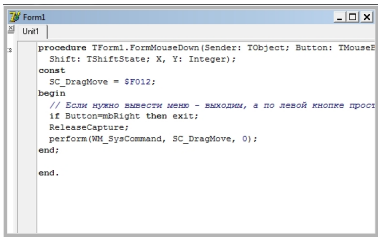
Приведу простой пример, как можно использовать данную форму. Допустим Вам надо сделать программу- напоминалку. Висит эта форма и на ней постоянно отображаются Ваши встречи, дела, праздники или другая полезная информация. Получится этакий *Acti ve Desktop*
:) Но будет он жрать ресурсов на порядки меньше.Реализуется все это очень просто.




SetWindowRgn(Handle, R, True);

**Handle** - указатель на форму, вид которой хотим поменять

**R** - указатель на регион, для установки R смотрите функцию **CreatePolygonRgn**

**True** - флаг, при значении TRUE сразу после установки перерисовки

## Прозрачная форма

Переписываем конструктор:

constructor TForm1.Create(AOwner: TComponent); begin inherited; HorzScrollBar.Visible:= False; // убираем сколлбары, чтобы не мешались VertScrollBar.Visible:= False; // при изменении размеров формы RebuildWindowRgn; // строим новый регион end;

А вот процедура "перестройки" региона формы:

procedure TForm1.RebuildWindowRgn; var FullRgn, Rgn: THandle; ClientX, ClientY, I: Integer; begin // определяем относительные координаты клиентской части ClientX:= (Width - ClientWidth) div 2; ClientY:= Height - ClientHeight - ClientX; // создаем регион для всей формы FullRgn:= CreateRectRgn(0, 0, Width, Height); // создаем регион для клиентской части формы и вычитаем его из FullRgn Rgn:= CreateRectRgn(ClientX, ClientY, ClientX + ClientWidth, ClientY + ClientHeight); CombineRgn(FullRgn, FullRgn, Rgn, rgn_Diff); // теперь добавляем к FullRgn регионы каждого контрольного элемента for I:= 0 to ControlCount - 1 do with Controls[I] do begin Rgn:=

CreateRectRgn(ClientX + Left, ClientY + Top, ClientX + Left +Width, ClientY + Top + Height); CombineRgn(FullRgn, FullRgn, Rgn, rgn_Or); end; SetWindowRgn(Handle, FullRgn, True); // устанавливаем новый регион окна end;

Еще одно. Если Ваша форма, будет с изменяемыми размерами, то Вам надо добавить:

procedure TForm1.Resize; begin inherited; RebuildWindowRgn; // строим новый регион end;

## Перемещение формы

И самый последний штрих - произвольное перемещение формы, а не за *Title Bar*. Так сделано в программе WinAmp. Пишем всего одну процедуру:

procedure TForm1.WMNCHitTest(var M: TWMNCHitTest); begin inherited; // вызов унаследованного обработчика if M.Result = htClient then // Мышь сидит на окне? Если да M.Result := htCaption; // - то пусть Windows думает, что мышь на caption bar end;

Если Вам понадобится вызов Popup меню, то можно поступить следующим образом:

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); const SC_DragMove = $F012; begin // Если нужно вывести меню - выходим, а по левой кнопке просто перетаскиваем. if Button=mbRight then exit; ReleaseCapture; perform(WM_SysCommand, SC_DragMove, 0); end;

Смотрится достаточно эффектно, но единственный минус скорость работы. Надеюсь, что этот раздел будет постепенно пополняться.

Да, хочется отметить что начиная с Windows 2000 на аппаратном уровне появилась поддержка прозрачных окон. Точнее сказать прозрачность поддерживается за счет использования нового графического интерфейса - GDI+. Этот интерфейс значительно расширяет функции стандартного GDI, но наданный момент все его функции раелизуются программно. Этот минус MS устранит в следующих версиях.