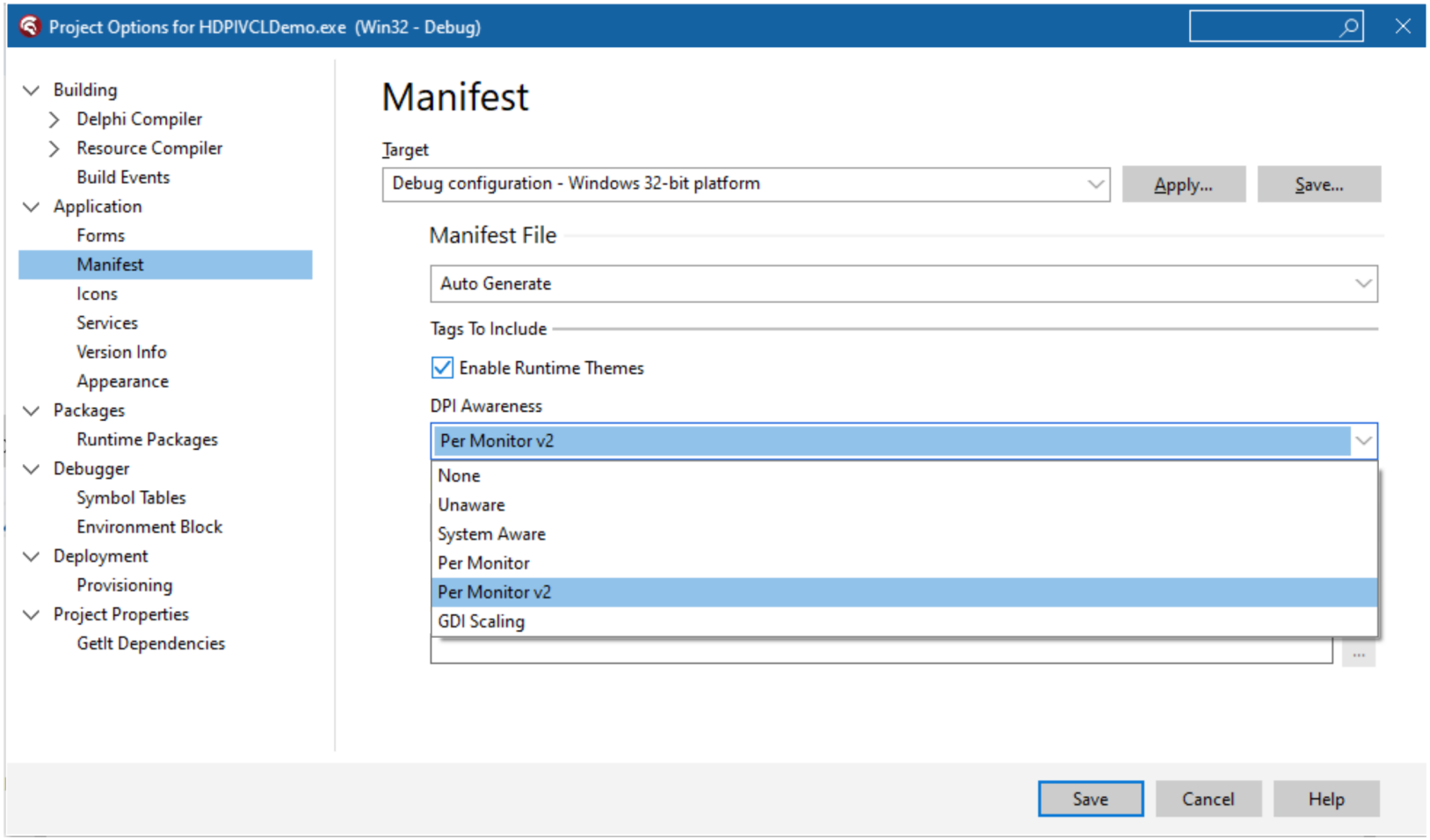
Responsywne interfejsy użytkownika dla ekranów wysokiej rozdzielczości

Zdzisław Sroczyński
Caprisoft

Zlot programistów Delphi 2022

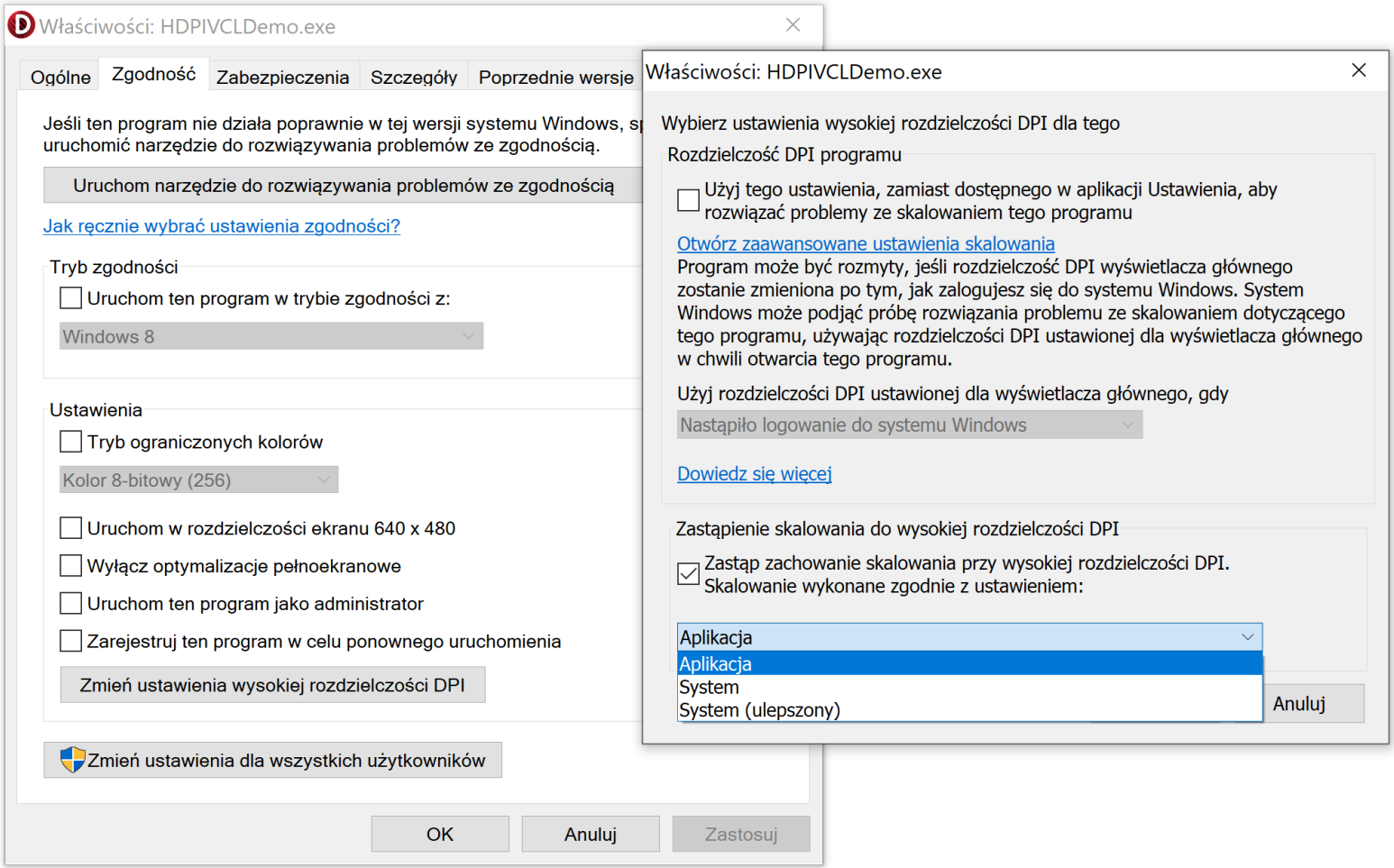
Zamiast wstępu (1)

HDPI (scaled!) opcje projektu/manifest



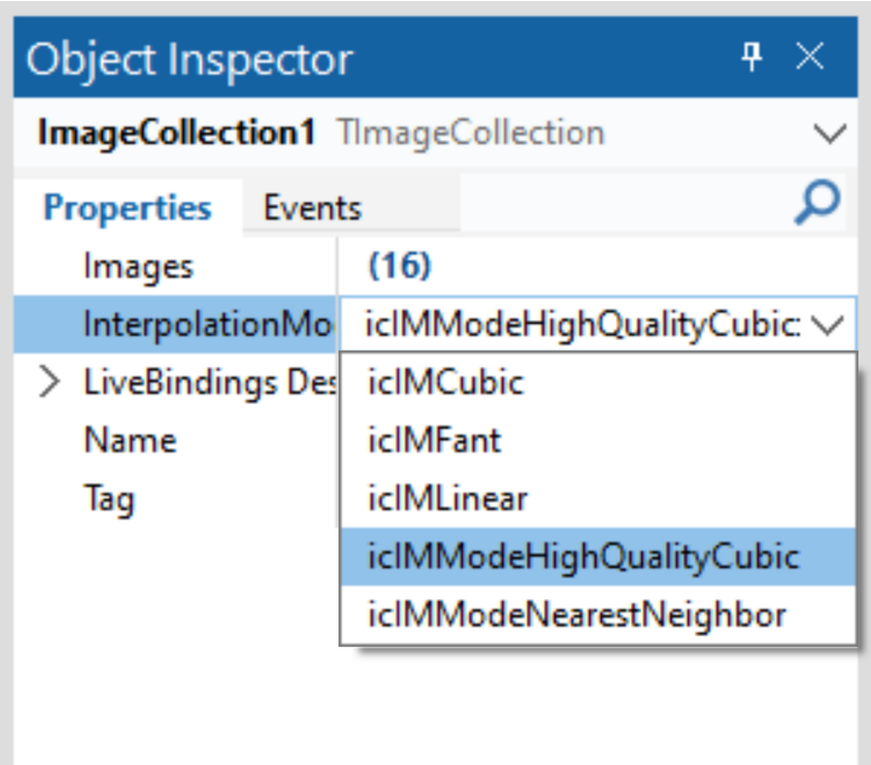
Zamiast wstępu (2)

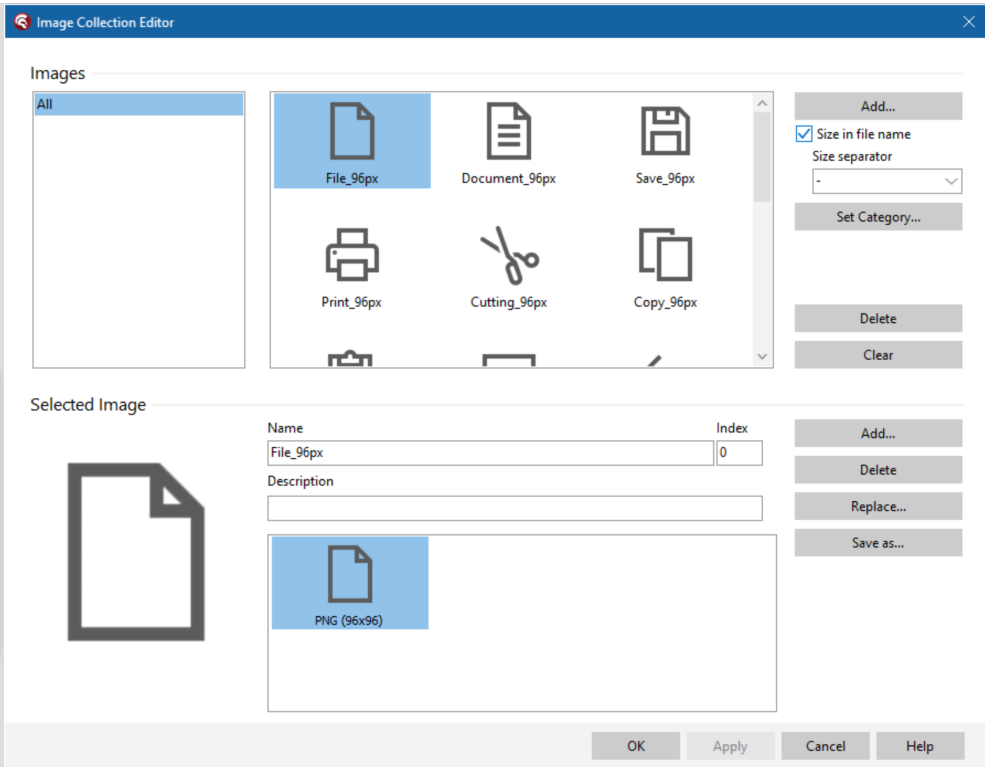
Windows 10 – system ulepszony=GDI scaling (powolne)



TImageCollection

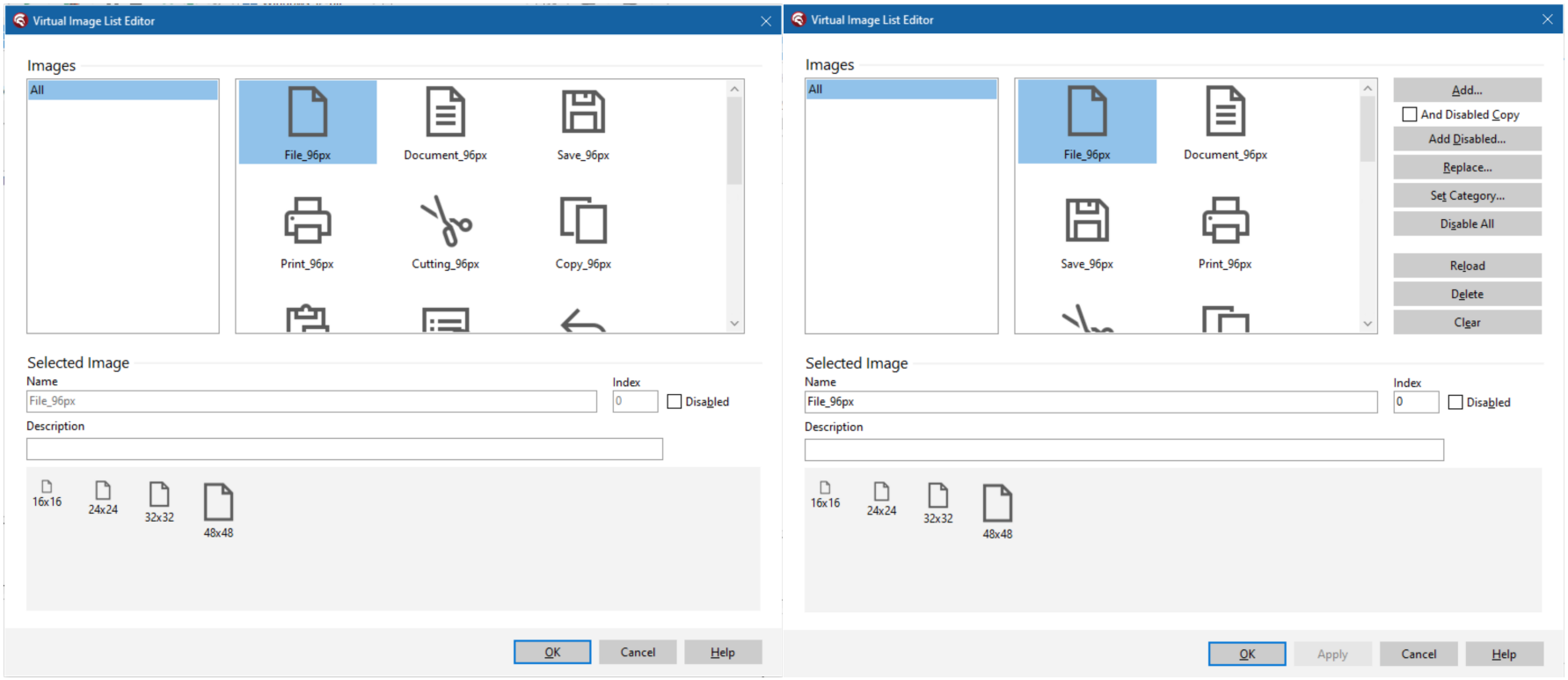
zestaw obrazów źródłowych





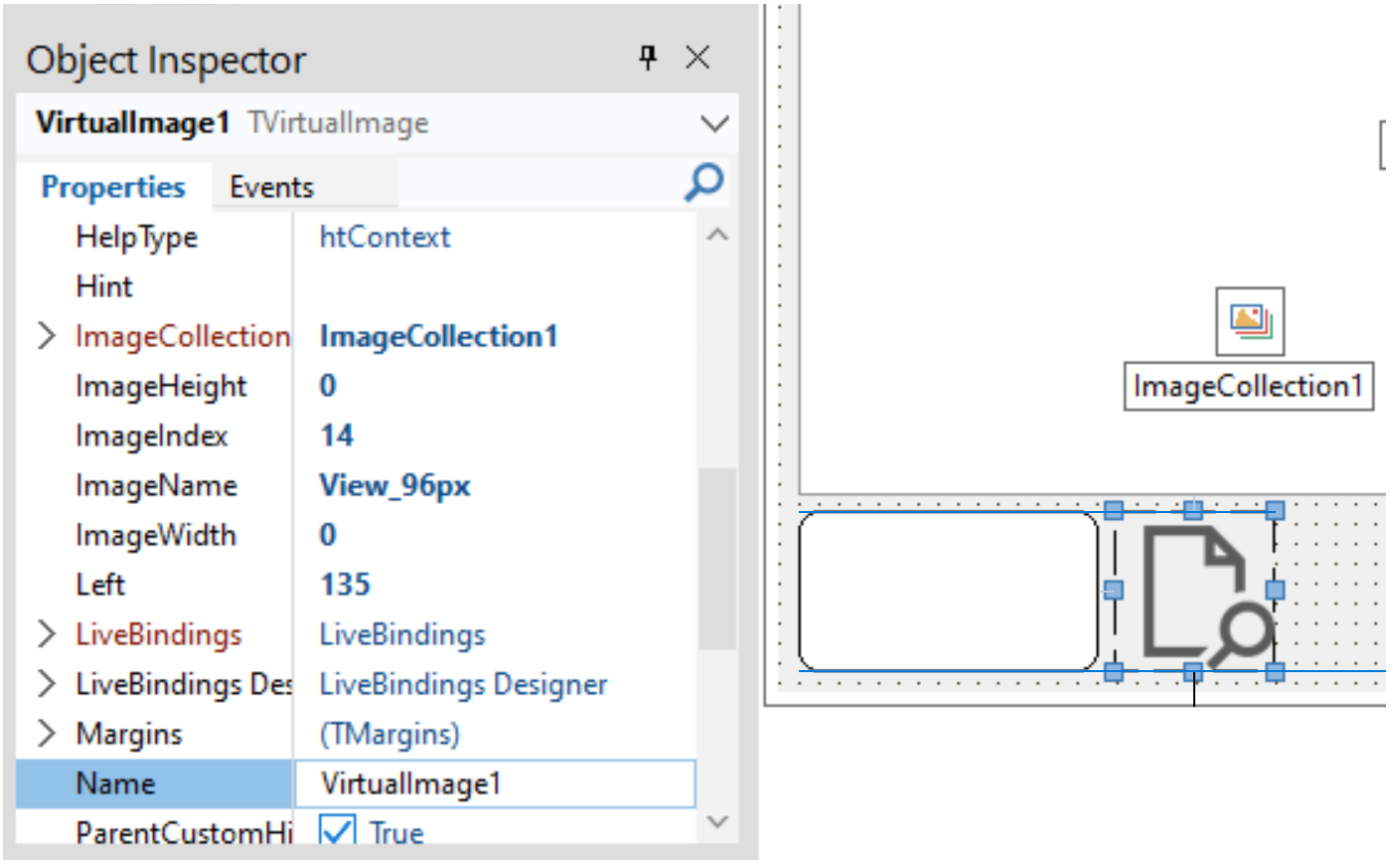
TVirtualImageList

kompatybilna z TImageList



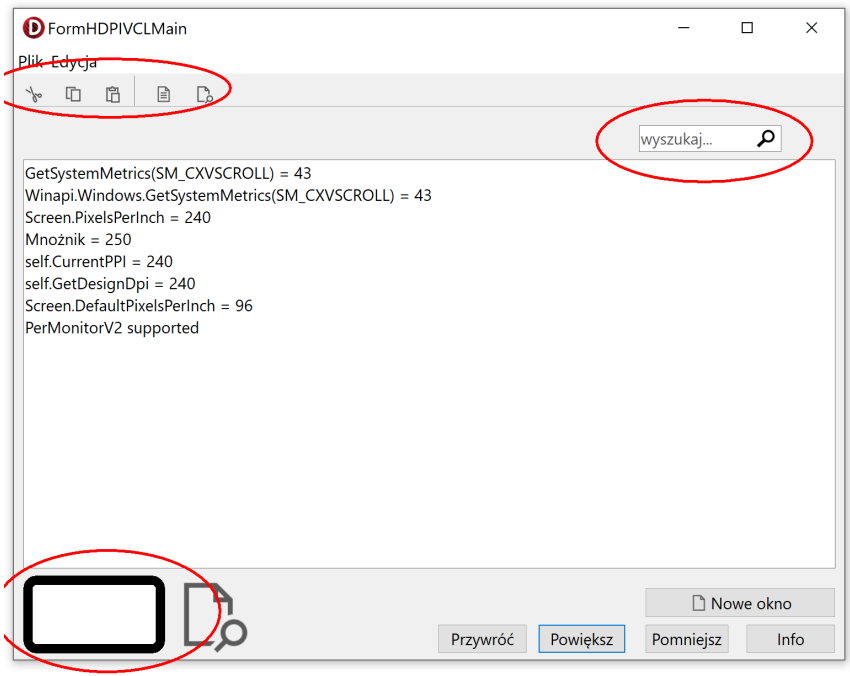
TVirtualImage

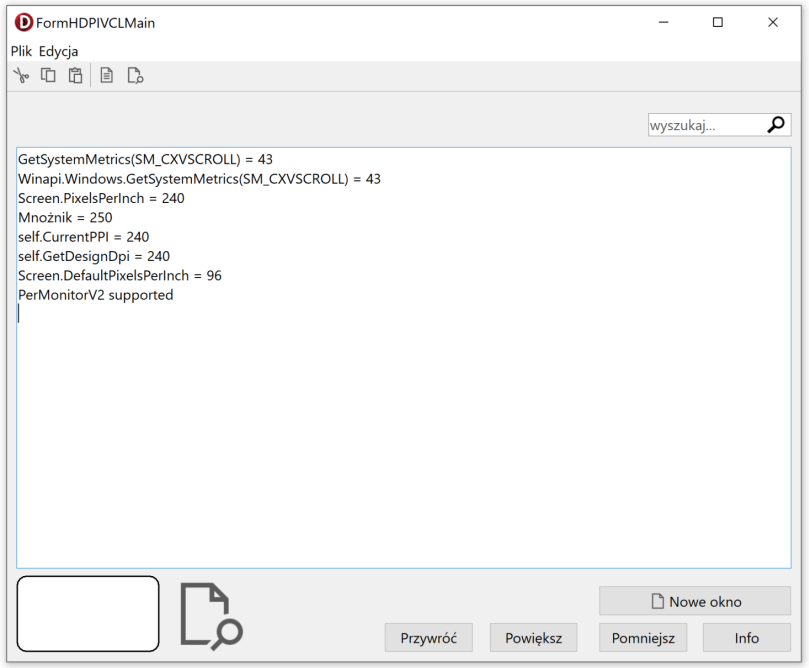
ImageHeight równe zero oznacza dopasowanie



Skalowanie komponentów VCL

- `self.ScaleBy(120,100);`
- liczy się proporcja liczb
- w VCL wartości całkowite – błędy zakrągleń!





Przywracanie formatki z zasobów dfm

```
procedure TFormHDPIVCLMain.UMReloadForm(var msg: TMessage);
var
  i: Integer;
  rs: TResourceStream;
begin
  {Block form redrawing}
  Perform(WM_SETREDRAW, 0, 0);
  try
    {Delete all components on the form}
    for i := ComponentCount - 1 downto 0 do
      Components[i].Free;
    {Find the forms resource}
    rs := TResourceStream.Create(FindClassHInstance(TFormHDPIVCLMain),
                                 Classname, RT_RCDATA);
    try
      rs.ReadComponent(self);
    finally
      rs.free
    end;
  finally
    {Redisplay form}
    self.ScaleBy(MulDiv(100, self.CurrentPPI, Screen.DefaultPixelsPerInch),100);
    Perform(WM_SETREDRAW, 1, 0);
    Invalidate;
    self.Left := (self.Monitor.Width  - self.Width)  div 2;
    self.Top  := (self.Monitor.Height - self.Height) div 2;
  end;
end;
```

Włączanie HDPI dla wybranych formatek

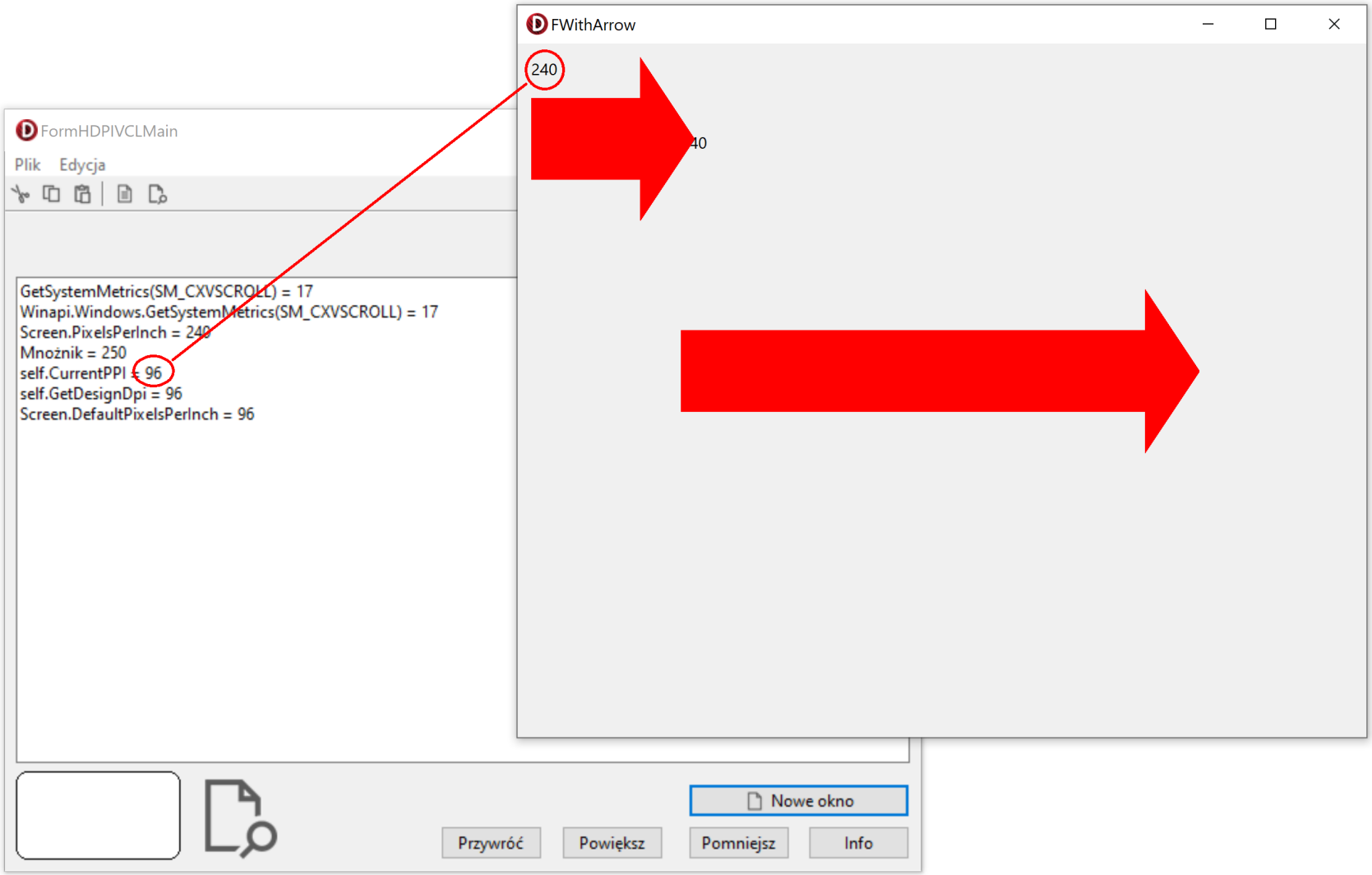
```
begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  SetThreadDpiAwarenessContext(DPI_AWARENESS_CONTEXT_UNAWARE);
  Application.CreateForm(TFormHDPIVCLMain, FormHDPIVCLMain);
  SetThreadDpiAwarenessContext(DPI_AWARENESS_CONTEXT_PER_MONITOR_AWARE_V2);
  Application.CreateForm(TFWithArrow, FWithArrow);
  SetThreadDpiAwarenessContext(DPI_AWARENESS_CONTEXT_UNAWARE);
  Application.Run;
end.
```

weryfikacja najważniejszych parametrów związanych z HDPI:

```
AMemo.Lines.Add('GetSystemMetrics(SM_CXVSCROLL) = ' + GetSystemMetrics(SM_CXVSCROLL).ToString);
AMemo.Lines.Add('Winapi.Windows.GetSystemMetrics(SM_CXVSCROLL) = ' + Winapi.Windows.GetSystemMetrics(SM_CXVSCROLL).ToString);
AMemo.Lines.Add('Screen.PixelsPerInch = '+Screen.PixelsPerInch.ToString);
AMemo.Lines.Add('Mnożnik = ' + MulDiv(100, Screen.PixelsPerInch, Screen.DefaultPixelsPerInch).ToString);
AMemo.Lines.Add('self.CurrentPPI = ' + self.CurrentPPI.ToString);
AMemo.Lines.Add('self.GetDesignDpi = ' + self.GetDesignDpi.ToString);
AMemo.Lines.Add('Screen.DefaultPixelsPerInch = ' + Screen.DefaultPixelsPerInch.ToString);

if CheckPerMonitorV2SupportForWindow(self.Handle) then
 AMemo.Lines.Add('PerMonitorV2 supported');
```

Różne DPI w formatkach – przykład



Uwzględnianie skali (1)

pozycjonowanie i rozmiary nie bezwzględnie – MulDiv, Align, Anchors

```
procedure TFWithArrow.PaintBox1Paint(Sender: TObject);
begin
 with PaintBox1.Canvas do
 begin
   TextOut(0,0,PaintBox1.CurrentPPI.ToString);
   Brush.Style := bsSolid;
   Brush.Color := clRed;
   Pen.Color := clRed;
   var rightm := (PaintBox1.Width) - MulDiv(40, CurrentPPI, Screen.DefaultPixelsPerInch {96});
   FillRect(Rect(MulDiv(0, CurrentPPI, Screen.DefaultPixelsPerInch {96}),
     (PaintBox1.Height div 2)-MulDiv(30, CurrentPPI, Screen.DefaultPixelsPerInch {96}),
     rightm,
     (PaintBox1.Height div 2) + MulDiv(30, CurrentPPI, Screen.DefaultPixelsPerInch {96})));
   Polygon( [Point(rightm,(PaintBox1.Height div 2)-MulDiv(60, CurrentPPI, Screen.DefaultPixelsPerInch {96})),
      Point((PaintBox1.Width),(PaintBox1.Height div 2)),
      Point(rightm,(PaintBox1.Height div 2)+MulDiv(60, CurrentPPI, Screen.DefaultPixelsPerInch {96}))]);
 end;

end;
```

Uwzględnianie skali (2)

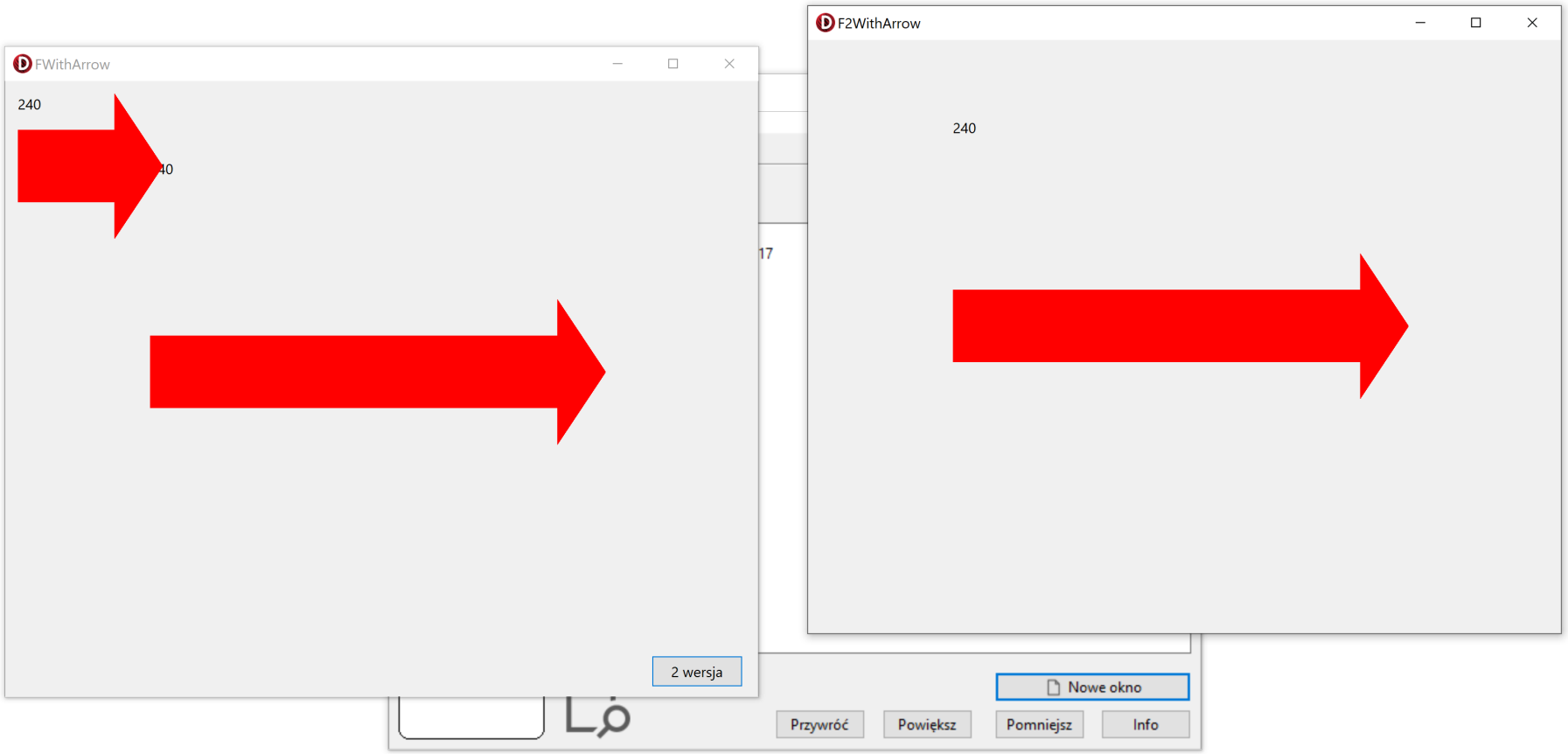
OnAfterMonitorDpiChanged, OnBeforeMonitorDpiChanged

```
function TF2WithArrow.ComputeDrawScale : double;
begin
  result := CurrentPPI/Screen.DefaultPixelsPerInch;
end;

procedure TF2WithArrow.FormBeforeMonitorDpiChanged(Sender: TObject; OldDPI,
  NewDPI: Integer);
begin
  DrawScale := ComputeDrawScale;
end;

procedure TF2WithArrow.PaintBox1Paint(Sender: TObject);
begin
with PaintBox1.Canvas do
 begin
   TextOut(0,0,PaintBox1.CurrentPPI.ToString);
   Brush.Style := bsSolid;
   Brush.Color := clRed;
   Pen.Color := clRed;
   var rightm := (PaintBox1.Width) - Round(40 * DrawScale);
   FillRect(Rect(0,
     (PaintBox1.Height div 2)-Round(30 * DrawScale),
     rightm,
     (PaintBox1.Height div 2) + Round(30 * DrawScale)));
   Polygon( [Point(rightm,(PaintBox1.Height div 2)-Round(60 * DrawScale)),
      Point((PaintBox1.Width),(PaintBox1.Height div 2)), Point(rightm,(PaintBox1.Height div 2)+Round(60 * DrawScale)
 end;
end;
```

Uwzględnianie skali – przykład



Uwzględnianie skali w komponencie

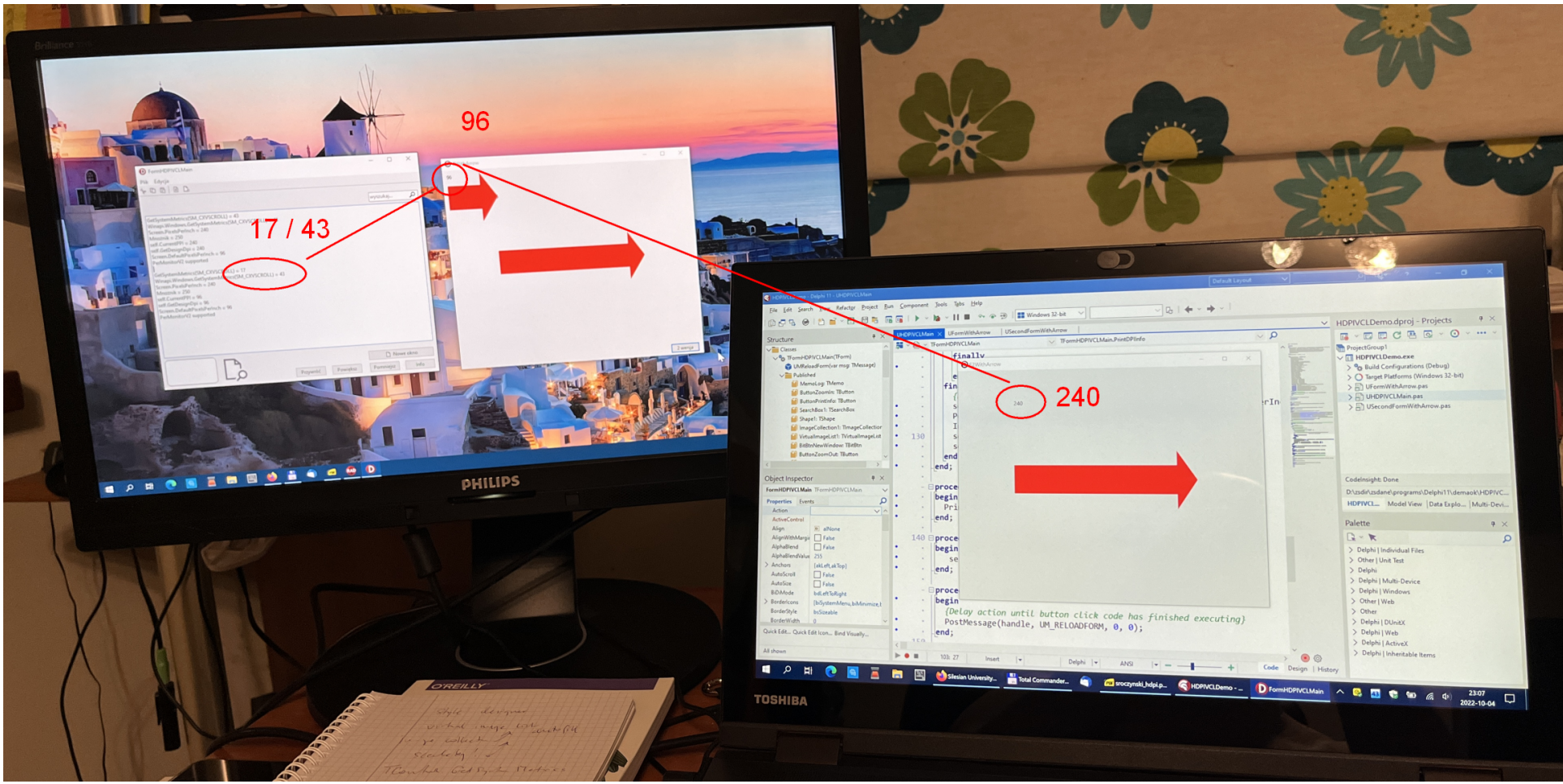
TControl.CurrentPPI – PPI dla komponentu

```
type
  TArrowShape = class( TPaintBox )
    procedure Paint;override;
  end;

procedure TArrowShape.Paint;
begin
  inherited;
  with self.Canvas do
 begin
   TextOut(0,0,self.CurrentPPI.ToString);
   Brush.Style := bsSolid;
   Brush.Color := clRed;
   Pen.Color := clRed;
   var rightm := (self.Width) - MulDiv(40, CurrentPPI, Screen.DefaultPixelsPerInch {96});
   FillRect(Rect(MulDiv(0, CurrentPPI, Screen.DefaultPixelsPerInch {96}),
     (self.Height div 2)-MulDiv(30, CurrentPPI, Screen.DefaultPixelsPerInch {96}),
     rightm,
     (self.Height div 2) + MulDiv(30, CurrentPPI, Screen.DefaultPixelsPerInch {96})));
   Polygon( [Point(rightm,(self.Height div 2)-MulDiv(60, CurrentPPI, Screen.DefaultPixelsPerInch {96})),
      Point((self.Width),(self.Height div 2)),
      Point(rightm,(self.Height div 2)+MulDiv(60, CurrentPPI, Screen.DefaultPixelsPerInch {96}))]);
 end;
end;
```

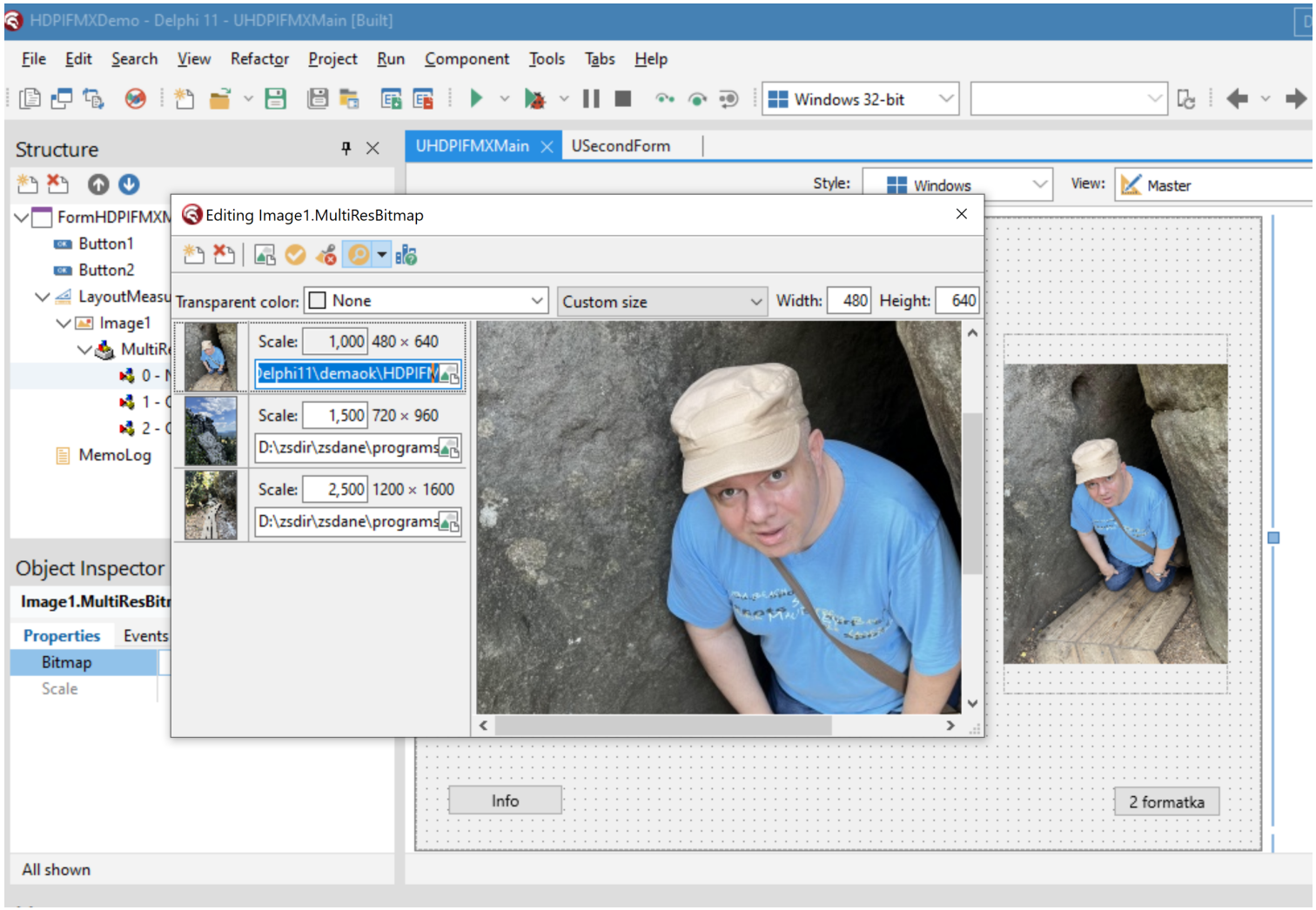
DPI awareness per monitor v2 (VCL)

`Winapi.Windows.GetSystemMetrics vs TControl.GetSystemMetrics`



FMX – wymiary typu double

`TMultiResBitmap`

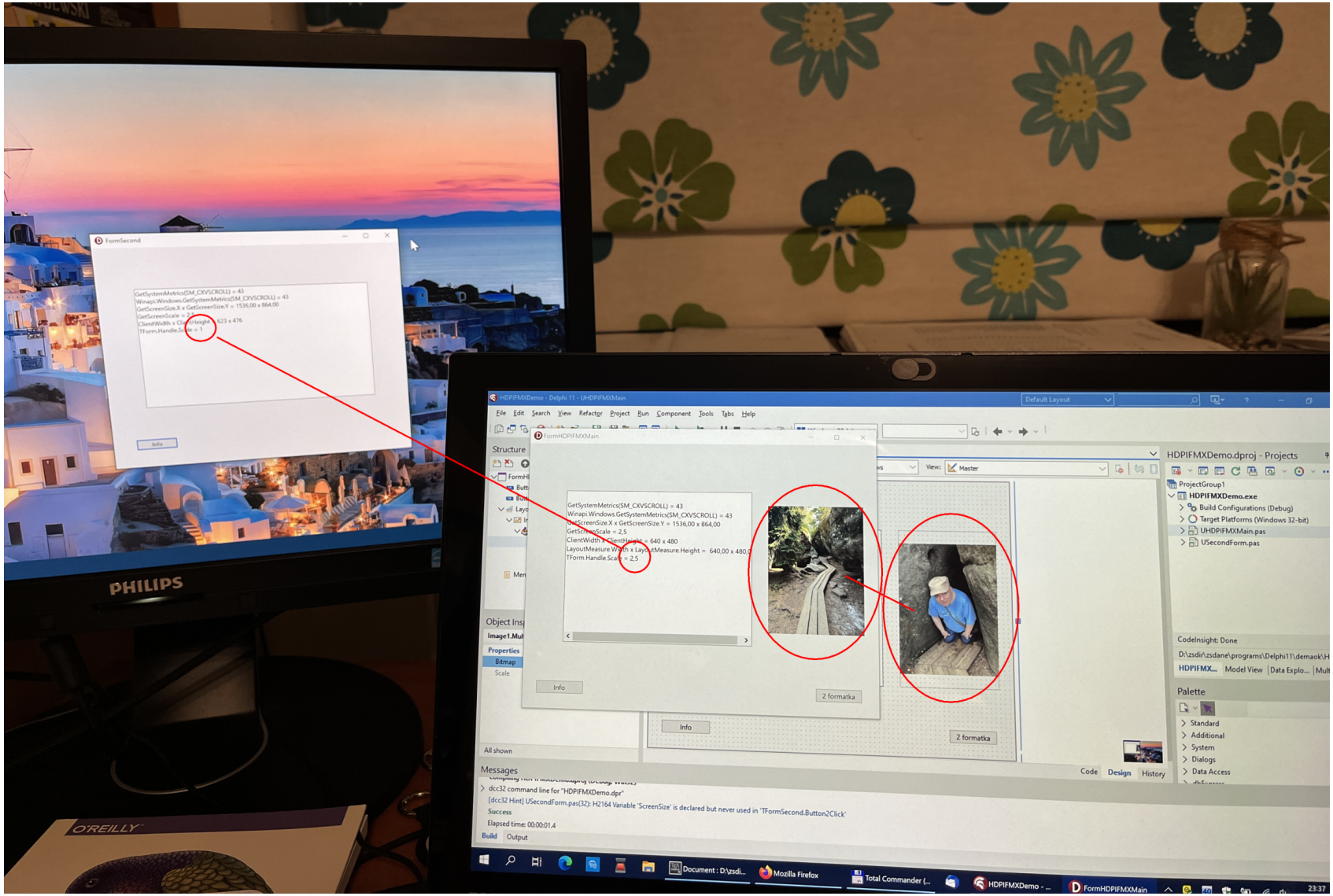


Odczyt parametrów w FMX

najważniejsze parametry związane z HDPI:

```
var
  ScreenSvc: IFMXScreenService;
  ScreenSize: TSize;
begin
  MemoLog.Lines.Add('GetSystemMetrics(SM_CXVSCROLL) = ' + GetSystemMetrics(SM_CXVSCROLL).ToString);
  MemoLog.Lines.Add('Winapi.Windows.GetSystemMetrics(SM_CXVSCROLL) = ' +
                                          Winapi.Windows.GetSystemMetrics(SM_CXVSCROLL).ToString);
  if TPlatformServices.Current.
     SupportsPlatformService(IFMXScreenService,IInterface(ScreenSvc))
  then
  begin
    MemoLog.Lines.Add(Format('GetScreenSize.X x GetScreenSize.Y = %f x %f', [ScreenSvc.GetScreenSize.X,
        ScreenSvc.GetScreenSize.Y]));
    MemoLog.Lines.Add('GetScreenScale = '+ScreenSvc.GetScreenScale.ToString);
  end;
  MemoLog.Lines.Add(Format('ClientWidth x ClientHeight = %d x %d',
                                                  [FormSecond.ClientWidth,FormSecond.ClientHeight]));
  MemoLog.Lines.Add('TForm.Handle.Scale = '+FormSecond.Handle.Scale.ToString);
end;
```

DPI awareness per monitor v2 (FMX)



Kody źródłowe przykładowych aplikacji:
https://github.com/zdzichs/ZlotDelphi2022

Dziękuję za uwagę

pytania: zdzislaw@sroczynski.pl