Informatik mit Matlab – Labor 6

Aufgabe gui1:

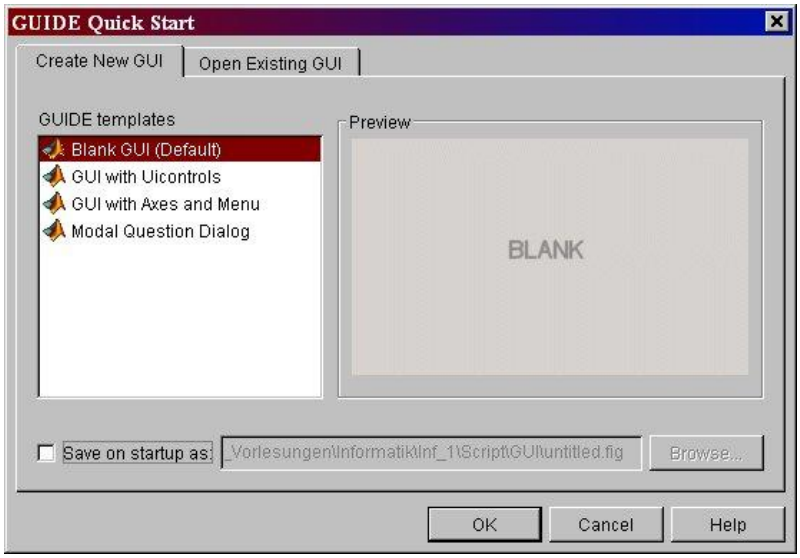
Erstellen Sie die die GUI-Fenster, wie sie in der Vorlesung vorgestellt wurden. Modifizieren Sie die Fenster, indem Sie andere GUI-Objekte einbauen.

Vorgehen:

MATLAB Command Window:
>> guide

Startdialog von GUIDE:

Create New GUI
+ Blank GUI
und **OK**.

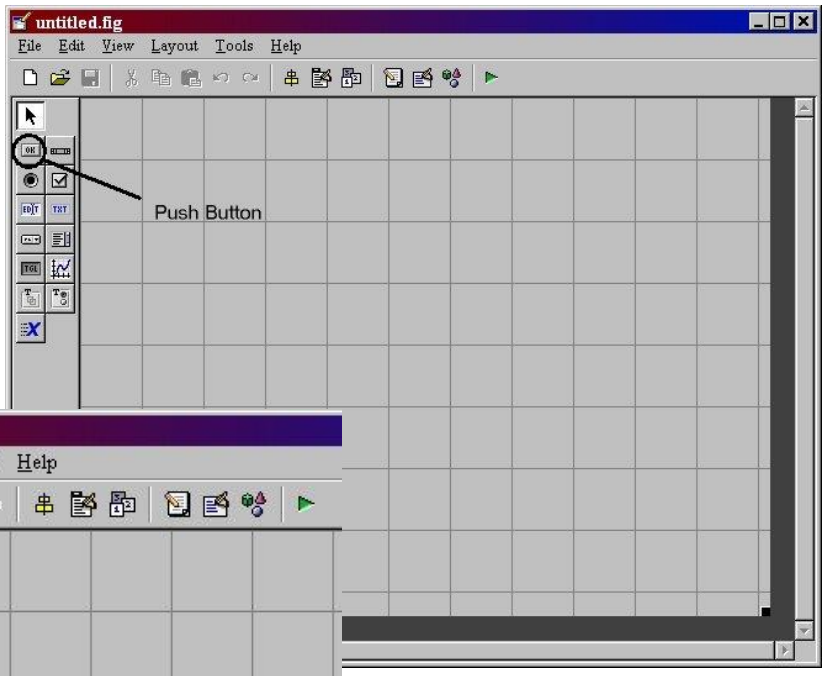


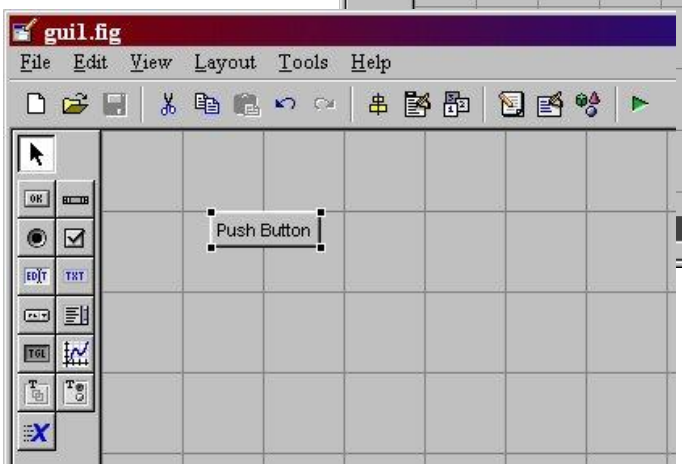
Es öffnet sich der
GUIDE Layout-Editor:

GUI-Element: **Push Button**:

Maus-Klick in der Iconleiste (links auf das oberste Feld der linken Reihe Aufschrift "OK").

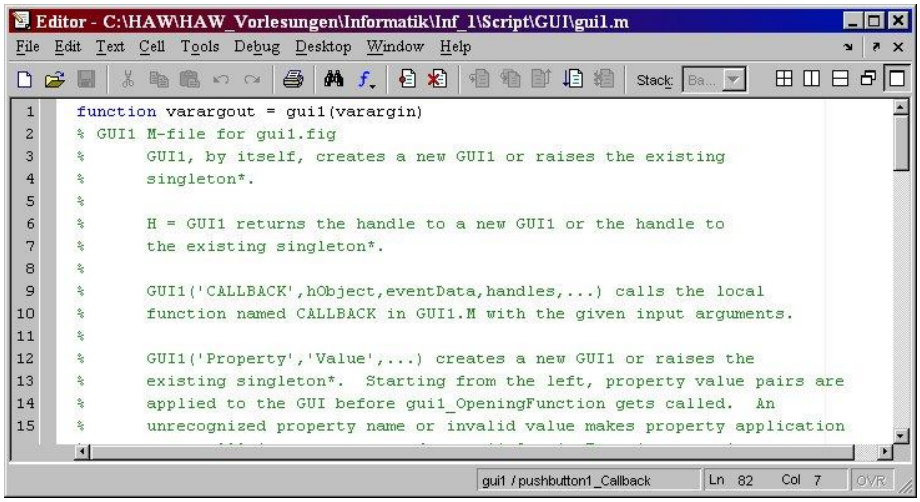
Element mit gedrückter linker Maustaste in den Layout-Bereich ziehen.





Abspeichern: Namen ***gui1.fig.***

M-File gui1.m:



Am Ende der Datei: Kopf der Funktion *pushbutton1_Callback* - **Callback-Funktion**:

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

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

      figure;
      plot( [2,5], [1,3] );
      axis( [0,8,0,5] );
```

Zur Erinnerung: Die **Köpfe** der Callback-Funktionen (hier: function pushbutton1_Callback) legt GUIDE 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*.

Speichern Sie den M-File und schließen Sie alle GUIDE-Fenster.

Aufruf im MATLAB **Command Window**:

```
>> gui1
```

Test der Funktionalität:
Push Button klicken -> Aufruf der Callback-Funktion *pushbutton1_Callback*

Vorsicht:
Wenn Sie Controls (GUI-Elemente) aus einem Layout ***löschen***, kann es sein, dass die (nun unverknüpften) Callbacks weiterhin im M-File erhalten bleiben. Ein danach eingefügter Push-Button erhält eine ***fortlaufende Nummerierung***, also an Stelle von pushbutton1_Callback müssen Sie in diesem Fall pushbutton2_Callback verwenden.

GUI-Rückgabewert abfragen bzw. ändern:

Im GUI-M-File in **gui1_OutputFcn** die Variable ***varargout*** abfragen:

```
varargout{1} = handles.output;
```

Aufruf im **Command Window**: *gui1* mit Rückgabewert:

```
>> h = gui1
h = Figure (figure1) with properties:
      Number: []
        Name: 'gui1'
       Color: [0.9400 0.9400 0.9400]
    Position: [135.8000 66.5385 63.2000 17.8462]
       Units: 'characters'
  Show all properties
```

Rückgabewert in der gui1_OutputFcn **ändern**

```
varargout{1} = 'Hello GUI';
```

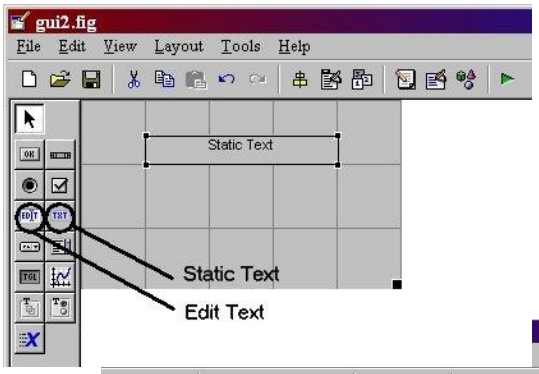
Aufruf im **Command Window**:

```
>> h = gui1;
h = Hello GUI
```

Aufgabe gui2:

Text-Ausgabefeld

GUIDE + leeres Layout,
"**Static Text**"-Element in Layout ziehen
+ Abspeichern unter dem Namen ***gui2***



Im GUI-M-File gibt es keine Callback-Funktion zu Ausgabefeldern.

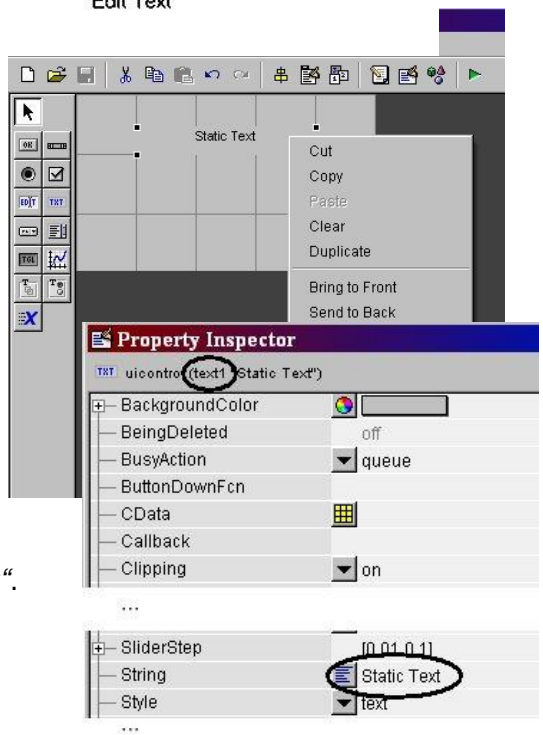
Aufruf: `>>gui2`

Ausgabe-Text **ändern**, zwei Möglichkeiten:

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

zu 1.: Aufruf des "**Property Inspector**"

Im Layout-Editor die Maus auf eingebautes
"Static Text"-Feld + rechte Maustaste.
–> Kontext-Menü, u.a. mit Eintrag "Property Inspector"
–> Dialog mit den Eigenschaften
des "Static Text"-Feldes:



–> Name des Textfeldes ***'text1'*** oder ***'text2'*** im Feld „*Tag*“.

–> Eigenschaft ***'String'***:
rechts daneben "Static Text" ändern in
"*Hello Text-GUI*".

Abspeichern + Aufruf: `>> gui2`

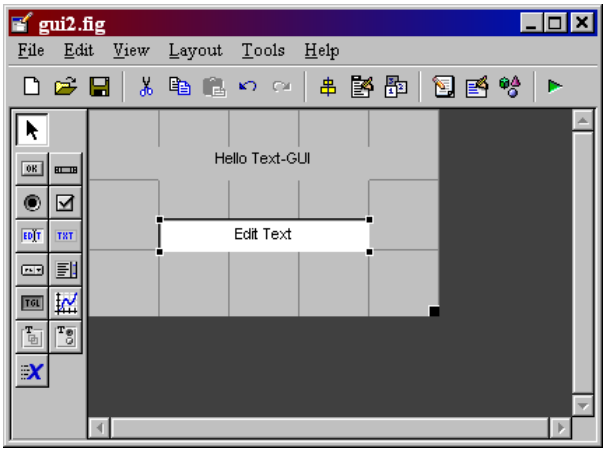
zu 2: im GUI-M-File in der **OpeningFcn**:

```
% --- Executes just before gui2 is made visible.
function gui2_OpeningFcn(hObject, eventdata, handles, varargin)
  % Choose default command line output for gui2
  handles.output = hObject;

  % NEU: im Textfeld 'text1' die Eigenschaft 'String' neu setzen
  %  in MATLAB 2015 heißt das erste Text-Feld 'text2' ???
  set( handles.text1, 'String', 'Hello OpeningFnc' );
  % Update handles structure
  guidata(hObject, handles);
  % UIWAIT makes gui2 wait for user response (see UIRESUME)
  % uiwait(handles.figure1);
```

Abspeichern + **Aufruf**: `>> gui2`

Text-Eingabefeld



GUIDE + "Open Existing GUI" –>gui2.fig

unterhalb des "Static Text"-Feldes
ein "**Edit Text**"-Feld:

"Property Inspector"
–> Namen im Feld *Tag*: ***'edit1'***

Abspeichern + M-File gui2.m öffnen:

In Callback-Funktion ***edit1_Callback***
den eingegebenen Text auslesen (***get***):

```
txt = get( handles.edit1,'String' );
```

bzw.

```
txt = get( hObject,'String' );
```

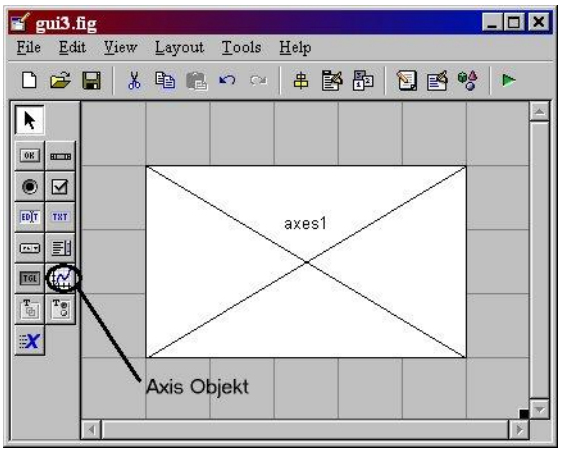
Text (in der Variablen *txt*) mit ***set*** in das Static-Text-Feld (***text1***) schreiben:

```
function edit1_Callback(hObject, eventdata, handles)
  txt = get( hObject,'String' );
  set( handles.text1, 'String', txt );  % in MATLAB 2015 ist Tag: 'text2'
```

Abspeichern + Aufruf: *>> gui2*

Test mit Eingabe in Edit-Feld + Eingabetaste

Aufgabe gui3:



GUI-Grafikobjekt

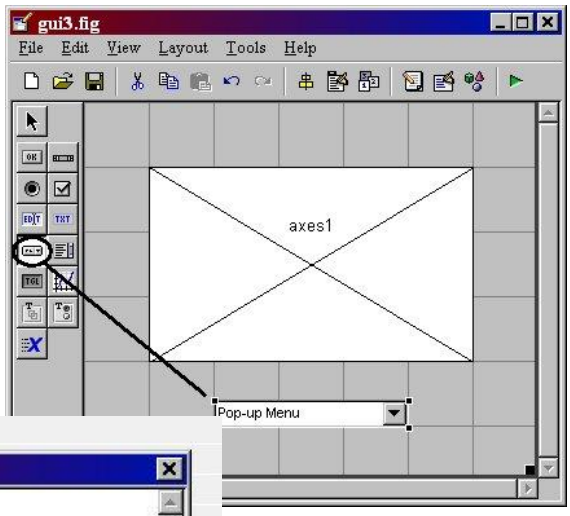
GUIDE + leeres Layout.

Grafikobjekt ***Axes***

Abspeichern *gui3* + Aufruf: `>> gui3`

3D-Grafik:
surf-Daten erzeugen in der **OpeningFcn**.
als Fläche "sin(R) / R":

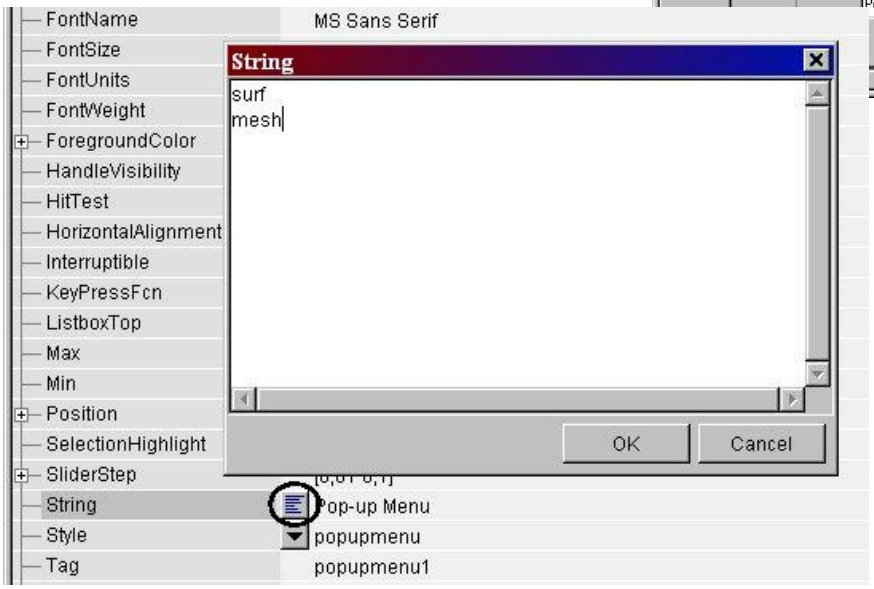
```
function gui3_OpeningFcn(hObject, eventdata, handles, varargin)
  % Grafik in aktuelle Axes zeichnen
  [X,Y] = meshgrid(-8:.5:8);
  R = sqrt(X.^2 + Y.^2) + eps;
  handles.Z = sin(R)./R;
  surf(handles.Z)
  % Choose default command line output for gui3
  handles.output = hObject;
  % Update handles structure
  guidata(hObject, handles);
```



neues GUI-Objekt - **Pop-up-Menü**,
unterhalb der Axes platzieren.

Auswahl-Möglichkeiten im Pop-up-Menü
–> Property Inspector + "**String**" :

–> Eintrag der Texte "**surf**" und "**mesh**":



Callback-Funktion zum Pop-up-Menü *popupmenu1_Callback*:

```
function popupmenu1_Callback(hObject, eventdata, handles)
  % Nummer des ausgewählten Menü-Eintrags
  val = get( hObject, 'Value' );
  % Liste der Texte im Pop-up-Menü
  str = get( hObject, 'String' );

  % Text zu ausgewähltem Menü-Eintrag
  switch val
      % Eintrag surf wurde ausgewählt -> Darstellung surf
      case 1
         surf(handles.Z);
      % Eintrag mesh wurde ausgewählt -> Darstellung mesh
      case 2
         mesh(handles.Z);
  end
```

Abspeichern + Aufruf: >> *gui3*

Test:
im Pop-up-Menü zwischen den Darstellungen surf und mesh umschalten

Erweiterungen

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

```
% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
%        contents = get(hObject,'Value') returns selected item from listbox1
```

`Mit set( hObject, 'String', var )` können Sie Werte in einer Listebox setzen, wobei var eine Cell-Array-Variable ist, die spaltenweise die EInträge der Listbox enhält, z.B.

```
var(1,1) = { 'Eintrag 1' };
var(1,2) = { 'String 2' };
var(1,3) = { 'String 3' };
set( handles.listbox1, 'String', var );  % oder in der OpeningFnc setzen
```

Mit `get( hObject, 'String' )` erhalten Sie alle Einträge der Listbox, mit `get( hObject, 'Value' )` die Nummer, die mit der Maus in der Listbox angewählt wurde.

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

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

bzw. den analogen Eintrag im Property Inspector.

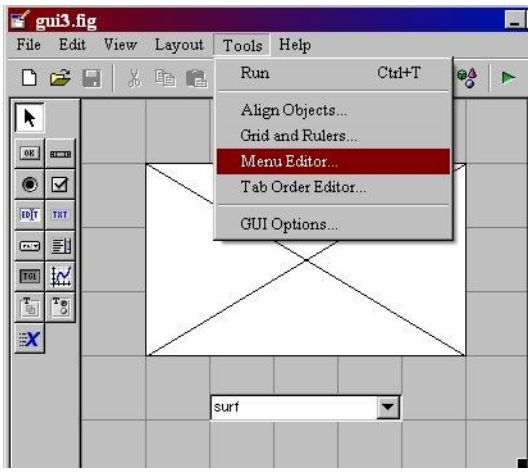
Aufgabe gui3: Menü-Erweiterung

Erstellen Sie GUI-Menüs, wie unten und in der Vorlesung vorgestellt. Modifizieren Sie die Fenster, indem Sie andere GUI-Objekte einbauen.

GUI-Menüs

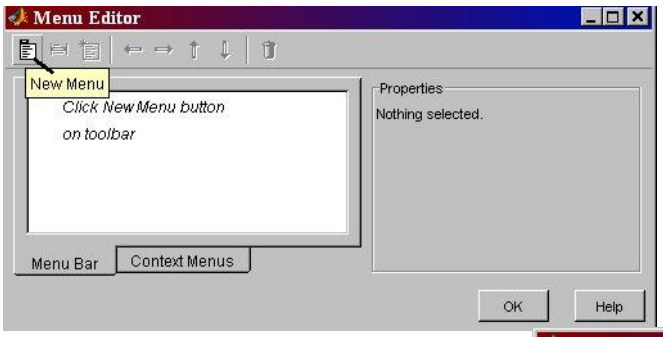
1. **Menu Bars**, als Drop-down Menüs am oberen Ende des GUI-Fensters
2. **Context Menüs**, mit einem Klick der rechten Maustaste.

zu 1.: GUIDE + öffnen von ***gui3*** aus Labor 1.

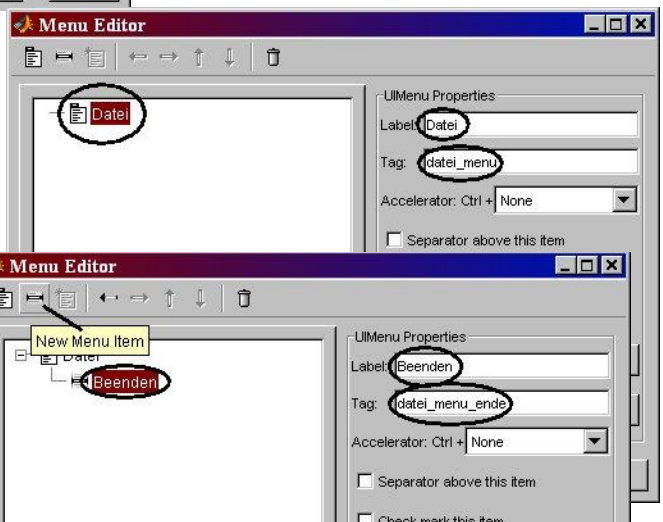


Drop-down-Menü "**Tools**" + "**Menu Editor**":

Tab "**Menu Bar**" + Icon "**New Menu**":



Eintrag "***Untitled 1***" ändern
–> Eigenschaft "**Label**" in "***Datei***" und
–> Eigenschaft "**Tag**" in "***datei_menu***":



+ OK

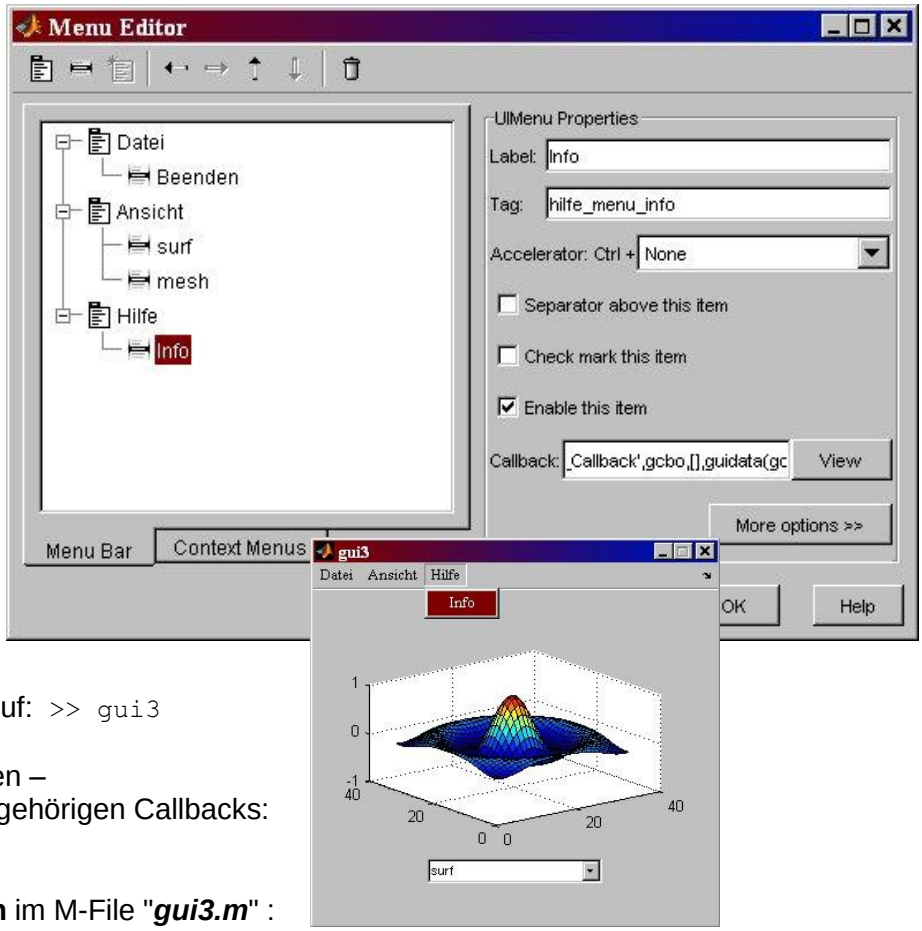
Untereintrag zum Eintrag "Datei"
–> Icon "**New Menu Item**"
(rechts neben Icon "New Menu").

Ändern Eigenschaft
–> "Label" in "***Beenden***" und
–> "Tag" in "***datei_menu_ende***":

–> Weiterer Eintrag "***Ansicht***"
über "**New Menu**" zum
"Tag" "ansicht_menu",
mit den "Menu Items" "***surf***" und "***mesh***"

–> und Eintrag "***Hilfe***" zum "Tag" "hilfe_menu"
mit dem "Menu Item" "***Info***".
Wählen Sie die Tags zu den Items entsprechend:

Vollständiges Menü:



Abspeichern und Aufruf: `>> gui3`

noch keinerlei Aktionen –
es fehlen noch die zugehörigen Callbacks:

Callback-Funktionen im M-File "***gui3.m***" :

Aktionen für Items "***surf***" und "***mesh***":
Umschalten der Darstellung über surf- bzw. mesh-Aufruf.
Zusätzlich Wert in "*popupmenu1*" auf aktuellen Wert "*Value*" umschalten:
1 für "surf" und 2 für "mesh":

```
function ansicht_menu_surf_Callback(hObject, eventdata, handles)
  surf(handles.Z);
  % Wert ebenfalls im Pop-up Menu ändern
  set( handles.popupmenu1, 'Value', 1 );

function ansicht_menu_mesh_Callback(hObject, eventdata, handles)
  mesh(handles.Z);
  % Wert ebenfalls im Pop-up Menu ändern
  set( handles.popupmenu1, 'Value', 2 );
```

Abspeichern und **Aufruf**: `>> gui3`

Menü-Item "***Beenden***": Aufruf "*delete*" für das gesamte GUI-Fenster:

```
function datei_menu_ende_Callback(hObject, eventdata, handles)
  delete(handles.figure1);
```

Menü-Item "***Info***" soll weiteres GUI-Fenster öffnen, Informationen über das Programm. Versuchen Sie sich selbst einmal daran.

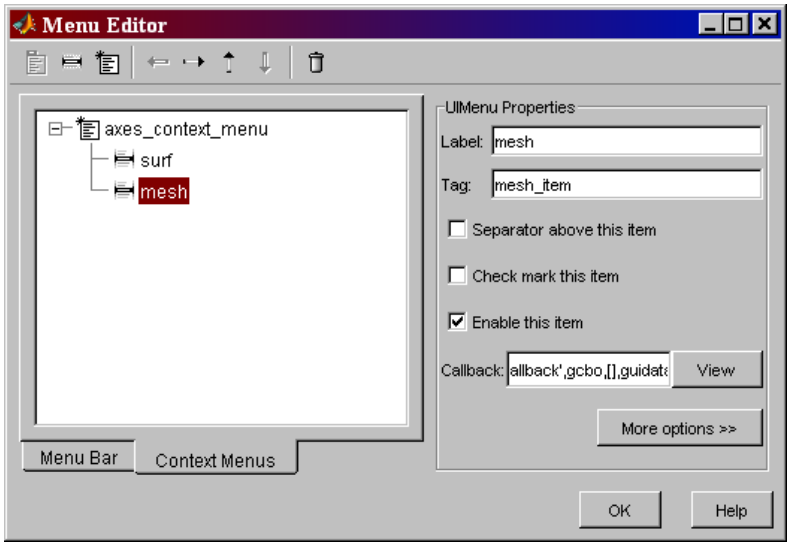
Zu 2.: "**Context Menü**".

Axes-Bereich der 3D-Grafik
mit solch einem Menü
mit Klick der rechten Maustaste aktivierbar

"Menu Editor" in GUIDE + "Tools".
Tab "**Context Menus**"
auf Icon "**New Context Menu**":

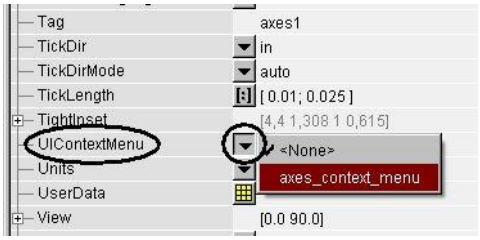
–> Eigenschaften
Tag "***axes_context_menu***"
+ zwei Items "**surf**" und
"**mesh**" mit Tags

"Menu Editor" schließen



Definition, zu welchem **GUI-Objekt** das Context-Menü gehört.
Selektieren Sie im "Layout Editor"
von GUIDE das Object "***axes1***"
und Aufruf "**Property Inspector**":

–> Eintrag "***UIContextMenu***" und
Klick auf das Icon rechts daneben,
Auswahl "***axes_context_menu***"



Callbacks zum Context-Menü im M-File "*gui3.m*" :

```
function surf_item_Callback(hObject, eventdata, handles)
  surf(handles.Z);
  % Wert ebenfalls im Pop-up Menu ändern
  set( handles.popupmenu1, 'Value', 1 );

function mesh_item_Callback(hObject, eventdata, handles)
  mesh(handles.Z);
  % Wert ebenfalls im Pop-up Menu ändern
  set( handles.popupmenu1, 'Value', 2 );
```

Hinweis: Zur Callback-Funktion eines Menüeintrags kommen Sie auch direkt, wenn Sie im Menu Editor auf den Button View, rechts unten neben dem Callback-Eingabefeld, klicken.

Aufgabe Standard-Dialoge

Experimentieren Sie mit den in MATLAB vorhandenen Standard-Dialogen:

- msgbox Erzeugt die Ausgabe einer einfachen Meldung
- warndlg Erzeugt die Ausgabe einer Warnungs-Meldung
- errordlg Erzeugt die Ausgabe einer Fehler-Meldung
- helpdlg Erzeugt die Ausgabe eines Hilfe-Textes
- questdlg Erzeugt einen "Ja-Nein-Abbruch" - Abfrage-Dialog
- inputdlg Erzeugt einen Dialog zur Eingabe eines Textes
- uiputfile Startet einen Dialog zur Auswahl einer Datei zum Speichern
- uigetfile Startet einen Dialog zur Auswahl einer Datei zum Lesen
- uigetdir Startet einen Dialog zur Auswahl eines Verzeichnisses (mit Pfad)
- printdlg Startet einen Druck-Dialog für die aktuelle figure
- uisetcolor Startet den Farb-Dialog zur Auswahl einer RGB-Farbe

Das Aussehen dieser Dialoge kann über Parameter verändert werden, z.B.:

```
>> msgbox( 'Das Ergebnis ist 42.' );
>> errordlg( 'Alles im Eimer !!!');
>> a = questdlg('Schluss jetzt?','Frage','Ja','Nein','Ja')
a = Ja
```

Rückgabe: Name des Buttons, der als Antwort gedrückt wurde, hier 'Ja'.

```
>> n = inputdlg( 'Name: ')
n = 'Willy'          % Rückgabe: Text im Eingabefeld, hier 'Willy'.
```

Auswahl einer Datei zum Abspeichern ***uiputfile***:

```
>> f = uiputfile
f = gui3.m
```

uisetcolor Farb-Dialog zur Auswahl einer RGB-Farbe:

```
>> uisetcolor
ans =  0    1.0000    0.5020
```

Schreiben Sie die Funktion **writeText**(), die folgende Aktionen durchführt:
Auswahl einer Datei zum Abspeichern mittels ***uiputfile***:

```
f = uiputfile;
```

Öffnen Sie die Datei zum Schreiben und schreiben Sie folgenden Text in die Datei:

Ach, armer Yorick! -
Ich kannte ihn, Horatio,
ein Bursche von unendlichem Humor

Vergessen Sie nicht, die Datei am Ende wieder zu schließen.

Ablauf:

```
fid = fopen( f, 'w' );
...
fprintf( fid, '...' );
...
fclose( fid );
```

Zusatzaufgabe: Maus- und Tastatur-Aktion

Durch Maus-Picks werden die Endpunkte von geraden Linien definiert.

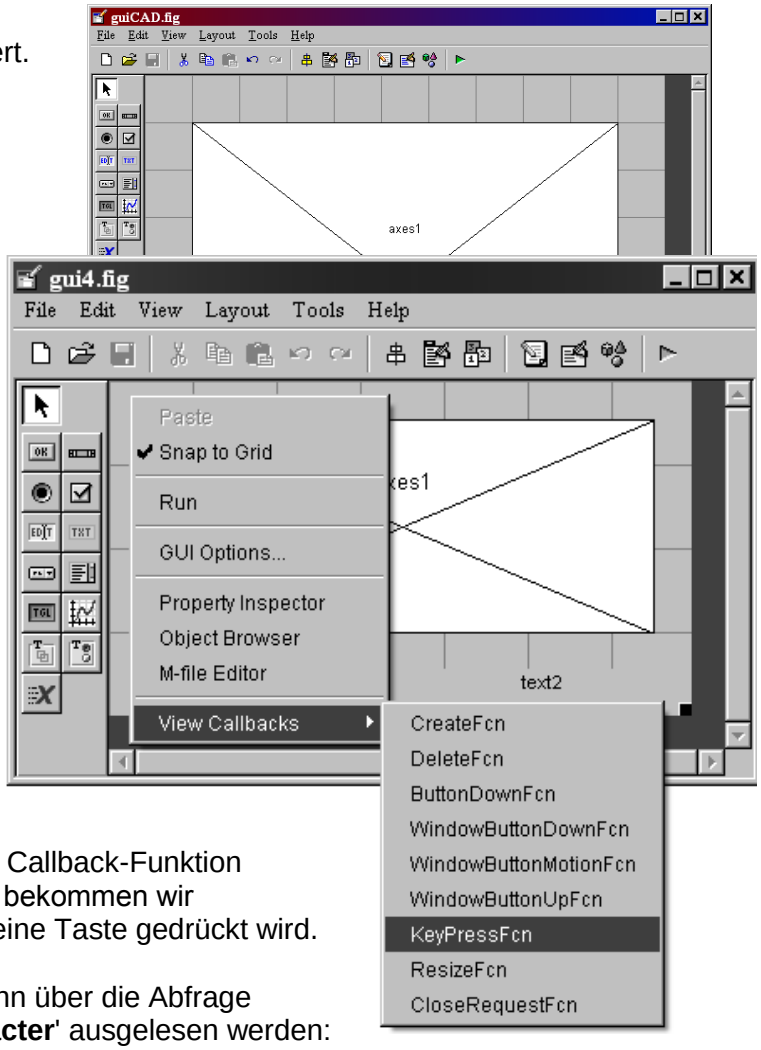
Ein neues GUI-Projekt ***guiCAD0***, mit einem Axes-Objekt *axes1* für die Grafikausgabe und die Maus-Klick-Aktion:

Die Maus-Klick-Aktion soll durch den Druck der Taste ‚g' (Gerade) gestartet werden.

Positionieren Sie die Maus im Layout, ohne das axes-Objekte im Fokus zu haben,

und rufen Sie über die rechte Maustaste das Context-Menü zum gesamten GUI-Fenster auf.

Wählen Sie unter „View Callbacks" den Eintrag „***KeyPressFcn***".



Im M-File guiCAD.m wird dadurch die Callback-Funktion „figure1_KeyPressFcn" erzeugt. Jetzt bekommen wir über einen Callback mitgeteilt, wenn eine Taste gedrückt wird.

Das zur Taste gehörende Zeichen kann über die Abfrage der figure-Eigenschaft '**CurrentCharacter**' ausgelesen werden:

```
% Executes on key press over figure1 with no controls selected.
function figure1_KeyPressFcn(hObject, eventdata, handles)

 % Zeichen zur gedrückten Taste in Variable c speichern
 c = get( hObject, 'CurrentCharacter' );
```

Testen Sie, ob das in der **Variablen c** gespeicherte Zeichen dem Buchstaben **'g'** entspricht.

Ist dies der Fall, starten Sie die Maus-Pick-Aktion über ***ginput***, die die x- und y-Koordinate des gepickten Punktes zurückgibt:

```
% Einen Punkt mit der Maus auswählen und
% die x-Koordinate des Punktes an x1, y-Koordinate an y1 zurückgeben
[x1,y1] = ginput(1);
```

Analog wird der **zweite Endpunkt** der Geraden [x2,y2] mit einem weiteren ginput-Aufruf gepickt: `[x2,y2] = ginput(1);`

Erstellen Sie nun eine Linie zwischen den beiden Punkten durch Aufruf der Funktion **plot**, analog den Grafik-Beispielen.