10 *Создание графических приложений в среде Scilab*

Scilab позволяет создавать не только обычные программы для автоматизации расчетов, но и визуальные приложения, которые будут запускаться в среде Scilab. Основным объектом в среде Scilab является графическое окно.

10.1 Работа с графическим окном

Для создания пустого графического окна служит функция *figure*.

```
F=figure();
```

В результате выполнения этой команды будет создано данном графическое окно с именем **objfigure1**. По умолчанию первое окно получает имя **objfigure1**, второе **objfigure2** и т.д. Указатель на графическое окно[1] записывается в переменную *F*. Размер и положение окна на экране компьютера можно задавать с помощью параметра

```
'position',[x y dx dy],
```

где

- *x, y* положение верхнего левого угла окна (по горизонтали и вертикали соответственно) относительно верхнего левого угла экрана;
- *dx* размер окна по горизонтали (ширина окна) в пикселях;
- *dy* размер окна по вертикали (высота окна) в пикселях.

Параметры окна можно задавать одним из двух способов.

1. Непосредственно при создании графического окна задаются его параметры. В этом случае обращение к функции figure имеет вид

```
F=figure('Свойство1', 'Значение1', 'Свойство2', 'Значение2', ...,'Свойствоn', 'Значениеn')
```

здесь *'Свойство1'* название первого параметра, *'Значение1'* его значение, *'Свойство2'* название второго параметра, *'Значение2'*[2] значение второго параметра и т.д.

Например, с помощью команды

```
F=figure('position', [10 100 300 200]);
```

будет создано окно, представленное на рис. 10.1.

2. После создания графического окна с помощью функции

```
set(f,'Свойство','Значение')
```

устанавливается значение параметров, здесь *f* - указатель на графическое окно, *'Свойство'* - имя параметра, *'Значение'* его значение.

Следующие две строки задают месторасположение и размер окна (рис. 10.2).

```
f=figure();
set(f,'position',[20,40,600,450])
```

Для изменения заголовка окна используется параметр `'figure_name', 'name'`, определяющий заголовок окна (*'name'*). На листинге 10.1 приведен пример создания окна с именем **FIRST WINDOWS** (см. рис. 10.3).

---

1 Под указателем мы будем понимать переменную, в которой хранится адрес окна или другого объекта.

2 `Значениеi` будет использоваться в кавычках, если значением параметра является строка, если же значением параметра является число, то кавычки использовать не надо.

Рисунок 10.1. Первое графическое окно

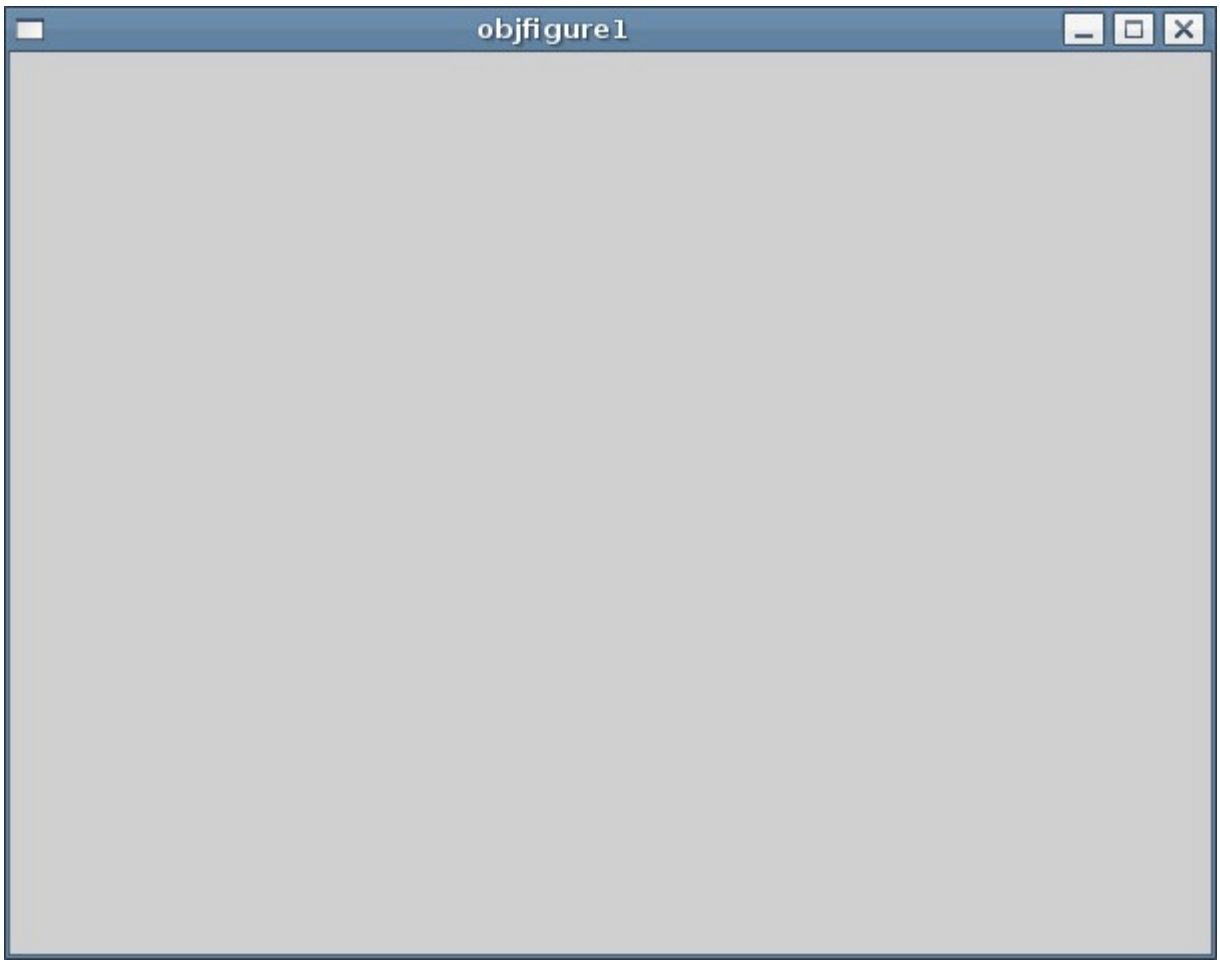

Рисунок 10.2. Графическое окно определенного размера и месторасположения

```
f=figure();
set(f,'position',[20,40,600,450]);
set(f,'figure_name','FIRST WINDOW');
```

Листинг 10.1. Создание окна с именем FIRST WINDOW

*Рисунок 10.3. Окно с заголовком **FIRST WINDOW***

Это же окно получить с помощью одной строки

```
f=figure('position',[20,40,600,450],'figure_name','FIRST WINDOW');
```

Графическое окно можно закрыть с помощью функции

```
close(f),
```

здесь *f* указатель на окно .

Удаляется окно с помощью функции

```
delete(f),
```

где *f* указатель на окно.

10.2 Динамическое создание интерфейсных элементов. Описание основных функций

В Scilab используется динамический способ создания интерфейсных компонентов. Он заключается в том, что на стадии выполнения программы могут создаваться (и удаляться) те или иные элементы управления (кнопки, метки, флажки и т.д.) и их свойствам присваиваются соответствующие значения.

Для создания любого интерфейсного компонента с заданными свойствами используется функция uicontrol, возвращающая указатель на формируемый компонент:

```
C=uicontrol(F, 'Style', 'тип_компонента', 'Свойство_1',
 Значение_1, 'Свойство_2', Значение_2,...,
 'Свойство_k',Значение_к);
```

Здесь

C указатель на создаваемый компонент ;

F указатель на объект ,внутри которого будет создаваться компонент (чаще всего этим компонентом будет окно); первый аргумент функции *uicontrol* не является обязательным, и если он отсутствует, то родителем (владельцем) создаваемого компонента является текущий графический объект текущее графическое окно ;

'Style'- служебная строка *Style*, указывает на стиль создаваемого компонента(символьное имя);

'тип_компонента' - определяет, к какому классу принадлежит создаваемый компонент, это может быть *PushButton*, *Radiobutton*, *Edit*, *StaticText*, *Slider*, *Panel*, *Button Group*, *Listbox* или др компоненты, это свойство будет указываться для каждого из компонентов;

'Свойство_к', Значение_к определяют свойства и значения отдельных компонентов , они будут описаны ниже конкретно для каждого компонента.

У существующего интерфейсного объекта можно изменить те или иные свойства с помощью функции *set*:

```
set(C,'Свойство_1',Значение_1, 'Свойство_2', Значение_2, ...,
'Свойство_к', Значение_к)
```

Здесь

C указатель на динамический компонент , параметры которого будут меняться ; *C* может быть и вектором динамических элементов, в этом случае функция set будет задавать значения свойств для всех объектов *C(i)*;

'Свойство_к', Значение_к изменяемые параметры и их значения .

Получить значение параметра компонентов можно с помощью функции *get* следующей структуры:

```
get(C,'Свойство')
```

Здесь

C указатель на динамический интерфейсный компонент ,значение параметра которого необходимо узнать;

'Свойство' имя параметра, значение которого нужно узнать.

Функция возвращает значение параметра.

Далее мы поговорим об особенностях создания различных компонентов.

10.2.1 Командная кнопка

Командная кнопка типа **PushButton** создается с помощью функции *uicontrol*, в которой параметру *'Style'* необходимо присвоить значение *'pushbutton'*. По умолчанию она не снабжается никакой надписью, имеет серый цвет и располагается в левом нижнем углу фигуры, Надпись на кнопке можно установить с помощью свойства *String* (см. листинг 10.2 и рис 10.4).

```
//Создаем окно
d=figure();
//Создаем кнопку, устанавливая свойство Style.
dbt=uicontrol(d,'Style','pushbutton');
//Изменяем надпись YES на кнопке
set(dbt,'String','YES');
```

Листинг 10.2. Создание окна с кнопкой

Рисунок 10.4. Кнопка YES в окне

Модифицируем программу создания кнопки, задав дополнительно значения некоторых свойств:

- месторасположение и заголовок окна;
- надпись на кнопке;
- месторасположение кнопки.

Текст программы приведен на листинге 10.3, а на рис. 10.5 можно увидеть окно, которое получилось в результате работы этой программы.

```
//Создаем окно.
f=figure();
//Определяем месторасположение окна.
set(f,'position',[0,0,250,100])
//Определяем имя (заголовок) окна.
set(f,'figure_name','Окно с кнопкой');
//Создаем кнопку (style – pushbutton), надпись на кнопке –
// Button, позиция кнопки определяется параметром position.
Button=uicontrol('style','pushbutton','string','Кнопка',...
'position',[50,50,100,20]);
```

Листинг 10.3. Определение свойств кнопки

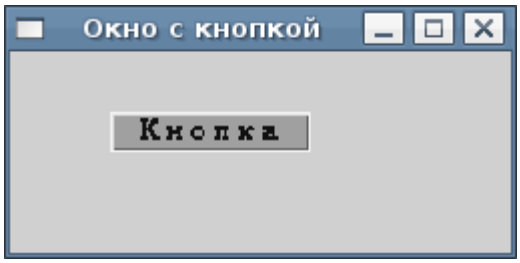

Рисунок 10.5. Окно с кнопкой

Теперь при щелчке на кнопке вокруг ее надписи появляется пунктирный прямоугольник, свидетельствующий о том, что кнопка "находится в фокусе". Щелчок за пределами поверхности кнопки выведет ее из фокуса, и пунктирная рамка пропадет. Главным назначением командной кнопки является вызов функции, реагирующей на щелчок по кнопке.

Щелчок по кнопке генерирует событие *Callback*, которое указывается как параметр функции *uicontrol*. Значением параметра *Callback* является строка с именем функции, вызываемой при щелчке по кнопке. В этом случае функция *uicontrol* становится такой

```
Button=uicontrol('style', 'pushbutton', 'string', 'Button',
```

```
 'Callback', 'Function');
```

Здесь *Function* имя вызываемой при наступлении события *Callback* функции.

В качестве примера рассмотрим окно с кнопкой, при щелчке по которой появляется окно с графиком функции *y=sin(x)* (см. листинг 10.4). После запуска этой программы появится окно, представленное на рис. 10.6, при щелчке по кнопке *Button* вызывается обработчик события функция *gr_sin*, в результате - появляется окно, изображенное на рис. 10.7.

```
f=figure();
set(f,'position',[0,0,250,100])
set(f,'figure_name','Grafik');
//Создаем кнопку, которая при щелчке по ней мышкой, вызывает
// функцию gr_sin.
Button=uicontrol('style','pushbutton','string','Button',...
 'position',[50,50,100,20],'CallBack','gr_sin');
function y=gr_sin()
x=-5:0.2:5;
y=sin(x);
plot(x,y);
xgrid();
endfunction
```

Листинг 10.4. Пример кнопки с обработчиком события CallBack

Рисунок 10.6. Окно программы

10.2.2 Метка

Следующим наиболее часто используемым компонентом является метка текстовое поле для отображения символьной информации. Для определения метки значения параметра *'Style'* в функции *uicontrol* должно иметь значение *'text'*. Компонент предназначен для вывода символьной строки (или нескольких строк). Выводимый на метку текст значение параметра *'String'* может быть изменен только из программы.

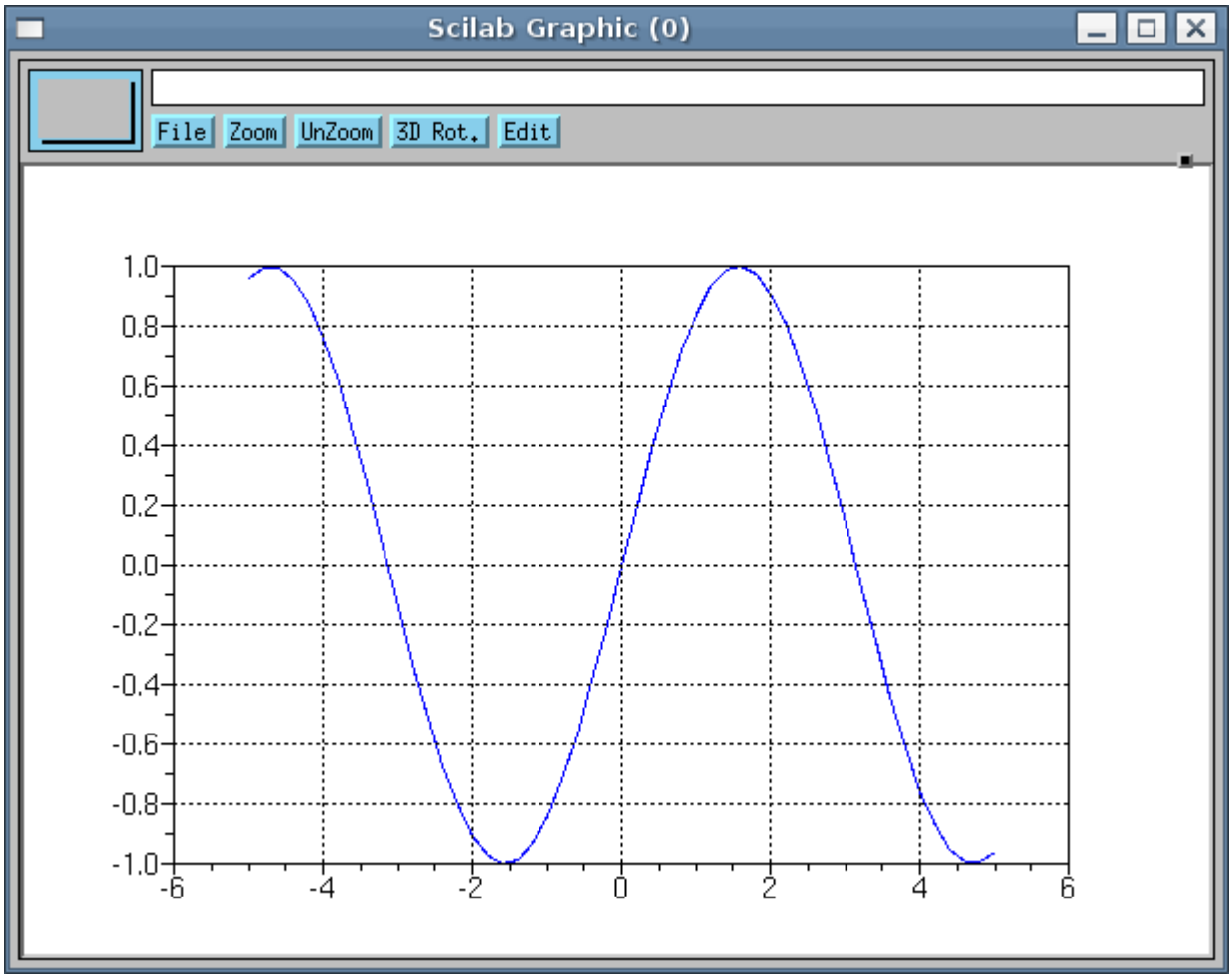

Рисунок 10.7.График функции y=sin(x)

Рассмотрим пример создания текстового поля (метки) с помощью функции uicontrol (см. листинг 10.5 и рис. 10.8):

```
f=figure();
uicontrol('Style','text','Position',
[10,130,150,20],'String',...
 'Metka');
```

Листинг 10.5. Создание метки

Одним из основных свойств метки является горизонтальное выравнивание текста, которое определяется свойством *HorizontalAlignment*. Это свойство может принимать одно из следующих значений:

- *left* выравнивание текста по левому краю ;
- *center* выравнивание текста по центру (значение по умолчанию);
- *right* выравнивание по правому краю .

В качестве примера рассмотрим окно, содержащее 4 текстовых поля с разными значениями свойства *HorizontalAlignment*. Текст программы представлен в листинге 10.6, а окно с четырьмя метками на рис .10.9.

Рисунок 10.8. Окно с меткой

```
hFig=figure();
set(hFig,'Position',[50,50,300,200]);
hSt1=uicontrol('Style','text','Position',[30,30,150,20],...
'String', 'Metka 1');
set(hSt1,'BackgroundColor',[1 1 1]);
set(hSt1,'HorizontalAlignment','left');
hSt2=uicontrol('Style', 'text', 'Position', [30,60,150,20],...
'HorizontalAlignment', 'center', 'BackgroundColor', [1 1
1],...
'String', 'Metka 2');
hSt3=uicontrol('Style','text','Position',[30,90,150,20],...
'HorizontalAlignment','right','BackgroundColor',[1 1 1],...
'String','Metka 3');
hSt4=uicontrol('Style','text','Position',[30,120,150,20],...
'BackgroundColor',[1 1 1],'String','Metka 4');
```

Листинг 10.6. Создание нескольких меток

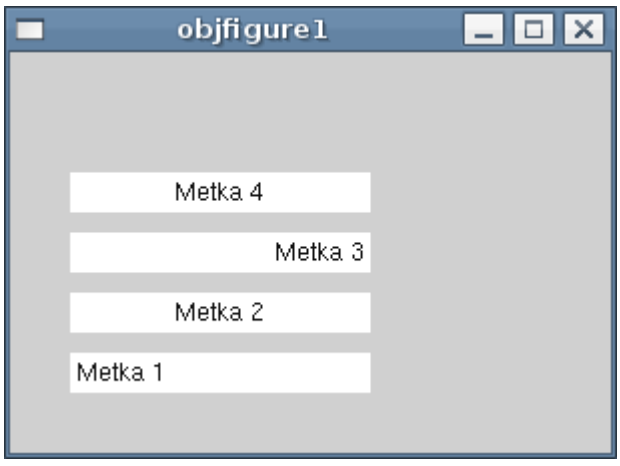

Рисунок 10.9.Окно с несколькими метками

10.2.3 Компоненты Переключатель и Флажок

Рассмотрим еще два компонента переключатель и флажок, которые позволяют

переключаться между состояниями или выключать одно из свойств.

У флажка свойство *'Style'* принимает значение *'checkbox'*, у переключателя свойство *'Style'* должно быть установлено в *'radiobutton'*.

Индикатором альтернативных комбинаций является переключатель (Radiobutton), который также создается с помощью функции uicontrol. Пример создания переключателя представлен на листинге 10.7 и рис. 10.10.

```
hFig=figure();
R=uicontrol('Style','radiobutton','String','name','value',
1,...,
 'Position', [25,150,70,30]);
```

Листинг 10.7. Создание переключателя

Рисунок 10.10.Окно с переключателем

При создании переключателя должно быть задано его состояние (параметр *'value'*), переключатель может быть активен (значение *'value'* равно *1*) или нет (значение *'value'* равно *0*). Задать значение свойства *'value'* можно также и с помощью функции *set*. Например,

```
set(Rb,'value',0)
```

Получить значение свойства *'value'* можно с помощью функции *get*.

Переключатель может реагировать на событие *'callback'*, и вызывать на выполнение определенную функцию. В этом случае создать кнопку можно с помощью вызова следующей функции *uicontrol*

```
r1=uicontrol('Style','radiobutton','String','sin(x)','value',
0,'callback','F1');
```

Здесь *F1* имя функции , которая будет вызываться при щечке по переключателю . Однако, когда Вы будете писать функцию - обработчик события *'callback'* переключателя, то следует помнить, что при щелчке по переключателю автоматически происходит смена его состояния.

Проиллюстрируем применение переключателей на примере программы, в которой с помощью переключателя можно выбрать функцию, график которой будет воспроизводиться в отдельном графическом окне при щелчке по кнопке **Plot**(см. листинг 10.8 и рис. 10.11).

На рис. 10.11 представлено окно приложения. При щелчке по кнопке **Plot** будет создано диалоговое окно в котором будут изображены графики выбранных с помощью переключателей функций (см. рис. 10.12). Если ни одна из функций не выбрана, то при щелчке по кнопке **Plot** ничего не происходит. Кнопка **Close** закрывает приложение. Изменение состояния переключателей происходит автоматически при щелчке по ним.

```
//Создаем графическое окно.
```

```
hFig=figure('Position',[50,50,200,200]);
//Создание радиокнопок
hRb1=uicontrol('Style','radiobutton','String','sin(x)',...
'value',1, 'Position',[25,100,60,20]);
hRb2=uicontrol('Style','radiobutton','String','cos(x)',...
'value',1, 'Position',[25,140,60,20]);
//Создаем кнопку с именем Plot, которая с помощью обработчика
//Radio строит график функции в соответствии с положением
// переключателей.
Button=uicontrol('style','pushbutton','string','Plot',...
'position',[20,50,80,20],'CallBack','Radio');
//Создаем кнопку с именем Close, которая с помощью обработчика
//Final закрывает окно.
Button1=uicontrol('style','pushbutton','string','Close',...
'position',[20,25,80,20],'CallBack','Final');
//Функция Radio, реагирующая на щелчок по кнопке
function Radio()
newaxes;
x=-2*%pi:0.1:2*%pi;
if get(hRb1,'value')==1 //Если активна первая кнопка,
y=sin(x);
plot(x,y,'-r');      //то построение синусоиды
xgrid();
end;
if get(hRb2,'value')==1 //Если активна вторая кнопка
y=cos(x);
plot(x,y,'-b');      //то построение косинусоиды
xgrid();             //Нанесение сетки на график
end;
endfunction
//Функция, отвечающая за кнопку Close и закрывающая окно.
function Final()
close(hFig);
endfunction
```

Листинг 10.8. Пример работы с переключателями

Компонент флажок используется для индикации неальтернативных комбинаций. Генерация события *'Callback'* и автоматическое выделение кнопки происходят при щелчке на квадратике или сопровождающей его надписи. Если флажок включен, то значение свойства '*value'* равно 1. Щелчок по флажку автоматически изменяет состояние на противоположное. Использование флажка аналогично переключателю.

Рисунок 10.11. Окно приложения

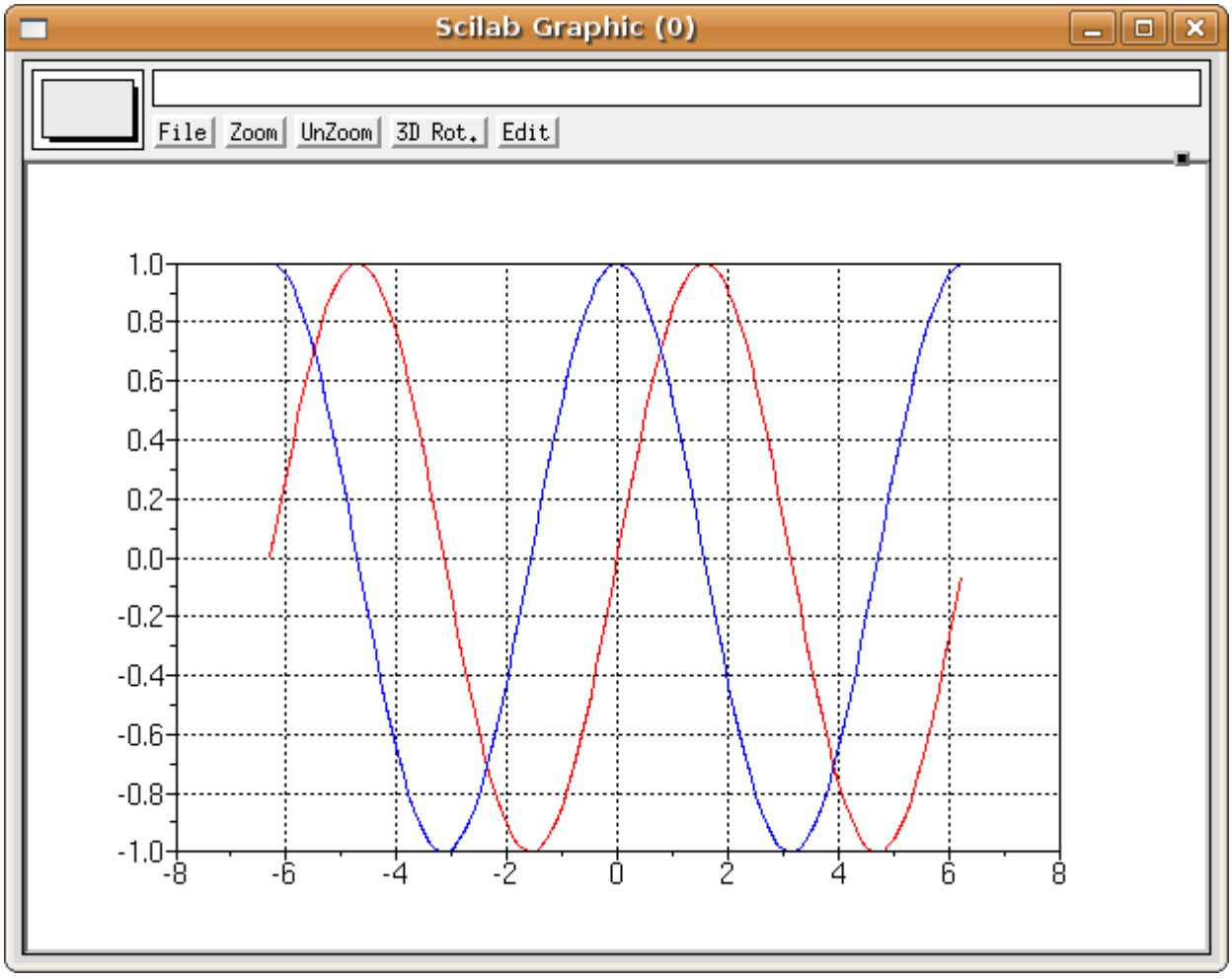

Рисунок 10.12. Построенные в отдельном окне графики функций

10.2.4 Компонент окно редактирования

Интерфейсный элемент окно редактирования (у того компонента свойство *'Style'* должно принимать значение *'edit'*) может использоваться для ввода и вывода символьной информации. Текст, набираемый в окне редактирования, можно корректировать. При работе с компонентом можно использовать операции с буфером обмена. Процедура ввода, завершаемая нажатием клавиши **Enter**, генерирует событие Callback.

Строка ввода определяется параметром *'String'*, которое определяет находящийся в компоненте текст. Для нормального функционирования компонента этот параметр необходимо обязательно задавать при определении компонента с помощью функции *uicontrol*. Изменить значение этого свойства можно с помощью функции *set*, а считать значение этого свойства можно с помощью функции *get*.

Вводимый текст может быть прижат к левому или правому краю окна ввода, если задать соответствующее значение свойства *HorizontalAlignment* (по аналогии с компонентом «Метка»). Если вводимый текст представляет собой числовое значение, которое должно быть использовано в работе программы, то содержимое свойства string переводится в числовой формат с помощью функции *eval* (можно было воспользоваться и функцией *evstr*) (будет рассмотрено далее на примере квадратного уравнения).

В качестве примера рассмотрим работы с несколькими компонентами рассмотрим следующую задачу.

ЗАДАЧА 10.1

Написать программу решения квадратного или биквадратного уравнения. Выбор типа уравнения будем проводить с помощью компонента Переключатель.

Программа с комментариями представлена на листинге 10.9.

```
f=figure(); //Создание графического объекта
//Устанавливаем размер окна
set(f,'position',[0,0,700,300])
//Устанавливаем заголовок окна.
set(f,'figure_name','УРАВНЕНИЕ');
//Создание текстовых полей для подписей полей ввода
// коэффициентов
//Подпись A=.
lab_a=uicontrol(f,'style','text','string','A=','position',...
[50, 250, 100, 20]);
//Подпись B=.
lab_b=uicontrol(f,'style','text','string','B=','position',...
[150, 250, 100, 20]);
//Подпись C=.
lab_c=uicontrol(f,'style','text','string','C=','position',...
[250, 250, 100, 20]);
//Поле редактирования для ввода коэффициента a.
edit_a=uicontrol(f,'style','edit','string','1','position',...
[50, 230, 100, 20]);
//Поле редактирования для ввода коэффициента b.
edit_b=uicontrol(f,'style','edit','string','2','position',...
[150, 230, 100, 20]);
//Поле редактирования для ввода коэффициента c.
edit_c=uicontrol(f,'style','edit','string','1','position',...
[250, 230, 100, 20]);
//Текстовое поле, определяющее вывод результатов
textresult=uicontrol(f,'style','text','string','','position',.
..
[5, 80, 650, 20]);
//Флажок, отвечающая за выбор типа уравнения
radio_bikv=uicontrol('style','radiobutton','string',...
'Биквадратное уравнение?', 'value',1,'position',...
```

```
    [100,100,300,20]);
    BtSolve=uicontrol('style','pushbutton','string','Решить',...
    'Callback', 'Solve','position',[50,50,120,20]);
    BtClose=uicontrol('style','pushbutton','string','Закрыть',...
    'Callback', '_Close','position',[300,50,120,20]);
    //Функция решения уравнения
    function Solve()
    // Считываем значение переменных из текстовых полей и
    // преобразовываем их числовому типу
    a=eval(get(edit_a,'string'));
    b=eval(get(edit_b,'string'));
    c=eval(get(edit_c,'string'));
    d=b*b-4*a*c;
    // Проверяем значение флажка, если флажок выключен,
    if get(radio_bikv,'value')==0
    //то решаем квадратное уравнение
    if d<0
    set(textresult,'string','Нет решения квадратного уравнения');
    else
    x1=(-b+sqrt(d))/2/a;
    x2=(-b-sqrt(d))/2/a;
    set(textresult,'string',sprintf
    ("2 корня квадратного уравнения\t x1=%1.2f\tx2=%1.2f",x1,x2));
    end;
    //если флажок включен,
    else
    //то решаем биквадратное уравнение.
    if d<0
    set(textresult,'string','Нет       решения       биквадратного
уравнения');
    else
      y1=(-b+sqrt(d))/2/a;
      y2=(-b-sqrt(d))/2/a;
      if(y1<0)&(y2<0)
    set(textresult,'string','Нет       решения       биквадратного
уравнения');
     elseif (y1>=0)&(y2>=0)
     x1=sqrt(y1);x2=-x1;x3=sqrt(y2);x4=-x3;
    set(textresult,'string',sprintf("4 корня биквадратного...
    уравнения \t x1=%1.2f\tx2=%1.2f\tx3=%1.2f\tx4=%1.2f",...
    x1,x2,x3,x4));
     else
     if y1>=0
        x1=sqrt(y1);x2=-x1;
     else
        x1=sqrt(y2);x2=-x1;
     end;
     set(textresult,'string',     sprintf
    ("2корня      биквадратного      уравнения\t      x1=%1.2f\tx2=
%1.2f",x1,x2));
```

```
 end;
end;
end
endfunction
// Функция закрытия окна
function _Close()
close(f)
endfunction
```

Листинг 10.9. Решения квадратного или биквадратного уравнения

На рисунке 10.13 представлено окно программы.

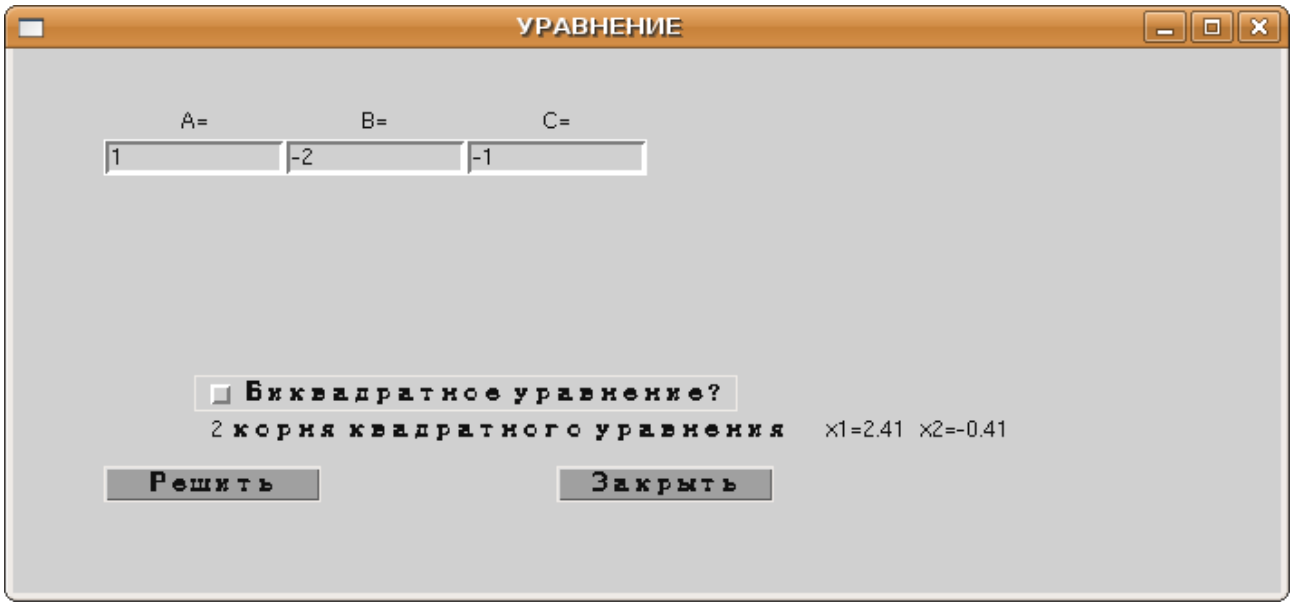

Рисунок 10.13. Окно программы решения уравнения

Авторы рекомендуют читателю разобраться с этой программой. Разобравшись с этой несложной программой, Вы сможете понять механизм взаимодействия стандартных компонентов Scilab и обработчиков событий, чтобы затем использовать эти знания при разработке собственных визуальных приложений.

10.2.5 Списки строк

Интерфейсный компонент «список строк» в простейшем случае можно рассматривать как окно с массивом строк в нем. Если длина списка превышает высоту окна, то для перемещения по списку может использоваться вертикальная полоса прокрутки, которая генерируется автоматически.

Создание списка строк производится с помощью функции *uicontrol*, при задании параметра *'Style' listbox'*. Рассмотрим это на простом примере (см листинг 10.10 и рис. 10.14).

```
// создание графического окна
f=figure();
// создание  listbox
h=uicontrol(f,'style','listbox','position', [10 10 150 160]);
// заполнение списка
set(h, 'string', "строка 1|строка 2|строка 3");
set(h, 'value', [1 3]);
// выделение item 1 и 3 в списке
```

Листинг 10.10. Создание списка

Рисунок 10.14. Список с выделенными элементами

Список позволяет пользователю выбрать одну или несколько строк и в зависимости от выбора произвести то или иное действие.

Выбор строки осуществляется щелчком левой кнопки мыши в тот момент, когда острие курсора указывает на выбираемую строку. Одновременно с подсветкой строки ее номер заносится в свойство *'value'* и генерируется событие *'Callback'*. Строки в списке нумеруются от 1. Для выбора разрозненных строк нужно зажать клавишу **Ctrl** и щелкать мышью по выделяемым строкам. При этом каждая выделяемая строка подсвечивается, а ее номер запоминается в векторе *'value'*. Для выбора группы подряд идущих строк можно нажать и удерживать клавишу **Shift**, а затем щелкнуть по первой и последней строке группы. Все промежуточные строки тоже будут выделены и все их номера запомнятся в векторе *'value'*.

С помощью рассмотренных в этой главе компонентов и встроенных функций Scilab можно создавать визуальные программы для решения инженерных и математические любой сложности, пример подобной программы будет рассмотрен в следующей главе.