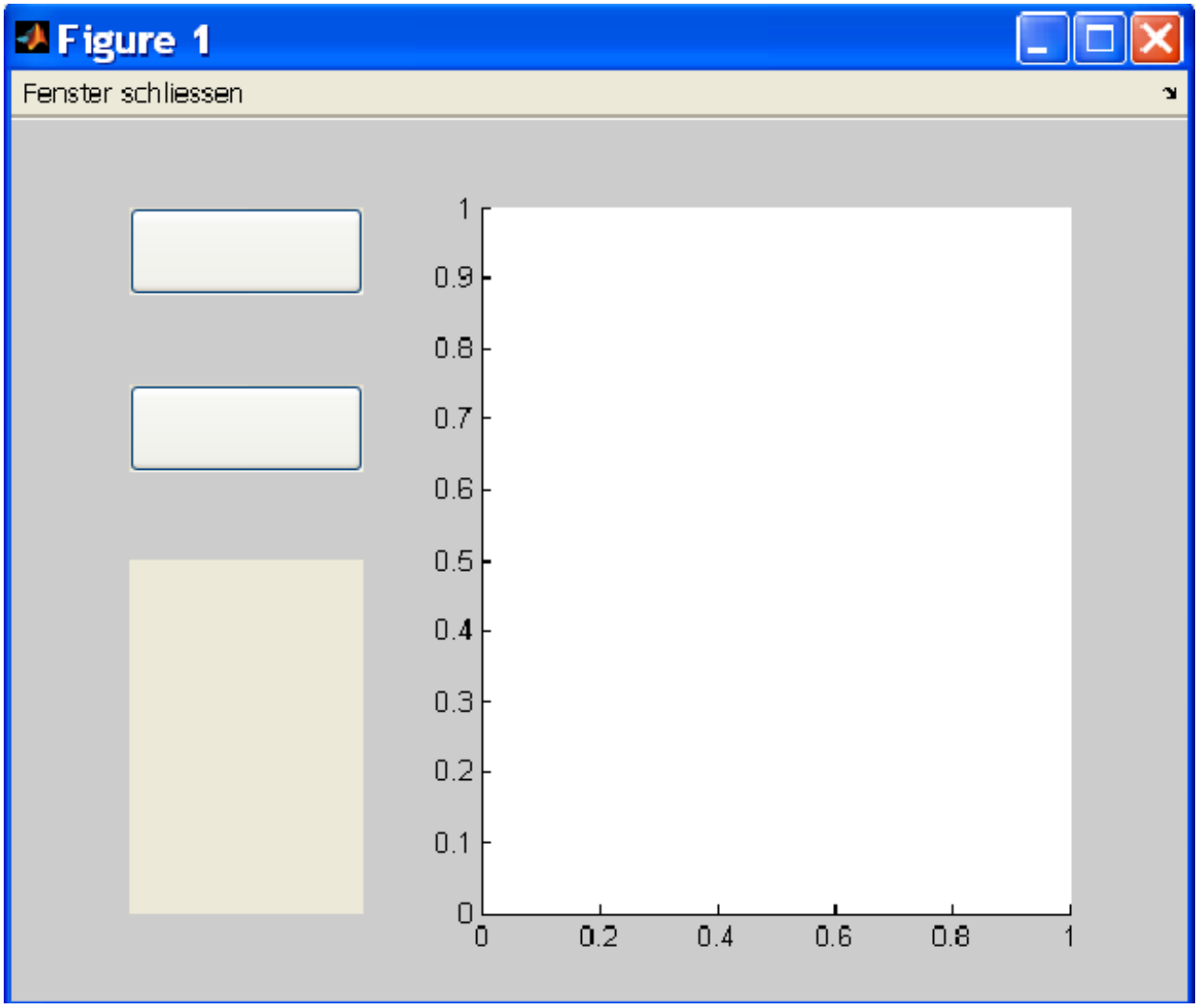
Buttons mit Callback verbinden

Die nachfolgend beschriebenen Matlabanweisungen erzeugen folgende graphische Benutzerschnittstelle mit einer Menüleiste, zwei Pushbuttons, einem Textfeld und einer Zeichenfläche.



Zunächst wird durch

```
figure(1)
set(gcf,'menubar','none','units','normalized')
uimenu('label','&Fenster schliessen','callback','close')
```

ein Fenster mit einer Menüzeile erzeugt. Dem Menü wird dabei mit dem Funktionsaufruf

```
uimenu('label','&Fenster schliessen','callback','close')
```

ein Menüpunkt mit dem Text '*Fenster schliessen*' hinzugefügt und mit der Callbackfunktion *close()* verknüpft, d.h. die Ereignismethode *close()* wird immer dann ausgeführt, wenn das Ereignis ***Auswahl des Menüpunktes*** *'Fenster schliessen*" eintritt.

Die Anweisung

```
set(gca,'position',[.4 .1 .5 .8])
```

fügt der Figur eine leere Zeichenfläche hinzu. Dabei ist *gca* der Zeiger auf das aktuelle Achsensystem mit den relativen Koordinaten (hier: prozentuale Werte) als Attributwerte des Feldes '*position*'. Der Figur werden dann durch die Anweisungen

```
uicontrol(  gcf,...
            'style','pushbutton', ...
            'string','',...
           'units','normalized','position',[.1 .8 .2 .1],...
           'callback', '');

uicontrol(  gcf,...
            'style','pushbutton',...
            'string','',...
           'units','normalized','position',[.1 .6 .2 .1],...
           'callback','');
```

zwei Buttonobjekte hinzugefügt.[1] Beide Buttons tragen noch keine Aufschrift und sind nur mit einem leeren Ereignis verbunden, d.h sie sind noch nicht mit einer Ereignismethode verknüpft. Die Anweisung

```
uicontrol(  gcf,...
            'style','text',...
            'string','',...
            'units','normalized',
            'position',[.1 .1 .2 .4])
```

fügt der Figur mit dem Attributwert 'text' des Feldes style ein nicht editierbares Textfeld hinzu. Das Textfeld ist noch leer.

1 Der Operator `', ...'` verkettet die Sequenzen der Felder und Feldwerte über den Zeilenumbruch zur Parameterliste.

Aufgabe (1)

Die folgende Funktion *wavDateiLesen()* liest die Datei '*ringin.wav*' und übergibt die eingelesenen Werte an den Vektor *sound*.. Die Funktion *wavDateiLesen()* erzeugt dann abhängig von der Länge der wav-Datei einen Vektor zur Beschriftung der x-Achse des Plots. Die Funktion *wavDateiLesen()* gibt den Vektor *sound* in der deklarierten Zeichenfläche der Figur aus. Die Funktion *wavDateiLesen()* ist wie folgt deklariert:

```
function [sound] = wavDateiLesen()

            [sound] = wavread('ringin.wav');
            figure
            [x] = (1 : (length (sound)));
            plot(x, sound)
end
```

Verbinden Sie den ersten Button des GUI mit der Funktion *wavDateiLesen()*. Der Button soll die Aufschrift 'WAV lesen' tragen.

Aufgabe (2)

Die folgende Funktion *wavDateiSpielen()* liest die Datei '*ringin.wav*' und übergibt die eingelesenen Werte an den Vektor *sound* übergeben, *sound* wird danach abgespielt.

```
function bool = wavDateiSpielen()

      [sound] = wavread('ringin.wav');
      wavplay(sound)
      bool = 1
end
```

Verbinden Sie den zweiten Button des GUI mit der Funktion wavDateiLesen(). Der Button soll die Aufschrift 'WAV abspielen' tragen.