Попросили меня тут сделать эффект расходящихся на воде кругов отдельно (и с использованием TImage), ну и что бы побыстрее было, чем в демке ASFDemoPool . Вот, собстно, и результат. Кому интересно – качаем.

Скачать модуль WaterCircle.pas - 5,38 КБ (5 514 байт)

Скачать демо WaterCircle.zip - 1,50 МБ (1 578 903 байт)

{slide=Посмотреть исходный код класса|grey|closed}

```
//---------------------------------------------------------------------------- // // Расходящиеся по воде круги // Автор класса Andy BitOff : bitoff(a)pisem.net // // Сделано на основе кода // _http://www.delphifr.com/codes/EFFET-EAU-AVEC-GETDIBITS-SETDIBITS_42445.aspx // Автор barbichette // _http://www.delphifr.com/auteur/BARBICHETTE/14124.aspx // //---------------------------------------------------------------------------- unit WaterCircle; interface uses Windows, Graphics; const maxcailloux = 5; type ttab = array [0 .. 0] of Integer; ptab = ^ttab; TPoscailloux = array [1 .. maxcailloux] of TRect; TWaterCircle = class private FPuissance: Integer; BitInfo: TBitmapInfo; im: ptab; Cailloux: ptab; Poscailloux: TPoscailloux; cwi, che: Integer; cpt: byte; wi, he: Integer; Step: Integer; mx, my, mc: Integer; Count: Integer; procedure SetPuissance(const Value: Integer); public constructor Create(Bitmap: TBitmap); destructor Destroy; override; procedure DrawImage(Bitmap: TBitmap); procedure Drop(X, Y: Integer); // интенсивность кругов property Puissance: Integer read FPuissance write SetPuissance; end; implementation const tablecailloux: TPoscailloux = ((Left: 000; Top: 0; Right: 058; Bottom: 76), (Left: 059; Top: 0; Right: 140; Bottom: 64), (Left: 141; Top: 0; Right: 210; Bottom: 69), (Left: 211; Top: 0; Right: 289; Bottom: 76), (Left: 290; Top: 0; Right: 340; Bottom: 106)); constructor TWaterCircle.Create(Bitmap: TBitmap); var bit: tbitmap; begin Puissance := 500; cpt := 0; Count := 0; Step := 0; mc := 0; Poscailloux := tablecailloux; bit := tbitmap.create; bit.Assign(Bitmap); bit.PixelFormat := pf32bit; getmem(cailloux, bit.Width * bit.Height * 4); cwi := bit.Width; che := bit.Height; with bitinfo.bmiHeader do begin biSize := sizeof(bitinfo.bmiHeader); biWidth := bit.Width; biHeight := -bit.Height; biPlanes := 1; biBitCount := 32; biCompression := BI_RGB; biSizeImage := 0; biXPelsPerMeter := 1; biYPelsPerMeter := 1; biClrUsed := 0; biClrImportant := 0; end; GetDIBits(bit.Canvas.Handle, bit.Handle, 0, bit.Height, cailloux, bitinfo, DIB_RGB_COLORS); wi := Bitmap.Width; he := Bitmap.Height; getmem(im, wi * he * 4 * 4); with bitinfo.bmiHeader do begin biWidth := wi; biHeight := -he; biSizeImage := 0; end; GetDIBits(Bitmap.Canvas.Handle, Bitmap.Handle, 0, he, im, bitinfo, DIB_RGB_COLORS); fillchar(im[wi * he * 2], wi * he * 4 * 2, 0); end; destructor TWaterCircle.Destroy; begin freemem(im); freemem(Cailloux); end; procedure TWaterCircle.DrawImage(Bitmap: TBitmap); var cp, sp: Integer; i, j: Integer; wn: Integer; nw: dword; dx, dy: Integer; function getpix(X, Y: Integer): Integer; var tx, ty, k: Integer; begin result := im[X + Y * wi]; ; for k := 1 to maxcailloux do if (X >= poscailloux[k].Left) and (Y >= poscailloux[k].Top) and (X <= poscailloux[k].Right) and (Y <= poscailloux[k].Bottom) then begin tx := X - poscailloux[k].Left + tablecailloux[k].Left; ty := Y - poscailloux[k].Top + tablecailloux[k].Top; if cailloux[tx + cwi * ty] $FF00FF then result := cailloux[tx + cwi * ty]; end; end; begin Inc(count); cp := wi * he * 2 + cpt * wi * he; sp := wi * he * 2 + (1 - cpt) * wi * he; for i := 1 to he - 2 do begin wn := i * wi; for j := 1 to wi - 2 do begin Inc(wn); nw := ((im[cp + wn - wi - 1] + im[cp + wn - wi] + im[cp + wn - wi + 1] + im[cp + wn - 1] + im[cp + wn + 1] + im[cp + wn + wi - 1] + im [cp + wn + wi] + im[cp + wn + wi + 1]) shr 2) - im[sp + wn]; im[sp + wn] := nw; if im[sp + wn] = wi then dx := wi - 1; if dx = he then dy := he - 1; if dy < 0 then dy := 0; im[wi * he + j + i * wi] := getpix(dx, dy); end; end; SetDIBits(Bitmap.Canvas.Handle, Bitmap.Handle, 0, he, @(im[wi * he]), bitinfo, DIB_RGB_COLORS); cpt := 1 - cpt; end; procedure TWaterCircle.Drop(X, Y: Integer); begin if x < 1 then exit; if y < 1 then exit; if x > wi - 2 then exit; if y > he - 2 then Exit; im[wi * he * 2 + wi * Y + X] := puissance; im[wi * he * 2 + wi * Y + X + 1] := puissance shr 1;
```

```
im[wi * he * 2 + wi * Y + X - 1] := puissance shr 1;    im[wi * he * 2 + wi * Y + X + wi] := puissance shr 1;    im[wi * he * 2 + wi * Y + X - wi] := puissance shr 1;    if mc = 0 then      exit;  mx := X - mx;    my := Y - my;    poscailloux[mc].Left := poscailloux[mc].Left + mx;  poscailloux[mc].Right := poscailloux[mc].Right + mx;    poscailloux[mc].Top := poscailloux[mc].Top + my;    poscailloux[mc].Bottom := poscailloux[mc].Bottom + my;    mx := X;    my := Y;  end;    procedure TWaterCircle.SetPuissance(const Value: Integer);  begin    if (Value > 200) and (Value < 5000) then      FPuissance := Value;  end;    end.
```

{slide=Посмотреть исходный код модуля демонстрации|grey|closed}

```
unit WaterCircleDemoMain;    interface    uses    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,    Dialogs, ExtCtrls, StdCtrls, WaterCircle;    const    DropDelay = 1; // не может быть 0    type    TForm1 = class(TForm)      Image1: TImage;      Timer1: TTimer;      procedure Timer1Timer(Sender: TObject);      procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);      procedure FormCreate(Sender: TObject);      procedure FormClose(Sender: TObject; var Action: TCloseAction);      procedure Image1MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);    private      WaterCircle: TWaterCircle;      Step: Integer;    public    end;    var    Form1: TForm1;    implementation    {$R *.dfm}      procedure TForm1.FormCreate(Sender: TObject);  begin    DoubleBuffered := True;    WaterCircle := TWaterCircle.Create(Image1.Picture.Bitmap);    WaterCircle.Puissance := 800;    Step := 0;    Timer1.Enabled := true;  end;    procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);  begin    Timer1.Enabled := false;    WaterCircle.Free;  end;    procedure TForm1.Timer1Timer(Sender: TObject);  begin    Inc(Step);    if (Step mod DropDelay) = 0 then      WaterCircle.Drop(Random(Image1.Picture.Bitmap.Width), Random(Image1.Picture.Bitmap.Height));    WaterCircle.DrawImage(Image1.Picture.Bitmap);    Image1.Refresh;  end;    procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;    Shift: TShiftState; X, Y: Integer);  begin    WaterCircle.Drop(X, Y);  end;    procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState;    X, Y: Integer);  begin    WaterCircle.Drop(X, Y);  end;    end.
```

{/slides}

{jcomments on}