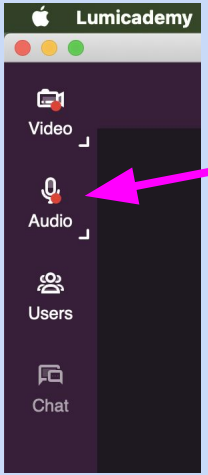
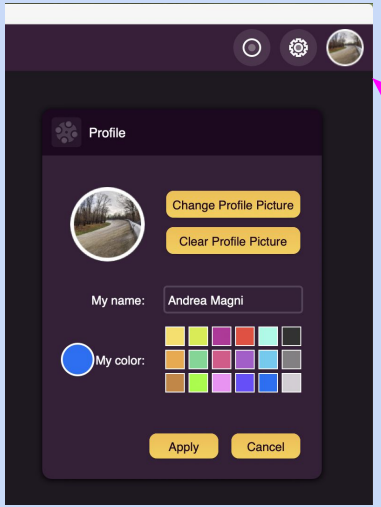
- La piattaforma permette ai partecipanti di condividere la vostra webcam e il microfono
  - Per cortesia, aprite il microfono solo per fare domande
- Per accedere alla chat Slack:
  - https://andreamagni.eu/live.html



Pallino rosso = spento, click per attivare (vale per Video e Audio)



Potete impostare il vostro nome utente, se volete

Delphi Live - Italia
n.7 - 29 Aprile 2020

Andrea Magni
me@andreamagni.eu

Andrea Magni

Ingegnere informatico, Monza

Sviluppatore, Formatore, Consulente

Desktop, Server, Web, Mobile e IoT

embarcadero MVP Delphi

TFrameStand, TFormStand, FMXER (FMX)

MARS-Curiosity (REST Library)

FMX Book (coming)

https://andreamagni.eu

Modernizzazione applicazioni Windows (VCL) con supporto High DPI

Carlo Barazzetta
carlo.barazzetta@ethea.it

Carlo Barazzetta

Ethea s.r.l. Carugate MI

Ethea: ISF / InstantObjects / Kitto / Kitto^2

BasketScouting / SportClubManager

IconFontsImageList / VCLThemeSelector

http://www.ethea.it

Agenda

- Perché è necessario modernizzare le app VCL?
- Come IconFontsImageList ci aiuta a risolvere diversi problemi
- Tema chiaro/scuro, va di moda: (usiamo il TVCLThemeSelector)
- Il supporto High-DPI: come impostare una app per gestire DPI diversi
- Come funzionano i Font (Application.DefaultFont e Screen.IconFont)
- I componenti per Windows 10: panoramica e fix
- High-DPI gestire e personalizzare i Font
- High-DPI gli eventi da conoscere a cui “rispondere”
- Resize in base al DPI o alla dimensione dello schermo: 2 logiche diverse

Materiale on-line di questa sessione:

- IconFontsImageList:
  - https://github.com/EtheaDev/IconFontsImageList
- VCLThemeSelector e Demo High-DPI:
  - https://github.com/EtheaDev/VCLThemeSelector
    - Form “elegante” per selezionare il tema della app
    - Demo completa con i concetti di modernizzazione e supporto High-DPI trattati in questa sessione.

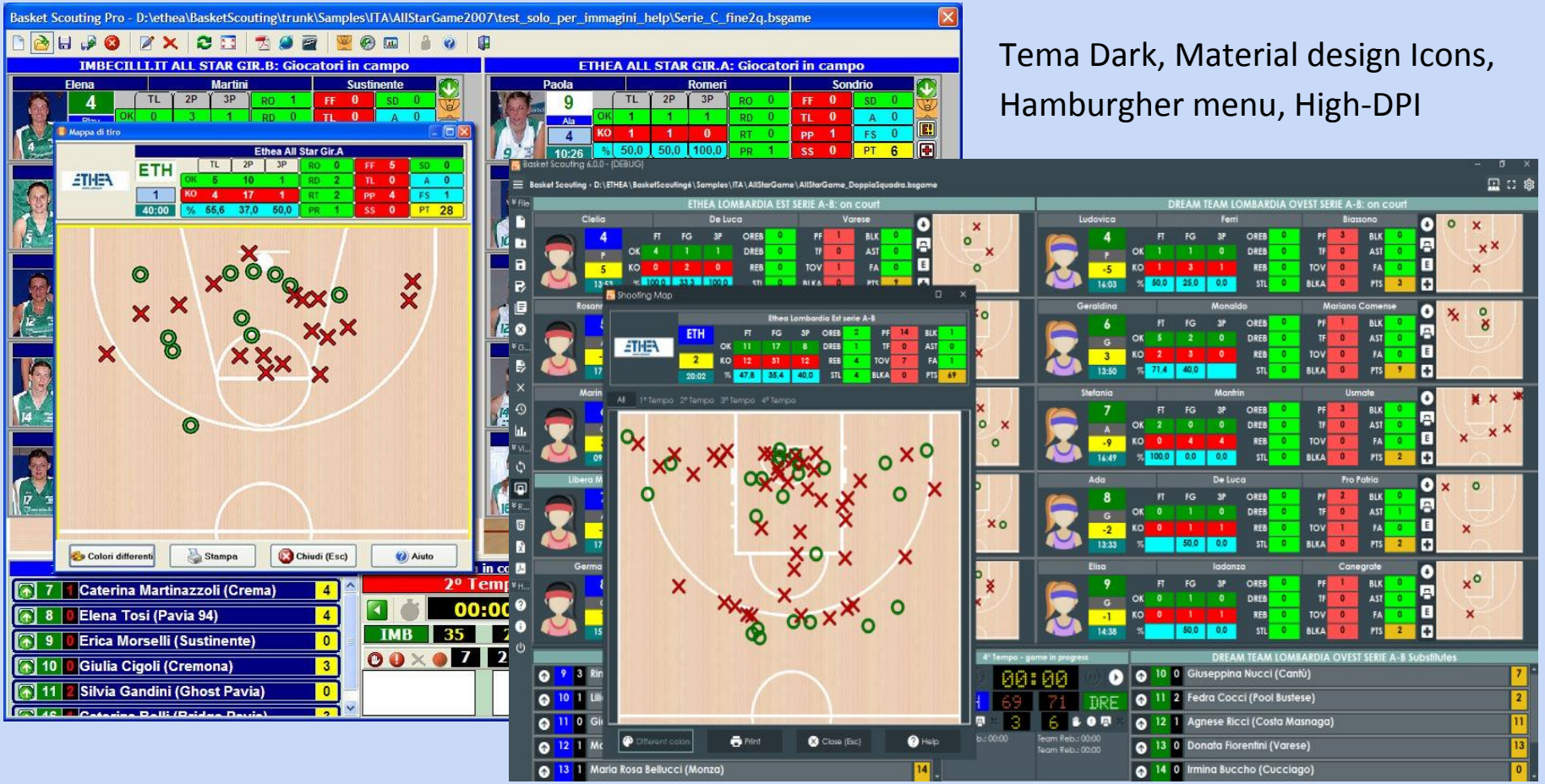
VCLThemeSelector: necessario almeno Delphi XE3.

ModernAppDemo: necessario almeno Delphi 10.1.

Perché è necessario modernizzare le app VCL?

- Problemi di App esistenti:
  - Le app VCL sono nate decine di anni fa (default Font Tahoma)
  - A volte utilizzano componenti di terze parti non più aggiornati
  - Alcuni paradigmi di UI sono vecchi: MainMenu, CheckBox?
  - Posizionamento in base ai “pixel” senza supporto High-DPI
- Problemi con le nuove App:
  - Supporto High-DPI di default
  - Nuovi componenti (es. FlowPanel) e componenti per Windows 10
  - Dark Mode / Light Mode: non è solo una moda
- Problemi hardware:
  - Schermi ad alta risoluzione 2K o 4K

Perché è necessario modernizzare le app VCL?

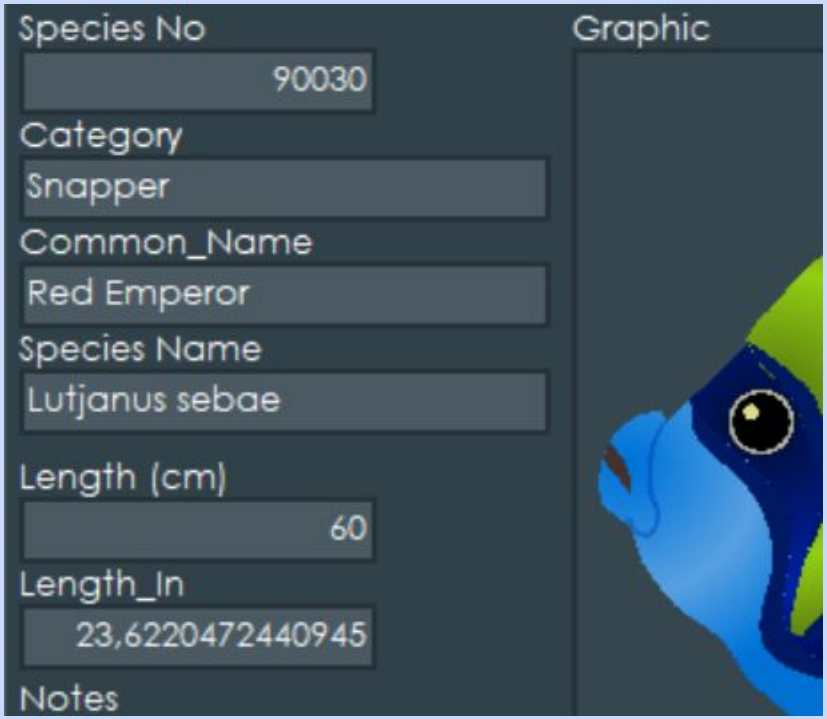


Tema Dark, Material design Icons, Hamburgher menu, High-DPI

Perché è necessario modernizzare le app VCL?

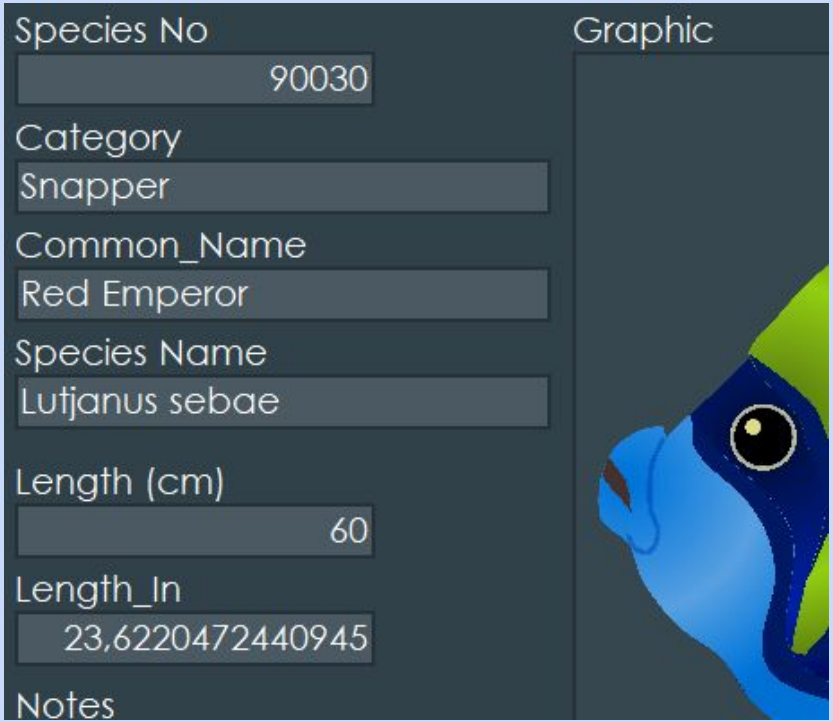
Applicazione senza High-DPI

DPI virtualization (da Windows Vista)



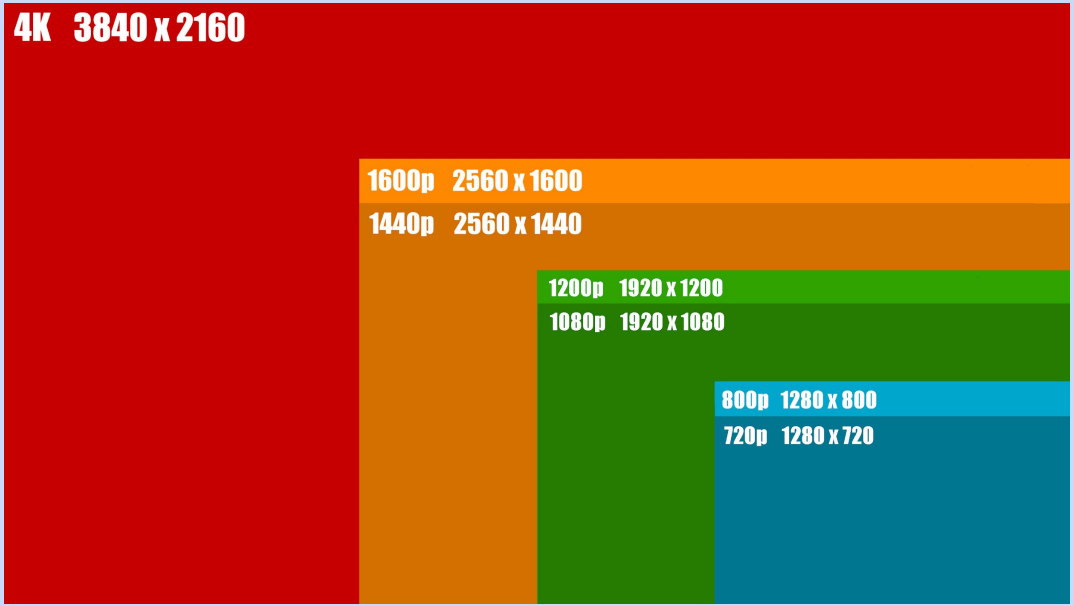
Applicazione con High-DPI

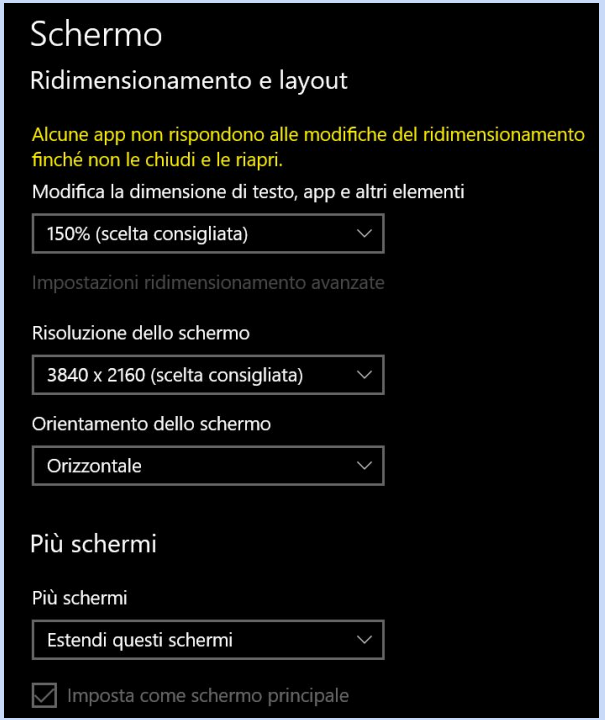
Per monitor DPI-scaling (da Windows 8.1)



Perché è necessario modernizzare le app VCL?

Diverse risoluzioni, ma anche diversa % del Font, cioè DPI (DotPerInch) = PPI (PixelsPerInch)





con gli schermi 2K su notebook da 14/15 pollici e sugli schermi

4K è facile trovare impostazioni di 125% o 150% o anche 200%.

.. e non vogliamo che la nostra app non risponda alle modifiche di ridimensionamento!

Come IconFontsImageList ci aiuta...

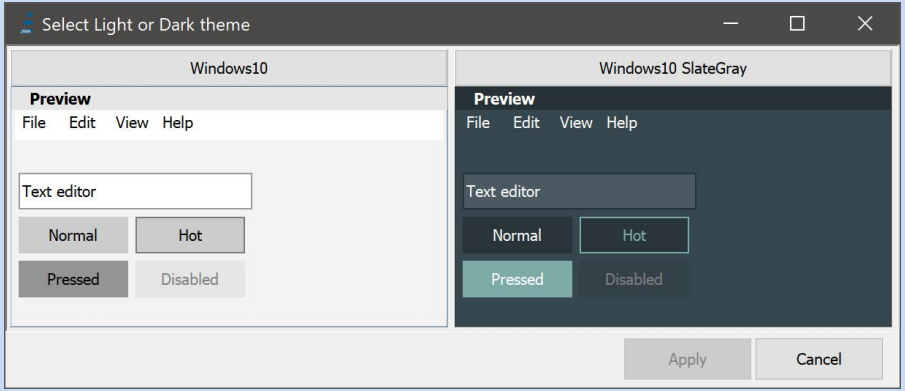
- Disponibile gratuitamente (da Delphi 7 a 10.3)
- IconFontsImageList
  - Disponibilità di molte icone già pronte
    - esempio: https://materialdesignicons.com/ (installarlo nel sistema)
  - Possibilità di cambiare colore e dimensione facilmente
  - Nella 1.5 è stato aggiunto il supporto alle icone “surrogate pair”
  - Nella 1.6 è stata aggiunta la CharMap
  - E’ disponibile anche una versione per FMX
- In alternativa si deve utilizzare TImageCollection e TVirtualImageList e lavorare con Bitmap pre-scalate.

Modernizzare con i VCLStyles

- Sfruttiamo gli stili della VCL
  - Come si abilitano e si aggiungono
  - Disponibilità di nuovi stili in GetIt
  - Attenzione alla proprietà: StyleElements (per “disattivarli”)
- Il “non stile” Windows
  - Attenzione ai componenti “owner-draw”
  - Attenzione agli attributi Ctrl3D, Bevel, Border, ecc…
- Uso dei colori:
  - TStyleManager.ActiveStyle.GetStyleColor, GetStyleFontColor, GetSystemColor, GetElementColor...

Tema chiaro/scuro

- Sfruttiamo gli stili della VCL
  - Attenzione alla proprietà StyleElements (per disattivarli).
- Usiamo il preview/selector TVCLThemeSelector per semplificare la scelta
  - Escludere o meno il “non stile” Windows
  - Autoridimensionamento (MaxCols e MaxRows)
  - Come si integra (provare il VCLThemeSelectorLauncher)
  - Basato su
    VCLStylePreview
    della libreria VCLStyleUtils
    ma con supporto High-DPI



VCLStyleUtils e oltre...

- Libreria molto utile per fixare alcuni comportamenti anomali degli stili.
  - https://github.com/RRUZ/vcl-styles-utils
  - Per fixare Menu, Dialogs, Components, TaskDialog
  - Soprattutto con vecchie versioni di Delphi (da XE2)
- E’ stato annunciato per Delphi 10.4:
  - il supporto “Per Control Styling”
  - il supporto High-DPI Styles
  - https://community.idera.com/developer-tools/b/blog/posts/get-ready-for-the-10-4-beta-with-update-subscription

Il supporto High-DPI

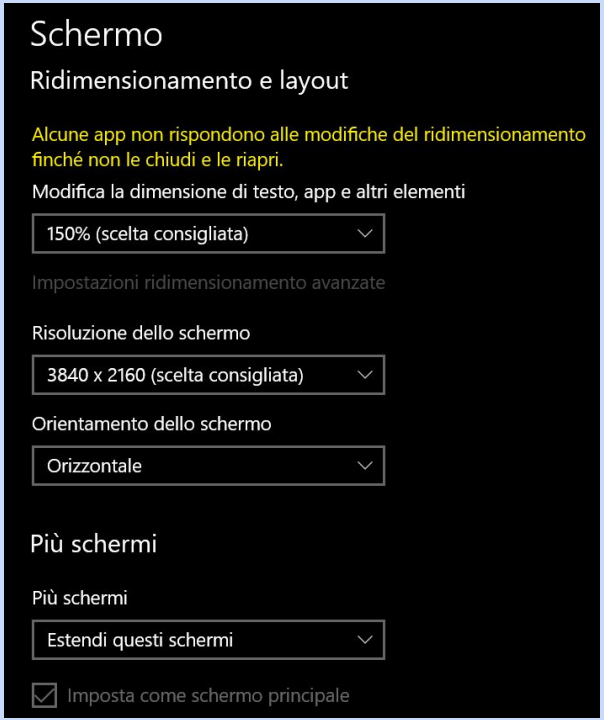
- Come funziona? Come si attiva in Delphi?
  - Documentazione M$:
    https://docs.microsoft.com/it-it/windows/win32/hidpi/high-dpi-desktop-application-development-on-windows?redirectedfrom=MSDN
  - Documentazione Embarcadero:
  - https://community.idera.com/developer-tools/b/blog/posts/how-to-guide-upgrading-your-delphi-vcl-applications-to-support-4k-displays
- Supporti differenti in base alla versione di Delphi (consigliato D10.3)
  - Delphi fa quasi tutto il lavoro in automatico se:
    - Form.Scaled = True, Form.ParentFont = False (Form.PixelsPerInch=96)
    - Attenzione alle impostazioni “autosize”
- In Windows parliamo di “logical DPI” che non corrisponde a quella fisica!

Il supporto High-DPI

Il valore DPI in Windows è “logico” e non ha a che fare con il DPI fisico del Display. Esso definisce quanto “largo” un testo è mostrato rispetto al valore costante di **96** (Form.PixelsPerInch default), che corrisponde al 100% nelle impostazioni di Windows.

Es. **144** DPI (= 96 x 1.5) significa che gli elementi sullo schermo (non solo il font) appariranno il 50% più grandi del "normale" (impostazione 150%).

Es. **192** DPI (= 96 x 2) è l’impostazione di uno schermo 4K al 200% che mostra sullo schermo le stesse “proporzioni” di un comune 2K al 100%.



TFont panoramica e utilizzo

- Proprietà Size e Height:
  - Size è il “corpo” che è disponibile nel font
  - Height è la dimensione in Pixel (con segno negativo)
  - PixelsPerInch è la proprietà DPI dello schermo principale
  - N.B. usare sempre Height per i calcoli (maggior precisione)
- Ruolo di Application.DefaultFont
- Ruolo di Screen.IconFont
- Vediamo la demo TestDPI per capire come funziona e come gestirli
- I font di sistema: Screen.MenuFont, Screen.MessageFont, Screen.HintFont, Screen.CaptionFont

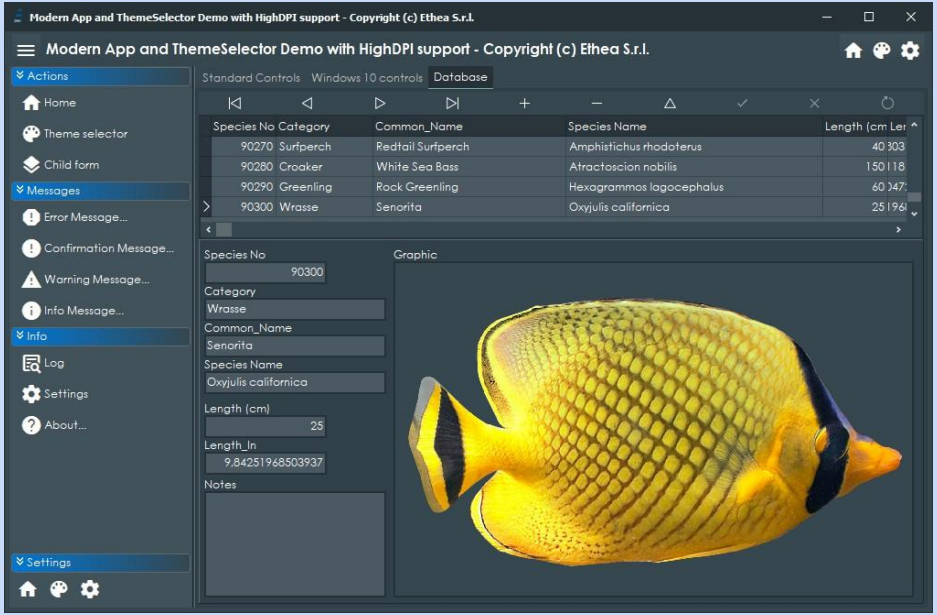
I componenti per Windows 10

- Da Delphi 10 Seattle sono stati introdotti alcuni componenti nuovi:
  - TRelativePanel, TSplitview*, TToggleSwitch*, TActivityIndicator, TSearchBox*
- Da Delphi 10.1.2 sono stati introdotti:
  - TCalendarView*, TCalendarPicker*
- Da Delphi 10.2.2 sono stati introdotti:
  - TDatePicker*, TTimePicker*

*Diamo uno sguardo alla ModernAppDemo per vederli all'opera e come vanno fixati (la demo è compatibile da Delphi 10.1 in poi).

Panoramica ModernAppDemo

- Utilizzo di IconFontsImageList (multicolor e styled)
- Utilizzo TSplitView e TCategoryButton
- Uso VCLStyles con VCLThemeSelector
- TSplitView Settings “senza” stile
- Utilizzo di controlli Windows 10 e fix
- Esempio Immagini Hi-Res - Biolife ;-)
- Personalizzazione Font Name, Height
- System fonts (Menu, Hint, Messages)
- VCLStyleUtils (TaskDialog, Dialog)
- Child Form con ParentFont
- High-DPI e cambio risoluzione al volo



High-DPI: Personalizzare il font

- N.B. ParentFont True o False cambia il comportamento:
  a. Con False viene scalato automaticamente il Font della Form
  b. Con True viene assegnato alla Form Application.DefaultFont
- 2 alternative:
  a. Main e ChildForm con ParentFont a False:
     - VCL fa il resize
     - Riassegnazione Font personalizzato in ogni form (o tramite una Child form di base)
  b. MainForm con ParentFont a False e Child forms con ParentFont a True:
     - VCL fa il resize solo della Main
     - La Child “eredita” Application.DefaultFont
     - Vantaggio: una sola assegnazione
     - Svantaggio: impostare ParentFont a true nei dfm di tutte le form (non è il default)
- Suggerimento: utilizzare una form di base per gestire le casistiche

High-DPI: gli eventi da utilizzare

- Screen.PixelsPerInch e Form.Monitor.PixelsPerInch
  - Screen.PixelsPerInch è il monitor principale
  - Font.Monitor.PixelsPerInch è legato al monitor in cui la form è visibile
- OnAfterMonitorDpiChanged
  - Quando si cambia monitor o quando si cambiano le impostazioni
  - Quando si lancia una ChildForm da un monitor diverso dal primario
- ScaleForPPI
  - Metodo della form che scala i componenti: utile per fixare componenti VCL o di terze parti che non scalano correttamente
- ChangeScale
  - Metodo che i componenti ricevono per scalare: utile per “sistemare” il comportamento internamente al componente creandosene di propri.

Resize in base alla dimensione dello schermo

- Considerare l’impostazione High-DPI non sempre è la soluzione
  - Dipende dal tipo di applicazione
  - Dipende sopratutto se è pensata per funzionare a tutto schermo (es. su Tablet Windows)
- Esempio di BasketScouting:
  - Il fattore di scala va ricalcolato solo in base alla dimensione dello schermo
  - L’interfaccia utente scala diversamente anche in base al contenuto/contesto

Materiale on-line di questa sessione:

- IconFontsImageList:
  - https://github.com/EtheaDev/IconFontsImageList
- VCLThemeSelector e Demo High-DPI:
  - https://github.com/EtheaDev/VCLThemeSelector
    - Form “elegante” per selezionare il tema della app
    - Demo completa con i concetti di modernizzazione e supporto High-DPI trattati in questa sessione.

VCLThemeSelector: necessario almeno Delphi XE3.

ModernAppDemo: necessario almeno Delphi 10.1.

Grazie