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 монитора 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,'Свйоство','Значение')` устанавливается значение параметров, здесь `f` - указатель на графическое окно, `'Свойство'` – имя параметра, `'Значение'` – его значение.

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

2 `Значениеi` – будет в кавычках, если значением будет строка и без кавычек, если число.

Рис. 10.1. Графическое окно

Например, следующие две строки (см. листинг 10.1) задают месторасположение и размер окна (см. рис. 10.2).

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

Листинг 10.1

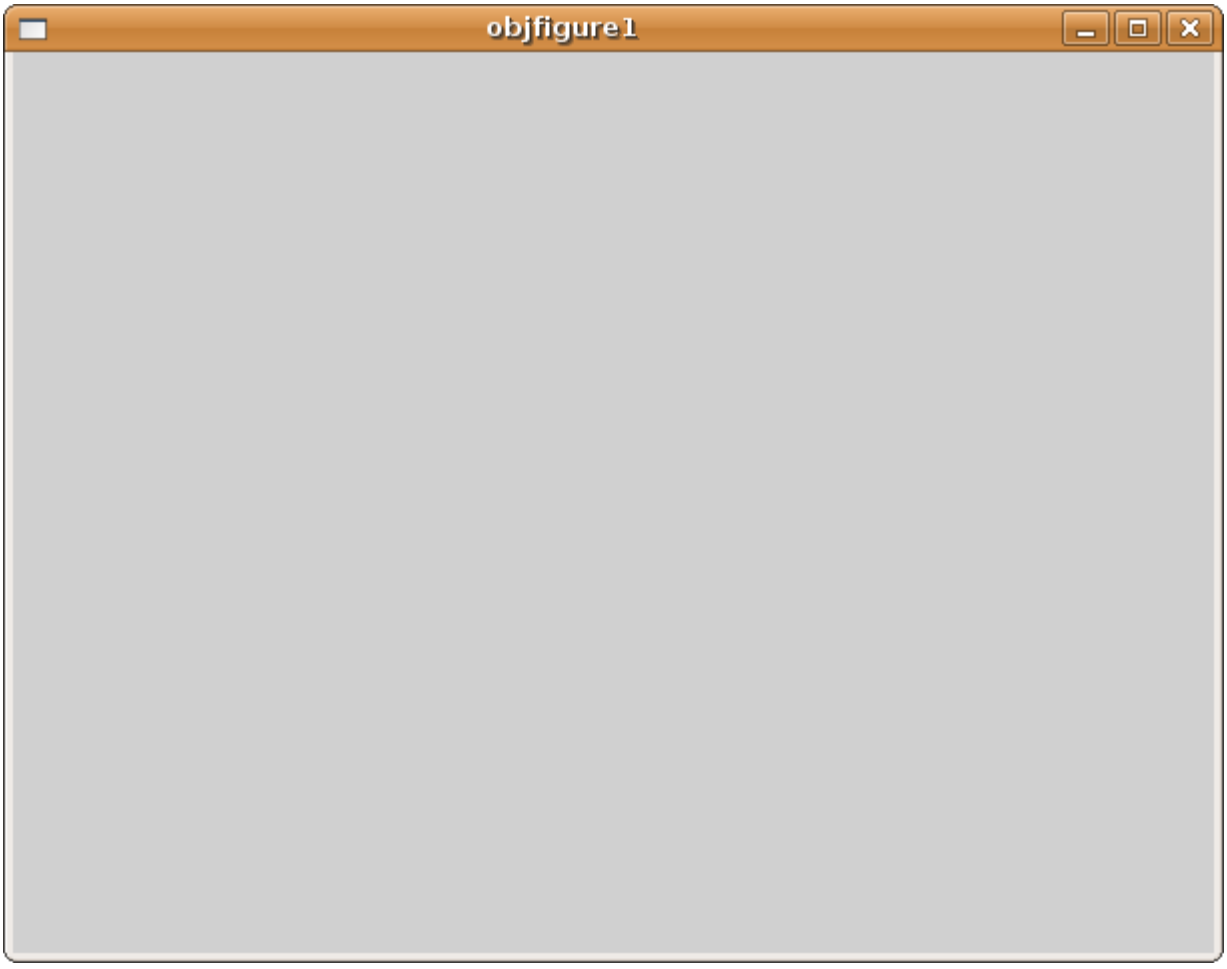

Рис. 10.2. Окно, созданное в результате выполнения листинга 10.1

Для изменения заголовка окна используется параметр `'figure_name', 'name'` определяющий заголовок окна (`'name'`). На листинге 10.2 приведен пример создания окна с именем **FIRST WINDOWS** (см. рис. 10.3). С помощью программы

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

Листинг 10.2. Создание окна с именем FIRST WINDOW

Рис.10. 3. Графическое окно с именем FIRST WINDOWS

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 функция `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 графические объекты (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; 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`, `Edittext`, `StaticText`, `Slider`, `Panel`, `Button Group`, `Listbox` или др компоненты;
- `'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)
```

- `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 10.3 и рис 10.4).

```
d=figure();
dbt=uicontrol(d,'Style','pushbutton');
set(dbt,'String','YES');
```

Листинг 10.3. Создание окна с кнопкой

Рис. 10.4. Окно с кнопкой

Модифицируем процедуру создания кнопки, задав дополнительно значения некоторых свойств: **Заголовок окна**, **надпись на кнопке** и ее **месторасположение**. Определяем позицию графического окна(левый верхний угол находится в точке с координатами (0,0)), ширина окна 250, а длина 100 пикселей. Аналогично определяются позиция и размеры кнопки с помощью параметра `position`, параметр `'string'` указывает имя динамического компонента (`'Button'`). Текст программы приведен на рис. 10.4, а на рис. 10.5 можно увидеть окно, которое получилось в результате работы этой программы.

```
f=figure();
set(f,'position',[0,0,250,100])
set(f,'figure_name','Button_example');
Button=uicontrol('style','pushbutton','string','Button',
'position',[50,50,100,20]);
```

Листинг 10.4. Определение свойств кнопки

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.

Рис. 10.5 Кнопка с заданными свойствами

Щелчок генерирует событие `callback`, которое указывается как параметр функции `uicontrol`, после чего в одинарных кавычках указывается имя вызываемой по щелчку функции (обработчика события `CallBack`).

```
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 10.5). После запуска этой программы появится окно, представленное на рис. 10.6, при щелчке по кнопке `Button` вызывается обработчик события функция `gr_sin`, в результате чего появится окно, изображенное на рис. 10.7.

```
f=figure();
set(f,'position',[0,0,250,100])
set(f,'figure_name','Grafik');
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 10.5. Пример кнопки с обработчиком события CallBack

Рис. 10.6. Окно с кнопкой

10.2.2. Метка

Следующим наиболее часто используемым компонентом является метка – текстовое поле для отображения текстовой информации. Для определения метки значения параметра `'Style'` в функции `uicontrol` должно иметь значение `'text'`. Компонент предназначен для вывода символьной строки (или нескольких строк). Выводимый текст - значение параметра `'string'` может быть изменен только из программы. Рассмотрим пример создания текстового поля (метки) с помощью функции `uicontrol` (см. листинг 10.6 и рис. 10.7):

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

Листинг 10.6. Создание метки

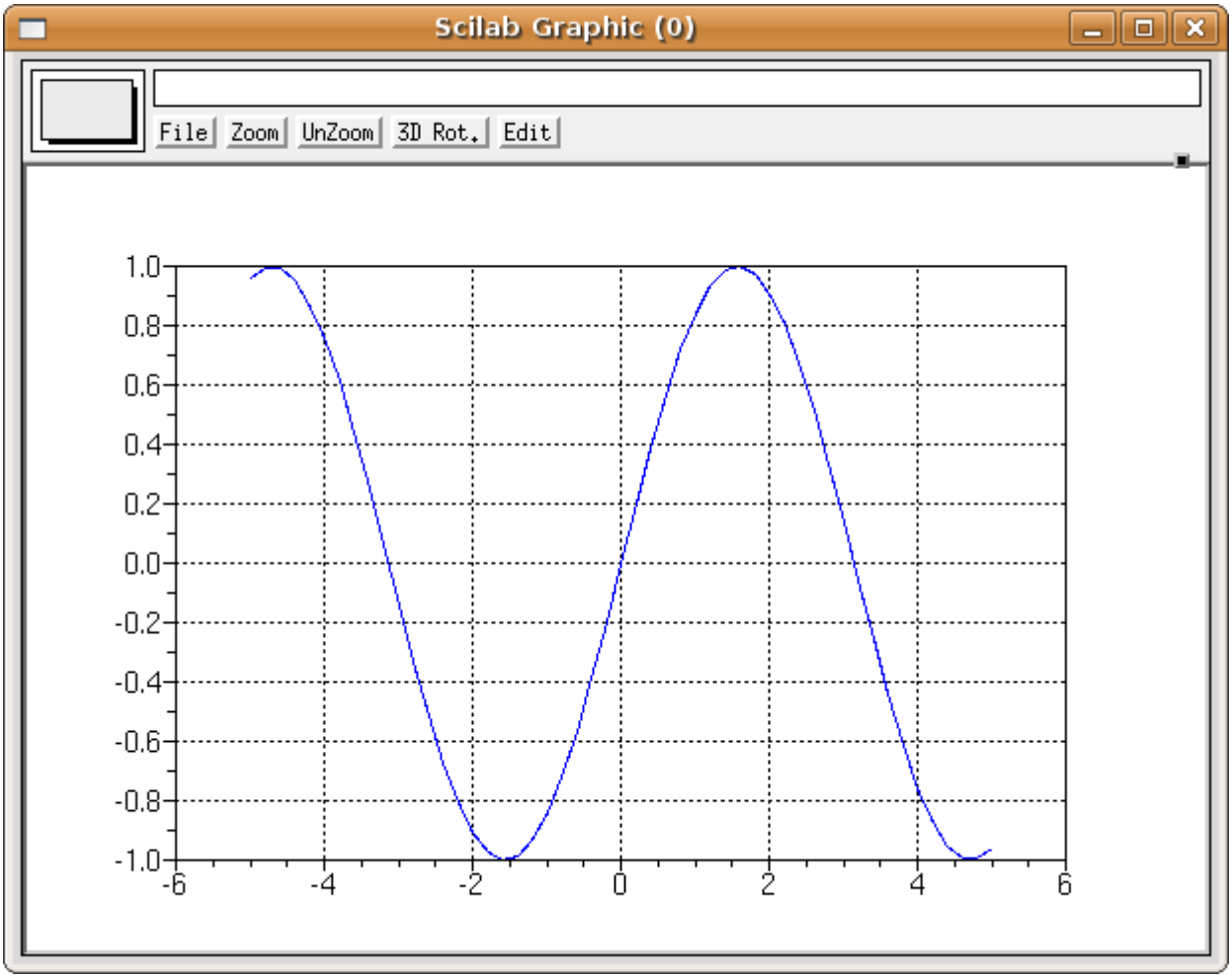

Рис. 10.7. График функции

Рис. 10.8. Окно с меткой

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

- `left` – выравнивание 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 10.7

```
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],'HorizontalA
lignment','right','BackgroundColor',[1 1 1],'String','Metka 3');
hSt4=uicontrol('Style','text','Position',[30,120,150,20],'Background
Color',[1 1 1],'String','Metka 4');
```

Листинг 10.7. Создание нескольких меток

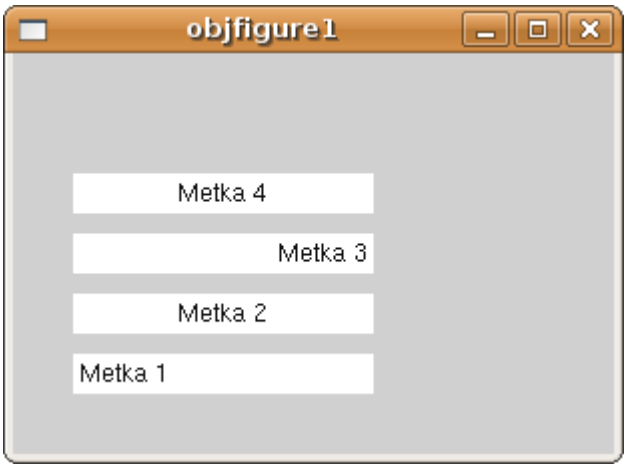

Рис. 10.9. Окно с несколькими метками

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'` – `'radiobutton'`.

Индикатором альтернативных комбинаций является переключатель (Radiobutton), который также создается с помощью функции uicontrol (см. листинг 10.8 и рис. 10.10):

```
hFig=figure();
Rb=uicontrol('Style','radiobutton','String','name','value’,      0,
'Position', [25,150,70,30]);
```

Листинг 10.8. Создание переключателя

При создании кнопки должно быть задано состояние кнопки (параметр `'value'`), это может быть 1 (кнопка активна) или 0 (кнопка не активна). Задать значение свойства `'value'` можно nfr;t и с помощью функции `set`.

```
set(Rb,’value’,0)
```

Переключатель, может реагировать на событие 'callback' и вызывать на выполнение определенную функцию.

Проиллюстрируем применение переключателей на примере выбора функции, график которой должен воспроизводиться в отдельном графическом окне (см. листинг 10.8 и рис. 10.11).

```
hFig=figure('Position',[50,50,200,200]);
//Создание радиокнопок
hRb1=uicontrol('Style','radiobutton','String','sin(x)','value',0,
'Position',[25,100,60,20],'callback','Radio');
```

```
  hRb2=uicontrol('Style','radiobutton','String','cos(x)','value',0,
'Position',[25,140,60,20],'callback','Radio');
```

Рис. 10. 10.Окно с переключателем

```
//Функция Radio, реагирующая на событие 'callback'
function Radio()
newaxes;
x=-2*%pi:0.1:2*%pi;
if get(hRb1,'value')==1 //Если активна первая кнопка
set(hRb2,'value',0);   //Сброс альтернативной кнопки
y=sin(x);
plot(x,y,'-r');        //Построение синусоиды
xgrid();
else
if get(hRb2,'value')==1
set(hRb1,'value',0);
y=cos(x);
plot(x,y,'-r');  //Построение косинусоиды
xgrid();               //Нанесение сетки на график
end;
end;
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.

10.2.4. Компонент окно редактирования

Интерфейсный элемент окно редактирования (у того компонента свойство `'Style'` должно принимать значение `'text'`) 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`.

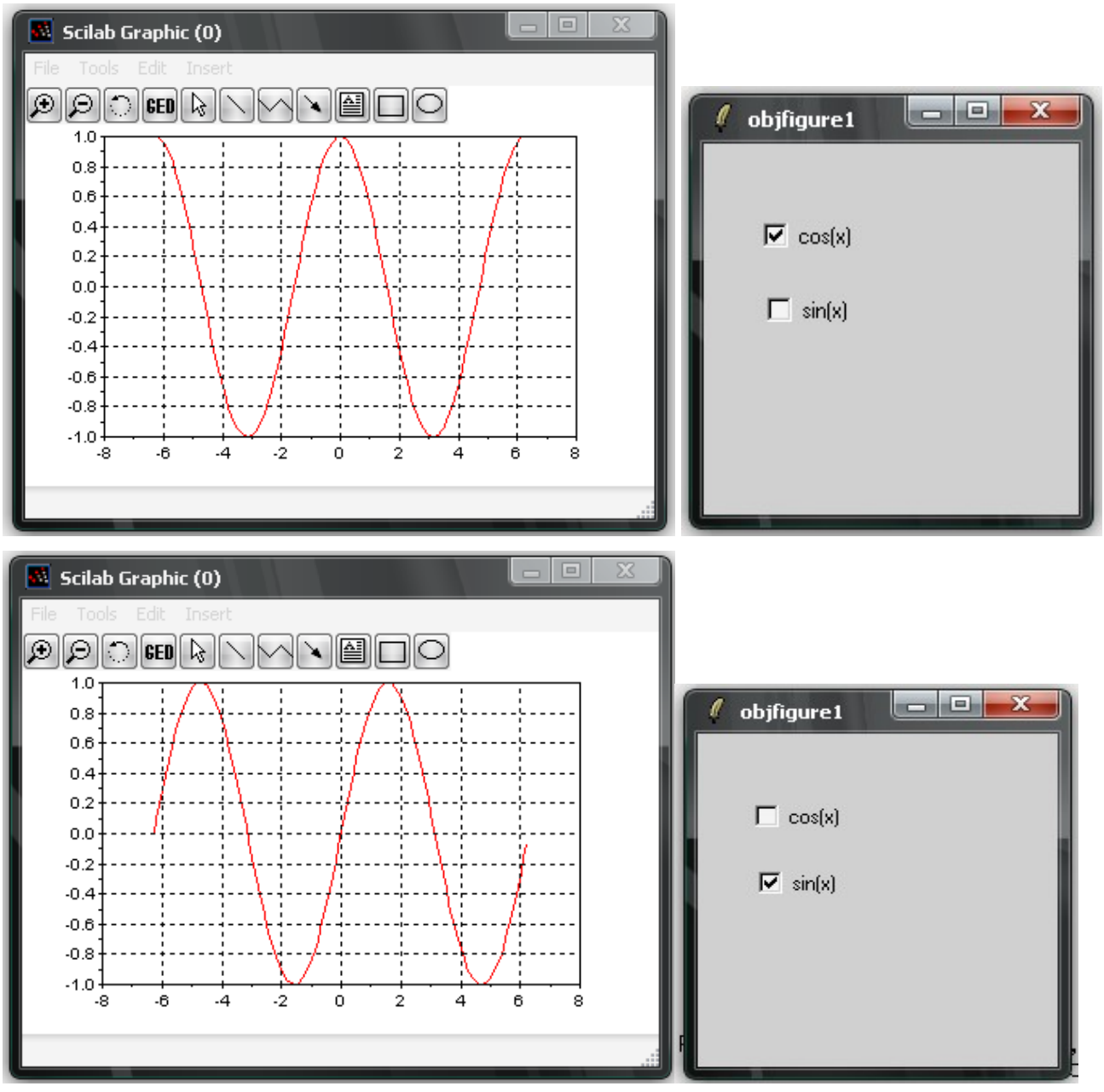

Рис. 10. 11. Окно с переключателем

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. 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 evstr (будет рассмотрено далее на примере квадратного уравнения).

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

Написать программу решения квадратного или биквадратного уравнения. Выбор типа уравнения будем проводить с помощью компонента **Переключатель**. Программа представлена на листинге 10.9.

```
f=figure(); //Создание графического объекта
set(f,'position',[0,0,500,300])
set(f,'figure_name','EQUATION');
//Создание меток дя полей ввода коэффициентов
lab_a=uicontrol(f,'style','text','string','A=','position', [50, 250, 100, 20]);
lab_b=uicontrol(f,'style','text','string','B=','position',[150, 250, 100, 20]);
lab_c=uicontrol(f,'style','text','string','C=','position',[250, 250, 100, 20]);
// полей ввода коэффициентов
edit_a=uicontrol(f,'style','edit','string','1','position',[50,  230, 100, 20]);
edit_b=uicontrol(f,'style','edit','string','2','position',[150, 230, 100, 20]);
edit_c=uicontrol(f,'style','edit','string','1','position',[250, 230, 100, 20]);
//Текстовое поле, определяющее вывод результатов
```

```
  text_result=uicontrol(f,'style','text','string','','position',[50,
80, 450, 20]);
  //Флажок, отвечающая за выбор типа уравнения
  radio_bikv=uicontrol('style','radiobutton','string','Bikvadrat?',
'value',1,'position',[100,100,200,20]);
  BtSolve=uicontrol('style','pushbutton','string','Solve','Callback',
'Solve','position',[50,50,60,20]);
  BtClose=uicontrol('style','pushbutton','string','Close','Callback',
'_Close','position',[400,50,60,20]);
  //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
  if get(radio_bikv,'value')==0
  if d<0
  set(text_result,'string','No Solve kvadrat');
  else
  x1=(-b+sqrt(d))/2/a;
  x2=(-b-sqrt(d))/2/a;
  set(text_result,'string',sprintf("2         kornya         kvadrat\t
x1=%1.2f\tx2=%1.2f",x1,x2));
  end;
  else
  if d<0
    set(text_result,'string','No Solve bikvadrat ');
  else
    y1=(-b+sqrt(d))/2/a;
    y2=(-b-sqrt(d))/2/a;
    if(y1<0)&(y2<0)
```

```
        set(text_result,'string','No Solve bikbadrat');
    elseif (y1>=0)&(y2>=0)
            x1=sqrt(y1);x2=-x1;x3=sqrt(y2);x4=-x3;
              set(text_result,'string',sprintf("4 kornya bikvadrat \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(text_result,'string',
         sprintf(" 2 kornya bikvadrat \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 10.12 представлено окно программы.

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. 10.13).

```
f=figure();
// создание графического окна
```

```
h=uicontrol(f,'style','listbox','position', [10 10 150 160]);
// создание  listbox
set(h, 'string', "item 1|item 2|item3");
// заполнение списка
set(h, 'value', [1 3]);
// выделение item 1 и 3 в списке
```

Листинг 10.10. Создание списка

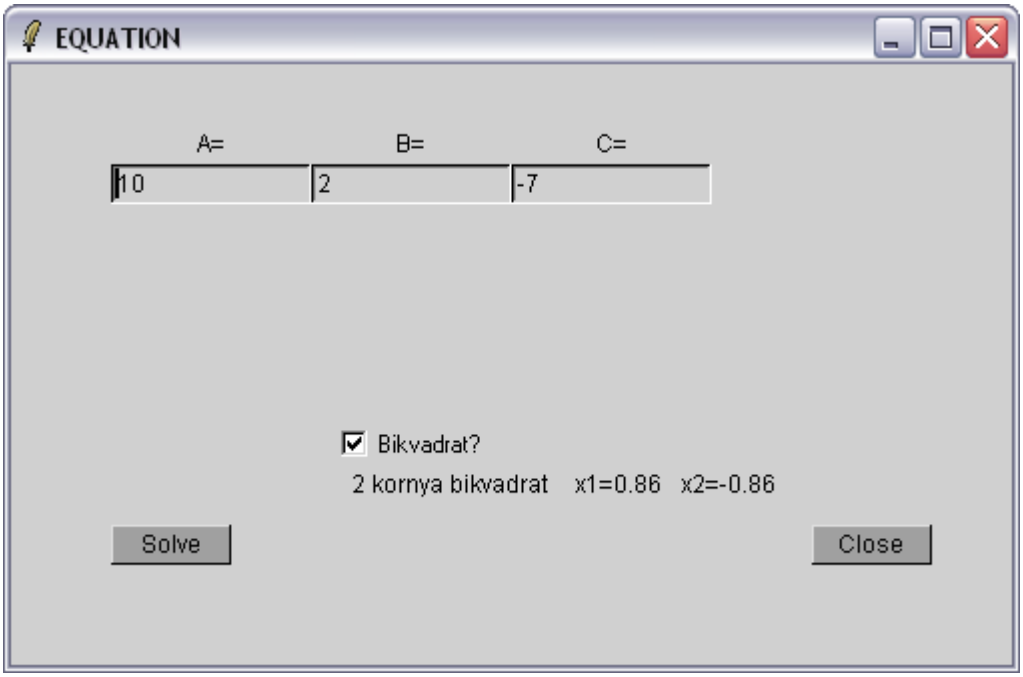

Рис. 10.12. Окно программы решения уравнения

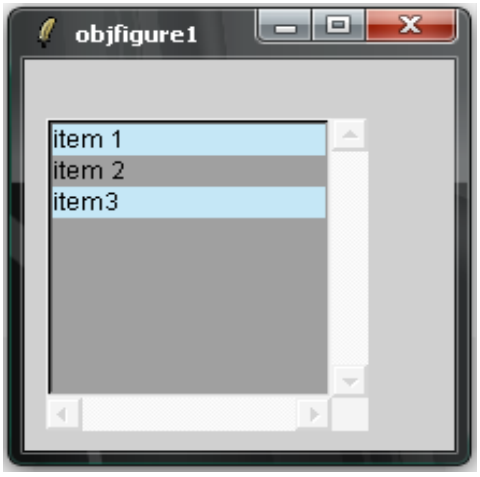

Рис. 10.13. Окно со списком

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`. В следующем параграфе будет приведено большое количество примеров использования этих (и других) компонентов при программировании различных задач.