МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Брянский государственный технический университет

Утверждаю
Ректор университета
_____А.В. Лагерев
“__”_____ 2011 г.

ИНФОРМАТИКА

«Среда GUIDE для создания приложений с графическим интерфейсом пользователя»

Методические указания к выполнению лабораторной работы № 4 для студентов очной формы обучения специальностей
140400 – "Электроэнергетика и электроника "
210100 – "Электроника и наноэлектроника"
210400 – "Радиотехника"

БРЯНСК 2011

УДК 519.682(076)

Информатика. **«Среда GUIDE для создания приложений с графическим интерфейсом пользователя»** [текст]+[электронный ресурс]: методические указания к выполнению лабораторной работы № 4 для студентов очной формы обучения специальностей 180400-«Электропривод и автоматика промышленных установок и технологических комплексов»; 210106-«Промышленная электроника»; 20010-«Микроэлектроника и твердотельная электроника»; 21030-«Радиоэлектронные системы». – Брянск: БГТУ, 2011 – 20 с.

Разработал:
В.В.Симкин
доцент, канд. техн. наук

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол № 7 от 03. 2011 г.)

ЧАСТЬ 1

Обработка события Callback

При программировании в MATLAB приложений с графическим интерфейсом пользователя возникает вопрос об обработке событий, последовательно возникающих при выборе пользователем того или иного элемента интерфейса приложения. Напомним, что при нажатии на кнопку, завершении ввода текста в строку ввода нажатием на <Enter>, установке или сбросе флага и т.д. возникает событие Callback соответствующего элемента интерфейса. Для обработки данного события требуется запрограммировать соответствующую функцию и связать ее с событием Callback элемента управления. В подразделах этого раздела мы рассмотрим несколько типичных ситуаций, возникающих при обработке события Callback элементов управления.

В данном разделе приведены примеры, в которых приложения с графическим интерфейсом пользователя создаются как с использованием среды визуального программирования GUIDE пакета MATLAB, так и без нее, когда все приложение с графическим интерфейсом самостоятельно программируется в функции, содержащей подфункции обработки событий элементов интерфейса. Описание способов создания приложений с графическим интерфейсом сопровождается необходимыми комментариями, касающимися программирования приложений с графическим интерфейсом пользователя.

Дополнительную информацию о разработке приложений с графическим интерфейсом пользователя можно почерпнуть в разделах:

- Приложения с GUI
- Новшества для создания приложений с графическим интерфейсом пользователя в 7-ой версии MATLAB

в справочной системе MATLAB в разделах:

- MATLAB: Creating Graphical User Interfaces;
- MATLAB: Functions -- Categorical List: Creating Graphical User Interfaces;
- MATLAB: Handle Graphics Property Browser;

а также на сайте www.mathworks.com.

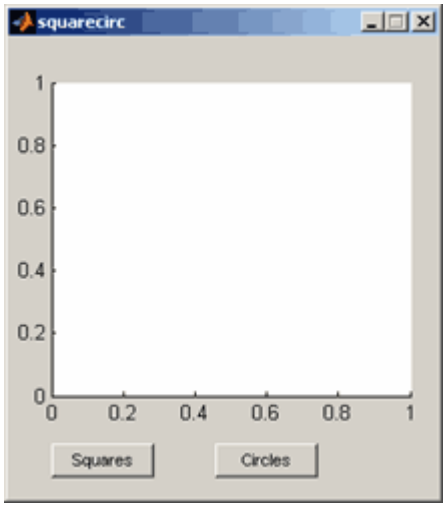
Разрешение на прерывание события, постановка событий в очередь

В подразделе Работа над приложением squarecirc в среде GUIDE мы разберем, как решается этот вопрос при создании приложения в среде визуального программирования GUIDE. А в подразделе Создание приложения squarecirc без среды GUIDE приведена функция, создающая окно приложения, с подфункциями обработки событий и указано, какие свойства элементов управления отвечают за обработку событий и как ими пользоваться при программировании приложения с графическим интерфейсом пользователя. Эти два подраздела, в принципе, можно читать независимо. В данном случае несущественно, как именно было создано приложение с графическим интерфейсом пользователя - важно понимать, как пользоваться соответствующими свойствами объектов приложения. Способ изменения их значений зависит от того, как разрабатывается приложение с графическим интерфейсом: в среде визуального

программирования GUIDE или программируется в функции с подфункциями обработки событий элементов интерфейса.

Работа над приложением squarecirc в среде GUIDE

Для начала создадим простое приложение squarecirc с графическим интерфейсом пользователя, окно которого содержит две кнопки Squares и Circles и оси. Окно приложения squarecirc приведено на рисунке ниже.



Окно приложения squarecirc

Смысл кнопок следующий:

- при нажатии на кнопку Squares на оси выводится 10 квадратов разных цветов с интервалом в 0.5 сек.
- при нажатии на кнопку Circles на оси выводится 10 кругов разных цветов с интервалом в 0.5 сек.

На примере приложения squarecirc мы разберем различные варианты обработки ситуации, в которой сразу после нажатия на кнопку Squares пользователь нажал на кнопку Circles (т.е. пока квадраты еще выводятся на оси приложения), и выясним, какие свойства элементов управления и команды MATLAB отвечают за порядок обработки событий Callback, возникающих при действиях пользователя.

Мы начнем с создания приложения в среде визуального программирования GUIDE пакета MATLAB. Для этого надо выполнить следующие действия.

1. Вызов среды визуального программирования GUIDE.

   Из командной строки выполнить команду

   ```
   >> guide
   ```

   (или нажать кнопку GUIDE на панели инструментов основного окна MATLAB).

2. Создание пустой заготовки окна приложения.

   В появившемся диалоговом окне GUIDE Quick Start на вкладке Create New GUI выбрать в списке GUIDE Templates пункт Blank GUI (Defaults) (он должен быть

выбран по умолчанию) и нажать кнопку OK. На экране отображается основное окно среды визуального программирования GUIDE MATLAB.

3. Добавление элементов интерфейса на заготовку окна приложения.

   В окне среды визуального программирования GUIDE с заголовком untitled.fig слева есть вертикальная панель инструментов, на которой расположены инструменты для нанесения на форму (заготовку окна приложения) различных элементов интерфейса. Сама заготовка окна приложения находится справа (она снабжена сеткой). При помощи этой панели инструментов нужно (см. рисунок Этапы работы в среде GUIDE Ссылка на рисунок ниже):

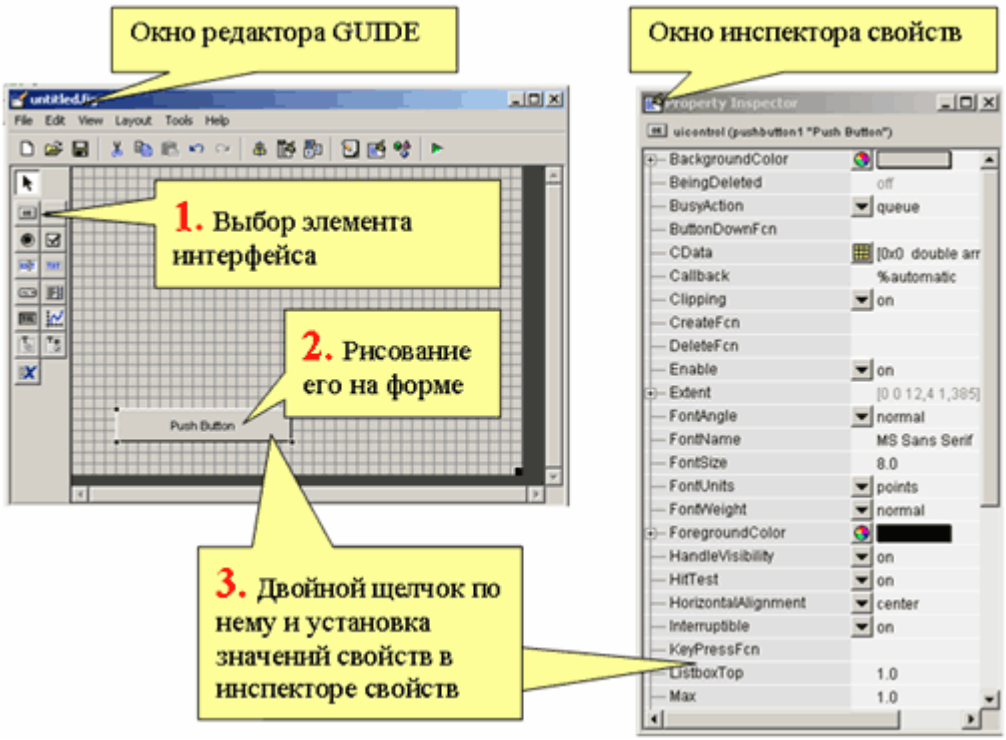
   3.1. выбрать оси (инструмент Axes) и нарисовать их на форме удерживая левую кнопку мыши;

   3.2. выбрать кнопку (инструмент Push Button) и нарисовать две кнопки под осями при помощи мыши на форме.

   Примечание.
   Размещенные на форме элементы интерфейса можно передвигать при помощи мыши (или при помощи клавиш со стрелками, предварительно выделив элемент интерфейса щелчком мыши), а также изменять их размеры, протаскивая мышью за квадратные маркеры в углах. Размер заготовки окна приложения так же можно менять, перетаскивая мышью квадратный маркер в нижнем правом углу заготовки.

4. Задание значений свойствам элементов интерфейса

   После нанесения на заготовку окна приложения необходимых элементов интерфейса следует установить некоторым их свойствам нужные значения. Задание значений производится в инспекторе свойств, окно которого Property Inspector открывается двойным щелчком мыши по элементу интерфейса на форме (см. рисунок ниже). В окне инспектора свойств размещена таблица, левый столбик которой содержит названия свойств выбранного элемента интерфейса, а рядом в правом столбике таблицы устанавливается нужное значение свойствам (вводом или выбором возможных значений из раскрывающегося списка).



Этапы работы в среде GUIDE

Мы установим значения следующим свойствам кнопок:

Для левой кнопки:

- Свойству String значение Squares;
- Свойству Tag значение btnSq.

Для правой кнопки:

- Свойству String значение Circles;
- Свойству Tag значение btnCirc.

Примечание
Значение свойства String это надпись на кнопке, а значение свойства Tag - уникальное имя (или тэг) кнопки. Тэг кнопки (да и вообще, любого объекта) нужен для обращения к свойствам объекта при программировании приложения с графическим интерфейсом пользователя. Тэг объекта также используется в среде GUIDE при генерации заголовков подфункций обработки различных событий объекта.

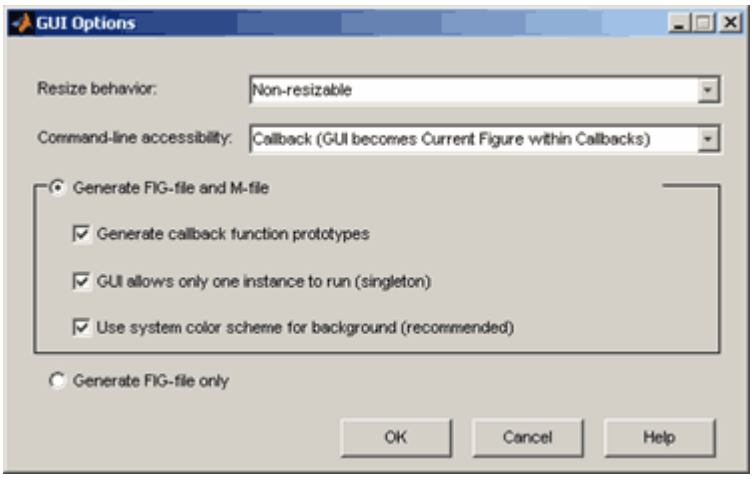
5. Сохранение приложения с графическим интерфейсом пользователя

Для сохранения нашего приложения достаточно нажать кнопку Save на горизонтальной панели инструментов под строкой меню среды GUIDE и в появившемся диалоговом окне сохранения файла задать имя squarecirc.

Примечание
По умолчанию созданное в среде визуального программирования GUIDE приложение с графическим интерфейсом пользователя хранится в двух файлах:

- файл с расширением fig (окно приложения вместе с размещенными на нем элементами интерфейса) и
- файл с расширением m (основная функция, выводящая окно приложения на экран, вместе с подфункциями обработки различных событий элементов интерфейса), этот файл создается автоматически после сохранения приложения.

Если файл с расширением m не создался, то надо в меню Tools среды GUIDE выбрать пункт GUI Options… и установить флаги и переключатели в появившемся диалоговом окне GUI Options так, как показано на рисунке ниже.



6. Программирование событий кнопок

Теперь перейдем к программированию события Callback, которое возникает при нажатии пользователем на кнопку Squares. Для программирования события Callback кнопки Squares следует в ее контекстном меню на заготовке окна приложения выбрать в пункте View Callbacks подпункт Callback. После этого в окне редактора m-файлов произойдет переход к автоматически созданному заголовку подфункции btnSq_Callback (заголовок подфункции состоит из тэга btnSq, который мы давали кнопке на 4-ом шаге, и названия события Callback, разделенных знаком подчеркивания), после которого идут три строки с комментариями, начинающиеся со знака процента:

```
function btnSq_Callback(hObject, eventdata, handles)
% hObject    handle to btnRect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

После комментариев в подфункции btnSq_Callback напишем операторы, приводящие к выводу на оси приложения десяти квадратов различных цветов (изменяющихся случайным образом) с интервалом в 0.5 секунды. Данные операторы приведены ниже с необходимыми комментариями. Перед выводом каждого следующего квадрата производится очистка осей. Для вывода квадрата используется функция rectangle, которая по умолчанию строит квадрат [0,1]x[0,1] как раз того же размера, что и оси, создаваемые в MATLAB. Графический вывод будет производиться на оси приложения, если они присутствуют в окне приложения (такие настройки для приложения с графическим интерфейсом сделаны по умолчанию в среде GUIDE).

```
% цикл по квадратам
for k=1:10
    % очистка осей
    cla
    % вывод квадрата размером с оси (т.е. 1 на 1) случайного цвета,
задаваемого в RGB
    rectangle('FaceColor',[rand(1) rand(1) rand(1)])
    % задержка на 0.5 сек.
    pause(0.5)
end
```

Аналогичным образом запрограммируем событие Callback для второй кнопки Circles, добавив в заготовку для подфункции btnCirc_Callback следующие операторы, приводящие к выводу на оси приложения десяти кругов различных цветов (изменяющихся случайным образом) с интервалом в 0.5 секунды. Эти операторы приведены ниже с соответствующими комментариями. Для вывода кругов используется та же самая функция rectangle, что и для вывода квадратов, в которой указана величина скругления углов квадрата при помощи свойства Curvature (его значение [1 1] приводит к скруглению углов квадрата как по вертикали, так и горизонтали до максимально возможного, т.е. до получения круга).

```
% цикл по кругам
for k=1:10
    % очистка осей
    cla
    % вывод круга диаметром 1 случайного цвета, задаваемого в RGB
    rectangle('FaceColor',[rand(1) rand(1) rand(1)],'Curvature',[1 1])
    % задержка на 0.5 сек.
    pause(0.5)
end
```

7. Запуск приложения

   Наше приложение squarecirc создано. Последний этап состоит в запуске приложения. Для запуска приложения следует нажать на кнопку Run на горизонтальной панели инструментов среды GUIDE, после чего появляется окно приложения, приведенное на рисунке Окно приложения squarecirc ссылка на рисунок выше.

Разберем теперь, что происходит при последовательном нажатии пользователем на кнопки Squares и Circles.

Нажатие в работающем приложении squarecirc на кнопку Squares squarecirc приводит к рисованию квадратов на осях приложения. Если не дожидаясь окончания вывода десяти квадратов нажать на кнопку Circles, то приостанавливается рисование квадратов и начинается вывод кругов. После вывода десяти кругов возобновляется вывод оставшихся квадратов. Так происходит потому, что свойство Interruptible кнопки Squares по умолчанию имеет значение 'on', т.е. событие Callback, возникшее после нажатия пользователем на кнопку Squares, может быть прервано другим событием, а именно событием Callback, возникшем после нажатия пользователем на кнопку Circles. Теперь начинает работу подфункция btnCirc_Callback, выводящая десять кругов. После завершения обработки события Callback кнопки Circles происходит возврат в точку прерывания в подфункции btnSq_Callback и продолжается выполнение события Callback кнопки Squares. Кроме этого важно, что подфункция btnSq_Callback обработки события

Callback кнопки Squares содержит функцию pause. Кроме функции pause есть и другие функций MATLAB, наличие которых в подфункции обработки одного события (свойство Interruptible которого установлено в 'on') приведет к прерыванию ее работы при возникновении другого события. Вот список этих функций:

- drawnow - обновление графического окна;
- figure - создание графического окна;
- getframe - создание массива с кадрами (для получения анимации из меняющегося содержимого осей);
- waitfor - приостановка выполнения команд до тех пор, пока некоторый объект не будет удален или заданное его свойство не изменится.

Изменим теперь значение свойства Interruptible кнопки Squares на 'off'. Для этого надо закрыть работающее приложение squarecirc и в окне инспектора свойств установить значение 'off' для свойства Interruptible кнопки Squares. После этого снова запустим приложение squarecirc. Теперь, если сразу после нажатия на кнопку Squares нажать на кнопку Circles, то сначала нарисуются все десять квадратов, а только затем начнут выводиться круги. Таким образом, событие Callback кнопки Squares не прерывается событием Callback кнопки Circles. Событие Callback кнопки Circles ставится в очередь событий и операторы подфункции btnCirc_Callback выполняются по завершении работы подфункции btnSq_Callback.

Событие Callback кнопки Circles ставится в очередь событий потому, что по умолчанию свойство BusyAction кнопки Circles имеет значение 'queue' (т.е. очередь в переводе с английского). В этом можно убедиться, перейдя к свойствам кнопки Circles в окне инспектора свойств. Снова закроем работающее приложение squarecirc, далее в окне инспектора свойств изменим значение свойства BusyAction кнопки Circles на 'cancel' и снова запустим наше приложение squarecirc. Теперь нажатие на кнопку Circles сразу после нажатия на кнопку Squares не приведет к постановке события Callback кнопки Circles в очередь событий, и когда подфункция btnSq_Callback нарисует все десять квадратов и завершит свою работу, то операторы подфункции btnCirc_Callback выполняться уже не будут и круги не будут выводиться на оси приложения.

Однако, если свойство Interruptible кнопки Squares установлено в 'on' (т.е. прерывание разрешено), то вне зависимости от значения свойства BusyAction кнопки Circles все равно произойдут следующие действия:

1. прерывание работы подфункции btnSq_Callback;
2. выполнение операторов подфункции btnCirc_Callback;
3. возврат в точку прерывания подфункции btnSq_Callback для завершения ее работы.

Примечание
Выполнение обработки события объекта может прерываться и вне зависимости от значения его свойства Interruptible. Это происходит в том случае, когда прерывающим событием является одно из следующих:

- событие, возникающее при удалении объекта, соответствующее свойство объекта DeleteFcn может содержать указатель на функцию обработки этого события;
- событие, возникающее при создании объекта, соответствующее свойство объекта CreateFcn может содержать указатель на функцию обработки этого события;

- событие, возникающее при закрытии графического окна, соответствующее свойство графического окна CloseRequest может содержать указатель на функцию обработки этого события;
- событие, возникающее при изменении размеров графического окна, соответствующее свойство графического окна ResizeFcn может содержать указатель на функцию обработки этого события;

У нашего приложения squarecirc есть один существенный недостаток. Если в момент вывода на оси квадратов (или кругов) закрыть окно приложения squarecirc, например при помощи кнопки с крестиком в правом верхнем углу на заголовке окна приложения, то создастся обычное графическое окно и оси, на которые будет продолжаться вывод квадратов (или кругов). Самый простой способ предотвращения этого состоит в проверке наличия какого-либо объекта приложения (например кнопки) перед графическим выводом.

Немного изменим подфункцию btnSq_Callback обработки события Callback кнопки Squares, использовав вместо цикла for цикл while, в условие продолжения которого внесем проверку на число нарисованных кругов и проверку на существование кнопки Squares. Проверку на существование кнопки выполним при помощи специальной функции ishandle. Ее входным аргументом должен быть указатель на объект, т.е. на кнопку Squares. Указатель на кнопку Squares содержится в первом входном аргументе hObject подфункции btnSq_Callback. Если объект с заданным указателем существует, то функция ishandle возвращает единицу, а иначе - ноль. Текст измененной подфункции btnSq_Callback обработки события Callback кнопки Squares приведен ниже.

```
function btnSq_Callback(hObject, eventdata, handles)
% задаем начальное значение счетчика квадратов
k=1;
% цикл по квадратам, проверка, что было выведено не более 10 квадратов
% и объект кнопка Squares существует
while (k<=10) & ishandle(hObject)
    % очистка осей
    cla
    % вывод квадрата размером с оси (т.е. 1 на 1) случайного цвета,
задаваемого в RGB
    rectangle('FaceColor',[rand(1) rand(1) rand(1)])
    % задержка на 0.5 сек.
    pause(0.5)
    % увеличение k на 1 для проверки при следующем входе в цикл
    k=k+1;
end
```

Аналогичным образом можно модифицировать и подфункцию btnCirc_Callback обработки события Callback кнопки Circles, добавив проверку на существование кнопки Circles. Измененная подфункция btnCirc_Callback приведена ниже.

```
function btnCirc_Callback(hObject, eventdata, handles)
% задаем начальное значение счетчика кругов
k=1;
% цикл по кругам, проверка, что было выведено не более 10 кругов
% и объект кнопка Circles существует
while (k<=10) & ishandle(hObject)
    % очистка осей
    cla
    % вывод круга диаметром 1 случайного цвета, задаваемого в RGB
    rectangle('FaceColor',[rand(1) rand(1) rand(1)],'Curvature',[1 1])
    % задержка на 0.5 сек.
```

```
    pause(0.5)
    % увеличение k на 1 для проверки при следующем входе в цикл
    k=k+1;
end
```

Создание приложения squarecirc без среды GUIDE

В этом разделе приводятся функция, создающая окно приложения squarecirc (описанное в разделе Работа над приложением squarecirc в среде GUIDE), вместе с подфункциями обработки событий элементов управления и поясняется, как задавать значения свойствам Interruptible и BusyAction для определения последовательности обработки событий, возникающих при последовательном нажатии пользователем на кнопки приложения.

В основной функции squarecirc создаются:

- окно приложения при помощи функции figure;
- оси в окне при помощи функции axes;
- две кнопки Squares и Circles при помощи функции uicontrol, нажатие на кнопку Squares приводит к выводу на оси десяти квадратов случайных цветов с интервалом в 0.5 секунды, а нажатие на кнопку Circles приводит к выводу на оси десяти кругов случайных цветов, так же с интервалом в 0.5 секунды.

С событиями Callback кнопок связываются следующие подфункции:

- для кнопки Squares - подфункция btnSq_Callback;
- для кнопки Circles - подфункция btnCirc_Callback.

Ниже приведен текст основной функции squarecirc с подфункциям обработки событий кнопок (это должен быть один m-файл squarecirc.m). В подфункциях btnSq_Callback и btnCirc_Callback при помощи функции ishandle делается проверка на наличие объекта (кнопки). Это нужно для обработки следующей ситуации. Предположим, что выводятся круги или квадраты, и в этот момент пользователь закрывает окно приложения squarecirc. Тогда объект кнопка (как и все объекты приложения) перестает существовать и вывод кругов или квадратов надо остановить. Если этого не сделать, то создастся новое графическое окно, в которое продолжится вывод.

```
function squarecirc
% вывод квадратов и кругов

% создание окна приложения без стандартных меню и панели инструментов
% с заголовком squarecirc
figure('Menubar','none', 'Position',[520 200 290 310],...
    'Name','squarecirc','NumberTitle','off')
% создание осей в графическом окне
axes('Position',[0.1 0.2 0.8 0.7])
% создание кнопки Squares и связывание с ее событием Callback
% функции btnSq_Callback
uicontrol('Style','pushbutton','Position',[30 15 70 20],...
    'String','Squares','Callback',@btnSq_Callback)
% создание кнопки Circles и связывание с ее событием Callback
% функции btnCirc_Callback
uicontrol('Style','pushbutton','Position',[140 15 70 20],...
    'String','Circles','Callback',@btnCirc_Callback)

function btnSq_Callback(hObject, eventdata)
% подфункция обработки события Callback кнопки Squares
```

```
% задаем начальное значение счетчика квадратов
k=1;
% цикл по квадратам, проверка, что было выведено не более 10 квадратов
% и объект кнопка Squares существует
while (k<=10) & ishandle(hObject)
    % очистка осей
    cla
    % вывод квадрата размером с оси (т.е. 1 на 1) случайного цвета,
задаваемого в RGB
    rectangle('FaceColor',[rand(1) rand(1) rand(1)])
    % задержка на 0.5 сек.
    pause(0.5)
    % увеличение k на 1 для проверки при следующем входе в цикл
    k=k+1;
end

function btnCirc_Callback(hObject, eventdata)
% подфункция обработки события Callback кнопки Stop

% задаем начальное значение счетчика кругов
k=1;
% цикл по кругам, проверка, что было выведено не более 10 кругов
% и объект кнопка Circles существует
while (k<=10) & ishandle(hObject)
    % очистка осей
    cla
    % вывод круга диаметром 1 случайного цвета, задаваемого в RGB
    rectangle('FaceColor',[rand(1)  rand(1)  rand(1)],'Curvature',[1  1])      %
задержка на 0.5 сек.
    pause(0.5)
    % увеличение k на 1 для проверки при следующем входе в цикл
    k=k+1;
end
```

После вызова функции squarecirc, например из командной строки (необходимо убедиться, что каталог, в котором находится файл mypi.m, является текущим или прописан в путях поиска MATLAB).

```
>> squarecirc
```

на экране появляется окно нашего приложения squarecirc. Нажатие на кнопку Squares приводит к отображению десяти цветных квадратов на осях с интервалом в 0.5 секунды. Если не дожидаясь завершения вывода квадратов нажать на кнопку Circles, то начнут выводится круги, а по завершении вывода кругов продолжится рисование квадратов. Так происходит потому, что созданные в окне приложения при помощи функции uicontrol кнопки Squares и Circles по умолчанию имеют значение свойства Interruptible равное 'on' и, кроме того, подфункция btnSq_Callback содержит вызов функции pause. Поэтому событие Callback кнопки Squares прерывается событием Callback кнопки Circles и начинает работу подфункция btnCirc_Callback, выводящая десять разноцветных кругов. По завершении работы подфункции btnCirc_Callback происходит возврат в точку прерывания в подфункции btnSq_Callback и она заканчивает вывод квадратов.

К прерыванию работы подфункции обработки события, кроме функции pause, приводят следующие функции:

- drawnow - обновление графического окна и проверка событий в очереди;
- figure - создание пустого стандартного графического окна;

- getframe - создание массива с кадрами (для получения анимации из меняющегося содержимого осей);
- waitfor - приостановка выполнения команд до тех пор, пока некоторый объект не будет удален или заданное его свойство не изменится, указатель на этот объект (и, возможно, свойство) задается во входном аргументе функции waitfor.

Предположим теперь, что мы не хотим, чтобы при нажатии на кнопку Circles работа подфункции btnSq_Callback прерывалась и подфункция btnCirc_Callback начинала бы рисовать круги. Вместо этого нам требуется завершение вывода всех десяти квадратов и только затем вывод десяти кругов. Для этого надо запретить прерывание обработки события Callback кнопки Squares. Для запрета прерывания события требуется установить ее свойству Interruptible значение 'off', что можно сделать, например, при создании кнопки Squares функцией uicontrol. Т.е. в приведенной выше функции squarecirc требуется заменить вызов

```
uicontrol('Style','pushbutton','Position',[30 15 70 20],...
    'String','Squares','Callback',@btnSq_Callback)
```

на

```
uicontrol('Style','pushbutton','Position',[30 15 70 20],...
    'String','Squares','Callback',@btnSq_Callback,...
    'Interruptible','off')
```

далее сохранить функцию squarecirc и снова запустить приложение squarecirc. Теперь последовательное нажатие на кнопки Squares и Circles приводит к выводу десяти квадратов и только затем к выводу десяти кругов.

Вывод кругов начинается из-за того, что событие Callback кнопки Circles ставится в очередь событий и подфункция btnCirc_Callback выполняется по завершении работы подфункции btnSq_Callback. Это происходит потому, что свойство BusyAction кнопки Circles по умолчанию имеет значение 'queue' (что значит очередь в переводе с английского). Если же мы не хотим, чтобы событие Callback кнопки Circles ставилось в очередь, т.е. чтобы приложение не реагировало на нажатие кнопки Circles пока выводятся квадраты, то требуется установить свойству BusyAction кнопки Circles значение 'cancel'. Для этого в приведенной выше функции squarecirc требуется заменить создание кнопки Circles функцией uicontrol со значением BusyAction по умолчанию

```
uicontrol('Style','pushbutton','Position',[140 15 70 20],...
    'String','Circles','Callback',@btnCirc_Callback)
```

на

```
uicontrol('Style','pushbutton','Position',[140 15 70 20],...
    'String','Circles','Callback',@btnCirc_Callback,...
    'BusyAction','cancel')
```

Далее, нужно сохранить функцию squarecirc и снова запустить приложение squarecirc. Теперь при нажатии на кнопку Squares выводятся квадраты и нажатие в это время на кнопку Circles не меняет работу нашего приложения.

Заметим, что если свойство Interruptible кнопки Squares имеет значение 'on' (т.е. прерывание ее события Callback разрешено), то вне зависимости от значения свойства BusyAction кнопки Circles все равно произойдет прерывание работы подфункции btnSq_Callback, выполнение подфункции btnCirc_Callback и возврат в точку прерывания подфункции btnSq_Callback для ее завершения работы. Для того, чтобы удостовериться в этом, достаточно в функции squarecirc заменить вызов функции uicontrol

```
uicontrol('Style','pushbutton','Position',[30 15 70 20],...
    'String','Squares','Callback',@btnSq_Callback,...
    'Interruptible','off')
```

на прежний вариант вызова функции uicontrol, в котором значение свойства Interruptible по умолчанию равно 'on'

```
uicontrol('Style','pushbutton','Position',[30 15 70 20],...
    'String','Squares','Callback',@btnSq_Callback)
```

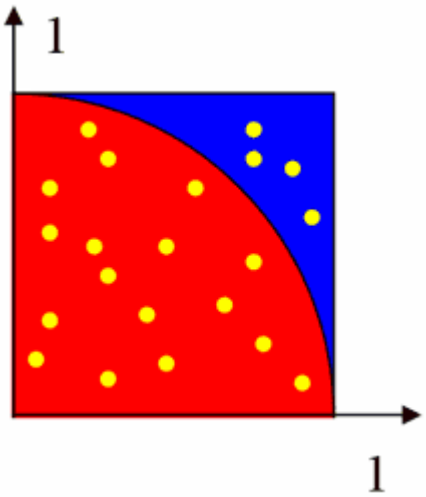
Примечание
Заметим, что выполнение обработки события объекта прерывается вне зависимости от значения его свойства Interruptible в том случае, когда прерывающим событием является одно из перечисленных ниже:

- событие, возникающее при удалении объекта, соответствующее свойство объекта DeleteFcn может содержать указатель на функцию обработки этого события или команду MATLAB;
- событие, возникающее при создании объекта, соответствующее свойство объекта CreateFcn может содержать указатель на функцию обработки этого события или команду MATLAB;
- событие, возникающее при закрытии графического окна, соответствующее свойство графического окна CloseRequest может содержать указатель на функцию обработки этого события или команду MATLAB;
- событие, возникающее при изменении размеров графического окна, соответствующее свойство графического окна ResizeFcn может содержать указатель на функцию обработки этого события или команду MATLAB.

Прерывание вычислений, кнопки Start и Stop

В этом разделе мы рассмотрим пример приложения, вычисляющего число π методом Монте-Карло.

Рассмотрим квадрат со стороной 1 и четверть круга радиуса 1, расположенные так, как на рисунке ниже.

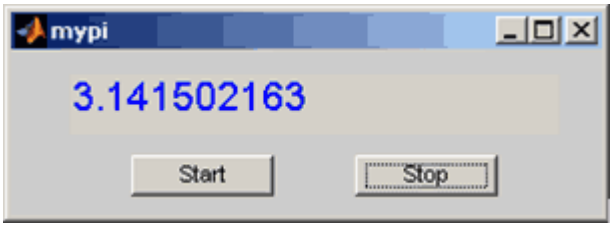


Метод Монте-Карло для нахождения числа π

Предположим, что в квадрат мы случайным образом бросаем точки (они на рисунке желтые), причем абсциссы и ординаты этих точек равномерно распределены на отрезке [0,1]. Часть точек оказывается внутри круга. Чем больше точек мы бросаем, тем плотнее они расположены. Ясно, что вероятность попадания точек в четверть круга равна ее площади $\frac{\pi}{4}$. Если через *k* обозначено число брошенных точек, а через *kc* - число точек, попавших в четверть круга, то с увеличением числа брошенных точек отношение $\frac{kc}{k}$ будет все ближе и ближе к значению $\frac{\pi}{4}$ и тогда можно приближенно положить

$$\pi \approx 4\frac{kc}{k}.$$

Мы напишем приложение mypi с графическим интерфейсом пользователя, в котором будет две кнопки Start и Stop и область для вывода результата так, как показано на рисунке ниже.



Окно приложения mypi

При нажатии на кнопку Start в цикле начинается процесс генерации точек со случайными значениями абсцисс и ординат из отрезка [0,1] (по 1000 точек за шаг цикла), после чего подсчитывается, сколько всего точек уже брошено в квадрат и сколько из них попало в четверть круга. Обновленное значение $\frac{kc}{k}$ выводится в область вывода текста. Процесс нахождения числа π можно остановить, нажав на кнопку Stop.

В основной функции mypi создаются следующие объекты приложения mypi:

- графическое окно приложения без стандартных меню и панели инструментов с заголовком mypi;
- объект статический текст, указатель на который записан в переменную ht, в него текст будет выводиться шрифтом синего цвета и размера 14пт с выравниванием по левому краю текстового объекта;
- кнопка Start, с событием Callback которой связывается подфункция btnStart_Callback, в подфункцию btnStart_Callback передается в качестве третьего входного аргумента указатель ht на текстовый объект для вывода в него полученного на каждом шаге цикла приближения к числу π;
- кнопка Stop, с событием Callback которой связывается подфункция btnStop_Callback со стандартным набором входных аргументов.

Вычисление числа π в подфункции btnStart_Callback организовано в цикле while, условием продолжения которого является равенство флага calcflag единице. Перед циклом ему присваивается единица и он сохраняется в данных приложения mypi при помощи функции guidata. В цикле после вычисления нового приближения к числу происходит его вывод в область статического текста и вызывается функция drawnow для проверки очереди

событий и обновления графического окна (т.е. для отображения результата в текстовой области).

Если пользователь нажал кнопку Stop, то выполнение подфункции btnStart_Callback прерывается и начинает работу подфункция btnStop_Callback, в которой флагу calcflag присваивается ноль и он сохраняется в данных приложения при помощи функции guidata. Далее происходит возврат в точку прерывания в подфункции btnStart_Callback.

В конце цикла while при помощи функции guidata считывается состояние флага calcflag, которое могло измениться. Если calcflag стал равным нулю, то работа подфункции btnStart_Callback прекращается.

Так происходит потому, что по умолчанию при создании кнопки Start функцией uicontrol значение ее свойства Interruptible равно 'on' и обработка ее события Callback прерывается другим событием Callback, которое возникает при нажатии пользователем кнопки Stop (подробно про прерывание событий и их постановку в очередь написано в разделе Разрешение на прерывание события, постановка событий в очередь).

Примечание.
В приложении mypi использована функция guidata для обмена данными (флагом calcflag) между подфункциями приложения, т.к. все переменные подфункций являются локальными и не видны из других подфункций. Если требуется обмениваться более чем одной переменной, то обычно заводят структуру, поля которой являются этими переменными. Эту структуру сохраняют и получают при помощи функции guidata. Например в одной подфункции можно написать

```
data.par1=1.5;
data.par2=[3, 5, 7];
guidata(src,data)
```

Здесь src - указатель на некоторый объект приложения. Далее, для получения значений параметров в другой подфункции приложения пишут

```
data=guidata(src);
```

Здесь src может быть указателем и на другой объект приложения (как правило, это указатель на объект, событие которого выполняется в данный момент). Структура данных data приложения получена, с ними можно выполнять различные действия, например:

```
a=data.par1;
b=data.par2;
```

Ниже приведен текст основной функции mypi вместе с подфункциями обработки событий кнопок Start и Stop. Функция mypi вместе с подфункциями btnStart_Callback и btnStop_Callback должны быть записаны в файле mypi.m. Для ее вызова достаточно в командной строке выполнить

```
>> mypi
```

(необходимо убедиться, что каталог, в котором находится файл mypi.m, является текущим или прописан в путях поиска MATLAB).

```
function mypi
% Вычисление pi по методу Монте-Карло
```

```
% создание окна приложения без стандартных меню и панели инструментов
% с заголовком mypi
figure('Menubar','none', 'Position',[520 200 290 80],...
    'Name','mypi','NumberTitle','off')
% создание статического текста и сохранение указателя на него
% в переменной ht
ht=uicontrol('Style','text','Position',[30 40 240 30],...
    'FontSize', 14, 'ForegroundColor', 'b',...
    'HorizontalAlignment','left')
% создание кнопки Start и связывание с ее событием Callback
% подфункции btnStart_Callback, указатель на статический текст
% передается в ее третьем дополнительном аргументе
uicontrol('Style','pushbutton','Position',[60 10 70 20],...
    'String','Start','Callback',{@btnStart_Callback, ht})
% создание кнопки Stop и связывание с ее событием Callback
% подфункции btnStop_Callback
uicontrol('Style','pushbutton','Position',[170 10 70 20],...
    'String','Stop','Callback',@btnStop_Callback)

function btnStart_Callback(src,evt,ht)
% Подфункция обработки нажатия на кнопку Start
% ht - указатель на статический текст

% начальное число брошенных точек равно 0
k=0;
% начальное число точек, попавших в четверть круга, равно 0
kc=0;
% присваиваем 1 флагу продолжения вычислений
calcflag=1;
% записываем его в данные приложения
guidata(src,calcflag)
% цикл по броскам точек
while calcflag
    % генерируем абсциссы и ординаты 1000 случайных точек
    % в единичном квадрате
    x=rand(1,1000);
    y=rand(1,1000);
    % увеличиваем число брошенных точек на 1000
    k=k+1000;
    % находим номера точек, попавших в четверть круга
    ind=find(x.^2+y.^2<1);
    % считаем, на сколько прибавилось точек внутри четверти круга
    kc=kc+length(ind);
    % вычисляем новое приближение для числа pi
    p=4*kc/k;
    % преобразуем числовое значение в строковое с 10-ю знаками
    str=num2str(p,10);
    % выводим результат в область статического текста
    set(ht, 'String', str)
    % вызываем drawnow для обновления окна и получения событий в очереди
    drawnow
    % получаем значение флага продолжения вычислений
    calcflag=guidata(src);
end

function btnStop_Callback(src,evt)
% Подфункция обработки нажатия на кнопку Stop

% обнуляем флаг продолжения вычислений
calcflag=0;
% сохраняем его как данные приложения
guidata(src,calcflag)
```

ЧАСТЬ 2

Фокус ввода, доступные и недоступные элементы интерфейса, всплывающие подсказки, обход клавишей Tab.

В этом разделе мы рассмотрим, как сделать интерфейс приложения дружественным и понятным пользователю. Рассмотрим простой пример приложения myplot с графическим интерфейсом пользователя, которое служит для построения графиков функций одной переменной. Окно приложения myplot (см. рис. 1a) содержит оси, область ввода Function для задания функции одной переменной, области ввода Left и Right для задания границ интервала для построения графика и две кнопки: Plot - для вывода графика очередной функции и Clear - для удаления графика.

Интерфейс приложения myplot организуем следующим образом.

1. В начале работы приложения myplot кнопки Plot и Clear недоступны (поскольку функция не задана).

2. После ввода формулы в Function и нажатия <Enter> фокус передается области ввода Left, затем Right.

3. После нажатия <Enter> в области ввода Right кнопка Plot становится доступной и находится в фокусе.

4. Если пользователь нажал на Plot, то на оси добавляется график, кнопка Clear становится доступной и фокус передается области ввода Function (см. рис. 1b).

5. Если пользователь нажал Clear, то все графики пропадают, области ввода Function, Left и Right очищаются, кнопки Plot и Clear становятся недоступными и фокус передается области ввода Function (см. рис. 1a).

Кнопки и области ввода снабдим всплывающими подсказками. Кроме того, установим порядок обхода элементов управления клавишей <Tab>.

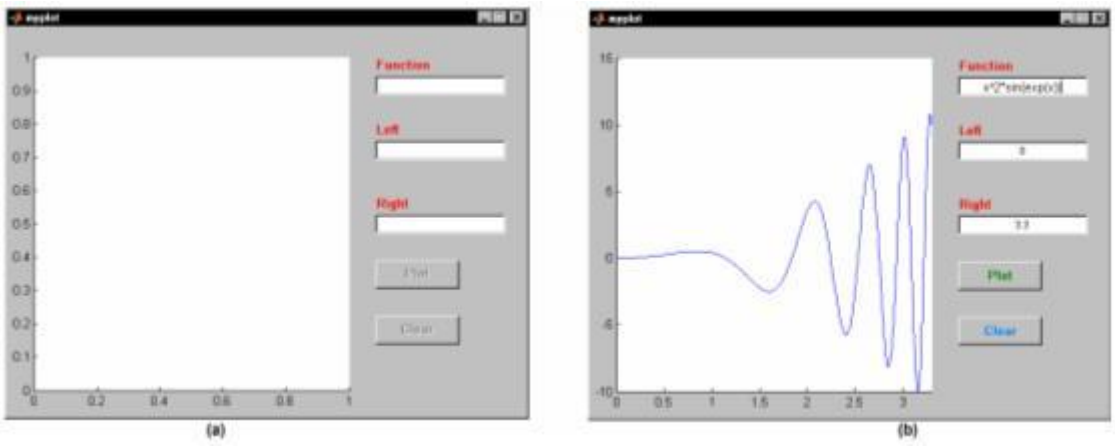


Рис. 1. Приложение myplot

В среде GUIDE создайте заготовку для нового приложения и разместите на ней (см. рис. 2):

· оси (при помощи инструмента Axes);

· три области ввода (при помощи инструмента Edit Text);

· три области статического текста (при помощи инструмента Static Text);

· и две кнопки (при помощи инструмента Push Button).

Размещение элементов управления мы обсуждали выше в разделе "Создание приложения hello в среде GUIDE"). На рис. 2 эти элементы приведены со своими тегами. Часть тегов для статического текста оставлена такими, какие даются по умолчанию (text1, text2, text3), поскольку мы не будем обращаться к статическому тексту при программировании приложения. Остальные теги желательно задать в соответствии с рис. 2, поскольку они понадобятся при программировании событий.

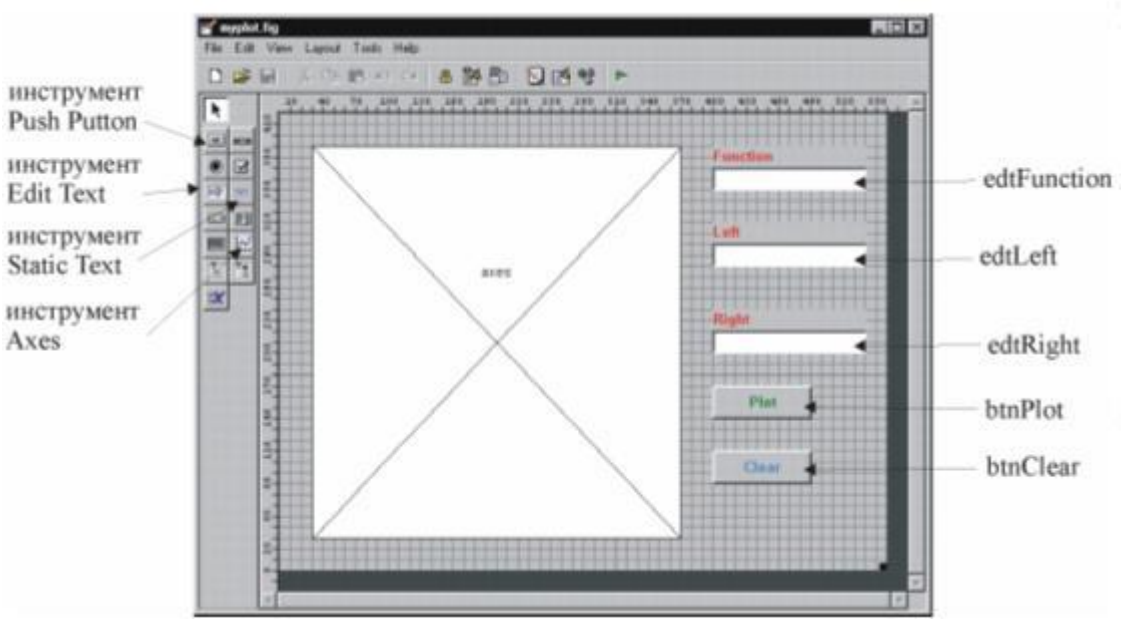


Рис. 2. Размещение элементов управления в среде GUIDE

Задайте теги в инспекторе свойств, а так же установите значения другим свойствам объектов:

· свойство осей NextPlot в add для того, чтобы линии графиков добавлялись на оси с сохранением предыдущих;

· свойство Enable обеих кнопок в off для того, чтобы в начале работы приложения кнопки были недоступны;

· свойство String областей статического текста в подходящие значения (Function, Left и Right);

· цвет статического текста и надписи на кнопках (Plot, Clear) при помощи свойства ForegroundColor;

· свойства шрифта статического текста при помощи свойств, названия которых начинаются со слова Font;

· свойство String областей ввода - пустую строку.

Примечание

Свойство Enable имеют все элементы управления. По умолчанию оно принимает значение 'on' и элемент управления доступен. Кроме 'on' и 'off', Enable может принимать значение 'inactive', при этом элемент управления выглядит как доступный, но не работает.

Примечание

В нашем случае область ввода является однострочной, такой она добавляется по умолчанию. Область ввода может допускать ввод нескольких строк, разделяемых нажатием на клавишу <Enter>. Для того, чтобы сделать область ввода многострочной, следует установить значения ее свойствам Max и Min так, чтобы их разность была больше единицы.

Примечание

Для завершения ввода многострочного текста используется сочетание клавиш <Ctrl>+<Enter>. В случае однострочного текста значением свойства String является строка, а в случае многострочного - массив строк.

Сохраните приложение с именем myplot. Далее требуется запрограммировать события Callback областей ввода и кнопок. Событие Callback области ввода, состоящей из одной строки, возникает, если пользователь нажал <Enter> или перешел к другому элементу управления (т.е. фокус был передан другому объекту), или щелкнул мышью по свободному месту в окне приложения.

Приступим к программированию событий в среде GUIDE, выбирая во всплывающем меню каждого элемента управления в пункте View Callbacks подпункт Callback для перехода к соответствующей подфункции в m-файле myplot.m, ассоциированном с приложением.

При возникновении события Callback области ввода Function требуется передать фокус области ввода Left. Для передачи фокуса объекту используется функция uicontrol, во входном аргументе которой задается указатель на объект (функция uicontrol служит также и для создания элементов управления Uicontrol с указанием их свойств). Как мы уже обсуждали, указатели на объекты содержатся в полях структуры handles, имена полей совпадают с тегами соответствующих объектов, например handles.edtLeft содержит указатель на область ввода Left. Поэтому, подфункция edtFunction_Callback должна иметь следующий вид.

```
function edtFunction_Callback(hObject, eventdata, handles)
% передаем фокус области ввода Left
uicontrol(handles.edtLeft)
```

Аналогичным образом запрограммируем события Callback областей ввода Left и Right, но в случае области ввода Right не только передадим фокус кнопке Plot, но и сделаем ее доступной. Подфункции edtLeft_Callback и edtRight_Callback приведены ниже.

```
function edtLeft_Callback(hObject, eventdata, handles)
% передаем фокус области ввода Right
uicontrol(handles.edtRight)

function edtRight_Callback(hObject, eventdata, handles)
% делаем кнопку Plot доступной
set(handles.btnPlot, 'Enable', 'on')
% передаем фокус кнопке Plot
uicontrol(handles.btnPlot)
```

При возникновении события Callback кнопки Plot выведем график, сделаем доступной кнопку Clear и передадим фокус области ввода Function. Для визуализации функции воспользуемся fplot, которая строит график функции с адаптивным подбором шага, учитывающим поведение функции. В самом простом случае обращение к fplot выглядит так:

fplot(fun, [a b])

Здесь fun - строка или строковая переменная с записью выражения для функции в соответствии с правилами MatLab, a и b - границы изменения аргумента. Строку с выражением для функции мы получим, обратившись к свойству String области ввода Function (при помощи функции get), а для получения границ изменения аргумента придется не только считать строковые значения свойства String области ввода Left и Right, но и преобразовать их в числовые, обратившись к функции str2num. Подфункция btnPlot_Callback может выглядеть следующим образом.

```
function btnPlot_Callback(hObject, eventdata, handles)
% записываем строку с формулой в строковую переменную fun
fun = get(handles.edtFunction, 'String');
% получаем содержимое области ввода Left, преобразуем в число
% и заносим в переменную Left
Left = str2num(get(handles.edtLeft, 'String'))
% получаем содержимое области ввода Right, преобразуем в число
% и заносим в переменную Right
Right = str2num(get(handles.edtRight, 'String'))
% вызываем fplot для построения графика функции
fplot(fun, [Left Right])
% делаем доступной кнопку Clear
set(handles.btnClear, 'Enable', 'on')
% передаем фокус области ввода Function
uicontrol(handles.edtFunction)
```

Примечание

В нашем случае график выведется на оси, размещенные в окне приложения. Так произойдет потому, что окно приложения становится текущим графическим окном при выполнении событий. Где устанавливаются подобного рода опции? В меню Tools среды GUIDE выберите пункт GUI Options. Появляется диалоговое окно GUI Options, раскрывающийся список которого Command-line accessibility содержит строку Callback (GUI becomes Current Figure within Callbacks). В этом диалоговом окне изменять ничего не нужно. Обсуждению этих опций мы посвятим отдельный раздел.

Для очистки осей проще всего применить команду cla, поскольку в нашем случае единственные оси, размещенные в окне приложения, являются текущими. Функция обработки события кнопки Clear должна также удалять содержимое областей ввода, делать кнопки Plot и Clear недоступными и передавать фокус области ввода Function.

```
function btnClear_Callback(hObject, eventdata, handles)
% очистка осей
cla
% удаление содержимого областей ввода
set(handles.edtFunction, 'String', '')
set(handles.edtLeft, 'String', '')
set(handles.edtRight, 'String', '')
% делаем кнопки Plot и Clear недоступными
set(handles.btnClear, 'Enable', 'off')
set(handles.btnPlot, 'Enable', 'off')
% передаем фокус области ввода Function
uicontrol(handles.edtFunction)
```

Запустите приложение myplot и убедитесь, что оно работает верно. Осталось снабдить элементы управления всплывающими подсказками и установить порядок их обхода клавишей Tab.

Для того, чтобы при наведении мыши на элемент управления появлялась всплывающая подсказка, требуется в инспекторе свойств установить его свойство TooltipString в подходящее значение, например:

· для области ввода Function - Input formula;

· для области ввода Left - Input left boundary of interval и т.д.

Для задания порядка обхода клавишей Tab служит редактор Tab Order Editor. Его окно появляется при выборе в меню Tools среды GUIDE пункта Tab Order Editor. Панель инструментов содержит всего две кнопки: Move Up и Move Down для перемещения текущей строки с элементом управления вверх или вниз списка, соответственно (см. рис. 3). Пользуясь ими можно задать любой порядок обхода.

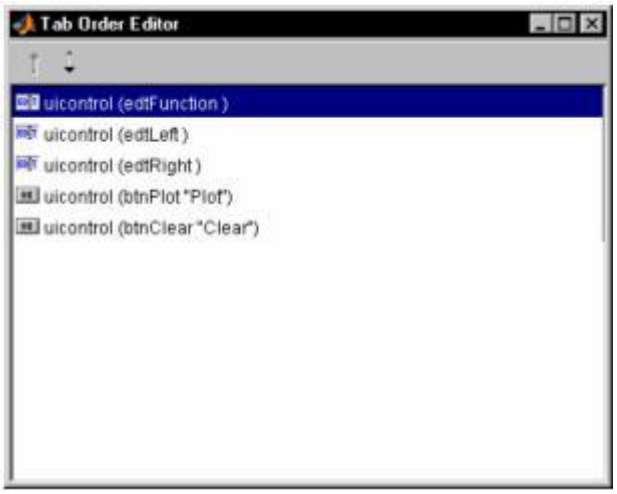
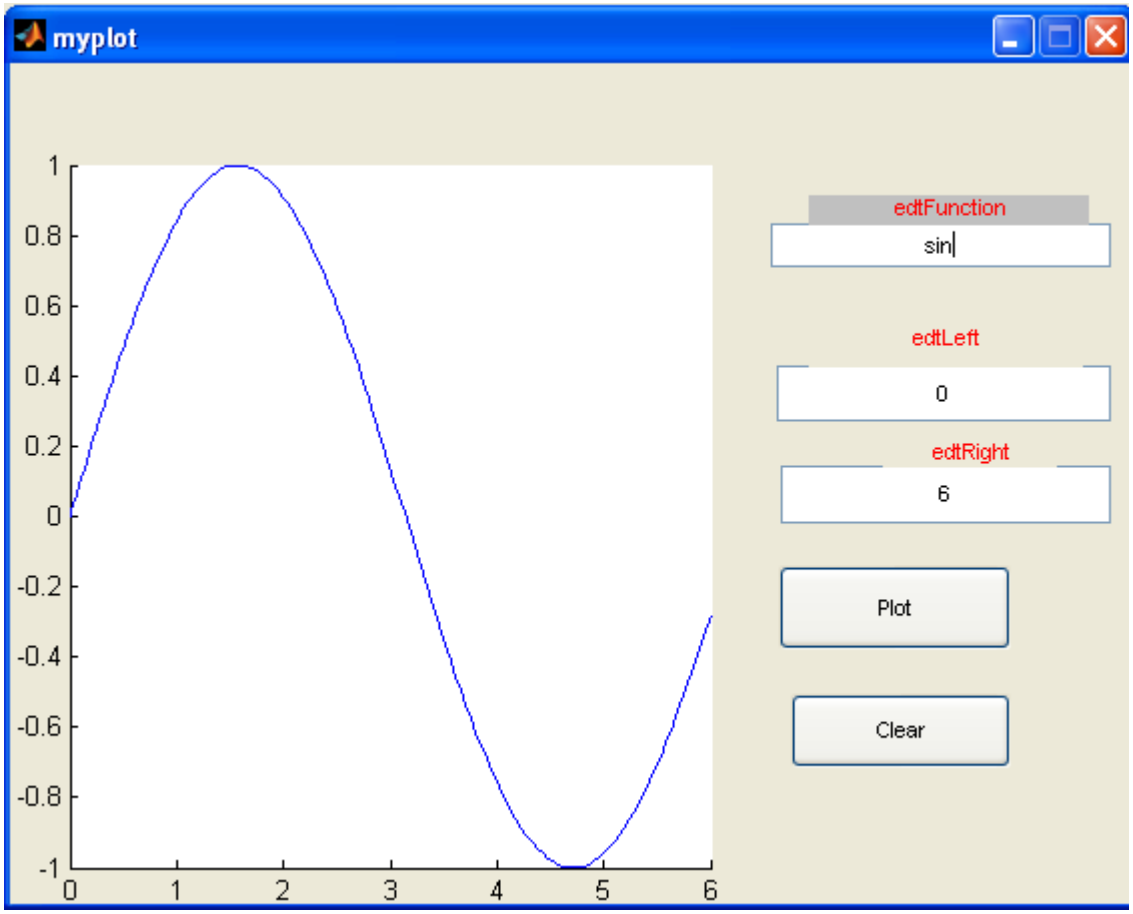


Рис. 3. Редактор Tab Order Editor


myplot
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
edtFunction
sin
edtLeft
0
edtRight
6
Plot
Clear