Лекция. Создание графического пользовательского интерфейса – GUI

к курсу «Пакеты компьютерной алгебры»

к. ф.-м. н. Курбатова Н.В.

2018 г.

Два подхода создания GUI

(I – эволюционный, II – революционный; идеально в комплексе – I и II)

I. Все элементы создает пользователь и процедуры	**II. Элементы GUI и процедуры создаются средствами 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 (выравнивать элементы и т.п.) • система генерирует guiname.m и исполняемый guiname.fig
минусы	минусы
• теряем время на этапе построения *геометрии* GUI	• требуется более высокий уровень понимания • сложности возникают на этапе модификации GUI

Элементы GUI

figure – родительский объект всех элементов GUI, поле для интерфейса

элементы управления – editableText, staticText, frames, checkboxes, listboxes, popupmenus, sliders, pushbuttons, radiobuttons (togglebuttons); uicontrol – единый конструктор элементов управления

axes – оси, конструктор осей

uigetdir, uigetfile – функции обеспечивают интерактивный выбор файлов (совместно с эу)

Root и Figure

- Построение GUI начинаем с команды: **f=figure**
- root (дескриптор - 0) – родительский объект для figure
- Размер Figure (position) привязан к размеру монитора, [x0,y0,hx,hy]=get(0, `screensize`)
- Цвет поля figure (зачем знать?)

  f=figure, colorfigure=get(f, `color `)
- set(gcf,'NumberTitle', 'off') – без заголовка figure
- set(gcf,'name', 'My interface') – заголовок пользователя

Figure. Свойства полей units и resize структуры figure

- размер figure изменяется некорректно, если

  ... `units` , ` pixels`, ...
- размер figure «привязан» к размеру монитора:

  [x0,y0,lx,ly]=get(0, `screensize`) и свойства

  поля *position* всех элементов GUI функционально зависят от *x0,y0, lx,ly (lx,ly* размеры экрана*)*
- Non-resizable — неизменяемый размер figure, задается свойство *... `resize` , ` off`, ...*
- Proportional — пропорционально изменяемый размер figure

  - в соответствии с правилом запрограммированном в ResizeFcn
  - нормированием с помощью свойства*... `units` , ` normalized`, ...*

  (в случае *normalized* размеры окна figure единичные)

Общие свойства всех элементов GUI

- поле **position**, значение [x0,y0,hx,hy]
- поля **backgroundcolor** (**color** – figure, axes),
  значение [r,g,b] r,g,b∈[0,1];
  [0,0,0] – ' *black* ', [1,1,1] – '*white* '
- поле **string**, значение '*надпись* ', или
  значение массив ячеек из строк:

… ' string ' ,{' строка1 ', ' строка2 ', …} …

Задание типа элемента управления

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

Стратегия построения GUI (I-й подход)

1) создаем головной файл, например, guiname.m – процедура без параметров
2) описываем переменные как глобальные: **global var1 var2**; если их нужно использовать во внешних процедурах; либо оперируем параметрами подпроцедур
3) строим окно для всех элементов GUI, со свойствами по умолчанию: **f=figure**
4) изменяем свойства, например, единиц измерения:

   set(f, 'units', 'normalized'), если необходимо,

1) создаем все элементы GUI (элементы управления, оси и т.д.)
2) задаем (или не задаем) программно «начальные» значения, предшествующие диалогу
3) программируем функционал, основанный на обработке событий, обеспечивающих ввод-вывод информации или тот или иной выбор
4) программируем функционал (процедуры в отдельных файлах или handle процедуры - подпроцедуры), основанный на обработке событий, обеспечивающих запуск процессов

Элементы управления - **text**

статический, неизменяемый элемент

Обязательны свойства:

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

- дескриптор f (или gcf) не обязателен, элемент строится в активном figure
- первые буквы в имени полей могут быть заглавными и строчными
- значения остальных свойств выбираются по умолчанию (см. set(text1))

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')** обеспечивает выбор цвета фона figure

Элементы управления - **edit**

динамический, изменяемый элемент

Обязательны свойства:

ed = uicontrol(f, ' Style', 'edit', 'String', ' x0 ', 'Position', [x0t y0t lxt lyt])

- поле edit предназначено для ввода данных, чаще пустое, но на этапе отладки целесообразно задать мотивированное значение, f.e.
 'String', ' x0 '

Дополнительные свойства (аналогичны text)

Обработка события - получение введенной информации :

Информация извлекается из поля String:

stringvalue=get(ed, 'String') % информация извлекается из поля String

doublevalue=str2num(stringvalue) % конвертируется в числовую (double)

Элементы управления: **checkbox, radiobutton**

динамические, изменяемые элементы

checkbox

Обязательны свойства:

```
chbox = uicontrol('Style', ' checkbox ','String', ' поясняющая надпись',...
        'Position', [x0chb  y0chb  lxchb   lychb]);
```

radiobutton

```
rb= uicontrol('Style', ' radiobutton ','String', ' поясняющая надпись',...
        'Position', [x0rb  y0rb  lxrb   lyrb]);
```

назначение обоих элементов – переключатель, разветвление (наличие или отсутствие конкретной опции)

checkbox и **radiobutton** отличаются по форме

Обработка события :

события: выбор опции – генерируется 1, в противном – 0

Система сохранаяет как свойства поля value структур chbox или rb;

извлекаются командой:

```
r=get(t,'value')
```

Построение **pop up menu**

popup

Обязательны свойства:

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

- аккумулирует свойства кнопки (запускается action) и списка (выбирается требуемая ветвь)

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

- если размера выделенной области недостаточно, то реализуется как выплывающее меню

Построение осей **Axes** конструктором **axes**

Axes (gca)

создание осей конструктором:

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

маленькие хитрости:

- axes(ha) - задание активных осей
- cla - очищение активных осей
- set(ha, ' Xtick', [], ' Ytick', []) – отказ от нанесения разметки на осях

Поиск элементов управления:

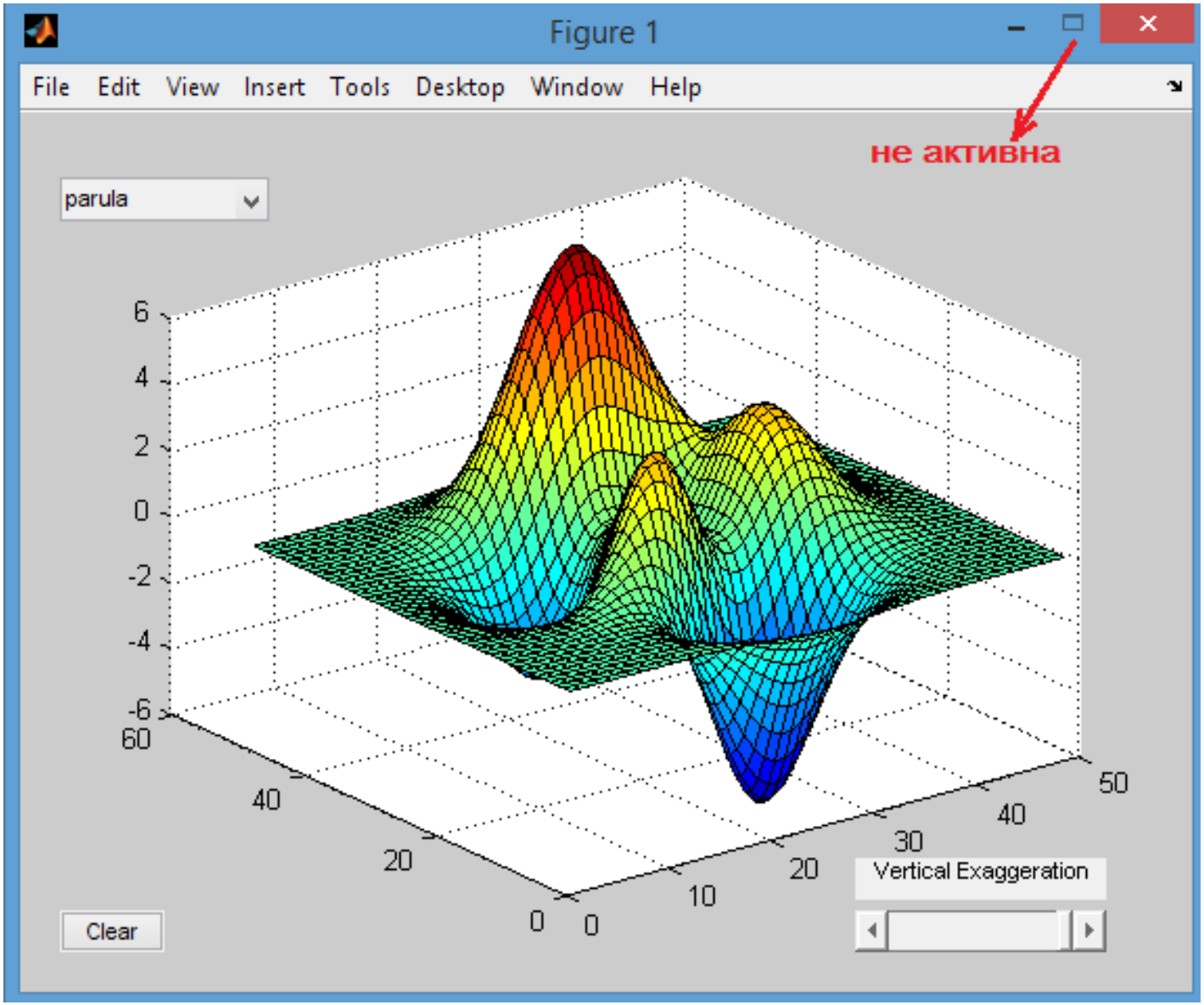
Obj= findobj(f, 'style', 'list'),

Obj= findobj(f, 'tag', 'specific'), (ярлык-метка присваивается пользователем объектам, связанным определенной логикой)

Стратегия построения GUI (II-й подход) факультативно

1) запускаем в CW редактор guide (построения GUI) в режиме blank GUI
2) строим, используя панель элементов (пиктограммы) необходимые объекты
3) меняем свойства элементов, если необходимо
4) меняем надписи, выделяем элемент; правой кнопкой мыши активируем динамическое меню, выбираем property inspector, с его помощью редактируем свойства
5) выравниваем в соответствии с предлагаемыми шаблонами в меню align objects все элементы
6) для элементов push button - кнопок также средствами динамического меню выбираем view callback и используем предлагаемый функционал (создаем-удаляем функции, обрабатываем нажатие клавиш и т.д.)
7) сохраняем результирующие файлы guiname.m guifigname.fig; генерируется головной файл и подпроцедуры, которые подлежат корректировке

Простейшее меню (resize-off)

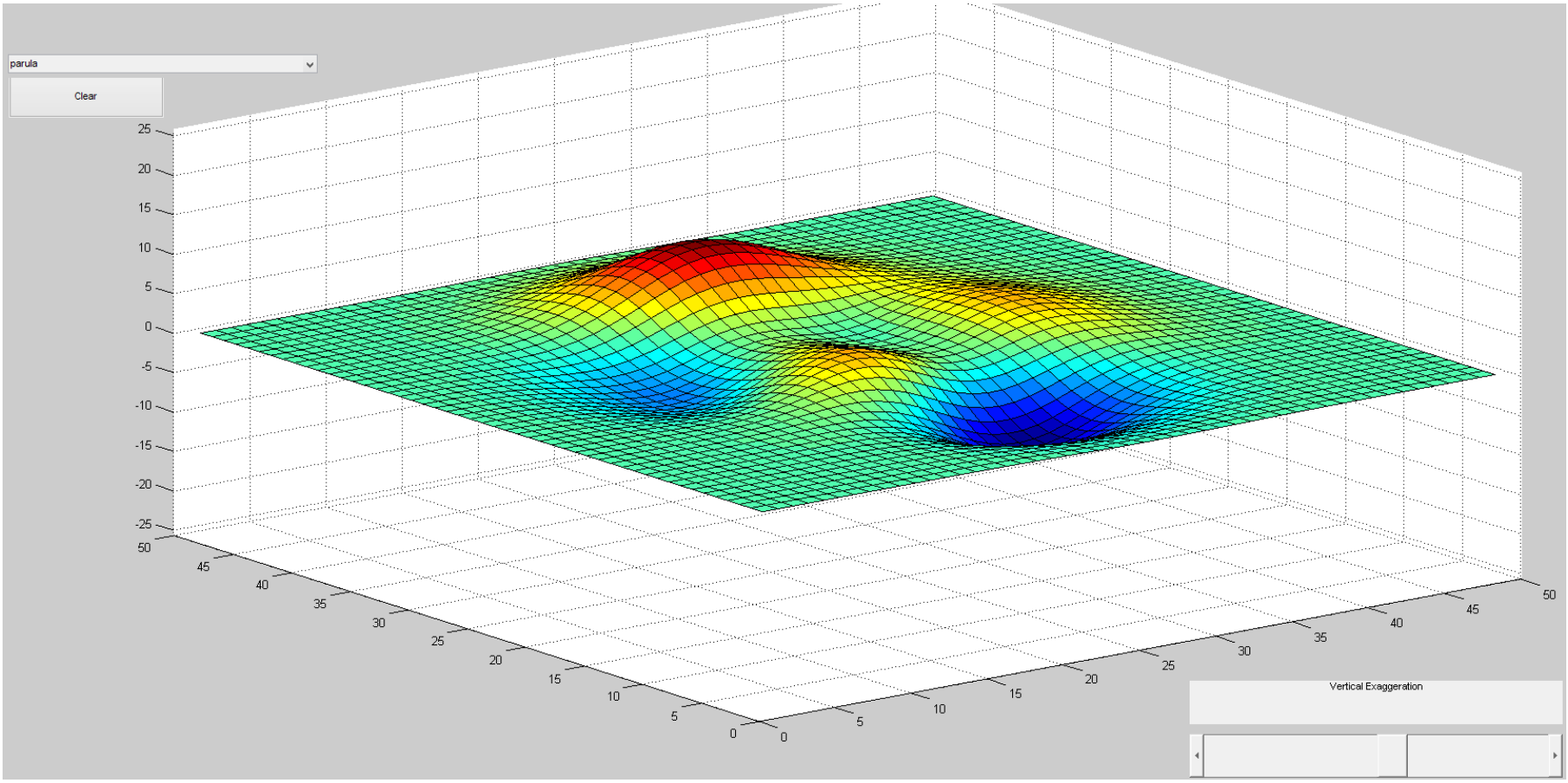


Пример кода (записано в одном файле nvgui.m)

```
function nvgui
%% Dot notation runs in R2014b
   % Create a figure , default pixels
   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
   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 visble after adding all
components
   set(f,'Visible','on');
   set(f,'Visible','on'); %  f.Visible='on'

   function setmap(source,event)
      val = get(source,'Value');
      maps = get(source,'String');
      % val = source.Value;
      % maps =  source.String;

      newmap = maps{val};
      colormap(newmap);
   end

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

Простейшее меню (units-normalized)



Пример кода (преобразование к автоматическому масштабированию)

```
function nvgui
% Create a figure and axes
   f = figure('Visible','off');
   sizefigure=get(f,'position')

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

   % pop-up menu - convert pixels [20 340 100 50]  to
normalized:

   hx=sizefigure(3) % нормирующий множитель по x -
длина figure по оси x
   % безразмерные координаты левого нижнего угла
'popup' меню
   x0pp=20/hx; %
   hy=sizefigure(4);% нормирующий множитель по y -
длина figure по оси y
   y0pp=340/hy;

% безразмерные длины 'popup' панели по оси x  и по
оси y -- по вертикали; pp-означает принадлежность
'popup' меню
   lxpp=100/hx;  lypp=50/hy;
 popup = uicontrol('Style', 'popup',...
         'String', {'parula','jet','hsv','hot','cool','gray'},...
         'Position', [x0pp y0pp  lxpp lypp],
'units','normalized',...
         'Callback', @setmap);
```

```
% push button  - convert pixels [20 20 50 20]  to
normalized:
  x0pb=20/hx;  y0pb=20/hy;  lxpb=50/hx;
lypb=20/hy;

   btn = uicontrol('Style', 'pushbutton', 'String',
'Clear',...
       'Position', [x0pb y0pb  lxpb
lypb],'units','normalized', 'Callback', 'cla');

  % slider - безразмерные  для [400 20 120 20] :

   sld = uicontrol('Style', 'slider',...
       'Min',1,'Max',45,'Value',41,...
       'Position', [400/hx 20/hy 120/hx 20/hy],
'units','normalized', 'Callback', @surfzlim);
% Add a text uicontrol to label the slider
 % безразмерные параметры  [400 45 120 20] text:
   txt = uicontrol('Style','text',...
       'Position',[400/nx 45/ny 120/nx 20/ny], ...
'units','normalized', 'String','Vertical Exaggeration');

% Exaggeration - раздувание
   % Make figure visible after adding all
далее  текст программы примера (resize off)  без
изменений
```

Выводы

- Если вы используете интерфейс всегда на одном и том же компьютере, то Units оставляйте по умолчанию; в противном случае Units - normalized или Resize - off
- Если нужно сделать однооконный простой интерфейс, используйте guide - редактор GUI
- Если предполагается непростой дизайн GUI, используйте guide на этапе создания геометрии панели GUI
- В отсутствии опыта использование guide не способствует освоению техники построения GUI
- Использование Guide требует квалификации

Спасибо за внимание!