How to read image pixels fast

Ever needed to modify/read a bitmap? Using *TBitamp.Canvas.Pixels* is works just fine, but when you try it with a larger bitmap you'll notice the problem.

TBitmap.Canvas.Pixels provides an easy way to access pixels but it's very slow. A way to get around this is to use *ScanLine()* to work on the bitmap. What do we get back? Every timem you change *TBitmap.Canvas.Pixels* the bitmap is redrawn. This of course consumes resources and valuable time. *ScanLine()* leaves this decision up to you, the programmer. There is a little catch here, though. When using *Canvas.Pixels[]* you get in return a *TColor* value, but when using *ScanLine()* you get back the bytes of the line. What does this mean? Well, you have to worry about what is Red, Green and Blue. With *Canvas.Pixels[]* you could've used *GetRValue()*, *GetBValue()* or *GetGValue()*, but with scan line you have to remember: each pixel is made of 3 values: Red, Green and Blue. You have to make sure you are reading the correct value.

Look at this code:

```
var
  iX  : Integer;
  Line: PByteArray;
...
  Line := Image1.ScanLine[0]; // We are scanning the first line
  iX := 0;
  // We can't use the 'for' loop because iX could not be modified from
  // within the loop
  repeat
    Line[iX]     := Line[iX] - $F; // Red value
    Line[iX + 1] := Line[iX] - $F; // Green value
    Line[iX + 2] := Line[iX] - $F; // Blue value
    Inc(iX, 3); // Move to next pixel
  until iX > (Image1.Width - 1) * 3;
```

As you can see you each color is now made of 3 values, so if you miscalculate the next trio (may happen) you will be editing/reading garbage. Not only will you mix up Red, Green and Blue but you will be reading values from two neighbouring pixels.

Here's code that show how to reads the Red and Blue values and switched them.

```
var
  btTemp: Byte; // Used to swap colors
  iY, iX: Integer;
  Line  : PByteArray;
...
  for iY := 0 to Image1.Height - 1 do begin
    Line := Image1.ScanLine[iY]; // Read the current line
    repeat
      btSwap       := Line[iX];     // Save red value
      Line[iX]     := Line[iX + 2]; // Switch red with blue
      Line[iX + 2] := btSwap;       // Switch blue with previously saved red
      // Line[iX + 1] - Green value, not used in example
      Inc(iX, 3);
    until iX > (Image1.Width - 1) * 3;
  end;
  Image1.Invalidate; // Redraw bitmap after everything's done
```

The example is quite simple yet it should explain the usage of this function.

Note: This code only works with 24 bit pixel format bitmaps. You should ensure the bitmap's *PixelFormat* property is set to *pf24bit*.

Author:	Unknown
Added:	2007-06-02
Last updated:	2011-01-16