Oberfläche mit GridBagSizer

Die Gestaltung einer
Oberfläche mit GridBagSizer
zum Raumplaner

Oberfläche mit GridBagSizer

- Ziel ist, zum Raumplaner eine Oberfläche zu gestalten, die (*weitgehend*) ohne Menue auskommt.
- Da nun sehr viele Elemente auf dem Panel des Grafikfensters dargestellt werden, wird eine Darstellung mit einem *Sizer* hergestellt, der ein automatisiertes einheitliches Layout gewährleistet.
- Wir verwenden einen GridBagSizer.

Oberfläche mit GridBagSizer

- Basis sind
  - das Raumplanerprojekt, in das die Gui vollständig integriert ist.
  - die Datei *Demobeispiel_GridBag-Sizer.py*

Oberfläche mit GridBagSizer

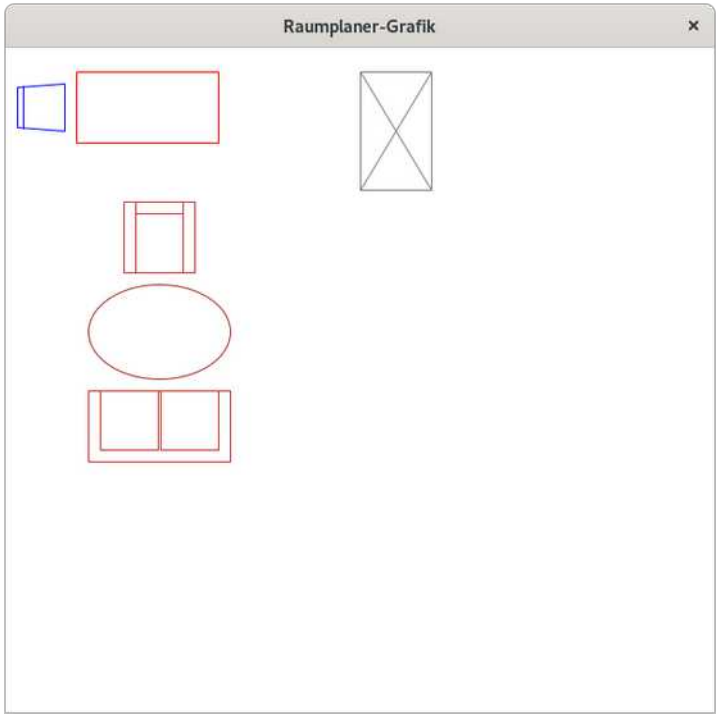
- Auf der Oberfläche sollen zeilenweise jeweils mit Beschriftung (Label -> *StaticText*) und Eingabefeld (*TextCtrl*) dargestellt werden:
  - x-Position
  - y-Position
  - Breite
  - Tiefe
  - Winkel (Orientierung)
  - Farbe

Oberfläche mit GridBagSizer

- Außerdem sollen Buttons zum Auswählen der Moebel-Objekte und zum Beenden vorhanden sein.
- Hilfreich ist auch eine Meldung, welches Objekt gerade ausgewählt ist. Das kann ein StaticText sein, da nichts einzugeben ist, sondern nur ein Wert dargestellt werden soll.

Oberfläche mit GridBagSizer

- Um alles unterzubringen muss die Fenstergröße der Gui geändert werden.



Oberfläche mit GridBagSizer

- Arbeitsschritte in der Klasse Gui
  - Erzeugen des GridBagSizer-Objekts

```
gbs = self.gbs = wx.GridBagSizer(vgap=5, hgap=5)
```

  - Erzeugen der Elemente, beispielsweise:

```
labelX = wx.StaticText(panel, -1, "x-Position")
self.textCtrlX = wx.TextCtrl(panel, -1,"",
                             size=(125, -1),
                             style=wx.TE_PROCESS_ENTER)
```

Oberfläche mit GridBagSizer

- Aufbau in gbs definieren

```
gbs.Add(self.labelAusgewaehlt, (0,0), flag=wx.ALL, border=10)
gbs.Add(self.labelX, (1,0), flag=wx.ALL, border=10)
gbs.Add(self.textCtrlX, (1,1), flag=wx.ALL, border=10)
gbs.Add(labelY, (2,0), flag=wx.ALL, border=10)
gbs.Add(self.textCtrlY, (2,1), flag=wx.ALL, border=10)
...
gbs.Add(self.labelF, (6,0), flag=wx.ALL, border=10)
gbs.Add(self.textCtrlF, (6,1), flag=wx.ALL, border=10)
gbs.Add(self.labelH, (7,1), flag=wx.ALL, border=10)
gbs.Add(waehleButton, (8,0), flag=wx.ALL, border=10)
gbs.Add(beendenButton, (8,1), flag=wx.ALL, border=10)
```

Oberfläche mit GridBagSizer

- gbs dem Panel-Objekt zuweisen und Layout umsetzen lassen.

panel.SetSizerAndFit(gbs)

Oberfläche mit GridBagSizer

- Ereignisbehandlung für das Ereignis EVT_TEXT_ENTER der TextCtrl-Objekte binden:
  (Hier gemeinsam in einer Methode für alle Textfelder)

```
self.Bind(wx.EVT_TEXT_ENTER, self.OnTextEnter, self.textCtrlX)
self.Bind(wx.EVT_TEXT_ENTER, self.OnTextEnter, self.textCtrlY)
self.Bind(wx.EVT_TEXT_ENTER, self.OnTextEnter, self.textCtrlB)
self.Bind(wx.EVT_TEXT_ENTER, self.OnTextEnter, self.textCtrlT)
self.Bind(wx.EVT_TEXT_ENTER, self.OnTextEnter, self.textCtrlW)
self.Bind(wx.EVT_TEXT_ENTER, self.OnTextEnter, self.textCtrlF)
```

Oberfläche mit GridBagSizer

- Ereignisbehandlung realisieren:

```
def OnTextEnter(self, event):
    xT=self.textCtrlX.GetValue()
    yT=self.textCtrlY.GetValue()
    ...
    f=self.textCtrlF.GetValue()   # ist String
    try:
        x=int(xT)
        ...
        w=int(wT)
    except ValueError as e:
        return
    self.__modell.SetzeDaten(x,y,b,t,w,f)
```

Oberfläche mit GridBagSizer

- OnWaehle neu:

```
def OnWaehle(self, event):
    if len(self.__modell.GibAlleMoebel())==0:
        self.labelAusgewaehlt.SetLabel("ausgewählt ist keines")
        return # nichts auszuwählen
    self.__modell.Waehle()
    x,y,b,t,w,f=self.__modell.GibDaten()
    self.textCtrlX.SetValue(str(x))
    ...
    self.textCtrlW.SetValue(str(w))
    self.textCtrlF.SetValue(f)
    =>
```

Oberfläche mit GridBagSizer

- OnWaehle neu:

  =>

```
if self.__modell.GibIndex()==-1:
   self.labelAusgewaehlt.SetLabel("ausgewählt ist keines")
   return # keines ausgewählt
auswahl=self.NameKlasse(self.__modell.GibAusgewaehltes())
auswahl+=' , Nummer '+str(self.__modell.GibIndex()+1)
self.labelAusgewaehlt.SetLabel("ausgewählt ist "+auswahl)
```

- mit der Hilfsfunktion

```
def NameKlasse(self, objekt):
   '''Hilfsfunktion für OnWaehle'''
   return str(objekt.__class__).partition('.')[2].partition("'")[0]
```