Glade 2.x — визуальная

ЧАСТЬ 4 Красивый интерфейс можно запрограммировать, вдалбливая бесконечные строчки кода, а можно – легким движением руки, в которой зажата мышь. **Андрей Боровский** рассматривает второй способ.

Каждый уважающий себя набор визуальных элементов располагает средством для того, чтобы бросить пару-тройку компонентов на заготовку окна приложения, настроить их свойства и получить остов будущей программы. У *Qt* есть *Qt Designer*, у *wxWidgets* есть *DialogBlocks*, а у *GTK+* есть *Glade*. Если ваш опыт программирования ограничивается такими средами, как Borland C++ Builder или Microsoft Visual C#, приступая к программированию с помощью *Glade*, вы должны понять одну важную вещь: *Glade* – это не IDE. В нем нет ни интегрированного компилятора, ни интерактивного отладчика, ни редактора кода с автоматическим завершением строки. *Glade* 2.x генерирует минимальный код, который вам придется дописывать в вашем любимом редакторе, а *Glade* 3.x вообще стремится возложить генерацию кода на другие инструменты. Несмотря на то, что в ближайшее время старый *Glade* будет вытеснен третьей версией, знакомство с этим инструментом разработки позволит нам взглянуть на структуру сложных программ *GTK+* глазами самих авторов набора компонентов. Поэтому данная статья посвящена *Glade 2.x*, а в следующей статье мы узнаем, чем *Glade 3.x* отличается от своего предшественника. Мы воспользуемся *Glade* для создания простой программы ImageViewer. Ее исходные тексты можно найти на диске, но я советую вам пройти вместе со мной все шаги по созданию данного приложения.

Каждый раз, когда мы создаем новый проект *Glade*, перед нами появляется диалоговое окно, в котором нам предлагается выбрать тип проекта – *GTK+* или GNOME (**Рис. 1**). Нам нужен *GTK+*.

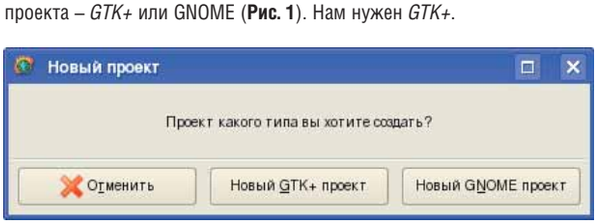


Рис. 1 Окно выбора типа проекта.

В среде *Glade* в процессе работы может быть открыто несколько окон, постоянными из которых являются три: главное окно, окно палитры и окно редактора свойств объектов. Главное окно (**Рис. 2**) содержит меню и панель инструментов для управления проектом. В рабочей части окна расположен список визуальных элементов верхнего уровня.

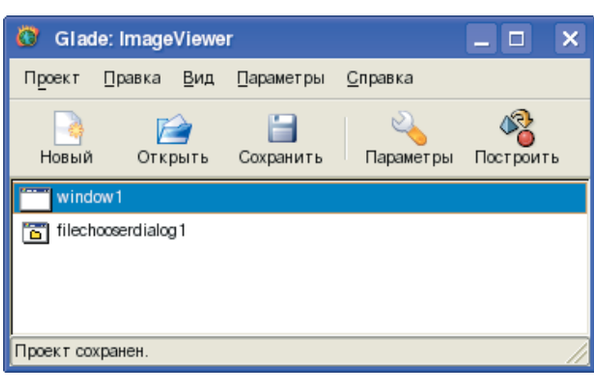


Рис. 2 Главное окно Glade.

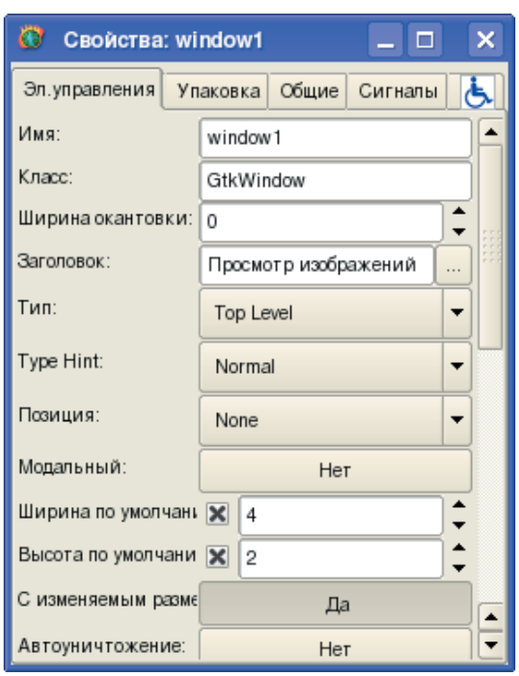
Окно «Палитра», которое, в том или ином виде, присутствует в каждой среде визуальной разработки, предоставляет нам доступ к пиктограммам визуальных элементов. Окно Палитра *Glade 2.** (**Рис. 3**) содержит несколько вкладок. Список вкладок зависит от типа проекта (*GTK+* или GNOME). При работе с нашим проектом палитра будет содержать три вкладки: GTK+ Основные, GTK+ Дополнительные и Устаревшие.

Окно редактора свойств (**Рис. 4**) предназначено, естественно, для редактирования свойств визуальных элементов. Это окно также содержит несколько вкладок. Вкладка Эл. управления позволяет настраивать специфические свойства визуального элемента. Вкладка Упаковка позволяет настраивать параметры упаковки тех визуальных элементов, которые вложены в какой-либо контейнер. С помощью вкладки Общие можно настраивать общие свойства визуальных элементов, а вкладка Сигналы предназначена для связывания сигналов и их обработчиков. Наконец, безымянная вкладка, пиктограмма которой изображает инвалидное кресло, позволяет настраивать параметры доступности

Месяц назад Мы разбирались с сигналами и событиями GTK+.

КЛАССИКА

(accessibility) визуальных элементов GNOME. Эту вкладку мы пока рассматривать не будем.



› Рис. 4 Окно редактора свойств.

Щелкните по элементу Окно на вкладке GTK+ Основные палитры компонентов. В результате в проект будет добавлен элемент верхнего уровня Окно (вы можете видеть его в списке элементов верхнего уровня в главном окне проекта), а перед нами откроется пустое окно со стандартным заголовком. Это окно станет главным окном создаваемого нами приложения. Программа *GTK+* может содержать несколько элементов верхнего уровня, и для каждого из них можно открыть независимое окно.

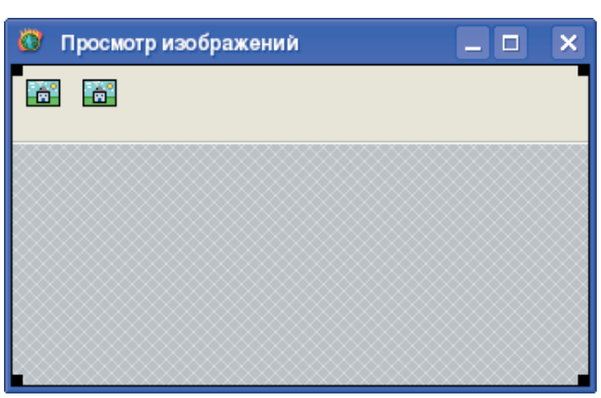
Давайте настроим параметры главного окна программы. Для того, чтобы открыть объект интерфейса верхнего уровня в редакторе свойств, нужно выделить этот объект в главном окне *Glade*. Перейдя в редакторе свойств на вкладку Эл. управления, мы можем указать текст заголовка окна (свойство Заголовок). Установите также флажки Ширина по умолчанию и Высота по умолчанию. В соответствующих наборных счетчиках укажите значения **200** и **150**. Таким образом вы зададите начальные размеры окна.

Вы, конечно, помните, что любое приложение *GTK+* с более-менее сложным интерфейсом должно использовать контейнеры для управления визуальными элементами. Разработка интерфейса в *Glade* также начинается с создания контейнера. Выберите на вкладке палитры GTK+ Основные элемент Вертикальный бокс. Этому элементу соответствует вертикальный контейнер GtkVBox. Перенесите вертикальный контейнер мышью в окно приложения. Во многих редакторах форм этот процесс называется перетаскиванием, поскольку компонент перетаскивается мышью с палитры компонентов в окно формы, но в *Glade* все происходит по-другому. Для того чтобы перенести компонент в форму, нужно сначала щелкнуть мышью по этому компоненту в палитре, а затем – по той области на форме, в которой этот компонент должен быть размещен (вот почему мы используем термин «перенос» а не «перетаскивание»). При переносе в форму вертикального контейнера открывается диалоговое окно, в котором *Glade* просит нас указать число строк в нем. Создайте контейнер с двумя строками. В результате главное окно будет разделено на две горизонтальные области, расположенные одна под другой (для вертикального расположения дочерних элементов). В верхнюю область перенесите с палитры объект Панель инструментов. Панель инструментов также представляет собой контейнер, так что при переносе панели в окно приложения мы снова должны указать количество дочерних элементов (на этот раз – кнопок панели). Панель инструментов нашего приложения должна содержать две кнопки. Вы можете заметить, что после переноса панели инструментов в окно формы, размеры верхней области вертикального контейнера подстраиваются под размеры панели, а на самой панели появляются две незаполненные области для размещения кнопок (**Рис. 5**). В эти области мы должны перенести два элемента Toolbar Button – кнопки панели инструментов, которым соответствует объект GtkToolButton. В окне панели инструментов появляются заготовки кнопок.

› (Рис. 3) Главное окно палитры.

Что в имени тебе моем

Поскольку на этом этапе мы уже работаем с несколькими объектами, нам следует познакомиться с понятием имени объекта. Каждому объекту *GTK+* может быть присвоено имя. Это имя хранится в одном из полей объекта, и его не следует путать с именем переменной, которая содержит указатель на объект (имя объекта, хранящееся в самом объекте, по праву можно назвать «именем собственным»). До тех пор, пока наши программы состояли из одного файла и все переменные, содержащие объекты, находились в одной области видимости, имена объектов не имели для нас значения. Однако в »



› Рис. 5 Главное окно *Glade*.

сложных проектах, состоящих из нескольких файлов, имена объектов *GTK+* играют важную роль, поскольку именно они служат для идентификации объектов. В редакторе свойств объектов на вкладке Эл. управления присутствует свойство Имя, которое позволяет вам получить доступ к имени объекта на этапе визуального программирования. Две кнопки, которые вы добавили на панель инструментов, имеют имена toolbutton1 и toolbutton2. Я предполагаю, что вы добавляли кнопки в порядке слева направо, то есть первой слева является кнопка toolbutton1.

Только что созданные кнопки выглядят совсем не так, как нам хотелось бы. Для того, чтобы придать кнопкам требуемый вид, мы должны отредактировать их свойства. Выберите в окне формы кнопку toolbutton1 и перейдите в редактор свойств, на вкладку Эл. управления. Свойству Метка присвойте значение Открыть. Затем щелкните по значку раскрывающегося списка в поле свойства Иконка, в открывшемся списке выберите пиктограмму Открыть. Нам необходимо создать заготовку обработчика для сигнала кнопки clicked. Перейдите на вкладку Сигналы редактора свойств и щелкните кнопку с многоточием справа от строки ввода Сигнал. Перед вами раскрывается список сигналов, в котором следует выбрать сигнал clicked. В строке Обработчик появляется имя обработчика сигнала (по умолчанию – on_toolbutton1_clicked). Щелкните кнопку Добавить. В результате новая функция будет добавлена в список обработчиков сигналов кнопки toolbutton1.

Перейдем теперь к редактированию свойств кнопки toolbutton2. Свойству Метка второй кнопки присвоим значение О программе. В списке значений свойства Иконка выберем значение Справка, затем создадим заготовку обработчика сигнала clicked (назовем его on_toolbutton2_clicked), так же как и в предыдущем случае.

В нижнюю область вертикального контейнера следует перенести элемент палитры Изображение (объект GtkImage, имя объекта по умолчанию – image1). Выделите объект image1 в окне формы. Перейдите на вкладку Упаковка редактора свойств и присвойте значение Да свойству Заполнение. Далее, выделите в окне формы палитру инструментов (объект toolbar1 типа GtkToolbar). Перейдите на вкладку Упаковка редактора свойств и также присвойте значение Да свойству Заполнение (**Рис. 6**).

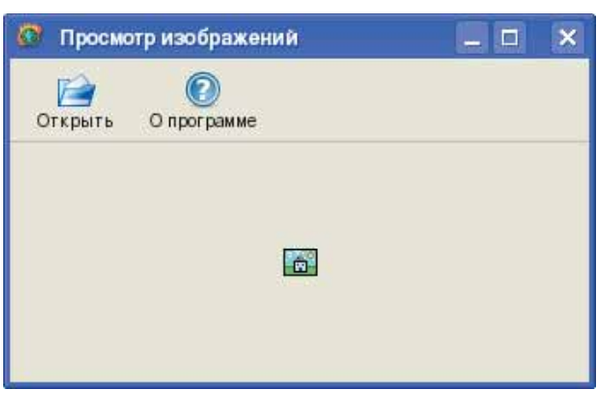


Рис. 6 Форма приложения с кнопками и объектом *GtkImage*.

Заголовок

Мы должны еще добавить в проект окно с информацией о программе. Это просто, поскольку необходимое окно входит в набор визуальных компонентов *GTK+* (**Рис. 7**). Вы найдете его на вкладке GTK+ Дополнительные палитры компонентов (элемент About Dialog, объект GtkAboutDialog). Окно с информацией о программе является элементом верхнего уровня, поэтому после щелчка мышью в палитре компонентов объект aboutdialog1 появится в списке элементов верхнего уровня. Перейдя на вкладку Эл. управления редактора свойств, вы сможете настроить основные свойства этого диалогового окна (название программы, имя автора, информацию о правах и т.д.). Никаких обработчиков сигналов для этого окна создавать не нужно, поскольку все необходимые обработчики создаются по умолчанию.

Рис. 7 Окно с информацией о программе.

Сохраним наш проект. Как уже отмечалось, *Glade* 2.* позволяет генерировать исходный код приложения. Для этого нужно щелкнуть на кнопку Построить. Теперь мы можем скомпилировать нашу программу (точнее, ту ее часть, которую мы создали). Перейдем в директорию проекта и выполним файл **autogen.sh**. В результате будут созданы сценарий configure и make-файл. Мы можем собрать программу с помощью команды **make**. Двоичный файл программы будет создан в директории **src**. Если вы запустите программу на этом этапе (я рекомендую первый раз запустить программу с консоли), то заметите две неприятные вещи. Во-первых, окно с информацией о программе само собой появляется вместе с главным окном программы. Во вторых, после закрытия главного окна программы ее работа не завершается (это заметно, если запустить программу из окна консоли). Для того, чтобы исправить ситуацию с окном aboutdialog1, нам нужно отредактировать файл **src/main.c**. Откройте этот файл в текстовом редакторе и удалите из него строки

```
aboutdialog1 = create_aboutdialog1 ();
gtk_widget_show (aboutdialog1);
```

Как вы, конечно, поняли, файл **main.c** содержит главную функцию программы. По умолчанию этот файл содержит код для создания и отображения на экране всех визуальных элементов верхнего уровня. Помимо файла **main.c** среда *Glade* создала еще несколько файлов исходных текстов в директории **src**. Файлы **interface.c** и **interface.h** содержат код, отвечающий за работу элементов управления, которые мы создали в режиме визуального программирования. Файлы **support.c/support.h** включают некоторые вспомогательные функции. Обработчики сигналов, которые мы создали в редакторе свойств *Glade*, объявлены в файле **callbacks.h**, а определены – в файле **callbacks.c** (вы можете изменить имена файлов исходных текстов, заданные по умолчанию, в окне настроек *Glade*). Только файлы **main.c** и **callbacks.c** предназначены для ручного редактирования. Текст остальных файлов перезаписывается *Glade* в процессе генерации кода. Теперь вспомним первые уроки программирования *GTK+*. Для того, чтобы программа завершалась вместе с закрытием главного окна, необходимо определить обработчик сигнала destroy объекта окна. В главном окне *Glade* выделите главное окно приложения и перейдите на вкладку «Сигналы» редактора свойств. Выберите в списке сигналов сигнал destroy (его не следует путать с доступным там же событием “destroy event”) и создайте заготовку обработчика для этого сигнала точно так же, как вы создавали заготовки обработчиков сигналов clicked. Сгенерируйте заново исходный код, нажав кнопку «Построить». Откройте в текстовом редакторе файл **callbacks.c**. Найдите заготовку обработчика on_window1_destroy() и добавьте в него вызов функции gtk_main_quit():

```
void on_window1_destroy (GtkObject * object, gpointer user_data)
{
  gtk_main_quit();
}
```

Мы используем тот же метод завершения программы, что и во всех предыдущих примерах. Сохраните файл **callbacks.c**. Теперь можете перекомпилировать и запустить программу снова. Окно описания программы уже не появляется, а закрытие главного окна программы приводит к ее завершению. После внесения изменений в свойства визуальных объектов вы должны каждый раз заново генерировать исходный код (и, конечно, сохранять все изменения, сделанные в исходных текстах вручную). Что произойдет, если вы сначала отредактируете код одного обработчика сигнала, а затем добавите новый обработчик в режиме визуального редактирования? Ничего страшного не случится. *Glade* добавит новый код в файл **callbacks.c**, и не тронет изменения, сделанные вами (этим свойством обладают только файлы **callbacks.c** и **main.c**).

Завершающая часть работы над нашей программой связана с добавлением кода в обработчики событий clicked двух кнопок. Сами функции-обработчики on_toolbutton1_clicked() и on_toolbutton2_clicked() уже определены в файле **callback.c**. Первая функция должна вызывать диалоговое окно выбора файла и загружать выбранный пользователем файл изображения в компонент image1:

```
void
on_toolbutton1_clicked (GtkToolButton * toolbutton, gpointer user_data)
{
  GtkWidget * file_chooser_dialog;
  file_chooser_dialog = gtk_file_chooser_dialog_new("Открыть", NULL,
    GTK_FILE_CHOOSER_ACTION_OPEN,
    GTK_STOCK_CANCEL, GTK_RESPONSE_CANCEL,
    GTK_STOCK_OPEN, GTK_RESPONSE_ACCEPT, NULL);
    if (gtk_dialog_run(GTK_DIALOG (file_chooser_dialog)) == GTK_
RESPONSE_ACCEPT)
    {
      GtkWidget * toplevel;
      GtkWidget * image;
      char *filename;
      filename = gtk_file_chooser_get_filename(GTK_FILE_
CHOOSER(file_chooser_dialog));
      toplevel = gtk_widget_get_toplevel(GTK_WIDGET(toolbutton));
      image = lookup_widget(GTK_WIDGET(toplevel), "image1");
      gtk_image_set_from_file(GTK_IMAGE(image), filename);
      gtk_window_set_title(GTK_WINDOW(toplevel), filename);
      g_free (filename);
    }
    gtk_widget_destroy (file_chooser_dialog);
}
```

Все необходимые заголовочные файлы включены в файл **callbacks.c**, так что мы можем о них не беспокоиться. Диалоговое окно выбора файла реализуется объектом GtkFileChooserDialog. Объект создается функцией gtk_file_chooser_dialog_new(). В первом параметре этой функции передается заголовок создаваемого окна. Второй параметр может содержать ссылку на родительский визуальный элемент. Третий параметр служит для передачи константы, которая указывает, для чего предназначено диалоговое окно – для открытия или для сохранения файла. Далее следует список параметров, состоящий из имен кнопок и констант, которые должны идентифицировать эти кнопки. Список может содержать описание одной или нескольких кнопок и должен заканчиваться значением **NULL**. Функция gtk_dialog_run() делает диалоговое окно видимым и возвращает управление после того, как пользователь нажмет одну из кнопок. При этом функция возвращает значение, сопоставленное нажатой кнопке при вызове функции gtk_file_chooser_dialog_new(). Если пользователь нажал кнопку «Открыть», имя которой задано в константе GTK_STOCK_OPEN, функция gtk_dialog_run() возвращает значение GTK_RESPONSE_ACCEPT. В случае выбора пользователем кнопки «Открыть» мы должны узнать имя выбранного файла, получить указатель на объект image1 и загрузить файл в этот объект. Если вас удивляет, что я включил получение указателя на объект image1 в список наших задач, то вспомните, что у вас нет доступа к переменной, которая хранит этот указатель. Для того, чтобы получить указатель на объект image1, мы должны воспользоваться функцией поиска объектов в иерархии визуальных компонентов нашего приложения. Мы начинаем поиск объектов с того, что находим визуальный элемент верхнего уровня. Указатель на визуальный элемент верхнего уровня мы получаем с помощью функции gtk_widget_get_toplevel(). В поисках нужного объекта функция gtk_widget_get_toplevel() перебирает иерархию визуальных элементов, начиная с того элемента, чей указатель передан ей в единственном параметре. В качестве стартового значения для поиска мы передаем функции gtk_widget_get_toplevel() указатель на объект-кнопку, для которой вызван обработчик on_toolbutton1_clicked() (адрес этого объекта передается функции обработчику в параметре toolbutton, и это единственный адрес объекта нашего приложения, которым мы располагаем в данный момент). Адрес объекта, представляющего элемент верхнего уровня (в нашем случае это объект window1 – главное окно приложения) мы сохраняем в переменной toplevel. Далее мы получаем указатель на объект image1 с помощью функции lookup_widget() (эта функция сгенерирована *Glade* для нашего проекта и объявлена в файле **support.h**). Функция lookup_widget() позволяет найти визуальный объект по его имени. Документация гласит, что первым аргументом lookup_widget() должен быть указатель на визуальный элемент верхнего уровня, либо указатель на любой другой элемент иерархии приложения. В качестве отправной точки поиска мы выбираем элемент верхнего уровня, во-первых, потому, что так красивее, и во-вторых, потому, что у нас все равно уже есть указатель на объект главного окна. Вторым аргументом lookup_widget() должно быть имя искомого визуального элемента. Указатель на объект image1 сохраняется нами в переменной image. Мы получаем имя выбранного пользователем файла с помощью функции gtk_file_chooser_get_filename() и загружаем хранящееся в файле изображение в объект image1 с помощью функции gtk_image_set_from_file(). Остальной код обработчика должен быть вам понятен. Осталось заполнить обработчик сигнала clicked кнопки toolbutton2:

› Рис. 8 Работающая программа *ImageView*.

```
void
on_toolbutton2_clicked(GtkToolButton   *toolbutton, gpointer user_data)
{
  GtkWidget * about_dialog;
  about_dialog = create_aboutdialog1();
  gtk_widget_show(about_dialog);
}
```

Получить указатель на объект aboutdialog1 очень просто. В файле **interface.h** определена функция create_aboutdialog1(), которая создает объект окна описания программы и возвращает указатель на него. Нам остается только вывести окно на экран с помощью gtk_widget_show(). Это те самые два вызова функций, которые нам пришлось удалить из файла **main.c**, для того, чтобы окно описания программы не появлялось тогда, когда не надо. Теперь мы можем проверить, как работает наша программа (**Рис. 8**). LXF

Через месяц Мы узнаем, что нового привносит Glade 3.x в процесс визуальной разработки.