Графические Системы. Часть II

Лекция № 5

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.
Widget'ы. Классы Widget'ов. Иерархия классов

Остановимся подробнее на иерархии классов widget'ов .

Итак, как уже неоднократно говорилось, Xt *(концептуально – любой ИПВУ)* предоставляет набор средств для создания объектов, которые используются программами для общения с пользователем, а в общем случае и с остальным внешним миром.

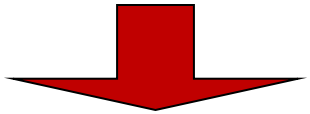
Каждый создаваемый программой widget есть представитель того или иного класса. Xt и пакеты, на нём основывающиеся, такие как OSF/Motif, Athena, Tcl/Tk, Open Look, *etc.*, имеют большое количество таких классов.

Создание новых widget'ов, не предусмотренных в стандартных библиотеках, требует и создания (определения) соответствующего класса, что, обычно, является трудоёмкой задачей.

Каждый класс имеет ряд фиксированных характеристик, являющихся общими для всех его экземпляров (например, список callback-процедур). Значения этих характеристик у самих объектов могут различаться.

Все классы Xt образуют иерархию.

Основной, реализованной в Xt и других ИПВУ концепцией, является концепция базового widget'a.



Иными словами, программист GUI вместо того, чтобы начинать процесс создания widget'a «на пустом месте», начинает с копирования другого «более базового» widget'a, а затем модифицирует его. Такой процесс получил название *subclassing'a*, при котором каждый генерируемый widget наследует характеристики своего *superclass'a.*

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

Если класс B ближе к вершине иерархии, чем класс D, то B называется базовым для D, а D называется производным классом (или подклассом) для B.

Например, класс **Core** является базовым по отношению к **Composite**, а **Composite** есть подкласс класса **Core**.

Подклассы наследуют характеристики всех своих базовых классов. Это означает, экземпляр класса имеет характеристики не только своего класса, но и атрибуты всех базовых классов.

Например, основным классом в Афинской иерархии является класс **Object**, все ресурсы которого непосредственно наследует класс **RectObj**. Класс **Shell** наследует ресурсы – характеристики классов **Core и Composite** и т.д.

Рассмотрим подробнее основные классы **Widget'ов**

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

Рассмотрим подробнее основные классы Widget'ов в X 11

Корнем афинской иерархии является класс с именем Object.

Это абстрактный класс (класс, не порождающий собственного объекта - widget'а), который используется в качестве корня дерева всех объектов.

Однако, этот класс задает некоторые характеристики, общие для всех widget'ов:

- **Возможность распознавать ресурсы;**
- **Возможность быть связанным с конкретным Приложением (Application) через механизм callback;**

(Например, если в GUI пользователь нажимает на кнопку Quit, то в Приложение передается «нечто», в результате чего в Приложении происходит другое «нечто», предусмотренное этим Приложением – в частности, выход из программы)

Вспомним, что это означает:

Callback - процедуры ("процедуры обратного вызова"). Для любого класса определена совокупность действий, на которые должны реагировать принадлежащие ему объекты. Например, для любого класса предусмотрена реакция на уничтожение widget'а. Когда действие производится, происходит вызов либо стандартной функции Xt, либо одной или нескольких процедур, предоставляемых программой. Такие функции и называются callback - процедурами или просто callback.

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

RectObject является подклассом класса **Object.**

Это тоже абстрактный класс (класс, не порождающий собственного объекта - **widget'a**), который используется для определения некоторых общих характеристик, необходимых для функционирования различных типов объектов (например, для объектов, не имеющих окна)

Самое главное предназначение класса **RectObject** - он добавляет ряд ресурсов, в соответствии с которыми декларируется, что окна **widget'ов** является прямоугольным:

- **width, height, borderWidth, x-** и **y-**позиция окна **widget'a**

Чтобы выяснить состав ресурсов каждого класса иерархии **X 11**, можно воспользоваться командой:

```
% listres <имя класса>
```

Кроме того, класс **RectObject** добавляет еще один ресурс для всех подклассов этого класса, т.е. – для всех **widget'ов**. Это ресурс **sensitive** (чувствительность), определяющий возможность существования **widget'a** в окружающей среде.

Например, при некоторых условиях **widget** в рамках **GUI** должен быть недоступным, в частности, в падающем меню какого-то **GUI** альтернатива «**Close File**» не может быть доступна пользователю, если он не открыл файл с помощью альтернативы «**Open File**».

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

Core является первым реальным классом, который может быть отображен на экране.

Object и RectObject не имеют окон, ассоциированных с ними, т.е. не могут быть созданы и размещены на экране.

Фактически **Core –** это корень дерева классов **widget'ов**, имеющих окна. Этот класс определяет характеристики, общие для всех объектов – т.е. все **widget'ы** «***SUBCLASSED***» из этого класса и наследуют все его ресурсы.

Список ресурсов класса **Core** приведен ниже (включает в себя ресурсы, унаследованные от классов **Object и RectObject** и добавленные в **Core**):

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

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

Некоторые из этих **Core** ресурсов устанавливают очевидные характеристики – ресурсы - **widget'a:**

Background (цвет), **borderColor, borderWidth** (в пикселях); **height** и **width** специфицируют размеры **widget'a** в пикселях; **x** и **y** задают **x,y**-координаты **widget'a** относительно родительского окна

Технически **foreground** не является ресурсом класса **Core.** Однако, поскольку этот ресурс определен для каждого класса **widget'ов**, для любых изображений и текстов в них, то он также может считаться базовым ресурсом в иерархии.

Выше были описаны не все ресурсы класса **Core**, а только те, которые могут быть установлены пользователем. Некоторые ресурсы, такие, например, как **callback**, устанавливаются исключительно программистом, и **X Toolkit** не поддерживает механизмов прописывания их в ресурсных файлах.

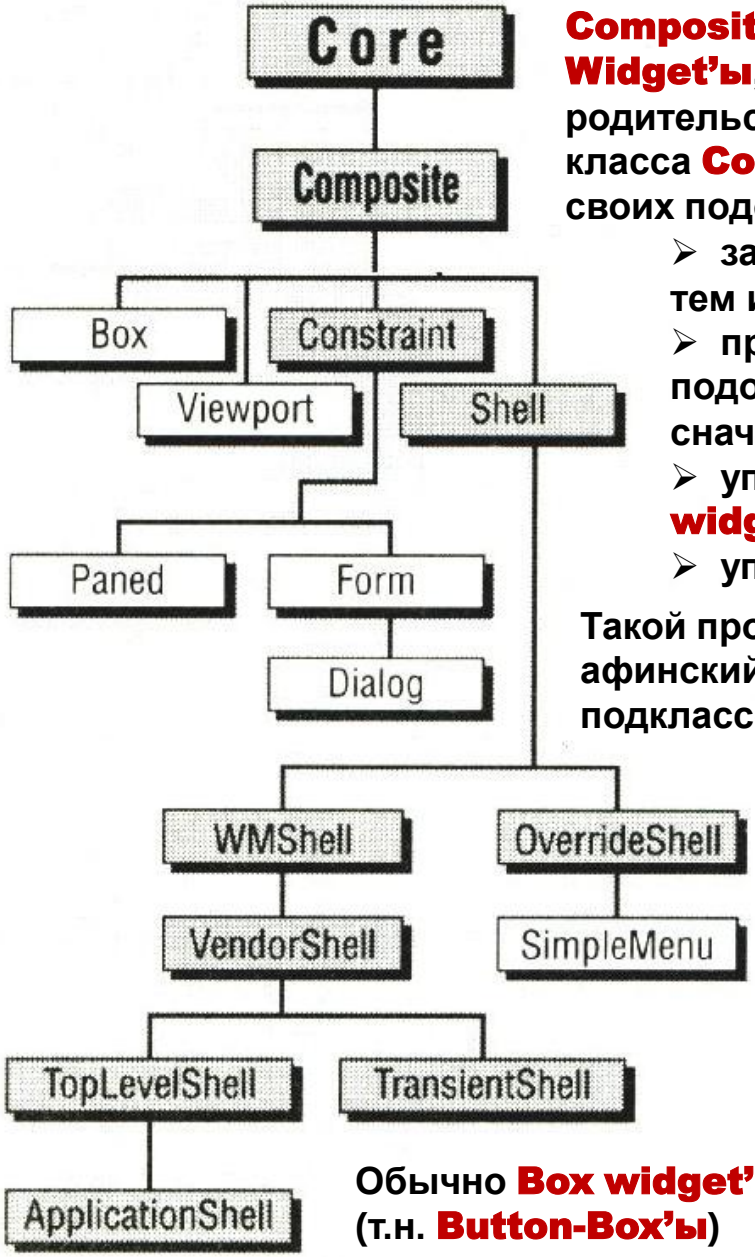
Далее в иерархии базовых классов **X Window** (общих для всех, базирующихся на **Xt**, наборов **widget'ов**, включая **Motif и Athena**, и обозначенных серым цветом) рассмотрим специальный класс **widget'ов**, чья работа заключается только в том, чтобы управлять размерами и/или позицией других **widget'ов**.

Такие **widget'ы** объединены в класс **Composite** (составные), и все геометрически-управляемые **widget'ы** являются потомками этого класса и его подклассов.

Правая ветвь иерархии (класс **Simple**) будет рассмотрена позже.

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Composite.

Widget'ы, относящиеся к данному классу, могут быть родительскими по отношению к другим объектам. Экземпляры класса **Composite** определяют следующие особенности поведения своих подобъектов:

- задаёт местоположение «дочерних» **widget'ов** согласно тем или иным ограничениям;
- при уничтожении освобождает память, используемую подобъектами (при уничтожении **widget'а** класса **Composite** сначала будут уничтожены все его «потомки»);
- управляет появлением на экране окон своих дочерних **widget'ов** ;
- управляет передачей фокуса ввода между объектами.

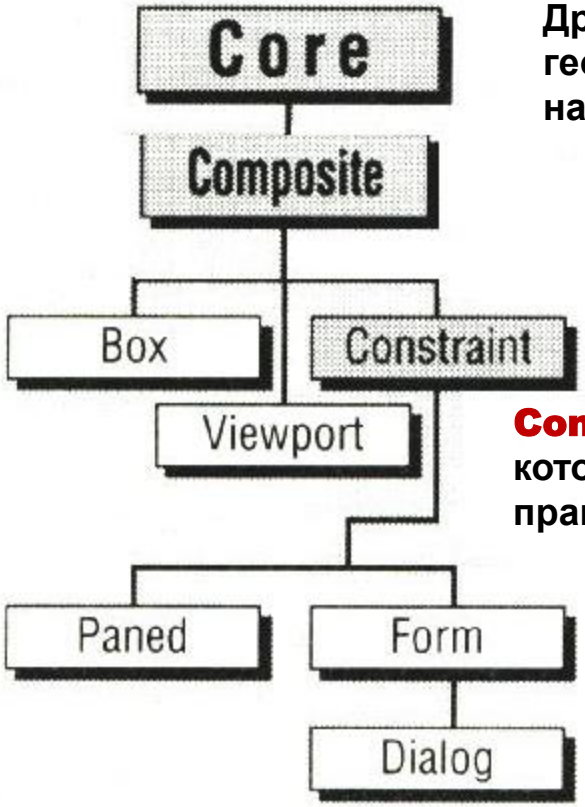
Такой простой класс геометрически управляемых **widget'ов** как афинский **Box widget-class** является непосредственным подклассом класса **Core.**

Box обеспечивает геометрическое управление произвольным числом **widget'ов**, сгруппированных в **Box** заданных размеров. Внутри **Box'а** можно перемещать его потомков – подокна **widget'ов**, но не изменять их размеры. Если размеры родительского окна **Box widget'а** изменяются, подокна **widget'ов** –потомков перестраиваются друг относительно друга. **Box widget** всегда старается максимально плотно упаковать своих потомков.

Обычно **Box widget'ы** используются для упаковки **widget'ов** типа **Command** (т.н. **Button-Box'ы**)

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Другой, позволяющий осуществлять более сложное управление геометрией потомков чем **Box**, подкласс класса **Composite** называется **Constrain**:

- Это класс представляет собой дальнейшее расширение базового класса **Composite**. Его экземпляры имеют дополнительные возможности для управления размером и местоположением своих потомков. Например, подобъекты могут размещаться в специальном порядке: в ряд, в столбец и т.д.

Constrain - widget определяет специальный класс ресурсов, которые называются **Constrain resources,** значениями которых как правило являются сами **widget'ы** данного класса.

Очень характерный пример того, что такое **Constrain resources** дает афинский класс **widget'ов Form.**

Widget'ы этого класса могут содержать произвольное число окон **widget'ов-**потомков любого класса. **Form widget** также управляет геометрией своих потомков, включая индивидуальное управление позицией каждого **widget'a -** потомка.

Если размеры родительского окна **Form widget'a** изменяются, то пресчитываются и размеры и позиция всех его потомков, при этом взаимное расположение подокон **widget'ов -** потомков **Form widget** сохраняется.

Рассмотрим правило формирования **Constrain resources** для **Form widget'a** на конкретном примере **X-клиента xcalc**:

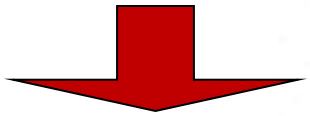
Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов

Вспомним: Во время работы программа X-Клиент создает сами объекты- **widget'ы** (экземпляры **widget**-классов). Они образуют совокупности, каждая из которых также представляет собой некоторую иерархию. Каждая такая иерархия называется деревом объектов – **widget tree.**

Widget- and window-tree X-Клиента xcalc:

Constrain resources Form-widget'a X-Клиента **xcalc** формируются следующим образом:



```
XCalc xcalc
    Form  ti  or  rpn        (the name depends on the mode)
        Form  bevel
            Form  screen
                Label   M       (the memory indicator on the screen)
                Toggle  LCD     (where the data is displayed)
                Label   INV     (the inverted indicator on the display)
                Label   DEG     (the degrees indicator on the display)
                Label   RAD     (the radians indicator on the display)
                Label   GRAD    (the gradians indicator on the display)
                Label   P       (the Parenthesis indicator on the display)
    Command   button1   (the actual calculator buttons)
    Command   button2   (buttons are numbered from right to left)
    Command   button3   (See the app-defaults file for associations
                         and so on...)
    Command   button38  (between widget names and default labels)
    Command   button39
    Command   button40  (Only 39 buttons in HP mode)
```

Form-widget

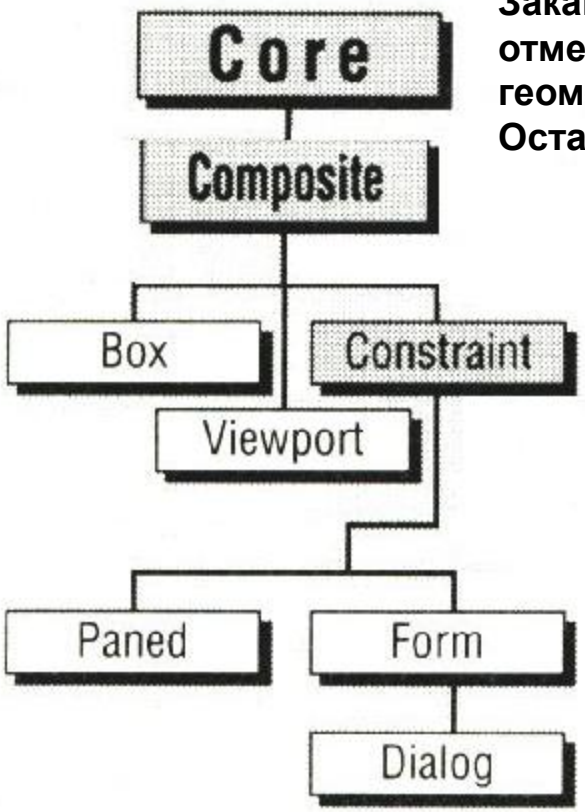
Command-widget - потомок Form-widget

Constrain resource Form-widget'a, значением которого является другой widget – потомок Form-widget'a

```
XCalc*ti.button12.fromHoriz:    button11
XCalc*ti.button12.fromVert:     button7
```

Основы программирования в системе X Window System.

Афинская Иерархия классов Widget'ов



Заканчивая рассмотрение группы подклассов класса **Composite**, отметим, что рассмотренные выше классы могут быть отнесены к т.н. геометрическим менеджерам.

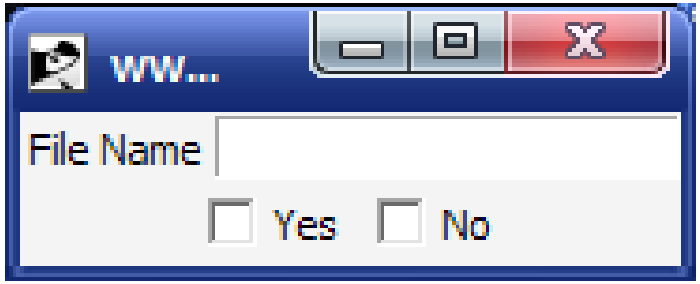
Остановимся коротко на оставшихся классах:

Viewport.

Widget этого класса представляет собой frame окна, содержащего один или два **scroillbar'a**, а также внутреннего окна, содержащего **widget'a -** потомка. Размер этого окна определяется объемом просматриваемых данных, а также размерами, заданными при создании **widget'a** класса **Viewport.** Если данные такровы. Что не требуют прокрутки в одном или обоих направлениях, то соответствующие **scroillbar'ы** автоматически исключаются.

Dialog.

Widget этого класса сообщает пользователю о том, что требуется дополнительный ввод. Типичный **Dialog-widget** выглядит следующим образом:



Dialog-widget фактически не является **widget'ом,** а представляет из себя интерфейс **widget'a.** Это в чистом виде составной (**Composite**) **widget,** включающий в свой состав **widget'ы label, command, text.**

Paned.

Widget этого класса управляет потомками, имеющими горизонтальную и вертикальную раскладку. Изменение размеров этих панелей может быть реализовано с помощью специального «захвата» grip в углах родительского **widget'a,** размеры которого при этом не меняются.