Курбатова Н.В. «**Методические указания Построения GUI»**

1. Графические окна системы MATLAB и элементы управления.

Работая в MATLAB, мы уже имели дело с графическими окнами (команда figure), в которых сама система создавала графический объект Axes, прорисовывала оси системы координат и графики функций (в том числе трёхмерные) командами plot, plot3, mesh, surf или surfl и т.п..

В системе MATLAB реализована возможность создавать графические пользовательские интерфейсы GUI и использовать для этого элементы управления (кнопки, поля редактирования, поясняющие тексты, слайдеры и т.д.). Они располагаются на поверхности графических окон и позволяют интерактивно вводить и читать числовую и текстовую информацию, нажатием кнопок инициировать выполнение нескольких M-функций, визуализировать результаты вычислений в поле графического объекта Axes.

Для построения GUI рассмотрим следующие графические объекты управления: объект Axes, предназначенном для построения графиков функций, визуализации фотографий, анимаций, а также несколько объектов общего типа, которые создаются конструктором uicontrol. К последним относятся *командные кнопки*, *текстовые поля* с возможностью редактирования текста и без такой возможности, *переключатели* и *списки, слайдеры…*

Все перечисленные выше графические объекты выполнены в системе MATLAB по *объектной ориентированной технологии* и характеризуются присущим им *набором свойств*. Меняя последние, изменяем внешний вид и поведение. Чтобы изменить свойства графических объектов, нужно получить доступ к ним по их *дескриптору-описателю.*

Рассмотрим процесс подробно, изнутри, не прибегая к guide – конструктору GUI. Впоследствии функционал guide будет прозрачнее.

Описатель графического пользовательского объекта типа handle содержит (возвращает) свойства этих объектов. Дескрипторы динамических (изменяемых) элементов управления в таком случае следует декларировать как глобальные. Описав эти переменные как глобальные, в головной программе и процедуре, где осуществляется обработка соответствующего события, обеспечиваем доступ к свойствам элементов управления.

Существуют системные дескрипторы такие, например, как функция gcf , что означает GetCurrentFigure, – возвращает описатель текущего графического окна; gca – текущих осей; gco – текущего объекта; gcbo – возвращает handle-дескриптор текущего объекта, чей callback сейчас выполняется; gcbf – возвращает дескриптор figure, который содержит объект, чей callback в данный момент выполняется.

Команда figure создаёт графическое окно цвета, заданного по умолчанию, этот цвет полезно запомнить, используя системный описатель gcf и *свойство* 'Color':

```
FigureColor=get( gcf, 'Color')
```

Изменим цвет figure радикально, на красный, не надолго:

set(gcf, 'Color', 'red')

Все графические объекты (являются окнами в терминах операционной системы Windows) на поверхности графического окна создаются по иерархичнской схеме "родитель - потомок". Родительским окном для элементов управления служит само графическое окно MATLABa – командное окно, оно имеет дескриптор – 0.

С учётом введённого понятия становится удобным применение для поиска описателей графических объектов функции findobj:

hArray = findobj(hParent, *'ИмяСвойства'*, *ЗначениеСвойства*)

Она отыскивает все объекты, являющиеся потомками объекта с описателем hParent и имеющие для свойства ***'ИмяСвойства'*** значение, указанное в параметре функции ***ЗначениеСвойства*** и возвращает их описатели в массиве **hArray**. Заметим, что вместо hParent можно использовать gcf.

В качестве удобного для поиска свойства часто используется свойство 'Tag', *ярлык*, его значение – текстовый идентификатор, выбирается программистом по логике объединения, возможно, разнотипных элементов управления. Например, требуется сделать невидимыми одновременно разные элементы, то удобно задать у всех таких элементов для поля структуры 'Tag' значение 'PanelUnvisible'.

Для конструктора uicontrol, создающего элемент управления самым важным параметром после описателя родительского окна является свойство 'Style', так как оно задаёт тип управляющего элемента, кнопка, слайдер и т.д. По умолчанию размеры всех графических элементов заданы в пикселях pixels.

Создадим кнопку, для нее 'Style', 'pushbutton ' предварительно сохранив дескриптор figure в переменной hF1:

hF1 = figure;

hpb1=uicontrol(hF1, 'Style','pushbutton', …

'String', 'Start',…

'Position', [50 50 100 25]);

на кнопке появится надпись Start, она определяется свойством поля String, размер и положение кнопки – свойством 'Position' и задается точкой левого нижнего угла прямоугольника с координатами (50,50) и длинами по горизонтали (оси ox) – 100, по вертикали – 25 пикселей. Остальные свойства остаются со значениями по умолчанию. Создадим еще одну кнопу, которая будет полезна:

```
hpb2=uicontrol( hF1, 'Style','pushbutton', String', 'Clear', 'Position', [50 75 100 25]);
```

Создайте скрипт, кнопку, посмотрите, какие предусмотрены все возможные свойства кнопки с помощью команды set(gco) или с использованием дескриптора пользователя hpb1, set(hpb1) . Кнопка нажимается :) .

Команда set, позволяет получить все возможные значения любого поля–свойства, которые *предусмотрены системой*, например, если мы используем команду в последнем коде в виде r = set(hpb1), тогда результатом r.Style{:} будут *все* возможные элементы управления. Заметим, что таким же образом можно узнать все предусмотренные свойства единиц измерения 'Units' , написав в командном окне r.Units {:}.

Важно! В MLv.2020 значения всех полей с заранее предусмотренным набором возможных значений являются массивом ячеек и написание имен полей зависит от верхнего-нижнего регистров (*строго соблюдайте – см. set(Интересующий дескриптор), получите информацию о названии всех свойств*), например, hpb1.Fontsize=12**,** однако в самом конструкторе uicontrol написание свойств полей свободно от верхнего-нижнего регистров, система поймет запись

hpb2=uicontrol(hF1**,** 'Style','pushbutton','String', 'Clear', 'Position', [50 75 100 25], 'fontsize',12**)** **!**

2. Создание основных элементов управления.

a) *Edit – редактируемый текст.*

В предыдущем параграфе мы создали командную кнопку. Теперь рассмотрим другие элементы управления в hParent, заметим, что hParent может быть gcf или дескриптор соответствующего родительского графического объекта. Для ввода и редактирования входной информации (числовой и текстовой) предназначены *текстовые поля с возможностью редактирования*. Они создаются командой

uicontrol(hParent, 'Style', 'edit',…)

К нашей кнопке добавим поля редактирования с заранее заданным текстом – 'x=*0:0.01:pi*', который станет вектором значений для функции. Здесь для Edit прописано свойство 'BackgroundColor' белым цветом (тёмно-серый – по умолчанию) и свойство 'HorizontalAlignment' (выравнивание текста вдоль горизонтали – по левой границе (выравнивание по центру – по умолчанию). Заметим, что свойство 'BackgroundColor' *нельзя было задавать для кнопок более ранних версий до MLv.16* (требование операционной системы Windows), теперь можно всё, раз уж даже складывать векторы и матрицы) **he=uicontrol(hF1, 'Style', 'edit', 'Position', [50 100 100 50],…**

'BackgroundColor', 'white',…

'HorizontalAlignment', 'left',…

'String', 'x=0:0.01:pi', 'Fontsize',12);

Если поле String не задано, то окно редактирования пустое. В любом случае оно готово отображать набираемый текст или редактировать уже имеющийся текст. Элемент управления Edit является динамическим.

b) *Text – нередактируемый текст.* Статическим текстовым элементом управления, текстом, является элемент типа Text, это поясняющий текст на поле GUI, в том числе, объясняет назначение динамических элементов, его можно изменить программно, но не интерактивно, с клавиатуры. Создадим конструктором нередактируемое текстовое поле:

ht=uicontrol(hF1, 'Style', 'text', 'Position', [50 150 100 35],…

'BackgroundColor', get(hf, 'color'), …

'String', 'input x');

Вместо поля *'BackgroundColor'* и его значения *get(hf, 'color')*, в последней команде можно использовать сохраненный ранее цвет фона, см. § 1 и зададим свойству 'BackgroundColor' значение FigureColor.

c) *Checkbox – флажок.* Checkbox является динамическим элементом управления, поддерживает обратную связь и служит аналогом бинарного переключателя:

hchb=uicontrol(hF1, 'Style', 'checkbox', 'Position',[200 300 200 50],…

'String','with out Ticks ',…

'Fontsize',11);

Свойство поля 'HorizontalAlignment' равно 'left' – *по умолчанию*.

Этот элемент управления может пребывать в *одном из двух состояний*: *отмеченном* или *неотмеченном*. В первом из них, в квадрате должна стоять "*галочка*", которую проставляют щелчком левой кнопки мыши. Повторный щелчок убирает "галочку". А система формирует значение 1 в отмеченном состоянии и 0 в отсутствии опционального выбора. Признак 1 – в нашем GUI соответствует наличию фотоэмблемы, 0 – Logo.

d) *Listbox – список*

Наконец, создадим *список*, содержащий несколько имён функций:

hlb=uicontrol(hF1, 'Style', 'listbox', 'Position',[410 320 70 65],…

'String',{'sin','cos','tan', 'abs'});

Так как список содержит сразу несколько строковых значений для свойства 'String', то значение этого поля структуры является *массивом ячеек* строк типа char. Имеем возможность прокрутки при создании окна меньшего размера или продвижения курсором по предлагаемому списку, с выделением текущего выбранного значения.

Существуют еще другие элементы управления, например, слайдер-движок, он позволяет задавать, регулируя, величину числового коэффициента, также является динамическим. Все динамические элементы предполагают *обработку события*.

3. Раздумья о корректности построения GUI

Тут самое время задуматься, а всё ли в порядке, в «датском королевстве». Поскольку все элементы строятся в пикселях, то при раскрытии окна, они свалятся в левый нижний угол, и GUI примет печальный вид. Простым способом борьбы с этим *проколом* является запрет на раскрытие figure.

set(hF1, 'resize', 'off') % запрет на изменение размера окна figure

Второй способ – строить все элементы управления в **'normalized'** – безразмерных единицах измерения **'Units'** (в т.ч. и корневое окно root с нулевым дескриптором), тогда максимальный размер экрана имеет единичный размер по высоте и ширине (с различными системными масштабирующими множителями), размер figure относительно root задается меньшим единицы, соотносясь с размером экрана, однако для своих потомков окно figure также имеет единичный размер по высоте и ширине, стоит лишь при создании потомков задать **'Units', 'normalized'**. Такой подход инвариантен как от редактирования размера окна figure, так и при запуске GUI на компьютере с экраном другого размера (в пикселях).

И, наконец, третий вариант – строить всё в пикселях, но ориентироваться на размер вашего (произвольного) экрана:

```
ScSz=get(0, 'screensize')
```

здесь ScSz – вектор с компонентами: [1,1, valueHx, valueHy], где (1,1) – координаты левого нижнего угла экрана, в пикселях, а также valueHx длина и valueHy ширина экрана. В этом случае координаты левого нижнего угла *figure* должны быть заданы не как абсолютные величины, а как функции от valueHx, valueHy, равно, как и координаты всех элементов управления на поле figure – относительно характеристик figure и тоже не в абсолютном выражении, а относительном – параметрически.

4. Создание графического объекта Axes. Графический объект Axes системно появляется при построении графиков функций. Axes самостоятельный элемент (сам себе конструктор) предназначен для визуализации.

Создаётся объект Axes с помощью функции axes (конструктор объекта). Поместим к уже построенным объектам оси; одни – для отображения графика, другие – эмблемы или фотографии:

```
ha1=axes(hF1, 'Color', [ 1 1 1], 'Units', 'points', 'Position', [ 160 50  200 150 ], …

'FontSize', 11 );

ha2=axes(hF1, 'Color', [ 1 1 1], 'Units', 'points', 'Position', [ 30 200  100 100 ], …

'FontSize', 11 );
```

В результате на панели figure имеем двое осей, две кнопки запуска процесса и очищения осей (здесь оба очищаем одновременно), окошко для ввода аргументов функций, с поясняющим текстом, список и опциональный признак (Checkbox). Создадим скрипт MainGui.m со всеми созданными ранее элементами управления:

hpb, hpb2, he, ht, hchb, hlb, ha1,ha2.

Все команды, создающие графическое окно, а также объекты Axes и элементы пользовательского интерфейса, размещаем в m-файл. Этот файл будет служить сценарием создания такого окна (лучше оформить его процедурой без входных-выходных параметров). Чтобы графическое окно GUI не содержало панель MatLab, следует изменить свойство 'menubar':

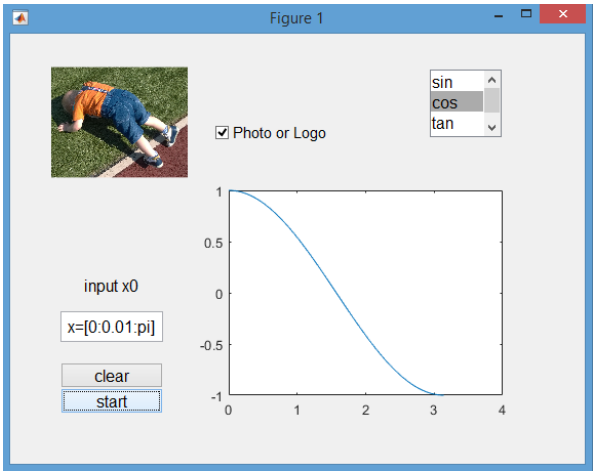
```
set(hf, 'menubar', 'none'),
```

когда ваш интерфейс будет отлажен, можно убрать строку Figure с номером, для этого отменяем значение по умолчание в поле 'numbertitle':

set(hf, 'numbertitle', 'off')

Все свойства figure доступны по команде **set(hf)**, а основные, например, в командном окне в режиме отладки или при выполнении скрипта, содержащего строку из одного идентификатора **hf** в отсутствии подавления вывода.

В результате мы получили небезупречную панель, например, с точки зрения, размера шрифта. Его можно было при создании не задавать, но в тех элементах, в которых шрифт следует потом изменить добавить поле 'Tag', например, с придуманным значением 'MyFontResize'. В данном случае в осях ha1, ha2 шрифт по умолчанию. Найдём все объекты с таким ярлыком и разом у всех заменим размер: Allobjects=findobj(hf, 'Tag', 'MyFontResize'); set(Allobjects, 'fontsize', 12) или set(findobj(hf, 'Tag', 'MyFontResize'), 'fontsize', 12)



Здесь опционально выбирается фото, в отсутствии галочки – логотип текстовый. *Лирическое отступление. Мой фотологотип имеет двойное назначение: планка – она во всех смыслах – стимул прогресса, внутреннего, физического и пожелание вам справиться с построением GUI, преодолеть трудности, разобраться и получить удовольствие. Успехов!*

Задекларируем все динамические элементы управления как глобальные в файле MainGui.m, где они создавались, а также и во внешних процедурах, ожидающих данные для реализации алгоритма. Данные передаются через динамические элементы управления в результате обработки событий, эти элементы и следует описать как глобальные в соответствующих процедурах. Такой подход позволит понять природу программирования интерфейса и легче ориентироваться при интерактивном программировании интерфейса в среде guide.

5. Callback функции. Функции, которые связывают с графическими элементами управления, называют callback-функциями (они вызываются средой MATLAB у нас "за спиной", поэтому и callback).

Чтобы связать наши кнопки с callback-функциями, которые будут вызываться средой MATLAB при нажатии на эти кнопки, необходимо при их создании с помощью uicontrol предусмотреть свойство 'Callback', указав ему в качестве значения имя m-функции или строку ($\in$Char и являющуюся командой или серией команд), которую следует запустить при нажатии на кнопку.

Первый путь – предусмотреть это сразу при создании hpb1 и hpb2, второй – добавить значение свойства уже созданному элементу:

set(hpb1, 'Callback','GraphChoice')

При нажатии кнопки будет запускаться функция GraphChoice.m, это касается кнопки с дескриптором hpb1, а для кнопки hpb2 предусмотрим очистку экрана, которая выполнится при запуске функции axesclear.m следующей командой:

set(hpb2, 'Callback', 'axesclear')

фактически мы расширили свойства кнопки, добавив поле и соответствующее свойство. Здесь приводится содержание внешней функции axesclear.m:

```
function axesclear
global ha1 ha2
axes(ha1); cla
axes(ha1); cla
```

По команде axes (ha1) оси ha1 назначаются активными (текущими), а функция cla очищает оси. Заметим, что clc – очистка командного окна (CLean ComandWindows); clf – очистка окна figure (CLean Figure).

Оформим далее внешнюю процедуру построения графика MainGui.m. В нашем случае построения графика, но на практике – содержательная составляющая вашего GUI.

6. Реализация алгоритма

В MainGui.m все описатели созданных динамических графических объектов следует задекларировать как глобальные, не обязательно в начале файла (желательно).

```
global hpb1 hpb2 he hchb hlb ha1 ha2;
```

Текстовый элемент ht не декларируется, т.к. он не изменяется интерактивно!

Все функции, где эти описатели маркированы как *глобальные*, имеют к ним непосредственный доступ.

Теперь наполним содержанием функцию GraphChoice.m, которая запускается по нажатию кнопки Start. Эта функция опрашивает редактируемые поля, в результате чего инициируются для счёта переменные (конвертируются в числовой формат или подходящий логике программы) и выполняет необходимые действия. В нашем случае выбираются функции и аргументы, строится график, а в поле ha2 опционально добавляется эмблема: есть галочка – фото, нет галочки – текст, аббревиатура (в нашем случае Logo).

```
function GraphChoice   % Построение графика:

global he hchb hlb ha1 ha2;
    % Получение информации из editboxes -----
    % обработка событий – интерактивных манипуляций с интерфейсом:

    str1=get( he1,'String' ); eval(str1);   % обработали Edit, загружен вектор x

      % получим номер элемента–функции (свойство поля Value),
      % в массиве ячеек ,{'sin','cos','tan', 'abs'}:

index=get( hlb,'Value' );
```

```
% получим саму функцию:
fun=get( hlb,'String');  % fun – cell array
currentfun=fun{index}   % currentfun – string
% Выполнить функцию – вычислить значения функции от x:
y=feval(currentfun,x);
plot( x , y );
% Формирование эмблемы в ha2:
axes(ha2) % активно второе окно
if hchb.Value==1
    % Фото эмблема в ha2:
   imjpg=imread('aim', 'jpg');% см. help о возможных форматах
   C=imshow(imjpg); % C – матрица цвета каждого пикселя
else
 % Logo в ha2:
text(0.5,0.5,'Logo', 'fontsize',10)
end
```

7. Дополнительные средства GUI:

В предложенной реализации GUI информация вводится посредством Edit, но если её необходимо получить из внешнего файла, следует предусмотреть соответствующий функционал. Так для чтения текстовой или графической информации в GUI подойдёт функция

[FileName, PathName]= uigetfile(FileSpec, 'explaining text'),

её необходимо поместить в файл, где будет выполняться процедура обмена в виде

[FileName, PathName]= uigetfile('*.bmp', 'Find your photo.jpg ')

в итоге появится стандартное окно Windows с заголовком *Find your photo.jpg*, и станет возможной навигация по папкам, найдется правильное фото, а после его выбора в переменных FileName, PathName сохранятся соответствующие имена файла и пути, которые пригодятся для функций ввода информации из файлов с расширениями xls, txt, mat и т.п., см. help.

Созданное нами законченное приложение в рамках среды MATLAB является модельной для пошаговой иллюстрации принципов построения GUI.

Рассмотренный подход, является наилучшим с точки зрения максимального контроля разработчика над процессом создания и редактирования интерфейса. В случае недостаточной квалификации разработчика (или необходимости очень быстрого выполнения лаконичного GUI) может пригодиться интерактивное средство разработки, входящее в пакет MATLAB – guide приложение – конструктор создания GUI. По этой команде на экран дисплея выводится специальное окно, содержащее палитру графических элементов управления. С помощью мыши можно методом буксировки "перетаскивать" эти элементы на создаваемое собственное графическое окно, выравнивать элементы управления, динамические элементы связывать с подфункциями. Callback-функции, содержательная часть алгоритма, должна быть написана пользователем. Для изучения приложения guide в помощь, например, ролики см. по ссылкам

https://www.youtube.com/watch?v=1KKAlyY3onI

на английском:

https://www.mathworks.com/videos/creating-a-gui-with-guide-68979.html