ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Приборостроение»

ПРАКТИКУМ «Синтез приложений с графическим интерфейсом пользователя»

по дисциплине

«Компьютерные технологии в медико-биологической практике»

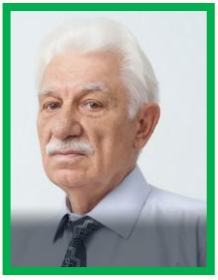
Авторы
Литвин А.В.,
Мороз К.А.,
Бабенко Е.В.

Ростов-на-Дону, 2016

Аннотация

Методические указания к практическим занятиям по дисциплине «Компьютерные технологии в медико-биологической практике» предназначены для студентов очной и заочной форм обучения по направлению бакалавриата 12.03.04 «Биотехнические системы и технологии».

Авторы



к.т.н., доцент
Литвин А.В.

к.т.н., доцент
Мороз К.А.

ст. преподаватель
Бабенко Е.В.

Оглавление

1 СОЗДАНИЕ ПРИЛОЖЕНИЯ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ ПОЛЬЗОВАТЕЛЯ В MATLAB

1.1 Принципы разработки приложений в среде GUIDE системы MATLAB

Для разработки приложений с графическим интерфейсом пользователя в состав MATLAB входит специализированная среда GUIDE. Приложения MATLAB (GUI-приложения) являются графическими окнами, содержащими элементы управления (кнопки, раскрывающиеся списки, переключатели, флаги, полосы скроллинга, меню и т.д.). Создание приложений включает расположение нужных элементов интерфейса в пределах графического окна и определение команд MATLAB, которые должны выполняться при обращении пользователя к данным элементам, например, при нажатии кнопки или выборе пункта меню.

Приложение может состоять как из одного основного окна, так и из нескольких окон, и осуществлять вывод текстовой и графической информации в основное окно приложения и в отдельные окна. Ряд функций MATLAB предназначен для создания стандартных диалоговых окон открытия и сохранения файла, печати, выбора шрифта, области ввода данных и др., которыми можно пользоваться в собственных приложениях. Процесс работы над приложением допускает постепенное добавление элементов в графическое окно, запуск и тестирование приложения, возврат в режим редактирования.

1.2 Заготовка окна приложения, добавление элементов управления

Создадим простое приложение в среде GUIDE, позволяющее поочередный вывод двух графиков (сигнала, состоящего из суммы синусоид с частотами 5, 15 и 25 Гц, и спектральной плотности мощности этого сигнала) с возможностью очистки осей, нанесения и удаления сетки, изменения цвета и толщины линий графиков, использования маркеров, а также сохранения окна приложения на жестком диске.

Создайте папку, в которой будете сохранять приложение и сделайте ее текущей.

Вызов среды GUIDE производится выполнением команды **guide** в командном окне MATLAB (рис. 1.1) или через меню **New** путем выбора пункта **Graphical User Interface** (рис. 1.2).

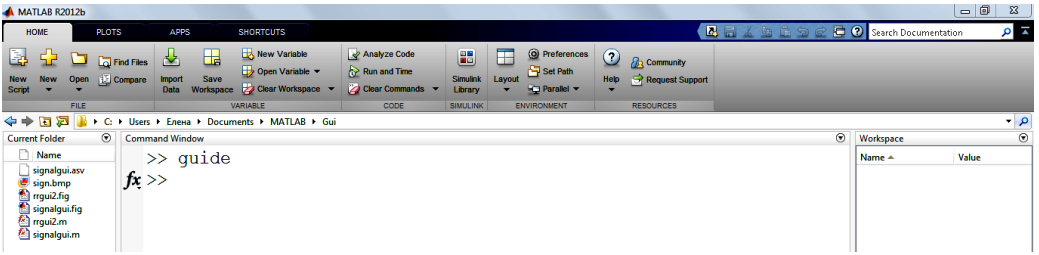


Рисунок 1.1 - Вызов среды GUIDE командой **guide**

Перейдите в среду GUIDE любым из вышеуказанных способов. В результате запустится мастер создания графического интерфейса и появится диалоговое окно **GUIDE Quick Start** (рис. 1.3).

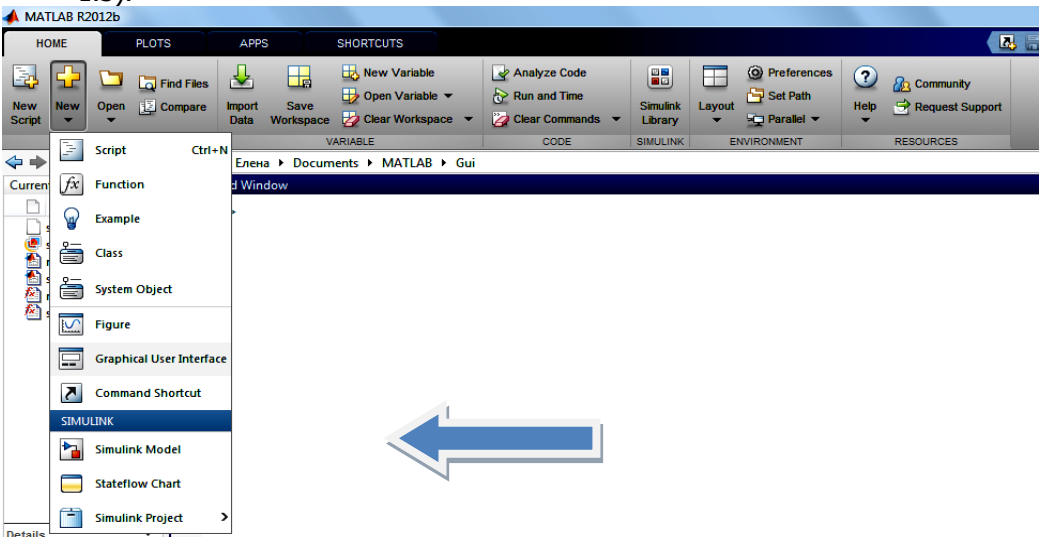


Рисунок 1.2 - Вызов среды GUIDE с помощью меню

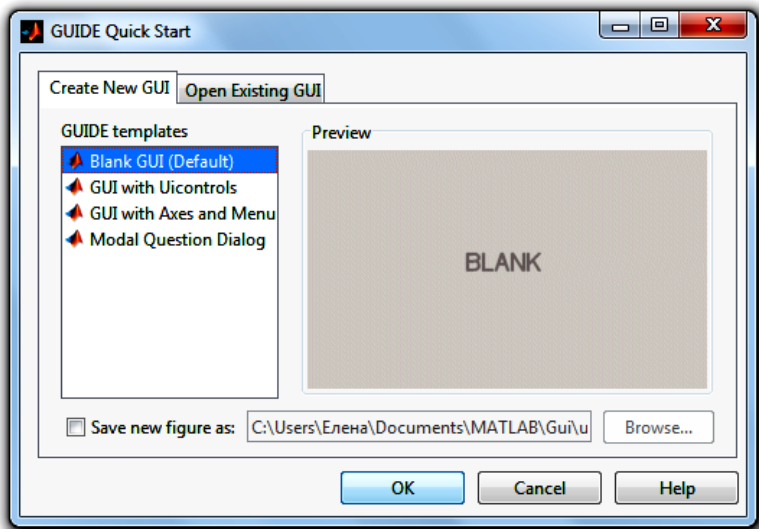


Рисунок 1.3 - Диалоговое окно **GUIDE Quick Start**

Диалоговое окно **GUIDE Quick Start** имеет две вкладки:

• вкладка **Create New GUI** (создание нового приложения), позволяющая создавать новое приложение. В ней можно выбрать четыре вида заготовки: **Blank GUI** (пустое окно приложения), **GUI with Uicontrols** (заготовка с кнопками, переключателями и областями ввода), **GUI with Axes and Menu** (заготовка с осями, меню, кнопкой и раскрывающимся списком), **Modal Question Dialog** (заготовка для модального окна);

• вкладка **Open Existing GUI** (открытие существующего приложения).

Кроме того, во вкладке **Create New GUI** внизу есть флаг, установка которого позволяет сразу задать имя файла, в котором будет храниться создаваемое приложение. Поскольку приложение всегда можно будет сохранить в процессе редактирования, то этот флаг устанавливать необязательно.

Во вкладке **Create New GUI** выберете строку **Blank GUI** и нажмите кнопку **<OK>.** При этом появится основное окно среды GUIDE**,** содержащее заготовку окна приложения, панель инструментов для добавления элементов интерфейса, меню, управляющую панель, горизонтальную и вертикальную линейки (рис. 1.4). Для добавления элементов управления в заготовку окна приложения необходимо на па- нели инструментов выбрать со-

ответствующий элемент и при помощи мыши перетащить его в окно приложения. Всегда можно изменить размер элемента управления, выделив его в заготовке окна при помощи мыши (должен быть включен режим выделения, т.е. выбран соответствующий инструмент **Select**) и потянув мышью за квадратные маркеры. В режиме выделения возможно перемещать элементы управления по заготовке окна приложения.

Часто требуется, чтобы небольшое перемещение мыши вызывало изменение положения элемента интерфейса на некоторый фиксированный шаг. Сетка редактора приложений позволяет осуществить такое движение. Выбор пункта **Grid and Rulers** меню **Tools** приводит к появлению диалогового окна **Grid and Rulers**, изображенного на рис. 1.5.

Флаги **Show rulers** и **Show grid** соответствуют отображению линеек и сетки в окне редактора приложения, а раскрывающийся список **Grid Size** позволяет выбрать размер ячеек сетки. Минимально допустимый размер (10 пикселов) позволяет достаточно точно располагать элементы управления в окне приложения. 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.

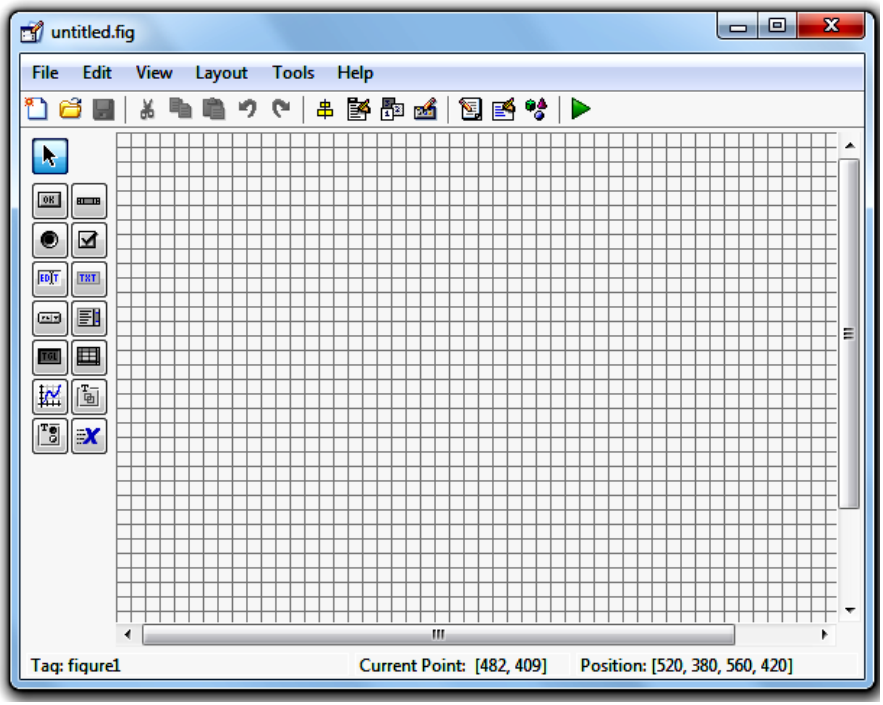


Рисунок 1.4 - Основное окно среды **GUIDE**

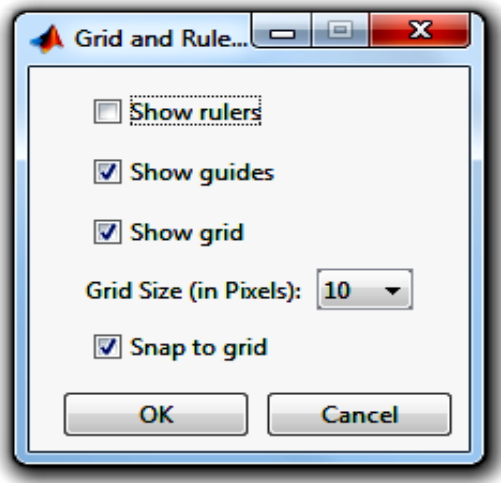


Рисунок 1.5 - Диалоговое окно **Grid and Rulers**

При работе с элементами интерфейса необходимо уметь обращаться к ним для получения и установки значений их свойств и программирования событий, возникающих при обращении пользователя к элементу управления. Например, при выборе пункта меню. При размещении пользователем элемента управления в заготовке окна средой GUIDE автоматически генерируется и присваивается созданному объекту приложения его уникальное имя, которое называют тегом. Значением свойства **Tag** является строка. Теги используются для доступа к свойствам объектов приложения (для получения и изменения значений свойств объектов приложения), чаще всего в подфункциях обработки различных событий других объектов. Можно оставить сгенерированный средой GUIDE тег объекта и без изменений, но при создании объемного приложения, содержащего большое количество объектов, удобнее давать им мнемонические имена (легко запоминающиеся и отражающие особенности объекта).

Выберите на панели инструментов элемент интерфейса **Axes** и расположите в окне заготовки приложения (рис. 1.6).

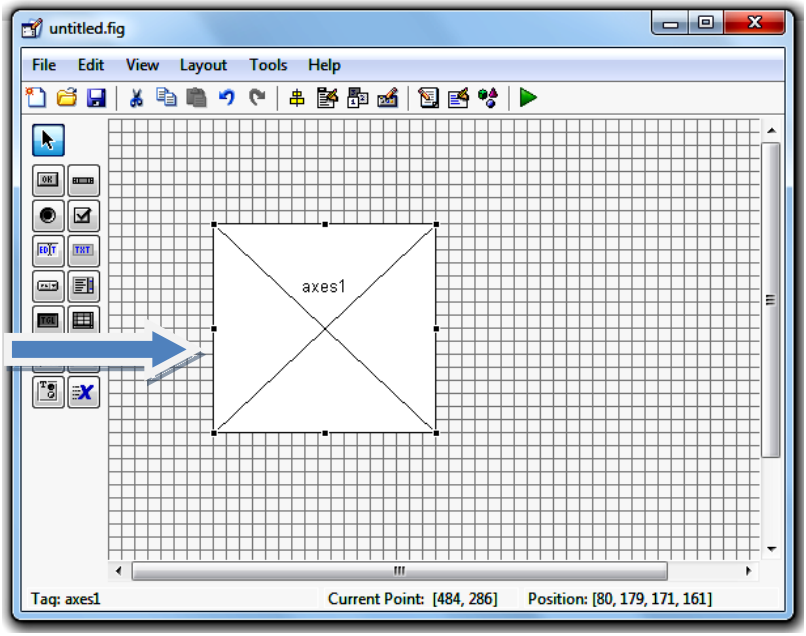


Рисунок 1.6 - Окно редактора приложения с элементом интерфейса **Axes**

Для задания тега перейдите к инспектору свойств. Это можно сделать либо двойным щелчком мыши по добавленному объекту (осям), либо щелкнув правой кнопкой мыши на объекте и выбрав из выпадающего меню строку **Property Inspector**. При этом появится окно инспектора свойств (**Inspector**), в котором отображены свойства выбранного объекта (рис. 1.7). Найдите в левом столбце таблицы свойство **Tag** и в области ввода справа от него измените автоматически сгенерированное значение **axes1** на **axPlot** (удобно задавать имена, часть которых определяет тип элемента интерфейса), затем нажмите клавишу <**Enter**> и закройте инспектор свойств. Обратите внимание на изменение имени элемента оси.

Сохраните шаблон интерфейса под именем signalgui. При сохранении приложения автоматически создается М-файл с таким же именем. Таким образом, созданное приложение хранится в двух файлах с одним именем, но разными расширениями: **fig –** графическое окно с размещенными на нем элементами управления и **m –** файл-функция с подфункциями, которые обрабатывают различные события, возникающие в ходе взаимодействия пользователя с приложением.

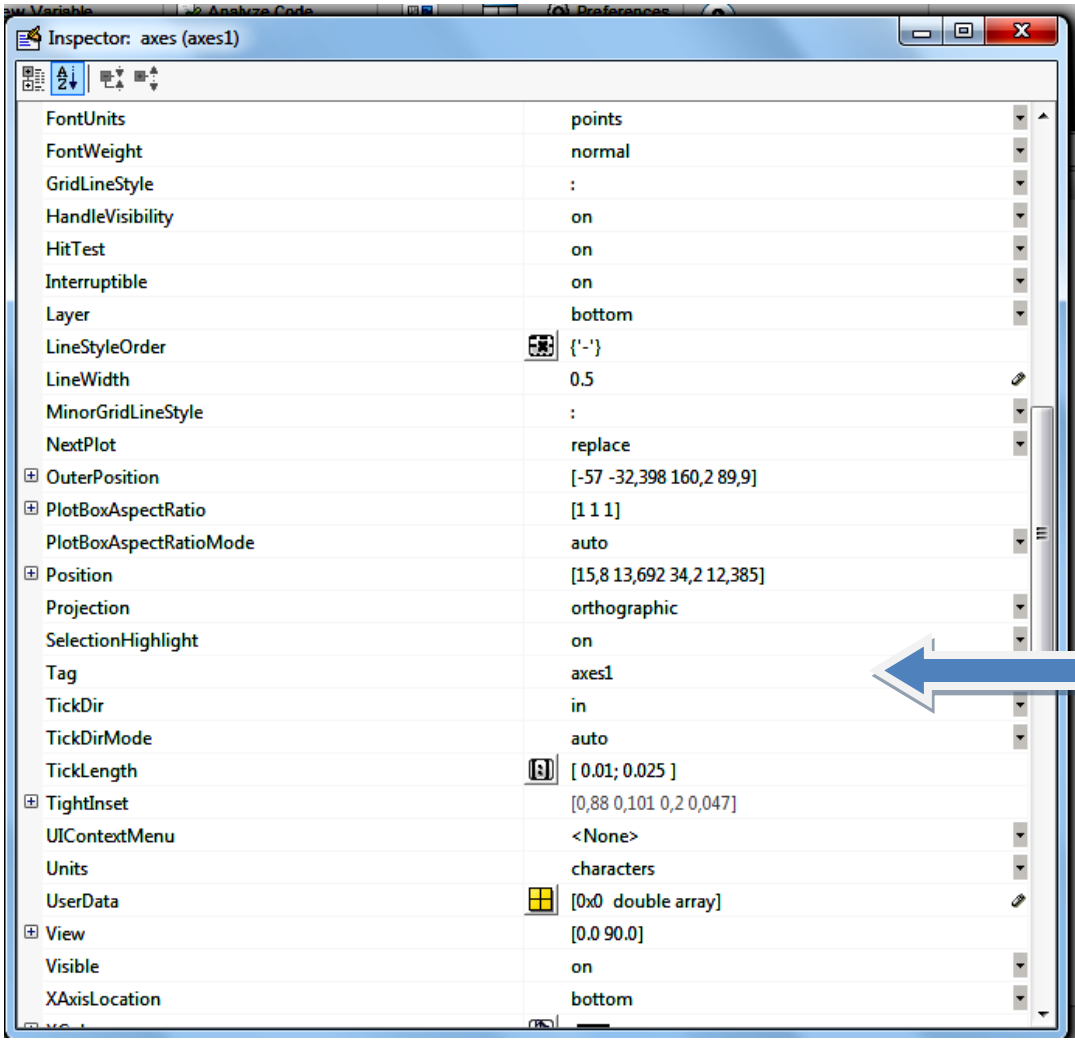


Рисунок 1.7 - Окно инспектора свойств элемента **Axes**

При обращении пользователя к элементу интерфейса (например, при выборе пункта меню или нажатии кнопки) возникает событие **Callback.** Поэтому свойству **Callback** выбранного элемента необходимо задать подфункцию обработки события **Callback**. Это можно сделать двумя способами. Во-первых, в программе (в файле с расширением **m)** задать в качестве значения свойства **Callback** указатель на подфункцию, обрабатывающую

событие **Callback,** и описать указанную подфункцию. Во-вторых, перейти к заготовке окна приложения и, вызвав контекстное меню элемента щелчком правой кнопки мыши, выбрать в пункте **View Callbacks** подпункт **Callback**. При этом происходит переход в редактор М-файлов к подфункции обработки события **<имя элемента>_Callback**, заголовок которой и комментарии генерируются автоматически.

Затем требуется описать тело функции, т.е. записать те операторы, которые будут выполняться при выборе пункта меню или при нажатии кнопки.

Имя подфункции состоит из тега объекта и названия события **Callback,** которое будет обрабатываться (существуют и другие события). Эта подфункция имеет следующие входные аргументы:

- аргумент **hObject** содержит указатель на кнопку или пункт меню, т.е. идентификатор (**handle**) элемента управления, для которого вызвана процедура обработки;
- аргумент **eventdata** зарезервирован для использования в следующих версиях MATLAB;
- аргумент **handles** является структурой с указателями на все объекты приложения. Названия полей структуры **handles** совпадают с названиями тегов созданных объектов. Например, **handles.figure1** содержит указатель на окно приложения.

Наиболее распространенными типами callback-процедур в GUIDE являются:

- **Callback** – вызывается при выполнении функционального действия элемента интерфейса (нажатия кнопки, выбора пункта меню);
- **CreateFcn** – вызывается при создании элемента управления, но до отображения формы на экране. Шаблон, генерируемый GUIDE для данной процедуры, обычно реализует установку цвета отображения элемента на экране;
- **DeleteFcn** – вызывается при удалении элемента из памяти;
- **KeyPressFcn** – вызывается в случае, когда пользователь нажимает клавишу на клавиатуре, а элемент управления является текущим.

Следует отметить, что при создании графических объектов полезно сохранять указатели на них в переменных, поскольку тогда появляется возможность манипулировать объектами (удалять, копировать, осуществлять поиск, менять их свойства). Некоторые

функции для получения идентификаторов графических объектов в программах MATLAB:

- **findobj** – выполняет поиск объекта по заданным свойствам (например, имени) и возвращает идентификатор объекта (указатель на объект);
- **gco** – получение указателя на текущий объект;
- **gcb** – вызывает идентификатор окна, содержащего объект, для которого вызвана процедура обработки события;
- **gcbo** – возвращает идентификатор объекта, для которого вызвана процедура обработки события;
- **gcf** – возвращает идентификатор текущего (активного) графического окна; обычно используется при работе с элементами управления;
- **gcbf** – возвращает идентификатор повторно вызываемого графического окна;
- **gca** – возвращает идентификатор текущего (активного) объекта **axes**; обычно используется при работе с графикой на дескрипторном («низком») уровне.

Все эти функции можно использовать, например, для установки значений свойствам только что созданным графическим объектам в приложении.

Руководствуясь вышесказанным, добавьте в окно приложения кнопку для очистки осей. Для этого при помощи мыши перетащите элемент интерфейса **Push Button** в окно приложения и расположите его, как показано на рис. 1.8.

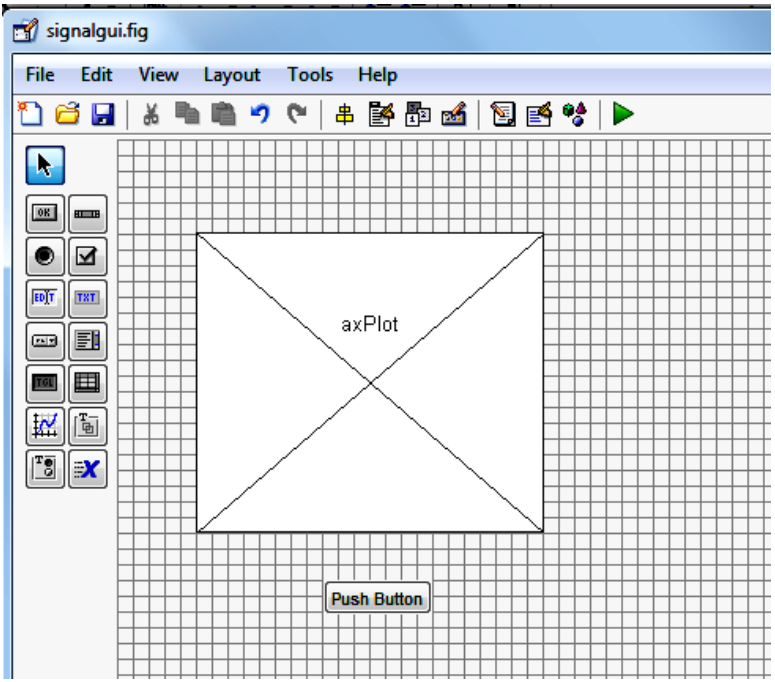


Рисунок 1.8 - Добавление элемента интерфейса **Push Button** в окно приложения

Вызовите для кнопки инспектор свойств и занесите в свойство **Tag** значение **btnClear,** а в свойство **String – Очистить оси**. Закройте инспектор свойств и обратите внимание на изменение надписи на кнопке. Перейдите к подфункции обработки события **Callback** добавленной кнопки, для чего следует выбрать пункт **View Callbacks->Callback** всплывающего меню (рис. 1.9). Выбор данного пункта делает активным редактор М-файлов. Наполните подфункцию **btnClear_Callback** оператором и комментарием, выделенными жирным шрифтом:

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

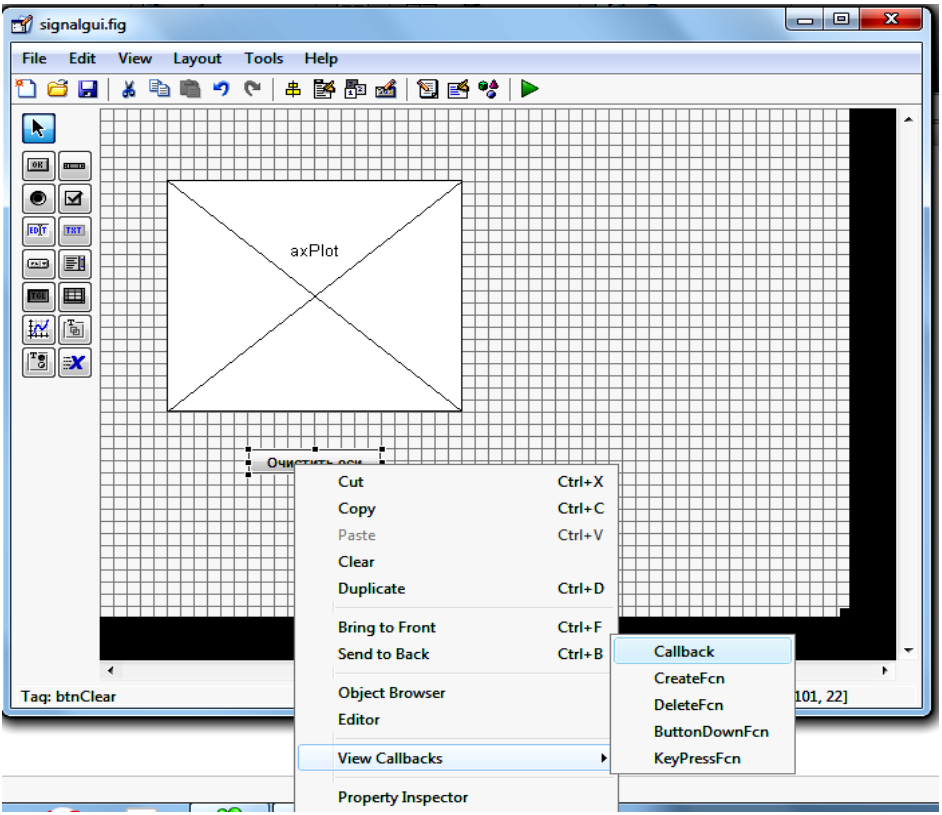


Рисунок 1.9 - Переход к подфункции обработки события **Callback** кнопки

Запустите приложение кнопкой **Run** (зеленый треугольник на верхней панели) и убедитесь в том, что приложение содержит оси и кнопку. Пока при обращении к кнопке ничего не происходит, т.к. событие **Callback** для нее не запрограммировано. Если приложение работает правильно, то в командном окне нет сообщений об ошибках.

Закройте окно приложения при помощи кнопки с крестиком в правом верхнем углу приложения и продолжите работать в режиме редактирования. При случайном выходе из режима редактирования в него легко можно вернуться из командного окна MATLAB, набрав в командной строке:

>> **guide <имя приложения>**

1.3 Создание меню и подменю приложения

Перейдите к созданию главного меню приложения, подменю и запрограммируйте выбор каждого из пунктов меню.

Приложения MATLAB могут использовать стандартное меню графического окна. Среда GUIDE позволяет разработчику дополнять стандартное меню и создавать собственные меню. Свойство **MenuBar** окна приложения (элемента **figure**) отвечает за наличие стандартных меню **File**, **Edit**, **Tools**, **Window** и **Help** в работающем приложении. Значение **figure** данного свойства соответствует отображению стандартных меню, а **none** приводит к приложению без строки с меню. Независимо 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.

Ознакомьтесь с работой свойства **MenuBar** окна приложения, изменяя его значения, затем установите значение **none.**

Перейдите к созданию меню. Кнопкой  на панели инструментов активизируйте редактор меню (**Menu Editor**) (рис. 1.10)

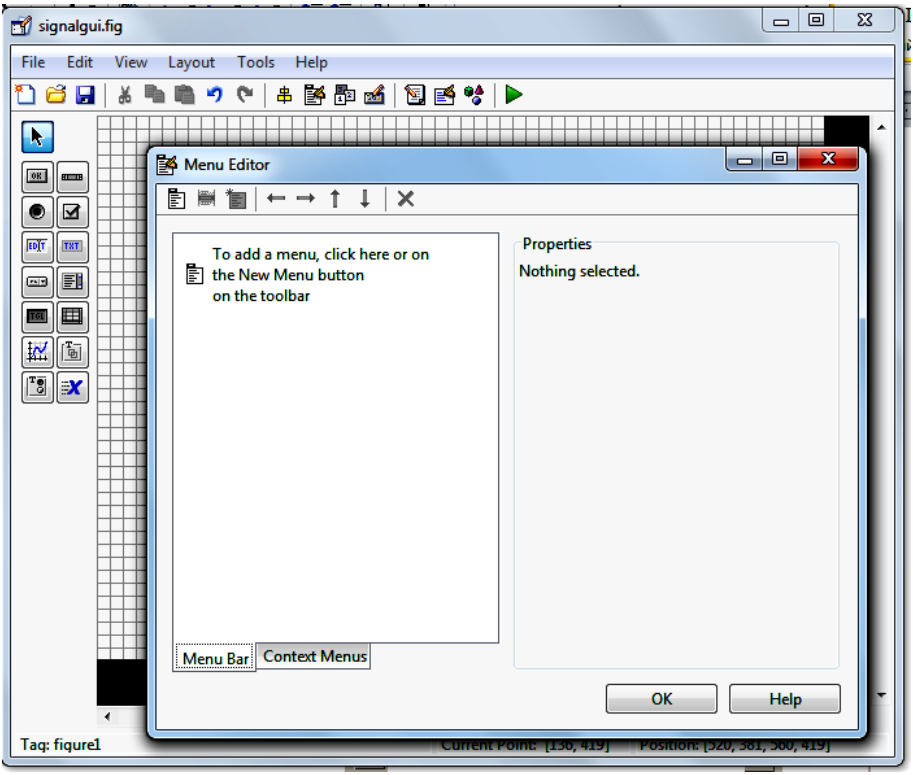


Рисунок 1.10 - Вызов редактора меню (**Menu Editor**)

Кнопками **Новое меню** 

и затем **Раздел нового меню** 

создайте меню. Заполните поля **Label** и **Tag** как указано на рис. 1.11. Для каждого пункта меню заполните поле **Tag** в соответствии с таблицей 1.1. Поле **Callback** заполняется автоматически при нажатии кнопки **View** (не нажимать до заполнения всех пунктов, подпунктов меню и тегов, иначе генерируемые подфункции получат имена **untitled** с порядковым номером, которые сохранятся, что при последующем изменении значений свойства **Tag** повлечет ошибки при выполнении приложения!).

Для подпункта меню **Выход** установите флаг для свойства **Separator above this item,** что позволит отделить этот пункт чертой.

Нажатием кнопки **View** создайте функцию **Callback.** Создать функцию **Callback** так же можно**,** выбрав раздел меню **CreateFcn** командой **View>View Callback>CreateFcn**, как показано на рис.1. 12.

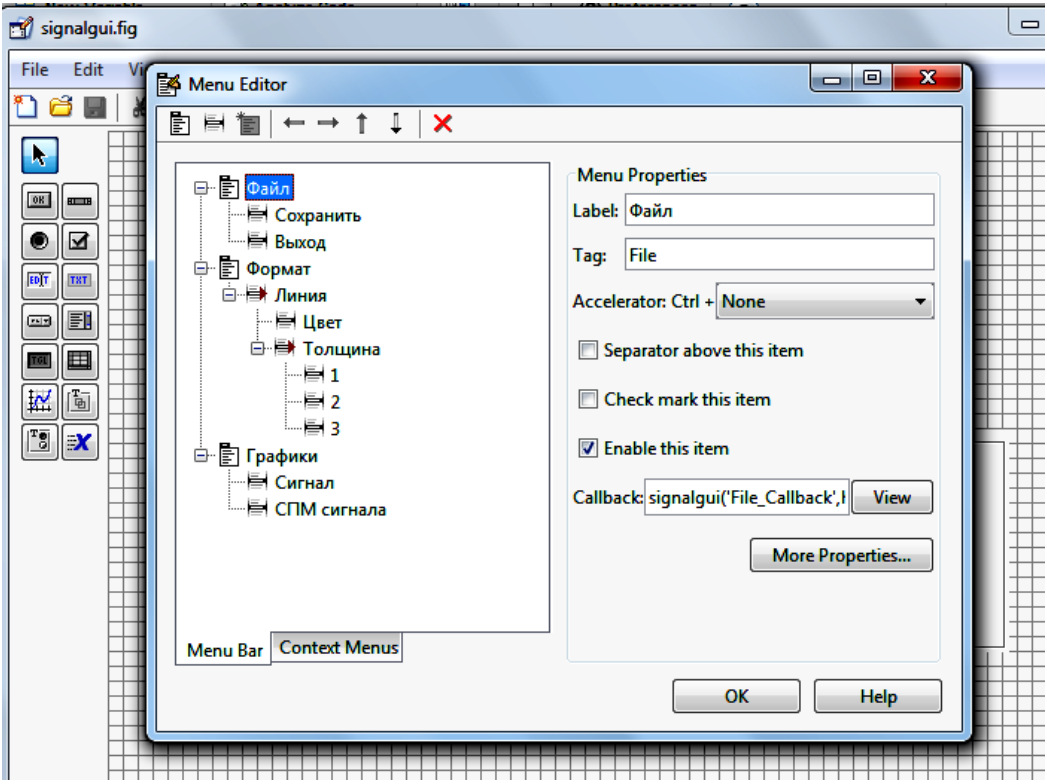


Рисунок 1.11 - Создание главного меню и подменю приложения

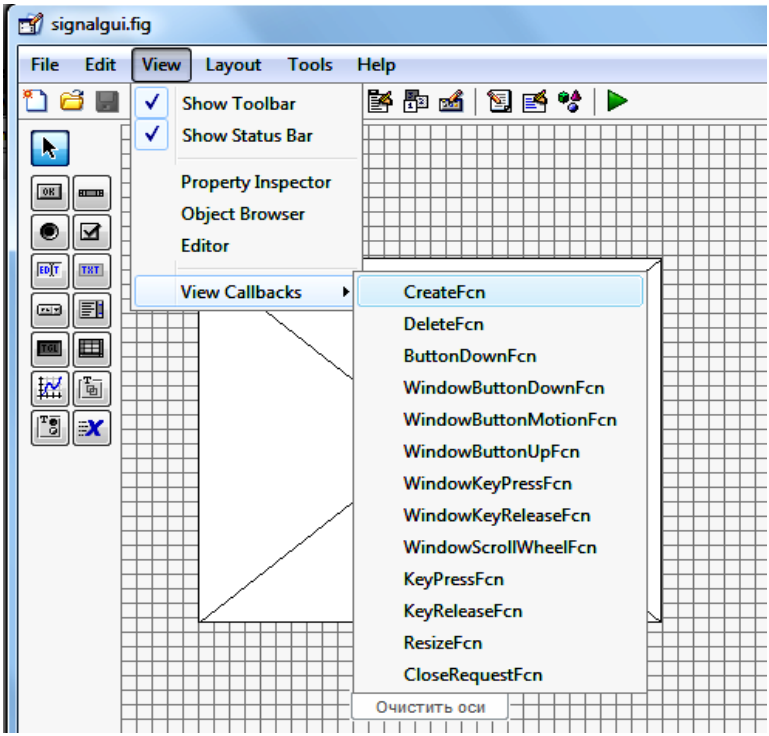


Рисунок 1.12 - Создание функции **Callback**

Запустите приложение и убедитесь в возможности обращения ко всем пунктам и подпунктам созданного меню (при обращении к подпунктам меню пока ничего не происходит, т.к. следует запрограммировать события).

Таблица 1.1 - Теги главного меню и подменю приложения signalgui

Пункт (подпункт) меню	**Значение свойства Tag**
Файл	**File**
Сохранить	**mnSave**
Выход	**mnExit**
Формат	**Format**
Линия	**mnLine**
Цвет	**mnColor**
Толщина	**mnWidth**
1	**mnWidth1**

2	**mnWidth2**
3	**mnWidth3**
Графики	**Graph**
Сигнал	**mnSign**
СПМ сигнала	**mnSpectr**

1.4 Использование свойства Enable

На данном этапе разработки приложения его можно усовершенствовать, сделав пункт меню **Формат** и кнопку **Очистить оси** неактивными до вывода графика. Для решения подобной задачи используют свойство **Enable**. Свойство элемента **Enable** отвечает за возможность доступа к нему, значение **on** разрешает доступ, а **off**, соответственно, запрещает. Установка значений свойствам объектов (элементов) в М-файле 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 быть 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.btnClear** является указателем на кнопку **Очистить оси**. Доступ к кнопке **Очистить оси** должен быть запрещен в начале работы приложения, пока пользователь не выберет пункты меню **Сигнал** или **СПМ сигнала**.

Установите в инспекторе свойств кнопки **Очистить оси** для свойства **Enable** значение **off**, используя кнопку со стрелкой в строке со значением свойства. Остальные изменения значения свойства **Enable** кнопки должны происходить в ходе работы приложения. Для разрешения и запрещения доступа к кнопке нужно внести дополнения в обработку ее события **Callback**.

Для пункта меню **Формат** выполните аналогичные действия (рис. 1.13).

Запустите приложение и проверьте результат. Пункт меню **Формат** и кнопка **Очистить оси** должны быть неактивными.

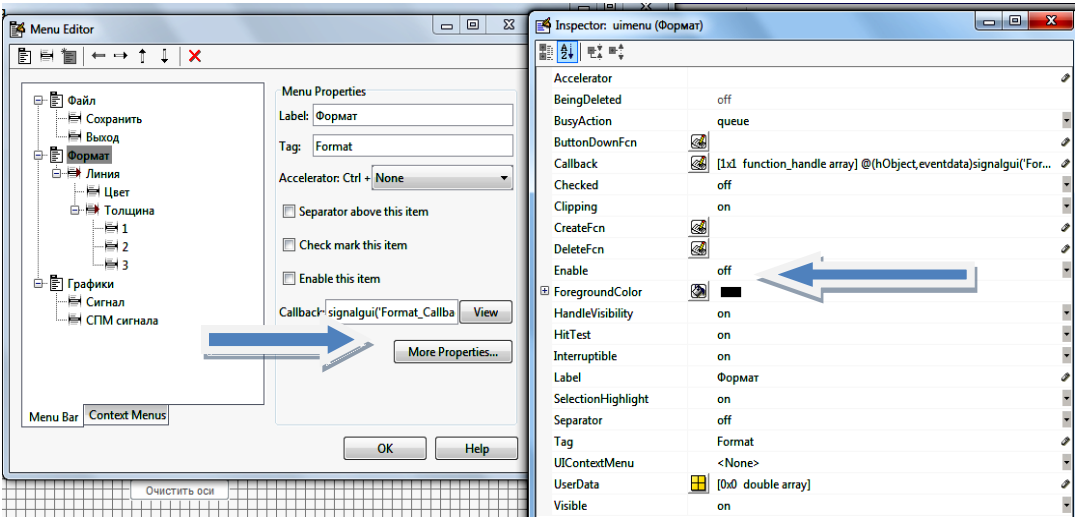


Рисунок 1.13 - Изменение свойств пункта меню **Формат**

1.5 Программирование вывода графиков и очистки осей

Запрограммируйте события для построения графиков и очистки осей.

При программировании событий следует учесть использование рассмотренных выше функции **set** и свойства **Enable**, а также последующую возможность изменения других свойств: цвета и толщины линии графика. Следовательно, необходимо обратиться к структуре указателей (идентификаторов) **handles** и функции **gcbo**. Кроме того, полезно использование глобальных переменных для исключения повторного ввода одних и тех же строк в программу. В нашем случае частота дискретизации **fs** и сигнал **y** используются в двух подфункциях. При создании больших приложений использование глобальных переменных необходимо.

В подфункции для построения графиков и очистки осей добавьте команды и комментарии, выделенные жирным шрифтом.

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

```
% Очистить оси
cla
% Кнопка Очистить должна стать недоступной после очистки осей
set(handles.btnClear,'Enable','off')
% Пункт меню Формат должен стать недоступным после очистки осей
set(handles.Format,'Enable','off')

function mnSign_Callback(hObject, eventdata, handles)
% hObject    handle to mnSign (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Объявим глобальными переменные y и fs для обращения к ним
% в других подфункциях
global y  fs
% Частота дискретизации fs=200 Гц,
% Вектор времени 5 с  с интервалом dt=0.01 с
fs=200; dt=1/fs; t=0:dt:5;
% Суммарный сигнал y из синусоид с частотами 5, 15 и 25 Гц,
y=sin(2*pi*5*t)+sin(2*pi*15*t)+sin(2*pi*25*t);
% Построение графика суммарного сигнала для 100 точек
handles.line=plot(1000*t(1:100),y(1:100))
title('Total Signal');xlabel('Time, ms');ylabel('Magnitude')
% Соханение идентификатора объекта линия  в структуре указателей
guidata(gcbo, handles);
% Кнопка Очистить оси должна стать доступной
set(handles.btnClear,'Enable','on')
% Пункт меню Формат должен стать доступным
set(handles.Format,'Enable','on')

function mnSpectr_Callback(hObject, eventdata, handles)
% hObject    handle to mnSpectr (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global y fs
% БПФ сигнала y для 512 точек отсчета
n=512; Y=fft(y,n);
% Спектральная плотность мощности сигнала
```

```
Pyy=Y.*conj(Y)/n;
% Значения частотной оси
f=fs*(0:n/2)/n;
% Построение графика СПМ  сигнала
np=round(n/4)+1; % Число выводимых точек
handles.line=plot(f(1:np),Pyy(1:np))
title('Signal Power Spectrum');xlabel('Frequency, Hz')
guidata(gcbo, handles);
% Кнопка очистить должна стать доступной
set(handles.btnClear,'Enable','on')
% Пункт меню Формат должен стать доступным
set(handles.Format,'Enable','on')
```

Сохраните файл и запустите приложение для проверки результата. Если приложение работает правильно, то в командном окне нет сообщений об ошибках, а выбор подпунктов меню **Сигнал** и **СПМ сигнала** приводит к появлению нужных графиков (рис. 1.14). Кнопка **Очистить оси** и пункт меню **Формат** становятся активными после вывода любого из графиков. Нажатие кнопки **Очистить оси** приводит к удалению графика, а кнопка и пункт меню **Формат** снова становятся неактивными.

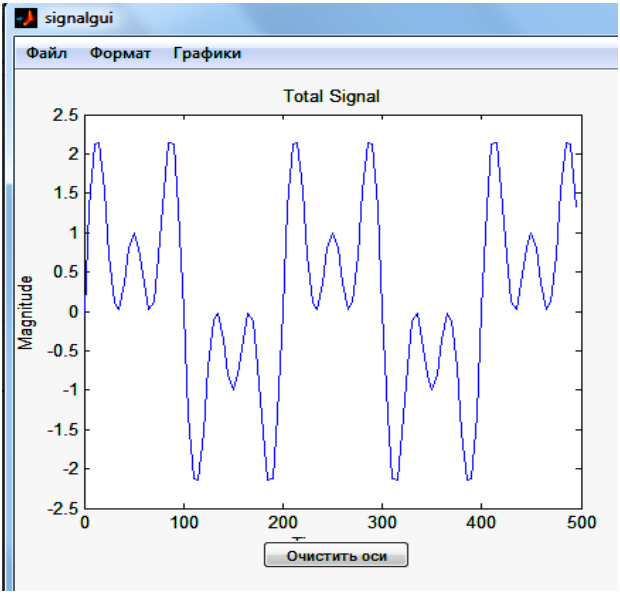


Рисунок 1.14 - Результат выбора подпункта меню **Сигнал**

1.6 Программирование выбора подпунктов меню Формат

1.6.1 Программирование выбора подпункта Толщина

Начните с программирования событий для трех вариантов выбора толщины линии, используя уже известную функцию **set**, указатель на объект, для которого устанавливается свойство – это линия, свойство **LineWidth** (толщина линии) и его значения 1, 2 и 3.

```
function mnWidth1_Callback(hObject, eventdata, handles)
% hObject    handle to mnWidth1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.line,'LineWidth', 1)
```

```
function mnWidth2_Callback(hObject, eventdata, handles)
% hObject    handle to mnWidth2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.line,'LineWidth', 2)
```

```
function mnWidth3_Callback(hObject, eventdata, handles)
% hObject    handle to mnWidth3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.line,'LineWidth', 3)
```

Запустите приложение, проверьте работу подпункта меню **Толщина.**

1.6.2 Программирование выбора подпункта Цвет. Использование диалоговых окон

Перейдите к программированию изменения цвета линии графика. Для изменения цвета используйте диалоговое окно.

Насколько удобен интерфейс приложения во многом определяется наличием диалоговых окон, облегчающих работу с файлами, или предназначенных для предостережения пользователя о событиях, которые могут повлечь его действия. MATLAB предоставляет разработчику приложения возможность использовать стандартные диалоговые окна Windows. Вид диалоговых окон может быть определен путем за- дания входных аргументов этих

функций. В данной работе рассматриваются диалоговые окна выбора цвета, сохранения файла и выхода из приложения.

Поскольку графика MATLAB обеспечивает получение цветных изображений, имеется ряд команд для управления цветом и различными цветовыми эффектами. Среди них важное место занимает команда установки палитры цветов. Палитра цветов RGB задается матрицей из трех столбцов, определяющих значения интенсивности красного (red), зеленого (green) и синего (blue) цветов. Их интенсивность задается в относительных единицах от 0.0 до 1.0. Например, [0 0 0] задает черный цвет, [1 1 1] – белый цвет, [0 0 1] – синий цвет. При изменении интенсивности цветов в указанных пределах возможно задание любого цвета. Таким образом, цвет соответствует общепринятому формату RGB.

Диалоговое окно выбора цвета отображается в результате выполнения функции **uisetcolor** (рис. 1.15).

```
function mnColor_Callback(hObject, eventdata, handles)
% hObject    handle to mnColor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uisetcolor(handles.line,'Select line color');
```

Запустите приложение, проверьте работу подпункта меню **Цвет.**

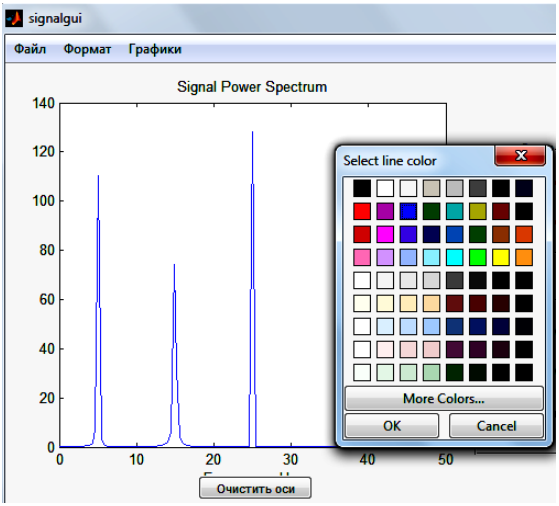


Рисунок 1.15 - Диалоговое окно выбора цвета линии

1.7 Программирование выбора подпункта меню Выход

Некоторые действия пользователя требуют повторного подтверждения. Например, пользователь может случайно выбрать подпункт меню **Выход**. Следует вывести диалоговое окно, в котором пользователь укажет, действительно ли требуется выход из приложения.

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.

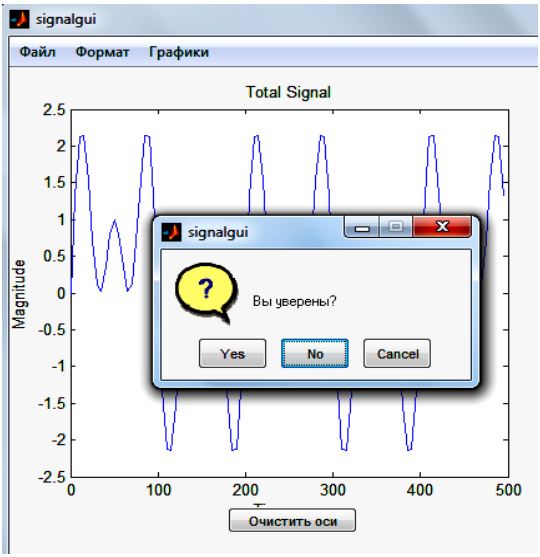


Рисунок 1.16 - Диалоговое окно подтверждения при выборе подпункта меню **Выход**

Запрограммируйте работу выбора подпункта меню **Выход** так, чтобы закрытие приложения выполнялось только в том случае, если пользователь нажал кнопку **Yes** в появляющемся диалоговом окне с текстом **Вы уверены?** и заголовком **signalgui** (рис. 1.16). Используйте условный оператор **if** и функцию **strcmp** для сравнения выходного аргу- мента **questdlg** со строкой **Yes**.

```
function mnExit_Callback(hObject, eventdata, handles)
% hObject    handle to mnExit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
button=questdlg('Вы уверены?','signalgui','No');
if strcmp(button, 'Yes')
   delete(handles.figure1);
end
```

Запустите приложение, проверьте его работу при выборе подпункта меню **Выход.**

1.8 Создания диалогового окна сохранения файла

Для создания диалогового окна сохранения файла в текущий каталог используется функция **uiputfile.** Фильтр файлов установлен в **ALL MATLAB files**; отображаются только те файлы, расширения которых поддерживаются MATLAB. В раскрывающемся списке **Files of type** можно выбрать только М-файлы или только графические окна, или все файлы.

Если в диалоговом окне сохранения файла пользователь выбирает имя существующего файла, то появляется окно подтверждения. Нажатие на кнопку **Yes** приводит к завершению диалога сохранения файла, а нажатие на кнопку **No** – к возврату

в окно сохранения файла. Если пользователь выбрал файл или набрал имя файла в строке **Имя файла** и нажал кнопку **Сохранить**, то FName будет содержать строку с именем файла и соответствующим расширением, а DirName **-** путь к файлу. Если пользователь не выбрал файл и закрыл окно нажатием на кнопку **Отмена**, то FName=0 и DirName=0. Поэтому после обращения к функции **uiputfile** следует проверить, был ли выбран файл. Если была нажата кнопка **Сохранить**, то необходимо объединить полученные строки в полный путь к файлу. Функция **uiputfile** не записывает данные на жесткий диск. С ее помощью возможно лишь отобразить диалоговое окно сохранения файла (рис. 1.17).

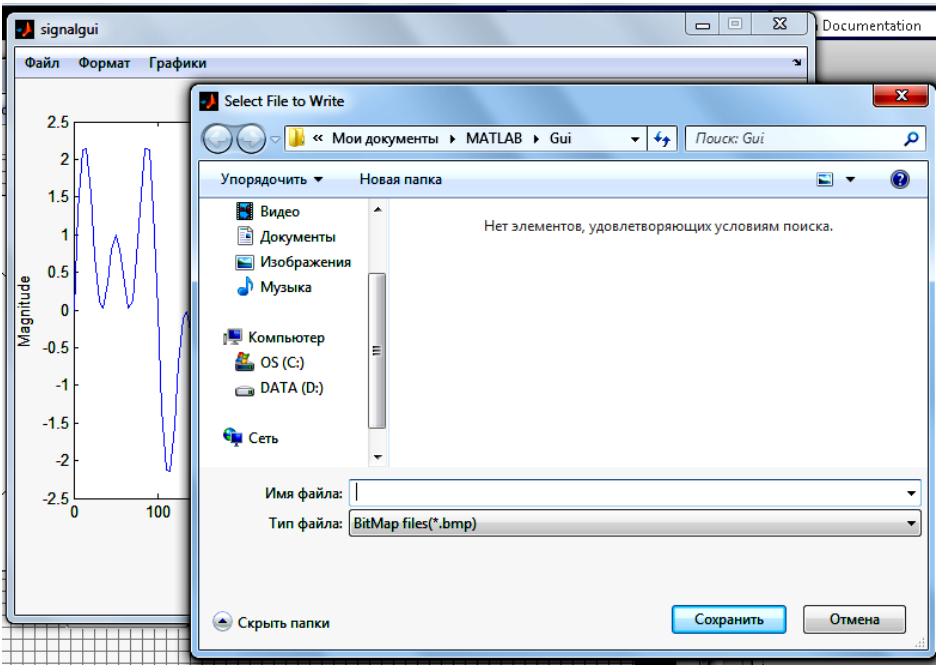


Рисунок 1.17 - Вызов окна сохранения файла

На основании вышесказанного запрограммируйте работу подпункта меню **Сохранить**:

```
function mnSave_Callback(hObject, eventdata, handles)
% hObject    handle to mnSave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Filter={'*.bmp','BitMap files(*.bmp)';'*.eps','Eps files'};
[FName, DirName, FilterIndex]=uiputfile(Filter);
if ~ isequal(FName, 0)
   FullName=strcat(DirName, FName);
   saveas(gcf, FullName, Filter{FilterIndex}(3:end));
end
```

1.9 Использование флагов, рамок и переключателей

1.9.1 Флаги и рамки

Флаги позволяют производить одну или несколько установок, определяющих ход работы приложения. Продолжите работу

над приложением signalgui, предоставив пользователю возможность наносить линии сетки на график. Окно приложения должно содержать два флага для нанесения сетки по оси x и по y. Если пользователь выбирает один из подпунктов меню Графики, то на оси наносится сетка в зависимости от установленных флагов. Нажатие на кнопку **Очистить оси** должно приводить не только к исчезновению графика, но и к скрытию сетки.

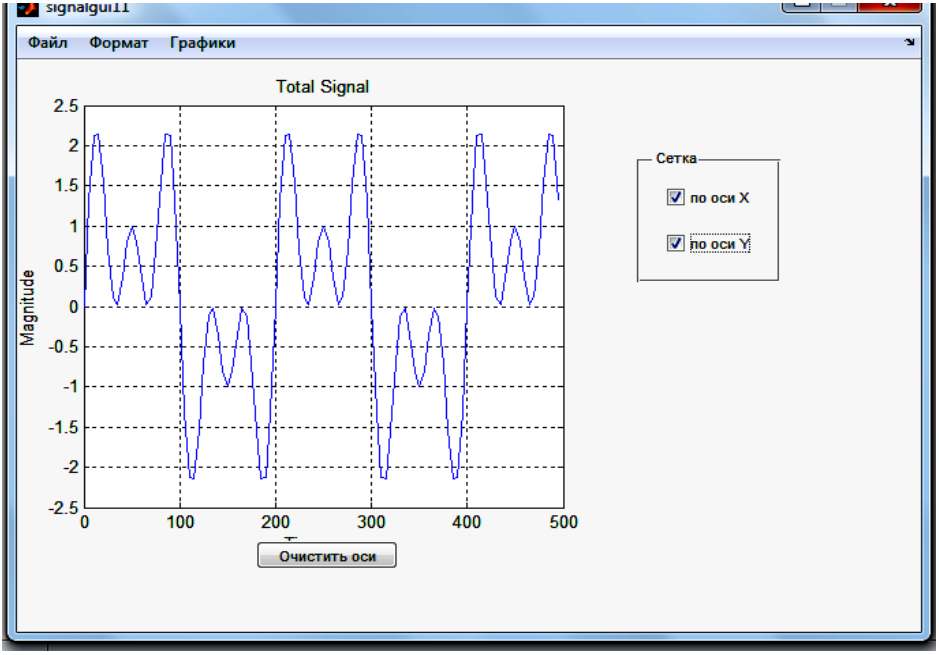


Рисунок 1.18 - Окно приложения с флагами

Обычно несколько элементов управления со схожими функциями группируют и помещают внутри рамки. Измените размеры окна приложения, освободив справа место для новых элементов управления. Вставьте рамку в окно приложения, выбрав элемент управления **Panel**. В рамку добавьте два флага **Check Box**. Дайте добавленным элементам имена и снабдите их поясняющими надписями. Задайте свойству **Tag** верхнего флага значение **chbxGridX,** а свойству **string**, отвечающему за подпись флага, значение **по x,** аналогичным образом определите свойства нижнего флага, установив свойству **Tag** значение **chbxGridY**, а свойству **string** - **по y**; свойство **Tag** элемента управления **Panel**

можно оставить без изменений, а свойству **string** установите значение **Сетка** (рис. 1.18**)**. Если текст не помещается рядом с флагом, увеличьте ширину области флага при помощи мыши, удерживая нажатой левую кнопку.

Перейдите в редактор М-файлов для автоматического создания подфункций обработки событий добавленных элементов. Осталось сделать так, чтобы при выборе пользователем подпунктов меню **Графики** происходило отображение линий сетки в зависимости от установленных флагов, а нажатие на кнопку **Очистить оси** приводило к скрытию сетки. Блоки обработки событий **Callback** кнопки и подпунктов меню 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.

Дополните в подфункции командами, выделенными жирным шрифтом.

```
function btnClear_Callback(hObject, eventdata, handles)
% hObject    handle to btnClear (see GCBO)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Пункт меню Формат должен стать недоступным после очистки осей
set(handles.Format,'Enable','off')
% Убрать сетку
set(gca,'XGrid','off')
set(gca,'YGrid','off')

function mnSign_Callback(hObject, eventdata, handles)
% hObject    handle to mnSign (see GCBO)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Сохранение идентификатора объекта линия в структуре указателей
guidata(gcbo, handles);
% Проверка флага сетки по оси 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 сетки по оси y
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 Очистить оси должна стать доступной
set(handles.btnClear,'Enable','on')
```

- -

По аналогии с подфункцией mnSign_Callback дополните подфункцию mnSpectr_Callback

Запустите приложение и протестируйте его.

1.9.2 Переключатели

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

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 signalgui, предоставив пользователю возможность выбора типа маркера (кружок, квадрат или отсутствие маркера).

Добавьте в окно приложения новую рамку и нанесите на нее три переключателя **Radio Button**, установите свойствам **Tag** значения **rbMarkCirc, rbMarkSq, rbMarkNone**, а **String** - **маркеры-круги, маркеры-квадраты, без маркеров** соответственно (рис. 1.19).

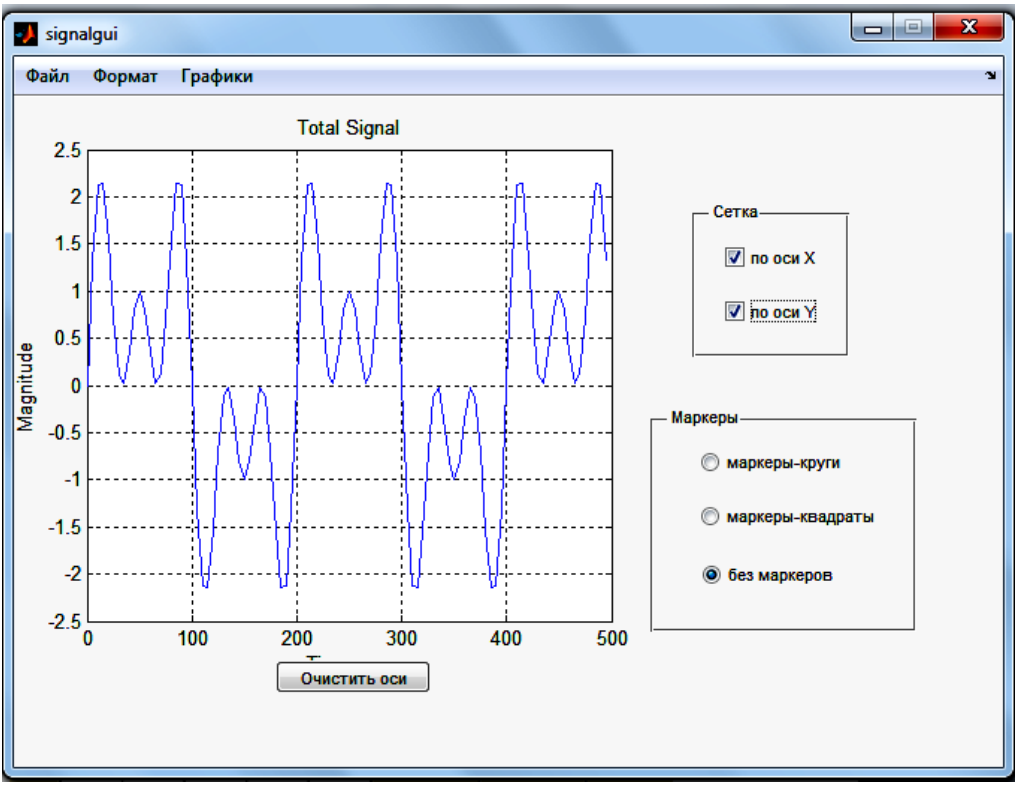


Рисунок 1.19 - Окно приложения с переключателями

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** устанавливается через редактор свойств переключателя **без маркеров**. В редакторе свойств нажмите кнопку в строке с **Value**. Появляется окно **Value**, изображенное на рис. 1.20.

Измените значение 0.0 на единицу и нажмите **OK**. 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 signalgui.

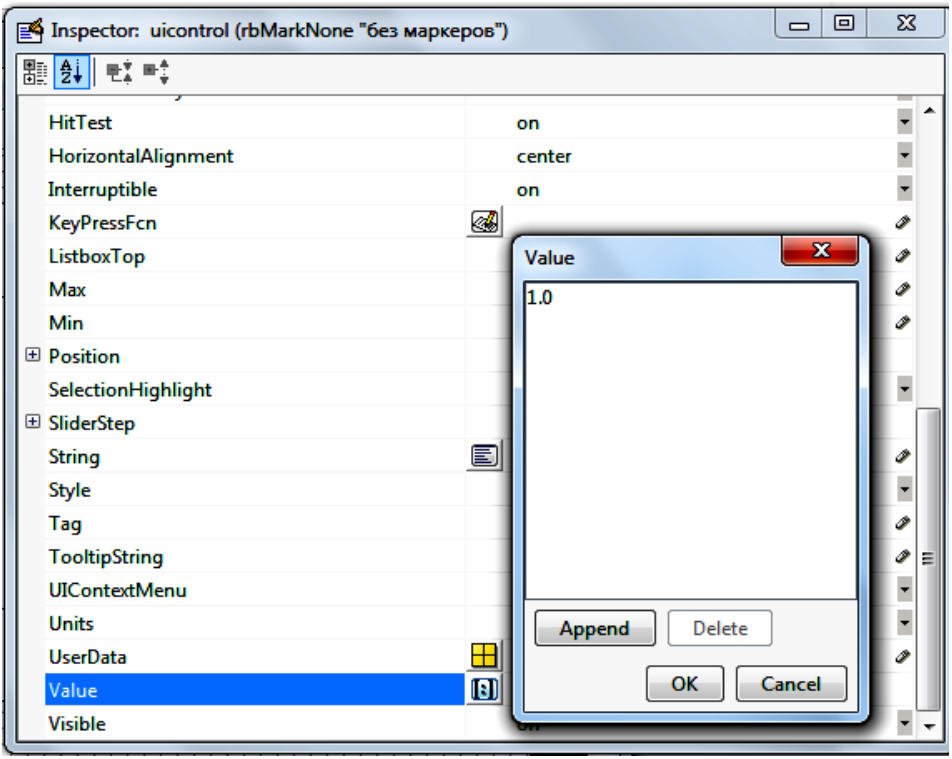


Рисунок 1.20 - Изменение значения свойства **Value**

В подфункции для обработки событий переключателей добавьте команды и комментарии, выделенные жирным шрифтом.

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

% Hint: get(hObject,'Value') returns toggle state of rbMarkCirc
% 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)

% --- Executes on button press in rbMarkSq.
function rbMarkSq_Callback(hObject, eventdata, handles)
% hObject    handle to rbMarkSq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of rbMarkSq
% 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)

% --- Executes on button press in rbMarkNone.
function rbMarkNone_Callback(hObject, eventdata, handles)
% hObject    handle to rbMarkNone (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of rbMarkNone
% Устанавливаем линию без маркеров
set(handles.line,'Marker','none')
% Переключатель Маркеры-круги должен быть выключен
set(handles.rbMarkCirc,'Value',0)
% Переключатель Маркеры-квадраты должен быть выключен
set(handles.rbMarkSq,'Value',0)
```

Протестируйте работу приложения с переключателями. Результат выбора переключателя **маркеры-квадраты** представлен на рис. 1.21.

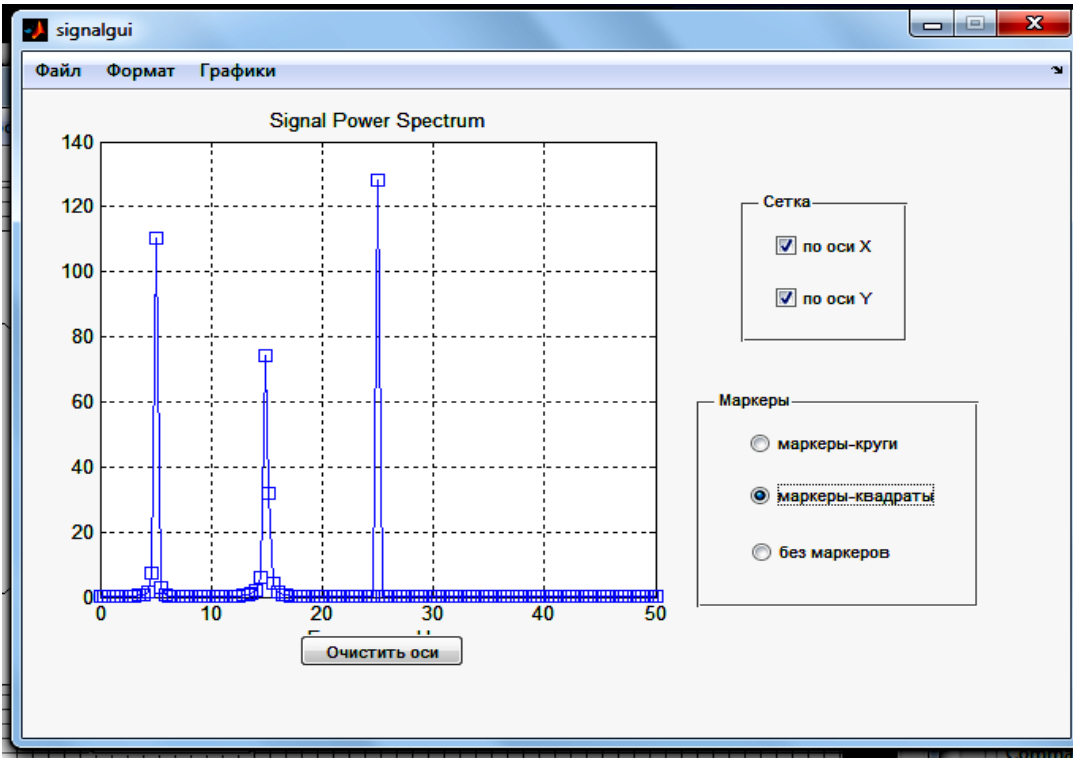


Рисунок 1.21. Результат выбора переключателя **маркеры-квадраты**.

2 СОЗДАНИЕ GUI-ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА ВАРИАБЕЛЬНОСТИ СЕРДЕЧНОГО РИТМА

2.1 Заготовка окна приложения rrgui

Создайте папку, в которой будете сохранять приложение и сделайте ее текущей. Перейдите в среду GUIDE, запустите мастер создания графического интерфейса и в появившемся диалоговом окне выберете пустой шаблон интерфейса **Blank GUI (Default)**. Добавьте в окно заготовки приложения элемент интерфейса **Axes**. При необходимости измените размер и расположение осей в соответствии с функцией элемента – вывод графиков RR-интервалов, гистограммы, ритмограммы, скатерограммы, графиков для спектрального и корреляционного анализа. Сохраните шаблон интерфейса под именем rrgui.

2.2 Меню приложения rrgui

Перейдите к созданию меню приложения с помощью редактора меню (**Menu Editor**) (рис. 2.1). Рекомендованные имена тегов указаны в таблице 2.1.

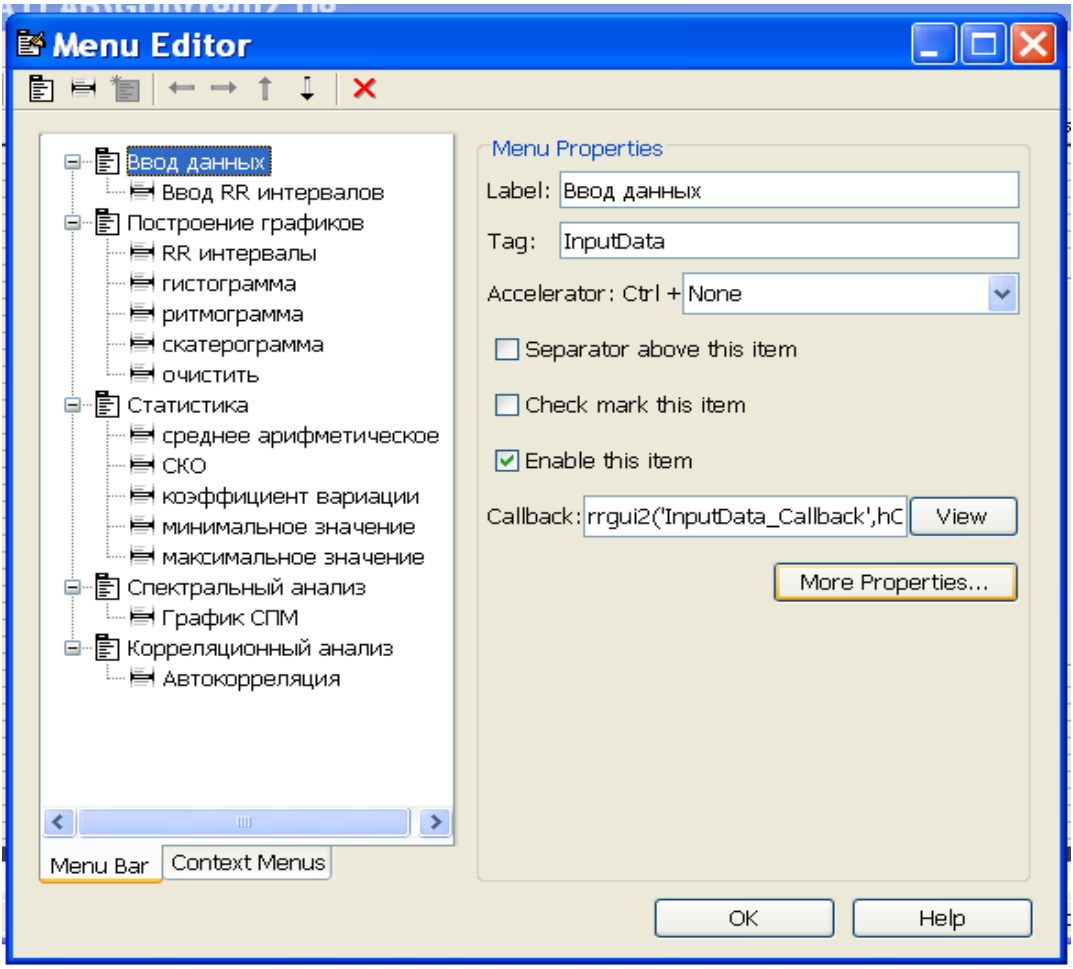


Рисунок 2.1 - Создание главного меню и подменю приложения rrgui

Для подпункта меню **очистить** установите флаг для свойства **Separator above this item,** что позволит отделить этот пункт от остальных.

Проверьте правильность заполнения тегов для каждого из пунктов и подпунктов меню во избежание создания безымянных подфункций. Нажатием кнопки **View** создайте функцию **Callback.**

Таблица 2.1 - Теги главного меню и подменю приложения rrgui

Пункт (подпункт) меню	**Значение свойства Tag**
Ввод данных	**InputData**
Ввод RR интервалов	**mnInputRR**
Посторенние графиков	**RRgraph**
RR интервалы	**mnPlotRR**
гистограмма	**mnHistRR**
ритмограмма	**mnRitmPlot**
скатерограмма	**mnScatPlot**
очистить	**mnClear**
Статистика	**Stat**
среднее арифметическое	**mnMeanRR**
СКО	**mnStdRR**
коэффициент вариации	**mnVarRR**
минимальное значение	**mnMinRR**
максимальное значение	**mnMaxRR**
Спектральный анализ	**Spectr**
график СПМ	**mnFftRR**
Корреляционный анализ	**Corr**
автокорреляция	**mnAcRR**

В функцию **rrgui_OpeningFcn** добавьте команды, выделенные жирным шрифтом**,** которые сделают неактивными пункты главного меню **Посторенние графиков, Статистика, Спектральный анализ** и **Корреляционный анализ** до ввода данных RR интервалов.

```
% --- Executes just before rrgui is made visible.
function rrgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to rrgui (see VARARGIN)
% Choose default command line output for rrgui
handles.output = hObject;
set(handles.RRgraph, 'Enable', 'Off')
set(handles.Stat,'Enable','Off')
set(handles.Spectr,'Enable','Off')
set(handles.Corr,'Enable','Off')
```

```
% Update handles structure
guidata(hObject, handles);
```

В функцию **mnInputRR_Callback** добавьте команды, выделенные жирным шрифтом, которые сделают активными меню **Посторенние графиков, Статистика, Спектральный анализ** и **Корреляционный анализ** после ввода данных RR интервалов.

```
function mnInputRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnInputRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
load('rrdat','rr');
set(handles.RRgraph, 'Enable', 'On')
set(handles.Stat,'Enable','On')
set(handles.Spectr,'Enable','On')
set(handles.Corr,'Enable','On')
```

Запустите приложение кнопкой **Run** и убедитесь в том, что приложение содержит все пункты и подпункты меню и нужные пункты неактивны до загрузки данных. Пока при обращении к подпунктам меню ничего не происходит, т.к. события **Callback** для них не запрограммированы. Если приложение работает правильно, то в командном окне нет сообщений об ошибках.

Закройте окно приложения при помощи кнопки с крестиком в правом верхнем углу приложения и продолжите работать в режиме редактирования.

2.3 Программирование событий для построения графиков

Запрограммируйте события для построения графиков. В функции для построения графиков добавьте команды и комментарии, выделенные жирным шрифтом. Поскольку в дальнейшем планируется изменение свойств линий графиков (изменение цвета и толщины) используйте структуру указателей.

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

```
%Создается новое поле в структуре handles и сохраняется в него значение
% переменной line для возможности обмена данными между подфункциями
handles.line=plot(rr),grid
% Сохраняется структура handles при помощи функции guidata
guidata(gcbo, handles);

function mnHistRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnHistRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
hist(rr),grid

function mnRitmPlot_Callback(hObject, eventdata, handles)
% hObject    handle to mnRitmPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
bar(rr),grid

function mnScatPlot_Callback(hObject, eventdata, handles)
% hObject    handle to mnScatPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
rrnumber=length(rr);
rrx=rr(2:2:rrnumber);
rry=rr(1:2:rrnumber-1);
handles.line=plot(rrx,rry,'b*'),grid
guidata(gcbo, handles);
function mnClear_Callback(hObject, eventdata, handles)
% hObject    handle to mnClear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla % убираем график
set(gca,'XGrid','off')% убираем сетку по оси x
set(gca,'YGrid','off')% убираем сетку по оси y

function mnAcRR_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to mnAcRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
MnRR=rr-mean(rr)
Acorr=xcorr(MnRR);
n=length(Acorr);
n1=n/2;
AcRR=Acorr./max(Acorr);
handles.line=plot(AcRR(n1-1:n)),grid
guidata(gcbo, handles);

function mnFftRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnFftRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
MnRR=rr-mean(rr);
FtRR=fft(MnRR);
RrRR=FtRR.*conj(FtRR);
handles.line=plot(RrRR),grid
guidata(gcbo, handles);
```

Сохраните файл и запустите приложение для проверки результата. Если приложение работает правильно, то в командном окне нет сообщений об ошибках, а выбор всех подпунктов пунктов главного меню **Ввод данных, Посторенние графиков, Спектральный анализ, Корреляционный анализ** приводит к получению нужного результата.

2.4 Расчет и вывод статистических характеристик ВСР

Рассчитайте и осуществите вывод статистических характеристик ВСР.

Выберите из меню элементов интерфейса **Panel** и расположите в окне приложения (рис. 2.2).

Откройте **Property Inspector** для компонента **Panel** и в свойстве **Title** укажите **Статистические характеристики.** Сохраните шаблон.

Для возможности вывода значений статистических характеристик разместите на панели **Статистические характеристики** пять элементов **Edit Text.** Откройте **Property Inspector** для

каждого из пяти элементов и в свойстве **Tag** укажите теги в соответствии с таблицей 2.2. Сохраните шаблон приложения.

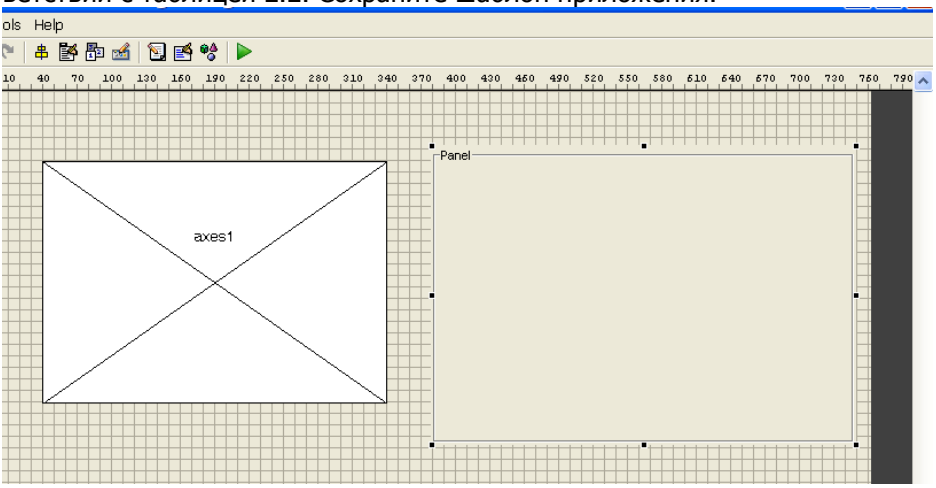


Рисунок 2.2 - Добавление элемента интерфейса **Panel**

Таблица 2.2. Теги для вывода статистических характеристик ВСР

Статистические характеристики	Значение свойства **Tag** компонента **Edit Text**
среднее арифметическое	**MeanOut**
СКО	**StdOut**
коэффициент вариации	**VarOut**
минимальное значение	**MinOut**
максимальное значение	**MaxOut**

В функции для расчета статистических характеристик ВСР добавьте команды, выделенные жирным шрифтом. Обратите внимание, что для вывода значения каждой из статистических характеристик после их названия обязательно должно быть место для числа. С помощью свойства **set** и структуры указателей **handles** строка с каждым из статистических параметров по тегу связывается с соответствующим элементом **Edit Text** для вывода значений на панели **Статистические характеристики.**

```
function mnMeanRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnMeanRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global rr
MeanRR=mean(rr);
StatRR=['Среднее значение RR интервалов ',num2str(MeanRR)];
set(handles.MeanOut,'String',StatRR)

 function mnStdRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnStdRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
StdRR=std(rr);
StatRR=['СКО RR интервалов     ',num2str(StdRR)];
set(handles.StdOut,'String',StatRR)
 function mnVarRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnVarRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
VarRR=(std(rr)/mean(rr))*100;
StatRR=['Коэффициент вариации      ',num2str(VarRR)];
set(handles.VarOut,'String',StatRR)

function mnMinRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnMinRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
MinRR=min(rr);
StatRR=['Минимальное значение      ',num2str(MinRR)];
set(handles.MinOut,'String',StatRR)

function mnMaxRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnMaxRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
MaxRR=max(rr);
StatRR=['Максимальное значение      ',num2str(MaxRR)];
set(handles.MaxOut,'String',StatRR)
```

Сохраните результат и запустите приложение. Проверьте работу пункта главного меню **Статистика.** Результат должен соответствовать рисунку 2.3.

Рисунок 2.3 - Результат работы пункта главного меню **Статистика**

Модернизируйте приложение, добавив на панель **Статистические характеристики** кнопку **Очистить** для удаления полученных значений (рис. 2.4). Значение свойства **Tag** для кнопки установите **pbtClear**

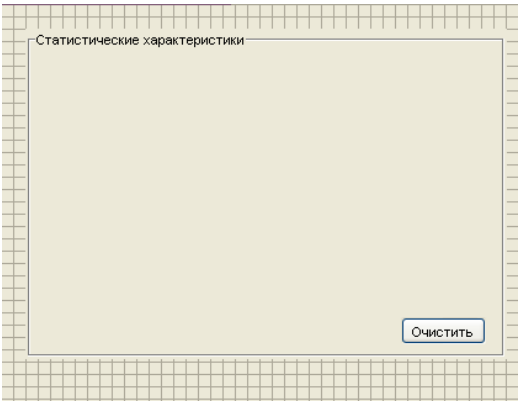


Рисунок 2.4 - Панель **Статистические характеристики** с кнопкой **Очистить**

Следует учесть, что кнопка **Очистить** должна быть неактивна с момента запуска приложения до вывода на панель **Статистические характеристики** любого из их значений**.** После очистки панели кнопка снова должна стать неактивной.

Запрограммируйте события, связанные с работой кнопки **Очистить.** Для этого в подфункцию **pbtClear_Callback** добавьте выделенные жирным шрифтом команды и комментарии.

```
function pbtClear_Callback(hObject, eventdata, handles)
% hObject    handle to pbtClear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% удаление содержимого областей ввода
  set(handles.MaxOut, 'String', '')
  set(handles.MinOut, 'String', '')
  set(handles.MeanOut,'String', '')
  set(handles.StdOut, 'String', '')
  set(handles.VarOut, 'String', '')
 % делаем кнопку Очистить недоступной
  set(handles.pbtClear, 'Enable', 'off')
```

Во все подфункции для расчета статистических характеристик добавьте выделенную строку по аналогии с указанной ниже подфункцией **mnMeanRR_Callback**

```
function mnMeanRR_Callback(hObject, eventdata, handles)
% hObject    handle to mnMeanRR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rr
MeanRR=mean(rr);
StatRR=['Среднее значение RR интервалов   ',num2str(MeanRR)];
set(handles.MeanOut,'String',StatRR)
set(handles. pbtClear,'Enable','On')
```

В подфункцию **rrgui_OpeningFcn** добавьте выделенную строку

```
function rrgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
set(handles.Corr,'Enable','Off')
```

```
set(handles. pbtClear,'Enable','Off')
% Update handles structure
guidata(hObject, handles);
```

Запустите приложение для проверки правильности работы кнопки **Очистить**

2.5 Использование списка для выбора цвета линии графика

Модернизируйте интерфейс приложения, 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 **Pop-up Menu** (рис.2.5). В редакторе свойств установите свойство **Tag** в значение **pmColor**

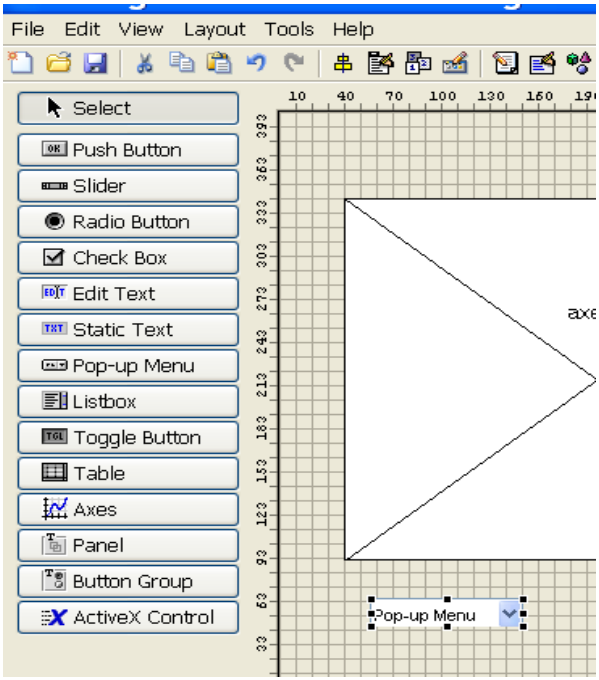


Рисунок 2.5 – Добавление раскрывающегося списка **Pop-up Menu**

Элементами раскрывающегося списка являются строки, ко-

торые вводятся в инспекторе свойств. Нажмите кнопку в строке со свойством **String** раскрывающегося списка, появится окно **String**. Наберите в нем строки **"синий"**, **"красный"**, **"зеленый"** (без кавычек), разделяя их при помощи клавиши **Enter** (рис. 2.6).

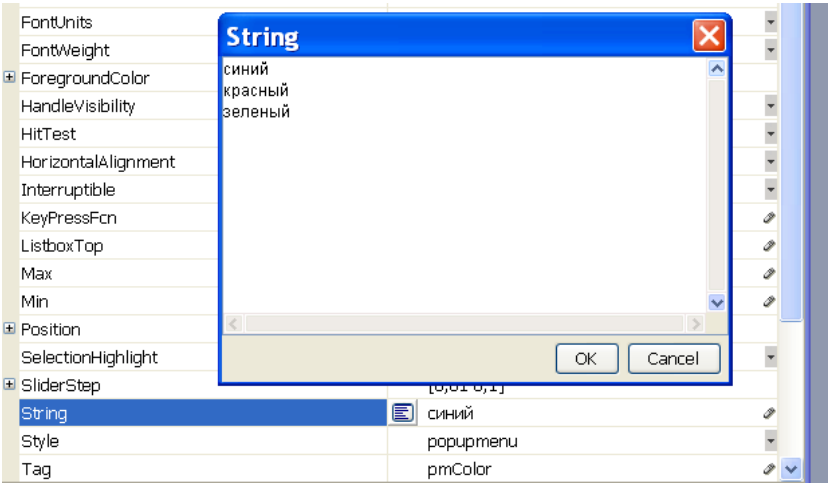


Рисунок 2.6 - Окном свойства **String** раскрывающегося списка

Запустите приложение 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.

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

% Hints: contents = cellstr(get(hObject,'String')) returns pmColor
```

```
contents
% as cell array
%   contents{get(hObject,'Value')} returns selected item from pmCol-
or
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
```

Запустите приложение и проверьте работу раскрывающегося списка. Самостоятельно добавьте пользователю возможность выбора желтого цвета (обозначение для желтого цвета - '**y**')

2.6 Контекстное меню для выбора цвета линии графика

2.6.1 Особенности конструирования контекстного меню

Элементы, 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 элемента. Конструирование контекстного меню состоит из трех этапов: создания его в редакторе меню, последующем связывании меню с элементом интерфейса и программировании событий **Callback** пунктов меню.

2.6.2 Создание контекстного меню

Перейдите к вкладке **Context Menus** в редакторе меню **Menu Editor** и нажмите кнопку создания контекстного меню **New Context Menu**, в редакторе меню появится 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 и имена в соот-

ветствии с таблицей 2.3. В результате навигатор меню должен содержать структуру, приведенную на рис. 2.7.

Таблица 2.3 - Теги пунктов контекстного меню

Пункт(подпункт) контекстного меню	**Значение свойства Tag**
cmLine	**cmLine**
синий	**cmLineBlue**
красный	**cmLineRed**
зеленый	**cmLineGreen**

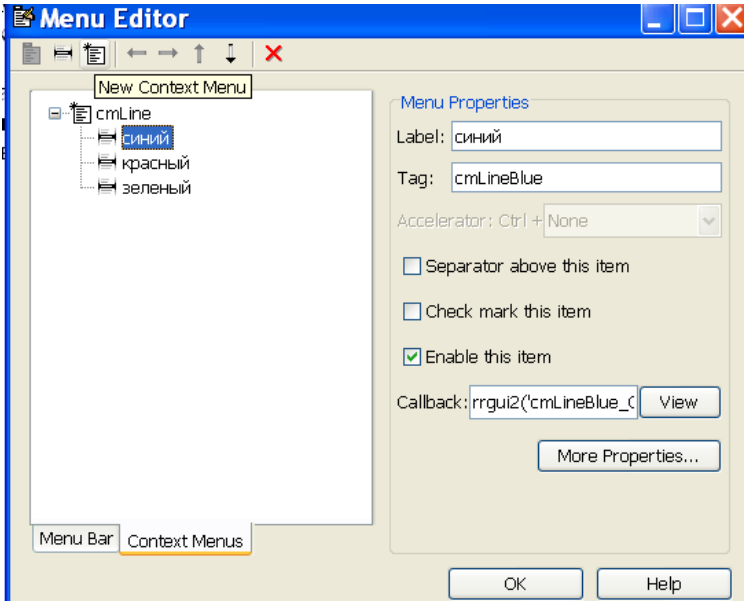


Рисунок 2.7 - Отображение контекстного меню в редакторе меню

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 выборе, например, подпункта меню RR интервалы, "не знает" о том, что у нее есть собственное контекстное меню. Следующий этап состоит в связывании линии с созданным меню **cmLine**.

2.6.3 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 в нашем случае, обладал контекстным меню, следует установить свойству **UIContextMenu** значение указателя на меню **cmLine**, содержащееся в структуре **handles**. Построение графика линией в приложении rrgui производится при выборе подпунктов меню **RR-интервалы**, **скатерограмма**, **график СПМ** и **автокорреляция**. Присвойте свойству линии **UIContextMenu** требуемое значение в каждой из четырех подфункций, обрабатывающих события выбора этих подпунктов меню. Для этого ниже строки с функцией **guidata** (**guidata(gcbo, handles);**) добавьте строки:

```
% Связывание контекстного меню cmLine с линией графика
set(handles.line, 'UIContextMenu', handles.cmLine)
```

Запустите приложение rrgui, постройте линию любым доступным способом и убедитесь, что щелчок правой кнопкой мыши по линии приводит к появлению контекстного меню с пунктами **синий, красный, зеленый**. Выбор пунктов пока не приводит к изменению цвета линии, т.к. следует запрограммировать событие **Callback** каждого пункта.

2.6.4 Программирование контекстного меню

Обработка событий **Callback** пунктов контекстного меню производится аналогично программированию меню приложения. Перейдите в редактор М-файлов нажатием кнопки View и дополните подфункции обработки событий выбора пунктов 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:

```
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)
```

2.7 Использование полосы скроллинга

Усовершенствуйте интерфейс приложения, предоставив пользователю возможность устанавливать ширину линии при помощи полосы скроллинга. Добавьте полосу скроллинга (элемент интерфейса **Slider**) в окно приложения (рис. 2.8) и задайте название **scrWidth** в ее свойстве **Tag**.

Теперь следует определить соответствие между положени-

ем 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.

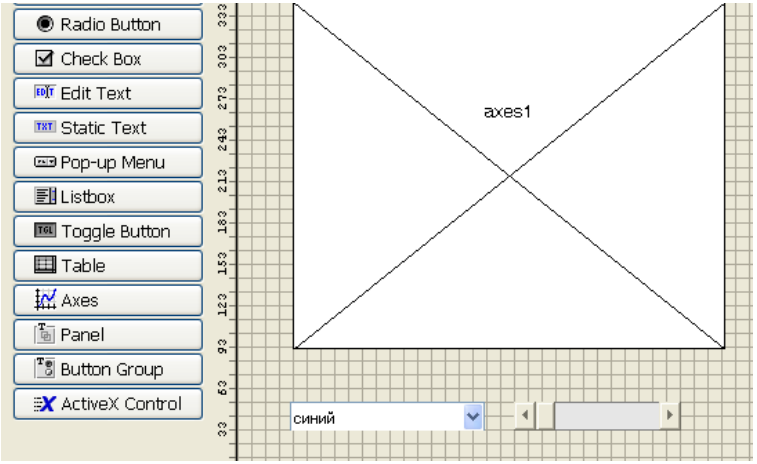


Рисунок 2.8 – Добавление полосы скроллинга

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 (рис. 2.9).

Рисунок 2.9 - Установка значений свойства **SliderStep**

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 выделенные жирным шрифтом операторы и комментарии.

```
function scrWidth_Callback(hObject, eventdata, handles)
% hObject    handle to scrWidth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% Получаем текущее значение скроллбара
w=get(hObject,'Value');
% Устанавливаем в качестве толщины линии округленное значение скроллбара
set(handles.line,'LineWidth',round(w));
```

Запустите приложение и проверьте работу полосы скроллинга.

2.8 Текстовые пояснения

Снабдите полосу скроллинга и раскрывающийся список текстовыми пояснениями "**Толщина линии**" и "**Цвет линии**", а так же добавьте общий заголовок **"АНАЛИЗ ВСР"**, как показано на рисунке 2.10. Используйте элементы интерфейса **Static Text.**

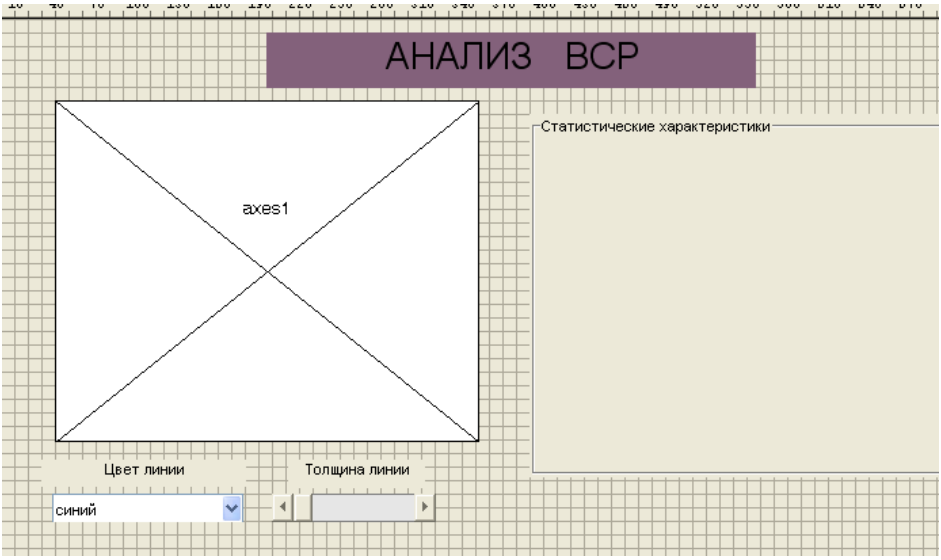


Рисунок 2.10 – Использование текстовых пояснений

2.9 Диалоговое окно подтверждения

Некоторые действия приложения требуют повторного подтверждения пользователя. Например, пользователь приложения rrgui может случайно выбрать пункт меню **Очистить**, предназначенный для очистки осей. Следует вывести диалоговое окно, в котором пользователь укажет, действительно ли он желает очистить оси.

Усовершенствуйте обработку выбора пункта меню **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 **rrgui**. Используйте условный оператор **if** и функцию **strcmp** для сравнения выходного аргумента **questdlg** со строкой **Yes.**

```
function mnClear_Callback(hObject, eventdata, handles)
% hObject    handle to mnClear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
button = questdlg('Очистить оси?','rrgui');
if strcmp(button, 'Yes')
 cla % убираем график
```

```
set(gca,'XGrid','off')% убираем сетку по оси x
set(gca,'YGrid','off')% убираем сетку по оси y
 end
```

Выбор пункта меню **Очистить** должен привести к появлению диалогового окна, изображенного на рис. 2.11. Выбор пользователя определяет дальнейшие действия приложения rrgui.

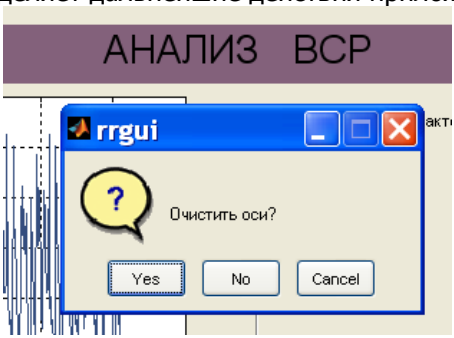


Рисунок 2.11 - Результата выбора пункта меню **Очистить**

На рисунке 2.12 приведен вид готового приложения **rrgui.**

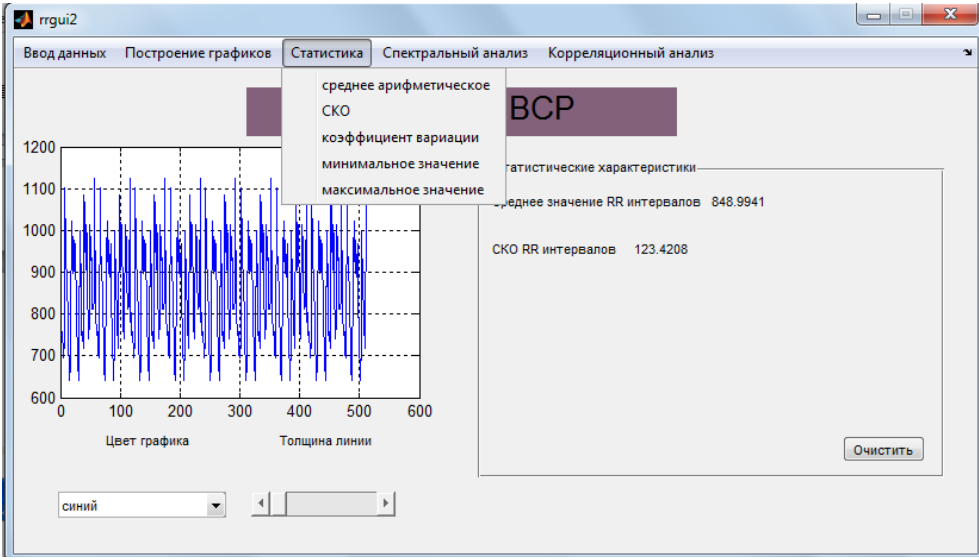


Рисунок 2.12 - Вид готового приложения **rrgui**

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Вариабельность сердечного ритма. Теоретические аспекты и возможности клинического применения/ Институт медико-биологических проблем, Московская медицинская академия им. И.М. Сеченова, Научно-исследовательская лаборатория «Динамика», Москва, Санкт-Петербург, 2002

2. Вариабельность сердечного ритма: Теоретические аспекты и практическое применение, Материалы V всероссийского симпозиума, Ижевск, 2011

3. Бадриев И.В., Бандеров В.В., Задворнов О.А. Разработка графического интерфейса пользователя в среде MATLAB. Учебное пособие – Казань: Казанский государственный университет, 2010. – 113 с

4. Селиванова И.А. Создание графического интерфейса пользователя в MATLAB/ Екатеринбург: ГОУ ВПО УГТУ - УПИ, 2006, 34 с

5. Дьяконов В.П. MATLAB 6.5 SP1/7+ Simulink 5/6 в математике и моделировании. - М.: СОЛОН-Пресс, 2005. - 576с.: ил.

6. Дьяконов В.П. MATLAB 6.5 SP1/7.0 Simulink 5/6 Обработка сигналов и проектирование фильтров. - М.: СОЛОН-Пресс, 2005. - 576с.: ил.

7. Рангаян Р.М. Анализ биомедицинских сигналов. Практический подход – М.: ФИЗМАТЛИТ, 2007. – 440 с.

8. А.В. Литвин, В.В. Головкин, К.А. Мороз, И.К. Цырий, Статистический анализ вариабельности сердечного цикла. Методические указания к лабораторному практикуму по дисциплине «Компьютерные технологии в медико-биологической практике». – Ростов-на-Дону, ГОУ ВПО ДГТУ, 2007

9. Литвин А.В., Мороз К.А. Спектральный анализ сигналов: метод. указания к лабораторному практикуму по дисциплине «Компьютерные технологии медико-биологических исследований», - Ростов н/Д: Издательский центр ДГТУ, 2011. – 15 с

10. Новиков Д.А., Новочадов В.В., Статистические методы в медико-биологическом эксперименте (типовые случаи). – Волгоград, ВолГМУ, 2005