nvkurbatova@sfedu.ru

Пакеты научных вычислений

Лекция 7. Создание графического пользовательского интерфейса – GUI!

Курбатова Наталья Викторовна, к.ф.-м.н.,
доцент кафедры математического моделирования, мехмат, ЮФУ

nvkurbatova@sfedu.ru

Содержание:

I. Стратегии создания GUI
II. Родительские объекты элементов управления. Свойства. Модификация
III. Единый конструктор элементов управления
IV. Элементы управления – статические и динамические. Изменение свойств, обработка событий.
V. Оси. Свойства.
VI. Функции, обеспечивающие доступ к файловой системе

nvkurbatova@sfedu.ru

Два подхода создания GUI

I – эволюционный, II – революционный; идеально - I ∪II

I. Все элементы и процедуры создает пользователь	II. Элементы GUI и процедуры создаются средствами редактора guide
плюсы:	плюсы:
• полностью контролируемый процесс • экономим время на этапе отладки и модификации GUI • в процессе разработки GUI достигается системное понимание функционала	• легко создавать элементы в редакторе • легко редактировать элементы, структуру всего окна (ансамблировать: выравнивать элементы и т.п.) • система генерирует guiname.m и исполняемый интерфейс guiname.fig
минусы	минусы
• теряем время на этапе построения *структуры* GUI	• требуется более высокий уровень понимания • сложности возникают на этапе усовершенствования функционала GUI

Элементы управления GUI

figure – родительский объект всех элементов GUI, поле для интерфейса

uicontrol – **единый конструктор** элементов управления (эу); он создаёт следующие

элементы управления: ***edit***ableText, static***text***, ***checkbox***es, ***list***boxes, ***popup***menus, ***sliders***, ***pushbutton***s, ***radiobutton***s (***togglebutton***s);

axes – оси, конструктор осей

uigetdir, uigetfile – функции обеспечивают интерактивный выбор файлов

Root и Figure

nvkurbatova@sfedu.ru

- родительский объект GUI – **figure**!

  определяем дескриптор для figure: **f=figure**

  важно для многооконных GUI

- root (дескриптор - **0**) – родительский объект для figure;

  важно для идентификации размера монитора, корректного дизайна

- размер figure, поле *position*, «привязан» к размеру монитора:

  [x0,y0,hx,hy]=get(0, 'screensize')

- цвет поля figure (фона) по умолчанию наследуется

  потомками, **проверьте в своей версии!**

  colorfigure=get(f, 'color ')

  фон у потомков – свойство поля *BackgroundColor*

- set(gcf, 'MenuBar', 'none') без заголовка figure
- set(gcf,'NumberTitle', 'off') – без номера заголовка figure
- set(gcf,'name', 'My interface') – заголовок пользователя
- set(gcf, 'Resize', 'off') запрет на изменение размера figure

Свойство поля *style* конструктора *uicontrol* определяет назначение (тип) элемента управления

Поле	Значение поля	Назначение элемента управления
style	**edit**	редактируемый текст, ввод информации, динамический элемент (**дэ**)
style	**text**	поясняющий текст (неизменяемый), статический элемент (**сэ**)
style	**checkbox**	динамический элемент, галочка (0 или 1) – разветвление алгоритма
style	**list**	список (**дэ**)
style	**sliders**	масштабирующий движок, слайдер (**дэ**)
style	**pushbutton**	кнопка для запуска алгоритма-процесса (**дэ**)
style	**radiobutton**	переключатель , признак, принимает значения 0 или 1 (**дэ**)
style	**frame**	рамка для визуального выделения группы объектов на поле figure (**сэ**)
style	**popup**	конструктор для мини меню (**дэ**)

Стратегия построения GUI (I-й подход)

1) создаем головной файл как процедуру без параметров, f.e. guiname.m
2) переменные, используемые во внешних процедурах описываем как глобальные: **global var1 var2**;
3) строим окно для всех элементов GUI, в окне **f=figure** либо со свойствами по умолчанию, либо пользователя, например, единиц измерения: **set(f, 'units', 'normalized')**
4) создаём **все** элементы GUI (э. у., оси и т.д.)
5) задаём! (целесообразно) «начальные» значения, предшествующие диалогу
6) программируем процедуры (в отдельных файлах и/или handle_function или подпроцедуры), обеспечивающие запуск процессов
7) программируем ту часть кода, которая основана на обработке событий, обеспечивающих ввод-вывод информации, выбор предлагаемых переходов, содержание проблемного алгоритма,…

Изменение свойств графических объектов

- **set** – установка свойств (полей структуры) графического объекта;
- **set(дескриптор) –** получение возможных свойств полей с указанием по умолчанию
- **get** – вывод (получение) текущих свойств полей структуры графического объекта;
- **reset(дескриптор)** – восстановление свойств графического объекта по умолчанию;
- **delete(дескриптор)** – удаление графического объекта, являющегося аргументом функции delete.

Элемент управления - **text** статический, неизменяемый

nvkurbatova@sfedu.ru

Обязательные свойства: **эут – обозначим элемент управления текст**

text1 = uicontrol(f, ' Style', 'text', 'String', ' надпись ', 'Position', [x0t y0t lxt lyt])

- дескриптор f или gcf) не обязателен, если элемент строится в активном figure;
- первые буквы в имени полей могут быть заглавными и (или) строчными (имена полей регистронезависимы, а в **нотации name.NameField – как?**)
- значения остальных свойств в этом примере для text1 выбираются по умолчанию (все значения полей по умолчанию здесь: set(имя_**эут**) ☺

Дополнительные свойства:

set(text1 , 'fontname', ' tex ', 'fontsize', 14, 'backgroundcolor ', get(f, 'color') , …,
'HorizontalAlignment ', ' left ', ' FontAngle ' , ' italic ')

- свойства задаются при создании или по мере надобности
- get(f, 'color') выбор цвета фона figure

в новых версиях не обязательно

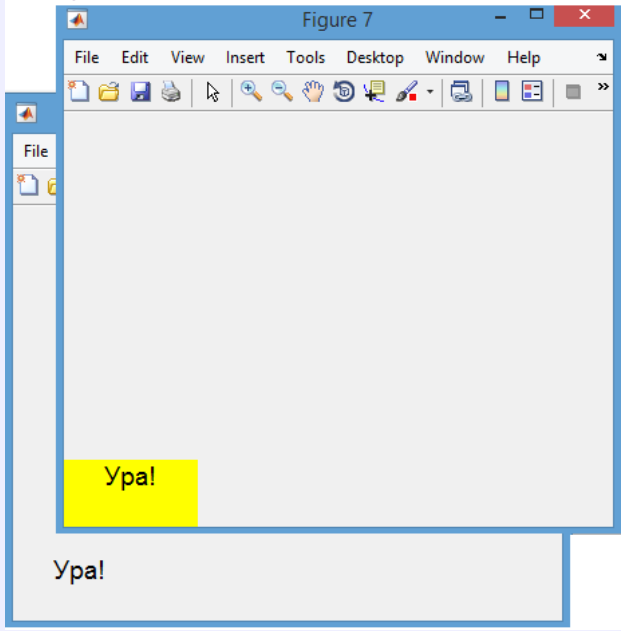
Создание text, пример

nvkurbatova@sfedu.ru

1) function partGui **Пример**
2) f=figure
3) tt=uicontrol('style','text','Position', [1, 1 100 50],...
'String','Ура!','fontsize',14) % **Проверьте независимость от регистра!**

По умолчанию цвет фона text совпадает с фоном figure!

4) set(tt) % так узнаем свойства полей надписи, в т.ч. по умолчанию
% HorizontalAlignment: {'left' 'center' 'right'}
5) tt.HorizontalAlignment % 'center'
6) tt.BackgroundColor % [1 1 0]
7) tt.Units % 'pixels'
8) tt.BackgroundColor='yellow'



Вопросы:

- Какие координаты левого нижнего угла figure?
- Объясните суть параметров поля position!
- Является ли поле figure(7) активным?
- Справедливо ли утверждение, что свойство **resize** текущего поля figure равно **off**?

Элемент управления - **edit** *динамический*

nvkurbatova@sfedu.ru

Обязательные свойства:

ed = uicontrol(f, ***'Style',*** 'edit', ***'String'***, ' x0 ', ***'Position'***, [x0e y0e lxe lye])

поле edit предназначено для ввода данных, чаще пустое,
на этапе отладки **целесообразно** задать значение, f.e.
'String', ' x0 ' или 'string ', num2str(double_value)

Дополнительные свойства (аналогичные text) изменяются по воле программиста!

Обработка события:

получение введенной информации
преобразование к необходимому типу данных

Charvalue=get(**ed**, 'String') информация (класса Char) извлекается из поля String

DoubleValue=str2num(Charvalue) – конвертируется в double

nvkurbatova@sfedu.ru

Элемент управления - **list** (*динамический*)

Обязательные свойства:

lst = uicontrol(f, ***'Style',*** 'list', ***'String***', Cell of String, ***'Position',*** [x0l,y0l,lxl,lyl])

- элемент list – переключатель, позволяющий реализовать несколько предусмотренных стратегий - переходов
- Cell of String – массив ячеек из строк, которые описывают набор «переходов» и отображаются на этом элементе list GUI
- в окне list системно обеспечивается прокрутка (по мере надобности)
- пользователь выбирает конкретный «переход»

Обработка события (многофункциональный переход):

Index= get(**lst**, 'value') % – номер выбранного элемента в списке Cell of String, который определяет «переход»

Весь список возможностей извлекается из поля ***String***:

ListNames=get(**lst**, ***'String'***) % **массив ячеек**

StepIN=ListNames{Index} % выбирается **пользовательский** переход

Элементы управления: **checkbox, radiobutton**
динамические элементы

Обязательные свойства:

checkbox

chbox = uicontrol(*'Style'*, ' checkbox ',*'String'*, ' поясняющий текст',...
'Position', [x0chb y0chb lxchb lychb]);

radiobutton

rb= uicontrol(*'Style'*, ' radiobutton ',*'String'*, ' поясняющий текст ',...
'Position', [x0rb y0rb lxrb lyrb]);

назначение обоих элементов – бинарные переключатели, разветвление алгоритма (наличие или отсутствие конкретной опции)

checkbox и **radiobutton** отличаются по «топологии» опционального выбора

Обработка события :

событие: выбор ***опции*** – генерируется 1, в противном – 0

Система передаёт опцию как свойство поля *value* структур chbox или rb;
извлекаются командой: r=get(rb,*'value'*) или r=get(chbox,*'value'*)

Чему может равняться r?

nvkurbatova@sfedu.ru

Элемент управления: **pushbutton**

динамический элемент

Обязательные свойства:

pb = uicontrol('***Style***', ' pushbutton ','***String***', ' текст на кнопке',...
'***Position***', [x0pb y0pb lxpb lypb], ' ***Callback*** ', **action**);

gcbo (GraphicCurrentButtonObject) **-** дескриптор объекта типа **pushbutton**, чей **Callback** выполняется в текущий момент

Назначение элемента: запуск алгоритма или процесса по нажатию кнопки

action может принимать следующие значения:

- строка, которая содержит последовательность команд, например:
  'cla, load XY, plot(X,Y) '
- имя подпроцедуры (в т.ч. function_handle)
- имя внешней процедуры (в т.ч. для сложного многооконного GUI)

Обработка события :

Выполнение действия, предусмотренного **action**

nvkurbatova@sfedu.ru

Элемент управления: **slider** (динамический)

Обязательные свойства:

sl = uicontrol(***'Style'***, ' slider ', ***'Min'***,vmin,***'Max'***,vmax, …
'SliderStep',[a b], ***'Value'***, position_slider, …
'Position', [x0sl y0sl lxsl lysl], ***'Callback'***, **action**);

Назначение:

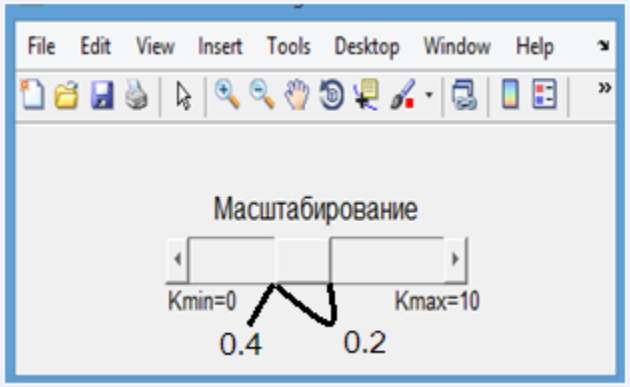
Масштабирование объектов, в т.ч. графических. Коэффициент масштабирования изменяется в пределах [vmin,vmax]

Обработка события :

аналогично **action** в **pushbutton**

Особенности параметров движка

```
function myslider
global sl scaling
sl =     uicontrol('Style', ' slider ', 'Min',0,'Max',10,...
   'SliderStep',[0.4 0.2], 'Value', 0.5, ...
   'Position', [100 100  200  25], 'Callback', 'CHf' );
tsl=uicontrol('Style', 'text','position',[100 125 200 25],...
  'string', ' Масштабирование ','fontsize',12)
tsl1=uicontrol('Style', 'text','position',[85 65 80 35], 'string',...
   'Kmin=0','fontsize',10,'fontname','Latex')
tsl2=uicontrol('Style', 'text','position',[240 65 80 35], 'string',
'Kmax=10','fontsize',10)
end
```




```
function val=CHf
global sl scaling
val=get(sl,'value')
scaling=get(sl,'Min')+(get(sl,'Max')-get(sl,'Min'))*val
end
```

Построение **pop up menu**

nvkurbatova@sfedu.ru

Popup – элемент управления со свойством списка и кнопки

Обязательные свойства:

```
pp = uicontrol('Style', 'popup',...
                'String', {'массив ', ' ячеек ', ' из строк '},    ...
                'Position', [x0pp y0pp  lxpp lypp],
                'Callback', action);
```

Обработка события:

аккумулирует свойства **списка**, т.е. выбирается требуемая ветвь,
и **кнопки** - запускается ***action***:

```
index=get(pp,'value')
cellchar=get(pp,'string')
newval=cellchar{index}
```

Если размера выделенной области для **popup** недостаточно,
то э.у. системно реализуется как выплывающее меню!

Построение осей конструктором **axes**

axes (name), name = gca или name = конкретный дескриптор созданных осей

создание осей конструктором axes:

ha=axes(~~'parent', f,~~ 'Units', 'points', 'Position', [x0ha y0ha lxha lyha], …
'Color', [1 1 1], 'FontSize', 14);

axis([Xmin Xmax Ymin Ymax]) – разберитесь самостоятельно!

маленькие хитрости:

- axes(**ha**) – назначение **ha** активными осями
- cla – очищение активных осей
- set(ha, ' Xtick', [], ' Ytick', []) – отказ от нанесения разметки на осях **(когда важно отказаться?)**

Поиск элементов управления с заданными свойствами:

Obj= findobj(f, 'style', 'list'), например.
Obj= findobj(f, 'tag', 'specific'),
'tag' – ярлык-метка присваивается пользователем любым элементам управления, в т.ч. разнородным, связанным определенной логикой – например, сделать группу **э.у.** невидимыми или применить некоторую операцию к группе **э.у.**

Функции, обеспечивающие доступ к файловой системе

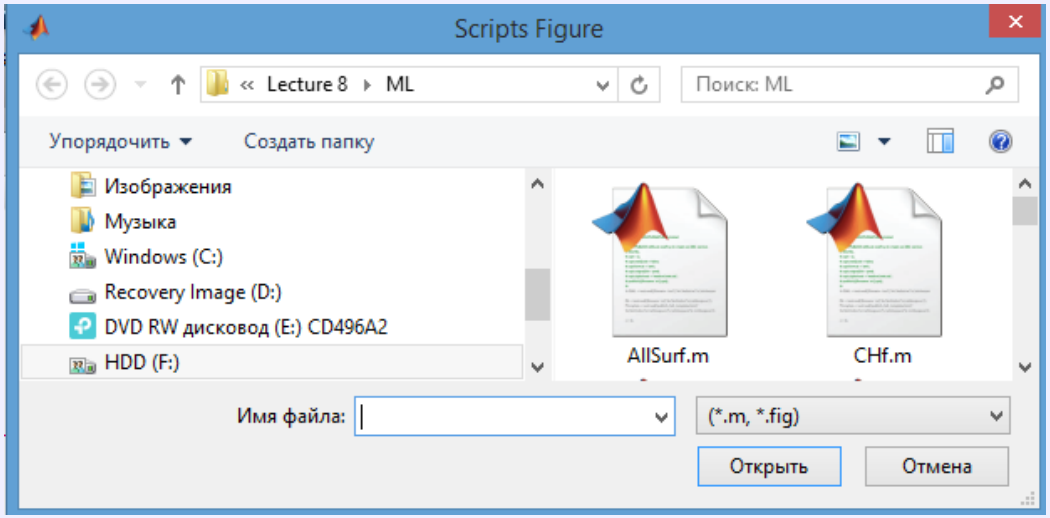
выбор рабочей-текущей директории:
folder_name = uigetdir

Навигация в файловой системе и
выбор папки в диалоговом окне:
folder_name = uigetdir(start_path,dialog_title)

выбор файла в текущей папке:
filename = uigetfile

Поиск в диалоговом окне:
[filename, pathname] =
uigetfile(FilterSpec,DialogTitle, Name)

Специфицированные файлы отображаются в диалоговом окне, например:
[FileName,PathName] = uigetfile({'*.m', '*.fig'}, 'Scripts Figure');



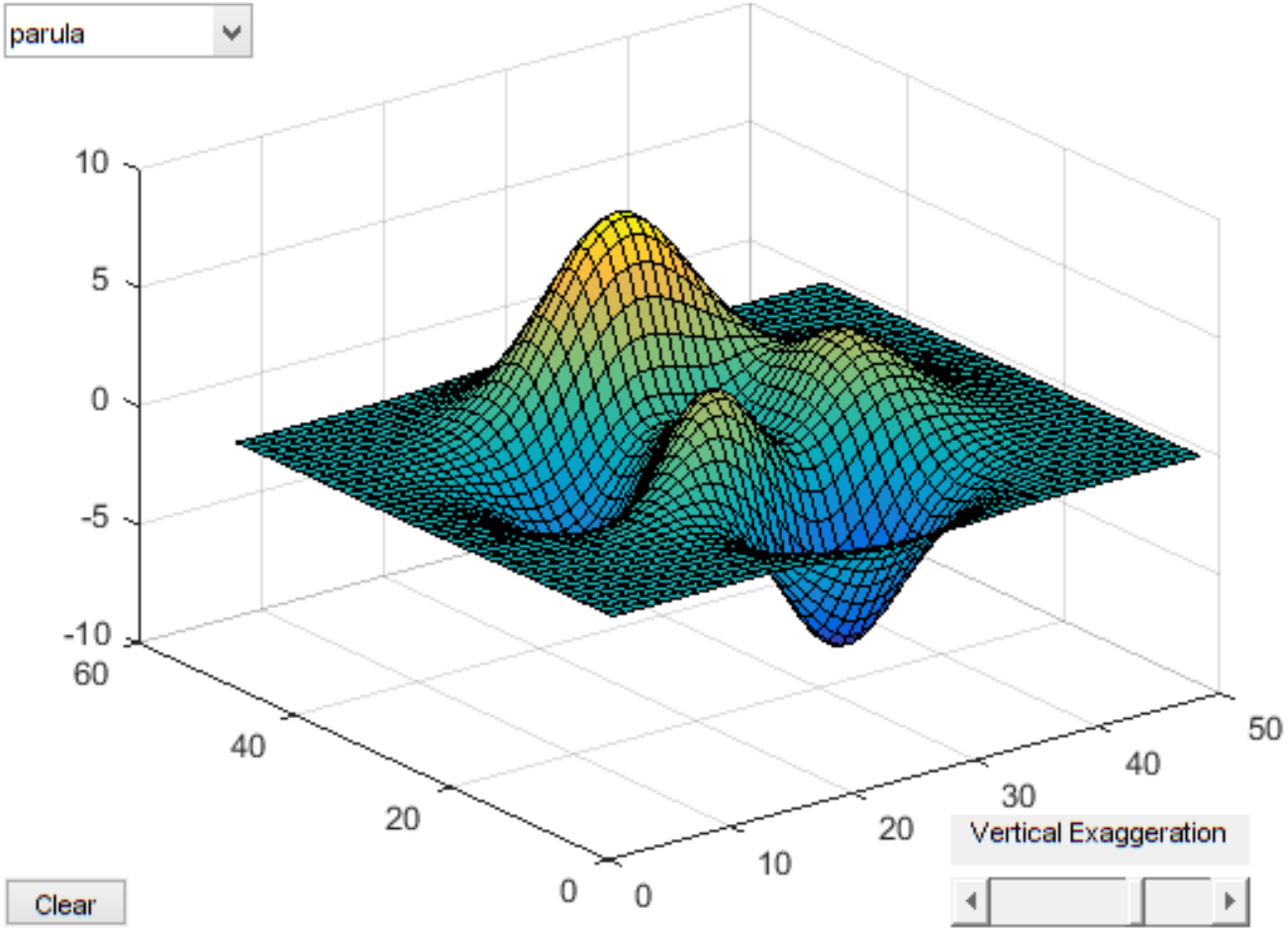
Пример кода (записан в файле netgui.m)

```
function   netgui

f = figure('visible','off')
   set(f,'resize','off') % fix size figure

   ax = axes('Units','pixels');
   surf(peaks)

   % Create pop up menu
   popup = uicontrol('style', 'popup',...
         'String', {'parula','jet','hsv','hot','cool','gray'},...
         'Position', [20 340 100 50],...
         'Callback', @setmap);

  % Create push button for clear axes
   btn = uicontrol('style', 'pushbutton', 'String', 'Clear',...
      'Position', [20 20 50 20], 'Callback', 'cla');

  % Create slider
   sld = uicontrol('Style', 'slider',...
      'Min',1,'Max',45,'Value',41,...
      'Position', [400 20 120 20],...
      'Callback', @surfzlim);

 % Add a text uicontrol to label the slider
   txt = uicontrol('Style','text',...
      'Position',[400 45 120 20],...
      'String','Vertical Exaggeration');

 % Make figure visible after adding all components
   set(f,'Visible','on');

function setmap(source,event)
      val = source.Value;
      maps =  source.String;
      newmap = maps{val};
      colormap(newmap);
   end

   function surfzlim(source, event)
     val = 51 - source.Value;
      zlim(ax,[-val val]);
   end
end
```

GUI - netgui.m

nvkurbatova@sfedu.ru

О свойствах полей в source!



Преобразование к безразмерным элементам управления

- Элементы управления создаются по умолчанию в **pixels**;
- Добавив при создании всех элементов (даже, если уже position задаётся в pixels) поле и свойство: '**units**','**normalized**' или переопределив **name.units='normalized'** , обеспечиваем возможность изменить размер figure, т.е. пререопределить : **set(f,'resize','on');** если даже до этого «заперли» resize →off

Создаем элемент управления в единицах по умолчанию, изменяем единицы измерения, например:

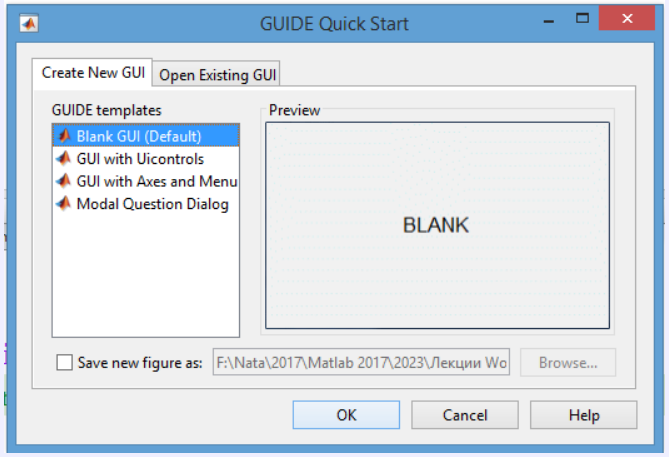
```
% Create slider
    sld = uicontrol('Style', 'slider',...
        'Min',1,'Max',45,'Value',41,...
        'Position', [400 20 120 20],...
        'Callback', @surfzlim, 'units','normalized' );
```

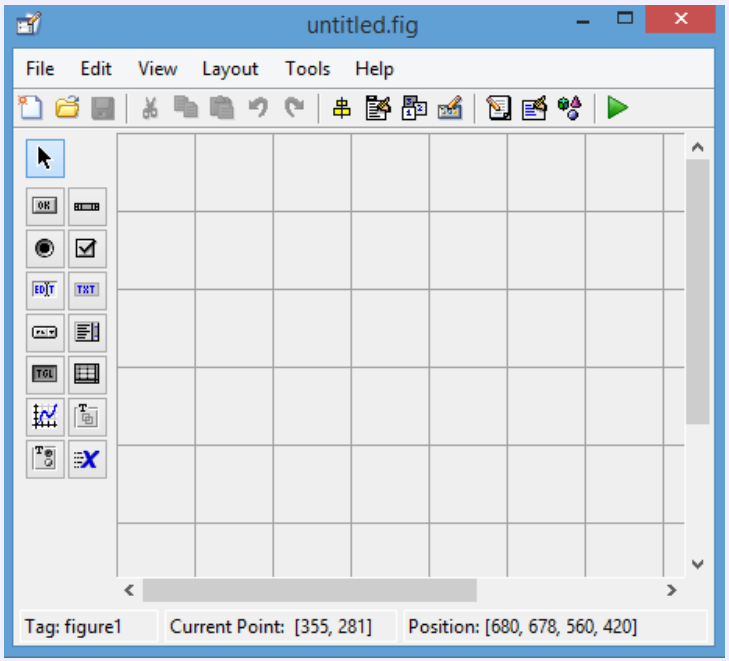
Кажется, противоречие?! Да, но цель - не травмировать бессознательного пользователя!

Конструктор GUI

nvkurbatova@sfedu.ru

```
K>> guide
```





Создание ЭУ в поле Blank –> OK

Преимущества: простота редактирования, изменение свойств.

Автоматически создается скрипт с выбранным именем, например, nameGUI.m

Недостатки: требуемая глубина постижения процесса;

сложность модификации связей и добавления новых элементов.

Простота конструктора GUI компенсируется сложностью реализации, особенно для многооконных алгоритмов, при условии его динамического реструктурирования!

Функции управления графическими объектами

nvkurbatova@sfedu.ru

gco – возвращает дескриптор («указатель») текущего графического объекта;
gcbo – возвращает дескриптор объекта (кнопки - pushbutton), чья функция в данный момент выполняется;
gcbf – возвращает дескриптор окна figure, содержащего объект, функция которого в данный момент выполняется – важно для многооконного интерфейса;
drawnow – выполнить очередь задержанных графических команд – важно при программировании анимации;
findobj – найти объекты с заданными свойствами (**поле-свойство структуры**);
copyobj – скопировать объект и порожденные им объекты.

Help по созданию GUI классическим способом
(I-м подходом) **получите в Moodle или Teams!**

nvkurbatova@sfedu.ru

Спасибо за внимание!

«Мечты, когда стареют, непременно становятся кошмарами» - Ромéн Гарú ~ Эжен Ажар

«Не тяните с мечтами!» NB