Практическое занятие 6. Отладка программы с развитым интерфейсом

Рассматриваются способы организации главного меню, всплывающего меню и панели инструментов, рассматривается компонент TImageList и возможность вывести изображение на пункты меню и кнопки панели.

Цель занятия

Практически отработать методику отладки приложения, содержащего главное и всплывающее меню, панели инструментов, изучение компонента `TImageList`.

Задание:

1. Отладить оконное приложение согласно прилагаемой ниже методике, применяя отличные от предлагаемых стили и дизайн. Предусмотреть обработку ввода ошибочной информации. Приветствуется усовершенствование приложения.
2. В качестве отчёта представить архив с проектом (без *.exe файла) и файлом отчёта, в котором описать отличия разработанного приложения от описанного в данном методическом пособии.

Главное меню

Интерфейс программы может содержать два типа меню: главное и всплывающее. Главное меню представляет собой строку в верхней части экрана. Первой командой меню, как правило, является "**Файл**", последней - "**Справка**". Реализуется главное меню компонентом `TMainMenu`, который находится на вкладке **Standard Палитры компонентов**:

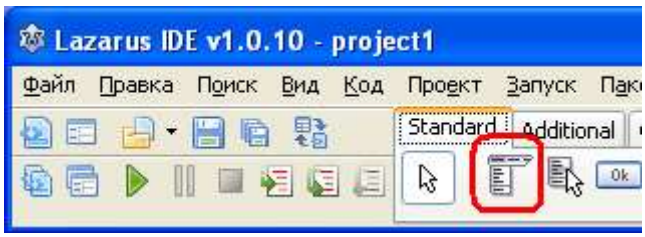


Рис. 6.1. Компонент TMainMenu

Но давайте знакомиться с этим компонентом сразу на практике. Итак, откройте **Lazarus** с новым проектом. Сразу же переименуйте главную форму в **fMain**, проект сохраните в папку **16-01** под именем **ImageViewer**, а модулю главной формы дайте имя `Main`. В `Caption` формы напишите `Просмотр изображений`. А саму форму сделайте побольше, хотя бы 450*700 пикселей.

Теперь установите на форму компонент `TMainMenu` - компонент невизуальный, поэтому его можно установить в любой место, например, прямо посередине формы. Нам не придется обращаться к компоненту по имени, так что имя можно не изменять. Чтобы создать меню, нужно дважды щелкнуть по компоненту, и откроется **Редактор меню**. По умолчанию, в **Инспекторе объектов** окажется выделенным первый (и единственный) пункт меню **MenuItem1**:

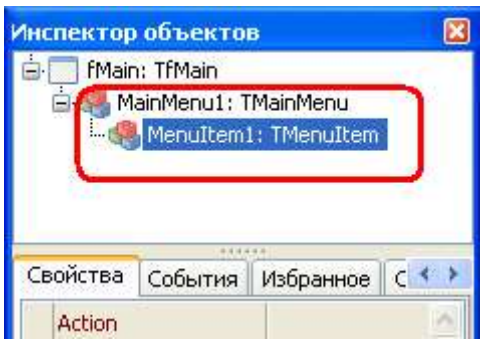


Рис. 6.2. Выделен пункт меню

Вот тут нам придется изменить некоторые настройки, так как имя `MenuItem1` нам ни о чем не говорит. Свойство `Name` переименуйте в `FileMenu`, в свойстве `Caption` напишите `Файл`. Сразу же вы увидите, как изменилось название на кнопке в **Редакторе меню**, и слово "Файл" появилось в верхней части окна формы. Щелкать по нему не нужно, иначе будет создан обработчик нажатия на пункт меню, а нам это не нужно - для пункта "Файл" обработчик не предусмотрен.

Далее, в **Редакторе меню** щелкните по кнопке "**Файл**" **правой** кнопкой мыши, и в открывшемся всплывающем меню выберите команду "**Вставить новый пункт (после)**". В **Редакторе меню** появилась вторая кнопочка с надписью "`New Item2`". Её придется переименовать, как и первую - убедитесь, что именно она выделена в **Инспекторе объектов**, переименуйте `Name` в `WindowMenu`, а в `Caption` напишите `Окно`. Теперь наше меню содержит два пункта: "**Файл**" и "**Окно**":

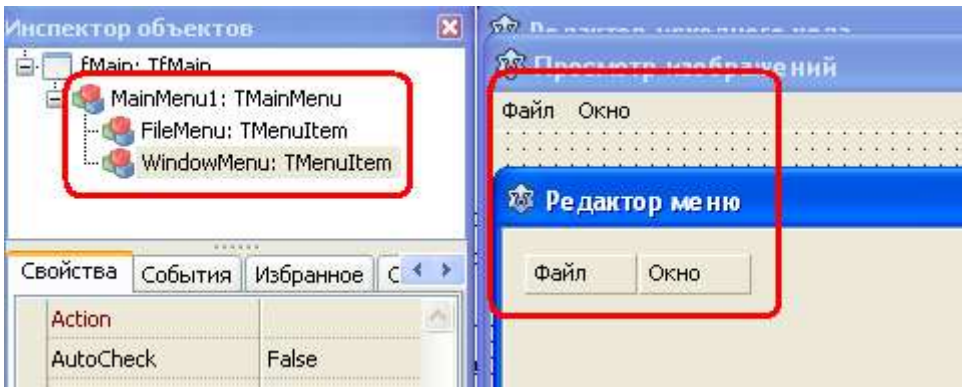


Рис. 6.3. Установка пунктов меню

Научимся создавать подпункты меню. Щелкните правой кнопкой мыши по кнопке "**Файл**" в **Редакторе меню** и выберите команду "**Создать подменю**". Сразу же на кнопке "**Файл**" появилась стрелка вниз, указывающая, что пункт меню содержит подпункты. И сам подпункт - кнопочку с надписью "**New Item3**", расположенную ниже "**Файл**". Убедимся, что в **Инспекторе объектов** выделен этот пункт, `Name` изменим на `FileOpen`, `Caption` - на `Открыть`.

Тут идея такова: главные пункты меню называть, как **FileMenu**, **WindowMenu**, **OptionsMenu**, **HelpMenu** и т.п. Последнее - **Menu** будет указывать на то, что это главный пункт. Подменю именовать иначе - вначале идет имя главного пункта, затем команда. Например, для главного пункта **FileMenu** мы могли бы создать такие команды, как `FileOpen`, `FileSaveAs`, `FileClose` и т.п. Это лишь рекомендация, правила вы можете придумать и свои, главное, чтобы вы не запутались в названиях главного пункта и его подпунктов.

Щелкните правой кнопкой мыши по кнопке "**Открыть**", выберите команду "**Вставить новый пункт (после)**". Полученный пункт переименуйте в **FileSaveAs**, а в `Caption` напишите `Сохранить как`.

Ниже аналогичным образом вставьте новый подпункт. У этого подпункта имя мы менять не будем, а в свойстве `Caption` мы впишем всего один символ "`-`". Таким образом в меню вставляются разделители - горизонтальные черточки между командами.

Далее вставим еще один подпункт. Назовите его **FileExit**, а в `Caption` напишите `Выход`. В результате у вас должно получиться нечто подобное:

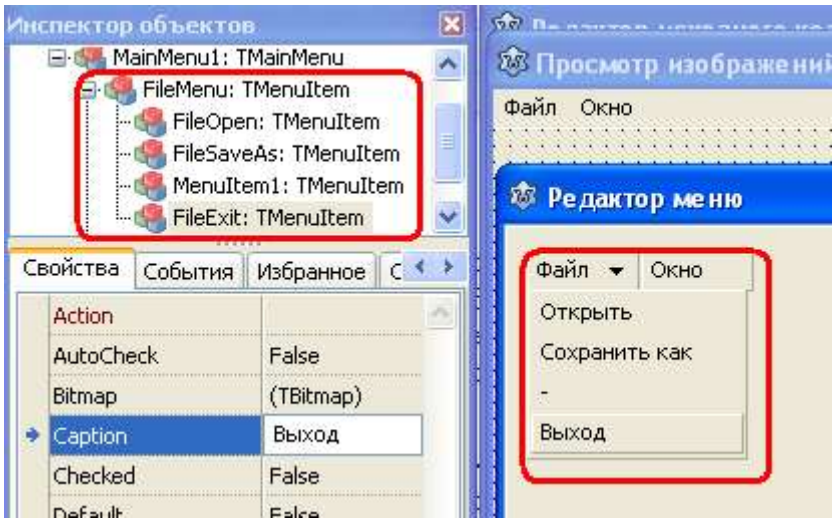


Рис. 6.4. Меню "Файл"

У нас ещё остался необработанным пункт "**Окно**". Выделите его и создайте следующие подпункты с парой свойств (`Name - Caption`): `WindowNormal` - Нормальное, `WindowMinimize` - Свернуть, `WindowMaximize` - Развернуть. Как только вы это сделаете, можете закрывать **Редактор меню** - он нам больше не нужен.

В центральную часть окна поместите компонент `TImage`, в который мы и будем загружать выбранные пользователем изображения. Настройте компонент следующим образом:

- `Align = alClient`
- `Center = True`
- `Proportional = True`
- `Stretch = True`

Кроме того, на форме нам понадобятся два диалога: `TOpenPictureDialog`, который мы переименуем в `OPD`, и `TSavePictureDialog`, который назовем `SPD`. Как и `TMainMenu`, диалоги будут невидимы пользователю, поэтому их можно установить на любое место формы.

Выделите компонент `TOpenDialog`, разберемся с его свойствами. Как видите, их не очень много, да и нужны нам будут не все.

`DefaultExt`	- расширение имени файла по умолчанию. В зависимости от того, с каким типом файлов нам придется работать, такое у них будет и расширение. Если мы работаем с текстом, то лучше именам файлов давать расширение txt. Это

	необязательно, но так системе проще будет понять, с каким типом файлов ей придется работать.
`FileName`	- имя файла. Можно сразу же указать имя файла, с адресом и расширением, но обычно это свойство оставляют пустым. После того, как диалог сработает, и пользователь выберет файл для открытия, то в этом свойстве будет и адрес, и имя этого файла. А они нам будут нужны в методе `Memo1.LoadFromFile()`
`Filter`	- фильтр типов файлов. Здесь можно задать фильтрацию файлов по их типам, используя маску файлов. В маске знак "`*`" означает любое количество любых символов. Например, в фильтре можно указать маски `*.txt`, что означает "любой файл с расширением txt", и(или) *.*, что означает "любой файл с любым расширением или без расширения".
`InitialDir`	- папка (директория, каталог), используемая по умолчанию. Здесь указывается адрес папки, с которой диалог начнет свою работу. Заполнять это свойство имеет смысл лишь тогда, когда нужные файлы у вас будут храниться в каком-то одном, конкретном месте. В нашем случае мы не знаем, где пользователю захочется сохранять свои файлы, поэтому заполнять это свойство не нужно.
`Name`	- имя компонента. С этим свойством вы уже знакомы, но тут нужно сделать одно замечание. Помните, в прошлой лекции я говорил, что если компонент один, то его можно не переименовывать? С кнопками этот совет хорош - к кнопкам в коде нам не приходится обращаться по имени. А вот к диалогам придется обращаться неоднократно. Судите сами, что проще будет набрать на клавиатуре: `OpenDialog1.FileName или OD.FileName?` Вот то-то. Поэтому совет: переименовывайте диалоги, оставляя лишь заглавные буквы имени. Если вам доведется в одной форме использовать два одинаковых диалога (а мне пока такого делать не доводилось), то можно их назвать `OD1` и `OD2`.
`Title`	- заголовок окна диалога. По умолчанию, он уже содержит нужный текст: "Открыть существующий файл". Если же вам захочется установить какое то свое, нестандартное название, например "Открыть мой текстовый файл", можете вписать его здесь.

Со свойствами разобрались, остался метод. Метод этот - `Execute`.

`Execute` вызывает стандартное диалоговое окно, и дает пользователю сделать выбор. Если пользователь сделал свой выбор, то `Execute` вернет `True`. Если пользователь отказался делать выбор и закрыл окно, не указав файла, метод вернет `False`.

Настроим наш `TOpenDialog`:

- `DefaultExt = .jpg`
- `Name = OD`

Теперь займемся фильтром. Выделите свойство `Filter` и щелкните по кнопке "**...**" справа от него. Откроется редактор фильтров, в нем надо поставить фильтры, обеспечивающие нам открытие файлов с изображениями, вместо текстовых файлов поставьте соответствующие расширения:

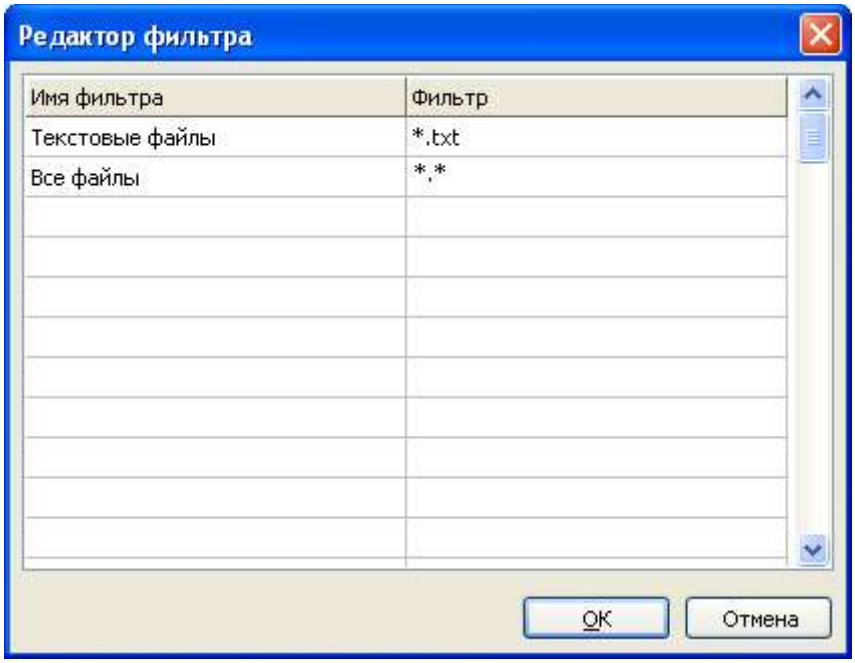


Рис. 6.4.1 Фильтры для TOpenDialog

Когда вы нажмете "**ОК**", Lazarus в свойстве `Filter` сгенерирует следующий фильтр:

```
Текстовые файлы|*.txt|Все файлы|*.*
```

Эту строку можно было бы написать и вручную, но воспользоваться **Редактором фильтров** все же удобней.

Теперь займемся программированием пунктов меню. Выберите на форме пункт "**Файл -> Выход**", и автоматически будет создан обработчик `OnClick` для этого пункта. В обработчике впишите всего одну команду:

```
procedure TfMain.FileExitClick(Sender: TObject);
begin
  Close;
end;
```

Оператор `Close` в дочернем (неглавном) окне закрывает это окно, а в главном окне завершает работу программы. Создайте таким же образом обработчик для "**Файл -> Открыть**", с его кодом вы должны быть знакомы по предыдущему проекту:

```
procedure TfMain.FileOpenClick(Sender: TObject);
begin
  if OPD.Execute then Image1.Picture.LoadFromFile(OPD.FileName);
end;
```

Для "**Файл -> Сохранить как**" код будет похожий:

```
procedure TfMain.FileSaveAsClick(Sender: TObject);
begin
  if SPD.Execute then Image1.Picture.SaveToFile(SPD.FileName);
end;
```

С пунктом "**Файл**" разобрались, остался пункт "**Окно**" и три его подпункта. Здесь мы будем менять состояние окна у `fMain.WindowState`. Код для этих событий следующий:

```
procedure TfMain.WindowMaximizeClick(Sender: TObject);
begin
  fMain.WindowState:= wsMaximized;
end;

procedure TfMain.WindowMinimizeClick(Sender: TObject);
begin
  fMain.WindowState:= wsMinimized;
end;

procedure TfMain.WindowNormalClick(Sender: TObject);
begin
  fMain.WindowState:= wsNormal;
end;
```

Не перепутайте названия событий. Сохраните проект, запустите его и попробуйте в работе. Программа должна нормально загружать файлы, менять состояние окна, завершать работу. При сохранении файла указывайте не только желаемое имя, но и расширение, например *newfile.jpg* - компонент `TImage` не умеет угадывать формат и самостоятельно подставлять нужное расширение.

Всплывающее меню

Мы с вами то и дело пользуемся всплывающим меню. Это меню, которое появляется, когда пользователь нажимает правую кнопку мыши. Реализуется оно компонентом `TPopupMenu`, который находится рядом с `TMainMenu` на **Палитре компонентов**. Поскольку оно также невизуально, устанавливайте его на любое место на форме. Редактор всплывающего меню также вызывается двойным нажатием на компонент (или нажатием на "…" в свойстве `Items`). Редактируется оно примерно также. Выделите первый (и пока единственный) пункт этого меню, в свойстве `Name` напишите `pFileMenu`, в свойстве `Caption` напишите Файл. Далее, щелкните правой кнопкой мыши по кнопке этого пункта в **Редакторе меню**, и выберите "**Создать подменю**". Создайте тут два подраздела: `pFileOpen` и `pFileSaveAs` со свойствами `Caption`, соответственно, **Открыть** и **Сохранить как**.

Аналогичным образом, создайте пункт `pWindowMenu (Caption = Окно)` с подпунктами `pWindowNormal`, `pWindowMinimize` и `pWindowMaximize (Caption = Нормальное, Свернуть, Развернуть)`.

Далее, создайте главный пункт с `Caption = "-"` (имя не меняем, это просто разделитель), и еще один пункт `Name = CloseMenu, Caption = Выход`. У вас должно было получиться такое меню:

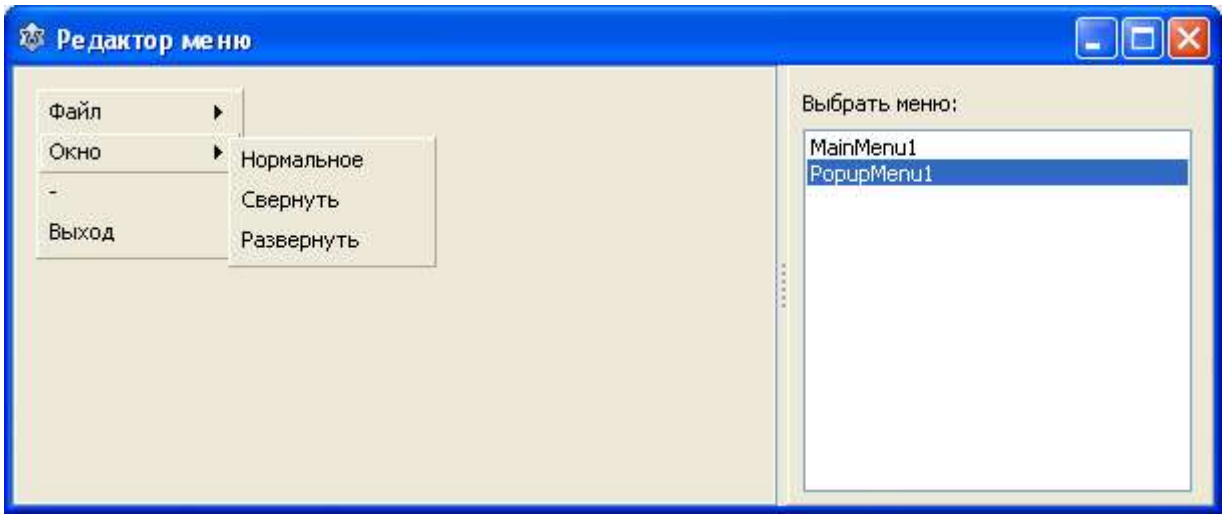


Рис. 6.5. Всплывающее меню

Меню мы сделали, но не спешите закрывать **Редактор меню**. Если вы думаете, что для всплывающего меню мы снова будем писать тот же самый код, то вы ошибаетесь, все намного проще. Выделите подпункт "**Файл -> Открыть**" - щелкайте один раз, а не дважды, иначе будет создан обработчик `OnClick`, а нам это не нужно. Если это всё же случилось, вам придется удалить пустой каркас процедуры этого обработчика, а также объявление этой процедуры в разделе `type` в верхней части модуля формы. Только не перепутайте названия процедур - все пункты всплывающего меню у нас начинаются с буквы p.

Итак, мы выделили подпункт всплывающего меню "**Файл -> Открыть**". В **Инспекторе объектов** перейдите на вкладку **События**, откройте список в событие `OnClick` и выберите из этого списка событие `FileOpenClick` - это событие открытия файла в главном меню, которое мы ранее запрограммировали:

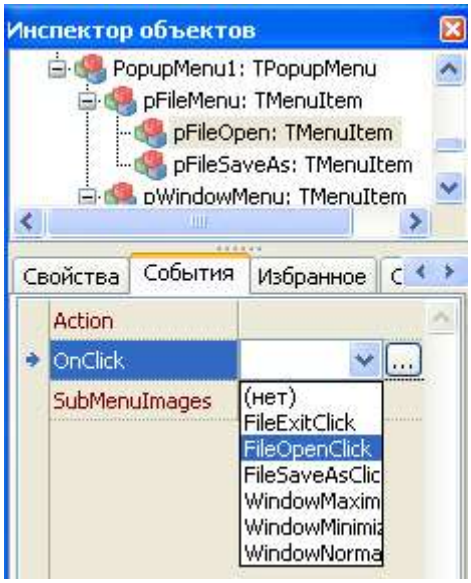
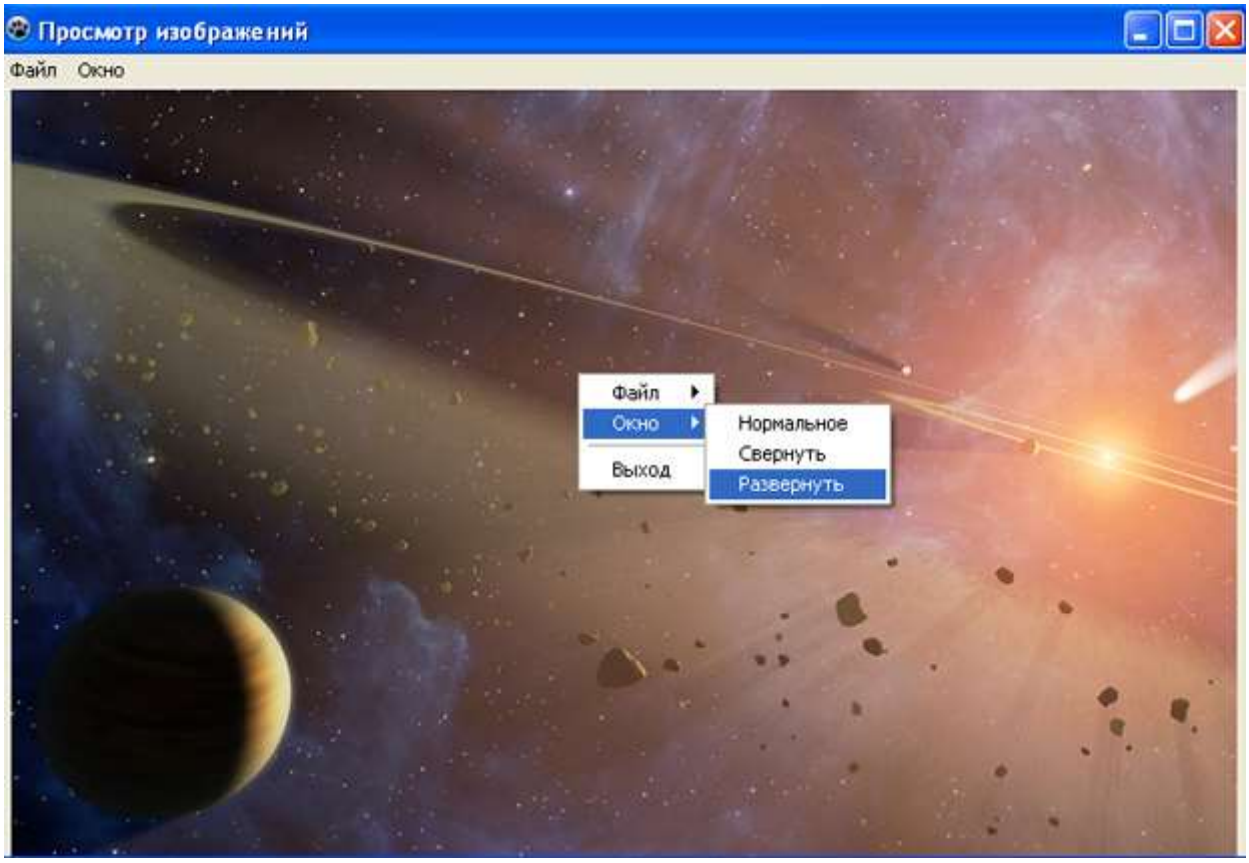


Рис. 6.6. Выбор готового события в OnClick

Таким образом, когда пользователь выберет эту команду в всплывающем меню, будет вызвано соответствующее событие главного меню, так что нам не придется дважды писать один и тот же код. Выберите соответствующие события главного меню для оставшихся команд всплывающего меню "**Файл -> Сохранить как**", "**Окно -> Нормальное**", "**Окно -> Свернуть**", "**Окно -> Развернуть**" и "**Выход**". После этого можно закрыть **Редактор меню**.

Однако и этого недостаточно, чтобы пользователь мог вызвать всплывающее меню - его еще нужно привязать к какому-нибудь компоненту. Обычно его привязывают либо к форме, либо к панели. Если на форме две панели, то для каждой можно сделать своё всплывающее меню, со своими командами. У нас всю часть формы занимает компонент `Image1`, к нему и привяжем всплывающее меню. Выделите `Image1`, перейдите на вкладку **Свойства Инспектора объектов**, и в свойстве `PopupMenu` выберите наш `PopupMenu1`. Теперь, если пользователь нажмет над `Image1` правую кнопку мыши, выйдет всплывающее меню. Если вы все сделали правильно, то команды всплывающего меню будут вызывать соответствующие команды главного меню, и все будет работать нормально:



увеличить изображение
Рис. 6.7. Работа всплывающего меню в загруженном приложении

Компонент TImageList

`TImageList` - это контейнер для хранения списка изображений. Как правило, в `TImageList` хранят изображения для меню и кнопок панелей инструментов. Компонент `TImageList` находится на вкладке **Common Controls Палитры компонентов**:

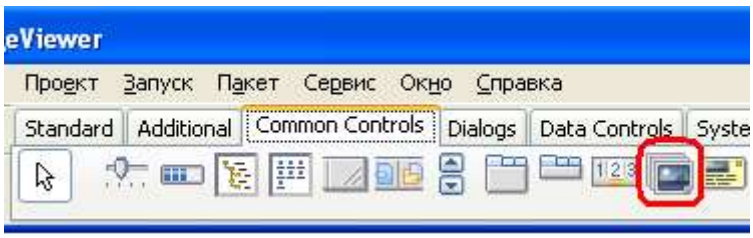


Рис. 6.8. Компонент TImageList

Компонент невизуальный, так что его также можно установить на любое свободное место. Щелкните дважды по нему, чтобы открыть **Редактор ImageList**. В **Редакторе** нажмите кнопку "**Добавить**", откроется диалоговое окно "**Добавить изображения**". Теперь вот что: для пунктов меню нам нужны маленькие изображения, 16*16 пикселей. По умолчанию, Lazarus устанавливается в папку **C:\Lazarus**. Если вы не меняли эту папку, то картинки для пунктов меню у вас будут по адресу

C:\Lazarus\images\menu

Изображений там не так, чтобы очень много, гораздо больше интересных изображений, да и целых бесплатных коллекций изображений можно найти на просторах Интернета. Однако на первое время обойдемся тем, что уже есть. Откройте указанный адрес в диалоге выбора изображений. Нам нужна картинка для пункта меню "**Файл -> Открыть**". Тут лучше подойдет файл **menu_project_open.png**. Выберите этот файл и нажмите кнопку "**Открыть**" - файл попадет в список под индексом 0. Как и в других списках, индексация в `TImageList` начинается с 0, индекс -1 означает, что изображения не выбрано. Снова нажмите кнопку "**Добавить**" в редакторе. Для "**Файл -> Сохранить как**" подойдет файл **menu_project_saveas.png**. Изображение встанет под индексом 1.

Впрочем, изображения можно перемещать по списку, менять их местами с помощью кнопок "**Переместить вверх**" и "**Переместить вниз**". Под индексом 2 установите файл **menu_exit.png** - это будет изображение для "**Файл -> Выход**". Для пунктов меню "**Окно**" подходящих изображений нет, поэтому мы оставим эти команды без соответствующих картинок.

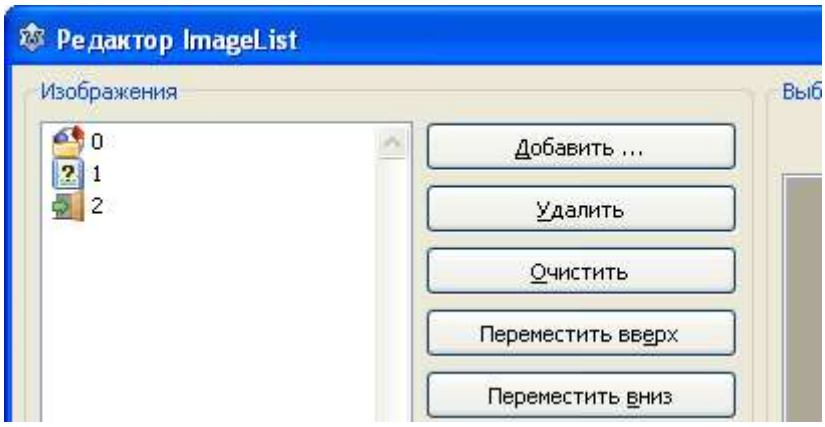


Рис. 6.9. Три установленных изображения в редакторе ImageList

Нажмите кнопку "**OK**", чтобы закрыть редактор. Теперь нам нужно, чтобы главное меню увидело эти изображения. Выделите компонент `MainMenu1`. В его свойстве `Images` выберите наш `ImageList1`. Таким образом, мы связали главное меню со списком изображений.

Далее, дважды щелкните по `MainMenu1`, чтобы открыть редактор меню. Выделите подпункт "**Открыть**" пункта "**Файл**". Вы видите, что его свойство `ImageIndex` равно -1, то есть, изображения не выбрано. Откройте список изображений и выберите первое изображение, с индексом 0. Для подпункта "**Сохранить как**" установите индекс 1. Для "**Выход**" - индекс 2. В результате пункты меню "**Файл**" станут отображаться вместе с изображениями:

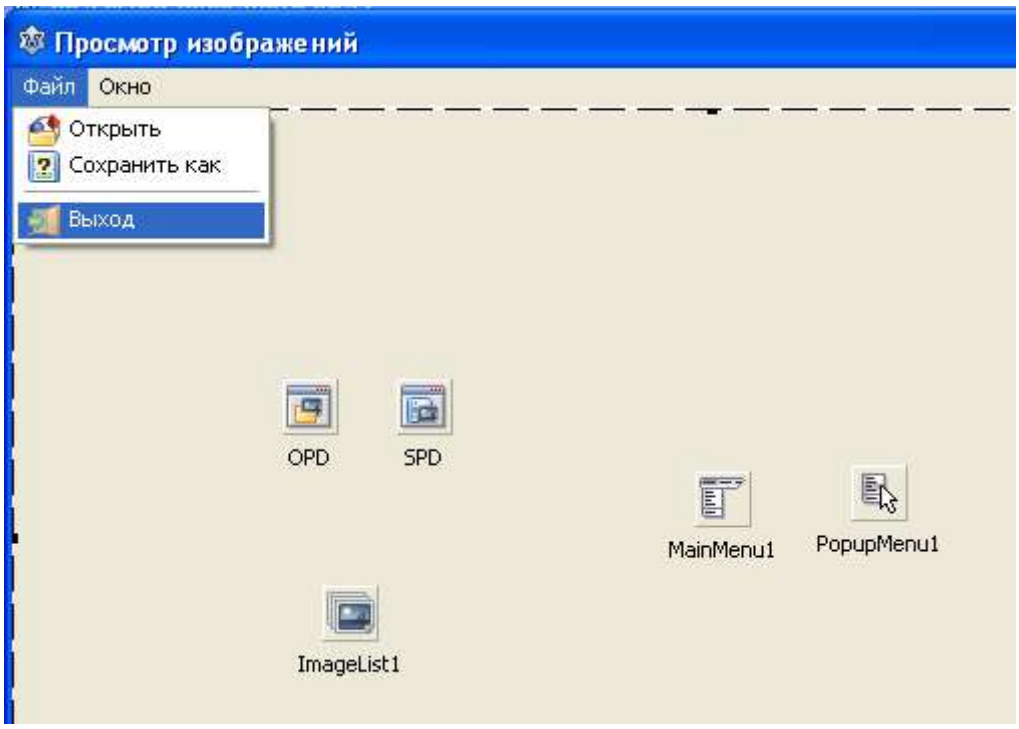


Рис. 6.10. Меню с изображениями

Большой пользы от этого нет, но проект украшает. То же самое сделайте для соответствующих пунктов всплывающего меню: сначала в его свойстве `Images` выберите наш список `ImageList1`. Затем откройте редактор меню, выделите поочередно пункты "**Открыть**", "**Сохранить как**" и "**Выход**", выберите для них соответствующие `ImageIndex`. Сразу же в редакторе меню вы не заметите изменений, однако закройте его, сохраните проект и запустите программу на выполнение. Щелкните правой кнопкой мыши по окну и, пожалуйста, меню с изображениями:

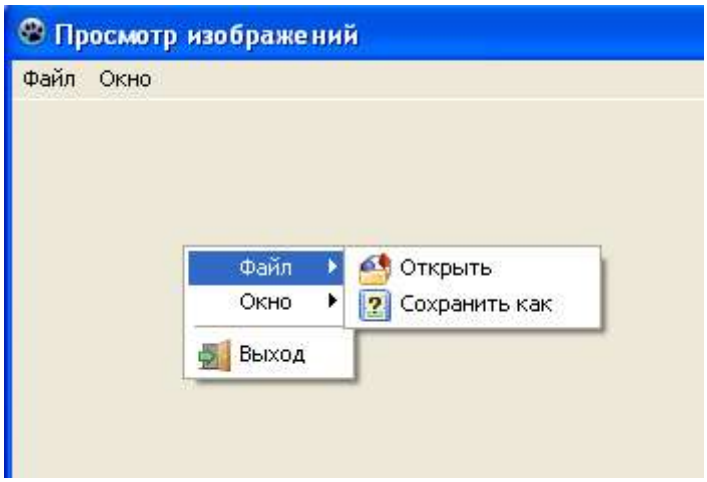


Рис. 6.11. Всплывающее меню с изображениями

Горячие клавиши

Тут самое время вспомнить об одном важном свойстве подпунктов меню, как главного, так и всплывающего - `ShortCut`.

`ShortCut` - свойство, позволяющее выбрать для команды меню сочетание клавиш, так называемые "горячие клавиши". Если какое то сочетание клавиш присвоено одной из команд, то пользователю необязательно будет лезть в меню, чтобы выполнить эту команду, достаточно будет нажать соответствующее сочетание клавиш. Так, для команды меню **Файл->Открыть** традиционно применяют сочетание **<Ctrl+0>**; для команды **Файл->Сохранить** - **<Ctrl+S>**; для команды **Файл->Выход** можно использовать **<Ctrl+Q>**. Откройте редактор главного меню, выделяйте по очереди в меню **Файл** подпункты "**Открыть**", "**Сохранить как**" и "**Выход**". Выберите для этих команд соответствующие сочетания в свойстве `ShortCut`. В результате меню приобретет такой вид:

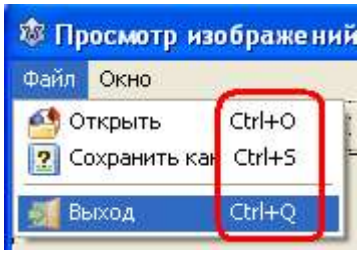


Рис. 6.12. Горячие клавиши в меню

Теперь пользователь сможет выполнять эти команды как с помощью меню, так и с помощью "горячих клавиш". Делать то же самое с пунктами всплывающего меню имеет смысл только в том случае, если его команды отличаются от команд главного меню. В нашем случае все равно будут выполняться команды главного меню, так что указывать эти же горячие клавиши во всплывающем меню не нужно.

Панель инструментов

Самый простой способ организовать **Панель инструментов** - установить обычную панель **TPanel** с настройками свойств `Align = alTop`, `AutoSize = True`, и уже на эту панель установить кнопки **TSpeedButton**. И панель, и кнопки мы с вами изучали в лекции №3, поэтому останавливаться на этом не будем. Изучим другой, более профессиональный способ организации панели инструментов.

Для этого нам потребуется компонент `TToolBar` с вкладки **Common Controls Палитры компонентов**:

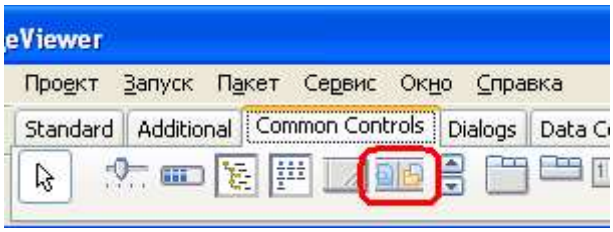


Рис. 6.13. Компонент TToolBar

Просто установите его на форму, и он сам займет нужное место - растянется по всей верхней границе окна. Обращу ваше внимание на некоторые свойства компонента, которые могут представлять для нас интерес.

Images	- ссылка на список изображений для кнопок панели. Выполняет такую же функцию, как соответствующее свойство меню.
DisabledImages	- ссылка на список изображений, которые будут привязаны к неактивным кнопкам. Чаще всего это свойство оставляют пустым, в таком случае у неактивных кнопок (у которых Enabled = False) будет такое же изображение, как у активных, но только серого цвета. При желании для неактивных кнопок можно подобрать альтернативные изображения.
HotImages	- ссылка на список изображений для кнопок, над которыми в данный момент находится указатель мыши. Чаще всего это свойство не заполняют, и тогда изображения у таких кнопок не меняется. При желании, можно подобрать альтернативные изображения, чтобы выделять текущую кнопку.
EdgeBorders	- установка бордюра по сторонам панели. Имеет 4 подсвойства, соответствующие границам панели, и если в соответствующем подсвойстве установлено True, то данная граница будет выделена. По умолчанию, включена только верхняя граница, отделяющая панель инструментов от строки с главным меню. Если желаете, можете включить отображение всех границ.
Flat	- если True, то кнопки плоские, как в современных панелях Windows, если False - выпуклые, как в старых программах. По умолчанию равно True, так лучше и оставить.
ShowHint	- если True, то всплывающие подсказки будут выходить, если False, то нет. С таким свойством нам уже доводилось встречаться. На панели инструментов обычно отображаются кнопки с изображениями, но без текста. Пользователь, если только знакомится с вашей программой, поначалу может и не сообразить, какая кнопка для чего нужна. Поэтому в панели инструментов данную подсказку лучше включать (по умолчанию ShowHint = False). Затем в свойстве Hint кнопок вы пропишите нужные подсказки, и пользователь сможет их прочитать, когда подведет указатель мыши к той или иной кнопке.

Сделаем на панели **ToolBar1** следующие установки:

EdgeBorders	- включаем все границы, если есть желание. А вообще это необязательно.
Images	- выберите наш ImageList1. Как видите, один и тот же список изображений мы используем и для главного меню, и для всплывающего, и для панели инструментов.

```
ShowHint = True
```

Остальные свойство можно не изменять.

Займемся созданием кнопок. Вначале сделаем кнопку "**Выход из программы**", затем установим разделитель, затем кнопки "**Открыть файл с изображением**" и "**Сохранить файл как**". Для команд пункта "**Окно**" мы кнопок делать не будем, тем более, что у нас нет для них изображений. Итак, щелкните правой кнопкой мыши по панели инструментов, и выберите команду "**Новая кнопка**". На панели инструментов появилась кнопка, которой Lazarus автоматически присвоил имя **ToolButton1**. Измените ее свойство `Name` на `bExit`, чтобы название соответствовало действию. В свойстве `Hint` напишите следующую подсказку:

```
Выход из программы
```

А в свойстве `ImageIndex` выберите изображение с индексом 2. Теперь перейдем на вкладку **События Инспектора объектов**, и в свойстве `OnClick` выберем `FileExitClick` - событие `OnClick` для пункта "**Выход**" главного меню. Как видите, для панели инструментов нам тоже не придется заново писать код.

Снова щелкните правой кнопкой по панели инструментов, и выберите команду "**Новый разделитель**". Тем самым вы вставите разделитель - вертикальную черту между предыдущей кнопкой и следующими. Ничего с этим разделителем делать больше не нужно, разве что вам покажется, что разделитель слишком (или наоборот, недостаточно) широк. В этом случае выделите разделитель и измените его свойство `Width`.

Вставьте еще одну кнопку. Назовите ее **bOpen**, выберите для нее изображение под индексом 0, в свойстве `Hint` напишите следующую подсказку:

```
Открыть файл с изображением
```

В событии `OnClick` этой кнопки выберите соответствующее событие из главного меню.

Вставим еще одну, последнюю кнопку. Назовем ее **bSaveAs**, в `ImageIndex` выберем картинку с индексом 1, в свойстве `Hint` напишем подсказку:

```
Сохранить файл как
```

Выберем для кнопки соответствующее событие `OnClick`.

В результате у вас должна получиться такая **Панель инструментов**:

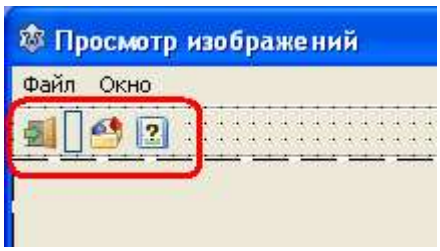


Рис. 6.14. Панель инструментов в проекте

Это все. Сохраните проект и запустите его на выполнение. Если вы все сделали правильно, теперь у вас есть правильно действующая панель управления с тремя кнопками и одним разделителем.