Графические Системы. Часть II

Практическое Занятие № 6

Программирование графического пользовательского интерфейса средствами X-WINDOW. ИПВУ. Tcl/TK

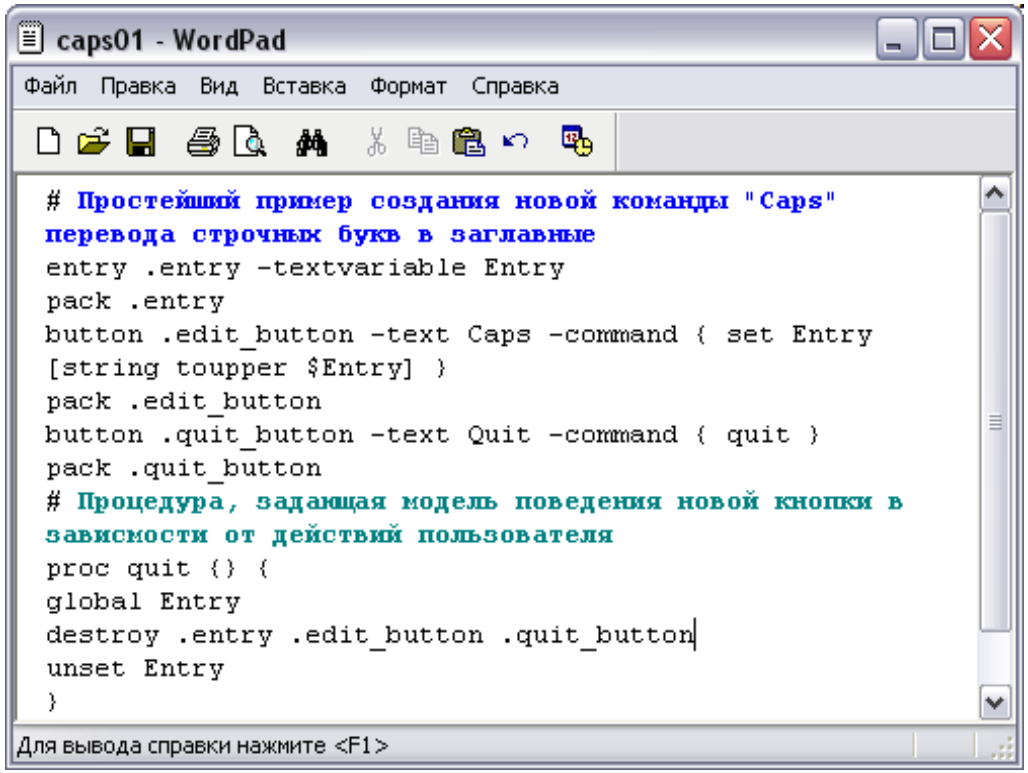
The Tk Way of Thinking – Что это значит – программировать с помощью Tcl/Tk?

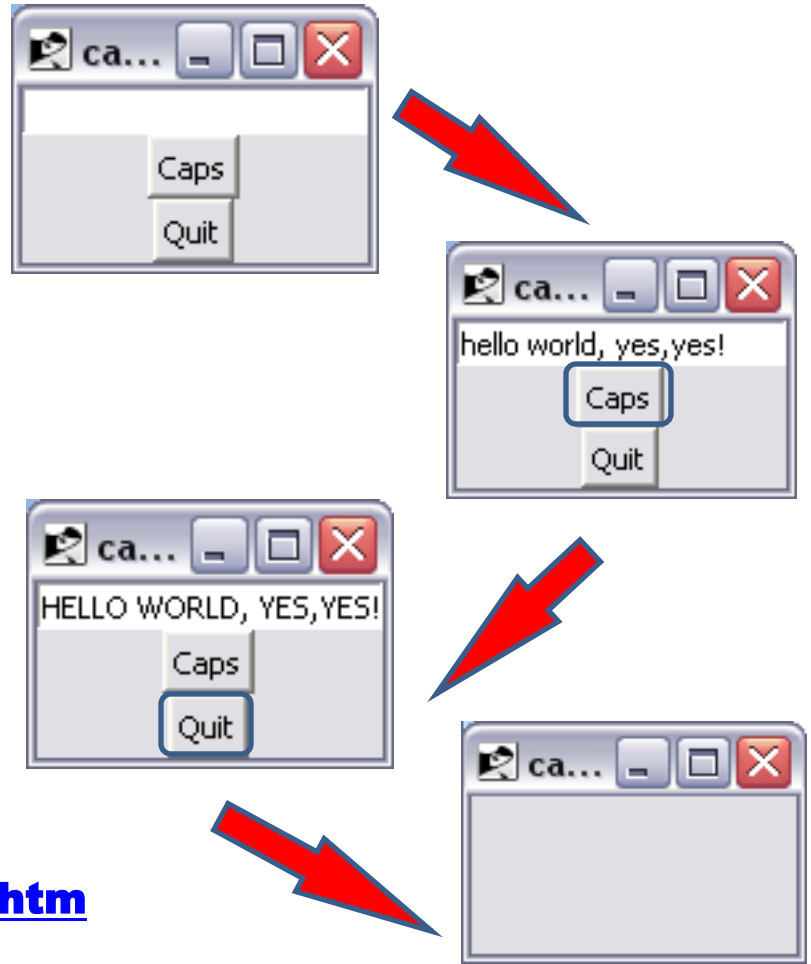
Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

Итак, на примере простейшего **GUI**, рассмотрим некоторые характерные особенности методики программирования графических интерфейсов с помощью **ИПВУ Tcl/Tk.**

Создадим **GUI**, в котором пользователь может вводить символьные последовательности и с помощью специальной **командной** кнопки преобразовывать введенные строчные символы в заглавные. Кроме того, предлагаемый **GUI** снабжен кнопкой, которая очищает его корневое окно ото всех сгенерированных **widget'ов**.





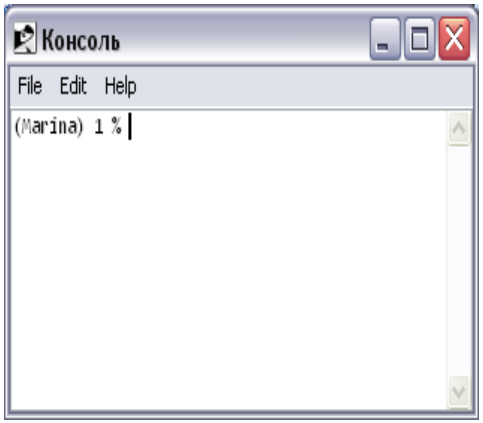
Встроенная Tcl - команда string рассмотрена здесь:
http://a0601.narod.ru/Zelenizina_2008/string.htm

Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

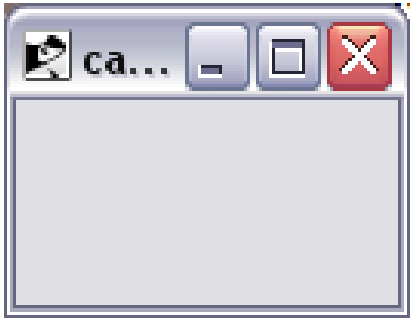
Что же можно делать со скриптом caps01 и реализованным GUI caps01?

- По разному стартовать (**running**) скрипт **caps01**, например, в интерактивном режиме в командном окне (Консоль) **wish**:



- Вводить при этом другие команды в командном окне (**Консоль**) **wish**.
- Вводить текст в **root**-окне **GUI caps01** после его старта:

- «Кликать» на любой кнопке в **root**-окне **GUI caps01**:



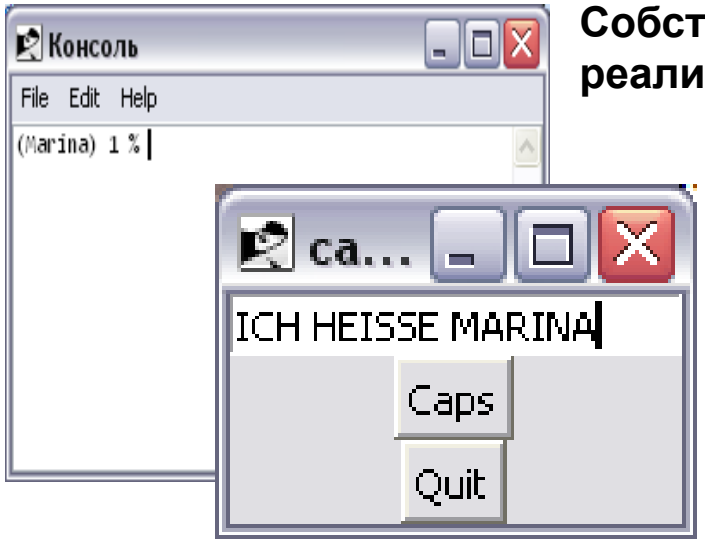
При этом действия пользователя этого **GUI** определяются и диктуются не теми, кто написал **wish** интерпретатор, и даже не теми, кто создал сам скрипт **caps01** (это – относительно), а самим пользователем.

Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

Если бы подобный **GUI** программировался традиционным способом (например, с помощью **ЯПВУ** – языков программирования высокого уровня и совмещенных с ними ГС), программисту бы пришлось вводить весьма запутанную совокупность операторов и процедур, реализующих привычную последовательность действия оператора и реакций на них самой программы.

Однако **GUI-приложения** пишутся не таким традиционным способом – т.н. метод **«top down»** и **«one step at a time»**. Вместо этого **«система времени исполнения GUI» (GUI runtime system)** генерирует **события (*events*)**, а программист просто ассоциирует с этими событиями те **действия (*actions*)**, которые возникают (могут возникнуть) в программируемом интерфейсе.



Собственно, работа по ассоциированию, это реализация (программирование) чего-то вроде:

- Для следующего ввода с клавиатуры (**keystroke**) в **командном окне**, сделать…..
- Для следующего ввода с клавиатуры (**keystroke**) в **root-окне GUI**, сделать…..
- Для **«mouse click»** на кнопке **«Caps»**, сделать…..
- Для **«mouse click»** на кнопке **«Quit»**, сделать…..

Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

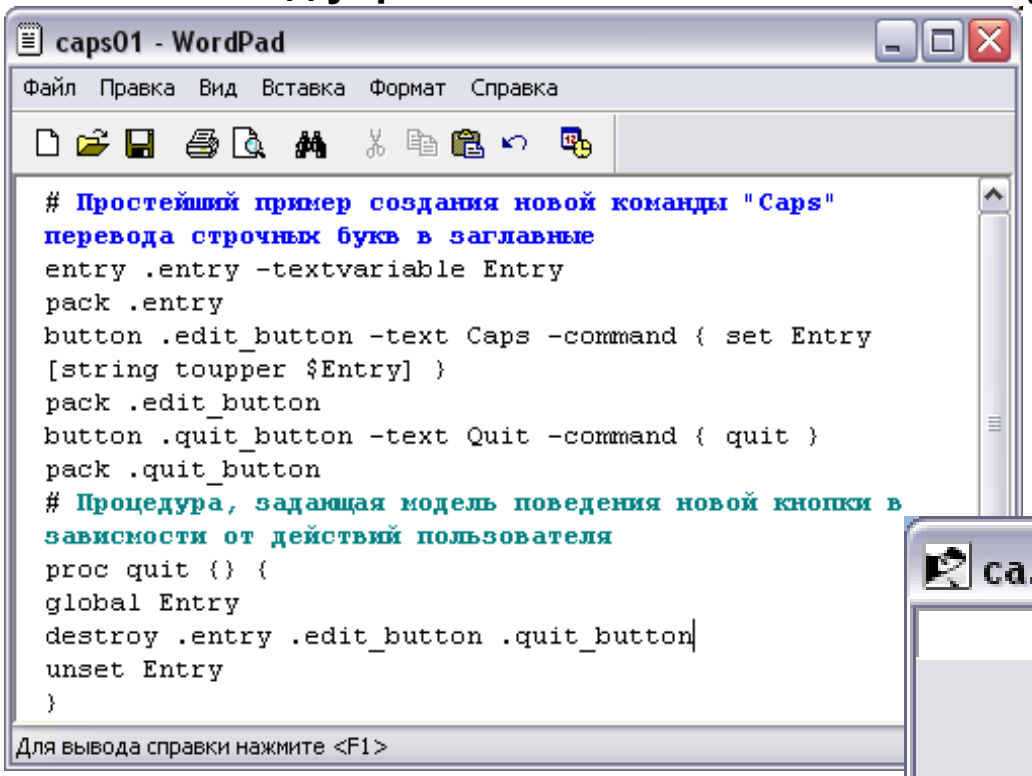
Window Manager'ы

С точки зрения программиста Tk - **GUI runtime system** состоит из двух частей:

1. Собственно **window manager**, из под которого запускается **wish-скрипт**; он обеспечивается операционной системой, либо «корневым» **GUI** (**X Window**, или любой, запущенный из под **X Window**) под управлением **ОС**.
2. **Tk - runtime system**

То, что происходит (отрабатывается) приложением – **Tk- runtime system** при реализации событий (**events**) типа **key press, key release, mouse movement**, или **mouse click**, осуществляется под управлением обоих этих частей (**runtime systems**).

Здесь важно обратить внимание на то, что программист **Tk - runtime system** работает на гораздо более высоком уровне абстракции, чем манипулирование перечисленными выше событиями (**events**) и программированием ассоциируемых с ними действий – **actions**:



Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

Что входит в функции Window Manager'а (назовем его «корневым»)?

Во-первых, это посылка сообщений Tk'ю о **key press, key release, mouse movement**, или **mouse click.** Будем называть такие сообщения ***user events***, чтобы отличить от **Input/Output (I/O) events**.

Вернемся к нашему **GUI caps01**.

Вспомним, что user не мог начать ввод текстовой строки в подокне **text entry** до тех пор, пока не осуществил первый клик в области этого подокна (**subwindow**).

Концепция, которая здесь реализуется получила в теории проектирования **GUI** название **фокуса**. В каждый момент времени только одно окно может «иметь» **фокус**. Это означает, что ввод (в нашем случае) возможен только в этом окне (**subwindow**).

Фокус определяется на двух уровнях. **Window Manager** определяет, какое окно должно иметь **фокус**, и передает его той программе, которая это окно стартовало.

В случае **GUI caps01** это окно – **wish**.

И уже **wish - интерпретатор** (программа) определяет, какое окно (**subwindow**) стартованной **Tk - runtime system** должно получить **фокус** для отработки действий пользователя.

По умолчанию в **Tk фокус** остается неизменным при движении мыши, а изменяется по клику. При программировании **GUI** можно изменять эти умолчания.

Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

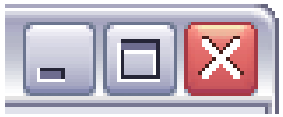
Вторая функция **Window Manager'a** - это контроль начала координат и размера всех дочерних окон.

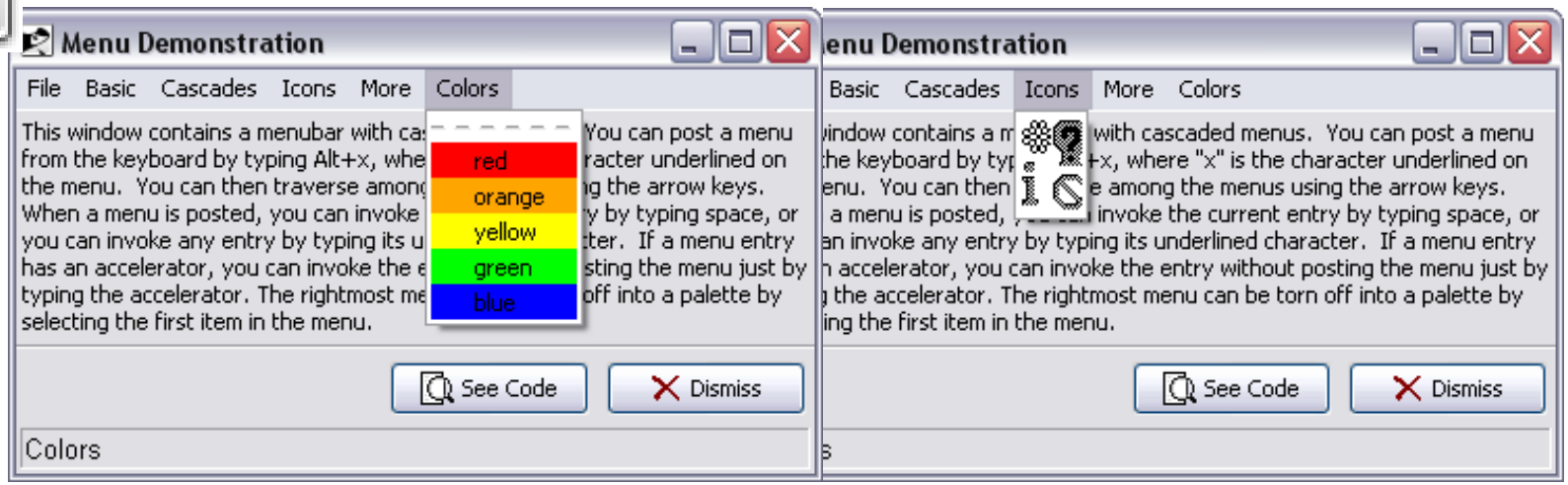
Tk заимствует у **X Window** термин (**term**) **geometry** для передачи этих параметров (позиция и размеры) генерируемому окну **Tk - runtime system.**

Используя мышь для перемещения окна и изменения его размеров, пользователь работает с **Window Manager.**

В результате **Window Manager** сообщает **Tk'ю**, что должно быть пересчитано, а **Tk**, в свою очередь, осуществляет этот пересчет наилучшим образом (вспомните менеджеры компоновки, в частности)

И наконец, **третьей** функцией **Window Manager'a** является определенное оформление корневого окна **Tk - runtime system**, включая его заголовок.



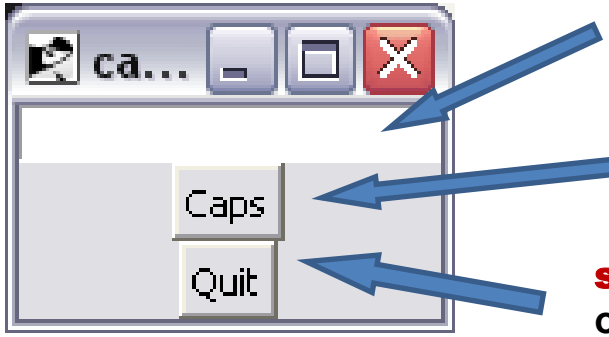


Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

Widget'ы.

Базовая абстракция **Tk** – это **widget**. **Widget** – это, как правило, окно или подокно (**subwindow**), для чего-то в рамках **GUI** предназначенное. Например:



subwindow entry - Ввод текстовой строки

subwindow button – старт новой функции перевода строчных букв в заглавные

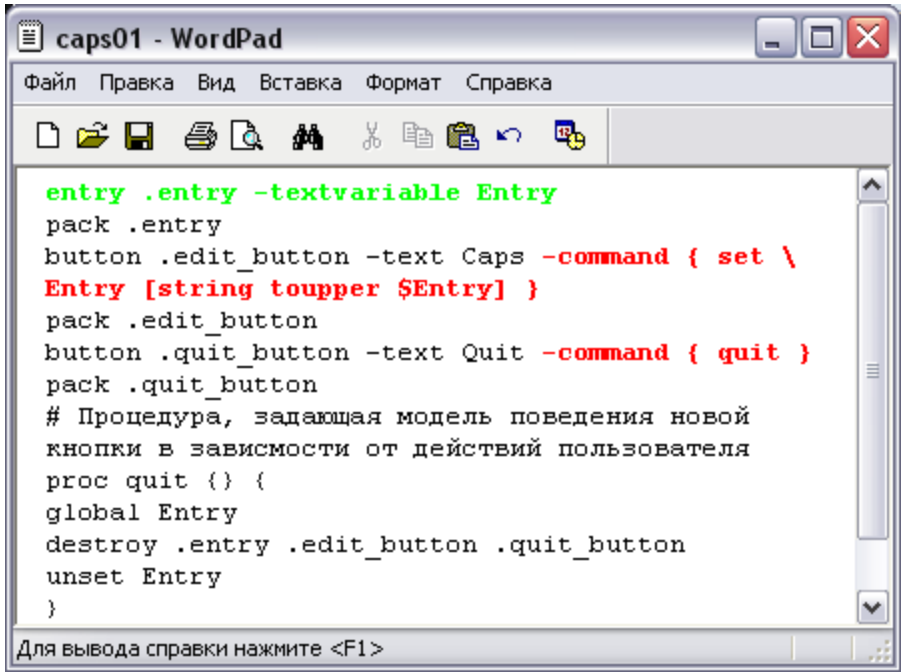
subwindow button – старт функции очистки корневого окна приложения **caps01**

Для обслуживания этих целей **(«чего-то») widget** по умолчанию имеет следующие методы:

- Он умеет сам себя воспроизводить на экране
- Он умеет обрабатывать ввод, направленный на него, и поступающий от клавиатуры или мыши

В нашем примере **GUI-Caps01** имеются три **widget'a**. Два из них – **button** – предназначены для действий (**action**), которые реализуются при клике на соответствующие **subwindows** (**WIDGET-SPECIFIC OPTIONS** **-command + процедура quit).**

С **widget'ом entry** не связано никаких действий (**action**). Для нашего **GUI** достаточно модели поведения по умолчанию этого **widget'a.**



Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

У программиста в Tcl/Tk существует 4 основных способа работы с Widget'ами.

1. **Можно создать widget одного из заданных типов. Это действие выполняется с помощью команды, которая относится к типу widget making command. Мнемоническое имя команды совпадает с типом widget'a (button, label, entry и т.д.); создаваемый widget обязательно должен быть поименован.**

   Например: `entry .entry –textvariable Entry`

2. **Можно изменять характеристики уже созданных widget'ов. Это может быть реализовано с помощью действий (actions) над объектами т.н. action family, которая создается вместе с widget'ом. Выполняться эти действия могут с помощью процедур, имена которых совпадают с именем widget'a. Эти процедуры называются widget command.**

   Например: `.entry configure –textvariable SecondEntry`

Здесь action `configure` **изменяет значение опции** `–textvariable` **widget'a с именем** `.entry`

Итак, можно констатировать, что в Tk различают три типа команд: widget making command, widget command и nonwidget command. Последний тип будет рассмотрен позже.

3. **Widget'ы могут по разному позиционироваться и заполнять область воспроизведения. Для этих целей в Tk служит геометрический менеджер, как совокупность диспетчеров компоновки. Widget не может себя отобразить в корневом окне Tk runtime sysytem не будучи обработанным одним из диспетчеров.**

   Например: `pack .entry`

4. **Widget'ы могут быть уничтожены. Для этого служит команда destroy.**

   Например: `destroy .entry`

Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

В общем случае – программирование в Tcl/Tk - это генерирование **widget'ов** и последующее изменение их предопределенного (заданного по умолчанию) внешнего вида (представления) и поведения (связанные с этим **widget'ом actions**), т.е. реализация **GUI**, умеющего обрабатывать **I/O events**.

Описанная задача может быть решена за счет задания значений опций первых двух типов команд, описанных выше: **widget making command, widget command**

Воспроизведение запрограммированного **GUI** возможно только после применения диспетчеров упаковки – геометрического менеджера (**pack, grid, place**).

Поведение ряда **widget'ов** определяется исполнением скрипта (**Tcl**) при выполнении некоторых условий. Контроль того, что за скрипт должен быть выполнен, осуществляется за счет задания значения опции **–command**.

Скрипт выполняется как результат действий пользователя с клавиатурой или мышью. За счет опции **–command Tk-**программист может работать с клавиатурой и мышью на самом высоком уровне абстракции.

Однако, в ряде случаев у программиста возникает потребность работать на более низком уровне – т.е. включать в создаваемый **GUI** обработку таких, зависящих от пользователя **events**, как **key press, key release, mouse movement**, или **mouse click.**

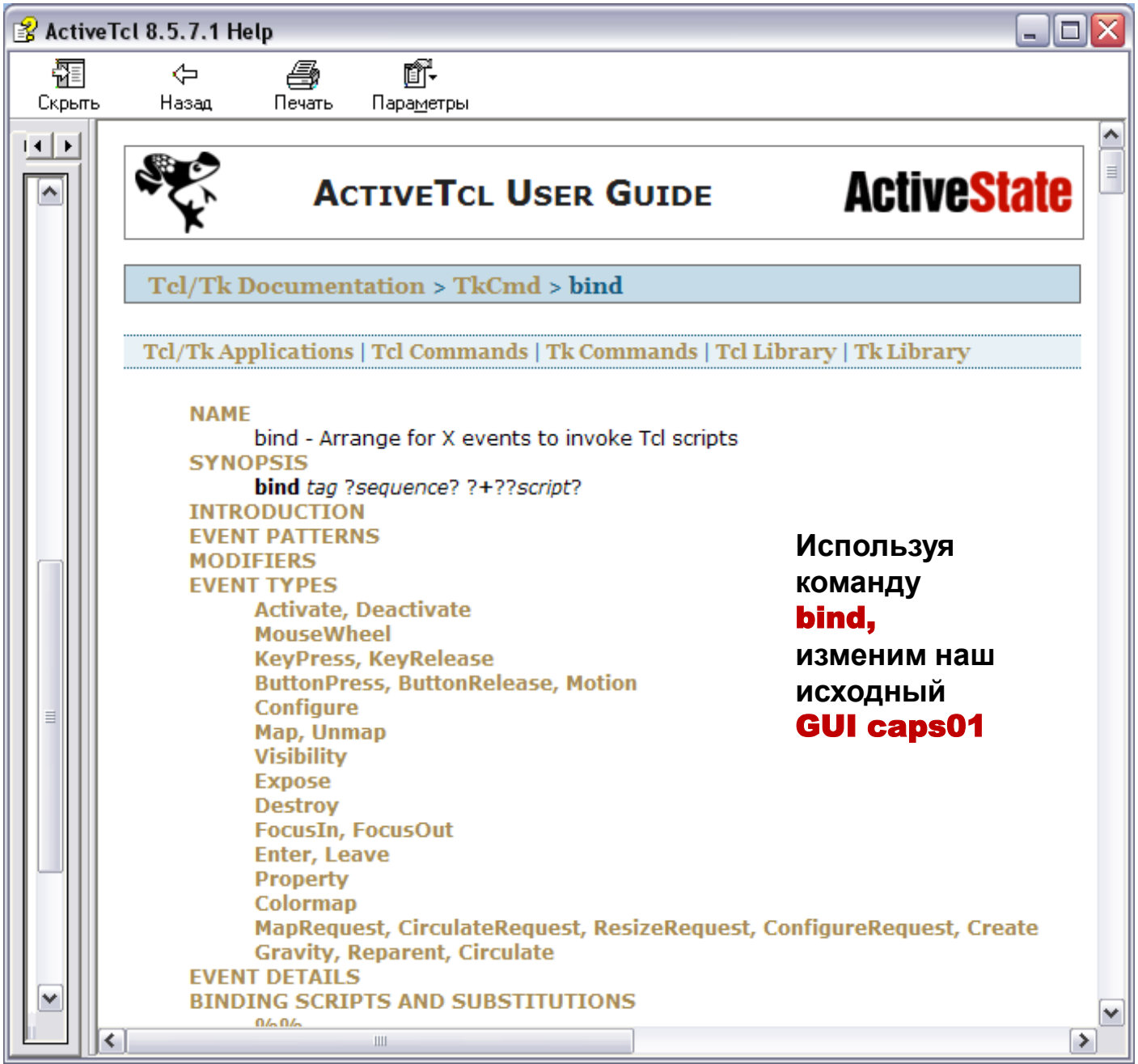
Осуществлять такой контроль в **Tk** возможно с помощью команды специального, третьего типа **nonwidget command –** команды **bind**.
Осуществлять такой контроль в **Tk** возможно с помощью команды специального, третьего типа **nonwidget command –** команды **bind**, которая позволяет непосредственно связать исполняемые скрипты с **user-related events**.

Иначе говоря, команды **bind** позволяет программисту сказать: «Если произошло то или иное пользовательское событие, я отвечаю за выполнение ассоциированного с этим событием скрипта».

Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

В общем случае, команда bind ассоциирует Tcl-скрипты с X – events.



Используя команду bind, изменим наш исходный GUI caps01

Средства разработки графических интерфейсов в системе X Window System.

Что это значит – программировать с помощью Tcl/Tk?

В GUI caps01 пользователь вводит символьные последовательности и с помощью специальной командной кнопки преобразует введенные строчные символы в заглавные. Добавим к нашему GUI обратную функцию – возврат преобразованных символов в их исходное состояние, не изменяя внешний вид самого интерфейса. Для этого задействуем стандартное X – events <Key-Tab>, т.е. нажатие клавиши <Tab>:

caps02 - WordPad

Файл Правка Вид Вставка Формат Справка

```
proc initForCaps {} {
global switchState
.edit_button configure -text Caps -command { set Entry [string
toupper $Entry] }
set switchState initForSmall
}
proc initForSmall {} {
global switchState
.edit_button configure -text Small -command { set Entry [string
tolower $Entry] }
set switchState initForCaps
}
proc quit {} {
global Entry switchState
destroy .entry .edit_button .quit_button
unset Entry switchState
bind . <Key-Tab> {}
}
entry .entry -textvariable Entry
pack .entry
button .edit_button
pack .edit_button
button .quit_button -text Quit -command { quit }
pack .quit_button
initForCaps
bind . <Key-Tab> { $switchState; break }
```

Для вывода справки нажмите <F1>