

Глава 6. Работа с компонентами.	84
6.1 Основная форма и её свойства.	84
6.2 Событийная модель Windows.	95
6.3 События главной формы.	96
6.3 Палитра компонентов.	97



Глава 6. Работа с компонентами.



Мы уже создали первую и самую простую программу, и разобрались, из чего она состоит. Помимо этого, я напихал в тебя теории по самым «не хочу», так что пора увидеть всё сказанное на практике.

И вот только теперь мы начнём знакомиться с компонентами и писать примеры с их использованием. Вот это как раз то, с чего начинается большинство книг по Delphi. Но я не буду их повторять, а постараюсь описать как можно подробнее и сказать то, что недосказано в других книгах.

Изучение компонентов будет постепенное. Сначала мы попробуем их в действии и познакомимся с их основными возможностями, а потом залезем внутрь и разберём все косточки. Но это потом. Сейчас мы смотрим только на одежду.

В Delphi для тебя уже собрана большая коллекция компонентов, позволяющих визуальнo создать любой интерфейс программы. В этой главе мы познакомимся с тремя основными закладками палитры компонентов: standard, additional и Win32. Но прежде чем это сделать, я расскажу про основную форму.

6.1 Основная форма и её свойства.

Основная форма – это окно будущей программы. На нём ты можешь располагать визуальные компоненты в любом виде и порядке.

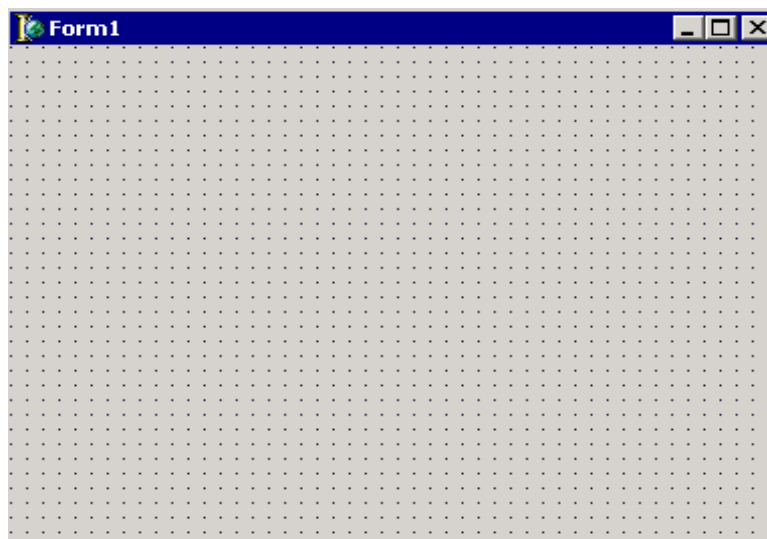


Рисунок 6.1 Основная форма

Как ты уже знаешь, если выделить какой-то компонент, то в объектном инспекторе появятся его свойства и события. Сейчас я рассмотрю основные свойства и события формы. Большинство из них присутствуют и у компонентов. Поэтому в будущем я уже не

буду повторяться: Когда я буду описывать событие, которое может быть не только у формы, но и у компонента, то я так и буду писать «форма/компонент».



Я буду показывать свойства и их назначения, а ты создай новый проект в Delphi и попробуй играть с этими свойствами. Так ты лучше сможешь понять, на что они влияют.

ActiveControl - Указывает на компонент, который должен быть активным по умолчанию.

Align - Выравнивание компонента. Любой компонент может быть выровнен по одной из сторон родительского компонента. Этому свойству можно присвоить следующие значения:

alNone – нет выравнивания. Как нарисовал, так и будет.

alBottom – выравнивание по нижнему краю.

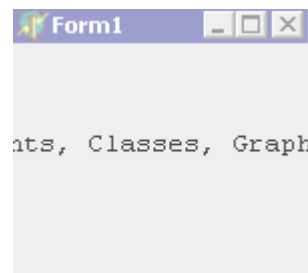
alLeft - выравнивание по левому краю.

AlRight - выравнивание по правому краю.

AlTop - выравнивание по верхнему краю.

Компоненты выравниваются относительно формы, а форма выравнивается относительно окна.

AlphaBlend - Тип свойства – логический. Свойство формы. Означает, имеет ли форма прозрачность. Если это свойство равно true, то окно будет прозрачным. На рисунке справа показан пример полупрозрачного окна. Степень прозрачности задаётся через свойство AlphaBlendValue.



ВНИМАНИЕ!!! Прозрачность работает не на всех системах.

AlphaBlendValue - Тип свойства – целое число. Степень прозрачности формы. Здесь можно задавать числовое значение степени прозрачности от 0 до 255. Если поставишь 0, то форма будет абсолютно прозрачной. 255 означает полную непрозрачность. Чтобы сделать форму полупрозрачной, нужно выставить какое-нибудь промежуточное значение (можно 127).



На компакт диске, в директории \Примеры\Глава 6\Прозрачное окно ты можешь увидеть пример программы. Могу дать гарантию, что он работает в Windows 2000. Но в Win9x пример может не работать.

Anchor - Это свойство есть и у формы и у компонентов. Оно показывает, как происходит закрепление к родительскому объекту. Это свойство раскрывающееся. Если ты щелкнешь по квадрату слева от имени свойства, то раскроется список из четырёх дополнительных свойств:

AlphaBlendVal	255
Anchor	[akLeft,akTop]
akLeft	True
akTop	True
akRight	False
akBottom	False
AutoScroll	True
AutoSize	False

akLeft – прикреплять левый край (по умолчанию true).

akTop – прикреплять верхний край (по умолчанию true).

akRight – прикреплять правый край (по умолчанию false).

akBottom – прикреплять нижний край (по умолчанию false).

По умолчанию прикрепление происходит по левому и верхнему краю.

AutoScroll - Тип свойства – логический. Будет ли форма автоматически производить скроллинг, или нет.

AutoSize - Тип свойства – логический. Должны ли компоненты на форме автоматически корректировать размеры.

BorderIcons - Свойство определяющее, какие кнопки должны присутствовать у окна. Это свойство раскрывающееся. Если ты щелкнешь по квадрату слева от имени свойства, то раскроется список из четырёх свойств:

biSystemMenu – показать меню (иконка слева в строке заголовка окна) и другие кнопки заголовка окна.

biMinimize – кнопка минимизации окна.

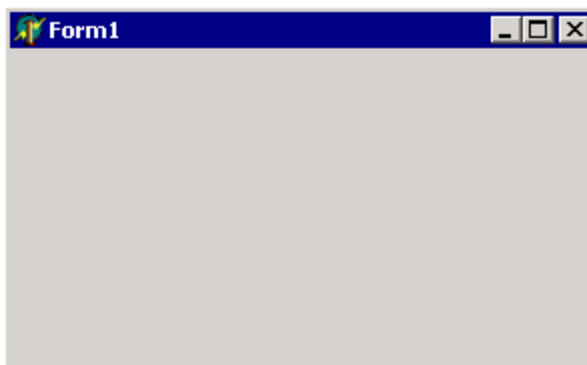
biMaximize – кнопка максимизации окна.

biHelp – кнопка помощи.

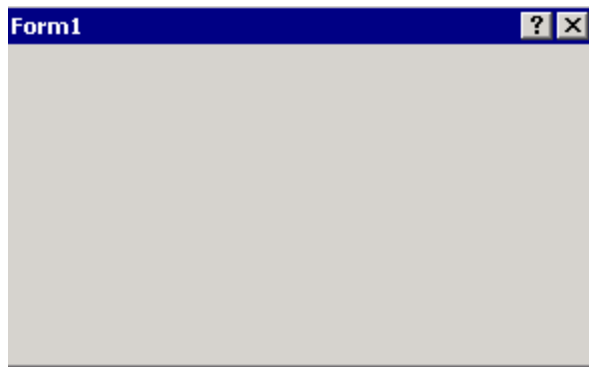


BorderStyle – Свойство формы. Отвечает за вид обложки окна. Это свойство может принимать следующие значения.

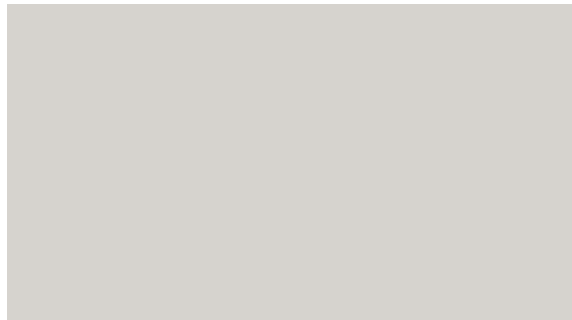
bsSizeable – установлено по умолчанию. Стандартное окно, с нормальной обложкой, которое может изменять свои размеры. Смотри скрин окна:



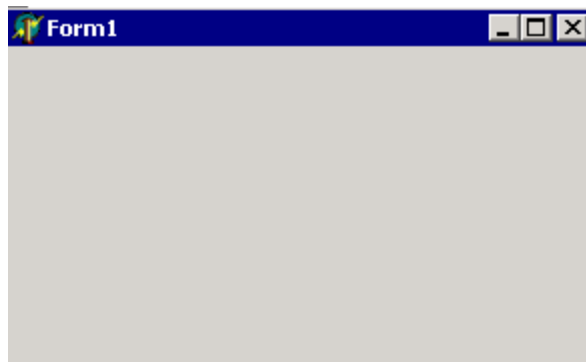
bsDialog – окно выглядит в виде диалога. Смотри скрин окна:



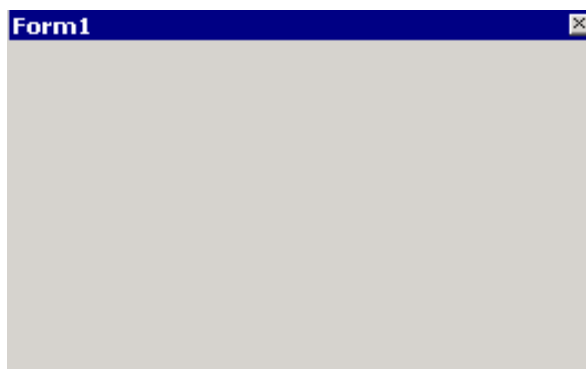
bsNone – окно вообще без обложки. Смотри скрин окна:




bsSingle – На первый взгляд это простое окно, а если попробовать изменить его размеры, то можно получить облом. Это окно с фиксированным размером и изменять его мышкой нельзя. Изменить размер можно только кнопкой Maximize. Смотри скрин окна:



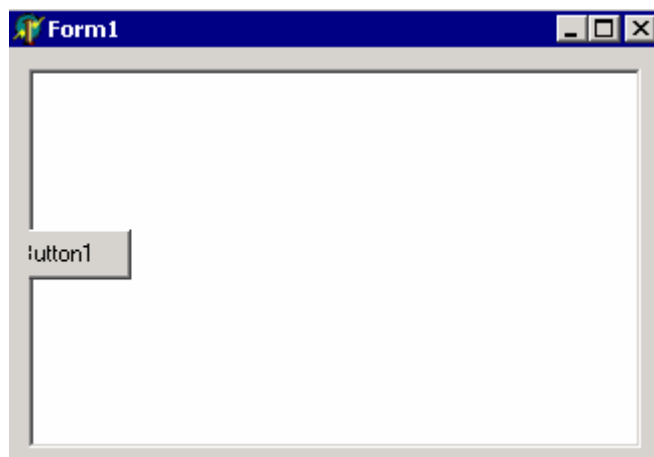
bsSizeToolWin – окно с тонкой обложкой. Особенно это заметно в заголовке окна. Смотри скрин окна:



bsToolWindow – я не буду приводить скрин этого окна, потому что он ничем не отличается от предыдущего. Единственная разница – у этого окна нельзя изменять размеры окна.

 Все примеры окон можно найти на компакт диске в директории \Примеры\Глава 6\Окна\. Но я тебе советую попробовать самому создать приложение и поиграть с его свойствами.

BorderWidth – ширина оборки окна. Пока что все окна, которые мы рассматривали, имели ширину оборки равную нулю. На скрине ниже показано окно с оборкой равной 10. В центре окна растянута на всю площадь поле для ввода текста и несмотря на это, в окне по краям видны широкие оборки. Для больше го эффекта я поместил на форму ещё и кнопку, которая исчезает при пересечения оборки.

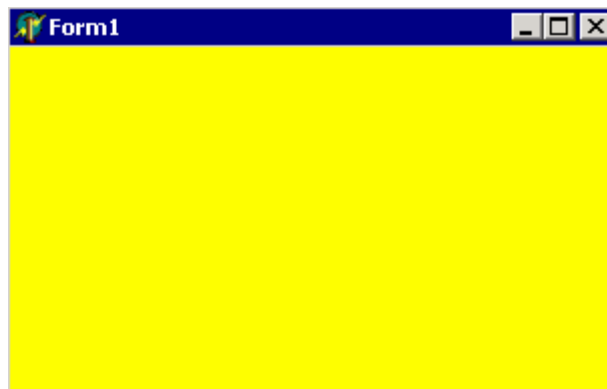



Caption – это строковое свойство, которое отвечает за заголовок окна. Мы уже использовали его, когда писали программу «Hello World».

ClientHeight – это свойство в виде целого числа показывает высоту клиентской области окна. Это высота без учёта ширины оборки и системного меню, только рабочая область.

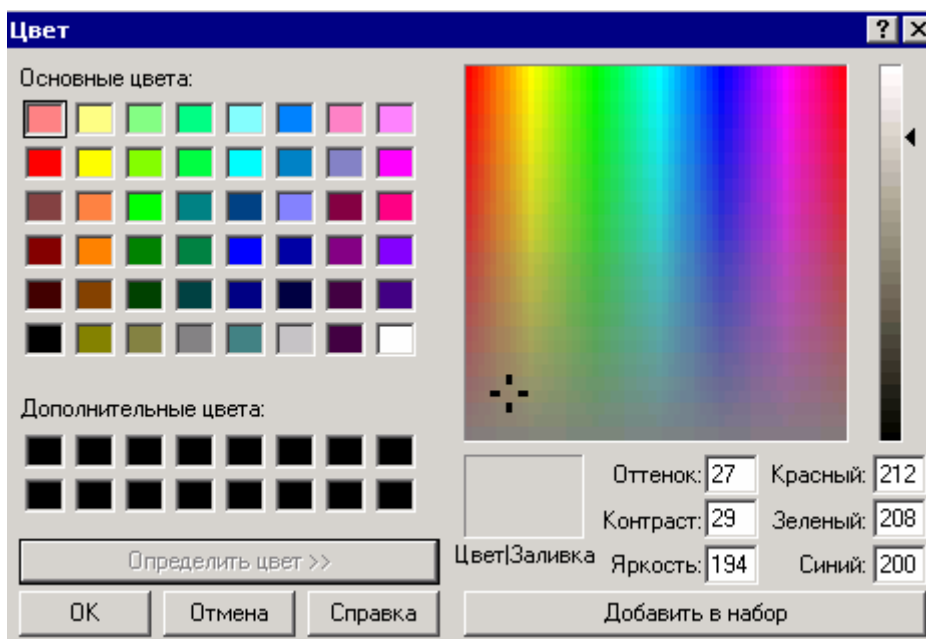
ClientWidth - это свойство в виде целого числа показывает ширину клиентской области окна. Это ширина без учёта ширины обёртки и системного меню, только рабочая область.

Color – цвет клиентской области окна.



 На компакт диске, в директории \Примеры\Глава 6\Цвет окна ты можешь увидеть пример программы.

В списке выбора есть все системные цвета, которые ты можешь выбрать. Но если ты хочешь использовать какой-то специфичный цвет, то можешь дважды щёлкнуть по этому параметру, и перед тобой откроется стандартное окно выбора цвета:



Constraints – в этом свойстве содержатся максимальные значения размеров окна.


MaxHeight – максимальная высота окна.

MaxWidth – максимальная ширина окна.

MinHeight – минимальная высота окна.





















MinWidth – минимальная ширина окна.

Если ты установишь эти значения, то окно нельзя будет растянуть больше максимального размера и уменьшить меньше минимального.


 На компакт диске, в директории \Примеры\Глава 6\Размер окна ты можешь увидеть пример программы в которой главное окно нельзя увеличить более чем 400x400 и меньше 200x200.

Ctrl3D – Тип свойства – логический. Оно указывает - показывать окно/компонент в псевдо 3D плоскости или нет. Этот параметр остался ещё от Windows 3.1, когда он действительно имел смысл. Сейчас даже если ты отключишь 3D, окно сильно не изменится. Поэтому про это свойство можно забыть.

Cursor – это свойство отвечает за курсор, который будет отображаться при наведении мышкой на форму/компонент. Тебе доступны следующие курсоры:

Имя курсора	Вид	Имя курсора	Вид
crNone	Нет	CrArrow	
crCross		crIBeam	
crSizeNESW		crSizeNS	
crSizeNWSE		crSizeWE	
crUpArrow		crHourGlass	
crDrag		crNoDrop	
crHSplit		crVSplit	
crMultiDrag		crSQLWait	
crNo		crAppStart	
crHelp		crHandPoint	
crSize		crSizeAll	

DockSite - Тип свойства – логический. Указывает, можно ли на форму/компонент бросать другие компоненты с помощью Drag&Drop. Это свойство создаёт эффект, который ты мог наблюдать в MS Office, когда панели инструментов можно отрывать от формы и прикреплять обратно. Вот это свойство как раз и разрешает прикреплять компоненты.

 На компакт диске, в директории \Примеры\Глава 6\Dock ты можешь увидеть пример программы в которой можно отрывать панель от формы и прикреплять обратно.

DragKind – вид перетаскивания объекта при Drag&Drop. Здесь тебе доступны два варианта:

dkDrag – стандартный Drag&Drop при котором объект остаётся на месте.

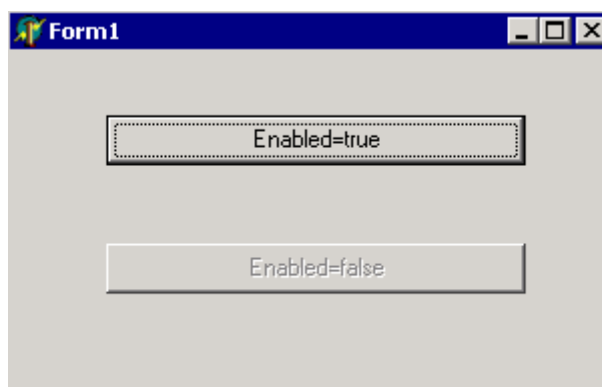
dkDock – перетаскивать сам объект. Этот параметр следует выбрать, если нужно чтобы компонент мог прикрепляться к другим компонентам или форме. В примере к предыдущему свойству панель имеет именно такое свойство, чтобы она могла прикрепиться к форме.


DragMode – режим Drag&Drop. Здесь тебе доступны два варианта:

dmManual – ручной режим. При таком режиме ты сам должен запускать перетаскивание объекта.

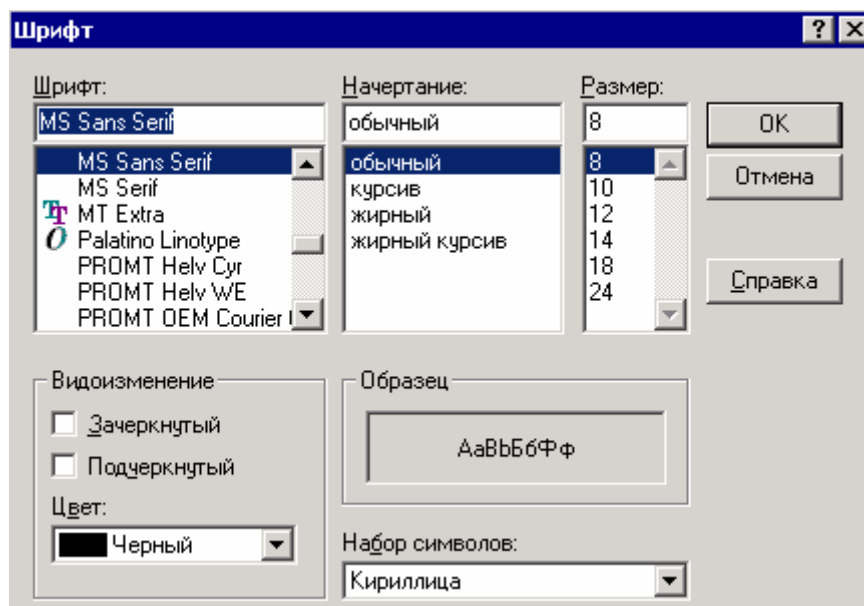
dmAutomatic – режим Drag&Drop будет включаться автоматически, если пользователь начал тащить мышкой компонент. При этом не нужно писать дополнительный код, как при ручном режиме.

Enabled – Тип свойства – логический. Доступность компонента. Если это свойство равно true, то пользователь может работать с этим компонентом. Иначе компонент недоступен и окрашен серым цветом.



 На компакт диске, в директории \Примеры\Глава 6\Enable ты можешь увидеть пример программы в которой есть две кнопки на форме. Одна из них доступна, а другая нет.

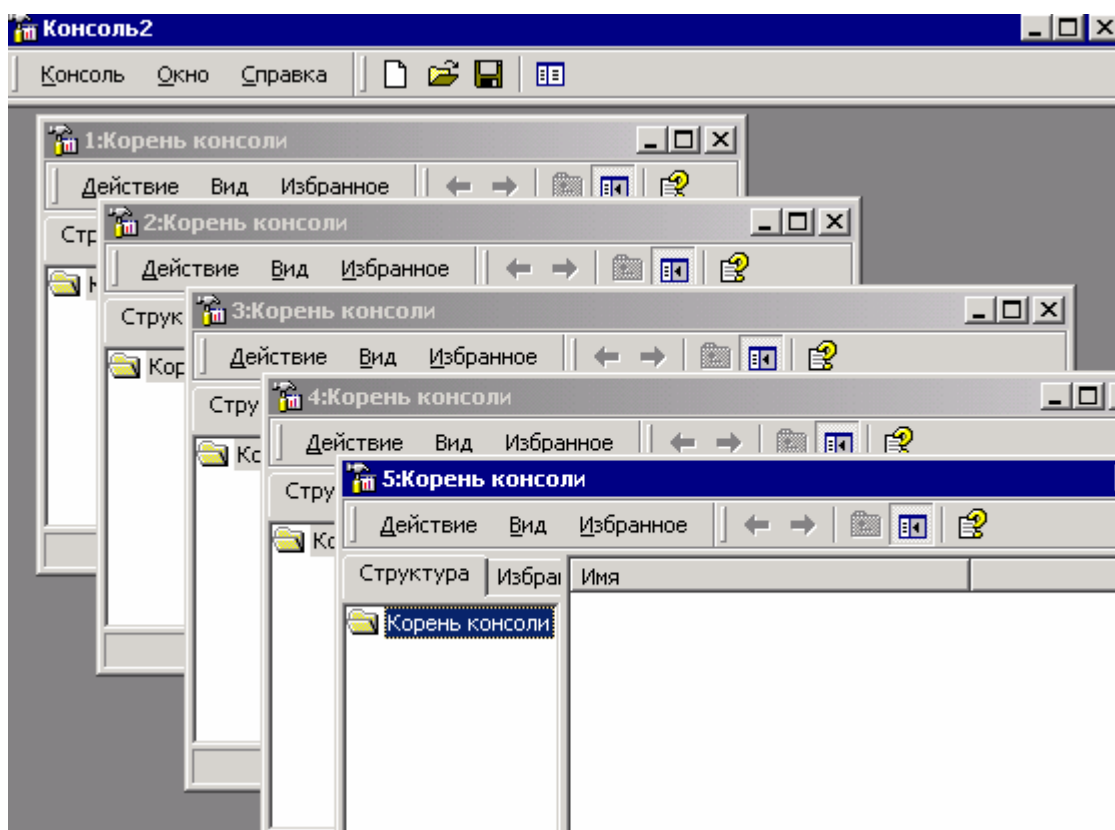
Font – шрифт используемый при выводе текста на форме. Если ты дважды щёлкнешь по этой строке, то перед тобой появится стандартное окно Windows выбора шрифта:



FormStyle - Стиль формы. Здесь тебе доступны для выбора следующие варианты

fsNormal – нормальное окно.

fsMDIForm – Окно является родительским для MDI окон. Если ты помнишь старый Office, то должен помнить, что там внутри основного окна можно было перемещать другие окна. Это окна относятся к классу MDI – мультидокументные окна. Хотя Microsoft не рекомендует использовать MDI окна и вроде как сама отказалась от их использования, а в Windows 2000 консоль MMC выполнена именно так:



fsMDIChild – окно является дочерним MDI окном. *fsMDIForm* – создаёт главное окно, а *fsMDIChild* создаёт дочернее, то есть то окно, которое будет внутри главного.

ВНИМАНИЕ!!! – главное окно не может быть такого типа.

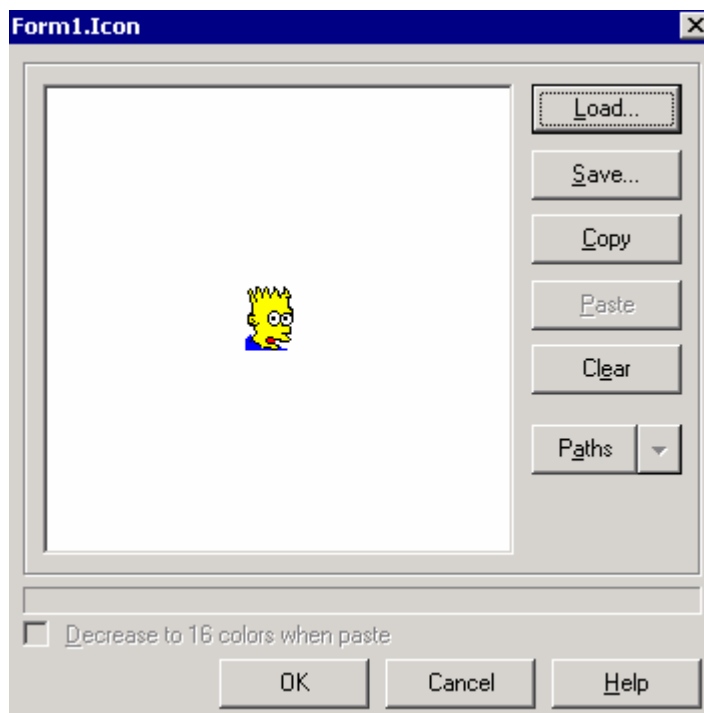
fsStayOnTop – Окно с этим параметром будет находиться всегда поверх остальных.

Height – Тип свойства – целое число. высота окна.

Hint – текст подсказки, который будет появляться в строке состояния при наведении мышкой на форму/компонент.

HorzScrollBar – параметры горизонтальной полосы прокрутки. Этот параметр я пока не буду рассматривать, потому что если его раскрыть, то там будет столько настроек, что это тема отдельного разговора.

Icon – иконка отображающаяся в заголовке окна. Если дважды щёлкнуть по этому свойству, то появится окно загрузки иконки.



В этом окне есть следующие кнопки:

Load – загрузить иконку из файла.

Save – сохранить иконку в файл.

Copy – копировать иконку в буфер обмена.

Paste – Вставить иконку из буфер обмена.

Clear – очистить текущую иконку.

Left – Тип свойства – целое число. левая позиция окна.

Menu – меню, которое используется в главном окне. Этот параметр стоит рассмотреть отдельно.

Name – имя формы/компонента. Помни, что какое имя ты здесь введёшь, так и будет называться объект, отвечающий за эту форму/компонент (только в начале добавится буква T).

Мы с тобой уже рассмотрели исходник, который сгенерировал Delphi для пустого проекта. Там объект назывался TForm1. Обрати внимание, что здесь написано просто From1. Если ты попробуешь изменить свойство *Name*, то и имя объекта измениться.



Старайся давать формам/компонентам понятные имена. Так легче будет понять, для чего они предназначены. Я думаю, что удобней будет работать с компонентом, если у него имя будет ExitButton или NewFileButton, а не просто Button1 и Button2.

ParentFont – Тип свойства – логический. Если это свойство равно true, то для вывода текста будет использоваться тот же шрифт, что и у родительского объекта. Иначе используется тот, что укажешь ты.

Position – Позиция окна при старте приложения. Здесь тебе доступны следующие варианты:

poDefault – Windows сам будет решать, где расположить окно и какие будут его размеры.

poDefaultPosOnly - Windows сам будет решать только где расположить окно, а размеры его будут такими, какими установишь ты в свойствах.

poDefaultSizeOnly - Windows будет решать только какими будут размеры окна, а позиция будет такая, какую ты укажешь в свойствах.

poDesigned – И размер, и позиция будут такими, какими ты укажешь в свойствах.

poDesktopCenter – окно будет располагаться по центру рабочего стола.

poMainFormCenter – окно будет располагаться по центру основной формы.

poOwnerFormCenter – окно будет располагаться по окна владельца. То есть того окна, которое вызвало это.

poScreenCenter - окно будет располагаться по центру экрана.

ShowHint – Тип свойства – логический. Оно показывает - нужно ли показывать подсказки.

Tag – это свойство имеет тип – целое число. Оно ни на что не влияет и ты можешь его использовать в своих целях.

Top – Тип свойства – целое число. верхняя позиция окна.

TransparentColor – Тип свойства – логический. Является ли форма или компонент прозрачным. В отличие от AlphaBlend, эта прозрачность работает всегда. Зато нельзя сделать полупрозрачные формы и компоненты.

TransparentColorValue - прозрачный цвет.

VertScrollBar – Вертикальная полоса прокрутки. Она имеет те же параметры, что и горизонтальная и мы рассмотрим её отдельно.

Visible - Тип свойства – логический. Если оно равно true, то форма/компонент видимые. Иначе форма/компонент невидим.

Width - Тип свойства – целое число. Ширина окна.

WindowState – состояние окна после запуска. Тебе доступны следующие параметры:

wsNormal – окно показывается в нормальном состоянии.

wsMaximized - окно показывается максимизированным.

wsMinimized - окно показывается минимизированным.

На этом обзор свойств формы считаю законченным.

6.2 Событийная модель Windows.

Вся работа операционной системы (ОС) Windows основана на понятии *события*. Что это значит? Давай попробуем разобраться.

Внутри ядра Windows создаётся очередь событий. Когда какое-нибудь приложение или устройство, изменило своё состояние и хочет сообщить об этом операционной системе, то оно помещает в эту очередь соответствующее сообщение. ОС Windows обрабатывает его и если необходимо, то реагирует на изменения.

Давай рассмотрим реальный пример события и реакции на него. Допустим, что мы передвинули курсор мыши. Она генерирует событие и помещает его в очередь сообщений. Когда Windows доходит до обработки этого сообщения, то он получает новые координаты курсора мышки. Так как положение курсора изменилось, ОС должна перерисовать его в новой позиции на экране. После этого, Windows переходит к обработке следующего сообщения.

Если в очереди нет сообщений, то Windows переходит в состояние ожидания. Но такое бывает очень редко. Даже когда ты не работаешь за компьютером, и он простаивает, в фоне работает очень много процессов, которые отнимают процессорное время и генерируют свои события.

Когда ты нажимаешь на кнопку, также генерируется событие, что кнопка нажата. Любое действие, которое несёт в себе какие либо изменения может генерировать системное событие. Это очень эффективная и удобная модель, благодаря которой и реализуется многозадачность Windows.

Давай взглянем на простую очередь (она не является очередью ОС Windows, это просто пример):

Событие	Идентификатор приложения	Дополнительно
Нажата клавиша	261	A
Перерисовать экран	385	(12, 46, 336, 267)
Перемещена мышь	261	(356,451)

Первая колонка показывает тип события. Вторая колонка показывает идентификатор приложения, которое сгенерировало событие. В третьей показываются дополнительные параметры. Так, например, при нажатии клавиши на клавиатуре, в качестве дополнительного параметра идёт буква, которую нажали. Конечно же, в реальной ситуации будет не буква, но у нас же это просто пример.

ОС Windows берёт первую строку из очереди и обрабатывает её. Потом берёт вторую строку. Она уже относится к другому приложению. Третья строка опять относится к первому приложению. Таким образом, ОС последовательно обрабатывает события разных приложений, что даёт многозадачность.

Конечно же, многозадачность построена не только на сообщениях и здесь много дополнительных факторов. Но очереди играют достаточно большую роль.

В Delphi все компоненты так же работают через события. Ты будешь постоянно создавать обработчики событий для разных ситуаций. Например, можно создать обработчик события для нажатия клавиши на клавиатуре и производить в нём какие-то действия. Например, по нажатию определённой клавиши, можно выводить окно (действие как у горячих клавиш).

Обработчик события – это простая процедура или функция, которая вызывается по наступлению какого-то события.

6.3 События главной формы.

Здесь я дам описание большинству событий, которые может отлавливать главная форма приложения. Конечно же, тебе доступно намного больше и ты можешь создать ещё свои обработчики, но я опишу наиболее часто используемые события. События можно увидеть на закладке Events объектного инспектора.

Событие	Описание
---------	----------

OnActivate	Когда приложение стало активным
OnCanResize	Это событие генерируется перед тем, как изменить размер окна. Здесь ты можешь запретить какие-либо изменения или производить какие-то подготовительные действия.
OnClick	Генерируется, когда пользователь щёлкнул по форме.
OnClose	Генерируется, когда окно закрывается.
OnCloseQuery	Генерируется до закрытия окна. В этом обработчике происходит запрос на закрытие, поэтому из этого обработчика можно вывести окно, которое будет запрашивать подтверждение на закрытие. Ты такие подтверждения видишь в каждом втором приложении типа «Вы уверены, что хотите закрыть окно?».
OnCreate	Генерируется, когда окно создаётся.
OnDblClick	Генерируется, когда пользователь дважды щёлкнул по окну.
OnDeactivate	Генерируется, когда окно деактивируется.
OnDestroy	Когда окно уничтожается.
OnHide	Генерируется, когда окно исчезает из виду. Событие генерируется даже тогда, когда память, выделенная для окна, не уничтожается.
OnKeyDown	Генерируется, когда нажата клавиша на клавиатуре.
OnKeyPress	Генерируется, когда нажата и отпущена клавиша на клавиатуре.
OnKeyUp	Генерируется, когда отпущена клавиша на клавиатуре.
OnMouseDown	Генерируется, когда нажата кнопка мыши.
OnMouseMove	Генерируется, когда двигается мышка.
OnMouseUp	Генерируется, когда отпускается кнопка мыши.
OnMouseWheel	Генерируется колёсиком мыши.
OnMouseWheelDown	Генерируется, когда колёсико мыши прокручено вниз.
OnMouseWheelUp	Генерируется, когда колёсико мыши прокручено вверх.
OnPaint	Генерируется, когда надо перерисовать окно.
OnResize	Генерируется, когда надо изменить размеры окна.
OnShortCut	Когда нажата горячая клавиша.
OnShow	Когда показывается окно, но до фактической прорисовки. В этот момент окно уже создано и готово к отображению, но ещё не прорисовалось на экране.

Это основные события, которые может генерировать форма. Когда я буду рассматривать компоненты, то я не буду снова расписывать те события, которые уже перечислены здесь, потому что разницы в них нет. Поэтому я буду затрагивать только те события, которые специфичны для конкретного компонента.

6.4 Палитра компонентов.

В этой части мы будем знакомиться с основными компонентами Delphi, которые расположены на палитре компонентов.



Палитра компонентов состоит из нескольких закладок:

1. **Standard.** Все эти компоненты являются аналогами Windows компонентов. Хотя то, что находится в Windows – это не компоненты, это кал :). А вот Borland превратил этот кал в конфетку и сделал его доступным нам.
2. **Additional** – дополнительные компоненты.
3. **Win32** – компоненты, которые есть только в семействе Win32 операционных систем. В это семейство входят Windows 9x, Windows ME, Windows 2000, Windows NT, Windows XP. Наверно легче было сказать, что не входит, потому что это только Windows 3.1.
4. **System** – системные компоненты, с помощью которых облегчается доступ к системе.
5. **Database Access** – компоненты доступа к базам данных.
6. **Data Controls** – компоненты для работы с базами данных.
7. **dbExpress** – ещё компоненты доступа к базам данных, которые пришли на смену BDE.
8. **BDE** – старые компоненты доступа к базам данных.
9. **ADO** – это тоже компоненты для доступа к базам данных, только по технологии Active Data Object (ADO).
10. **InterBase** – компоненты доступа к базе данных InterBase.
11. **WebServices** – компоненты доступа к сети Интернет.
12. **InternetExpress** – компоненты доступа к сети Internet.
13. **FastNet** – Сетевые компоненты. Мы врят ли будем их использовать.
14. **QReport** – компоненты для создания отчётности.
15. **Dialogs** – компоненты облегчающие доступ к стандартным диалогам.
16. **Win3.11** – компоненты доступа к компонентам Win 3.1.
17. **Samples** – различные примеры. Некоторые из этих компонентов доступны в исходных кодах и поставляются вместе с Delphi.

На этом пока хватит. Давай переходить к следующей главе, где мы познакомимся с палитрой компонентов Standard и входящими в неё компонентами. В процессе этого мы будем учиться работать не только с компонентами, но и с самим языком программирования Delphi.