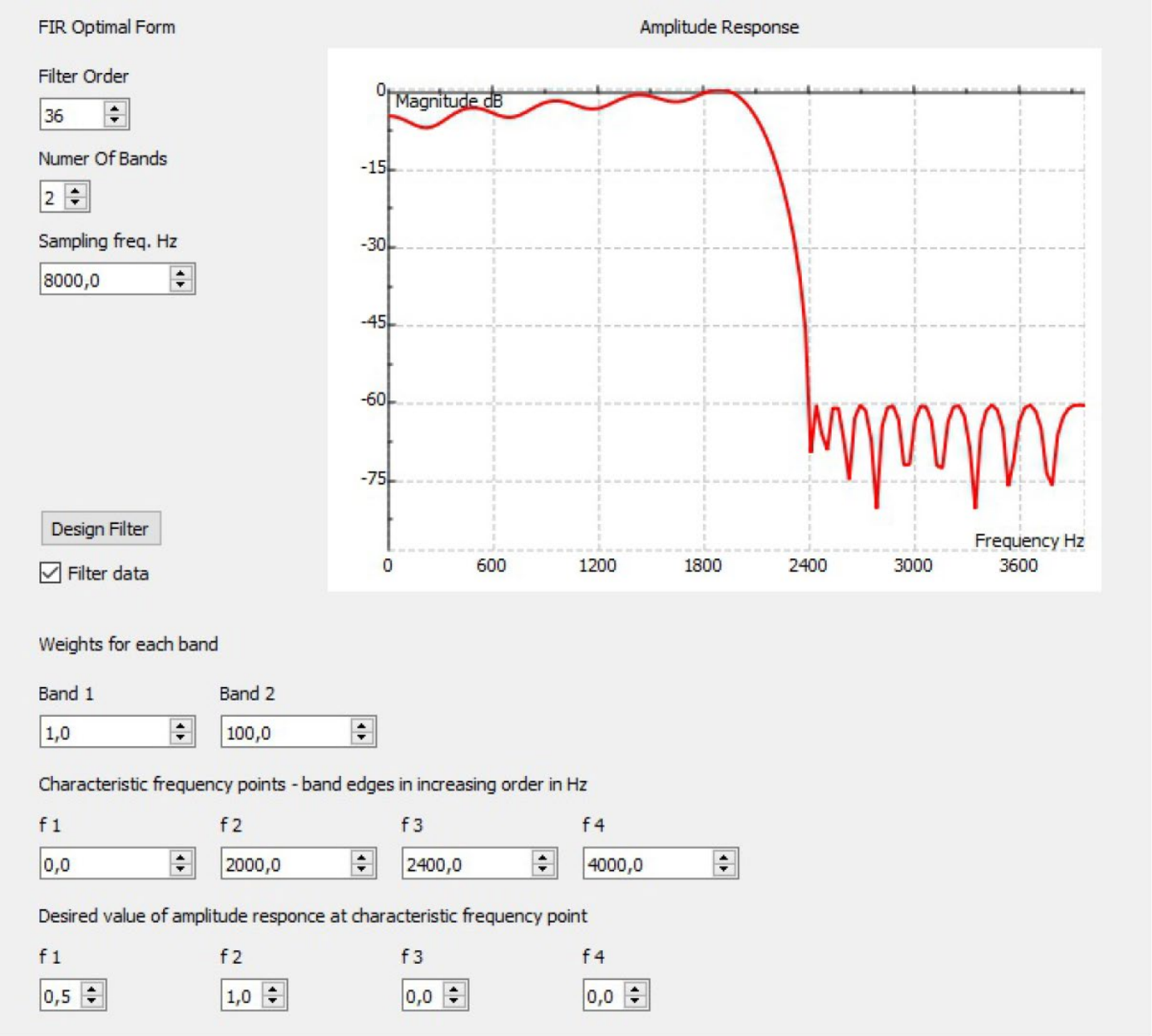
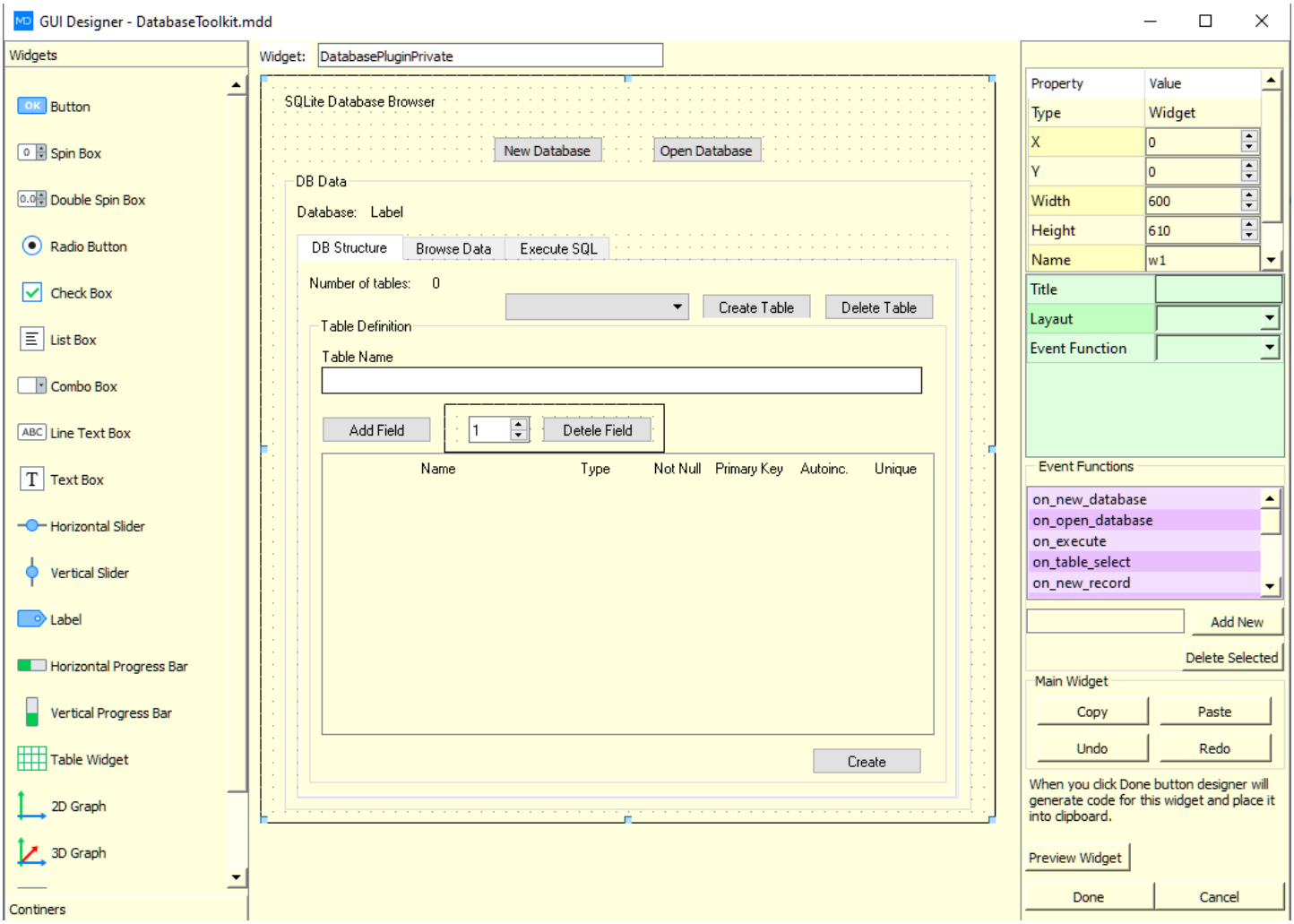
MatDeck GUI Designer

MatDeck contains graphical user interface elements of different types. A list of the implemented elements, ways to create them and implement in MatDeck documents, descriptions of GUI function arguments and much more is included in the MatDeck GUI User Manual. Beside these individual GUI elements, we have implemented the *GUI Designer*.



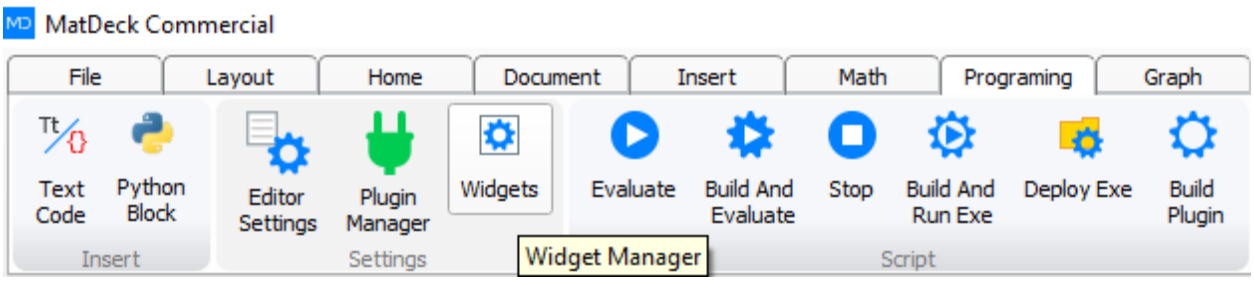
The image above is an example of the types of applications you can make graphical user interface using the GUI designer. *GUI Designer* is a software development tool which is used to simplify the creation of widgets and GUI applications. By arranging and placing GUI elements through the use of drag and dropping; creating, placing and editing is made much easier.



This is the what the actual GUI designer used for the form above looks like. As you can see the from itself includes no code and the form above was created solely through the use of dragging and dropping GUI elements into the work area.

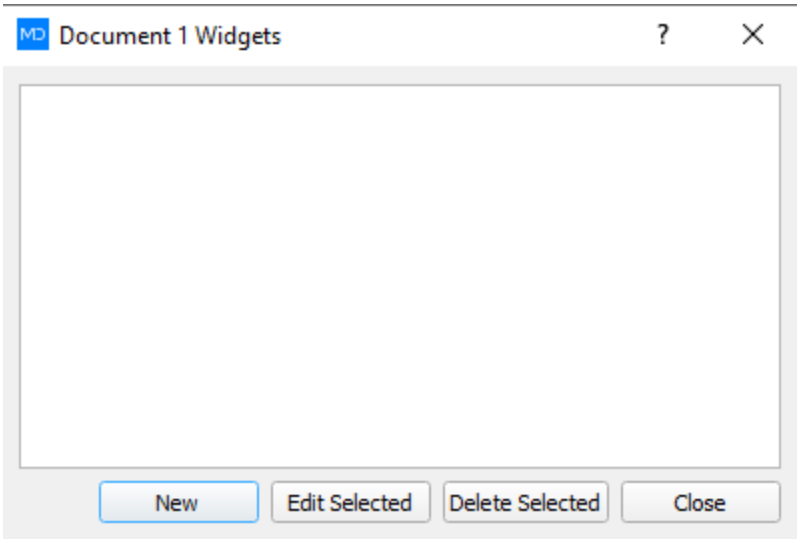
In MatDeck's GUI Designer, the main work space acts as the main widget where all other GUI elements are placed. This widget is the parent object for all other elements that we have placed on the form. Inserting any new container or element on top of another object in the GUI designer makes the select object the parent for newly inserted object. Note that there is no limit to how many iterations of nesting can be done.

To create a widget and use the GUI Designer in a MatDeck document, go to Programing tab and press the 'Widgets' button (Picture 1).



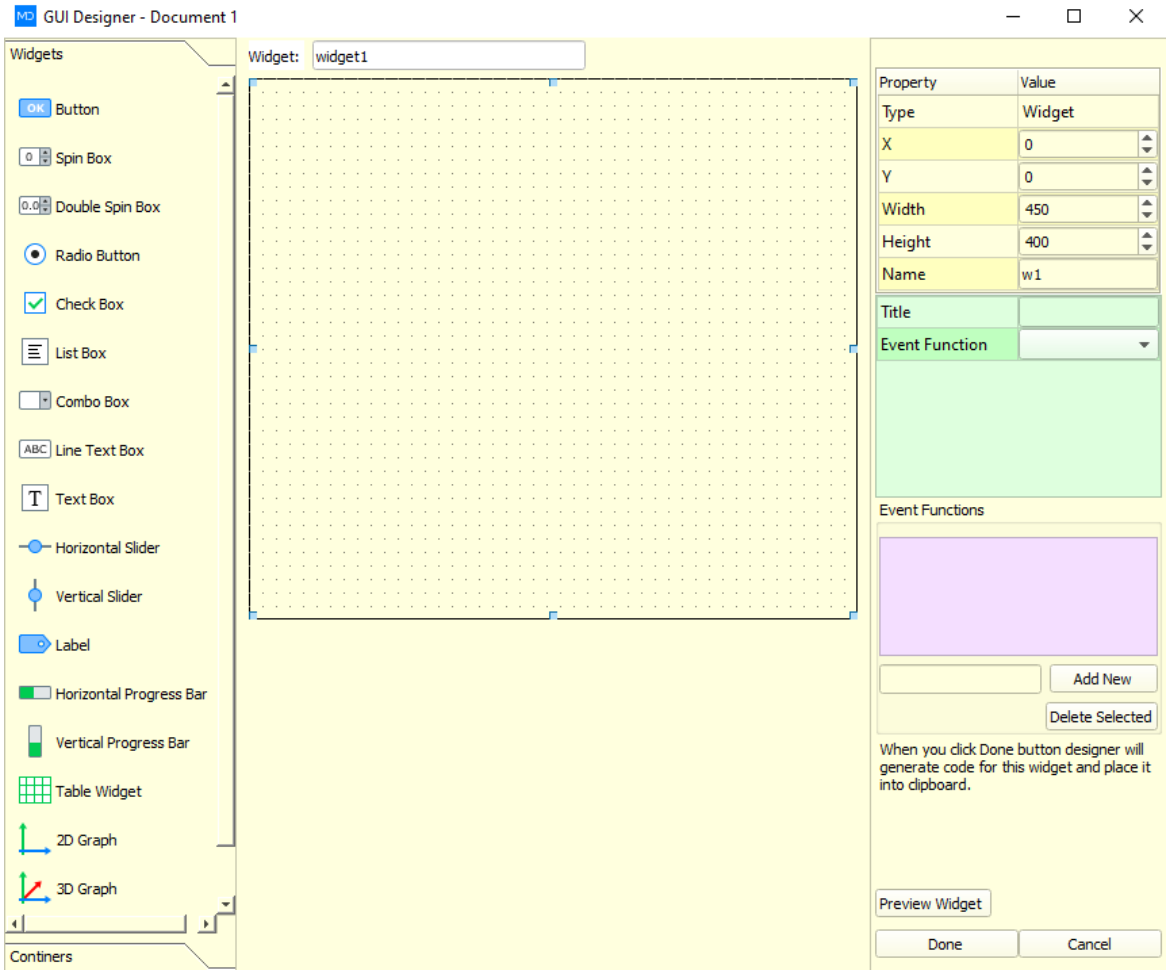
Picture 1: Widgets button

A new window will open which will contain a list of all the widgets that have been created in the current MatDeck document (Picture 2). From this window you can create a new widget, edit or delete the existing one.



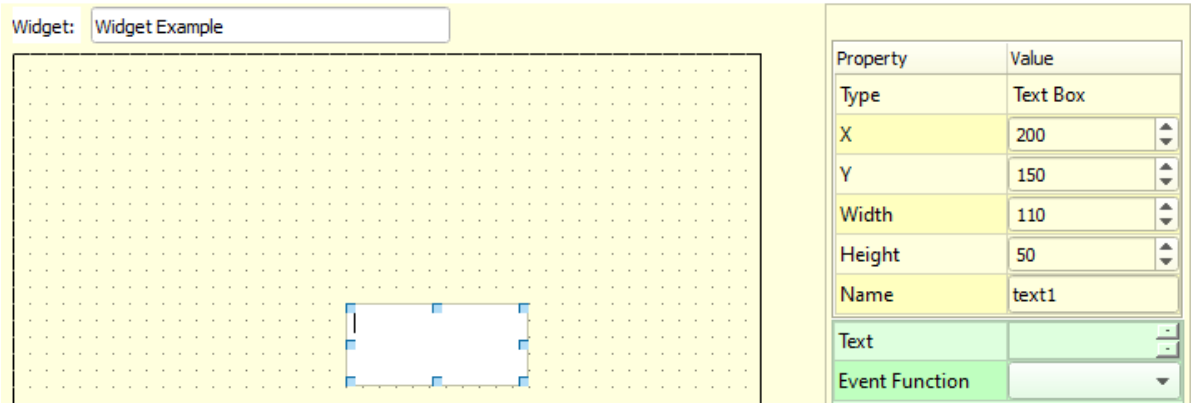
Picture 2: Document widgets editor

To create a new widget, simply, press 'New' and a new empty GUI designer tab will open. When doing so make sure that you haven't selected any widgets that may be present in the window. To edit or delete a pre-existing widget, select the said widget first and then press on the suitable button at the bottom of the tab. Either you choose to create new widget or to edit existing one, the GUI Designer will open. The default look of empty GUI Designer is shown on picture bellow.



Picture 3: GUI Designer

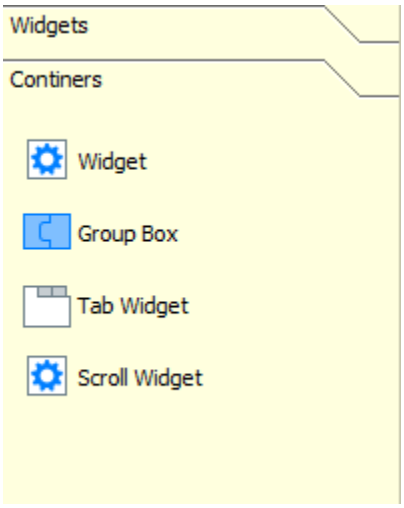
On the left side of the GUI Designer are the GUI elements; in the middle of Designer is working area of widget where GUI elements can be placed. To place one of the elements just press preferred GUI element and select the place on widget working area to place it.
For example, we will place one Text Box on the widget. On the right side of Designer, the properties of the selected element are displayed and these properties can be modified from here (Picture 4). In the example below, we have also edited the name of the widget to 'Widget Example'. To do this, you just have to write your preferred name in the field above the main working area.



Picture 4: GUI Element Property

Please note that to actually connect or interact between GUI elements, you have to add necessary code for the interaction itself. To do so, edit the generated code in a MatDeck document by adding the necessary pieces of code.

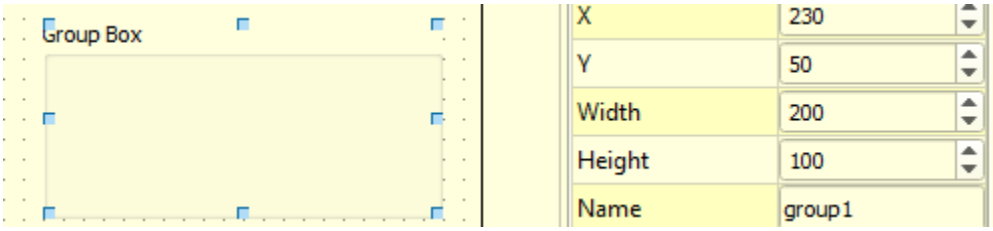
If we open the Containers menu, from the bottom left part of the GUI Designer, a new list of GUI elements will appear. These elements are called containers and can be placed on main widget area. They act as carriers for other GUI elements.



Picture 5: Containers

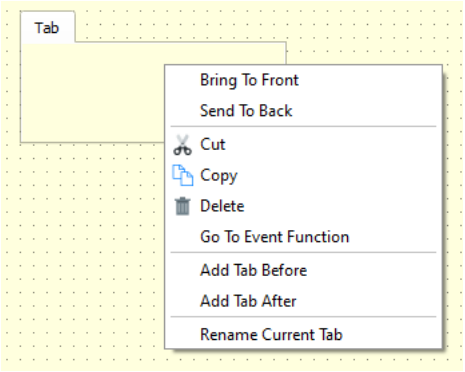
To insert any of these containers on the main form uses the same method as inserting other GUI elements. First, select the preferred container and then, select the place on the working area where you want to place it.

When a container or element is selected, it can be resized by moving the blue squares around it or it can be resized by changing the Width and Height values from the Property table of the selected object.



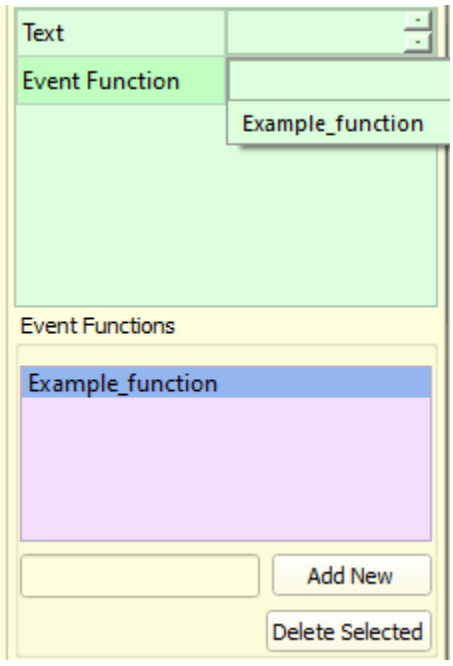
Picture 6: Resizing of object

If you wish to preview your widget and check on your progress at any point during your work you can do so by pressing the 'Preview Widget' in the bottom right corner. It will output a new interactive window where you can check and use your widget.



When a Tab widget is used, we can use the context menu to rename the current tab or to insert a new one.

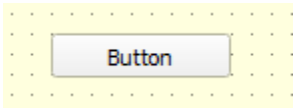
There is option to link an event function for selected GUI element. Event function list is empty on all newly created Widgets; we have to define a new event function by using the Event Functions part of the Designer. To do so, input the name of the new event function you wish to add in the empty field in the event function window and press 'Add New'. After creating a new event function, it will appear in the Event Function list for the selected GUI element (Picture 5).



Picture 7: Event Functions

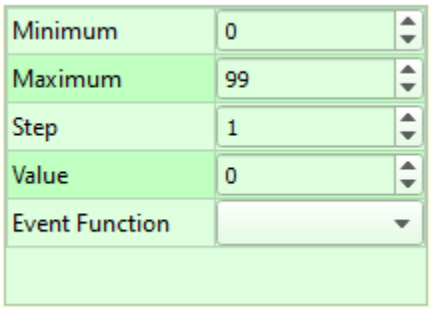
The Event Functions container is a global container which contains all the created event functions for the Widget we are currently creating/editing. In the field named ‘Event Function’ in the green tab above, you can select one of the defined functions by choosing one of the options that appears in the drop down list when clicked.

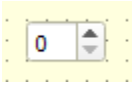
The green window explained above which is used to edit certain aspects of GUI elements, primarily the event function, can be unique to certain elements. Meaning, not all GUI elements will have the same options present and some elements will have options only specific to that element and not present in other elements. Below, all examples of this are explained.



- **Button**

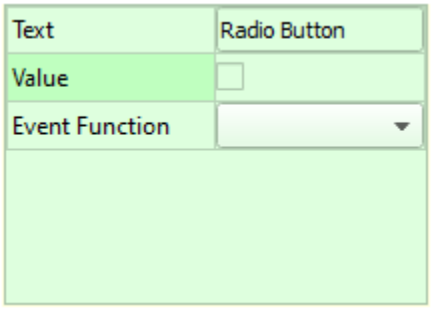
When the Button GUI element is inserted into the working area, the green box above appears. The ‘Text’ field can be used to change and set the visible label on the button. Additionally, you can choose the Event Function that will be used on the Button’s event.

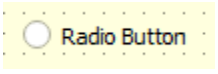




- **Spin box**

When a spin box GUI element is inserted, element specific settings that can be changed will appear. From here, the Minimum and Maximum value of the spin box can be set by using the ‘Minimum’ and ‘Maximum’ fields respectively. The step at which the spin box’s value increments by can be also set by using the ‘Step’ field. The starting value of the spin box can be set by entering your chosen value in the ‘Value’ field. Lastly, if an event function has been, the event function used can be changed by selecting an option on the drop down menu at the bottom.

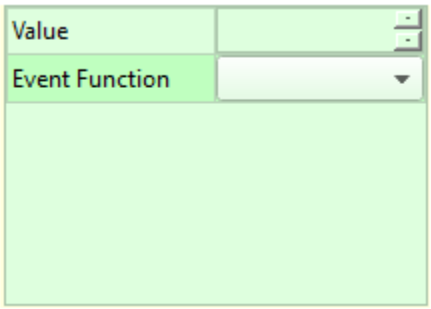




- **Radio button**

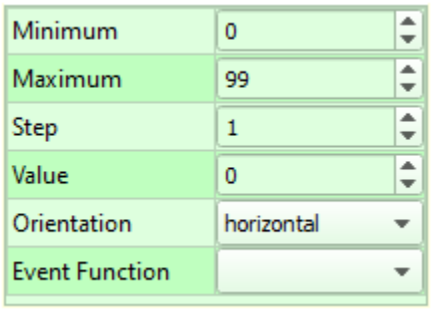
When a Radio button GUI element is inserted, the settings above will appear. The visible label of the radio button can be altered by writing in the ‘Text attribute’ field. By default, the label used is ‘Radio Button’. The ‘Value’ attribute can be used to define the default value of the selected radio button (whether the radio button is checked or not). On

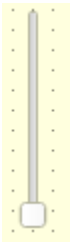
containers/elements that contains several radio buttons; only one of them can be checked at a time. Event Function will occur when Radio Button is checked.



- **List box**

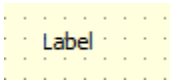
When a List box GUI element is inserted, the following options will appear. Using the 'Value' attribute, the default value of the list box can be defined. Event Functions will occur when one of the list boxes containing rows is selected.





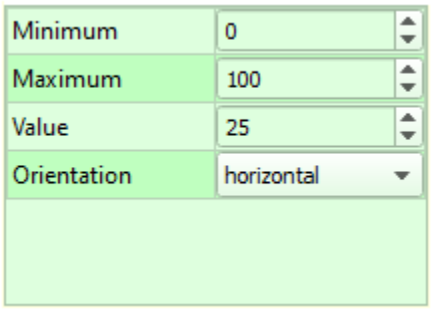
- **Horizontal slider**

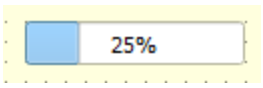
When a Slider GUI element is inserted, the following options will appear. The range of values the slider can reach can be set using the 'Minimum' and 'Maximum' fields respectively. The step of the slider which defines how much the slider increments by is set using the 'Step' attribute and the default (starting) value of slider is set using the 'Value' attribute. The 'Orientation' drop down menu is used to change the slider's orientation. Only horizontal and vertical orientations can be used. If set, Event Function is used when slider value is changed.

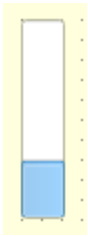


- **Label**

When Label GUI elements are inserted, they only have one customizable setting. Only the Label's visible text can edited. This is done by writing in your preferred text in the 'Text' field at the top.

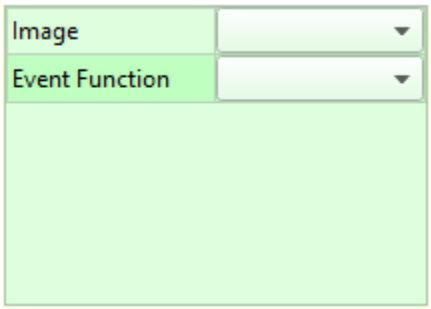




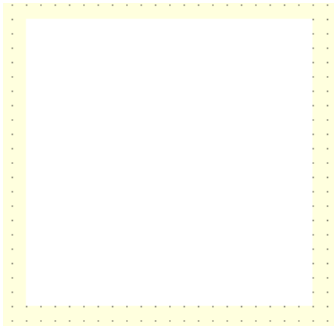


- **Horizontal progress bar**

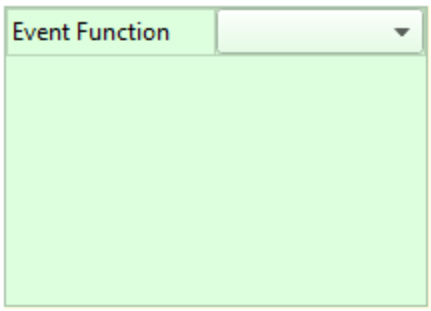
When a Progress bar GUI element is inserted, the following attributes can be customized using the available settings. Firstly, the range of allowed values the Progress bar can extend to can be set using the 'Minimum' and 'Maximum' attributes respectively. The default (starting) value of the Progress bar can be set by editing the 'Value' field. The orientation of the Progress bar can be changed by selecting an option in the drop down menu of the 'Orientation' option. Only horizontal and vertical orientations can be used on the Progress bar.



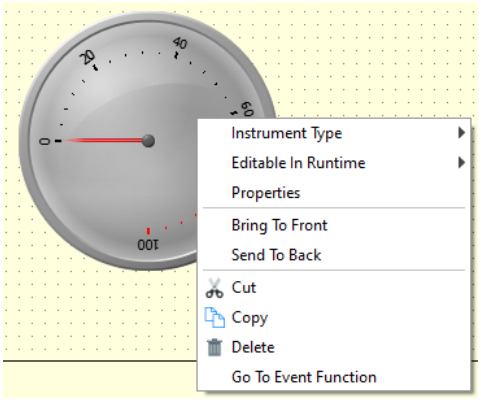
- **Image widget**



When an Image GUI element is inserted, the picture used can be set by using the 'Image' field at the top. Event Function will occur when image is selected.



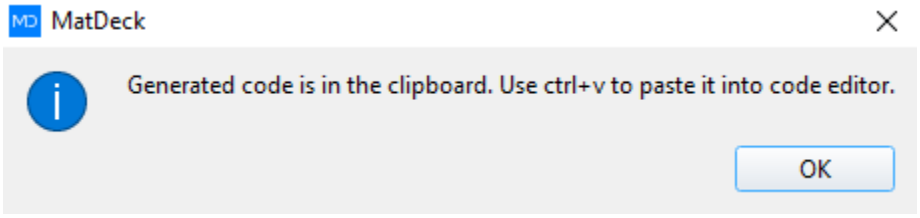
- **Instrument**



When an Instrument GUI element is inserted, the only editable option present on the working panel is the Event Function that will occur when the instrument's value changes. Every instrument has a context menu from which you can change the instrument in use and the properties of the chosen instrument.

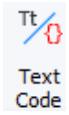
Tables, 2D and 3D graphs don't have any settings at all. All properties of those GUI elements can be customized by using the settings present in their context menus.

Once you are finished working on your widget design, you can press the 'Done' button and the designer will generate the code for the widget and copy it to your clipboard. If you have created a new widget and generated the code by pressing 'Done' for the first time, the notification window (Picture 8) will open. Once 'Done' has been pressed for the first time, Every time new code is generated it will be updated without showing this notification.



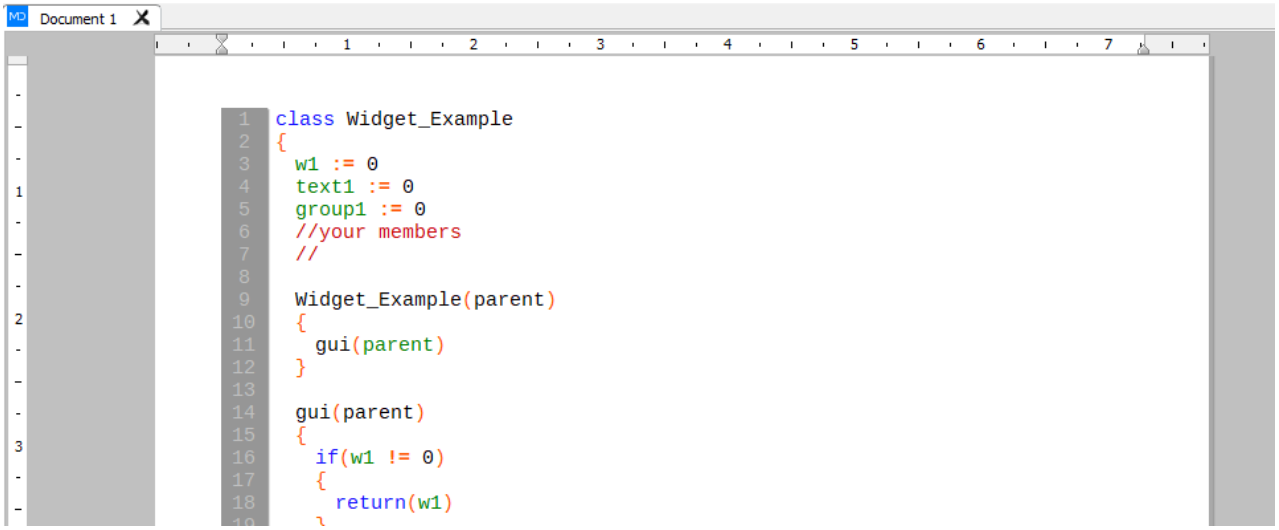
Picture 8: Notification Window

To place the generated code on a MatDeck document, we have to place the cursor on the preferred position and turn on Code mode. We can turn this mode on from Programming tab using



button, or by using the *Ctrl. + i* key combination. When Code mode is on, paste the code from the clipboard on document.

The generated code is copied to your clipboard as soon as you press 'OK'. To paste the generated code onto a MatDeck document, you must first open a new script file. Do this by opening the drop down list on the 'new' page button and pressing on the 'new script' option. In the new script file, you can paste your generated code and use it as you wish.



Picture 9: Widget Code

For every Event function we have created on the GUI Designer, the function in the code will appear with an empty body for the user to enter code for whatever they have planned.

```
31    Example_function()
32    {
33
34    }
35  }
36
37  a := Widget_Example(0)
38  show(a.w1)
```

Picture 10: Event Function In Code

At the end of code, a constructor for the created widget is added followed by the show() function. The purpose of these two lines is to create the widget as a stand-alone application when the current document is evaluated.

Notice: The preferred order of creating/changing widget designs and generating or updating code originates from the GUI Designer. If you change code directly in the MatDeck document and then, open and save these changes in the GUI Designer afterwards, all changes that were made directly in the code will be lost.

Interacting with generated GUI code

The GUI designer generates code which is used to create the visual aspects of the GUI. This means the generated code will output the GUI as it was designed by the users. The position of the GUI elements, their titles and event functions will all be coded as designed in the GUI designer window. In short, the code created by the GUI designer is for creating the individual GUI elements and their properties, any interactions or changes need to be added by the user.

Functions and code that can be used on GUIs are all included and explained in another manual. However, in this segment of the manual we will cover two important functions that are most commonly used. These functions are also included in the example at the end.

Retrieving the value of a widget / widget_value()

To retrieved and store the value of a widget present in a GUI, the function widget_value() needs to be used. It has only one argument, the name of the variable the widget is stored in or the name of the widget which is used.

```
1  A := widget_value(ExampleWidget)
```

In the example above, the function retrieves the value of the widget stored in the variable "ExampleWidget" and stores the value in the variable "A".

Connecting widget values with other widgets/ set_widget_value()

To display the value of a widget on a separate widget, the function set_widget_value() needs to be used. It has only two arguments, the widget where the value will be set and the value that will displayed on that widget respectively.

```
set_widget_value(instrument1, widget_value(vslider1))
```

In the example above, the first argument (widget where value will be applied) is a circular instrument gauge. The second argument is the value of the vertical slider present in the same GUI application. The value of the slider is retrieved using the function widget_value(). Once the function has been used, the value of the instrument will be that of the slider.

When altering or interacting with the value of a GUI element, it should be done in the event function used on it. In this case, when interacting with the value of the slider, the function set_widget_value() is used in the slider's event function.

Below is an example of this.

```
SliderEventFunction()
{
    set_widget_value(instrument1, widget_value(vslider1))
}
```

Other custom GUI functions

The functions above are used to manipulate GUI element's values and event functions. Other GUI functions also exist that can be used to interact with GUI elements and their properties. The following functions are all examples of this.

To use these functions to edit a GUI element's properties, make sure to execute the functions within the GUI code. In other words, make sure to use these functions within the function defined at the start of the code for them to have effect.

Engaging GUIs when they are used

When using these functinos in the event function segment of the code they will apply when the event function's widget is used. For example, if set_widget_text() is used in the event function of a button widget, the button's text will only change when pressed. In other words, the functions present in a event function will only be triggered when the event function is used.

Widget Text functions

The following functions are used to manipulate the text present on a GUI element.

set_widget_text() – this function can be used to set the text present on a GUI element. The function has only two arguments, the widget being used and the string text respectively.

```
set_widget_text(button1, "text")
```

widget_text() – when used, it will retrieve the text present on a GUI element. Only one argument is used for the function, the name of the variable or widget that is used. In the example below, the text retrieved is stored in a varaiable. The function can be used outside of variables and nested in other functions as well.

```
b := widget_text(button1)
```

Widget Position functions

The following functions are used to manipulate the position of GUI elements in the GUI application.

set_pos() – the position of GUI elements can be changed via the GUI designer and custom GUI functions. GUI functions may be preferred for positioning as they are more precise. ***set_pos()*** can be used to position GUI elements. The 3 arguments are the widget used, position regarding X axis and positioning regarding Y axis respectively.

```
set_pos(button1, 100, 100)
```

pos_x() – used to retrieve the position of the widget regarding the X axis. The function only has one argument, the widget or variable used.

```
b := pos_x(button1)
```

pos_y() – used to retrieve the position of the widget regarding the Y axis. The function only has one argument, the widget or variable used.

```
b := pos_y(button1)
```

Widget Sizing functions

The following functions are used to manipulate the size of GUI elements.

set_size – the size of GUI elements can be changed via the GUI designer and custom GUI functions. The custom GUI function has 3 arguments. The arguments used are the widget/variable used, the width of the widget and height of the widget respectively.

```
set_size(button1, 100, 100)
```

width() – this function can be used to retrieve the horizontal width of a selected widget. Only one argument is needed; the variable or widget used.

```
b := width(button1)
```

height() – this function can be used to retrieve the vertical height of a selected widget. Only one argument is needed; the variable or widget used.

```
b := height(button1)
```

Please note: when using these functinos in the event function segment of the code they will apply when the event function's widget is used. For example, if set_widget_text() is used in the event function of a button widget, the button's text will only change when pressed. In other words, the functions present in a event function will only be triggered when the event function is used.

Executing and Outputting GUIs

Once GUI designs are completed, the generated code is copied to the clipboard. This generated code can be used in different formats which include: MatDeck documents, Python script documents, and MatDeck script documents.

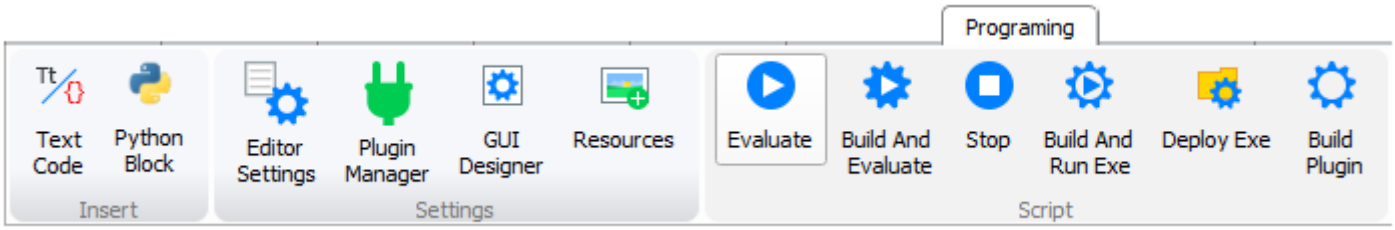
GUIs can be outputted using the following methods:

- Evaluation of generated code
- Embedding the widget

Evaluation of generated code

GUI applications can be executed by evaluating the generated code in MatDeck document. By default, the GUI application will appear as a standalone window separate to the MatDeck document in which it is evaluated. To execute generated code in MatDeck, first, the code must be placed in the appropriate format. This means the generated code can be copied to a programming block, MatDeck script document and Python script document.

Once the code is placed in the appropriate location, press the 'Evaluate' icon to evaluate the current code. Evaluate is located in the 'Programming' toolbar.



Embedding the widget

For users that prefer to use their designed GUI in the MatDeck document instead of a standalone window, it can also be embedded. Embedded GUI need to be embedded in canvases only.

To embed your GUI, the function embed_widget() needs to be used. It has only one argument, the name of the variable in which the widget is stored or the name of the widget which is to be embedded. The widget is used in canvases. In the example below, the generated code was placed in Text/Code block in a standard MatDeck document.

```
embed_widget(a.w1)
```

In the example above, the argument used for designed GUI is a.w1. When GUIs are being embedded, the GUIs class and the variable in which the widget is stored both need to be used. The GUI's class is written first followed by the widget's variable. The two variables are separated by a single full stop.

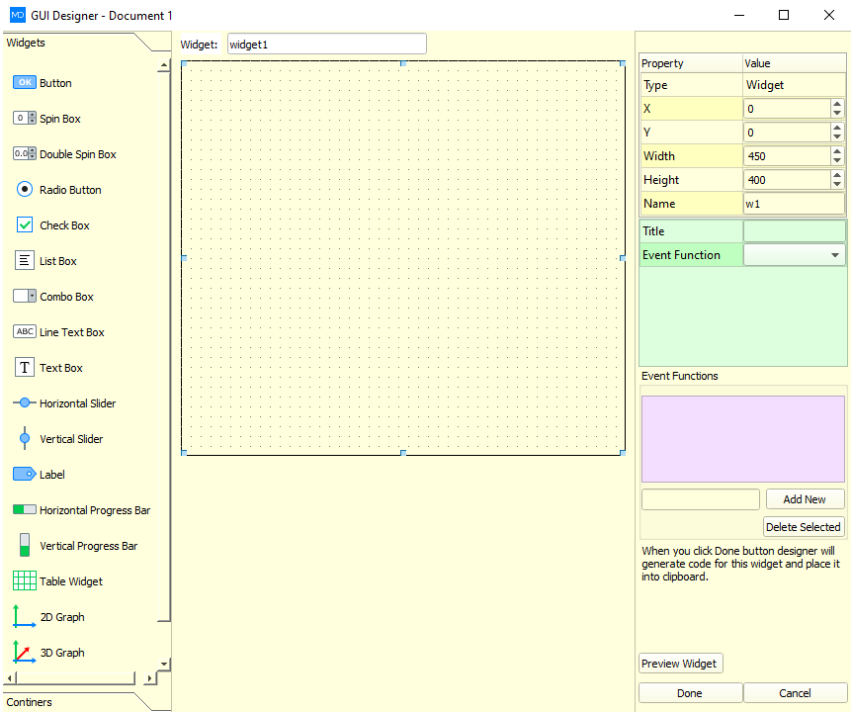
```
embed_widget(a.w1)
```

The name of variable in which the GUI's class is stored. This variable is normally located close to the end of the generated code.

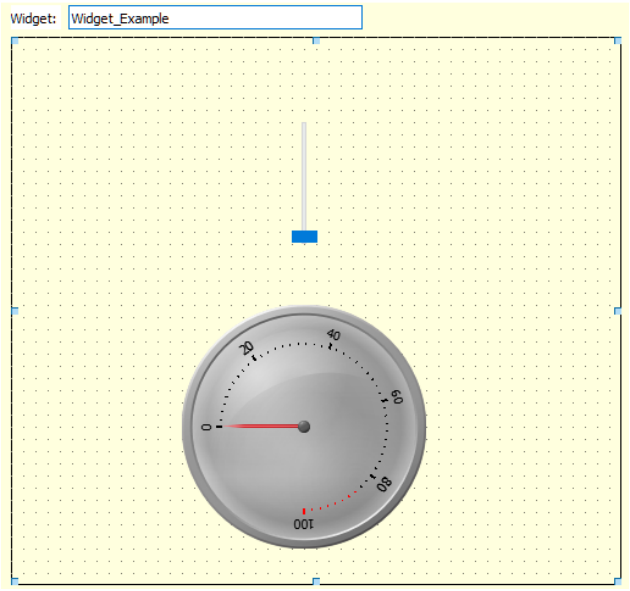
The second variable used is the variable which stores the actual widget. The widget is normally initiated using the function widget().

GUI Designer Instrument Example

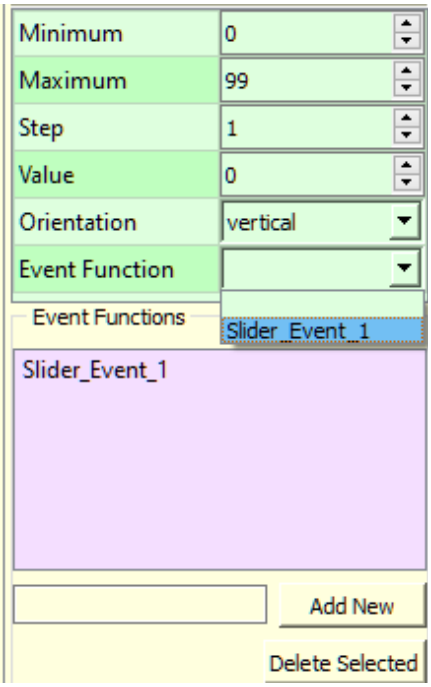
As mentioned in the manual earlier, to open the GUI designer we press on 'Widget' under the 'Programming' toolbar. Once the new window opens we will make sure to not to select any preexisting widgets in the file and press 'New' to open a new empty GUI designer window. This to make sure we don't delete or replace any existing widgets with our new one.



The window above will open. In this example, we will connect two GUI elements together and interact between them. For this one, we will use an 'Instrument Widget' and a 'Vertical Slider' located on the left-hand side 'Widgets' list. By pressing on the icons of each and then pressing on the main parent widget workspace, you will be able to place the GUI elements as so:



Once you have arranged the necessary GUI elements like the image above, we will move on to adding a new event function for the vertical slider. We will do this by writing in the 'Add New' field of the event functions window and pressing 'Add New' to add this new event function to the widget. To add this specific event function to the slider, it has to be selected from the drop down list that appears when 'Event Function' is pressed. It will appear as below:



The reason we do this is so that the function/capabilities of the slider are now named and can be interacted with by referring to its event function. In short, its event function acts as a way for us to connect that slider to other elements on the parent-widget work area.

Now that you have the necessary GUI elements placed and edited on the parent widget, we can press 'Done' and move onto adding the generated code to a new MatDeck document.

Like the manual we are now going to open a new MatDeck script file where we will our generated code will be pasted in once we are finished and pressed 'Done'. Your generated code should appear as so:

```
1
2
3  class Widget_Exmaple
4  {
5      w1 := 0
6      instrument1 := 0
7      vslider1 := 0
8  //your members
9  //
10
11     Widget_Exmaple(parent)
12     {
13         gui(parent)
14     }
15
16     gui(parent) {...}
32
33     Slider_Event_1()
34     {
35
36     }
37 }
38
39 a := Widget_Exmaple(0)
40 show(a.w1)
```

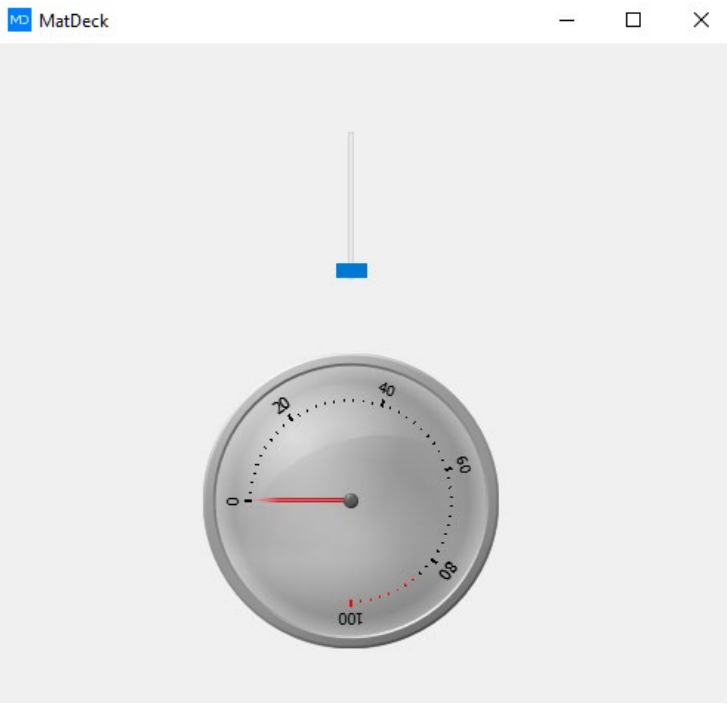
The collapsed brackets on line 16 contain all necessary Meta data such as the size, positioning and the name of event functions associated with our widgets. So far, we have added the necessary GUI elements and added an event function for the slider. For our final step we simply have to use the event function we created earlier to connect the Vertical Slider to the Instrument Widget. On line 33 our event function is located.

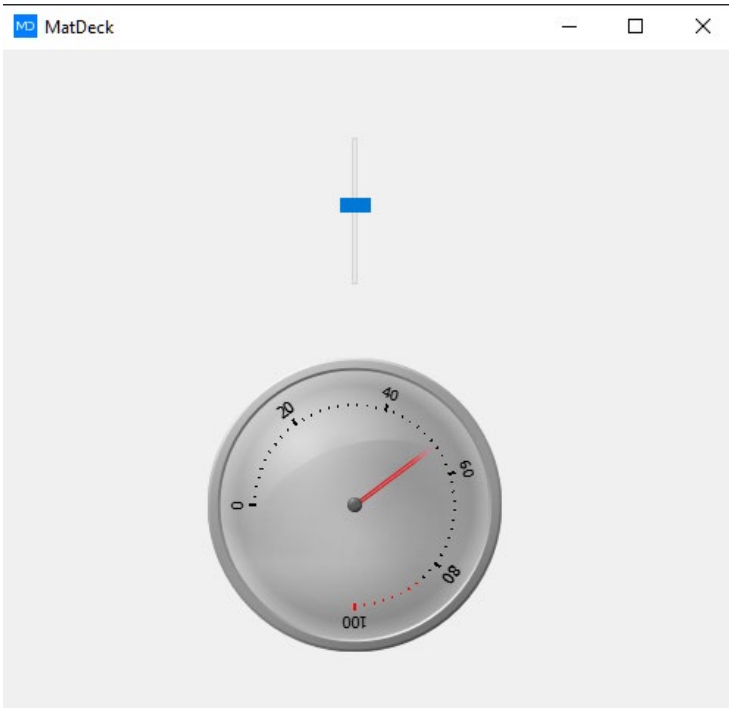
```
33     Slider_Event_1()
34     {
35
36     }
```

As you can see, the brackets are empty as we haven't used the slider's function or data for anything yet. To connect the slider's value/data to the instrument we will use the function set_widget_value(,). Its first argument is the name of the widget we wish to use the value on and its second argument is the value we wish to use. As we are using the value generated by the slider we have to use the function widget_value() in the second argument so that we can use the slider's value. Its argument is the name of the widget or GUI element we wish to use. The new line of code should look so using the arguments mentioned above.

```
33     Slider_Event_1()
34     {
35         set_widget_value(instrument1, widget_value(vslider1))
36     }
```

Now the event function of the slider sets the value of the instrument widget to the value of the slider. This means moving the slider will change the value of the instrument widget. Press the 'Evaluate' button on the programming tab to execute your newly created widget. It will appear and interact as below:





Note: Be careful to not forget to name your GUI elements and widgets with appropriate names while working with the GUI designer. This will both make your code easier for others to interpret and easier for you to read and make necessary changes including mistakes.