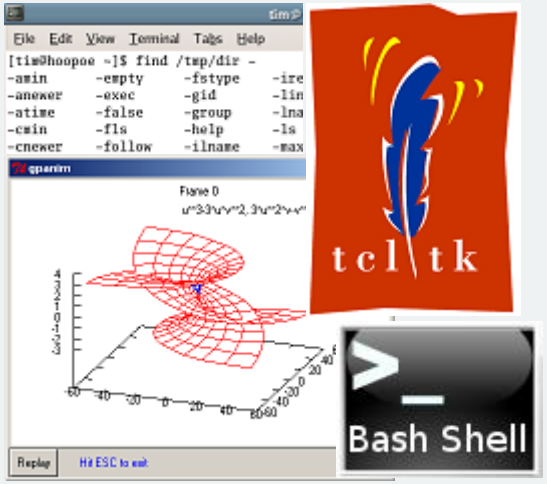
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Интерпретируемые языки программирования

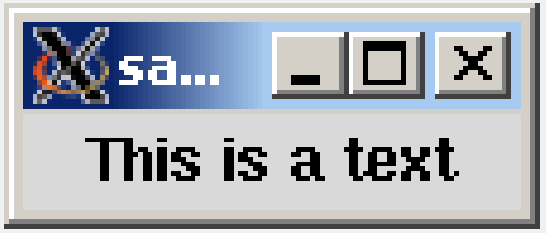


Лабораторная работа №4

Интерпретируемый язык Tcl.

Библиотека Tk.

Программирование с библиотекой Tk



Указать интерпретатор

Создать компоненты

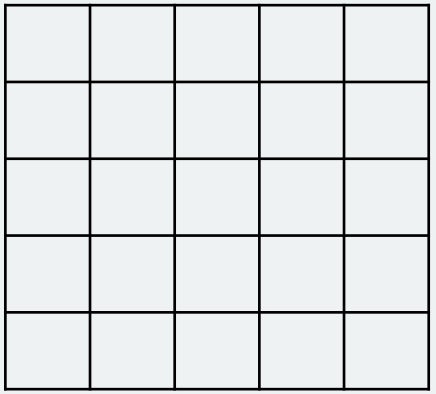
Расположить компоненты

```
#!/usr/bin/wish

label .l1 -text "This is a text"

grid .l1 -row 0 -column 0
```

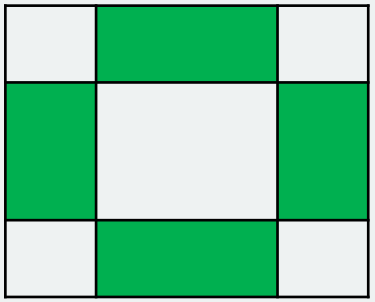
+ Задать обработчики

Менеджеры компоновки

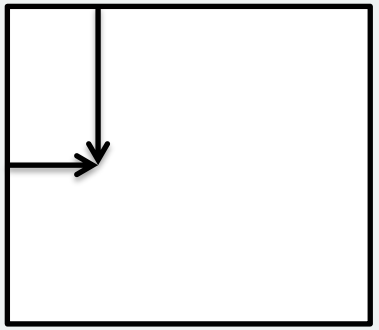
В Tcl/Tk доступны 4 вида менеджеров компоновки:



- grid – разделяет рабочую область главного окна на виртуальные ячейки, располагает все компоненты внутри ячеек;



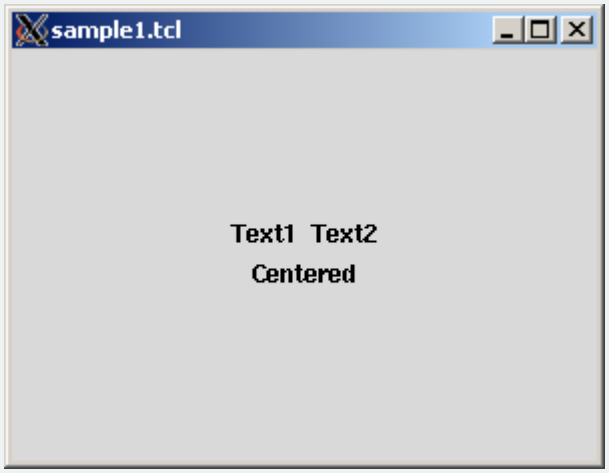
- pack – позволяет расположить компоненты внутри окна относительно границ окна;
- place – позволяет задать абсолютное позиционирование компонентов на окне.

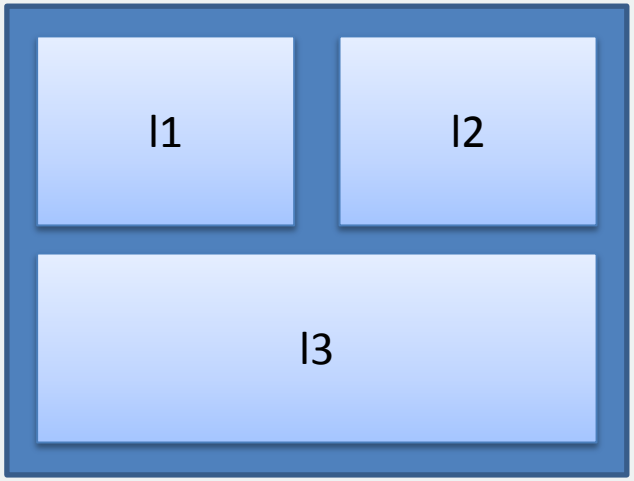


- frame – контейнер для других виджетов

Менеджер компоновки grid

```
#!/usr/bin/wish

label .l1 -text "Text1"
label .l2 -text "Text2"
label .l3 -text "Centered"

grid .l1 -row 0 -column 0
grid .l2 -row 0 -column 1
grid .l3 -row 1 -column 0 -columnspan 2
```





Менеджер компоновки grid

Основные параметры расположения при использовании grid

-row - указание ряда, в котором располагается компонент

-column - указание столбца, в котором располагается компонент

-rowspan - указание числа строк для объединения

-columnspan - указание числа столбцов для объединения

-padx - указание отступа в пикселях от края или краёв по X

-pady - указание отступа в пикселях от края или краёв по Y

Менеджер компоновки pack

```
#!/usr/bin/wish

# Make the widgets
label .t -text "Top"
label .b -text "Bottom"
label .l -text "Left\nSide"
label .r -text "Right\nSide"

# Lay them out
pack .t   -side top
pack .b   -side bottom
pack .l   -side left
pack .r   -side right
```

Менеджер компоновки pack

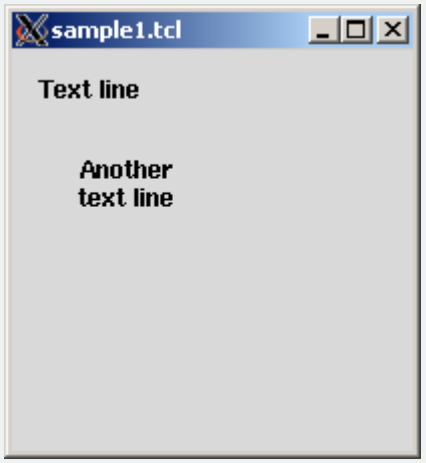
```
#!/usr/bin/wish

# Make the widgets
label .t -text "Top"          -background blue -foreground white
label .b -text "Bottom"       -background yellow
label .l -text "Left\nSide"   -background red
label .r -text "Right\nSide" -background green

# Lay them out
pack .t   -side top    -fill x
pack .b   -side bottom -fill x
pack .l   -side left   -fill y
pack .r   -side right  -fill y
```

Менеджер компоновки pack

```
#!/usr/bin/wish

# Make the widgets
label .t -text "Top"          -background blue
label .b -text "Bottom"       -background yellow
label .l -text "Left\nSide"   -background red
label .r -text "Right\nSide" -background green

# Lay them out
pack .t   -side top    -anchor w
pack .b   -side bottom -anchor e
pack .l   -side left   -anchor s
pack .r   -side right  -anchor n
```

Менеджер компоновки place

```
#!/usr/bin/wish

label .l1 -text "Text line"
label .l2 -text "Another\ntext line"


place .l1 -x 10 -y 10
place .l2 -x 30 -y 50
```



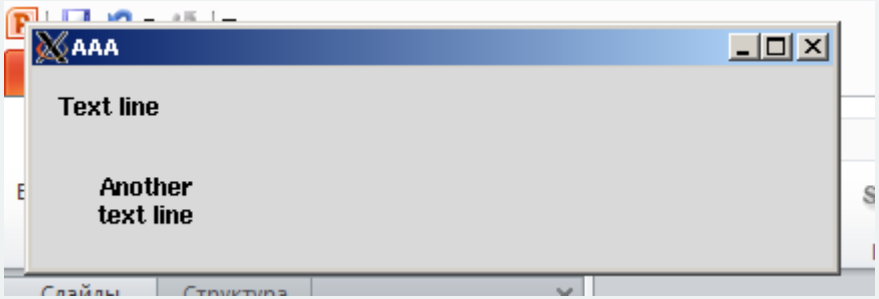
Управление главным окном – команда wm

Основные аргументы:

title – задание заголовка окна;

resizable – возможность менять размеры;

geometry – задание положения окна.

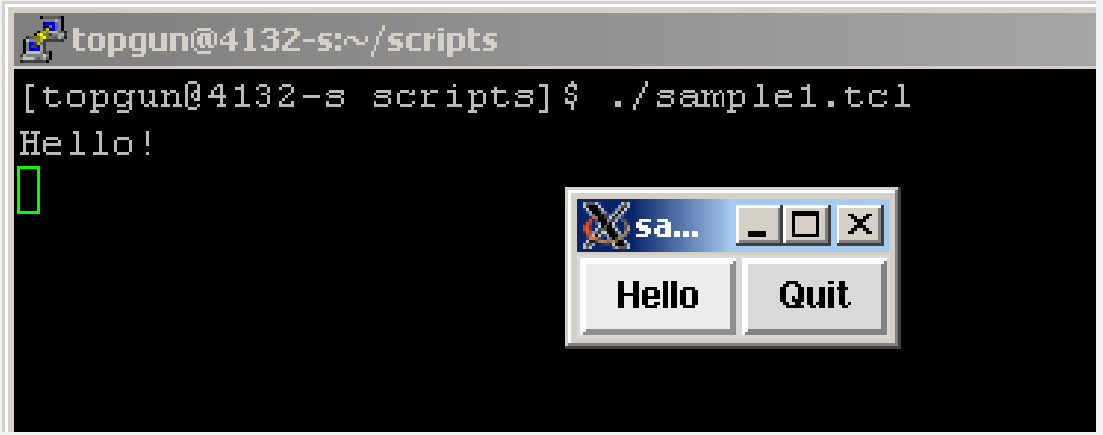



```
#!/usr/bin/wish

wm title . "AAA"
wm resizable . 0 0
wm geometry . =400x100+10+10

label .l1 -text "Text line"
label .l2 -text "Another\ntext line"

place .l1 -x 10 -y 10
place .l2 -x 30 -y 50
```

Метки

```
#!/usr/bin/wish

label .l1 -text "First line"

label .l2 -text "Second line" -foreground Yellow \
                              -background Blue

label .l3 -text "Third line"  -font { -size 24 }

label .l4 -text "Forth line"  -relief raised

grid .l1 -row 0
grid .l2 -row 1
grid .l3 -row 2
grid .l4 -row 3
```

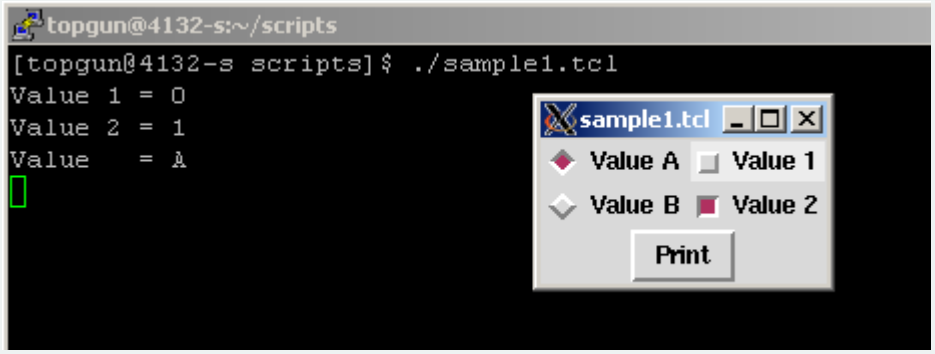
Кнопки: задание обработчика нажатия в виде процедуры



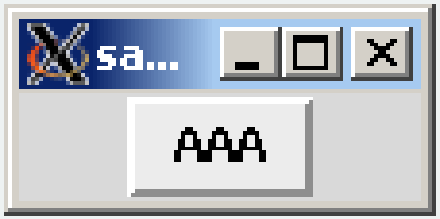

```
#!/usr/bin/wish

proc b1_proc {} {
  puts stdout "Hello!"
}

button .b1 -text "Hello" -command b1_proc
button .b2 -text "Quit"  -command { destroy . }

grid .b1 -row 0 -column 0
grid .b2 -row 0 -column 1
```

Переключатели

```
#!/usr/bin/wish

proc print_vars {} {
  global cValue1 cValue2 rValue

  puts stdout "Value 1 = $cValue1"
  puts stdout "Value 2 = $cValue2"
  puts stdout "Value   = $rValue"
}

checkbutton .c1 -text "Value 1" -variable cValue1
checkbutton .c2 -text "Value 2" -variable cValue2

radiobutton .r1 -text "Value A" -variable rValue -value "A"
radiobutton .r2 -text "Value B" -variable rValue -value "B"

button .b -text "Print" -command print_vars


grid .c1 -row 0 -column 1
grid .c2 -row 1 -column 1
grid .r1 -row 0 -column 0
grid .r2 -row 1 -column 0
grid .b -row 2 -column 0 -columnspan 2
```



Изменение свойств объектов

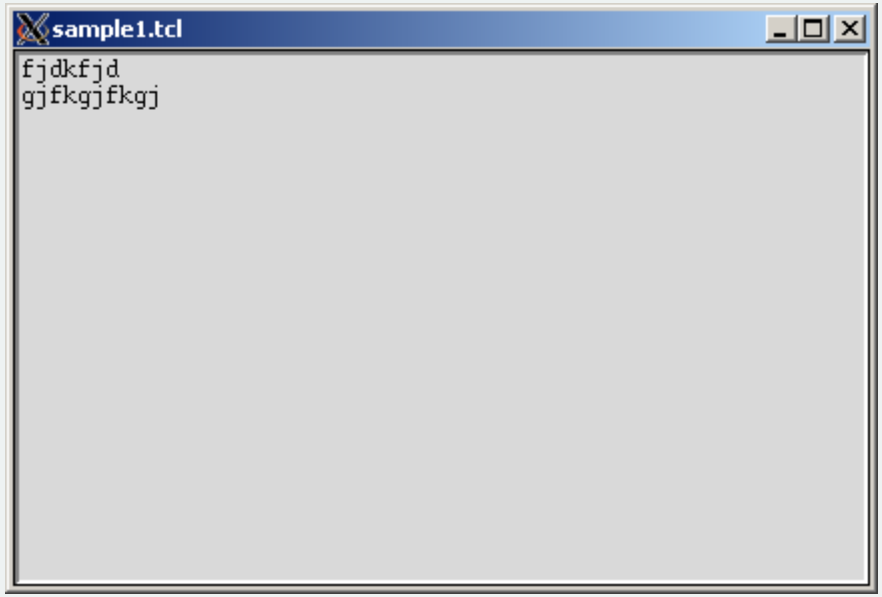



```
#!/usr/bin/wish

proc change_text { } {
  global .b1
  .b1 configure -text "AAA"
}

button .b1 -text "Button" -command change_text

grid .b1 -row 0 -column 0
```

Поля ввода


```
#!/usr/bin/wish

label .l -text "Enter:"

entry .e -width 40 -textvariable var1

button .b -text "Clear" -command {set var1 ""}

focus .e

grid .l -row 0 -column 0 -sticky e
grid .e -row 0 -column 1 -sticky w
grid .b -row 1 -column 0 -columnspan 2
```

Текстовые поля

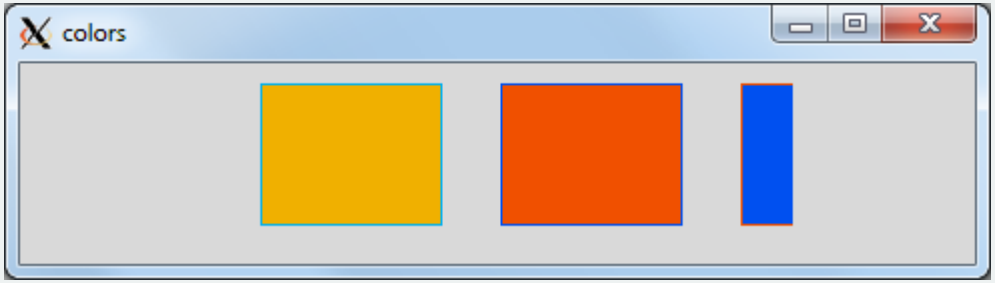



```
#!/usr/bin/wish

text .t -width 60 -height 20 -wrap word

pack .t -fill both
```

Текстовые поля

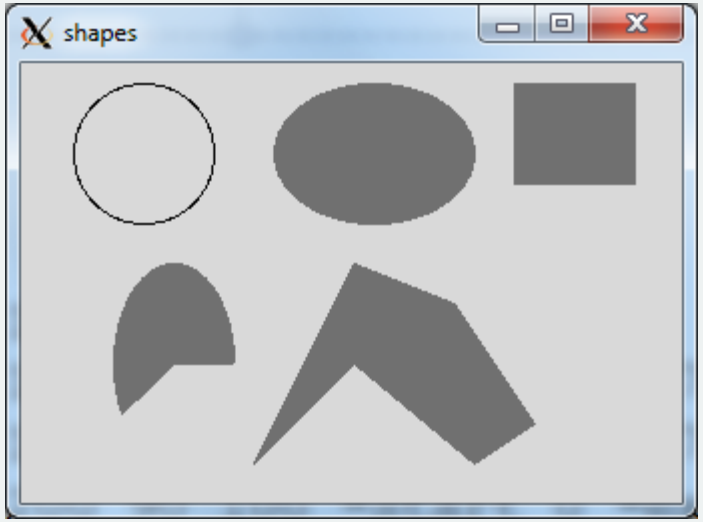
```
#!/usr/bin/wish

text .t -width 60 -height 20 -wrap word

pack .t -fill both

.t insert 0.0 "Hello"

.t insert end "Hello"

.t delete 0.0 end
```

Контекст графического устройства




```
#!/usr/bin/wish
canvas .can
.can create rect  30 10 120 80 -outline #0bf -fill #fb0
.can create rect 150 10 240 80 -outline #05f -fill #f50
.can create rect 270 10 370 80 -outline #f50 -fill #05f

pack .can

wm title . "colors"
wm geometry . 400x100+300+300
```

Контекст графического устройства




```
#!/usr/bin/wish

canvas .can

.can create oval  10 10  80  80
.can create oval 110 10 210  80 \
 -outline #777 -fill #777
.can create rect 230 10 290  60 \
 -outline #777 -fill #777
.can create arc   30 200 90 100 -start 0 -extent 210 \
                                -outline #777 -fill #777

set points [ list 150 100 200 120 240 180 210  200 \
             150 150 100 200 ]
.can create polygon $points -outline #777 -fill #777

pack .can
wm title . "shapes"
wm geometry . 330x220+300+300
```

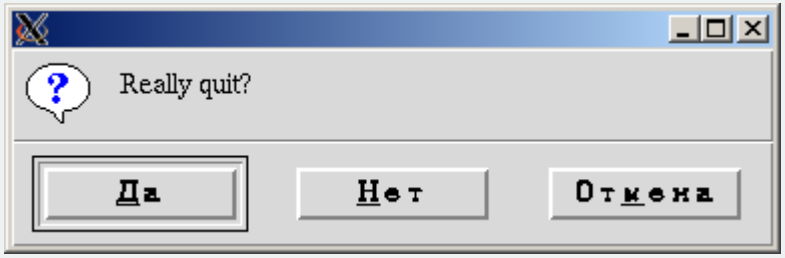
Информационные диалоги

```
tk_messageBox -message "Really quit?" \
              -type yesnocancel \
              -icon question
```

-message - текст сообщения
-parent - окно, из которого вызван диалог
-title - заголовок
-type - набор кнопок
- ok
- okcancel
- yesno
- yesnocancel
- retrycancel
- abortretryignore

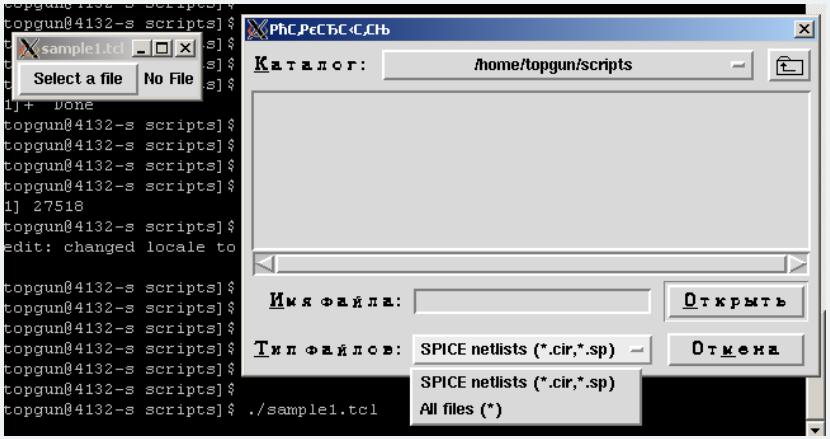
-icon - иконка сообщения
- error
- info
- question
- warning

Информационные диалоги




```
set answer [tk_messageBox -message "Really quit?" \
                          -type yesnocancel -icon question]

switch -- $answer {
    yes exit
    no { tk_messageBox -message "Quit canceled!" \
                       -type ok -icon info}
}
```

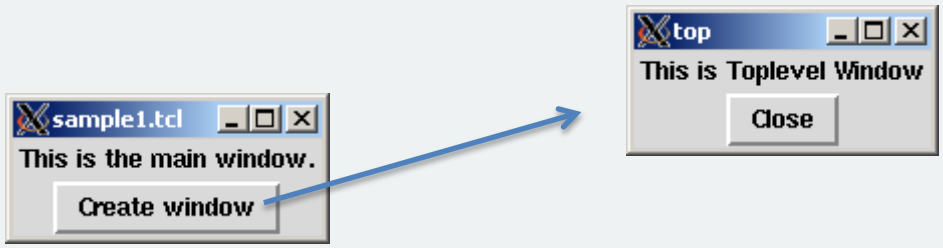
Диалог открытия файлов




```
set types {
  {"SPICE netlists" {.cir .sp}   }
  {"All files"              *    }
}

proc doIt {label} {
  global types
  set file [tk_getOpenFile -filetypes $types -parent .]
  $label configure -text $file
}

label .l -text "No File"
button .b -text "Select a file" \
        -command "doIt .l"

grid .b -row 0 -column 0
grid .l -row 0 -column 1
```

Создание дополнительных окон

```
proc makeTop { } {
  toplevel .top ;#Make the window
  label .top.lab -text "This is Toplevel Window”

  button .top.but -text "Close" -command { destroy .top }
  grid .top.lab –row 0
  grid .top.but –row 1
}

label .lab -text "This is the main window."
button .but -text "Create window" -command { makeTop }
grid .lab –row 0
grid .but –row 1
```



Работа с изображениями

```
#!/usr/bin/wish

image create photo imgobj -file "picture.gif"
label .l1 –image imgobj

pack .l1
```

Создание меню

```
menu .menubar
. configure -menu .menubar

menu .menubar.file
.menubar.file add command -label Exit \
                          -command exit


.menubar add cascade -label File \
                     -menu .menubar.file
```

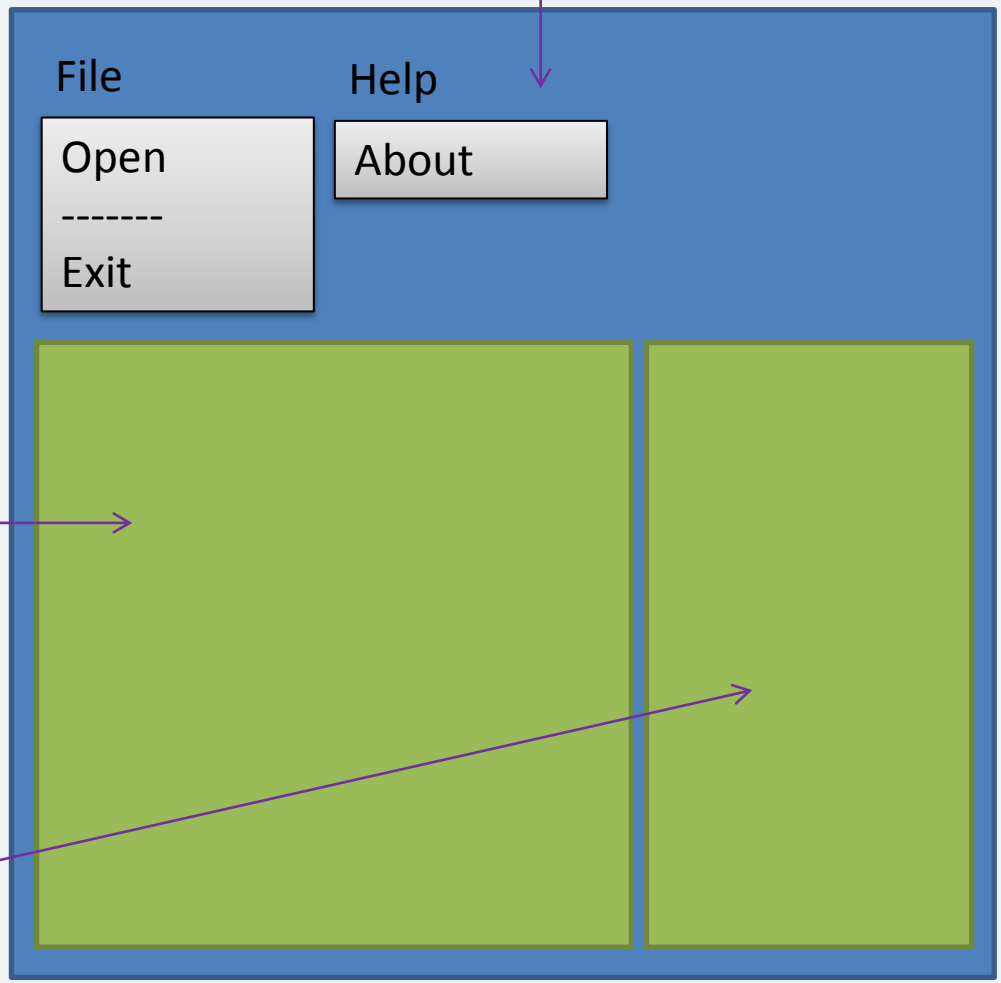
Задание

Диалог «о программе»
С картинкой и кнопкой «Close»

Меню

Вызывает диалог открытия файлов

Запрашивает выход из программы, выходим, если разрешили



Текстовое поле.
Сюда попадает код:

- чётные варианты – SPICE,
- нечётные – Verilog

Вырезаем однострочные комментарии (считаем, что начинаются только с начала строки)

Текстовое поле.
Сюда попадают:

- чётные варианты – имена subckt,
- нечётные – имена модулей Verilog