МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_25_» _09_____ 2023 г.

ВВЕДЕНИЕ В MATLAB

Методические указания по выполнению самостоятельной работы
для студентов специальности
30.05.03 «Медицинская кибернетика»

Курск 2023

УДК 004.93:61

Составитель: О.В. Шаталова.

Рецензент
Кандидат технических наук, доцент *М.А. Ефремов*

Введение в MATLAB: методические указания по выполнению самостоятельной работы для студентов специальности 30.05.03 «Медицинская кибернетика» / Юго-Зап. гос. ун-т; сост.: О.В. Шаталова. - Курск, 2023. - 43 с.

Предназначено для студентов специальности 30.05.03 «Медицинская кибернетика» по дисциплине «Введение в MATLAB». Может быть использована аспирантами, обучающимися по направлению подготовки 1.5.8. Математическая биология, биоинформатика.

Текст печатается в авторской редакции

Подписано в печать . Формат 60×84 1/16.
Усл. печ. л. 2,50. Уч.-изд. л. 2,26. Тираж 100 экз. Заказ 984. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Принципы создания приложений с GUI

Приложения MATLAB являются графическими окнами, содержащими элементы управления (кнопки, списки, переключатели, флаги, полосы скроллинга, области ввода, меню), а также оси и текстовые области для вывода результатов работы. Создание приложений включает следующие основные этапы - расположение нужных элементов интерфейса в пределах графического окна и определение действий (команд MATLAB), которые выполняются при обращении пользователя к данным объектам, например при нажатии кнопки. Процесс работы над приложением допускает постепенное добавление элементов в графическое окно, запуск и тестирование приложения и возврат в режим редактирования. Конечным результатом является программа с графическим интерфейсом пользователя (GUI), содержащаяся в нескольких файлах, запуск которой производится указанием ее имени в командной строке MATLAB или в другом приложении.

Рассмотрим основные принципы создания приложений в MATLAB.

Среда GUIDE

Перейдите в среду GUIDE, выполнив guide в командной строке. В появившемся окне выберите Blank GUI и нажмите OK. Появится редактор окна приложения (рисунок 1), заголовок которого untitled.fig означает, что в нем открыт новый файл.

Редактор приложения содержит:

- строку меню;
- панель инструментов управления приложением;
- заготовку окна приложения с нанесенной сеткой;
- вертикальную и горизонтальную линейки;
- панель инструментов для добавления элементов интерфейса на окно приложения.

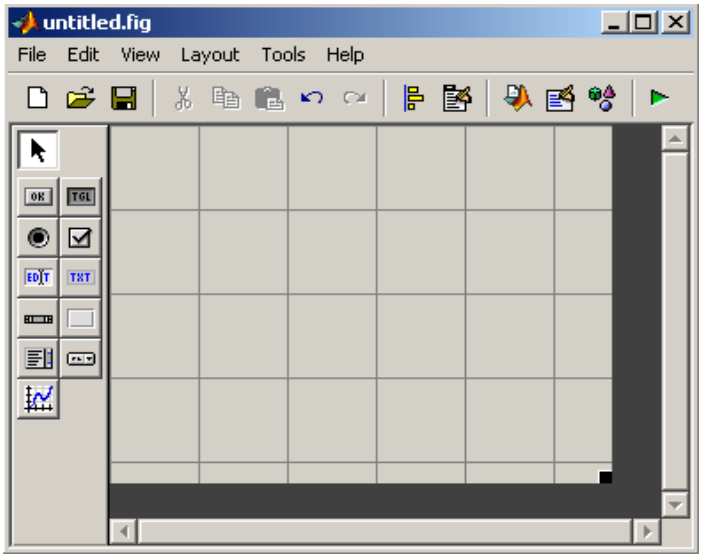

Рисунок 1 - Редактор приложения

Редактор приложения MATLAB позволяет разместить различные элементы интерфейса (рисунок 2). Требуется нажать соответствующую кнопку на панели инструментов и поместить выбранный объект щелчком мыши в требуемое место заготовки окна приложения. Другой способ состоит в задании прямоугольной области объекта перемещением мыши по области заготовки окна с удержанием левой кнопки. Размер и положение добавленных объектов изменяются при помощи мыши. Перед изменением размера следует выбрать режим выделения объектов и сделать объект текущим, щелкнув по нему клавишей мыши.

Рисунок 2 - Панель инструментов для добавления элементов интерфейса

Приложение в данный момент находится в режиме редактирования. Любой объект можно удалить из окна при помощи <Delete>, предварительно его выделив. Запуск приложения производится при помощи кнопки Run либо выбором соответствующего пункта меню Tools. Появляется диалоговое окно GUIDE, которое сообщает о необходимости сохранить приложение. Нажмите Yes и сохраните приложение в файле с расширением fig.

Приложение запускается в отдельном окне с заголовком Untitled. Пользователь может нажимать на кнопки, устанавливать флаги, переключатели, обращаться к спискам. При этом ничего полезного пока не происходит.

Недостаточно разместить элементы интерфейса в окне приложения, следует позаботиться о том, чтобы каждый элемент выполнял нужные функции при обращении к нему пользователя. Например, при нажатии на кнопку производятся вычисления и строятся графики полученных результатов, переключатели позволяют установить цвет линий, полоса скроллинга изменяет толщину линии, в области ввода пользователь указывает некоторые параметры, управляющие ходом вычислений.

Программирование событий

Приложение в MATLAB хранится в двух файлах с расширениями fig и m, первый из них содержит информацию о размещенных в окне приложения объектах, а второй является M-файлом с основной функцией и подфункциями. Добавление элемента интерфейса из редактора приложения приводит к автоматическому созданию соответствующей подфункции. Данную подфункцию следует наполнить содержимым - операторами, которые выполняют обработку события, возникающего при обращении пользователя к элементу интерфейса.

Создадим приложение, окно которого содержит оси и две кнопки, предназначенные для построения графика функции и очистки осей.

Перейдите в среду создания приложения командой guide.

Расположите на форме кнопку и оси так, как показано на рисунок 3. На кнопке автоматически размещается надпись

PushButton. Кнопка является элементом интерфейса, ей следует дать имя, которое уникальным образом идентифицировало бы ее среди всех объектов окна приложения.

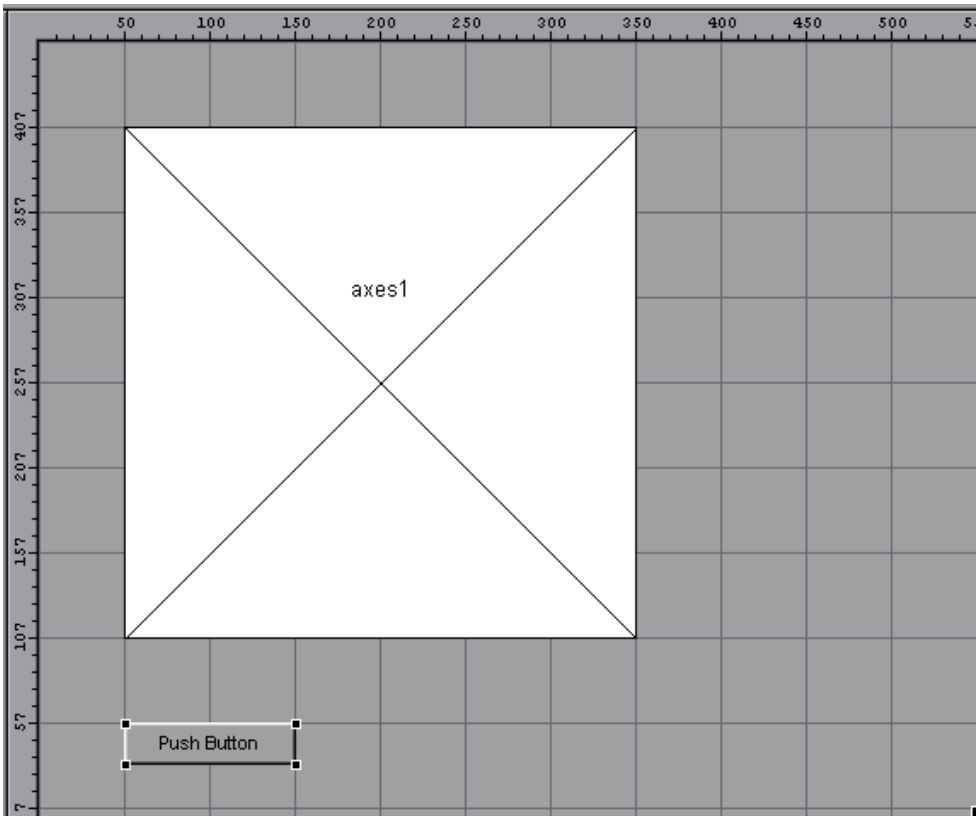

Рисунок 3 - Расположение кнопки и осей в окне приложения

Выделите кнопку PushButton и вызовите редактор свойств Property Inspector при помощи панели инструментов управления приложением. Появляется окно редактора свойств, в котором содержится таблица названий свойств кнопки и их значений. Занесите в свойство Tag значение btnPlot, щелкните мышью по строке справа от названия свойства, наберите требуемое значение и нажмите <Enter>. btnPlot теперь является именем кнопки PushButton. Удобно задавать имена, часть которых определяет тип элемента управления (btn соответствует button - кнопке). Аналогичным образом дайте осям имя axMain. Выберите в меню File редактора приложения пункт Save as, создайте папку MyFirstGui и сохраните приложение в файле mygui.fig. При этом открывается редактор М-файлов, содержащий файл mygui.m. Данный файл имеет структуру, схематично представленную в листинге 1.

Листинг 1. Структура М-файла приложения с графическим интерфейсом.

```
function varargout = mygui (varargin)
% Операторы инициализации приложения
% ABOUT CALLBACKS :
% Краткая информация о программировании событий
function btnPlot_Callback(hObject, eventdata, handles)
% Подфункция обработки события Callback кнопки с именем btnPlot
```

Приложение mygui содержит одну кнопку PushButton. Когда пользователь нажимает на Push Button в работающем приложении, то происходит событие Callback данного элемента управления. Вызывается подфункция btnPlot_Callback. Сейчас она не содержит операторов. Имя подфункции образовано названием кнопки и события. Очень важно задавать имена объектам в свойстве Tag сразу после их добавления на окно приложения в редакторе приложений, иначе генерируемая подфункция получит имя, которое сохранится при последующем изменении значения Tag и повлечет ошибки при выполнении приложения. Завершающий этап состоит в программировании действий, которые выполняются при нажатии пользователем на кнопку PushButton. Измените функцию обработки события нажатия на PushButton в соответствии с листингом 2.

Листинг 2. Обработка события Callback кнопки с именем btnPlot.

```
function btnPlot_Callback(hObject, eventdata, handles)
x=[-2:0.2:2];
y=exp(-x.^2);
plot (x,y)
```

Сохраните файл mygui.m в редакторе М-файлов и запустите приложение из редактора приложений, нажав кнопку Run. Нажатие на PushButton в запущенном приложении приводит к отображению графика функции на осях. Закройте окно приложения при помощи кнопки с крестиком в правом верхнем углу и продолжите работу над mygui в редакторе приложений.

Добавьте кнопку так, как показано на рисунке 4, задайте ей имя btnClear в редакторе свойств. Быстрый доступ к свойствам выделенного объекта в редакторе приложений производится из пункта Inspect Properties всплывающего меню при нажатии правой кнопки мыши на объекте. Перейдите к подфункции обработки события Callback добавленной кнопки, для чего следует выбрать пункт View Callbacks->Callback всплывающего меню.

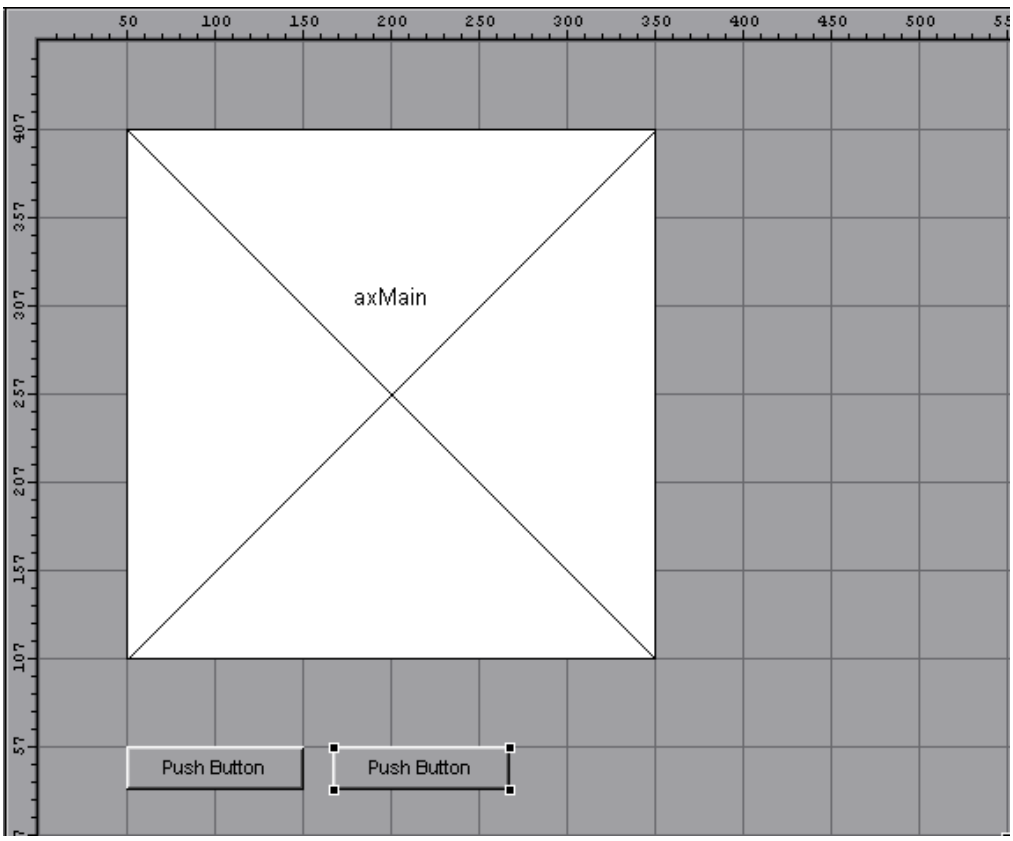

Рисунок4 - Окно приложения с двумя кнопками

Выбор данного пункта делает активным редактор М-файлов. Разместите единственный оператор очистки осей cla в подфункции (листинг 3).

Листинг 3. Обработка события кнопки с именем btnClear.

```
    function            btnClear_Callback(hObject,
eventdata, handles)
    cla
```

Запустите приложение и убедитесь, что нажатие на левую кнопку приводит к отображению графика функции, а правая служит для очистки осей.

Конструирование интерфейса

Управление свойствами объектов

Разработка приложения сопряжена с изменением свойств объектов, которые они получают по умолчанию при размещении их на заготовке окна. Некоторые из свойств, например надпись на кнопке или ее размер, устанавливаются при создании объекта в режиме редактирования. Другие свойства могут изменяться программно в работающем приложении.

Установка свойств при редактировании

Продолжите работу над приложением mygui, окно которого было изображено на рисунке 4. Очевидно, что следует подписать кнопки, например Построить и Очистить. Кнопки являются графическими объектами с определенными свойствами, среди которых имеется свойство, отвечающее за надпись на кнопке. Сделайте левую кнопку приложения mygui текущей и вызовите редактор свойств Property Inspector. Установите свойство String левой кнопки в значение Построить (рисунок 5.)

Значение свойства string соответствует надписи на кнопке, а Tag - имени или тегу кнопки, как объекта. Имена объектов используются для изменения их свойств в ходе работы приложения при выполнении блоков обработки событий от других элементов интерфейса. Перейдите теперь к свойствам правой кнопки и установите String в Очистить.

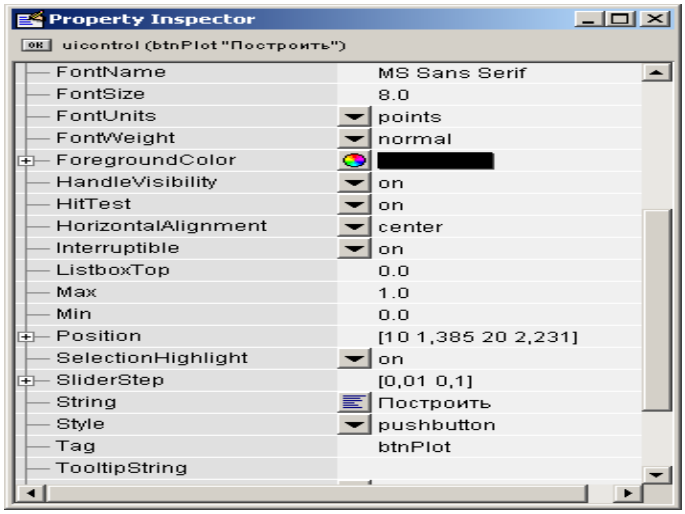

Рисунок 5 - Окно свойств Property Inspector

Доступ к редактору свойств выделенного объекта производится либо из панели инструментов управления приложением, либо из меню View редактора приложений, либо при помощи пункта Property Inspector всплывающего меню.

Значение свойства string сразу отображается на кнопке приложения, находящегося в режиме редактирования. Запустите приложение.

Программное изменение свойств

Большинство свойств объектов можно устанавливать программно прямо в ходе работы приложения. Усовершенствуйте приложение mygui следующим образом. Пусть при запуске доступной является только кнопка Построить, при нажатии на кнопку Построить выводится график и она становится недоступной, зато пользователь может нажать кнопку Очистить для очистки осей, и наоборот.

Решение поставленной задачи требует привлечения свойства Enable. Свойство Enable объекта отвечает за возможность доступа к нему пользователем, значение on разрешает доступ, а off, соответственно, запрещает. Установка значений свойствам объектов в программе производится при помощи функции set.

Функция set вызывается с тремя входными аргументами - указателем на объект, названием свойства и его значением, последние два аргумента заключаются в апострофы. Свойства

одного объекта должны изменяться в блоке операторов обработки события Callback другого объекта. Следовательно, должна иметься возможность доступа к указателю на любой существующий объект. Аргументы hObject и handles подфункций, которые обрабатывают события элементов управления, содержат требуемые указатели. В hObject хранится указатель на тот объект, событие которого обрабатывается в данный момент, а handles является структурой указателей. Поля структуры совпадают со значениями свойств Tag существующих элементов интерфейса. Например, handles.btnPlot является указателем на кнопку Построить с именем btnPlot. Доступ к Очистить должен быть запрещен в начале работы приложения, пока пользователь не нажмет Построить Установите в редакторе свойств для кнопки Очистить свойство Enable в off, используйте кнопку со стрелкой в строке со значением свойства. Остальные изменения значения Enable кнопок должны происходить в ходе работы приложения. Для разрешения и запрещения доступа к кнопкам нужно внести дополнения в обработку их событий Callback.

В подфункцию обработки события Callback кнопки Построить добавьте при помощи редактора вызовов:

- установку свойства Enable кнопки Очистить в значение on (после вывода графика следует разрешить доступ к Очистить);
- установку свойства Enable кнопки Построить в значение off (после вывода графика следует запретить доступ к Построить);

Аналогичные изменения произведите в обработке события Callback кнопки Очистить, а именно:

- установку свойства Enable кнопки Построить в значение on (после очистки осей следует разрешить доступ к Построить);
- установку свойства Enable кнопки Очистить в значение off (после очистки осей следует запретить доступ к кнопке);

Подфункции btnPlot_Callback и btnClear_Callback должны быть запрограммированы так, как показано на листинге 4.

Листинг 4. Обработка событий Callback кнопок btnPlot и btnClear.

```
    function btnPlot_Callback(hObject, eventdata,
handles)
    % Построение графика функции
    x=[-2:0.2:2];
    y=exp(-x.^2);
    plot(x,y)
    % Кнопка Построить должна стать недоступной
после вывода графика
    set(hObject, 'Enable', 'off')
    % Кнопка Очистить должна стать доступной
    set(handles.btnClear, 'Enable', 'on')
    function           btnClear_Callback(hObject,
eventdata, handles)
    cla % очистка осей
    % Кнопка Очистить должна стать недоступной
после очистки осей
    set(hObject, 'Enable', 'off')
    % Кнопка Построить должна стать доступной
    set(handles.btnPlot, 'Enable', 'on')
```

Сохраните изменения в редакторе М-файлов. Запустите приложение mygui и убедитесь, что всегда доступной является только одна из кнопок Построить или Очистить, что является хорошей подсказкой для пользователя о возможных действиях. Закройте окно приложения и редактор приложений.

Работа над приложением

Запуск приложения

Запуск приложения осуществляется не только из редактора приложений. Для запуска приложения достаточно в качестве команды задать его имя в командной строке

```
>>mygui
```

Появляется окно приложения, обращение к элементам интерфейса окна приводит к соответствующим действиям.

Каталог с приложением должен содержаться в путях поиска MATLAB или являться текущим.

Оформление интерфейса

Часто требуется, чтобы небольшое перемещение мыши вызывало изменение положения объекта на некоторый фиксированный шаг. Сетка редактора приложений позволяет осуществить такое дискретное движение. Выбор пункта Grid and Rulers меню Tools приводит к появлению диалогового окна Grid and Rulers, изображенного на рисунок 6.

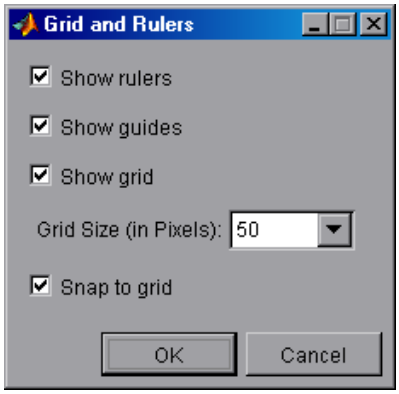

Рисунок 6 - Диалоговое окно Grid and Rulers

Флаги Show rulers и Show grid соответствуют отображению линеек и сетки в редакторе приложений, а раскрывающийся список Grid Size позволяет выбрать размер ячеек сетки. Минимально допустимый размер -десять пикселов - позволяет достаточно точно располагать элементы управления в окне приложений. Привязка перемещения к линиям сетки происходит при установленном флаге Snap to grid. Привязка разрешает разместить объект и изменить его размеры только при условии прохождения границы объекта по линиям сетки. Выбор мелкого шага сетки в сочетании с привязкой предоставляет разработчику возможность быстро оформить приложение. Плавно изменять положение выделенного объекта можно при помощи клавиш со стрелками. Одновременное удержание <Ctrl> приводит к перемещению с учетом привязки к сетке.

Программирование элементов интерфейса
Флаги и рамки

Флаги позволяют произвести одну или несколько установок, определяющих ход работы приложения. Продолжите работу над mygui, предоставив пользователю возможность наносить линии сетки на график. Окно приложения должно содержать два флага с названиями сетка по x и сетка по y. Если пользователь нажимает кнопку Построить, то на оси наносится сетка по выбранным координатам. Нажатие на Очистить должно приводить не только к исчезновению графика функции, но и скрытию сетки.

Обычно несколько элементов управления со схожим назначением группируются и помещаются внутри рамки. Измените размеры осей, освободив справа место для рамки. Нанесите рамку на окно приложения при помощи соответствующей кнопки. В рамку добавьте два флага. Разместите поясняющие подписи рядом с флагами и дайте им имена. Задайте свойству Tag верхнего флага значение chbxGridx, а свойству string, отвечающему за подпись флага, значение Сетка по x.

Аналогичным образом определите свойства нижнего флага, установите свойство Tag в chbxGridY, и свойство string в сетка по y. Если текст не помещается рядом с флагом, увеличьте ширину области флага при помощи мыши, удерживая нажатой левую кнопку. Сохраните приложение в редакторе приложений для автоматического создания в редакторе M-файлов заготовок для подфункций обработки события добавленных объектов.

Осталось сделать так, чтобы при нажатии пользователем кнопки Построить происходило отображение линий сетки в зависимости от установленных флагов, а нажатие на Очистить приводило к скрытию сетки. Блок обработки события Callback кнопки Построить следует дополнить проверкой состояния флагов. Свойство флага value принимает значение логической единицы при включении флага пользователем, и, соответственно, равно нулю, если флаг выключен. Указатели на флаги содержатся в полях chbxGridX и chbxGridY структуры handles. Состояние флагов определяет значение свойств XGrid и YGrid осей.

Произведите необходимые изменения в подфункции обработки события Callback кнопки Построить с именем btnPlot (листинг 5).

Листинг 5. Обработка события кнопки btnPlot с учетом состояния флагов.

```
    function btnPlot_Callback(hObject, eventdata,
handles)
    % Построение графика функции
    x=[-2:0.2:2];
    y=exp(-x.^2);
    plot (x,y)
    %Проверка флага сетка по x
    if get(handles.chbxGridX, 'Value')
    % Флаг включен, следует добавить линии сетки
    set(gca, 'XGrid', 'on')
    else
    % Флаг выключен, следует убрать линии сетки
    set(gca, 'XGrid', 'off')
    end
    % Проверка флага сетка по x
    if get(handles.chbxGridY, 'Value')
    % Флаг включен, следует добавить линии сетки
    set(gca, 'YGrid', 'on')
    else
    % Флаг выключен, следует убрать линии сетки
    set(gca, 'YGrid', 'off')
    end
    % Кнопка Построить должна стать недоступной
после вывода графика
    set(hObject, 'Enable', 'off')
    % Кнопка Очистить должна стать доступной
    set(handles.btnClear, 'Enable', 'on')
```

Флаги предоставляют пользователю возможность выбора одной или сразу нескольких опций. Одновременный выбор только одной опции осуществляется при помощи переключателей.

Переключатели

Переключатели обычно группируются по их предназначению, и пользователь может выбрать только одну опцию. Всегда установлен единственный переключатель из группы. Обработка событий переключателя должна влиять на состояние остальных переключателей всей группы. Модернизируйте интерфейс приложения mygui, предоставьте пользователю возможность выбирать тип маркера (кружок, квадрат или отсутствие маркера).

Добавьте в окно приложения новую рамку и нанесите на нее три переключателя, установите свойствам Tag значения rbMarkcirc, rbMarkSq, rbMarkNone, а String - маркеры-круги, маркеры-квадраты, без маркеров соответственно (рисунок 7).

Состояние переключателя определяется его свойством value: если value равно единице, то переключатель включен, ноль - нет. Задайте в редакторе свойств значение 1 свойству value переключателя с надписью без маркеров, он будет включен при запуске программы. Значение свойства Value в версии MATLAB 6.x устанавливается следующим образом. Выделите переключатель и перейдите к его свойствам. В редакторе свойств нажмите кнопку в строке с Value. Появляется окно Value, изображенное на рисунке 8.

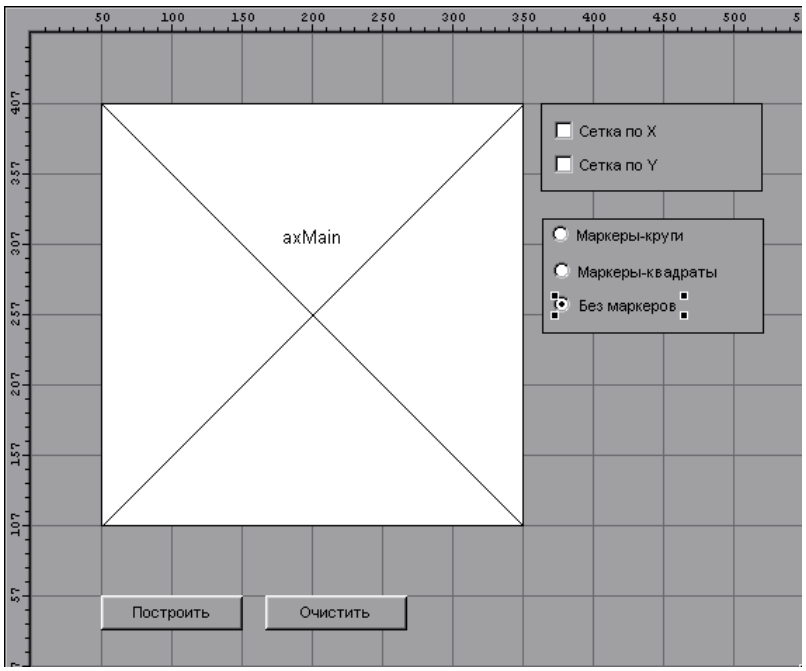

Рисунок 7 - Добавление группы переключателей

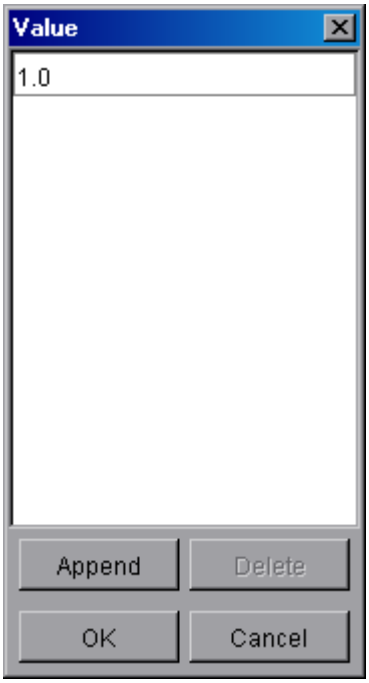

Рисунок 8 - Окно Value для установки значения

Выделите при помощи мыши строку со значением 0.0 и перейдите в режим редактирования значения двойным щелчком мыши. Измените 0.0 на единицу и нажмите ОК. Обратите внимание, что в редакторе свойств значение Value изменилось на единицу, и включился переключатель без маркеров на окне приложения в редакторе приложений. Вышеописанным образом устанавливаются значения Value в редакторе свойств. Дальнейшее управление значением Value переключателей должно осуществляться программно в ходе работы приложения mygui.

Листинг 6. Выбор типа маркеров.

```
    function btnPlot_Callback(hObject, eventdata,
handles)
    % Построение графика функции
    x=[-2:0.2:2];
    y=exp(-x.^2);
    handles.line = plot (x,y)
    guidata(gcbo,handles)
    function          rbMarkcirc_Callback(hObject,
eventdata, handles)
    % Устанавливаем маркеры-круги
```

```
    set(handles.line, 'Marker', 'o')
    % Переключатель Маркеры-квадраты должен быть
выключен
    set(handles.rbMarkSq, 'Value',0)
    % Переключатель Без маркеров должен быть
выключен
    set(handles.rbMarkNone, 'Value',0)
    function rbMarkSq_Callback(hObject,
eventdata, handles)
    % Устанавливаем маркеры-квадраты
    set(handles.line,'Marker','s')
    % Переключатель Маркеры-круги должен быть
выключен
    set(handles.rbMarkcirc, 'Value',0)
    % Переключатель Без маркеров должен быть
выключен
    set(handles.rbMarkNone, 'Value',0)
    function rbMarkNone_Callback(hObject,
eventdata, handles)
    % Устанавливаем оттображение графика без
маркеров
    set(handles.line,'Marker','none')
    % Переключатель Маркеры-круги должен быть
выключен
    set(handles.rbMarkcirc, 'Value',0)
    % Переключатель Без маркеров должен быть
выключен
    set(handles.rbMarkSq, 'Value', 0)
```

Списки

Модернизируйте интерфейс приложения mygui, предоставьте пользователю возможность выбора цвета линии графика из раскрывающегося списка (синий, красный, зеленый). Перейдите в режим редактирования и добавьте при помощи панели управления раскрывающийся список (рисунок 9). В редакторе свойств установите свойство Tag в значение 'pmColor'.

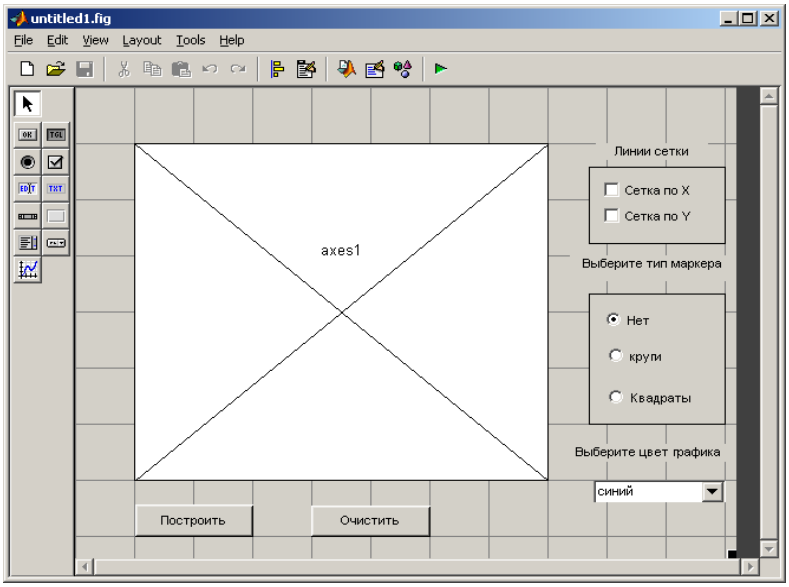

Рисунок 9 - Добавление раскрывающегося списка

Элементами раскрывающегося списка являются строки, которые вводятся в редакторе свойств. Нажмите кнопку в строке со свойством String раскрывающегося списка, появляется окно String. Наберите в нем строки "синий", "красный", "зеленый" (без кавычек), разделяя их при помощи клавиши <Enter> (рисунок 10).

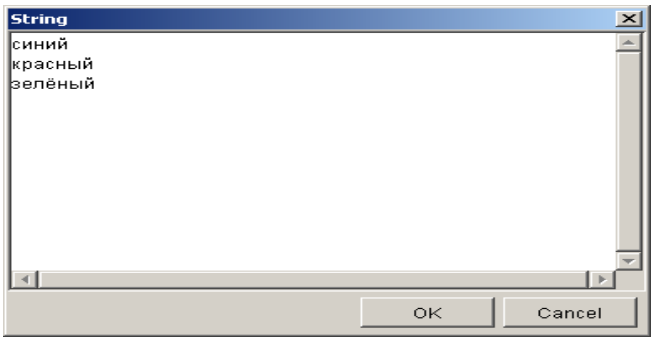

Рисунок 10 - Окно String

Запустите mygui и убедитесь, что раскрывающийся список содержит требуемые строки. Выбор различных строк пока не приводит к изменению цвета линии - требуется запрограммировать событие Callback раскрывающегося списка.

Обработка события Callback раскрывающегося списка состоит в определении выбора пользователя и соответствующем изменении цвета линии. Свойство списка value содержит номер выбранной строки (строки списка нумеруются с единицы). Перейдите к подфункции pmColor_Callback и запрограммируйте обработку выбора цвета линии

пользователем. Используйте оператор switch для установки цвета линии в зависимости от номера выбранной строки списка.

Листинг 7. Изменение цвета линии.

```
    function pmColor_Callback(hObject, eventdata,
handles)
    Num=get(hObject, 'Value');
    switch Num
     case 1
     % Устанавливаем синий цвет линии
     set (handles.line, 'Color','b');
     case 2
     % Устанавливаем красный цвет линии
     set (handles.line, 'Color','r');
     case 3
     % Устанавливаем зеленый цвет линии
     set (handles.line, 'Color','g');
    end
```

Полосы скроллинга

Усовершенствуйте интерфейс приложения mygui, предоставив пользователю возможность устанавливать ширину линии при помощи полосы скроллинга. Добавьте полосу скроллинга в окно приложения и задайте название scrWidth в свойстве Tag полосы. Снабдите полосу скроллинга текстовым пояснением "Толщина линии" так же, как и раскрывающийся список (рисунок 11).

Теперь следует определить соответствие между положением бегунка полосы и числовым значением свойства value.

Выполните следующие установки из редактора свойств.

1. В Max занесите десять, а в Min - единицу. Свойства Max и Min полосы скроллинга отвечают за границы значений, записываемых в value, при перемещении бегунка.

2. Определите начальное положение, записав в value единицу. Нажмите кнопку в строке с названием свойства и в появившемся окне Value измените значение на единицу.

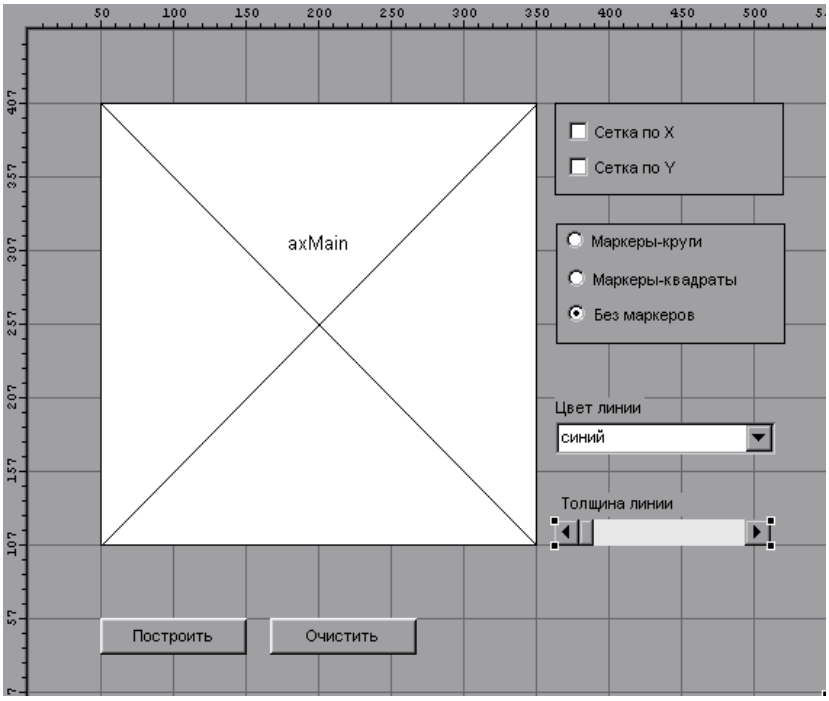

Рисунок 11 - Окно с полосой скроллинга

3. Обратитесь к свойству Sliderstep. Его значением является вектор из двух компонентов, первый из которых определяет относительное изменение value при нажатии на кнопки со стрелками полосы скроллинга, а второй - при перетаскивании бегунка мышью. Следует установить значение [0.1 0.2] свойства Sliderstep для того, чтобы нажатие на кнопки полосы изменяло value на десять процентов, а щелчок мыши справа или слева от бегунка на двадцать. Раскройте строку Sliderstep щелчком мыши по знаку плюс слева от названия свойства и в появившихся строках x и y введите 0.1 и 0.2 (рисунок 12).

Осталось запрограммировать событие Callback полосы скроллинга с именем scrWidth, которое состоит в задании ширины линии, равной округленному значению value. Перейдите к подфункции scrWidth_Callback и добавьте в ней оператор установки ширины линии

Рисунок 12 - SliderStep

Листинг 8. Изменение толщины линии.

```
    function scrWidth_Callback(hObject,
eventdata, handles)
    % Получаем текущее значение value скроллбара
    w = get(hObject,'Value');
    % Устанавливаем в качестве толщины линии
округленное значение value
set(handles.line,'LineWidth',round(w));
```

Область ввода текста

Обычные текстовые области, использовавшиеся на протяжении предыдущих разделов, позволяют лишь вывести некоторый текст в окно приложения. Обмен текстовой информацией между пользователем и приложением осуществляется при помощи областей ввода текста. Предоставьте пользователю возможность размещать заголовок на графике. Текст заголовка пользователь вводит в соответствующей строке.

Добавьте в окно приложения область ввода текста, установите значение editTitle свойству Tag области ввода и снабдите ее пояснением в текстовой области, расположенной выше так, как показано на рисунке 13. В редакторе свойств удалите из String строку Edit Text, для чего нажмите кнопку в строке с названием свойства и сотрите текст в окне String.

Листинг 9. Изменение заголовка графика.

```
    function btnPlot_Callback(hObject, eventdata,
handles)
    %
    % Операторы, отвечающие за построение графика
и т.д.
    %
    title(get(handles.editTitle,'String'))
```

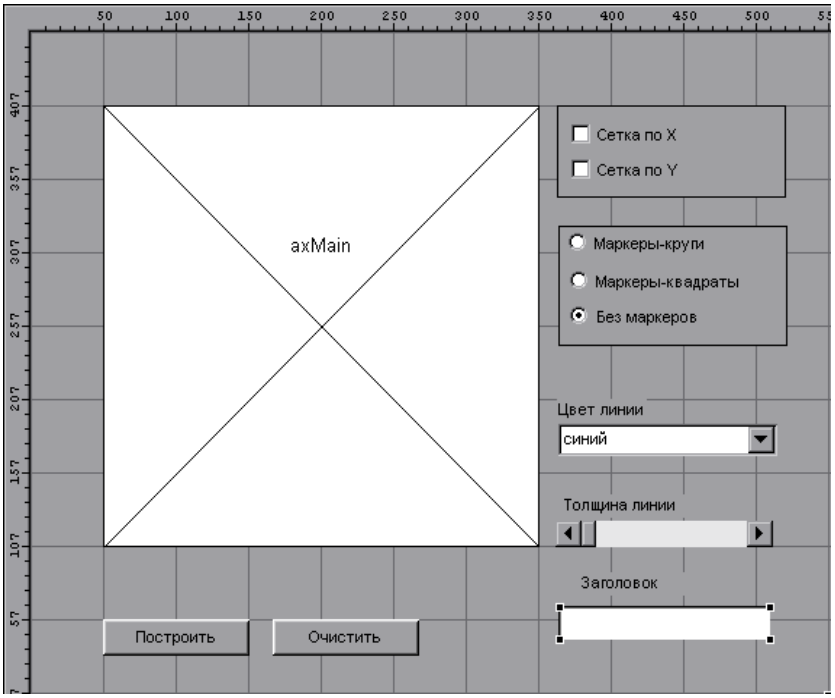

Рисунок 13 - Окно с областью ввода текста

Диалоговые окна и меню приложения

Виды диалоговых окон

Удобный интерфейс приложения во многом определяется диалоговыми окнами, облегчающими работу с файлами, или предназначенными для предостережения пользователя о событиях, которые могут повлечь его действия. MATLAB предоставляет разработчику приложения возможность использовать стандартные диалоговые окна Windows.

Окно подтверждения

Некоторые действия приложения требуют повторного подтверждения пользователя. Например, пользователь приложения mygui может случайно нажать кнопку Очистить, предназначенную для очистки осей. Следует вывести диалоговое окно, в котором пользователь укажет, действительно ли требуется очистить оси.

Диалоговое окно подтверждения создается функцией questdlg, которая в самом простом случае имеет два входных параметра - строки с текстом внутри диалогового окна и заголовком окна. Окно, создаваемое таким образом, имеет три кнопки - Yes, No и

Cancel. Выбор пользователя возвращается в строковом выходном аргументе функции questdlg, его значение совпадает с надписью на кнопке.

Усовершенствуйте обработку нажатия кнопки Очистить так, чтобы соответствующие операторы выполнялись только в том случае, если пользователь нажал кнопку Yes в появляющемся диалоговом окне с текстом Очистить оси? и заголовком mygui. Используйте условный оператор if и функцию strcmp для сравнения выходного аргумента questdlg со строкой Yes (листинг 6).

Листинг 10. Программирование диалогового окна запроса.

```
button = questdlg('Очистить оси?','mygui');
if strcmp(button, 'Yes')
% здесь размещаются все операторы,
% обрабатывающие нажатие на кнопку Очистить
end
```

Нажатие на кнопку Очистить приводит к появлению диалогового окна, изображенного на рисунке 14. Выбор пользователя определяет дальнейшие действия приложения mygui.

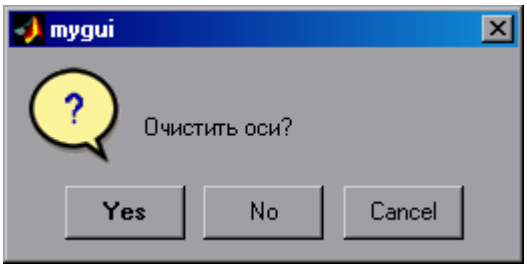

Рисунок 14 - Диалоговое окно подтверждения

Функция questdlg позволяет управлять видом диалогового окна. Строка с названием кнопки, переданная в третьем дополнительном аргументе, определяет кнопку окна, которая может быть нажата пользователем при помощи клавиши <Enter>. Например, вызов

```
Button = questdlg('Очистить оси?','mygui',
'Yes');
```

предполагает, что в диалоговом окне нажатие клавиши <Enter> эквивалентно выбору кнопки Yes.

Число кнопок и надписи на них определяются создателем приложения, например, следующая форма обращения к функции questdlg

```
    Button     =     questdlg('Очистить     оси?',
'mygui','Да','Нет','Нет')
```

приводит к появлению диалогового окна с текстом Очистить оси?, заголовком mygui и двумя кнопками Да и Нет, причем нажатие <Enter> заменяет выбор Нет.

Окно с сообщением об ошибке

Некоторые действия пользователя, в частности открытие файла с данными в неизвестном формате, могут привести к ошибке в работе приложения. Такие исключительные ситуации следует предусматривать при написании алгоритма приложения и сопровождать их сообщением об ошибке. Лучше всего выводить сообщение в диалоговое окно, которое автоматически размещается поверх всех остальных окон и требует нажатия кнопки ОК для продолжения работы.

Функция errordlg предназначена для создания диалогового окна с сообщением об ошибке. Входными аргументами errordlg являются строки с текстом и заголовком окна.

Дополните построение графика данных проверкой на размерность и тип содержимого массива Mas при помощи функций size, ndims и isnumeric и выведите сообщение в случае несоответствующего формата данных. Заключите считывание и визуализацию данных в блок try...catch end для предотвращения ошибки при обращении к load (листинг 7). При работе в MATLAB 6.x следует записать указатель на линию в структуру handles.line и сохранить, используя функцию guidata.

Листинг 11. Обработка исключительных ситуаций с сообщением об ошибке.

```
    try
    % Считывание данных из файла в массив
    Mas = load('data.txt');
    % Определение размеров массива
    SMas = size(Mas);
    % Проверка массива данных
    if ((SMas(2) ~= 2) | (ndims(Mas) ~= 2) |
~isnumeric(Mas))
    errordlg('Неизвестный формат файла с
данными', 'Ошибка!')
    else
    % Графическое отображение данных
    headers.line = plot(Mas(:,1), Mas(: , 2) ) ;
    end
    catch
    % Произошла ошибка при выполнении load
    errordlg('Неизвестный формат файла с
данными', 'Ошибка!')
    end
```

Меню графического окна

Приложение MATLAB может использовать стандартное меню графического окна. Среда GUIDE позволяет программисту дополнять стандартное меню или создать собственные меню. Свойство MenuBar окна приложения (объекта figure) отвечает за наличие стандартных меню File, Edit, Tools, Window и Help в работающем приложении. Значение figure данного свойства соответствует отображению стандартных меню, a none приводит к приложению без строки с меню. Вне зависимости от значения свойства MenuBar, разработчик приложения имеет возможность размещать собственные меню, которые в случае значения figure добавляются к стандартным меню графического окна. Размещение и программирование меню производится при помощи редактора меню.

Создание меню в редакторе

Перейдите в режим редактирования приложения в среде GUIDE. Принцип конструирования меню проще всего понять, создавая новое меню - убедитесь, что свойство MenuBar графического окна установлено в none. Запустите редактор меню из панели управления (или выбором пункта меню Tools->Menu Editor…), появляется окно Menu Editor, изображенное на рисунке 15.

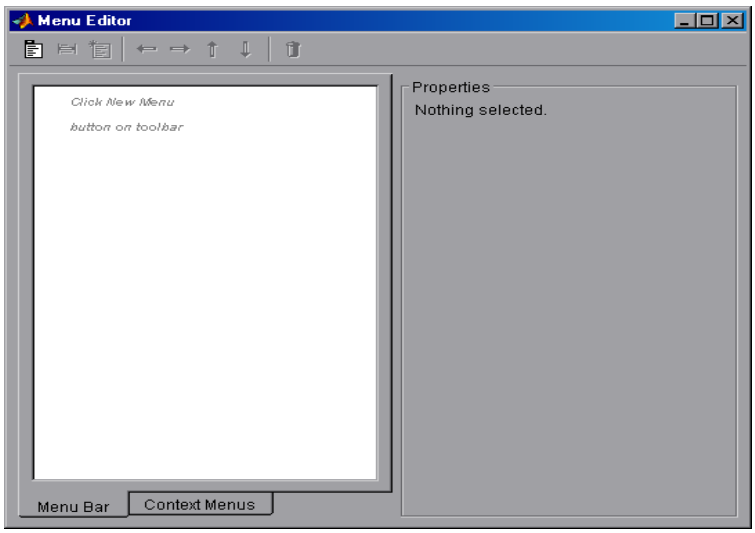

Рисунок 15 - Редактор меню Guide Menu Editor

Окно редактора меню содержит две вкладки: Menu Bar, предназначенную для создания строки меню приложения, и Context Menus для контекстного меню. Области навигатора и свойств элементов меню пока пустые. Создайте меню, нажав соответствующую кнопку на панели инструментов редактора меню (убедитесь, что выбрана вкладка Menu Bar), в навигаторе появилась строка Untitled 1, сделайте ее текущей щелчком мыши. В области свойств находятся строки ввода (рисунок 16).

Строка Label служит для задания надписи меню или пункта меню, а Tag -для определения названия созданного объекта. Введите текст "График" в строку Label (без кавычек) и задайте имя mnGraph. Запустите приложение mygui и убедитесь в наличии меню График. Выбор меню График в работающем приложении не приводит к раскрытию меню, следует создать пункты меню.

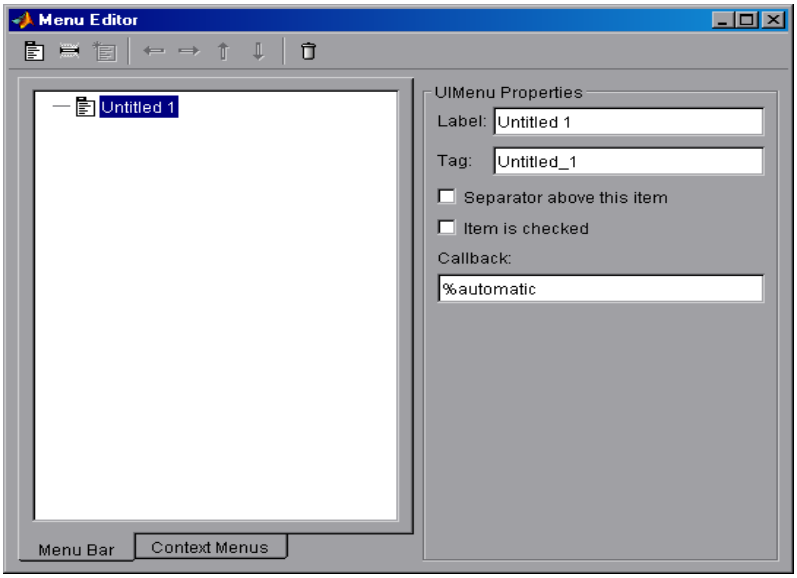

Рисунок 16 - Задание свойств меню в редакторе

Перейдите в режим редактирования, сделайте текущей строку График в навигаторе редактора меню и добавьте пункт, нажав соответствующую кнопку на панели инструментов редактора меню. Установите надпись пункта Построить и дайте ему имя mnGraphPlot. Добавьте еще один пункт меню, сделав предварительно текущей строку График в навигаторе. Аналогичным образом задайте надпись Очистить и имя mnGraphClear. Навигатор меню должен содержать структуру, изображенную на рисунке 17. Меню График имеет первый уровень, а пункты Построить, Очистить - второй.

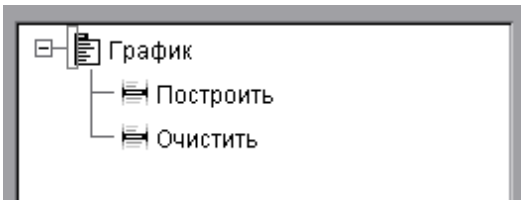

Рисунок 17 - Иерархия элементов меню

Запустите приложение mygui. Выбор меню График приводит к раскрытию меню. Пока при обращении к пунктам Построить и Очистить ничего не происходит, следует запрограммировать события Callback пунктов меню. Событие Callback самого меню График не требует обработки, т.к. происходит автоматическое раскрытие меню.

Программирование пунктов меню

Выбор элемента меню в навигаторе редактора меню приводит к отображению его свойств на панели Properties. Строка ввода Callback предназначена для вызова подфункции М-файла приложения, содержащего обработку событии элементов интерфейса.

В соответствующем приложению М-файле автоматически создаются подфункции обработки Callback созданных элементов меню. Программисту, как и в случае создания других элементов управления, следует наполнить их содержимым - операторами, производящими требуемые действия.

Откройте файл mygui.m в редакторе М-файлов и определите две подфункции для созданных элементов меню.

Листинг 12. Программирование пунктов меню.

```
    function mnGraphPlot_Callback(hObject,
eventdata, handles)
    % Вызываем обработчик нажатия кнопки
Построить
    btnPlot_Callback(hObject, eventdata, handles)
    % Кнопка Построить должна быть недоступна
    set(handles.btnPlot,'Enable','off')
    % Кнопка Очистить должна быть доступна
    set(handles.btnClear,'Enable','on')
    function mnGraphClear_Callback(hObject,
eventdata, handles)
    % Вызываем обработчик нажатия кнопки Очистить
    btnClear_Callback(hObject, eventdata,
handles)
    % Кнопка Построить должна быть доступна
    set(handles.btnPlot,'Enable','on')
    % Кнопка Очистить должна быть недоступна
    set(handles.btnClear,'Enable','off')
```

Контекстное меню объектов

Объекты, в том числе и созданные в ходе работы приложения, могут иметь собственное контекстное меню, которое активизируется щелчком левой кнопки мыши. Контекстное меню позволяет получить быстрый доступ к часто используемым свойствам объекта. Конструирование контекстного меню состоит в создании его в редакторе меню, определении событий Callback пунктов меню и последующем связывании меню с объектом.

Создание меню

Перейдите к вкладке Context Menus в редакторе меню и нажмите кнопку создания контекстного меню (рисунок 18), в навигаторе меню появляется строка для меню. Задайте ему имя cmLine. Обратите внимание, что на панели свойств нет строки ввода Label, т. к. раскрывающееся меню не должно иметь надписи. Создайте три пункта меню при помощи той же кнопки, что применяется для добавления пунктов меню окна приложения. Определите для них надписи синий, красный, зеленый и имена cmLineBlue, cmLineRed, cmLineGreen соответственно. В результате навигатор меню должен содержать структуру, приведенную на рисунке 18.

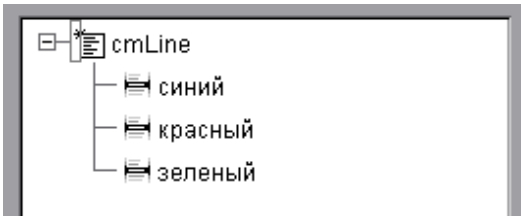

Рисунок 18 - Отображение контекстного меню в навигаторе объектов

В работающем приложении щелчок правой кнопкой мыши по линии графика не приводит к отображению контекстного меню. Сейчас контекстное меню cmLine присутствует в приложении как объект, но другой объект - линия, создаваемая при нажатии, например, на кнопку Построить, "не знает" о том, что у нее есть собственное контекстное меню. Следующий этап состоит в связывании линии с созданным меню cmLine.

Связывание контекстного меню с объектом

Любой объект, размещенный в окне приложения, имеет свойство UIContextMenu, значением которого может являться указатель на имеющееся контекстное меню. Для того чтобы созданный объект, т. е. линия графика, обладал контекстным меню, следует установить свойству UIContextMenu значение указателя на меню cmLine, содержащееся в структуре handles. Построение линии в приложении mygui производится или при нажатии пользователем кнопки Построить, или при выборе пункта Построить меню График. Присвойте свойству линии UIContextMenu требуемое значение во всех блоках М-файла, связанного с mygui, которые отвечают за отображение графика функции (листинг 8).

Листинг 13. Связывание контекстного меню с линией.

```
    handles.line = plot (x, y);
    guidata(gcbo, handles);
    % Связывание контекстного меню cmLine с
линией графика
    set(handles.line,            'UIContextMenu',
handles.cmLine)
```

Запустите приложение mygui, постройте линию любым из доступных способов и убедитесь, что щелчок правой кнопкой мыши по линии приводит к появлению контекстного меню с пунктами синий, красный, зеленый. Выбор пунктов не приводит к изменению цвета линии, очевидно, что следует запрограммировать событие Callback каждого пункта.

Программирование контекстного меню

Обработка событий Callback пунктов контекстного меню производится аналогично программированию меню приложения. Установите в редакторе меню для пунктов cmLineBlue, cmLineRed, cmLineGreen соответственно вызовы:

```
mygui('cmLineBlue_Callback',gcbo,[ ],guidata(gcbo))
mygui('cmLineRed_Callback',gcbo,[],guidata(gcbo))
mygui('cmLineGreen_Callback',gcbo,[],guidata(gcbo))
```

а в файле mygui.m опишите данные подфункции в соответствии с листингом 14.

Листинг 14. Программирование контекстного меню.

```
    function          cmLineBlue_Callback(hObject,
eventdata, handles)
    %Пользователь выбрал синий цвет линии в
контекстном меню
    set(handles.line,'Color', 'b')
    function          cmLineRed_Callback(hObject,
eventdata, handles)
    % Пользователь выбрал красный цвет линии в
контекстном меню
    set(handles.line,'Color', 'r')
    function        cmLineGreen_Callback(hObject,
eventdata, handles)
    % Пользователь выбрал зеленый цвет линии в
контекстном меню
    set(handles.line,'Color', 'g')
```

Запрограммированное и связанное с линией контекстное меню разрешает быстрый доступ пользователя к цвету линии. Осталось обеспечить согласованную работу контекстного меню со списком Цвет линии с именем pmColor. Выбор цвета из меню должен приводить не только к изменению цвета линии, но и к появлению соответствующей строки в раскрывающемся списке. В каждую подфункцию обработки события Callback пункта контекстного меню следует добавить операторы, устанавливающие нужное значение (1, 2 или 3) свойства value раскрывающегося списка (листинг 15).

Листинг 15. Согласованная работа меню и списка выбора цвета.

```
    function cmLineBlue_Callback(hObject,
eventdata, handles)
    %Пользователь выбрал синий цвет линии в
контекстном меню
    set(handles.line,'Color', 'b')
    set(handles.pmColor, 'Value', 1)
    function cmLineRed_Callback(hObject,
eventdata, handles)
    % Пользователь выбрал красный цвет линии в
контекстном меню
    set(handles.line,'Color', 'r')
    set(handles.pmColor, 'Value', 2)
    function cmLineGreen_Callback(hObject,
eventdata, handles)
    % Пользователь выбрал зеленый цвет линии в
контекстном меню
    set(handles.line,'Color', 'g')
    set(handles.pmColor, 'Value', 3)
```

Задание

Необходимо при помощи пакета MATLAB создать приложение, окно которого содержит следующие элементы: оси, три кнопки, два поля ввода текста.

В одно из текстовых полей вводится функция, в другое – количество членов, которые нужно получить из разложения данной функции в ряд Тейлора. Нажатие на одну кнопку позволяет получить график исходной функции. Нажатие на вторую кнопку позволяет получить график суммы заданного количества членов ряда Тейлора и добавить его к ранее построенным графикам. Графики должны иметь разный цвет. Нажатие на третью кнопку приводит к очистке окна, в которое выводятся графики.

Приложение

Листинг программы

```
    function varargout = mygui(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name', mfilename, ...
     'gui_Singleton', gui_Singleton, ...
     'gui_OpeningFcn', @mygui_OpeningFcn, ...
     'gui_OutputFcn', @mygui_OutputFcn, ...
     'gui_LayoutFcn', [] , ...
     'gui_Callback', []);
    if nargin & isstr(varargin{1})
     gui_State.gui_Callback                    =
str2func(varargin{1});
    end
    if nargout
     [varargout{1:nargout}]                    =
gui_mainfcn(gui_State, varargin{:});
    else
     gui_mainfcn(gui_State, varargin{:});
    end
    % --------------------------------------
    function mygui_OpeningFcn(hObject, eventdata,
handles, varargin)
    handles.output = hObject;
    guidata(hObject, handles);
    % --------------------------------------
    function varargout = mygui_OutputFcn(hObject,
eventdata, handles)
    varargout{1} = handles.output;
    % --------------------------------------
    function btnPlot_Callback(hObject, eventdata,
handles)
    % Выводим диалоговое окно для выбора выриантa
получения исходных данных
    button    =    questdlg('Данные','mygui','По
умолчанию','Из файла','По умолчанию');
```

```
    % Выбран вариант "Из файла"
    if strcmp(button, 'Из файла')
     try
     % Считывание данных из файла в массив
     Mas = load('data.txt');
     % Определение размеров массива
     SMas = size(Mas);
     % Проверка массива данных
     if ((SMas(2) ~= 2) | (ndims(Mas) ~= 2) |
~isnumeric(Mas))
     errordlg('Неизвестный   формат   файла   с
данными', 'Ошибка!')
     else
     % Графическое отображение данных
     handles.line = plot(Mas(:,1),Mas(:,2));
     guidata(gcbo, handles);
     set(handles.line,           'UIContextMenu',
handles.cmLine);
     end
     catch
     % Произошла ошибка при выполнении load
     errordlg('Неизвестный   формат   файла   с
данными', 'Ошибка!')
     end

    else
     % Выбран выриант "По умолчанию"
     % Построение графика функции
     x=[-2:0.2:2];
     y=exp(-x.^2);
     handles.line = plot (x,y);
     guidata(gcbo,handles);
     set(handles.line,           'UIContextMenu',
handles.cmLine)
     %Проверка флага сетка по x
     if get(handles.chbxGridX, 'Value')
     % Флаг включен, следует добавить линии сетки
     set(gca, 'XGrid', 'on')
```

```
     else
     % Флаг выключен, следует убрать линии сетки
     set(gca, 'XGrid', 'off')
     end
     % Проверка флага сетка по x
     if get(handles.chbxGridY, 'Value')
     % Флаг включен, следует добавить линии сетки
     set(gca, 'YGrid', 'on')
     else
     % Флаг выключен, следует убрать линии сетки
     set(gca, 'YGrid', 'off')
     end
    end
    % Кнопка Построить должна стать недоступной
после вывода графика
    set(hObject, 'Enable', 'off')
    % Кнопка Очистить должна стать доступной
    set(handles.btnClear, 'Enable', 'on')
    % Пункт меню График->Построить должен стать
недоступным
    set(handles.mnGraphPlot,'Enable','off')
    % Пункт меню График->Очистить должен стать
доступным
    set(handles.mnGraphClear,'Enable','on')
    % Переключатель Маркеры-круги должен стать
доступным
    set(handles.rbMarkcirc,'Enable','on')
    % Переключатель Маркеры-квадраты должен стать
доступным
    set(handles.rbMarkSq,'Enable','on')
    % Переключатель Без маркеров должен стать
доступным
    set(handles.rbMarkNone,'Enable','on')
    % Список Цвет линии должен стать доступным
    set(handles.pmColor,'Enable','on')
    % Скроллбар Толщина линии должен стать
доступным
    set(handles.scrWidth,'Enable','on')
```

```
    % Заголовок графика из текстового поля
    title(get(handles.editTitle,'String'))
    % --------------------------------------
    function           btnClear_Callback(hObject,
eventdata, handles)
    % Выводим диалоговое окно
    button = questdlg('Очистить оси?','mygui');
    % Если выбрано "Yes"
    if strcmp(button, 'Yes')
    % очистка осей
    cla
    % Кнопка Очистить должна стать недоступной
после очистки осей
    set(hObject, 'Enable', 'off')
    % Кнопка Построить должна стать доступной
    set(handles.btnPlot, 'Enable', 'on')
    % Убираем сетку
    set(gca, 'XGrid', 'off')
    set(gca, 'YGrid', 'off')
    % Переключатель Маркеры-круги должен стать
недоступным
    set(handles.rbMarkcirc,'Enable','off')
    % Переключатель Маркеры-квадраты должен стать
недоступным
    set(handles.rbMarkSq,'Enable','off')
    % Переключатель Без маркеров должен стать
недоступным
    set(handles.rbMarkNone,'Enable','off')
    % Список Цвет линии должен стать недоступным
    set(handles.pmColor,'Enable','off')
    % Скроллбар Толщина линии должен стать
недоступным
    set(handles.scrWidth,'Enable','off')
    % Убираем заголовок графика
    title('')
    % Пункт меню График->Построить должен стать
доступным
    set(handles.mnGraphPlot,'Enable','on')
```

```
    % Пункт меню График->Очистить должен стать
недоступным
    set(handles.mnGraphClear,'Enable','off')
    end
    % --------------------------------------
    function          chbxGridx_Callback(hObject,
eventdata, handles)
    % --------------------------------------
    function          chbxGridY_Callback(hObject,
eventdata, handles)
    % --------------------------------------
    function          chbxGridX_Callback(hObject,
eventdata, handles)
    % --------------------------------------
    function         rbMarkcirc_Callback(hObject,
eventdata, handles)
    % Устанавливаем маркеры-круги
    set(handles.line, 'Marker', 'o')
    % Переключатель Маркеры-квадраты должен быть
выключен
    set(handles.rbMarkSq, 'Value',0)
    % Переключатель Без маркеров должен быть
выключен
    set(handles.rbMarkNone, 'Value',0)
    % --------------------------------------
    function          rbMarkSq_Callback(hObject,
eventdata, handles)
    % Устанавливаем маркеры-квадраты
    set(handles.line,'Marker','s')
    % Переключатель Маркеры-круги должен быть
выключен
    set(handles.rbMarkcirc, 'Value',0)
    % Переключатель Без маркеров должен быть
выключен
    set(handles.rbMarkNone, 'Value',0)
    % --------------------------------------
    function         rbMarkNone_Callback(hObject,
eventdata, handles)
```

```
    % Устанавливаем оттображение графика без
маркеров
    set(handles.line,'Marker','none')
    % Переключатель Маркеры-круги должен быть
выключен
    set(handles.rbMarkcirc, 'Value',0)
    % Переключатель Без маркеров должен быть
выключен
    set(handles.rbMarkSq, 'Value', 0)
    % --------------------------------------
    function           pmColor_CreateFcn(hObject,
eventdata, handles)
    if ispc
     set(hObject,'BackgroundColor','white');
    else

set(hObject,'BackgroundColor',get(0,'defaultUico
ntrolBackgroundColor'));
    end
    % --------------------------------------
    function pmColor_Callback(hObject, eventdata,
handles)
    Num=get(hObject, 'Value');
    switch Num
     case 1
     % Устанавливаем синий цвет линии
     set (handles.line, 'Color','b');
     case 2
     % Устанавливаем красный цвет линии
     set (handles.line, 'Color','r');
     case 3
     % Устанавливаем зеленый цвет линии
     set (handles.line, 'Color','g');
    end
    % --------------------------------------
    function          scrWidth_CreateFcn(hObject,
eventdata, handles)
    usewhitebg = 1;
```

```
    if usewhitebg
     set(hObject,'BackgroundColor',[.9 .9 .9]);
    else

set(hObject,'BackgroundColor',get(0,'defaultUico
ntrolBackgroundColor'));
    end
    % --------------------------------------------
    function           scrWidth_Callback(hObject,
eventdata, handles)
    % Получаем текущее значение скроллбара
    w = get(hObject,'Value');
    % Устанавливаем в качестве толщины линее
округленное значение скроллбара
    set(handles.line,'LineWidth',round(w));
    % --------------------------------------------
    function         editTitle_CreateFcn(hObject,
eventdata, handles)
    if ispc
     set(hObject,'BackgroundColor','white');
    else

set(hObject,'BackgroundColor',get(0,'defaultUico
ntrolBackgroundColor'));
    end
    % --------------------------------------------
    function          editTitle_Callback(hObject,
eventdata, handles)
    % --------------------------------------------
    function mnGraph_Callback(hObject, eventdata,
handles)
    % --------------------------------------------
    function        mnGraphPlot_Callback(hObject,
eventdata, handles)
    % Вызываем обработчик нажатия кнопки
Построить
    btnPlot_Callback(hObject, eventdata, handles)
    % Кнопка Построить должна быть недоступна
```

```
    set(handles.btnPlot,'Enable','off')
    % Кнопка Очистить должна быть доступна
    set(handles.btnClear,'Enable','on')
    % ----------------------------------------
    function mnGraphClear_Callback(hObject,
eventdata, handles)
    % Вызываем обработчик нажатия кнопки Очистить
    btnClear_Callback(hObject, eventdata,
handles)
    % Кнопка Построить должна быть доступна
    set(handles.btnPlot,'Enable','on')
    % Кнопка Очистить должна быть недоступна
    set(handles.btnClear,'Enable','off')
    % ----------------------------------------
    function cmLine_Callback(hObject, eventdata,
handles)
    % ----------------------------------------
    function cmLineBlue_Callback(hObject,
eventdata, handles)
    % Пользователь выбрал синий цвет линии в
контекстном меню
    % Устанавливаем синий цвет линии
    set(handles.line,'Color', 'b')
    % Изменяем значение списка Цвет линии на
"синий"
    set(handles.pmColor, 'Value', 1)
    % ----------------------------------------
    function cmLineRed_Callback(hObject,
eventdata, handles)
    % Пользователь выбрал красный цвет линии в
контекстном меню
    % Устанавливаем красный цвет линии
    set(handles.line,'Color', 'r')
    % Изменяем значение списка Цвет линии на
"красный"
    set(handles.pmColor, 'Value', 2)
    % ----------------------------------------
```

```
    function        cmLineGreen_Callback(hObject,
eventdata, handles)
    % Пользователь выбрал зеленый цвет линии в
контекстном меню
    % Устанавливаем зеленый цвет линии
    set(handles.line,'Color', 'g')
    % Изменяем значение списка Цвет линии на
"зеленый"
    set(handles.pmColor, 'Value', 3)
```

Библиографический список

1. Ануфриев, И.Е. Самоучитель MATLAB 5.3/6.x / И.Е. Ануфриев. – СПб.: БХВ-Петербург, 2002. - 736 с. - ISBN 5-94157-107-0. - Текст: непосредственный.
2. Ануфриев, И.Е. MATLAB 7 / И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова. – СПб.: БХВ-Петербург, 2005. – 1104 с. - ISBN 5-94157-494-0. - Текст: непосредственный.
3. Дьяконов, В.П. MATLAB 6.5 SPI/7 + Simulink 5/6 в математике и моделировании. Серия «Библиотека профессионала» / В.П. Дьяконов. – М.: СОЛОН-Пресс, 2019. – 576 с. - ISBN 5-98003-209-6. - Текст: непосредственный.