Желание творить что-либо собственными руками - это, на мой взгляд, один из тех не самых существенных в больших масштабах, но весьма серьёзных в масштабах одной конкретной личности двигателей прогресса, который дремлет внутри каждого человека. То есть, конечно, дремлет он не у всех - у очень многих он активно работает.

Конечно, проявляться желание творить может самыми разнообразными способами, и, на мой взгляд, программирование - хороший способ самовыражения. Поэтому сейчас я хочу рассказать тем из наших читателей, у кого есть творческая жилка программиста, о том, как создать собственную экранную заставку, известную также под именами хранителя экрана и скринсейвера.

Поэтому сначала немного о том, что собой скринсейвер представляет. Для пользователя это заставка, которая появляется на экране, когда компьютер некоторое время находится без дела. С точки зрения программиста, это самая обычная программа в виде *EXE-файла*
, правда, имеющего несколько нестандартное расширение -
.SCR
. Само собой, создать скринсейвер можно с помощью любой среды программирования для Windows; мы же с вами воспользуемся Delphi.

Уметь делать скринсейвер должен не так уж и много. Во-первых, отображать себя непосредственно на экране плюс в окне предварительного просмотра, во-вторых, конечно же, вовремя прятаться, а в-третьих, показывать окно конфигурации. Вот это-то мы с вами сейчас и реализуем.

Каким образом скринсейвер при запуске узнаёт о том, что ему сейчас показывать - саму заставку, предварительный просмотр заставки или её настройки? Очень просто: система передаёт ему через командную строку управляющий параметр.

Если никаких параметров нет, то показывается окно настроек, если параметр /**p**, то показывать следует предпросмотр, ну а если
/**s**
, то следует запускать саму заставку в полноэкранном режиме. Как видите, нужно рисовать один и тот же рисунок два раза: и на панели предварительного просмотра, и в

полноэкранном режиме.

Поэтому логично вынести рисование в отдельную процедуру. Её вы сможете изменить самостоятельно, используя как стандартные рисовальные инструменты Windows API и Delphi, так и специализированные библиотеки **OpenGL** и **DirectX**. У меня же скринсейвер просто выводит на экран заданную фотографию или рисунок. Вот код процедуры, которая это реализует:

```
procedure Draw(Canvas: TCanvas; FileName: string);  var   picture: TPicture;  begin
picture := TPicture.Create;   picture.LoadFromFile(FileName);   Canvas.Draw(0, 0,
picture.Graphic);   picture.Free;  end;
```

В общем-то, здесь всё, на мой взгляд, очевидно, но я поясню. Сначала объявляем переменную типа **TPicture**, которую через строку инициализируем. После этого загружаем картинку из указанного файла с помощью метода **LoadFromFile** класса **TPicture**, а затем отображаем его на канве (какую именно канву - окна предпросмотра или же основного окна, мы будем использовать, станет известно позже, поэтому она у нас идёт как параметр этой процедуры).

Далее удаляем уже не нужный экземпляр класса **TPicture**. Почему используется **TPicture**, а не **TPBitmap**? Дело в том, что первый класс - более общий, он позволяет работать с разными графическими форматами, а не только BMP. Для расширения их количества можно, например, воспользоваться замечательной библиотекой **GraphicEx**, которая находится по адресу www.soft-gems.net.

Самая сложная процедура во всей нашей программе-скринсейвере - это, безусловно, отображение "превьюшки", т.е. показ нашего скринсейвера в окне предварительного просмотра. Именно ею мы сейчас и займёмся. Но сначала нужны некоторые предварительные приготовления. Нужно объявить в интерфейсном разделе модуля с главной формой приложения следующие переменные:

```
Cnvs: TCanvas;  FileToShow: string;  PreviewWindow: hWnd;  Rect: TRect;
```

Пока я не буду на них подробно останавливаться, поскольку их смысл станет вполне

ясен из кода процедуры, отображающей "превьюшку". Кроме указанных выше переменных, нам нужна оконная процедура.

Дело в том, что предварительный просмотр мы будем делать таким образом: будем получать от системы координаты и другие параметры области на окне настроек экрана, где нужно отобразить превью, а затем строить там своё собственное окошко, синхронизированное средствами Windows API с главным окном. Ну и уже на этом нашем маленьком окошке рисовать с помощью процедуры **Draw**. Так что вот он, текст оконной процедуры того самого окна:

```
function ScreenSaver_WndProc(Wnd: hWnd; Msg: integer; wParam, lParam: longint): integer; stdcall; begin case Msg of WM_DESTROY: begin PostQuitMessage(0); Result := 0; end; WM_PAINT: begin Draw(Cnvs, FileToShow); Result := DefWindowProc(Wnd, Msg, wParam, lParam); end; else Result := DefWindowProc(Wnd, Msg, wParam, lParam); end; end;
```

Это просто обработчик системных событий: если система скажет уничтожить окно, то оно уничтожится, если скажет перерисовать - что ж, перерисуем. Теперь можно написать и, собственно, процедуру отрисовки превьюшки:

```
procedure Preview; var Parent: hWnd; WndClass: TWndClass; Msg: TMsg; begin Parent := StrToInt(ParamStr(2)); with WndClass do begin Style := CS_PARENTDC; lpfnWndProc := addr(ScreenSaver_WndProc); cbClsExtra := 0; cbWndExtra := 0; hIcon := 0; hCursor := 0; hbrBackground := 0; lpszMenuName := nil; lpszClassName := 'ScreenSaverClass01'#0; end; WndClass.hInstance := hInstance; Windows.RegisterClass(WndClass); GetWindowRect(Parent, rect); PreviewWindow := CreateWindow('ScreenSaverClass01'#0, 'ScreenSaver01', WS_CHILDWINDOW or WS_VISIBLE or WS_BORDER, 0, 0, rect.Right - rect.Left, rect.Bottom - rect.Top, Parent, 0, hInstance, nil); Cnvs := TCanvas.Create; Cnvs.Handle := GetDC(PreviewWindow); repeat if PeekMessage(Msg, 0, 0, 0, PM_REMOVE) then begin if Msg.message = WM_QUIT then Break; TranslateMessage(Msg); DispatchMessage(Msg); end else Draw(Cnvs, FileToShow); until false; ReleaseDC(PreviewWindow, Cnvs.Handle); Cnvs.Destroy; end;
```

Что ж, теперь давайте разбираться. Сначала объявляются необходимые переменные - дескриптор родительского (главного) окна, класс нашего дочернего окна, а также переменная для обработки сообщений. Сначала мы вытягиваем из параметров командной строки дескриптор родительского окна, после чего инициализируем класс своего. Как видите, почти все параметры устанавливаются в ноль - значит, наше окно не будет иметь ни иконки, ни меню, только имя класса **ScreenSaverClass01**, тоже оканчивающееся нулём.

Далее регистрируем класс окна с помощью процедуры **RegisterClass**, а затем - получаем размеры области, которую мы должны заполнить своим окном. Сделав это, создаём дочернее окно, потом создаём канву, на которой будем рисовать (это просто VCL-обёртка над контекстом окна "превьюшки"), а затем идёт стандартный для всех Windows-приложений цикл обработки сообщений. Как видите, он может оборваться, только если того захочет сама система - а её это заставит захотеть пользователь. После прерывания псевдобесконечного цикла мы "убираем мусор" - контекст окна и канву.

Теперь будет идти процедура реакции главного окна на разные системные сообщения. Именно она отвечает за закрытие скринсейвера при разных действиях пользователя. Удобнее реализовать её так, чем вешать процедуру на все обработчики событий нашей формы. Положите на форму компонент **ApplicationEvents** с закладки *Additional*. Дважды кликните на событии
onMessage
в инспекторе объектов и заполните процедуру следующим образом:

```
    case Msg.message of   WM_KEYDOWN, WM_KEYUP,   WM_SYSKEYDOWN,
WM_SYSKEYUP,   WM_LBUTTONDOWN, WM_RBUTTONDOWN,   WM_MBUTTONDOWN,
WM_MOUSEMOVE: Close;  end;
```

Теперь нужно добавить ещё обработчик события onIdle того же компонента. Он будет содержать всего две строчки:

```
    Draw(Canvas, FileToShow);  Done := false;
```

Как видите, здесь мы просто рисуем - как раз в те моменты, когда система ничем не занята. Что ж, теперь готово практически всё. Остаётся только поменять некоторые свойства главной формы и написать ещё немного кода. Свойства меняем такие: цвет (**Color**) на чёрный (*clBlack*), **BorderStyle** ставим в *bsNone*, а **WindowState** в *wsMaximized*.

Делается это всё с помощью инспектора объектов. Далее в меню *Project* выбираем *View*

Source
, объявляем переменную с типа char, добавляем в раздел
uses
модуль
Dialogs
, стираем всё между
begin
и
end
и пишем следующие строки:

```
    if (ParamCount >= 1) and        (Length(ParamStr(1)) > 1) then   c :=
UpCase(ParamStr(1)[2])  else   c := #0;   case c of   'P': Preview;   'S': begin
Application.Initialize;    Application.CreateForm(TForm1, Form1);    Application.Run;    end;
else begin    Application.Initialize;    ShowMessage('This is a very cool screensaver!');
Application.Run;   end;  end;
```

Вот теперь скринсейвер готов.

Внимательные читатели, должно быть, заметили, что вместо нормального диалога настроек будет отображаться окошко-сообщение с надписью "This is a very cool screensaver!". Это, как вы сами понимаете, можно доработать. Ещё можно доработать перемещение изображения по экрану с течением времени, добавить проверку, не был ли скринсейвер уже запущен (хотя, в принципе, для хранителей экрана такая проверка не критична, поскольку ситуация одновременного запуска нескольких экземпляров маловероятна и не несёт никаких разрушительных для системы последствий).

Ещё - заранее предупреждаю - в нём есть один баг. Какой? Дело в том, что сообщение
WM_MOUSEMOVE
может и не означать действительного перемещения курсора мыши... Убрать этот баг очень легко: достаточно отслеживать координаты курсора до и после прихода этого сообщения. Почему я сам не написал этот код? Чтобы у вас было больше практики в устранении багов. Думаю, эта статья заинтересует, в основном, программистов начинающих, а для них более полезную вещь, чем практика, вряд ли можно придумать.