Основное свойство главного меню - это **Items**. Его заполнение происходит в редакторе меню, который вызывается двойным щелчком на компоненте или на многоточии напротив соответствующего пункта в Инспекторе объектов. Для добавления нового пункта меню можно воспользоваться клавишей **Insert**или заполнить свойство Caption у последней нижней ячейки. Удалить ненужный пункт меню можно клавишей **Delete**.

Основные свойства пункта меню:

* Bitmap - рисунок, расположенный слева от заголовка пункта меню. По умолчанию рисунок отсутствует и это свойство равно nil;
* Break - задает разделение на столбцы;
* Caption - задает заголовок меню. символ "-" в качестве заголовка будет отображаться в меню разделительная линия;
* Checked - согласно этому признаку определяется помечен (TRUE) пункт меню или нет (FALSE). Пункт меню, по умолчанию не отмечен;
* AutoCheck - при равном TRUE, автоматически изменяет свойства Checked на противоположное, если выбран данный пункт меню;
* Count - количество подпунктов в данном пункте меню;
* Enabled - свойство, которое блокирует пункт меню. При равном FALSE оно не активно и им нельзя воспользоваться;
* RadioItem - позволяет изменять вид отметки при FALSE  (по умолчанию) - галочка, TRUE - жирная точка;
* ShortCut - определяет горячие клавиши;
* Visible - отвечает за видимость пункта меню.

Основным событием является событие **onClick**. Мы его можем задать при проектировании двойным щелчком на пункте меню.

Компонент **TTimer**

Данный компонент очень очень простой, который не виден на экране, но тем не менее. **Timer** выполняет очень важные функции в программе, например позволяет вводить необходимые задержки между выполнением тех или иных функций. Данный компонент имеет всего четыре свойства и одно событие, и работать с ним очень легко.

Основные свойства компонента **TTimer:**

* Enabled – включение – выключение таймера;
* Interval – интервал срабатывания (в миллисекундах);
* Name – имя компонента в программе;
* Tag – произвольный числовой параметр.

**Form**

Как и любой другой визуальный компонент, форма имеет свойства, методы и события, общие для всех визуальных компонентов.

Наряду с ними у формы есть и спефические свойства, методы и события, определяемые ее особым значением. Некоторые из них характеризуют форму как главный объект приложения, скажем, свойство BorderIcons, другие присущи форме как контейнеру других компонентов, например, свойства AutoScroll и ActiveControl.

Система Delphi при добавлении новой формы в проект автоматически создает один экземпляр класса (Form1, Form2 и т. д.), внося соответствующие изменения в файл проекта, например, добавляя строку кода:

Метод **Сreate**

Для создания экземпляров форм служит метод (конструктор) Сreate. Сам класс формы обычно предварительно описывается при конструировании приложения, и для формы уже существуют файлы формы (dfm) и программного модуля (pas).

**События формы**

При создании и использовании формы генерируются следующие *события* типа TNotifyEvent, указанные в порядке их возникновения:

* OnCreate;
* OnShow;
* OnResize;
* OnActivate;
* OnPaint.

Событие **OnCreate**

Событие OnCreate генерируется только один раз— при создании формы, остальные события происходят при каждом отображении, активизации и каждой прорисовке формы соответственно.

В обработчик события OnCreate обычно включается код, устанавливающий начальные значения свойств формы, а также ее элементов управления, т. е. выполняющий начальную инициализацию формы в дополнение к установленным на этапе разработки приложения параметрам. Кроме того, в обработчик включаются дополнительные операции, которые должны происходить *однократно* при создании формы, например, чтение к из файла некоторой информации и загрузка ее в список.

Свойство **Visible**

Из всех созданных форм Delphi при выполнении приложения автоматически делает видимой главную форму, для этого свойство Visible этой формы устанавливается в значение True. Для остальных форм значение данного свойства по умолчанию равно False, и после запуска приложения они на экране не отображаются. Если формы создаются вручную, то их отображение и скрытие в процессе работы приложения регулируется программистом через свойство Visible. Даже если форма невидима, ее компонентами можно управлять, например, из других форм.

Дочерние формы многодокументного приложения становятся видимыми на экране сразу после их создания.

Методы **Show** и **Hide**

Управлять видимостью форм на экране можно также с помощью методов Show и Hide. Процедура Show отображает форму в немодальном режиме, при этом свойство Visible устанавливается в значение True, а сама форма переводится на передний план. Процедура Hide скрывает форму, устанавливая ее свойство Visible в значение False.

Если окно видимо, то вызов метода Show переводит форму на передний план и передает ей фокус ввода.

В момент отображения формы на экране ее свойство visible принимает значение True, и возникает событиеOnShow. Соответственно при скрытии формы свойство visible принимает значение False, и возбуждается событие OnHide.

События  **OnActivate** и **OnDeActivate**

При получении формой фокуса ввода, например при нажатии кнопки мыши в области формы, происходит ее активизация и возникает событие OnActivate, а при потере фокуса— событие OnDeActivate.

Событие **OnPaint**

Событие OnPaint генерируется при необходимости перерисовки формы, например, при активизации формы, если до этого часть ее была закрыта другими окнами.

Метод **Сlose**

Для закрытия формы используется метод Close, который, если это возможно, удаляет ее с экрана. В случае закрытия главной формы прекращается работа врего приложения.

Процедура Close не уничтожает созданный экземпляр формы, и форма может быть снова вызвана на экран, в частности, с помощью методов Show или ShowModal.

Уничтожение формы (**Release, Free** или **Destroy**)

Уничтожение формы происходит с помощью методов Release, Free или Destroy, после чего работа с этой формой становится невозможна, и любая попытка обратиться к ней или ее компонентам вызовет исключение (ошибку). Необходимость уничтожения формы может возникнуть при оформлении заставок или при разработке больших приложений, требующих экономии оперативной памяти. Предпочтительным методом удаления формы считается метод Free, поскольку он предварительно проверяет возможность удаления.

События при **закрытии и уничтожении формы**

При закрытии и уничтожении формы генерируются следующие события, указанные в порядке их возникновения:

* OnCloseQuery;
* OnClose;
* OnDeActivate;
* OnHide;
* OnDestroy.

Событие **OnCloseQuery**

Событие OnCloseQuery типа TcloseQueryEvent возникает в ответ на попытку закрытия формы. Обработчик события получает логическую переменную-признак CanClose, определяющую, может ли быть закрыта данная форма. По умолчанию эта переменная имеет значение True, и форму можно закрыть. Если установить параметр CanClose в значение False, то форма остается открытой. Такую возможность стоит использовать, например, для подтверждения закрытия окна или проверки, сохранена ли редактируемая информация на диске. Событие OnCloseQuery вызывается всегда, независимо от способа закрытия формы.

Здесь при закрытии формы *Form2* выдается запрос на подтверждение операции, который представляет собой модальное диалоговое окно с текстом и двумя кнопками — *Yes* и *No*. Нажатие кнопки Yes вызывает закрытие формы, при нажатии кнопки No закрытие формы не происходит.

Событие **OnClose**

Событие OnClose типа TCioseEvent возникает непосредственно перед закрытием формы. Обычно оно используется для изменения стандартного поведения формы при закрытии. Для этого обработчику события передается переменная Action типа TCloseAction, которая может принимать следующие значения:

* caNone (форму закрыть нельзя);
* caHide (форма делается невидимой);
* caFree (форма уничтожается, а связанная с ней память освобождается);
* caMinimize (окно формы сворачивается) — значение по умолчанию для MDI-форм.

При закрытии окна методом Close переменная Action по умолчанию получает значение caHide, и форма делается невидимой. При уничтожении формы, например, методом Destroy, переменная Action по умолчанию получает значение caFree, и форма уничтожается.

Событие OnClose возникает при закрытии формы щелчком мыши на кнопке закрытия системного меню или при вызове метода Close. Когда закрывается главная форма приложения, все остальные окна закрываются без вызова события OnClose.

Событие onDestroy типа TNotifyEvent возникает непосредственно перед уничтожением формы и обычно используется для освобождения ресурсов.

Событие **OnResize**

При каждом изменении размеров формы в процессе выполнения приложения возникает событие onResizeтипа TNotifyEvent. В обработчике этого события может размещаться код, например, выполняющий изменение положения и размеров элементов управления окна, не имеющих свойства Align.

При изменении размеров формы Form1 выполняется пересчет параметров, задающих размеры и положение сетки строк, а также положение кнопки.

Cвойство **FormStyle**

Стиль формы определяется свойством FormStyle типа TFormstyle, принимающим следующие значения:

* fsNormal (стандартный стиль, используемый для большинства окон, в том числе и диалоговых);
* fsMDiChild (дочерняя форма в многодокументном приложении);
* fsMDiForm (родительская форма в многодокументном приложении);
* fsStayOnTop (форма, которая после запуска всегда отображается поверх других окон) — обычно используется при выводе системной информации или информационной панели программы.

Форма может изменять стиль динамически — в процессе выполнения программы, например, при выборе пункта меню. При изменении формой стиля возникает событие OnShow.

Cвойство **BorderStyle**

Каждая форма имеет ограничивающую рамку. Вид и поведение рамки определяет свойство BorderStyle типа TFormBorderstyle. Оно может принимать следующие значения:

* bsDialog (диалоговая форма);
* bsSingle (форма с неизменяемыми размерами);
* bsNone (форма не имеет видимой рамки и заголовка и не может изменять свои размеры) — часто используется для заставок;
* bsSizeable (обычная форма с изменяемыми размерами) — по умолчанию, имеет строку заголовка и может содержать любой набор кнопок;
* bsToolwindow (форма панели инструментов);
* bsSizeToolwin (форма панели инструментов с изменяемыми размерами).

Визуальное отличие между диалоговой и обычной формами заключается в том, что диалоговая форма может содержать в своем заголовке только кнопки закрытия и справки. Кроме того, пользователь не может изменять размеры диалоговой формы.

Невозможность изменения размеров форм некоторых стилей относится только к пользователю — нельзя с помощью мыши передвинуть границу формы в ту или иную сторону. Программа при выполнении приложения для формы любого стиля можно устанавливать любые допустимые размеры окна, а также изменять их.

При нажатии кнопки *btnResizeForm* ширина формы Form2 увеличивается на 100 пикселов, даже если ее свойство BorderStyle имеет значение, равное bsDialog, bsSingle или bsNone.

Метод **ShowModal**

Если установить диалоговый стиль формы, то она не становится модальной и позволяет пользователю переходить в другие окна приложения. Для запуска формы, в том числе любой диалоговой, в модальном режиме следует использовать метод ShowModal. Таким образом, стиль определяет внешний вид формы, но не ее поведение.

Cвойство **BorderIcons**

В области заголовка могут отображаться 4 вида кнопок. Реализуемый набор кнопок определяет свойство BorderIcons типа TBorderIcons, которое может принимать комбинации следующих значений:

* blSystemMenu (окно имеет системное меню и может содержать кнопки системного меню);
* blMinimize (окно содержит кнопку свертывания);
* blMaximize (окно содержит кнопку развертывания/восстановления);
* blHelp (окно содержит кнопку справки, которая отображает вопросительный знак и вызывает контекстно-зависимую справку).

Системное меню представляет собой набор общих для всех окон Windows команд, например, *Свернуть* или *Закрыть*. При наличии у окна системного меню в области заголовка слева отображается значок приложения, при щелчке на котором и появляются команды этого меню, а в области заголовка справа имеется кнопка закрытия формы .

Различные значения свойства BorderIcons не являются независимыми друг от друга. Так, если отсутствует системное меню, то ни одна кнопка не отображается. Если имеются кнопки развертывания и свертывания, то не отображается кнопка справки. Возможность появления кнопок также зависит от стиля формы. Например, отображение кнопок развертывания и свертывания возможно только для обычной формы и формы панели инструментов с изменяемыми размерами.

Обычно стиль формы и набор кнопок заголовка задаются на этапе разработки приложения в окне Инспектора объектов. При этом в проектируемой форме всегда видны обычная рамка и три кнопки (развертывания, свертывания и закрытия формы), независимо от значения свойств FormStyle и BorderIcons. Заданные стиль формы и набор кнопок становятся видимыми при выполнении программы.

Cвойства **ClientWidth** и **ClientHeight**

В клиентской области обычно размещаются различные элементы управления, выводится текст или отображается графика. Аналогично тому как свойства Width и Height определяют размеры всей формы, свойства ClientWidth и ClientHeight типа Integer задают ширину и высоту (в пикселах) клиентской части формы.

Обычно форму перетаскивают мышью, указатель которой устанавливается в любом месте области заголовка. При необходимости можно переместить форму и при помещении указателя на ее клиентскую область, для чего требуется описать соответствующие операции программно. Одним из способов является перехват системного сообщения WM\_NCHitTest. Для этого создается процедура FormMove, которая анализирует, в каком месте формы находится указатель мыши при нажатии кнопки. Код местоположения указателя мыши содержится в поле Result системного сообщения типа TMessage. Если значение Result равно 1, что соответствует нажатию кнопки мыши в клиентской области, то полю Result присваивается новое значение, равное 2, имитирующее нахождение указателя мыши в области заголовка. В процедуре FormMove первая инструкция inherited осуществляет вызов предопределенного обработчика перехватываемого события.

Чтобы указать в среде Delphi, что процедура FormMove является обработчиком события WM\_NCHitTest, при ее описании в классе формы TForm1 используется специальный синтаксис, включающий ключевое слово*message*. Как обработчик системного сообщения, процедура содержит один параметр типа TMessage.

Свойство **Menu**

Отображаемое формой меню задается свойством Menu типа TMainMenu. При разработке приложения размещение компонента MainMenu главного меню в форме вызывает автоматическое присвоение значения MainMenu1 свойству Menu. Это самый простой способ ссылки на главное меню. Если в ходе выполнения приложения какая-либо форма должна отображать различные меню, то через свойство Menu можно указать другое главное меню, например, следующим образом: Form1.Menu := MainMenu2;

Свойство **Icon**

Каждая форма отображает в левой стороне области заголовка свой значок, определяемый свойством Iconтипа Ticon. Если форма не является главной в приложении, то этот значок отображается при свертывании формы. Для любой формы свойство Icon можно задать с помощью *Инспектора объектов* или динамически (при выполнении приложения). Если значок не задан, то форма использует значок, указанный в свойстве Iconобъекта Application. Последний выводится также при свертывании и отображении в панели задач Windows значка главной формы приложения.

Свойство **Position**

Размещение и размер формы при отображении определяет свойство Position типа TPosition. Оно может принимать значения, перечисленные далее:

* poDesigned (форма отображается в той позиции и с теми размерами, которые были установлены при ее конструировании) — значение по умолчанию. Положение и размеры формы определяются свойствами Left, Tор, Width и Height. Если приложение запускается на мониторе с более низким разрешением, чем у того, на котором оно разрабатывалось, часть формы может выйти за пределы экрана;
* poScreenCenter (форма выводится в центре экрана, ее высота и ширина— свойства Height и width— не изменяются);
* poDefault (Windows автоматически определяет начальную позицию и размеры формы) — при этом значении программист не имеет возможности управлять этими параметрами, поэтому оно не допускается для форм многодокументных приложений;
* poDefaultPosOnly (Windows определяет начальную позицию формы, ее размеры не изменяются);
* poDefaultSizeOnly (Windows определяет начальные ширину и высоту формы и помещает форму в позицию, определенную при разработке);
* PoDesktopCenter (форма выводится в центре экрана, ее высота и ширина не изменяются);
* PoMainFormCenter (форма выводится в центре главной формы приложения, ее высота и ширина не изменяются) — это значение используется для вторичных форм, при применении его для главной формы оно действует как значение poScreenCenter;
* PoOwnerFormCenter (форма выводится в центре формы, которая является ее владельцем, высота и ширина формы не изменяются) — если для формы не указан владелец (свойство Owner), то данное значение аналогично значению poMainFormCenter.

Приложение может запоминать расположение и размеры форм и при последующем выполнении правильно отображать формы на экране. Для этого программист должен записать соответствующие данные в инициализационный файл приложения или в системный реестр Windows, а при последующем выполнении приложения считать эти данные и установить их для форм.

Свойство **Active**

Свойство Active типа Boolean позволяет определить активность формы. В любой момент времени активной может быть только одна форма, при этом ее заголовок выделяется особым цветом (обычно синим). Если свойство Active имеет значение True, то форма активна (находится в фокусе ввода), если False — то неактивна. Это свойство доступно для чтения во время выполнения программы. Если требуется активизировать форму программно, следует использовать свойство WindowState или метод Show(showModal).

Cвойство **ChildActiveForm**

В многодокументном приложении родительское окно не может быть активным независимо от цвета заголовка. Для определения активного дочернего окна многодокументного приложения служит свойство ChildActiveFormтипа TForm родительской формы.

Свойство **WindowState**

Свойство WindowState типа TWindowstate определяет состояние отображения формы и может принимать одно из трех значений:

* wsNormal (обычное состояние) — по умолчанию;
* wsMinimized (свернута);
* wsMaximized (развернута).

Кнопки btnMiniForm и btnNormalForm в форме Form1 сворачивают и восстанавливают обычное состояние формы Form2 соответственно.

Форма, для которой изменяется состояние отображения на экране, предварительно должна быть создана методами CreateForm или Create. Если форма не создана, то при обращении к ней будет сгенерировано исключение, несмотря на то, что переменная формы объявлена в модуле. Если форма создана, но не отображается на экране, то изменения ее состояния (свойства windowstate) происходят, однако пользователь не видит этого до тех пор, пока форма не будет отображена на экране.

Свойство **ActiveControl**

Будучи контейнером, форма содержит другие элементы управления. Оконные элементы управления (потомки класса TWinControl) могут получать фокус ввода. Свойство ActiveControl типа TWinControl определяет, какой элемент формы находится в фокусе. Для выбора элемента, находящегося в фокусе ввода (активного элемента), можно устанавливать это свойство в нужное значение при выполнении программы:

В случае, когда размеры окна недостаточны для отображения всех содержащихся в форме интерфейсных компонентов, у формы могут появляться полосы прокрутки. Свойство AutoScroll типа Boolean определяет, появляются ли они автоматически, Если свойство AutoScroll имеет значение True (по умолчанию), то полосы прокрутки появляются и исчезают автоматически, без каких-либо действий программиста. Необходимость в полосах прокрутки может возникнуть, например, в случае, если пользователь уменьшит размеры формы так, что не все элементы управления будут полностью видны. Если же свойство AutoScroll установлено в значение False, то программист реализует управление просмотром информации вручную через свойства HorzScrollBar(горизонтальная прокрутка) и VertScrollBar (вертикальная прокрутка) типа TControlScrollBar формы.

Метод **ScrollInView**

Для программного управления полосами прокрутки можно использовать метод ScrollInView. Процедура ScrollInView (AControl: TControl) автоматически изменяет позиции полос прокрутки так, чтобы заданный параметром AControl элемент управления стал виден в отображаемой области.

Свойство **KeyPreview**

Свойство KeyPreview типа Boolean определяет, будет ли форма обрабатывать события клавиатуры, прежде чем их обработают элементы управления формы. Если свойство имеет значение False (по умолчанию), то клавиатурные события поступают к активному элементу управления (имеющему фокус ввода). При установке свойства KeyPreview в значение True форма первой получает сообщения о нажатии клавиш и может на них реагировать, что обычно используется для обработки комбинаций клавиш, независимо от активности элементов управления формы.

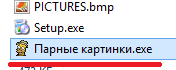
Cвойство **MDichildCount** и метод **Cascade**

У формы имеется ряд свойств и методов, например свойство MDichildCount и метод Cascade, предназначенных для организации многодокументных приложений.



# 3.3 Описание интерфейса для пользователя

Для запуска приложения необходимо запустить файл Парные картинки.exe



**Рисунок 3.3.1 - Файл для запуска игры**

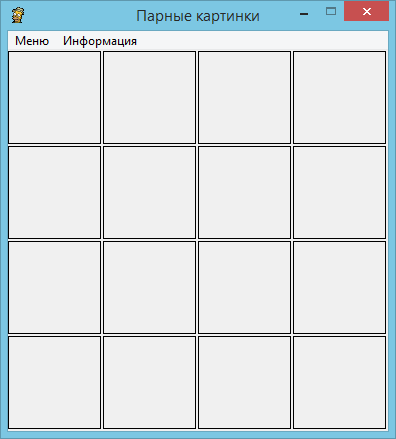
После запуска программы появится диалоговое окно с игровой областью, в котором вы сразу можете начать наслаждаться приятным интерфейсом и веселыми картинками.

Меню программы – в нём находится два выбора:

1. Новая игра – при запуске запустится игра;
2. Выйти – игровое приложение закроется.

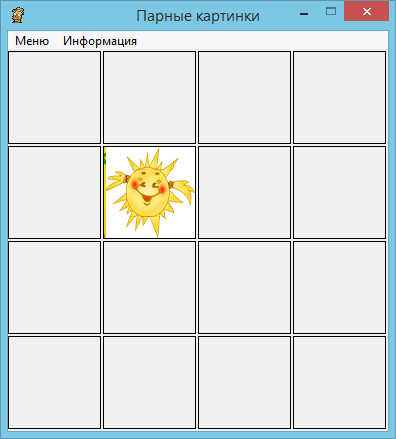
При запуске игры открывается форма с приложением.

После запуска проекта, перед вами появится главное окно в котором можно сразу начать играть.

****

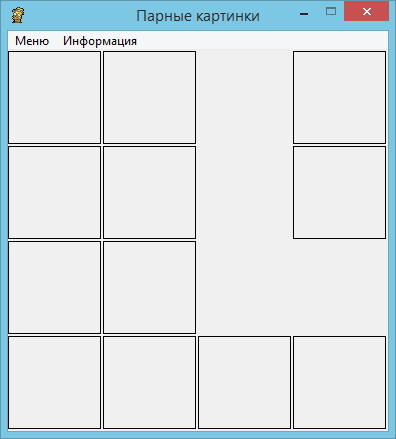
**Рисунок 3.3.2 – Главное окно проекта**

После выбора одной из клеточек, перед вами появится красочная картинка, к которой нужно найти пару.



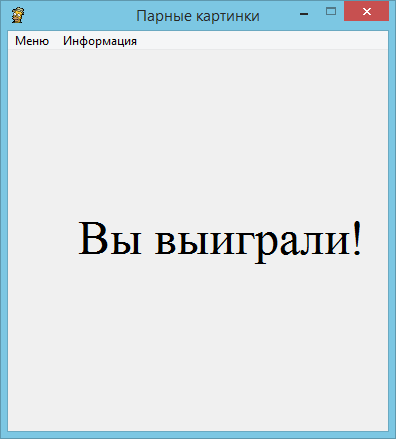
**Рисунок 3.3.3 – Начало игры**

Для продолжения игры следует выбрать вторую клетку и найти такую же картинку. Если вы все сделали правильно, то данные две картинки исчезнут.



**Рисунок 3.3.4 – Пример пропадания клеток**

После полного прохождения игры, будет показано окно с поздравлением.

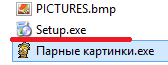


**Рисунок 3.3.5 – Итог прохождения игры**

# 3.4 Инсталляция и настройка проекта

Установка программы

Для установки программы достаточно запустить файл Setup.exe и начать установку приложения.



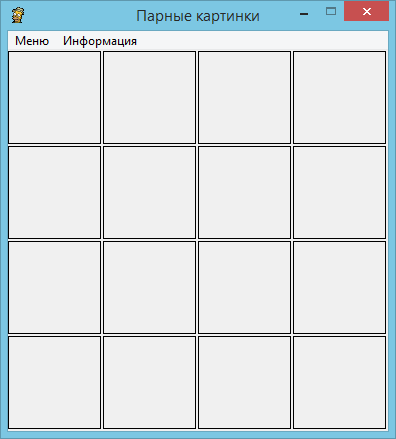
**Рисунок 3.4.1 – Файл для установки игры**

Удалить программу

Щелкнуть один раз левой кнопкой мыши на папку «Игра». Нажать клавишу «Delete». Появится окно подтверждения удаления программы, после чего нажать клавишу «Enter».

Открытие программы

После установки навести курсор мыши на ярлык «Парные картинки.exe», щелкнуть два раза левой кнопкой мыши. Откроется стартовое окно программы (рисунок 3.4.1).



**Рисунок 3.4.1 - Стартовое окно программы**

Форма позволяет перейти к другим окнам программы.

Возможно открыть следующие окна:

* Нажмите левой кнопкой мыши «Меню» для начала новой игры либо выхода;

нажмите левой кнопкой мыши «Информация» для открытия формы с информацией о создателе.