Графические Системы. Часть II

Лекция № 6

Программирование графического пользовательского интерфейса средствами X-WINDOW.

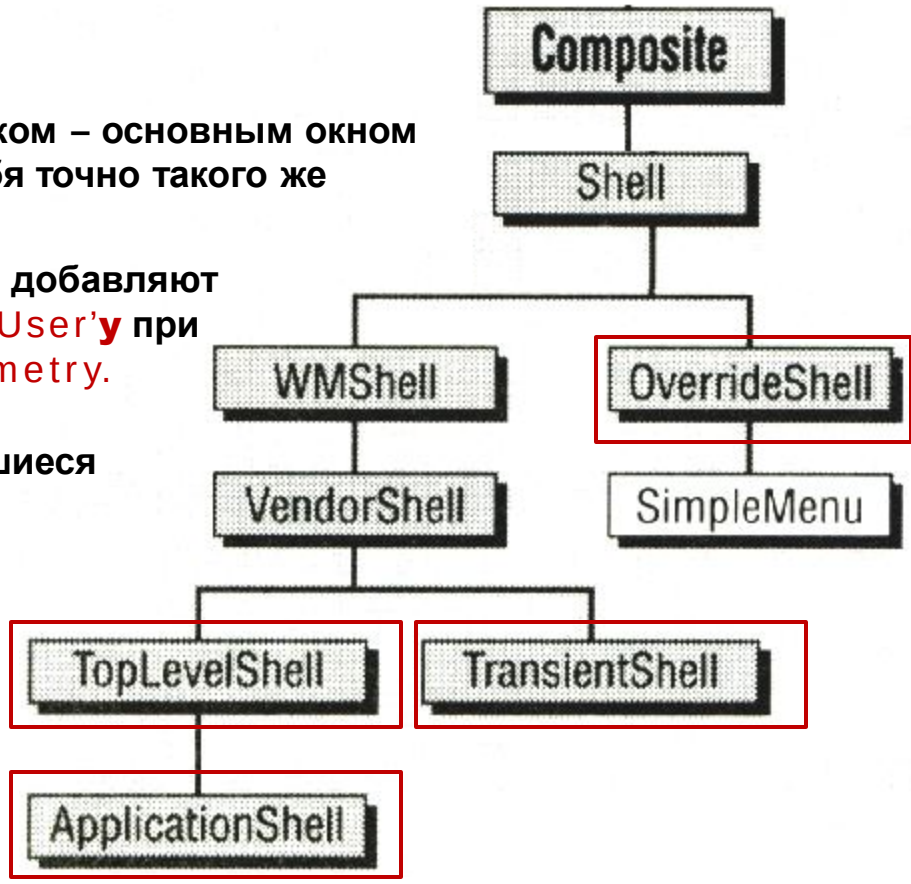
ИПВУ. Управляющие объекты – widget’ы и их ресурсы. Иерархия widget’ов

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

Рассмотрим еще одну очень важную группу подклассов класса Composite, а именно подветку Shell:

Shell.
Widget этого класса управляет только одним потомком – основным окном приложения. Этот widget не виден, но он делает себя точно такого же размера, как окно приложения, располагаясь за ним.

Widget'ы этого класса очень важны, т.к. именно они добавляют ресурсы, чрезвычайно важные для программистов. User'у при этом остается доступным только один ресурс – geometry. Shell класс имеет семь подклассов, три из которых используются для внутреннего управления, а оставшиеся четыре – для программирования приложений.



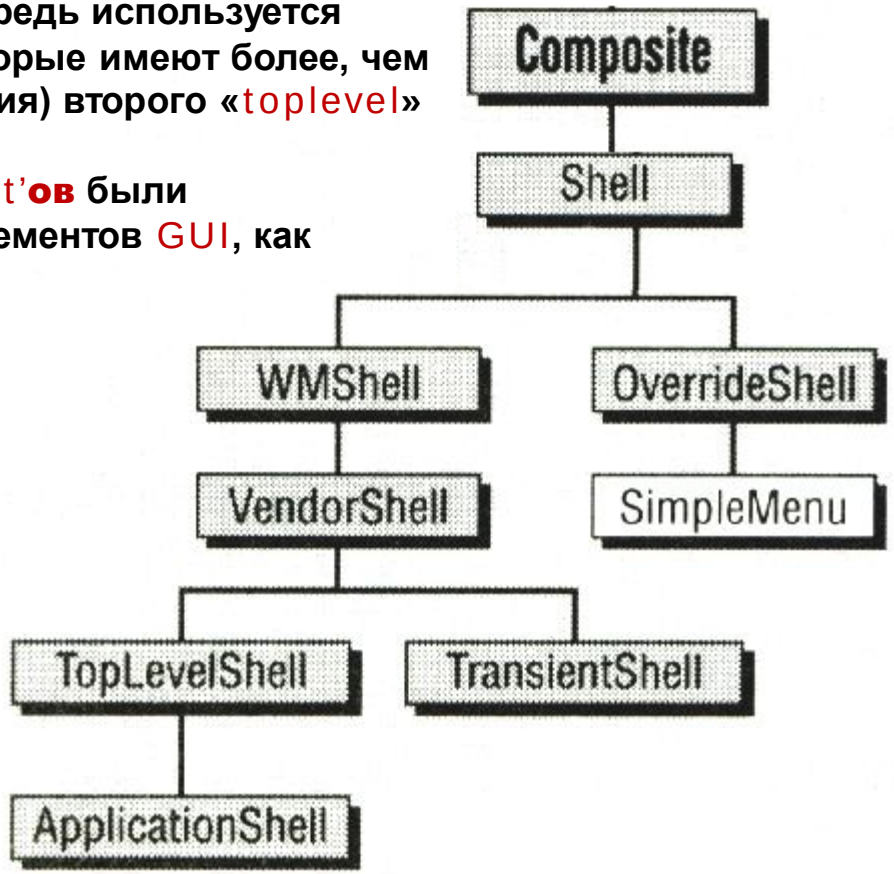
Одной из важнейших задач, для решения которой предназначены widget'ы класса Shell – это взаимодействие с менеджером окон. Для этого в афинской иерархии предусмотрен специальный подкласс класса Shell – WMShell. Он содержит дополнительные поля, необходимые для взаимодействия с менеджером окон.

WMShell содержит класс VendorShell. Этот класс предназначен для того, чтобы дать возможность взаимодействия со специальными менеджерами окон.

Подкласс этого класса TopLevelShell предназначен для назначения ресурсов основного окна приложения, управляемого при этом базовым WM (это такие ресурсы, как иконизация, возможность минимизации, максимизации, перемещения по экрану, всплытия и погружения).

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

Подкласс этого класса ApplicationShell в свою очередь используется прикладными программистами для приложений, которые имеют более, чем одно независимое окно, как класс для его (приложения) второго «toplevel» окна. Также может управляться базовым WM.
В рассматриваемой нами Афинской иерархии widget'ов были предусмотрены и специальные классы для таких элементов GUI, как всплывающие окна. Это классы: TransientShell и OverrideShell, для которых взаимоотношения с базовым оконным менеджером строятся по разному:



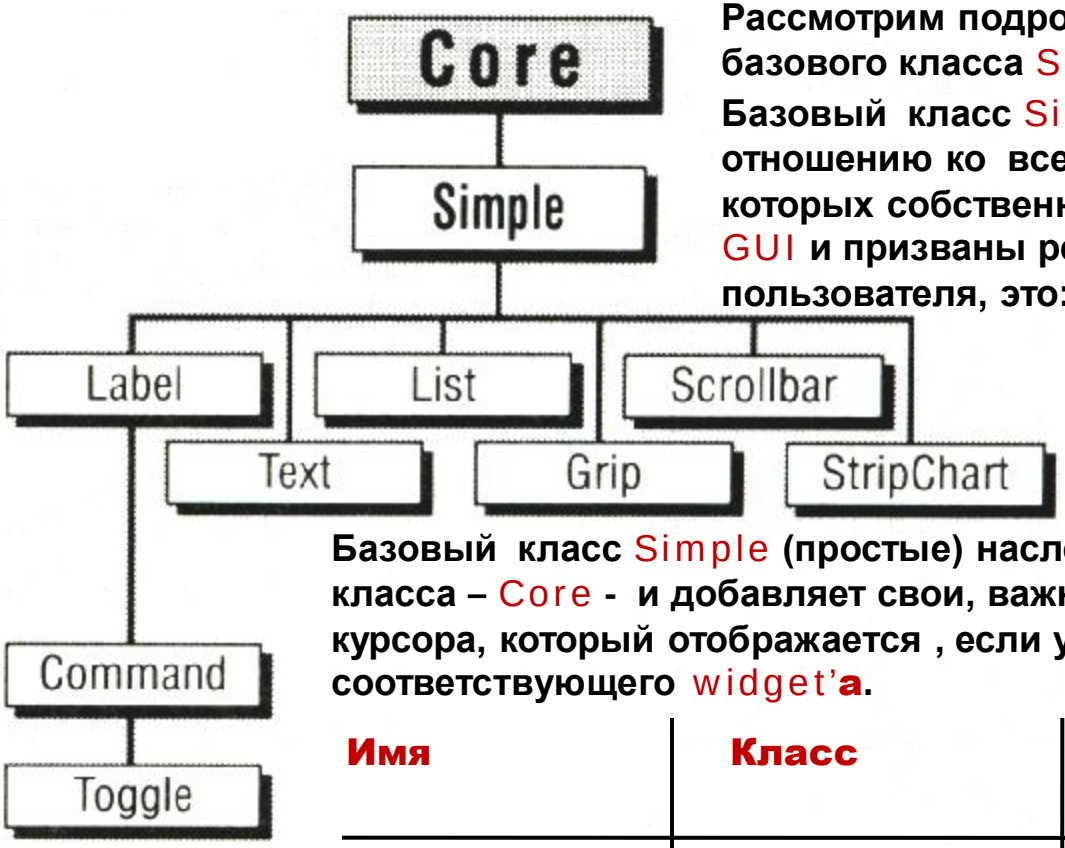
Widget'ы класса TransientShell – предназначены для всплывающих меню (popups), но могут быть управляемы базовым WM: независимо от основного окна приложения эти widget'ы могут быть перемещены или их размер может быть изменен. Иконизация для таких widget'ов запрещена. Но если в пиктограмму превращается родитель объекта класса TransientShell, то окно widget'а убирается с экрана. Класс TransientShell используется для создания диалоговых окон и Help-Window.

OverrideShell widget'ы не управляются базовым WM: для окна widget'а данного класса атрибут *override_redirect* устанавливается в значение *True*, т.е. менеджер окон его не контролирует. Как правило, окна объектов этого класса не имеют аксессуаров, добавляемых менеджером окон (заголовок, рамка, стандартные кнопочки), и используются в основном для создания всплывающих меню.

Переходим к рассмотрению правой части иерархии – подклассов базового класса Simple

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Рассмотрим подробнее основные подклассы Widget'ов базового класса Simple

Базовый класс Simple является корневым по отношению ко всем подклассам, инстанции - widget'ы - которых собственно и формируют содержательную часть GUI и призваны реагировать на действия конечного пользователя, это:

всевозможные метки, списки, меню, скроллбары, текстовые окна, команды, переключатели и пр.

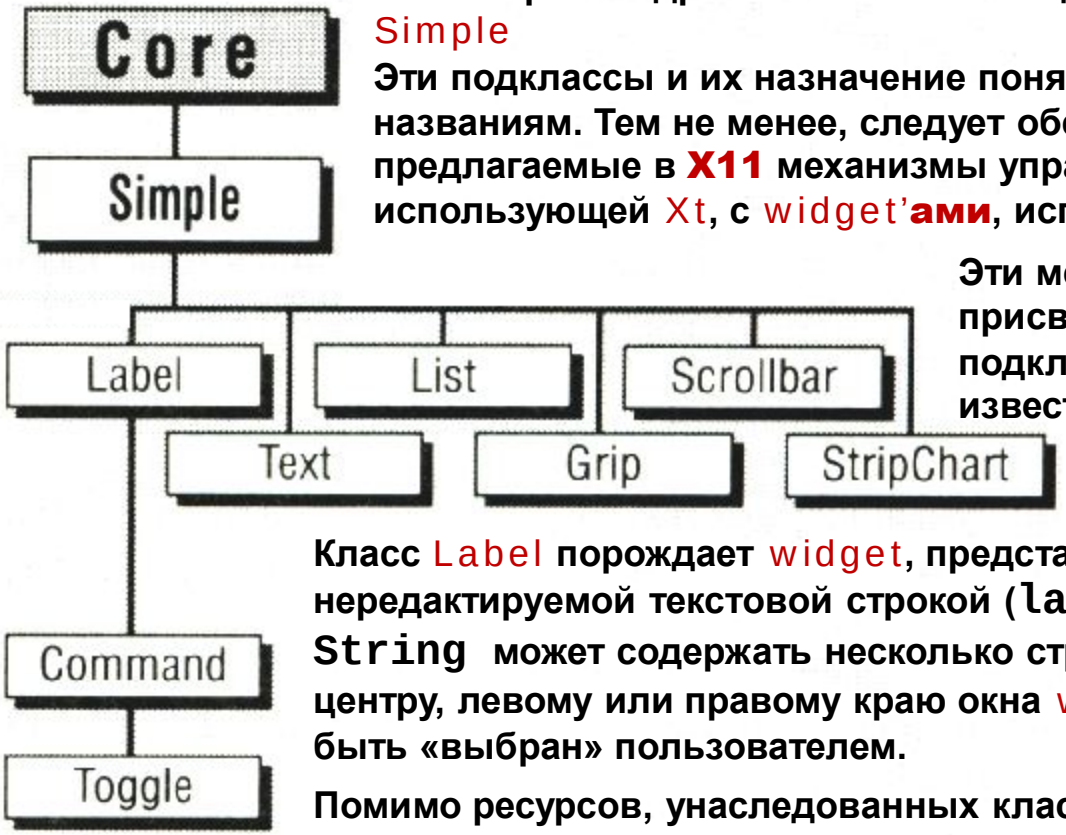
Базовый класс Simple (простые) наследует все ресурсы своего базового класса – Core - и добавляет свои, важнейшим из которых является тип курсора, который отображается , если указатель оказывается в области окна соответствующего widget'а.

Имя	Класс	Значение по умолчанию
background	Background	
borderColor	BorderColor	
borderWidth	BorderWidth	1
height	Height	0
width	Width	0
X	Position	0
Y	Position	0
cursor	Cursor	0

Кроме того, базовый класс Simple (простые) добавляет ресурс insensitiveBorder – границы «нечувствительности», если widget получает ресурс insensitive

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Рассмотрим подробнее основные подклассы Widget'ов базового класса Simple

Эти подклассы и их назначение понятно со всей очевидностью по их названиям. Тем не менее, следует обсудить некоторые их особенности и предлагаемые в **X11** механизмы управления взаимодействия программы, использующей Xt, с widget'**ами**, используемыми в GUI.

Эти механизмы также реализуются за счет присвоения widget'**ам** – инстанциям этих подклассов ресурсов особого типа, помимо уже известных нам т.н. **статических ресурсов**.

Итак, первым рассмотрим подкласс Label базового класса Simple:

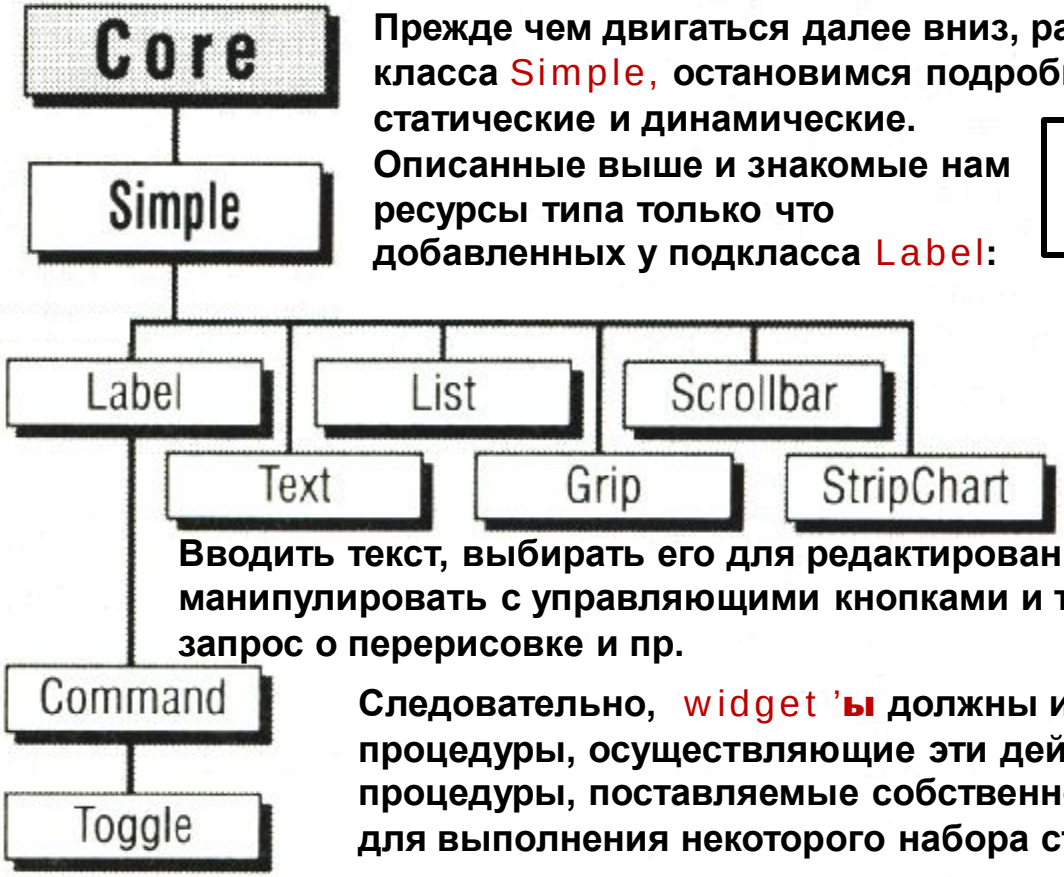
Класс Label порождает widget, представляющий из себя подокно с нередактируемой текстовой строкой (`label string`) или `bitmap'ом`. `Label String` может содержать несколько строк символов, выравненных (`justify`) по центру, левому или правому краю окна widget '**а.** Widget класса Label не может быть «выбран» пользователем.

Помимо ресурсов, унаследованных классом Label из своих базовых классов, управляющие элементы GUI – widget'**ы** этого класса гарантированно получают следующие ресурсы своего подкласса:

```
Bitmap (class Pixmap), font, foreground, internalHeight
(class Height), internalWidth (class Width), justify,
label, resize, rowspacing.
```

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Прежде чем двигаться далее вниз, рассматривая widget'**ы** подклассов базового класса Simple, остановимся подробнее на классификации ресурсов на статические и динамические.

Описанные выше и знакомые нам ресурсы типа только что добавленных у подкласса Label:

```
Bitmap, font, foreground,
internalHeight, internalWidth,
justify, label, resize, rowspacing.
```

в большинстве своем задают т.н. «статику» widget'**а** – размер, цвет, фонт и т.д. Но кроме этого, управляющий элемент предназначен для того, чтобы дать возможность пользователю выполнять определенные действия:

Вводить текст, выбирать его для редактирования, выбирать элементы из списков данных, манипулировать с управляющими кнопками и т.д. и т.п. Widget должен реагировать на запрос о перерисовке и пр.

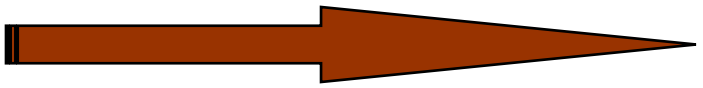
Следовательно, widget '**ы** должны иметь возможность «вызывать» процедуры, осуществляющие эти действия. При этом очевидно, что процедуры, поставляемые собственно X11, можно использовать только для выполнения некоторого набора стандартных действий.

Для реализации специфических, обусловленных алгоритмом самого приложения, операций, ему необходимо предоставить механизмы, позволяющие устанавливать связь между widget'**ами** и процедурами самого приложения, которые эти операции выполняют.

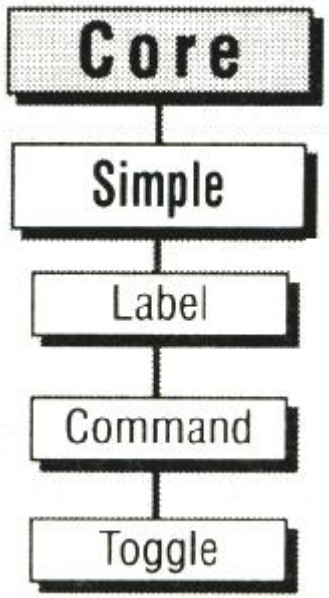
Нам уже известны те три способа (механизма), которые предусмотрены в инструментариях X11 для осуществления такой связи и передачи управления в процедуры приложения:

это использование Callback – процедур, Action –процедур и Event handlers - обработчиков событий.

Рассмотрим их подробнее в связи с классом Command



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



Command widget представляет из себя подокно, чаще всего – прямоугольное, содержащее text- или pixmap-label, которое осуществляет вызов процедуры приложения с ним связанную, если произведен выбор этой кнопки (Command widget) путем «нажатия» после совмещения с подокном pointer'а мыши. Ресурсы Command widget позволяют осуществлять индикацию его состояния – подсветка границы при попадании pointer'а мыши в область кнопки, реверсирование цветов fg и bg при нажатии на кнопку.

Помимо ресурсов, унаследованных от своих базовых классов (Core, Simple, Label), класс Command добавляет своим инстанциям новые:

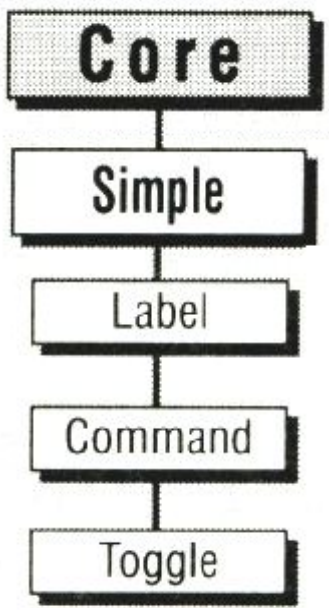
- `highlightThikness (class Thickness)-` толщина линии подсветки;
- `shapeStyle (class ShapeStyle)-` стиль кнопки; значения – `rectangular, oval, ellipse`, в некоторых случаях – `roundedRectangle;`
- `cornerRoundPercent(class CornerRoundPercent)-` в случае, если кнопка не прямоугольна, ресурс контролирует радиус скругления ее углов;

Перечисленные выше ресурсы относятся к разряду **статических**. Рассмотрим подробнее **динамические** ресурсы на примере класса Command.

Как уже отмечалось, с каждым widget'ом связан набор операций, которые над ним можно производить. Так, любой управляющий элемент может быть **создан и уничтожен**, нажимаемая кнопка (push button) может быть **«нажата»** с помощью мыши или **клавиши «пробел»** и т.д. Эти действия (события) являются **предопределенными**. Они не связаны собственно в алгоритмом, реализованным в приложении.

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Каждому из таких **предопределенных** действий соответствует ресурс widget'**а** (динамический ресурс), **значением** которого является **список процедур** библиотеки Xt и самого приложения. Эти процедуры автоматически вызываются, если происходит **предопределенное** действие.

Эти процедуры и называются **процедурами обратного вызова** – callback-**процедурами.** Тип этих ресурсов – `CallbackList`.

В X11 все widget'**ы** имеют по крайней мере один список callback. Он соответствует операции уничтожения управляющего объекта - widget'**а** и наследуется из класса Core. Имя этого ресурса:

- `destroyCallback (class Callback)`

Приложение может добавлять и убирать в (из) список (списка) callback. Для этого существуют специальные процедуры Xt.

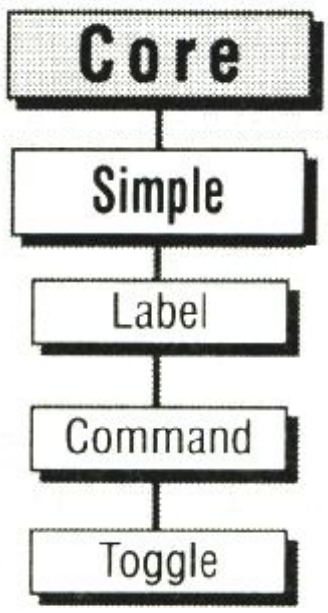
Механизм callback-**процедур** прост и удобен, однако имеет один существенный недостаток. В частности, callback-**процедуры** не могут вызываться в ответ на действие, не определенное для данного класса управляющих объектов - widget'**ов**. А предусмотреть все возможные действия, которые могут возникать при взаимодействии GUI **приложения** и его **пользователя**, в списке стандартных процедур невозможно.

Для преодоления этого недостатка в **X11** был сформулирован и реализован еще один механизм , дающий возможность прикладному программисту (и пользователю) связать код приложения с управляющими объектами GUI – widget'**ами**.

Этот механизм состоит в использовании специальных процедур – «ответов на действия» или action-**процедур.** Action-**процедура** – это функция, которая вызывается в ответ на определенные события (events), порождаемые X-**сервером**. Эти events связаны с теми или иными действиями пользователя - нажатием на кнопку мыши или клавишу на клавиатуре и пр.

Рассмотрим, как работает этот механизм на примере класса Command

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



У класса Command впервые появляется конструкция, получившая название Translation & Action.

[Event]Translation («перевод события») - еще один тип ресурсов widget'а, который может реализовать связь GUI, из этих widget'ов состоящего, с Приложением, обеспечив стандартную модель поведения интерфейса с точки зрения реакции его элементов на действия пользователя.

Рассмотрим этот тип динамических ресурсов widget'ов.

Во время работы с Приложением пользователь может взаимодействовать с его GUI, в частности с его элементами типа push button (command widget), особым способом:

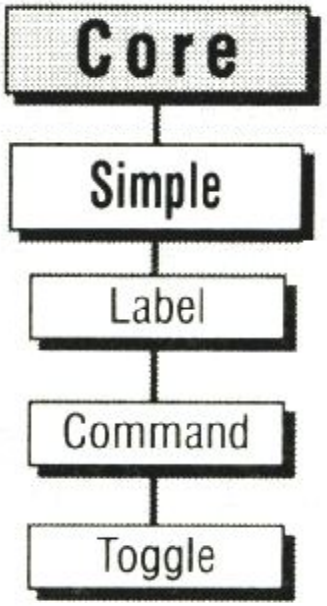
- **переместить pointer в область окна widget'а;**
- **«нажать клавишу» УВВ типа мышь;**
- **«отжать клавишу» УВВ типа мышь;**
- **вывести pointer из области окна widget'а;**

Список подобных действий пользователя в GUI Приложения (не только с command widget'ами) имеет конечную длину. Эти действия пользователя вызывают порождение определенных событий (events) X-сервером, и эти события, а самое главное – реакцию на них со стороны системы, тоже хотелось бы стандартизовать за счет связывания их с управляющими элементами GUI (widget'ами) через механизм ресурсов, в частности – динамического ресурса типа Event Translation .

Событие – event, порожденное X-сервером в ответ на перечисленные выше действия пользователя, представляет из себя пакет информации, который сообщает Приложению (в частности) нечто, что требует (может требовать) обработки. Если Приложение распознает действия пользователя как такое event, то обработка (реакция - action) полностью зависит от Приложения, но при этом оказываются задействованными специальные action-процедуры.

Этот алгоритм реализуется за счет создания, инициализации и связывания нескольких таблиц.

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



Основной среди них является т.н. «таблица соответствия» (или «таблица трансляции» - translation table), которая осуществляет перевод события - [Event]Translation в действие (action). Каждый элемент такой таблицы представляет из себя пару:
«[последовательность] event[s]» / имя [имена] action.
Собственно эти translation table и являются динамическим ресурсом «кликабельных» widget'ов, классы которых являются подклассами базового класса Simple. Именно этот ресурс и предназначен для реализации специфических, обусловленных алгоритмом самого приложения (в том числе и), операций, путем устанавливания связи между widget'ами и процедурами приложения, которые эти операции выполняют.

Вернемся к translation table для command widget:

Event Name	Action
<EnterWindow>: <LeaveWindow>: <Btn1Down>: <Btn1Up>	highlight () reset () set () notify () unset ()

Помимо этой таблицы создается специальная «таблица действий» (action table), связывающая символические имена «действий» (action) с указателями на соответствующие action-процедуры. С помощью специальной процедуры Xt («добавить action») эта таблица регистрируется в системе.

Рассмотрим, что из себя представляют action, поддерживаемые widget'ами класса Command

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

highlight ()	Окрашивает внутреннюю границу подсветки в цвет (foreground или background), контрастный цвету внутренней области окна command widget. Это действие (action) стандартно связано с <EnterWindow> event (попадание pointer'а в подокно widget'а)
unhighlight ()	Окрашивает внутреннюю границу подсветки в цвет (foreground или background), соответствует цвету внутренней области окна command widget. Это действие (action) выполняется внутри reset () и не требует специального translation.
set ()	Вводит состояние set, в котором возможно action notify, и окрашивает внутреннюю область кнопки, включая и внутреннюю границу подсветки в цвет foreground. Цвет метки кнопки – background. Обычно это action происходит вследствие <Btn1Down> event.
unset ()	Отменяет состояние set и окрашивает внутреннюю область кнопки, включая и внутреннюю границу подсветки в цвет background. Цвет метки кнопки – foreground, соответствует цвету внутренней области окна command widget. Это действие (action) выполняется внутри reset () и не требует специального translation.
reset ()	Отменяет любые set или highlight, окрашивает внутреннюю область кнопки в цвет background. Цвет метки кнопки – foreground, соответствует цвету внутренней области окна command widget. Обычно это action происходит вследствие <Btn1Up> event (вместе с notify ()) для возвращения кнопки в ее исходное состояние после выполнения связанной callback-**процедуры.**
notify ()	Исполняет список callback , специфицированный ресурсом callback, если до этого было установлено состояние set. Это действие (action) фактически вызывает **процедуры Приложения**, которые должны быть выполнены в соответствии с его алгоритмом при таких входных events.

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

Несколько важных замечаний.

Action table, а также Динамический ресурс translation table для каждого widget'а в конкретном GUI конкретного Приложения создается (модифицируется) и регистрируется с помощью специальных процедур Xt:

Event Name	Action
<EnterWindow>:	highlight ()
<LeaveWindow>:	reset ()
<Btn1Down>:	set ()
<Btn1Up>	notify () unset ()

Часть events являются стандартными и распознаются большинством оконных менеджеров (WM) для перевода (translation) в соответствующие действия (action), как например

<EnterWindow>: highlight ()

Другая часть должна быть связана с соответствующими действиями (action) для вызова процедур самого Приложения нестандартно, т.е. – прикладным программистом , как например

<Btn1Down>: set ()
<Btn1Up> notify () unset ()

В связи с этим следует обратить внимание на то, что ресурс translation table размещается либо в ресурсном файле Приложения (подобласти rdb), либо непосредственно в коде этого Приложения. При этом перед использованием таблица должна быть переведена специальной процедурой во внутреннее представление Xt и зарегистрирована для соответствующего управляющего объекта - widget'а.
В последнем случае кастомизация поведения GUI конечным пользователем невозможна.

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

Несколько важных замечаний.

С учетом перечисленных выше шагов, вызов action-**процедуры**, когда в GUI в соответствующем widget'**е** происходит какое-то событие (события) – events, в производится следующим образом:

- Xt проверяет translation table для данного widget'а; если в ней такого события (event) нет – ничего не происходит;
- если event есть, то по нему определяют имена соответствующих действий - action;
- сканируется зарегистрированная таблица action table, и по именам action определяются адреса процедур; когда адрес становится известен, соответствующая функция вызывается;

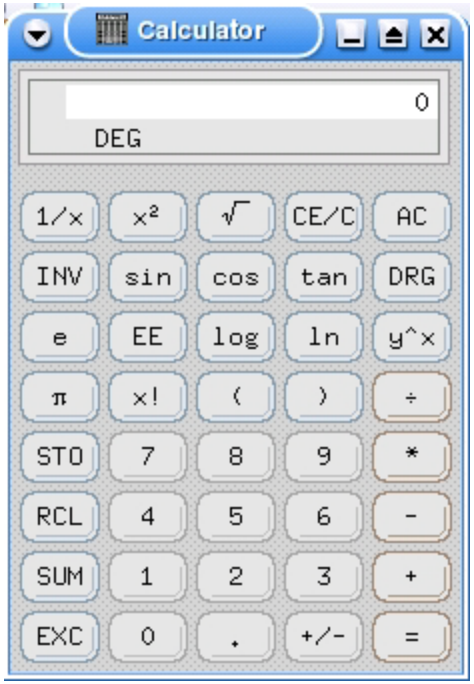
Здесь очень важно отметить, что в реальной практике применения иерархии widget'**ов** X11 использование именно **X-ового** множества управляющих элементов - widget'**ов** гарантирует для каждого класса наличие набора стандартных action-**процедур**, которые могут использоваться всеми инстанциями объектов данного класса.

Функции же, специально регистрируемые с помощью специальных процедур Xt, могут быть используемы произвольными widget' **ами** данного Приложения.

Каждый widget' имеет ресурс с именем `XtNtranslations`, значением которого является translation table, представленная во внутреннем формате Xt.

Если translation table записывается в ресурсном файле, то для ее представления в виде текстовой строки существует специальный формат, который мы рассмотрим на примере уже известного нам **Ресурсного файла для X-клиента Xcalc.**

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



Итак, в ресурсном файле (и в исходном коде Приложения – тоже), эта translation table представляется как строка, содержащая пары :

- **«последовательность events / имя (или имена) действий (action)», разделенные символом '\n'.**

В начале строки может стоять директива

- **"#replace", что означает, что данная translation table должна заменить уже имеющуюся у widget'а таблицу;**
- **"#augment", что означает, что данная translation table должна быть объединена с уже имеющейся у widget'а таблицей; при совпадении имен action в такой таблице приоритет имеет старая запись;**
- **"#override", что означает, что данная translation table должна быть объединена с уже имеющейся у widget'а таблицей; при совпадении имен action в такой таблице приоритет имеет новая запись;**

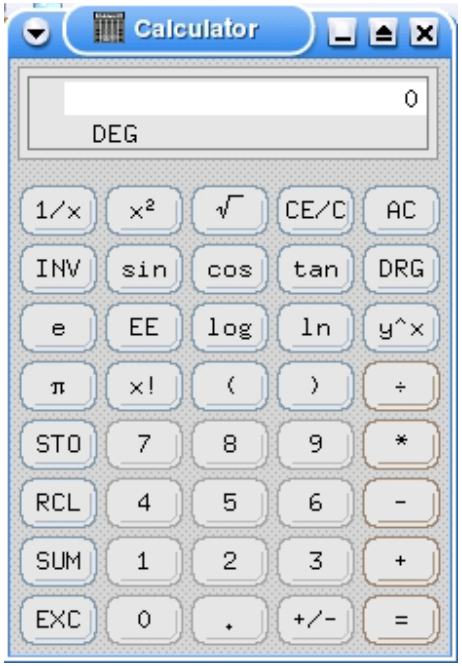
Тогда в ресурсном файле каждая translation (перевод события) будет иметь следующий вид:

`[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 (`KeyPress, KeyUp, ButtonPress, ButtonRelease, Btn1Down, Btn1Up, MotionNotify – MouseMoved` и т.д. – см. документацию на X11), а в поле «`модификатор`» могут использоваться стандартные модификаторы, поддерживаемые Xt: `Ctrl, Shift, Lock, Any, None` и т.д.

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Теперь ранее непонятные 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'**а** `XCalc*ti.button6`

Значение Ресурса `translations` **для** widget'**а** `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.**