Informatik mit Matlab – Labor 6 / App Designer

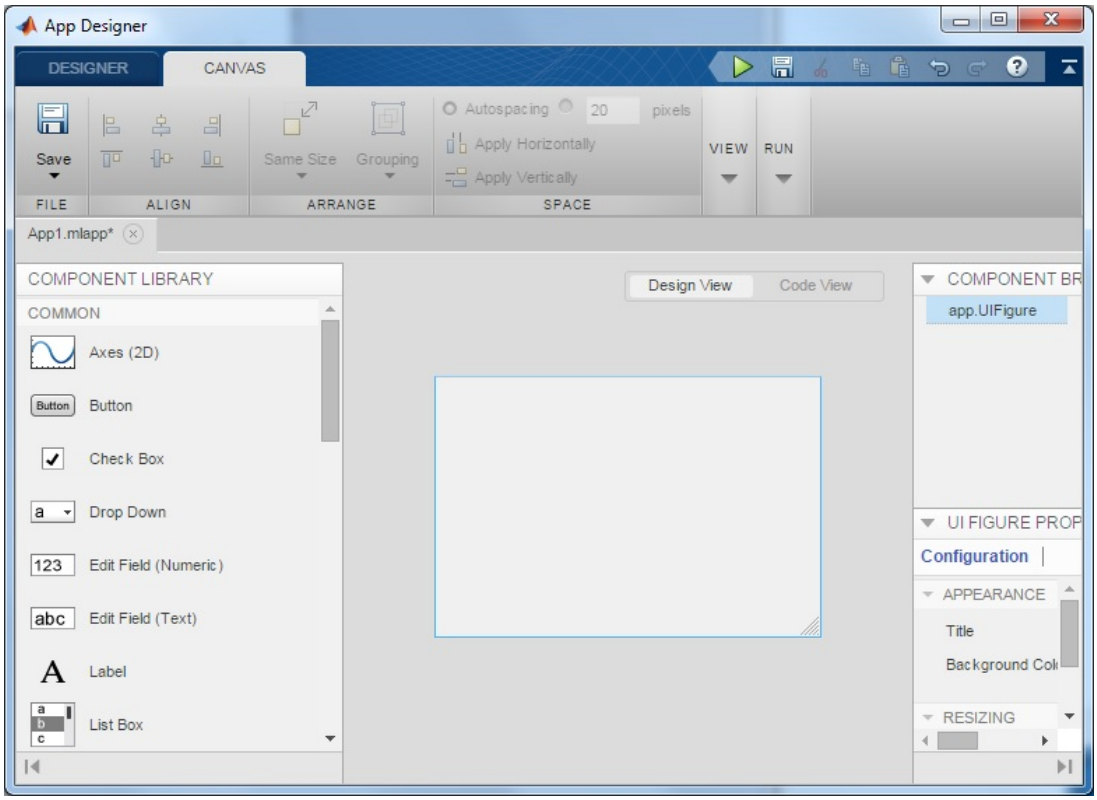
Aufgabe gui1:

Erstellen Sie die die App Designer-Fenster, wie sie in der Vorlesung vorgestellt wurden. Modifizieren Sie die Fenster, indem Sie andere App Designer-Objekte einbauen.

Vorgehen:

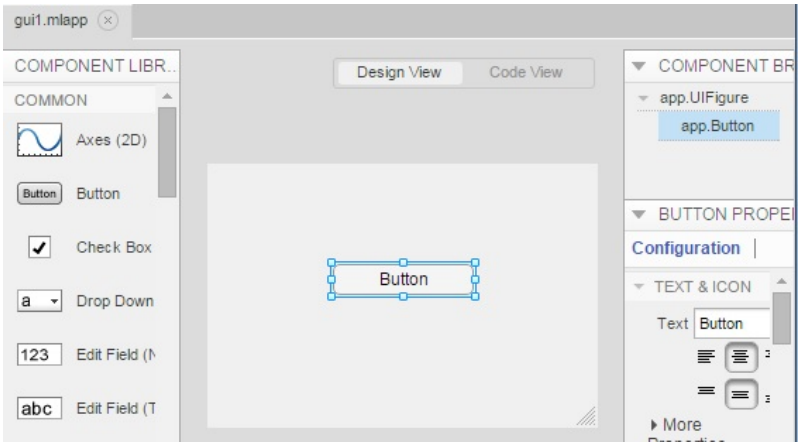
Aufruf im MATLAB Command Window: `>> appdesigner`

Es öffnet sich der **Layout-Editor** im Design View:

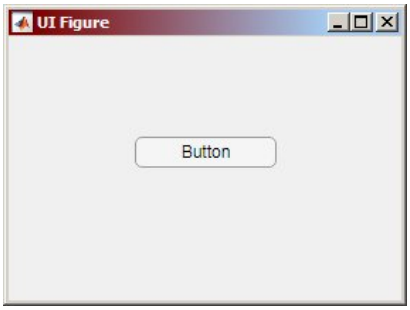


App Designer-Element: **Button** einfügen:

Maus-Klick in der Iconleiste (links auf das oberste Feld der linken Reihe Aufschrift „Button“).



Element mit gedrückter linker Maustaste in den Layout-Bereich ziehen.



Abspeichern: Namen ***gui1.mlapp***
Aufruf: `>> gui1`

Umschalten auf Tab: **Code View**

```
classdef gui1 < matlab.apps.AppBase   % child class of AppBase < handle

  % Properties that correspond to app components
  properties (Access = public)
    UIFigure  matlab.ui.Figure           % Figure
    Button    matlab.ui.control.Button   % Push Button
  end

  % App initialization and construction
  methods (Access = private)

    % Create UIFigure and components, called by the constructor
    function createComponents(app)
      % Create UIFigure
      app.UIFigure = uifigure;
      app.UIFigure.Position = [100 100 277 187];
      app.UIFigure.Name = 'UI Figure';
      setAutoResize(app, app.UIFigure, true)
      % Create Button
      app.Button = uibutton(app.UIFigure, 'push');
      app.Button.Position = [90 95 100 22];
    end
  end

  methods (Access = public)

    % Constructor for app
    function app = gui1()
      % Create and configure components
      createComponents(app);               % Method of gui1
      % Register the app with App Designer
      registerApp(app, app.UIFigure);
      if nargout == 0
        clear app
      end
    end

    % destructor: Code that executes before app deletion
    function delete(app)
      % Delete UIFigure when app is deleted
      delete(app.UIFigure)
    end
  end
end
```

Hinweis: Im Editor des App Designers lassen sich die Zeilen jedoch nicht editieren. Sie können aber aus dem Editor des App Designers die Zeilen des Codes herauskopieren und die gesamte Klasse als „gui1.m“ abspeichern. Diese Variante läuft auch ohne die Oberfläche des App Designers. Ein Teil der App Designer Elemente sind in der MATLAB-Hilfe nicht dokumentiert, so genannte „undocumented features“, die sich in späteren Releases von MATLAB ändern können, siehe z.B. „http://undocumentedmatlab.com/“.

Teile der Klasse gui1:

Eine Übersicht über die dokumentierten Features findet man in
https://de.mathworks.com/help/matlab/components-in-app-designer.html und in
https://de.mathworks.com/help/matlab/creating_guis/choose-components-for-your-app-designer-app.html

gui1 ist von ***matlab.apps.AppBase*** abgeleitet: `classdef gui1 < matlab.apps.AppBase`
Die Handle-Klasse *AppBase* ist zurzeit nicht dokumentiert. Sie steht im M-File unter
„C:\Program Files\MATLAB\R2018a\toolbox\matlab\appdesigner\appdesigner\runtime\+matlab\+apps\AppBase.m"
AppBase hat keine weiteren properties und die folgenden Methoden:
delete, *createCallbackFcn*, *runStartupFcn*, *registerApp*, *setAutoResize*

Properties von *gui1* sind die ui-Elemente der App, hier also eine *Figure* und ein *Button*:

```
UIFigure    matlab.ui.Figure
Button      matlab.ui.control.Button
```

Zur Klasse *ui.Figure* gibt es einen „Hilfs-Konstruktor" ***uifigure***, beschrieben in
https://de.mathworks.com/help/matlab/ref/uifigure.html und in
https://de.mathworks.com/help/matlab/ref/matlab.ui.figureappd-properties.html
für die „UI Figure Properties".

Zur Klasse *ui.control.Button* einen „Hilfs-Konstruktor" ***uibutton***, beschrieben in
https://de.mathworks.com/help/matlab/ref/uibutton.html und in
https://de.mathworks.com/help/matlab/ref/matlab.ui.control.button-properties.html
für die „Button Properties".

Konstruktor *gui1*: `function app = gui1`
erzeugt die Komponenten mittels der Methode *createComponents* und registriert anschließend die App mit der AppBase-Methode *registerApp*.

Methode *createComponents*:
erzeugt in unserem ersten Beispiel ein Fenster:

```
app.UIFigure = uifigure;
```

und setzt anschließend dessen Position und Name.
Als Weiteres wird der Push Button erzeugt:

```
app.Button = uibutton(app.UIFigure, 'push');
```

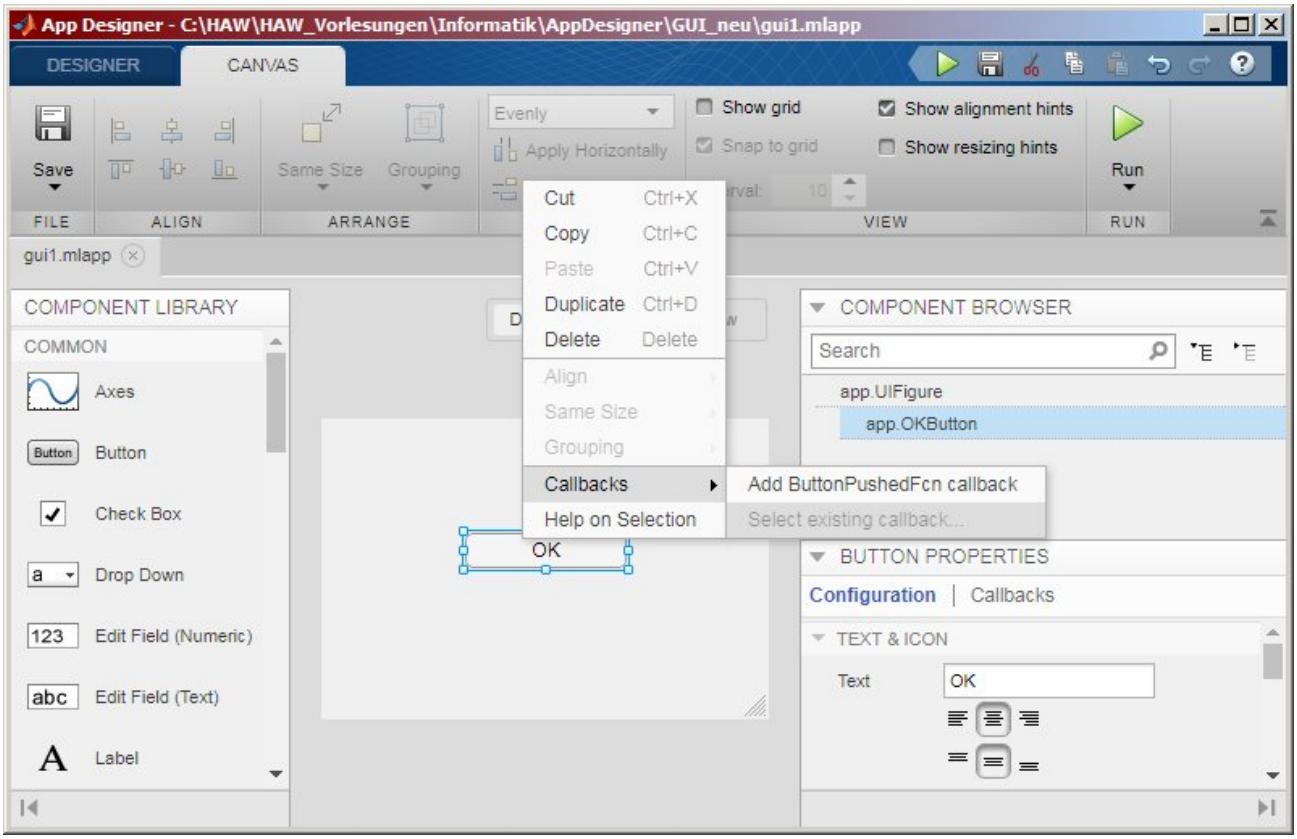
und dessen Position gesetzt.

Callback Function für den Button

Ein Push Button-Callback war in unserem ersten Entwurf noch nicht vorgesehen. Deshalb zurück zum App Designer in den Design View:

Im Component Browser können die Eigenschaften des Buttons verändert werden, z.B. im Feld „Text“ die Beschriftung des Buttons auf „OK“.

Für die Callback Function: Klick rechte Maustaste auf den Button,
Auswahl „Callbacks > Add ButtonPushedFcn callback”:



Im Tab Code View erscheint nun die neue, editierbare Methode *OKButtonPushed*:

```
% Button pushed function: OKButton
function OKButtonPushed(app, event)

end
```

Die Callback Function *OKButtonPushed* wird außerdem in der Methode *createComponents* mit dem Push Button über die AppBase-Methode *createCallbackFcn* verknüpft:

```
app.OKButton.ButtonPushedFcn =
        createCallbackFcn(app, @OKButtonPushed, true);
```

Die Callback-Funktion ist noch leer. Erweitern Sie den Callback-Funktionsrumpf mit den unten angegebenen vier Zeilen (neues Grafik-Fensters: *figure*, *axes*, Geraden-Plot und Axis-Definition):

```
% Button pushed function: OKButton
function OKButtonPushed(app, event)
    f = figure;
    a = axes(f);
    plot( a, [2,5], [1,3] );
    axis( [0,8,0,5] );
end
```

Die **Köpfe** der Callback-Funktionen (hier: function *OKButtonPushed*) legt der App Designer bereits automatisch an. Sie müssen nur noch die Befehle im Rumpf der Funktionen hinzufügen, also die drei Zeilen mit den Befehlen *figure*, *plot* und *axis*.

Aufruf im MATLAB Command Window: `>> gui1`

Test der Funktionalität: Push Button klicken -> Aufruf von *OKButtonPushed.*

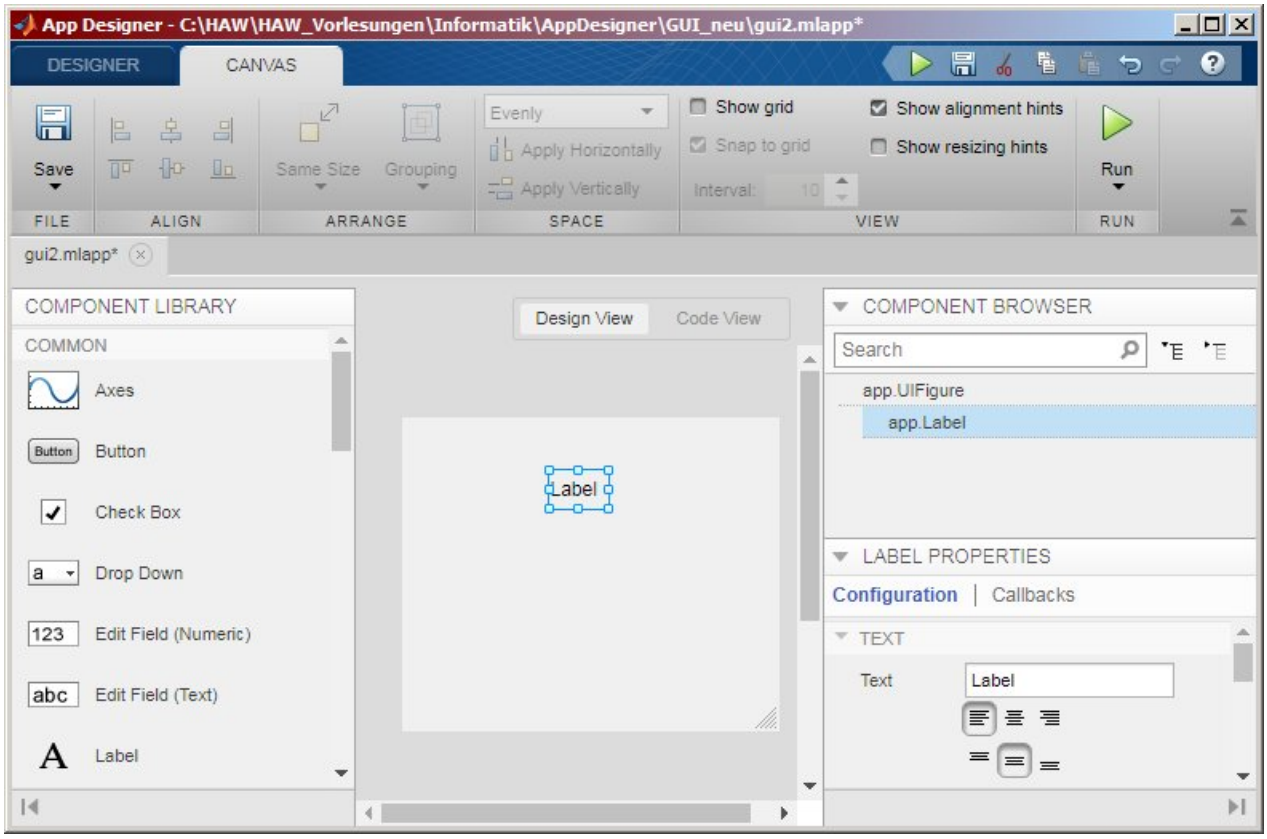
Aufgabe gui2:

Text-Ausgabefeld (Label)

Erneut den App Designer aufrufen mit einem neuen Layout:

```
>> appdesigner
```

Label-Element in Layout ziehen + Abspeichern unter dem Namen ***gui2***



Aufruf: `>>gui2`

Zum **Ändern** des Ausgabe-Textes gibt es zwei Möglichkeiten:

1. im ***Layout*** fest einen Text eintragen.
2. Textfeld zur ***Laufzeit*** mit Text belegen.

zu 1.: Ändern Sie rechts unten in den **Label Properties** den Eintrag im Feld „Text“, z.B. in „Hello, world“.

zu 2.: Als M-File abspeichern und die Methode *createComponents* editieren:

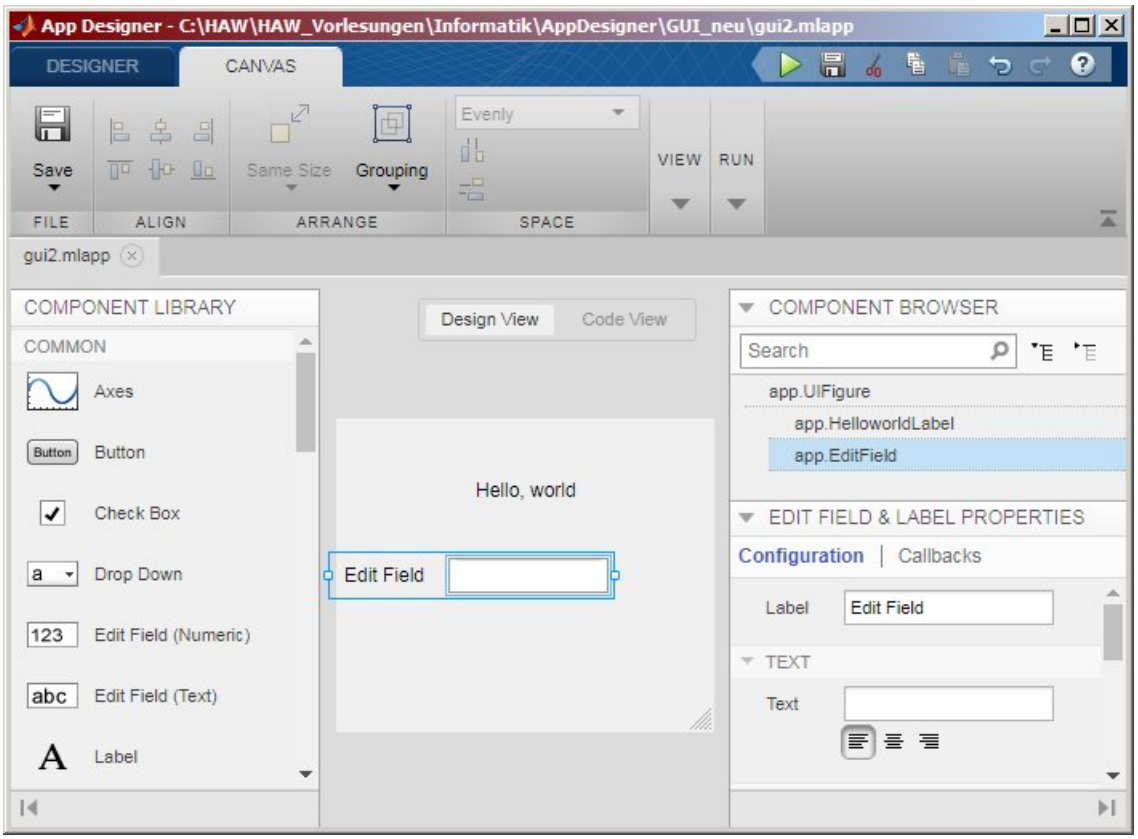
```
% Create HelloworldLabel
app.HelloworldLabel = uilabel(app.UIFigure);
app.HelloworldLabel.Position = [88 141 68 22];
app.HelloworldLabel.Text = 'Hello, world';
```

Text-Eingabefeld (Edit Field)

```
>> appdesigner
```

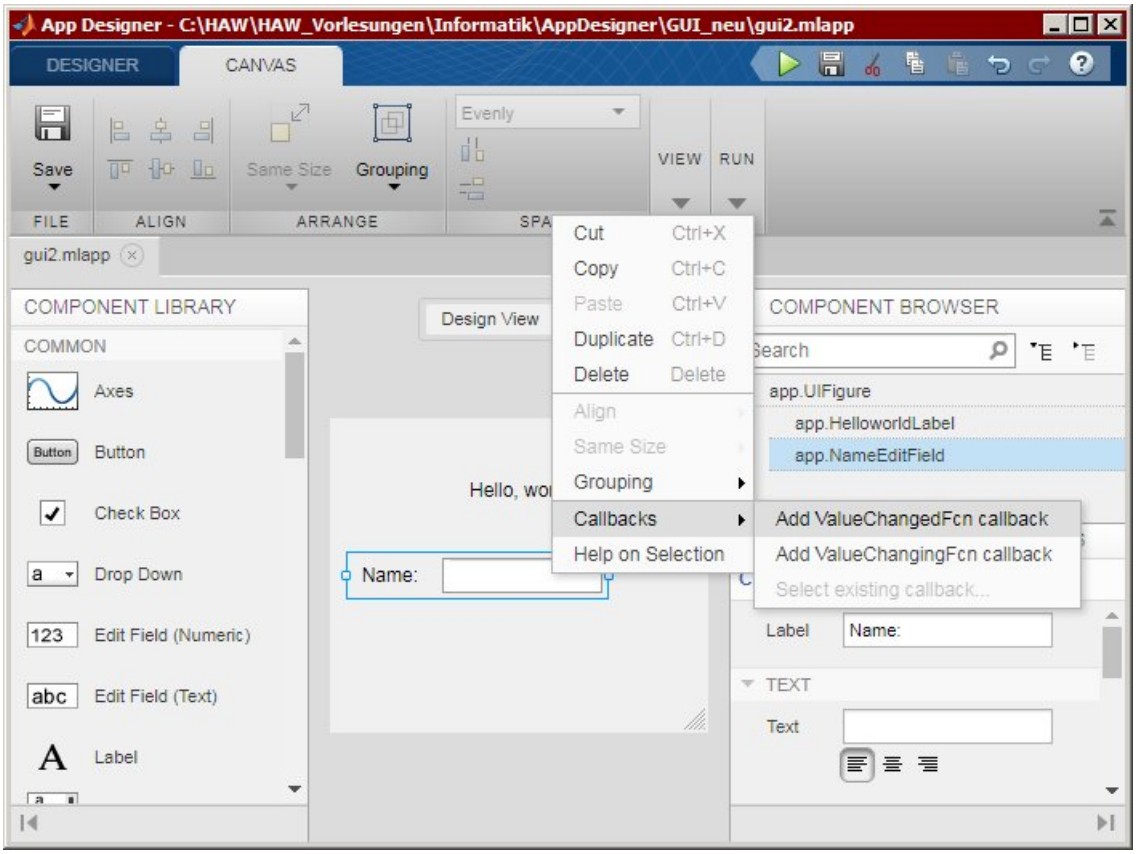
Menu *Open:* „gui2.mlapp“

Unterhalb des Label-Feldes ein „**Edit Field (Text)**“ einfügen:



Den Text links vor dem Edit Field können Sie rechts unten in "EDIT FIELD & LABEL PROPERTIES" im Feld **Label**, z.B. von „Edit Field“ in „Name:“.

Um den Text auszulesen, den Sie später im Edit Field eingeben können, brauchen Sie noch eine Callback Function zum Edit Field, über Klick mit rechter Maustaste auf Edit Field + Auswahl „Callbacks > Add ValueChangedFcn callback“:



Im Tab Code View erscheint nun folgender Code:

```
classdef gui2 < matlab.apps.AppBase

  % Properties that correspond to app components
  properties (Access = public)
    UIFigure            matlab.ui.Figure
    HelloworldLabel     matlab.ui.control.Label
    NameEditFieldLabel  matlab.ui.control.Label
    NameEditField       matlab.ui.control.EditField
  end

  methods (Access = private)

    % Value changed function: NameEditField
    function NameEditFieldValueChanged(app, event)
      value = app.NameEditField.Value;

    end
  end

  % App initialization and construction
  methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)
      % Create UIFigure
      app.UIFigure = uifigure;
      app.UIFigure.Position = [100 100 236 196];
      app.UIFigure.Name = 'UI Figure';
```

```
      % Create HelloworldLabel
      app.HelloworldLabel = uilabel(app.UIFigure);
      app.HelloworldLabel.Position = [88 141 68 22];
      app.HelloworldLabel.Text = 'Hello, world';

      % Create NameEditFieldLabel
      app.NameEditFieldLabel = uilabel(app.UIFigure);
      app.NameEditFieldLabel.HorizontalAlignment = 'right';
      app.NameEditFieldLabel.Position = [15 88 41 22];
      app.NameEditFieldLabel.Text = 'Name:';

      % Create NameEditField
      app.NameEditField = uieditfield(app.UIFigure, 'text');
      app.NameEditField.ValueChangedFcn =
               createCallbackFcn(app, @NameEditFieldValueChanged, true);
      app.NameEditField.Position = [71 88 100 22];
    end
  end

  methods (Access = public)

    % Construct app
    function app = gui2
      % Create and configure components
      createComponents(app);
      % Register the app with App Designer
      registerApp(app, app.UIFigure);
      if nargout == 0
        clear app
      end
    end

    % Code that executes before app deletion
    function delete(app)
      % Delete UIFigure when app is deleted
      delete(app.UIFigure)
    end
  end
end
```

Der einzige Bereich, der im App Designer editierbar ist, befindet sich in der Methode *NameEditFieldValueChanged*. Dort ist bereits die Variable *value* angelegt, die den vom Anwender eingegebenen Text ausliest. Als ersten Test lassen wir uns diesen Wert in einer *msgbox* ausgeben:

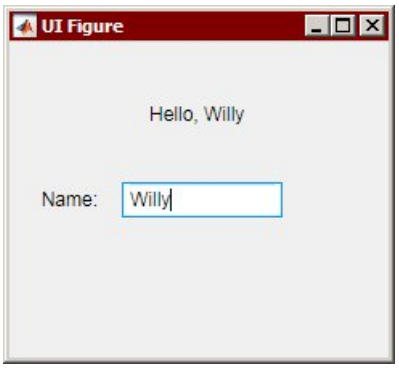
```
    % Value changed function: NameEditField
    function NameEditFieldValueChanged(app, event)
      value = app.NameEditField.Value;
      msgbox( value );
    end
```

Beim Test der App erhält man nun nach Eingabe eines Textes und der Bestätigung mit der Enter-Taste den Text in einer *msgbox* angezeigt.

Wir übernehmen den Text jetzt aber in das Label-Feld, indem wir statt des msgbox-Aufrufs schreiben:

```
      app.HelloworldLabel.Text = [ 'Hello, ', value];
```

Geben Sie nun nach dem Aufruf der App als Name „Willy“ ein, erhalten Sie die Ausgabe:



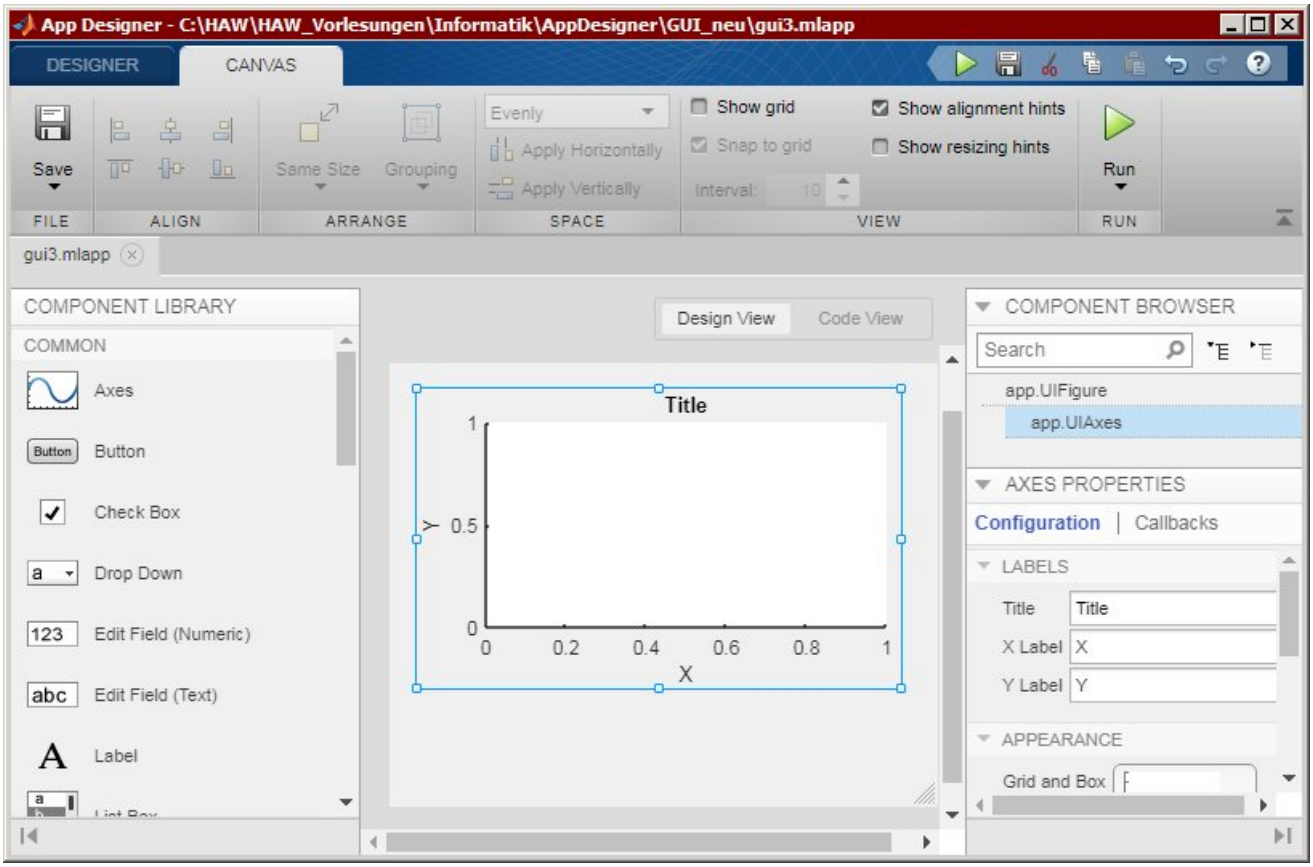
Aufgabe gui3:

Grafikobjekt

Erneut den App Designer aufrufen mit einem neuen Layout:

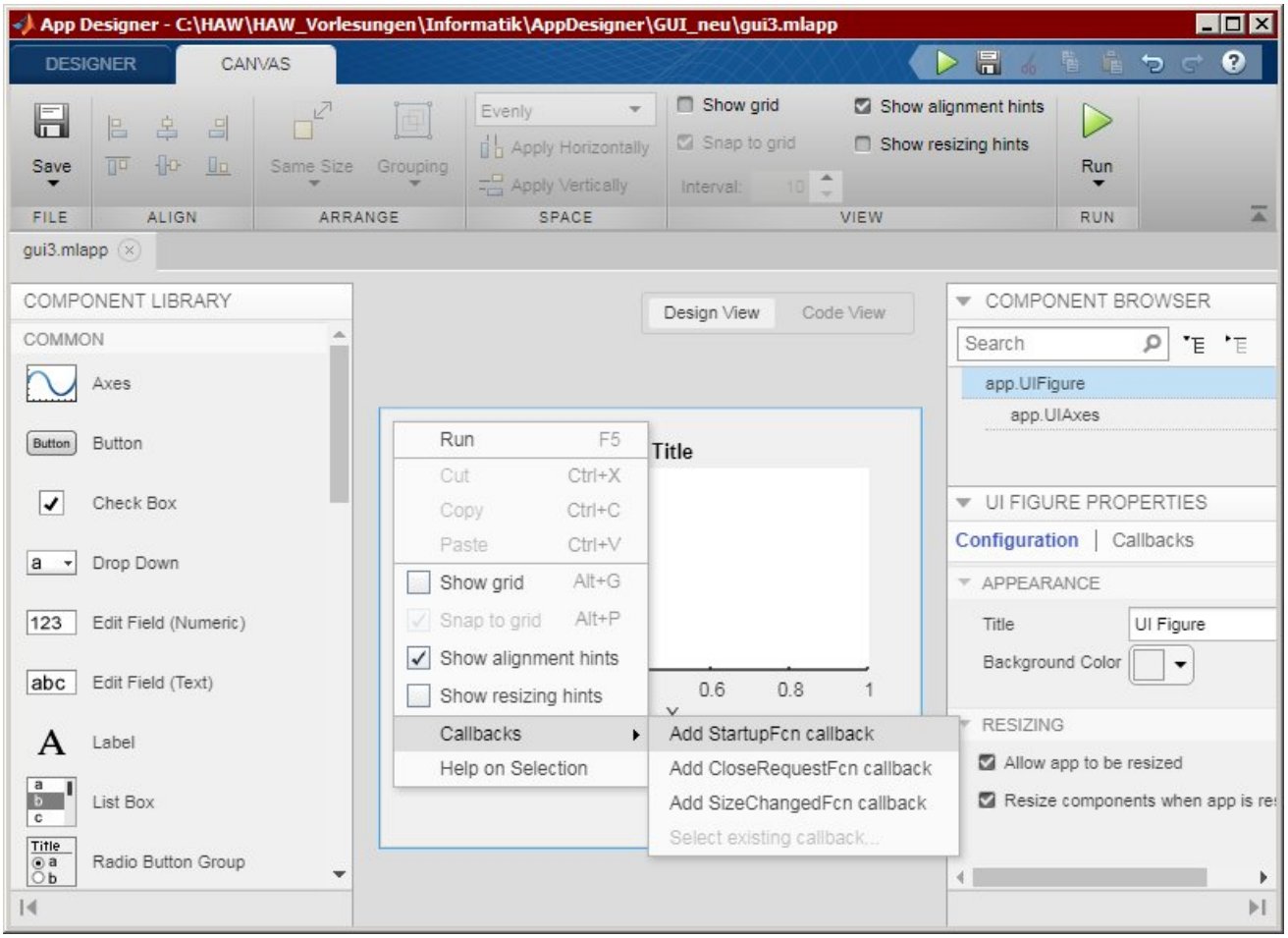
```
>> appdesigner
```

Axes-Element in Layout ziehen + Abspeichern unter dem Namen ***gui3***



Im Axes-Bereich soll sofort nach dem Start der App eine Grafik angezeigt werden. Dazu müssen wir als nächstes eine Callback Function erzeugen, die nach dem Start aufgerufen wird.

Wählen Sie dazu mit der Maus im Design View die *Figure* an, nicht die gerade eingefügte *Axes*. Klick mit rechter Maustaste und Auswahl „Callbacks > Add StartupFcn callback“:



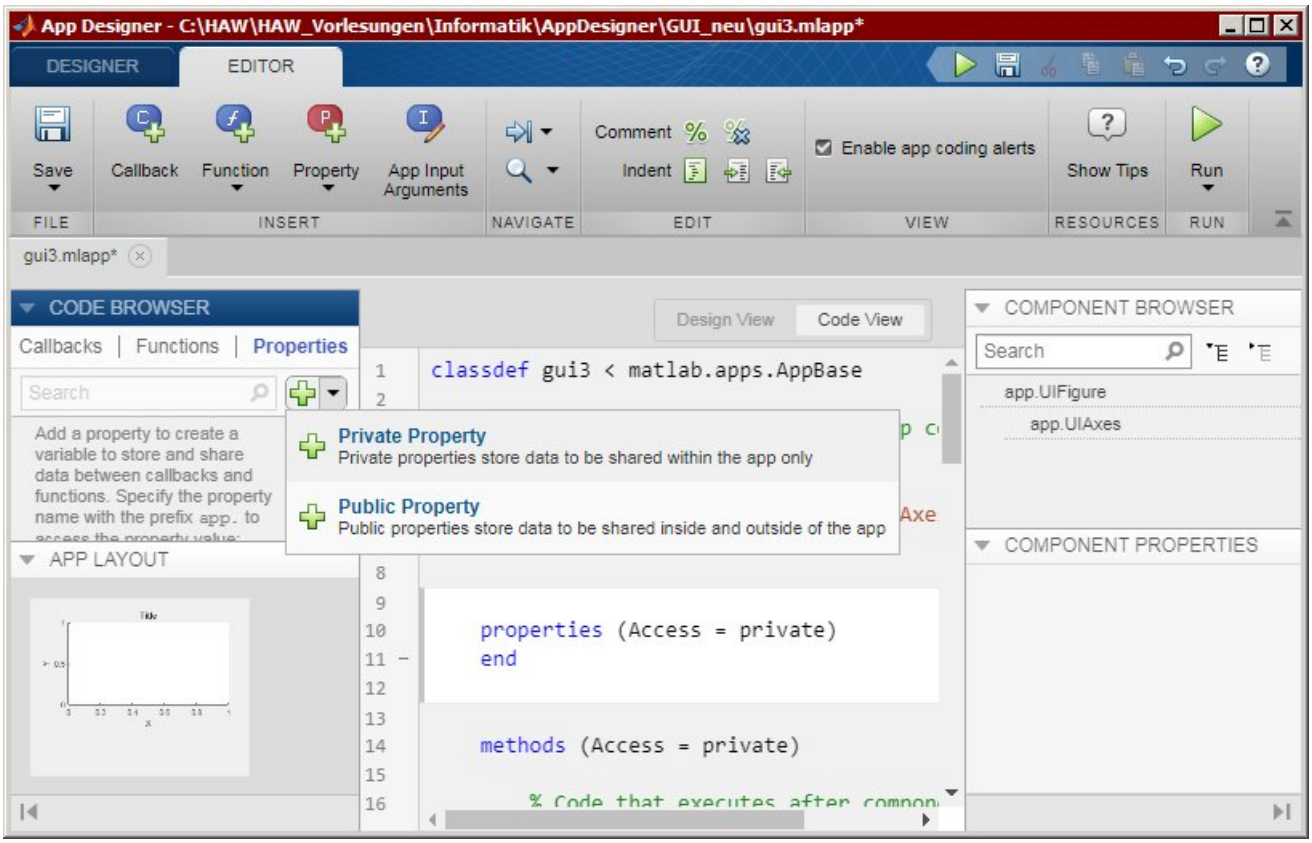
Im Code View editieren Sie die startupFcn, um mit der Funktion *surf* eine 3D-Grafik zur Fläche „sin(R) / R“ zu erzeugen:

```
% Code that executes after component creation
function startupFcn(app)
  % Grafik in Axes zeichnen
  [X,Y] = meshgrid(-8:.5:8);
  R = sqrt(X.^2 + Y.^2) + eps;
  app.handles.Z = sin(R)./R;
  surf( app.UIAxes, app.handles.Z )
end
```

Um eindeutig zu sein, wird der surf-Funktion als erstes Argument die *Axes* übergeben, in die die Grafik gezeichnet werden soll, also die *app.UIAxes* unserer Klasse *app*.

Die Z-Werte der Grafik werden wir später noch benötigen. Deshalb erzeugen wir für die Klasse *gui3* eine weitere property *handles* für unsere Daten.

Gehen Sie dazu in den Code View und wählen Sie im CODE BROWSER auf der linken Seite den Tab Properties. Wählen Sie im Pop-up Menü unter dem Pfeil **v** die Option „Private Property“:



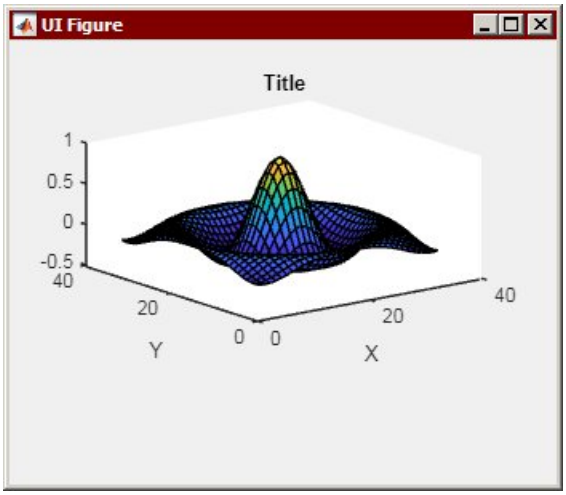
Im Code View erscheint nun ein neuer, editierbarer *properties* Bereich. Hier legen Sie die neue struct-Variable *handles* an:

```
properties (Access = private)
  handles     % data struct
end
```

Die Objekt-Variable *handles* haben wir in der *startupFcn* verwendet, um die Z-Werte abzuspeichern:

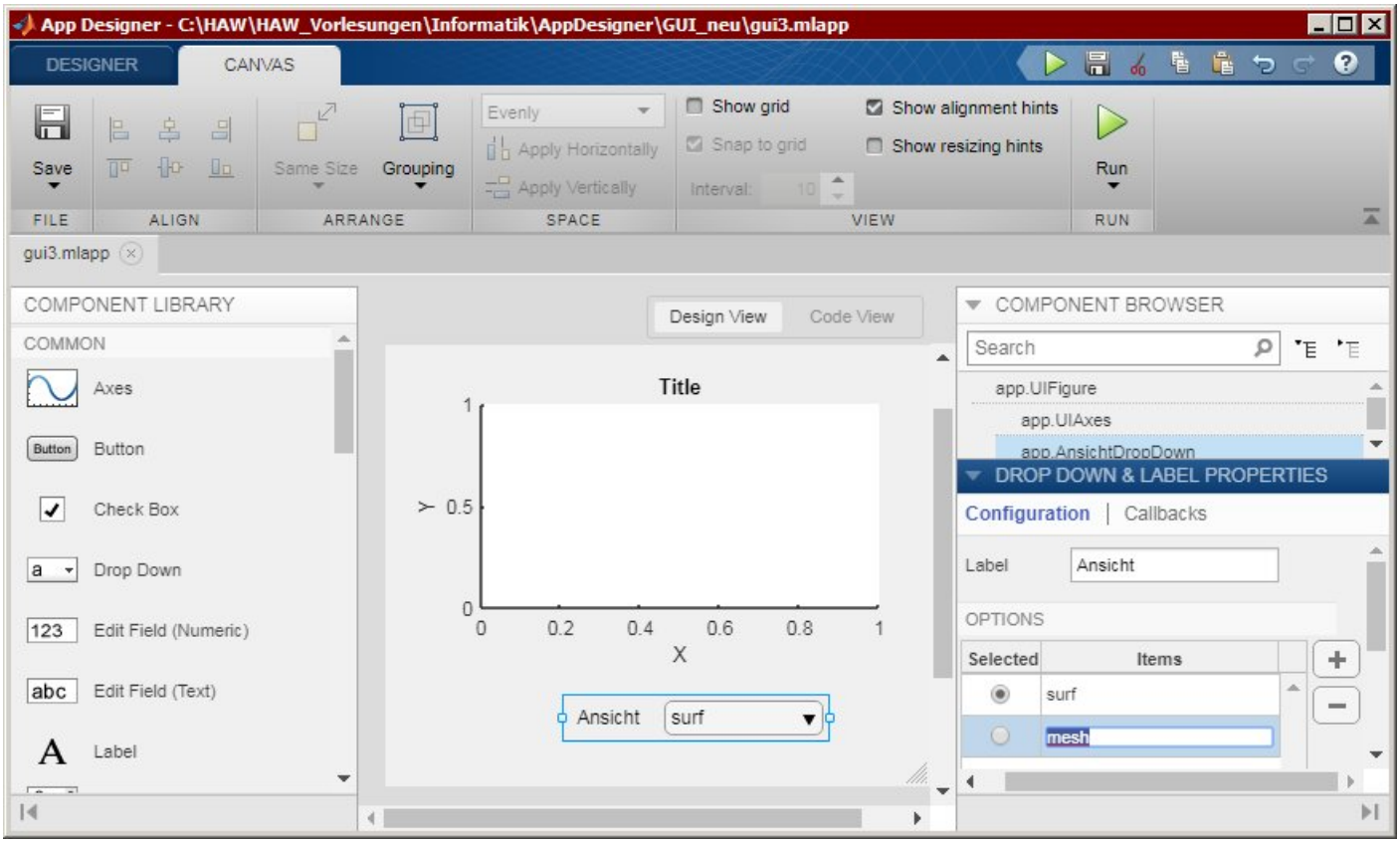
```
app.handles.Z = sin(R)./R;
```

Rufen wir jetzt unsere App auf, so wird die Grafik gezeichnet:



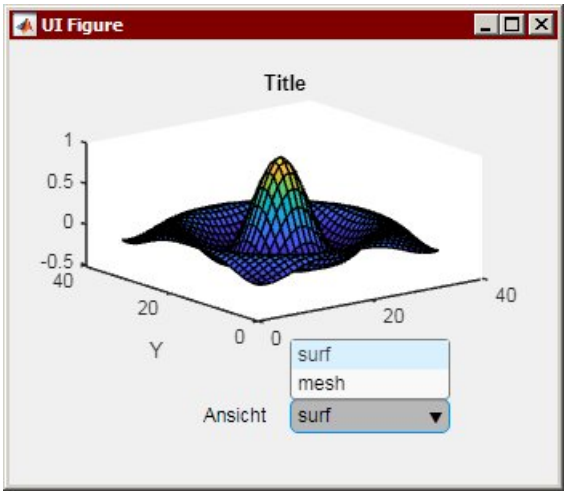
Drop-Down-Menü

Wir erweitern unsere App durch ein neues Control-Element, ein Drop-Down-Menü, das wir im Design View unterhalb der Axes platzieren.

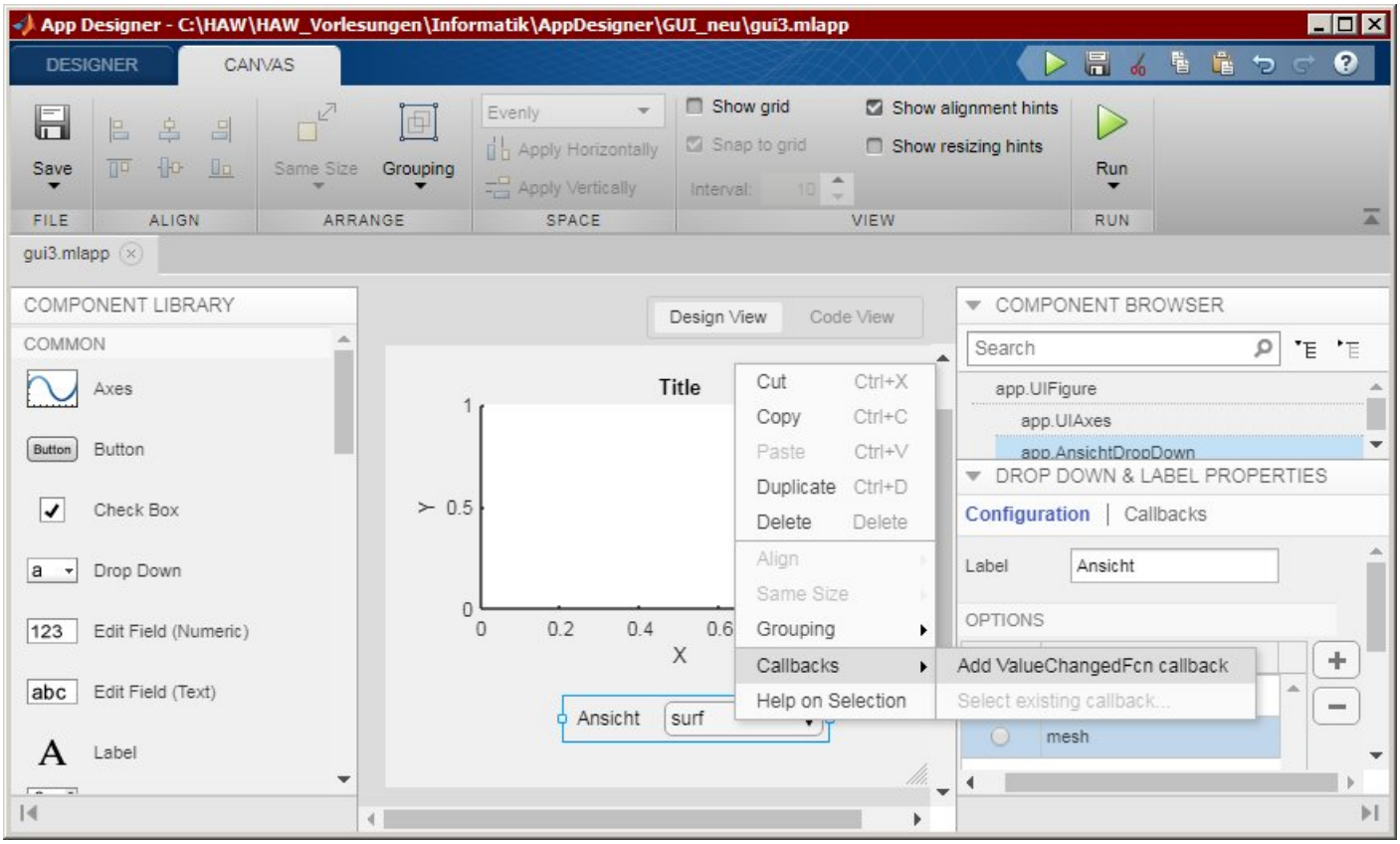


Rechts unten bei den „DROP DOWN & LABEL PROPERTIES“ spezifizieren Sie unter dem Eintrag OPTIONS die Texte, die das Drop-Down-Menü anzeigen soll, in unserem Fall „surf“ und „mesh“. Überzählige Einträge, die automatisch angelegt wurden, löschen Sie mit dem Button „-“, rechts neben der Spalte „Items“.

Wenn Sie die App aufrufen, können Sie jetzt zwischen den Einträgen „surf“ und „mesh“ wählen:



Aber hinter der Auswahl steht noch keine Funktionalität. Dazu benötigen wir wieder eine Callback Function, die auf die Auswahl reagiert. Also rechte Maustaste mit Auswahl „Callbacks > Add ValueChangedFcn callback“:



Im Code View erschient nun die Methode *AnsichtDropDownValueChanged*, die uns Informationen über den ausgewählten Wert des Drop-Down-Menüs gibt:

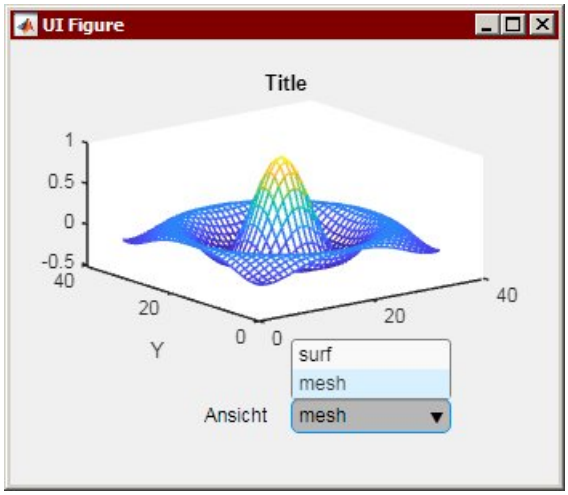
```
% Value changed function: AnsichtDropDown
function AnsichtDropDownValueChanged(app, event)
    value = app.AnsichtDropDown.Value;
end
```

Als *value* wird einer der beiden Texte „surf“ oder „mesh“ zurückgegeben, je nach Auswahl.

Erweitern Sie die Methode *AnsichtDropDownValueChanged*, um zwischen den Darstellungen *surf* und *mesh* umzuschalten:

```
% Value changed function: AnsichtDropDown
function AnsichtDropDownValueChanged(app, event)
  value = app.AnsichtDropDown.Value;
  if( strcmp( value, 'surf' ) )
    surf( app.UIAxes, app.handles.Z );
  else
    mesh( app.UIAxes, app.handles.Z );
  end
end
```

Rufen Sie die App auf und testen Sie die beiden Darstellungsoptionen:



Erweiterungen

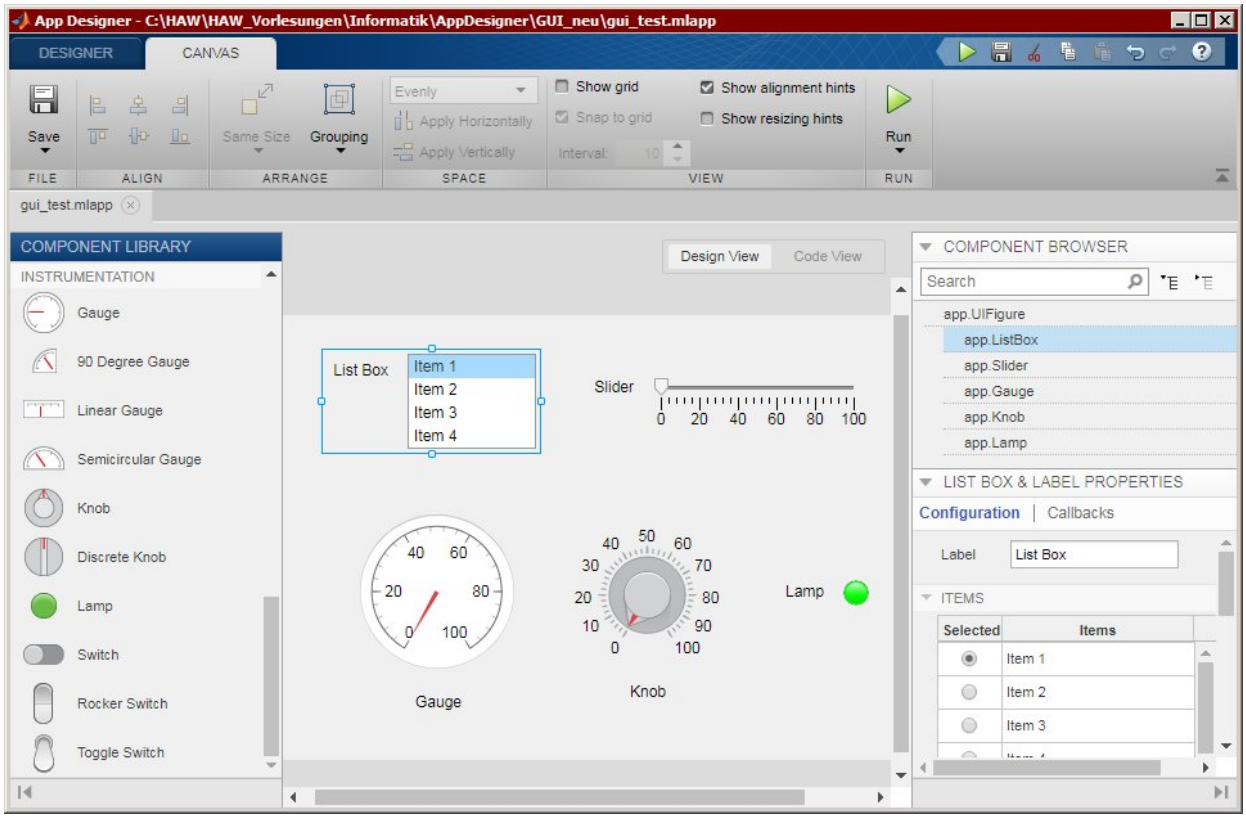
Bauen Sie in Ihre App noch weitere Controls ein und testen Sie deren Verhalten bei einer Anwahl. Die erzeugten Callback-Funktionen im Code View erhalten bereits kurze Hilfen, wie man die Daten der Controls auslesen bzw. setzen kann, z.B. bei einer *Listbox* steht:

```
% Value changed function: ListBox
function ListBoxValueChanged(app, event)
  value = app.ListBox.Value;
end
```

Hinweis: Die Beschriftung der Achsen können Sie ausschalten über den Befehl

```
set( gca, 'XTick', [] )
set( gca, 'YTick', [] )
```

Versuchen Sie sich auch einmal mit den Controls aus der Sektion INSTRUMENTATION, z.B. mit einer *Gauge*, einem *Knob* oder einer *Lamp*.



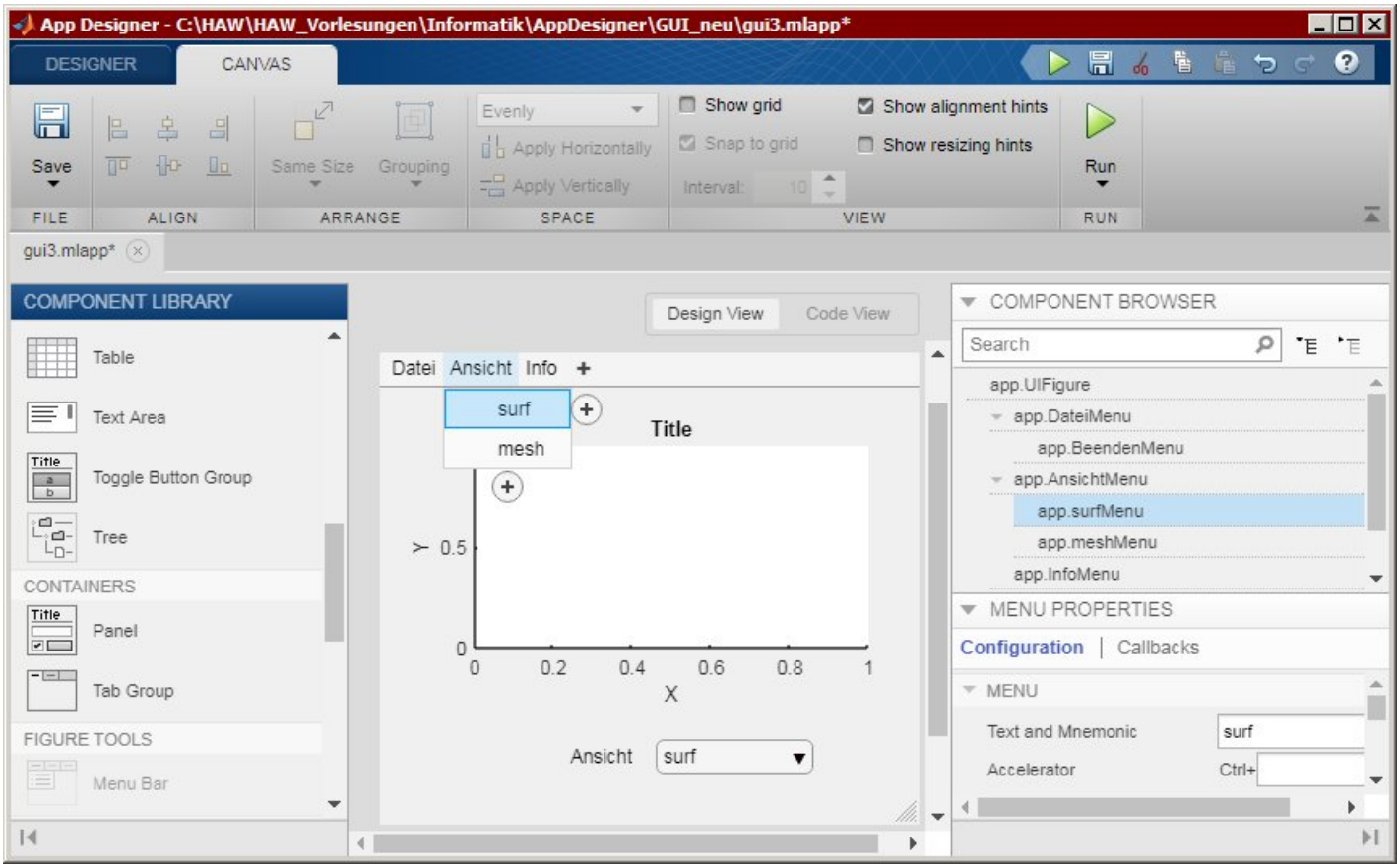
Aufgabe gui3: Menü-Erweiterung

Menu Bar

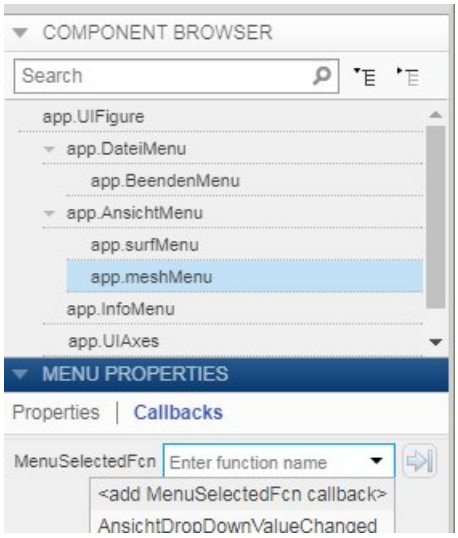
Wir möchten nun am oberen Ende der App ein Pull-down-Menü als Menu Bar anbringen.

Öffnen Sie dazu wieder unsere *gui3* und ziehen Sie aus der COMPONENT LIBRARY auf der linken Seite einen „Menu Bar“ aus den „FIGURE TOOLS“ in das Layout an das obere Ende.

Ändern Sie dann auf der rechten Seite unter „MENU PROPERTIES“ die Texte für die einzelnen Menü-Einträge im Feld „Text and Mnemonics“ in „Datei“, „Ansicht“ und „Info“. Fügen Sie im Layout mit Hilfe der ‚+‘-Buttons für „Info“ einen weiteren Menü-Eintrag hinzu. Unterhalb Des Menüs „Datei“ legen Sie mit ‚+‘ das weitere Item „Beenden“ an, unterhalb von „Ansicht“ die beiden Items „surf“ und „mesh“:



Jetzt müssen wir noch die Callbacks zu den Menü-Einträgen erzeugen. Dazu gehen Sie entweder ins Layout und klicken mit der rechten Maustaste auf einen Menü-Eintrag. Sie können aber auch auf der rechten Seite unter „MENU PROPERTIES“ in den Tab Callbacks umschalten und die Option wählen: <add MenuSelectedFcn callback>.



Im Code View erscheinen dann die noch leeren Callback-Funktionen für diese Menü-Einträge, z.B.

```
function surfMenuSelected(app, event)
```

etc.

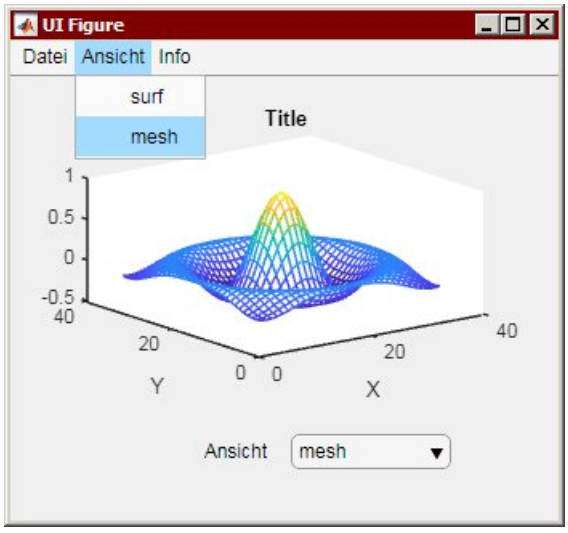
Hier tragen wir die gewünschten Aktionen ein:

```
% Menu selected function: surfMenu
function surfMenuSelected(app, event)
  surf( app.UIAxes, app.handles.Z );
end
% Menu selected function: meshMenu
function meshMenuSelected(app, event)
  mesh( app.UIAxes, app.handles.Z );
end
% Menu selected function: BeendenMenu
function BeendenMenuSelected(app, event)
  delete( app.UIFigure );
end
% Menu selected function: InfoMenu
function InfoMenuSelected(app, event)
  msgbox( {'Grafik-Beispiel','zum App Designer in MATLAB'},'Info' );
end
```

Die Aktionen für „surf“ und „mesh“ kennen wir bereits. Zum Beenden der Ap dient der Aufruf `delete( app.UIFigure );`.

Etwas unschön ist es noch, dass nach dem Umschalten der Ansicht über das Menü die gewählte Ansicht nicht auch im Drop-Down-Menü angezeigt wird. Dies beheben wir durch ein explizites Setzen von *app.AnsichtDropDown.Value*:

```
% Menu selected function: surfMenu
function surfMenuSelected(app, event)
  surf( app.UIAxes, app.handles.Z );
  app.AnsichtDropDown.Value = 'surf';
end
% Menu selected function: meshMenu
function meshMenuSelected(app, event)
  mesh( app.UIAxes, app.handles.Z );
  app.AnsichtDropDown.Value = 'mesh';
end
```



Context-Menüs scheint es aktuell (Release 2018a) im App Designer noch nicht zu geben.