# Графические Системы. Часть II

# Лекция № 7

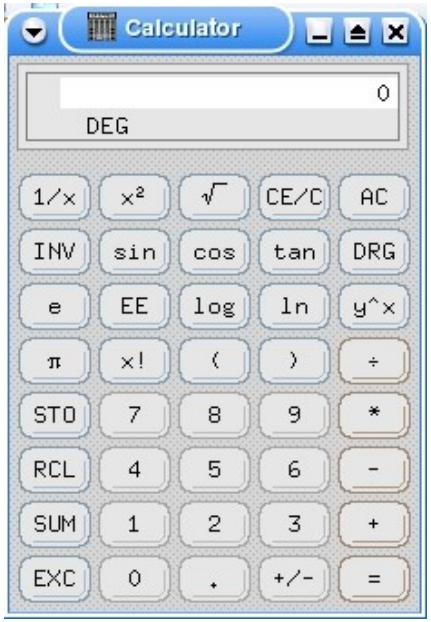
## Программирование графического пользовательского интерфейса средствами X-WINDOW.

## ИПВУ. Управляющие объекты – widget’ы и их ресурсы. Иерархия widget’ов

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

**Вспомним окончание прошлой лекции:**




**Строки Ресурсного файла XcalcAppDefaults для X-Клиента Xcalc легко могут быть проинтерпретированы:**

```
XCalc*ti.button3.translations:
#override<Btn1Down>,<Btn1Up>:squareRoot()unset()
XCalc*ti.button4.label: CE/C
XCalc*ti.button4.translations:
#override<Btn1Down>,<Btn1Up>:clear()unset()
XCalc*ti.button5.label: AC
XCalc*ti.button5.translations:
#override<Btn1Down>,<Btn1Up>:off()unset()\n\
<Btn3Down>,<Btn3Up>:quit()
XCalc*ti.button6.label: INV
XCalc*ti.button6.translations:
#override<Btn1Down>,<Btn1Up>:inverse()unset()
XCalc*ti.button7.label: sin
XCalc*ti.button7.translations:
#override<Btn1Down>,<Btn1Up>:sine()unset()
XCalc*ti.button8.label: cos
```

events

actions

Ресурс `translations` для **widget'a** `XCalc*ti.button6`

Значение Ресурса `translations` для **widget'a** `XCalc*ti.button6`

**Таким образом формируется translation table для Приложения X-Клиента Xcalc.**

**При этом среди action могут быть как стандартные, так и написанные программистом.**

# Основы программирования в системе X Window System.
# Афинская Иерархия классов Widget'ов

Остановимся подробнее на правилах формирования и синтаксисе **translation** для записи их в ресурсных файлах или исходном коде приложения прикладным программистом:

```
[модификатор, …, модификатор]<event>,…,<event>[число повторений события - event][детализация]   :   имя action ([аргументы])
[имя action ([аргументы])…. ]
```

- Поле `<event>` включает одно или несколько событий из списка стандартных для **X Window** (`Key[Press], KeyUp, ButtonPress, ButtonRelease, Btn1Down, Btn1Up, MotionNotify – MouseMoved` и т.д. – см. документацию на **X11**).
- В поле «`модификатор`» могут использоваться стандартные модификаторы, поддерживаемые **Xt**: `Ctrl, Shift, Lock, Any, None, Mod1,….Mod5` (дополнительные клавиши-модификаторы), `Button1,…, Button5` (кнопки мыши) и т.д.
- Если модификаторов в **translation** нет, то указанные в строке (записи) **action** (действия) и соответствующие им процедуры будут вызываться в ответ на определенные в угловых скобках события (`<event>`) или последовательности событий независимо от того, в каком состоянии находятся клавиши-модификаторы и кнопки мыши. Например:

```
<Key[Press]>    :   KeyboardPrint ()
```

Означает, что процедура, соответствующая **action** с именем «`KeyboardPrint`», будет вызвана при нажатии любой клавиши, при этом кнопки мыши или клавиши-модификаторы могут находиться как в нажатом, так и в отжатом положении.

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

- Для конкретизации различных событий предусмотрено специальное поле «детализация». Так для событий от клавиатуры в этом поле указывается символ клавиши, который для X11 определен в файле «keysymdef.h». Например, следующая запись:

```
<Key[Press]>a     :   KeyboardPrint ()
```

используется для определения события , соответствующего нажатию клавиши «a». Для событий `ButtonPress, ButtonRelease, MotionNotify` детализация есть номер нажатой кнопки.

- Технология [клавиш] - модификаторов позволяет специфицировать специальное использование привычных клавиш-[модификаторов] , таких как `Ctrl, Shift, Meta, Lock, Alt` и т.д. (усугубить, так сказать), а также наоборот – принудительно отменить или уточнить действие любого случайно или специально вызванного модификатора перед (вместе с) отрабатываемым событием. Для этого служат модификаторы **None, !, :, ~**.
- Следующие таблицы определяют основные правила использования этих модификаторов и клавиш-модификаторов:

| Модификатор | Описание |
|---|---|
| None | Ни одна клавиша-модификатор не должна быть нажата; |
| ! | Событие должно сопровождаться только указанными модификаторами и никакими другими; |
| : | Различает события от нажатия клавиш в зависимости от регистра; |
| ~ | Клавиша-Модификатор, следующий за данным символом, не должна быть в нажатом состоянии; |

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

Следующие таблицы определяют основные правила использования этих модификаторов и клавиш-модификаторов:

| Клавиша-Модификатор | Описание |
|---|---|
| Ctrl<Key[Press]> | Событие <Key[Press]>, когда нажата клавиша **Control**; |
| Shift<Key[Press]> | Событие <Key[Press]>, когда нажата клавиша **Shift**; |
| ..... | И т.д.; |
| Btn1Down/Btn1Up | Событие <ButtonPress/ButtonRelease> для |
| ..... | |
| Btn5Down/Btn5Up | 1,.....5 – ой кнопки мыши; |
| BtnMotion | Событие <MotionNotify> (движение мыши); |
| Btn1Motion ,...., | Событие <MotionNotify>, когда нажата 1,.....,5 кнопка мыши; |
| Btn5Motion | |

Кроме этого важно использовать для записей строк **translation** следующие правила:

- Модификаторы, расположенные в начале **translation**, относятся ко всем событиям, указанным в ней. Например, следующие две записи эквиваленты:

```
Shift<Btn2Down>,<Btn2Up>          :       KeyboardPrint ()
Shift<Btn2Down>,Shift<Btn2Up>     :       KeyboardPrint ()
```

- В записи **translation** можно использовать несколько Клавиш-Модификаторов, например:

```
Ctrl Shift<Btn2Down>          :       KeyboardPrint ()
```

# Основы программирования в системе X Window System.
## Афинская Иерархия классов Widget'ов

- Для определения в **translation** событий от кнопок мыши используется модификатор `Button1,…..Button5`. Например, следующая запись

```
Button2<Key>a          :                KeyboardPrint ()
```

означает, что процедура **action** `KeyboardPrint` будет вызвана, когда нажимается вторая кнопка мыши и клавиша «а».

- Каждая **translation** может определить одно или несколько событий, разделяемых запятыми. Например, для определения последовательности событий, соответствующих нажатию и отжатию первой кнопки мыши, можно использовать следующую строку:

```
<Btn1Down>,<Btn1Up> :                KeyboardPrint ()
```

- Для задания нескольких нажатий и отжатий используется поле «`число повторений события`». Так, например, **translation** :

```
<Btn2Up>(2)    :                KeyboardPrint ()
```

Указывает, что соответствующая процедура будет вызвана при двойном нажатии второй кнопки мыши. Приведенная выше запись эквивалентна следующей:

```
<Btn2Down>,<Btn2Up>, <Btn2Down>,<Btn2Up> :     KeyboardPrint ()
```

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

- Для задания большего числа повторений в **translation** можно использовать символ «+». Например, следующая запись

```
<Btn3Up>(2+)        :                KeyboardPrint ()
```

означает, что процедура `KeyboardPrint` будет вызвана при двух и более нажатиях на третью кнопку мыши. Максимальное число «повторений» в **Xt** – 9. По умолчанию временной интервал, во время которого считаются нажатия кнопки мыши, составляет 200 миллисекунд. Для изменения этого параметра существует специальный ресурс `[XtSet]MultiClickTime())`

- Следует иметь в виду, что порядок расположения **translation** в соответствующей таблице (**translation table** ) существен, т.к. поиск нужного события в таблице начинается сверху и выполняется до певрого найденного. Поэтому записи **translation** следует таким образом, чтобы наиболее существенные события были записаны раньше. Например, если **translation table** содержит следующий фрагмент:

```
<Key>          :          InputSymbol () \n\
<Key>Return    :          EndInput ()
```

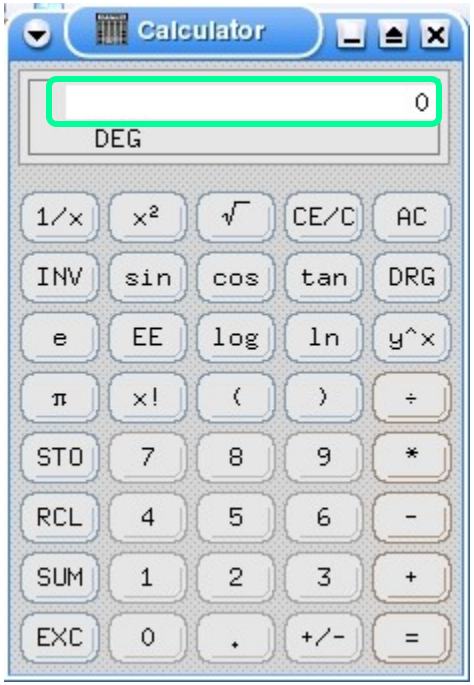
то процедура, соответствующая **action** `InputSymbol`, будет вызвана при нажатии клавиши `Return`, а процедура для `EndInput` никогда не будет вызвана.

Рассмотрим, как формируется **translation table** с использованием этих правил на примере **X-Клиента Xcalc**

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

**translation** из Ресурсного файла **XcalcAppDefaults** для **X-Клиента Xcalc**:




```
XCalc*ti.bevel.screen.LCD.width: 186
XCalc*ti.bevel.screen.LCD.translations:
#replace\n\
Ctrl<Key>c:quit()\n\
Ctrl<Key>h:clear()\n\
None<Key>0:digit(0)\n\
None<Key>1:digit(1)\n\
None<Key>2:digit(2)\n\
None<Key>3:digit(3)\n\
None<Key>4:digit(4)\n\
None<Key>5:digit(5)\n\
None<Key>6:digit(6)\n\
None<Key>7:digit(7)\n\
None<Key>8:digit(8)\n\
None<Key>9:digit(9)\n\
```

Процедура, соответствующая **action** «quit»

Процедура, соответствующая **action** «очистить bevel.screen.LCD»

Процедура, соответствующая **action** «вывод цифры «2» на bevel.screen.LCD»

Клавиша-модификатор Ctrl+ **event** <нажать клавишу c>

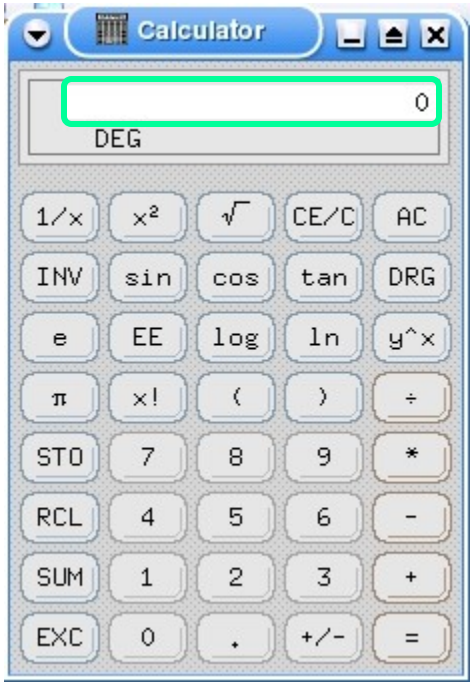
Клавиша-модификатор Ctrl+ **event** <нажать клавишу h>

**event** <нажать клавишу 2>; использовать какую-либо иную Клавишу-модификатор запрещено

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

**translation** из Ресурсного файла **XcalcAppDefaults** для **X-Клиента Xcalc**:




```
XCalc*ti.bevel.screen.LCD.width: 186
XCalc*ti.bevel.screen.LCD.translations:
#replace\n\
.....
:<Key>.: decimal()\n\
:<Key>+: add()\n\
:<Key>-: subtract()\n\
:<Key>*: multiply()\n\
:<Key>/: divide()\n\
:<Key>(: leftParen()\n\
:<Key>): rightParen()\n\
:<Key>!: factorial()\n\
.....
<Key>space      :   clear()\n\
<Key>q          :   quit()\n\
<Key>Delete     :   clear()\n\
<Key>BackSpace  :   clear()\n\
<Btn1Down>,<Btn1Up> : toggle()selection()\n
```

Процедуры, соответствующие указанным **actions**

Процедуры, соответствующие очистке экрана и выходу из **Xcalc**

Модификатор «:» вместе с **events** <нажать клавишу> .,+,-,*,/,(,),! На верхнем регистре

**events** <нажать клавишу> space, q, Delete, BackSpace.
При этом неважно – нажаты другие клавиши-модификаторы или нет!

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

### Основные events и Key Modifiers (Клавиши-Модификаторы) в X11

*Table F-1. Event Types and Their Abbreviations*

| Event Name | Event Type | Abbreviations/Synonyms |
|---|---|---|
| KeyPress | Keyboard | Key, KeyDown |
| KeyUp | Keyboard | KeyRelease |
| ButtonPress | Mouse Button | BtnDown |
| ButtonRelease | Mouse Button | BtnUp |
| Btn1Down | Mouse Button Press | |
| . | | |
| . | | |
| Btn5Down | | |
| Btn1Up | Mouse Button Release | |
| . | | |
| . | | |
| Btn5Up | | |

*Table F-2. Key Modifiers*

| Event Modifiers | Abbreviation |
|---|---|
| Ctrl | c |
| Meta | m |
| Shift | s |
| Lock | l |
| Any | |
| ANY | |
| None | |
| Mod1 | 1 |
| . | . |
| . | . |
| Mod5 | 5 |

**Следует помнить, что модификаторы Mod1 … Mod5как правило очень сильно системно-заивисимы и не могут быть применены на всех серверах**

*Table F-1. Event Types and Their Abbreviations (continued)*

| Event Name | Event Type | Abbreviations/Synonyms |
|---|---|---|
| MotionNotify | Mouse Motion | Motion, MouseMoved, PtrMoved |
| ButtonMotion | Motion w/any Button Down | BtnMotion |
| Button1Motion | Motion w/Button Down | Btn1Motion |
| . | | . |
| . | | . |
| Button5Motion | | Btn5Motion |
| EnterNotify | Mouse in Window | Enter, EnterWindow |
| LeaveNotify | | LeaveWindow, Leave |
| FocusIn | Keyboard Input Focus | |
| FocusOut | | |
| KeymapNotify | Changed Key Map | Keymap |
| ColormapNotify | Changed Color Map | Clrmap |
| Expose | Related Exposure Events | |
| GraphicsExpose | | GrExp |
| NoExpose | | NoExp |
| VisibilityNotify | | Visible |
| CreateNotify | Window Management | Create |
| DestroyNotify | | Destroy |
| UnmapNotify | | Unmap |
| MapNotify | | Map |
| MapRequest | | MapReq |
| ReparentNotify | | Reparent |
| ConfigureNotify | | Configure |
| ConfigureRequest | | ConfigureReq |
| GravityNotify | | Grav |
| ResizeRequest | | ResReq |
| CirculateNotify | | Circ |
| CirculateRequest | | CircReq |
| PropertyNotify | | Prop |
| SelectionClear | Intra-client Selection | SelClr |
| SelectionRequest | | SelReq |
| SelectionNotify | | Select |

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

**И наконец, последний тип динамических ресурсов – это обработчики событий Event Handler.**

**В Xt предусмотрен еще один механизм использования процедур для выполнения определенных действий (action) при наступлении тех или иных событий. Это механизм обработчиков событий – event handler. Он позволяет определить отдельное событие (или их группу) - event, при наступлении которого будет вызвана указанная программой процедура-обработчик. Обработчики событий менее гибки, чем action-процедуры, но зато значительно быстрее. Последнее происходит потому, что при вызове event handler не требуется поиск процедур по соответствующим таблицам.**

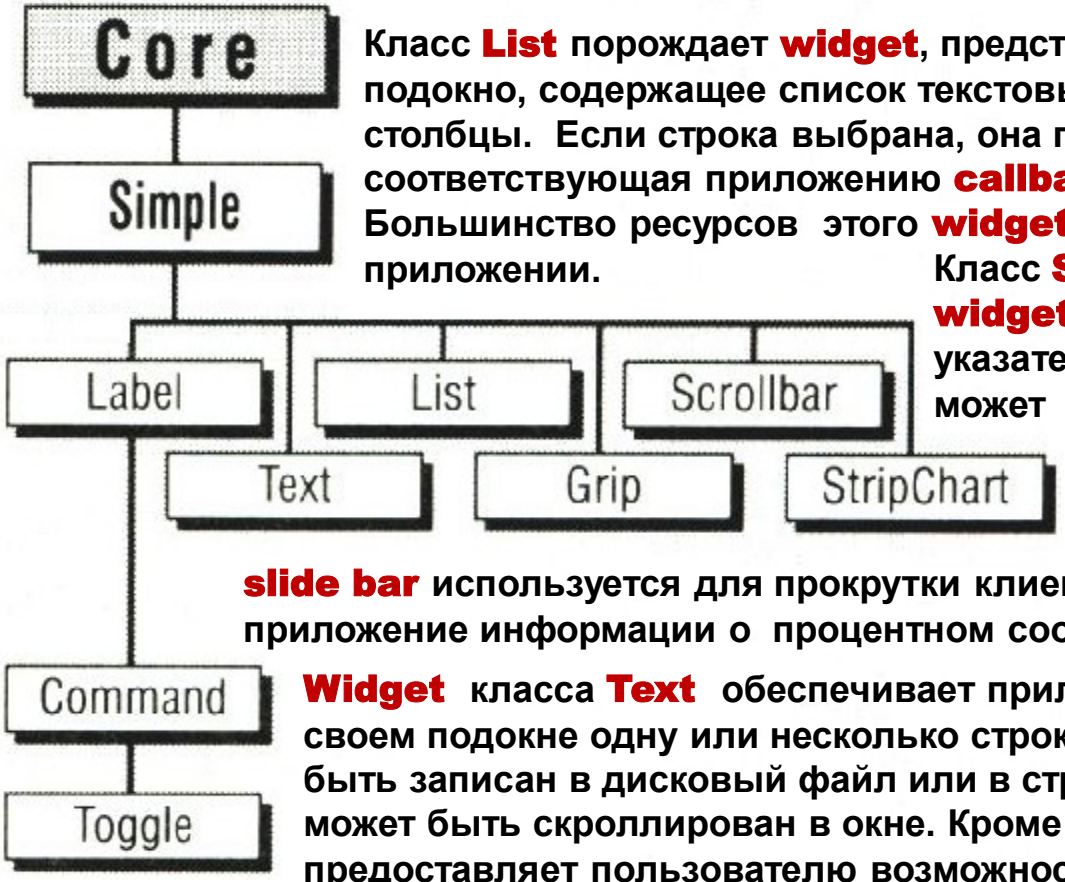
**Для того, чтобы приложение имело возможность использовать данную процедуру в качестве event handler, ее необходимо зарегистрировать с помощью специальных процедур Xt. При этом в данных процедурах (XtAddEventHandler) используются в качестве параметров не только идентификатор управляющего объекта – widget'a, к которому добавляется процедура-обработчик, и сама процедура-обработчик, но и комбинация флагов, задающая события, в ответ на которые будет вызываться регистрируемая процедура, а также указатель на данные, передаваемые в event handler при его вызове.**

**Для одного и того же события в списке обработчиков может быть зарегистрировано несколько функций.**

# Основы программирования в системе X Window System.

## Афинская Иерархия классов Widget'ов

В заключение коротко остановимся на оставшихся подклассах базового класса **Siumple**:




Класс **List** порождает **widget**, представляющий из себя прямоугольное подокно, содержащее список текстовых строк, сгруппированных в строки и столбцы. Если строка выбрана, она подсвечивается, и инициируется соответствующая приложению **callback-процедура** для выполнения. Большинство ресурсов этого **widget'a** предназначено для использования в приложении.

Класс **Scrollbar** порождает прямоугольный **widget**, содержащий область прокрутки и указатель (**slide region & slide bar**). **Scrollbar** может использоваться самостоятельно - для реализации градуированных шкал, или в составе composite **widget'ов** например – **Viewport**.

**slide bar** используется для прокрутки клиентских данных или возвращения в приложение информации о процентном соотношении просмотренных данных.

**Widget** класса **Text** обеспечивает приложению возможность отобразить в своем подокне одну или несколько строк текста. Отображенный текст может быть записан в дисковый файл или в строку памяти. Отображаемый текст может быть скроллирован в окне. Кроме того, **widget** класса **Text** предоставляет пользователю возможность редактировать отображенный текст, осуществлять поиск специфицированных сочетаний символов.


**Widget** класса **StripChart** используется для обеспечения графиков в режиме реального времени в виде единичных столбиков. Этот **widget** используется **X-Клиентом xload** для отображения графика загрузки процессора. Данные считываются приложением и изменяют величину столбца на интервале, заданном ресурсом `update`.

# Основы программирования в системе X Window System.

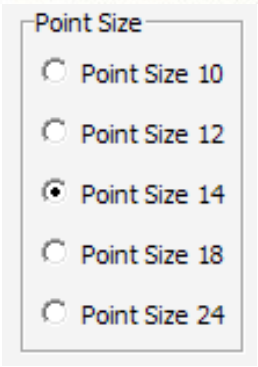
## Афинская Иерархия классов Widget'ов


Класс **Grip** обеспечивает небольшую область, которая позволяет нажать и отжать кнопку – иными словами, этот **widget** может быть определен как область чувствительности. **Grip widget** используется для добавления точки, которая используется для перепозиционирования объекта. Пример – **pane border** для **Paned widget'a.**

Класс **Toggle** порождает прямоугольный **widget**, содержащий **текстовую** или **pixmap label**. Этот **widget** поддерживает **Boolean**-состояние (**True/False** или **On/Off**) и меняет его после выбора.

Когда курсор попадает в область подокна **widget'a**, его границы подсвечиваются, сигнализируя о том, что он готов к выбору. Если **button1** мыши нажата и отжата, **Toggle widget** меняет свое состояние, сигнализируя об это реверсированием **fg- и bg-цветов**. При этом вызывается его **notify-action**.

Если курсор оказывается за пределами подокна **Toggle widget'a** до того, как кнопка мыши была отжата, **widget** возвращается в свое нормальное цветовое (**fg и bg**) состояние, а нажатие клавиши не дает никакого эффекта. Такое поведение **Toggle widget'a** дает возможность **user'y** прервать выполнение **action**.




**Toggle widget** часто используется для реализации кнопок типа **radio group**.

**Radio group** – это список кнопок типа **Toggle widget**, из которых в данный момент времени может быть выбрана только одна.