Icone in Delphi: nuove possibilità con IconFontsImageList e SVGIconImageList

di Carlo Barazzetta* (carlo.barazzetta@ethea.it) - 14 Settembre 2020
*autore di IconFontsImageList e SVGIconImageList

Indice degli argomenti

Premessa 2

Un po' di storia 3
- Le icone in Windows 3
- Google - Material Design 3

Tema chiaro e scuro 4

L'avvento degli schermi High-DPI (2K, 4K, 8K) 4

Le icone in Delphi e la VCL 5
- TImageList 5
- Delphi 10.3: TImageCollection e TVirtualImageList 6
- TImageCollection 6
- TVirtualImageList 7
- Vantaggi nell'uso di TImageCollection e TVirtualImageList 8
- Limiti nell'uso di immagini Bitmap 8
- **La soluzione? Font di Icone o SVG** 8
- Un elemento imprescindibile: la "scalabilità" 9
  - I font di Icone 9
  - Le Icone SVG 9

Evoluzione di IconFontsImageList 10

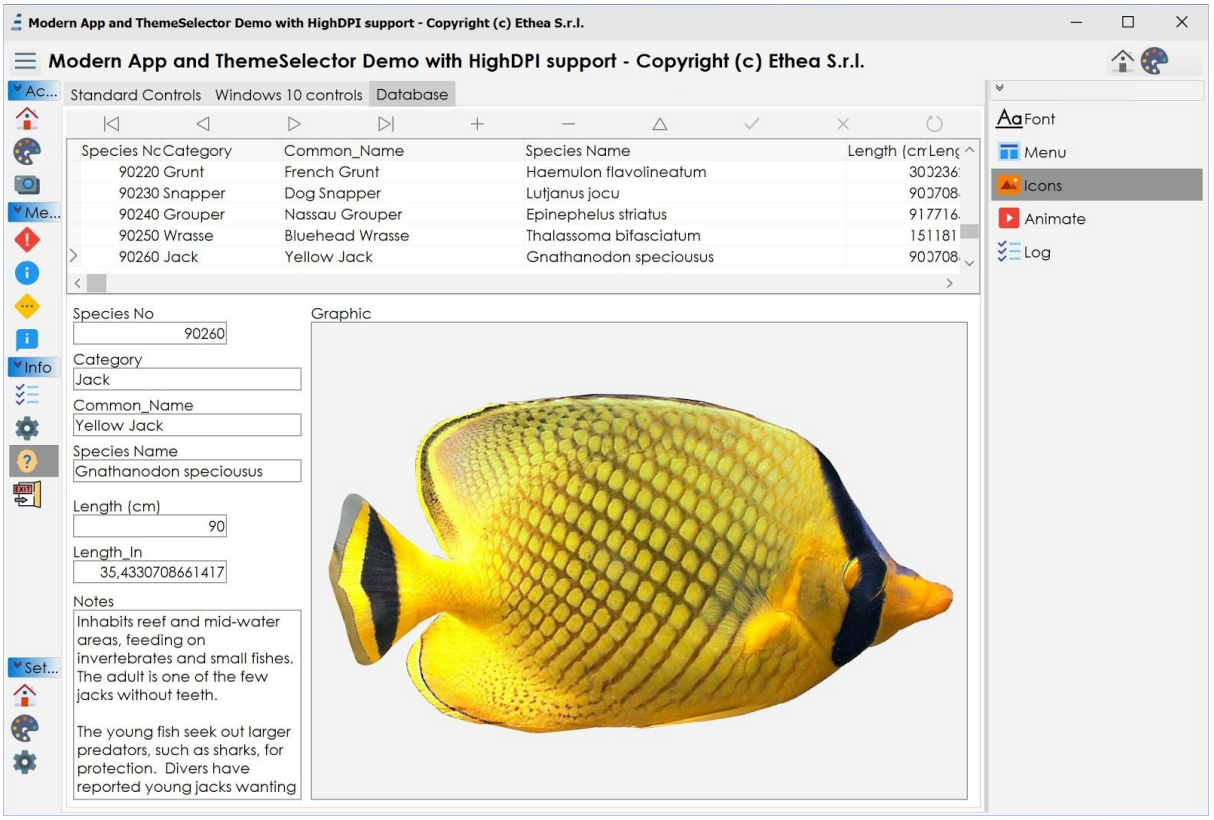
Evoluzione di SVGIconImageList 11
- 2 motori di rendering SVG 12
- Pubblicazione su Get-It 13

IconFontsImageList e SVGImageList: caratteristiche e istruzioni per l'uso 14
- I Componenti 14
  - Collection + VirtualImageList per vecchie versioni Delphi 14
  - Il nuovo component editor di IconFontsImageList 15
  - Il nuovo component editor di SVGIconImageList 16
  - Supporto alla VirtualImageList nativa 17
- Utilities 18
  - IconFonts CharMap e catalogo Icone 18
  - SVG Icons Explorer 19
- Demos e documentazione 19
- Ringraziamenti 20
- FireMonkey version 20

Premessa

Circa 10 mesi fa ho iniziato a scrivere IconFontsImageList (e più tardi anche SVGIconImageList), perché mi servivano delle funzionalità aggiuntive per affrontare al meglio il tema della modernizzazione di una applicazione (sia VCL che Firemonkey).
L'idea di fornirle gratuitamente è nata dalla sensazione che ci fosse bisogno di un maggiore supporto all'utilizzo delle icone in Delphi alla portata di tutti, senza necessariamente ricorrere a componenti di terze parti. Il fatto che siano diventate molto popolari in pochi mesi significa che non ero l'unico a sentirne la mancanza!

Ora che entrambe le librerie hanno raggiunto un alto livello di stabilità e performances, ho deciso di scrivere questo articolo per spiegare perché sono nate e come si utilizzando, dato che le ritengo indispensabili nello sviluppo di applicazioni moderne, come quella mostrata in questa immagine:



Dopo una prima parte, nella quale intendo focalizzare il tema ed evidenziare alcuni limiti derivati dall'utilizzo dei componenti nativi, verrà illustrata la genesi e l'evoluzione dei componenti, fino ad arrivare alla versione definitiva.

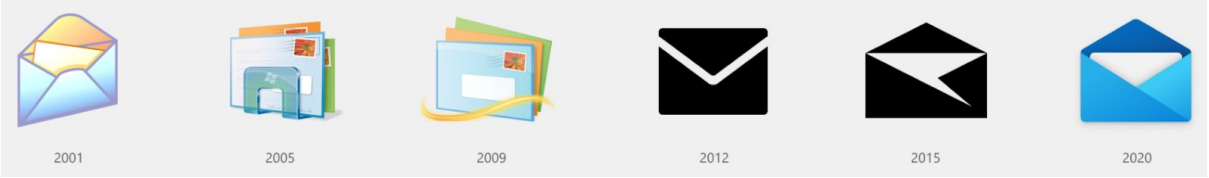
Uno dei temi principali è la transizione dall'utilizzo di una classica TImageList (VCL) all'accoppiata TImageCollection + TVirtualImageList, introdotta in Delphi 10.3, che è il presupposto all'abilitazione del supporto High-DPI di una moderna applicazione.
L'articolo comunque si rivolge anche a chi utilizza versioni precedenti alla 10.3, in quanto è stato implementato un supporto simile per essere disponibile con versioni più datate (in un caso fino a Delphi 7).
In un prossimo articolo sarà trattata anche la versione per Firemonkey...
Buona lettura.

Un po’ di storia

Per focalizzare il contesto è utile fare un passo indietro ed osservare l’evoluzione delle Icone nel mondo informatico ed in particolare nell’ambiente Windows.

Le icone in Windows

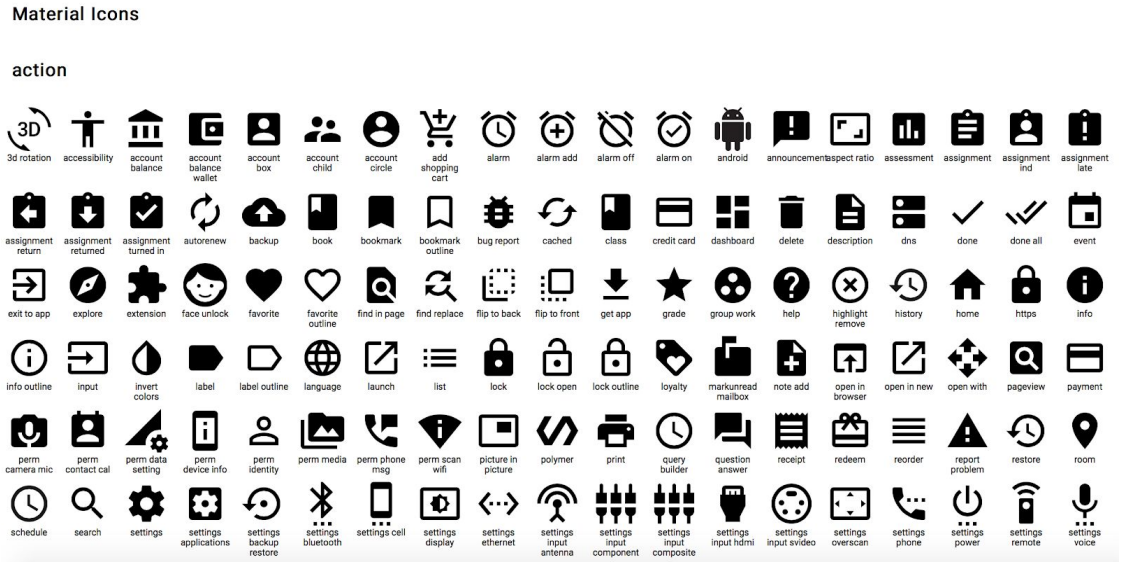
Senza entrare troppo nel dettaglio dell’evoluzione delle icone nel mondo informatico (qui si trova un esempio ben fatto: https://historyoficons.com/), in ambito Windows eravamo abituati ad utilizzare icone molto colorate e sempre più simili alla realtà, fino al 2009, quando arrivò la rivoluzione con Windows 8 e successivamente portata avanti con Windows 10.



Di colpo avvennero 2 fatti: le icone divennero piatte e stilizzate all’osso, fino all’utilizzo di un solo colore. Nelle versioni successive di Windows 10 le icone hanno cambiato ancora faccia: ora sono ancora molto stilizzate, tendenzialmente con un colore primario e sfumature che le hanno ridato un po’ di tridimensionalità:

Google - Material Design

Anche il mondo Web ha visto la sua rivoluzione spinta dalle linee guida di Google riferite all’ecosistema “Material Design”. Il grosso successo di questo approccio ha anche portato ad una unificazione del significato delle icone, perché sono identificate con un nome e una categoria, per cui in applicazioni diverse si trovano le medesime icone utilizzate per gli stessi scopi (nella figura un esempio con le icone “Material” della categoria “action”).

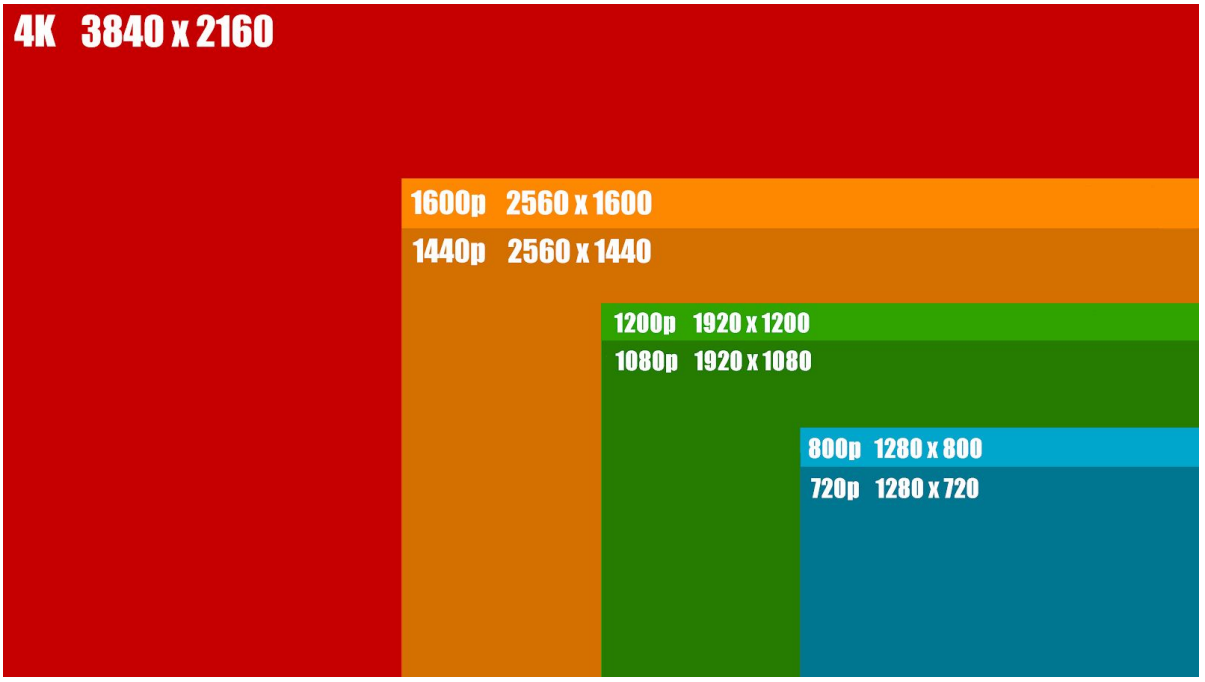


Tema chiaro e scuro

Un'altra piccola rivoluzione ancora molto recente (fine 2018), che ha coinvolto tutte le applicazioni più famose è stata l'introduzione del tema "scuro": in prima battuta come scelta comoda per risparmiare batteria ai cellulari (Windows Phone 7), ma poi si è rivelata molto apprezzata, soprattutto dai giovani, che la preferiscono anche nell'utilizzo con il PC.

L'avvento degli schermi High-DPI (2K, 4K, 8K)

Da ultimo, sempre più notebook sono equipaggiati con schermi di risoluzione 2K e sempre più monitor per uso professionale arrivano a 4K (e la frontiera degli 8K è dietro l'angolo). Con schermi di alta risoluzione, l'utente ha la necessità di intervenire sulla dimensione del carattere, o per essere più precisi sul fattore DPI (punti per pollice): uno dei valori immutabili per decenni nello sviluppo (in Delphi PixelsPerInch) è diventato significativo.
In una applicazione moderna, per avere una resa ottimale, le icone dell'applicazione devono avere la capacità di adattarsi (scalare) a qualsiasi DPI. Con uno schermo 4K ci sono 5 fattori DPI selezionabili dall'utente (100%, 125%, 150%, 175%, 200%).



A partire dalle ultime versioni di Delphi è stato aggiunto il supporto High-DPI: abilitando tale supporto un'applicazione dovrà gestire in maniera appropriata l'aspetto, in modo tale da adattarsi ai diversi fattori di scala. Ma mentre i componenti, a secondo delle varie versioni di Delphi, si sono sempre più evoluti per adattarsi automaticamente, la gestione delle Icone rimane un aspetto ostico e per molti aspetti problematico: vediamo perché.

Le icone in Delphi e la VCL

Partiamo da che cosa offre Delphi e l'evoluzione "Virtual" introdotta in 10.3, per scoprire i notevoli vantaggi e alcuni limiti dei componenti nativi.

TImageList

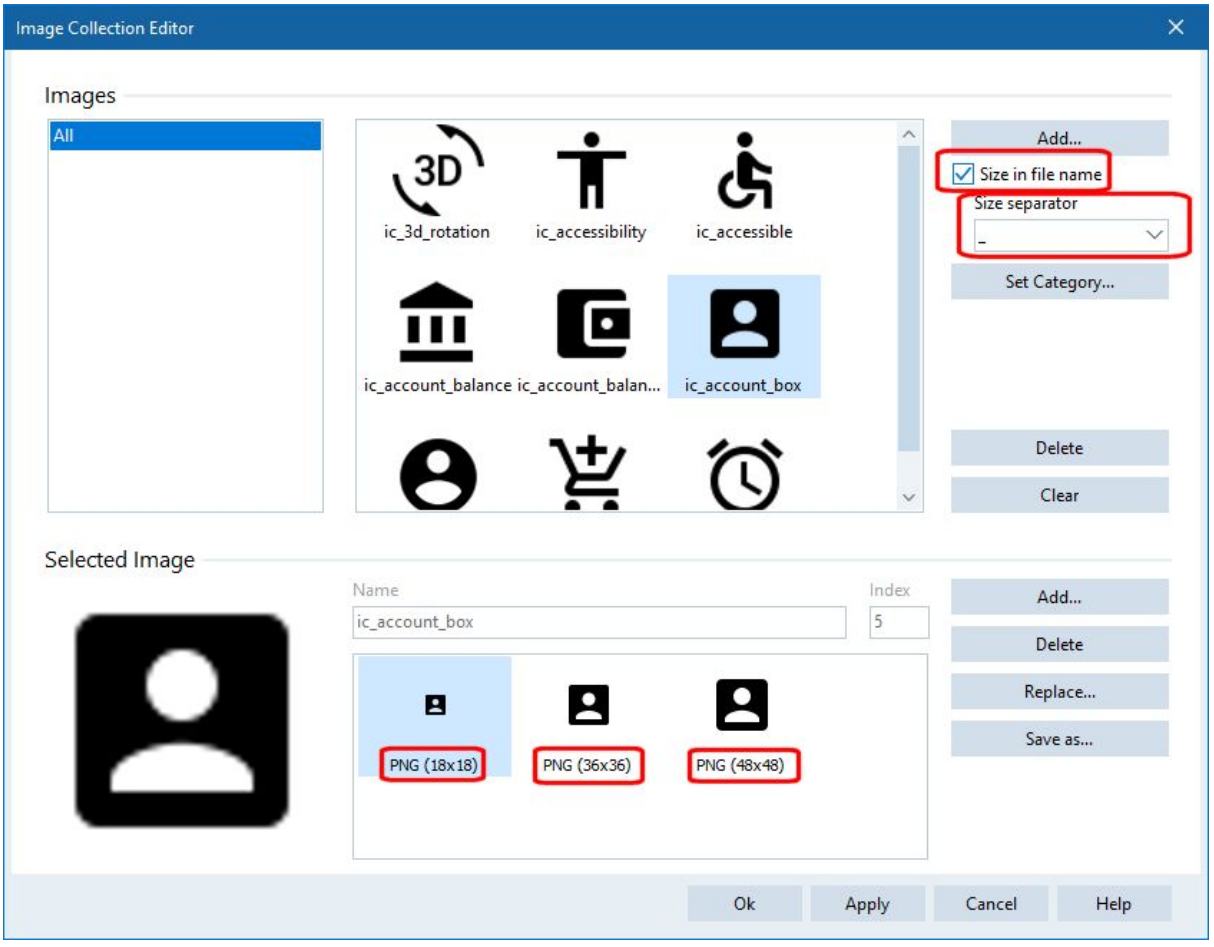
Storicamente, il componente che si è sempre occupato di gestire le Icone in una applicazione Delphi è stata la **TImageList**, una collezione di immagini **bitmap** di dimensione fissa in vari formati (bmp o gif o png a seconda del supporto grafico che man mano è stato aggiunto). La TImageList veniva quindi agganciata alla GUI (TAction, TToolBar, TButton, ecc…) per renderizzare le icone attraverso un indice che individuasse l'icona stessa.
Un'altro aspetto che si è evoluto negli anni è stata la gestione della trasparenza: dal famoso pixel in basso a sinistra dell'immagine fino ai grafici (GIF, PNG…) in cui la trasparenza era contenuta direttamente nelle informazioni dell'immagine.
Con l'arrivo del supporto GDI+ (da Delphi XE4) c'è stata un'evoluzione sulla qualità e sulle performance.

Delphi 10.3: TImageCollection e TVirtualImageList

In una applicazione High-DPI è necessario avere le stesse icone in grado di adattarsi a diversi fattori di DPI (dimensioni), pertanto si è reso necessario separare il concetto di "dimensione" dal concetto di "aspetto".
A partire da Delphi 10.3 sono disponibili 2 nuovi componenti che servono per sostituire la vecchia TImageList: **TImageCollection** e **TVirtualImageList**.

TImageCollection

E' un componente che raccoglie una lista di immagini "Bitmap" caricabili da file (di qualsiasi formato e dimensione) per definire la "collezione" di immagini.
Normalmente il componente lo si mette in un datamodule in modo da poter essere utilizzato da tutte le form dell'applicazione che ne hanno la necessità.
Uno dei vantaggi (non sempre noto a tutti) è quello di poter avere la stessa icona nella collezione ma con risoluzioni diverse: è necessario fare attenzione nell'impostare "Size in filename" e "Size separator" prima di caricare le icone con il pulsante "Add", dentro il component editor.
Se ad esempio si imposta come "size separator" il carattere underscore e si caricano 3 file: ic_account_circle_18pt.png, ic_account_circle_36pt.png, ic_account_circle_48pt.png, la collection avrà una sola icona dal nome ic_account_circle che conterrà tre immagini di 3 risoluzioni diverse: 18, 36 e 48 pixel.

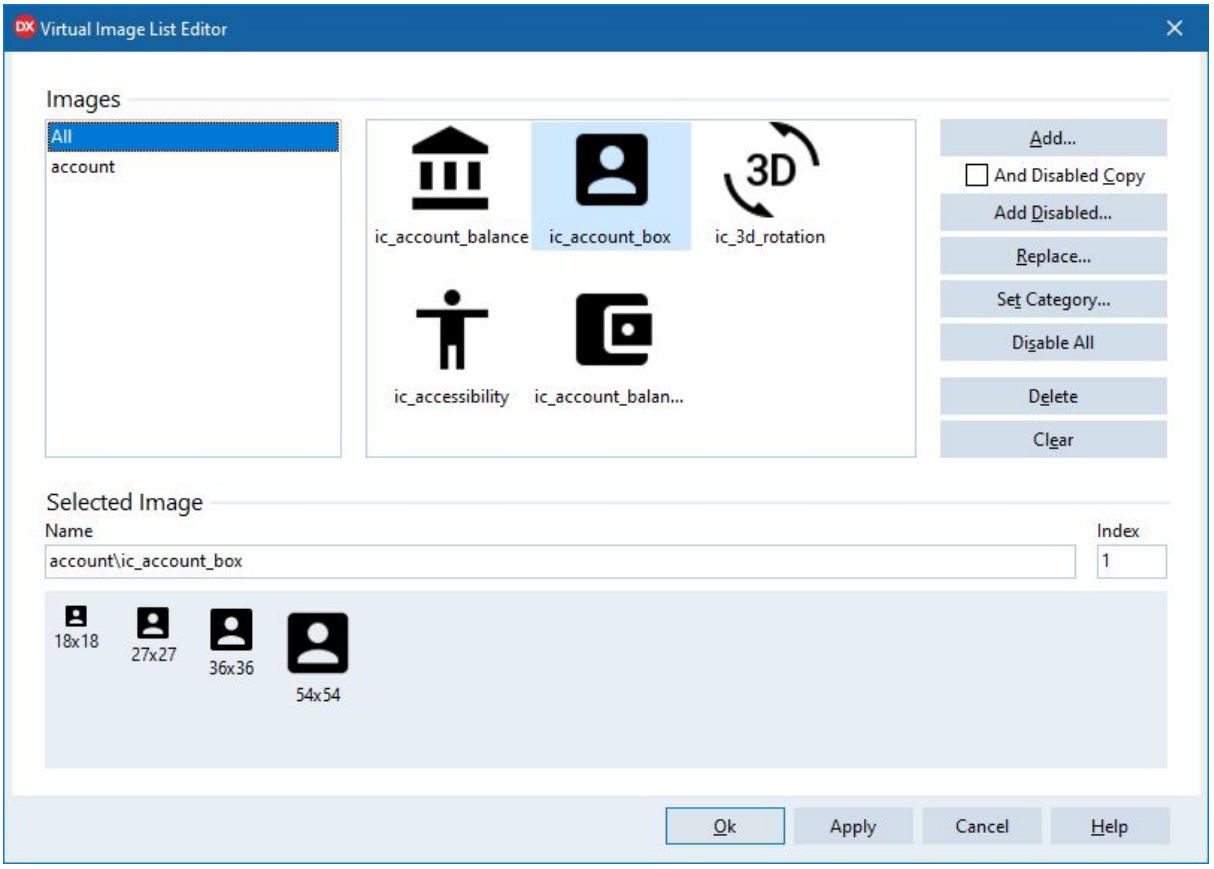


Da notare che le differenti risoluzioni sono solo il punto di partenza per poterle successivamente scalare alla dimensione richiesta. Infatti il componente contiene anche

l'algoritmo di scaling definito nella proprietà "Interpolation mode" che di default è icIMModeHighQualityCubic: a seconda dell'algoritmo scelto si determina la qualità a scapito delle performance e viceversa.
Importante anche il fatto che un'icona è identificata per nome, ed è possibile anche assegnare una Categoria, per poter essere meglio "filtrate" quando verranno utilizzate dalle VirtualImageList.

TVirtualImageList

Per utilizzare la raccolta di icone contenute nella TImageCollection si utilizza il componente TVirtualImageList, collocandolo su ogni form e lo si collega alla Collection.
Chiaramente la TVirtualImageList definisce le dimensioni (Width e Height) delle icone che vuole utilizzare, ma occorre ricordare che è la dimensione sempre relativa ad uno schermo con 96 DPI.
Il vantaggio di collocare una VirtualImageList su ogni form permette anche di scegliere "solo" le immagini che ci servono dalla nostra collection nel contesto di quella form: in questo esempio ci sono solo 5 delle immagini della collezione, anche in posizione diversa, perché sono identificate per nome. E' anche possibile definire una categoria alle icone: di fatto l'icona si chiamerà "category\name" (es. "account\ic_account_box") anche se purtroppo le categorie definite nella collection non vengono recuperate automaticamente nel passaggio alla VirtualImageList.



Nella parte inferiore del component editor viene mostrata una preview dell'icona in caso di "fattori di scala" diversi: dato che la VirtualImageList di esempio ha le dimensioni 18x18, mostra l'icona di partenza oltre alle altre tre: 150% (27), 200% (36) e 300% (54).
E' anche possibile stabilire che ci sono icone "disabled" e a livello di componente si può specificare il suffisso (_disabled) e il valore di opacità (default 125) per rendere le icone "disabilitate".

Vantaggi nell'uso di TImageCollection e TVirtualImageList

I vantaggi di questa impostazione sono notevoli e per questo motivo è caldamente consigliato abbandonare le TImageList per passare alla nuova coppia di componenti.
I principali sono:

1) ogni form può contenere diverse VirtualImageList di dimensioni diverse (es.18x18 per le icone su una toolbar e 24x24 per le icone sui pulsanti) che condividono la stessa collezione.
2) ogni VirtualImageList si collega alla collection per nome e non per indice, permettendo di aggiungere icone alla collection anche a posteriori perfino spostandone la posizione: è la VirtualImageList che poi si collega a sua volta agli elementi della GUI tramite "ImageIndex" (e da Delphi 10.4 anche tramite "name").
3) E' possibile definire nelle VirtualImageList solo le icone strettamente necessarie alla form stessa, senza quindi dover "generare" tutto il set di icone previsto nella collection. Bisogna infatti considerare che è la VirtualImageList che conosce il DPI del monitor sul quale la form viene visualizzata e che quindi "richiede" alla collection di "disegnare" l'icona della dimensione definita, ma con il fattore di scala imposto dal monitor!

Oltre ai vantaggi descritti, va ricordato che questa architettura ***è l'unica possibile in una applicazione High-DPI che abbia più di una form da visualizzare***, perché "potenzialmente" le diverse form potrebbero essere trascinata su schermi con fattore DPI diversi ed il resize delle icone deve riguardare solamente la singola form e non l'intera collezione.

Limiti nell'uso di immagini Bitmap

Il maggior limite della TImageCollection è legato al fatto di dover utilizzare immagini bitmap di dimensioni "fisse", questo comporta una serie di problemi:

1) Avere a disposizioni icone a diverse risoluzioni e caricarle tutte nella collection, determinando occupazione di spazio di memoria inutile, ma soprattutto di risorse (tutte le bitmap in tutte le risoluzioni) che magari non verranno mai utilizzate nel ciclo di vita dell'applicazione.
2) Doversi affidare agli algoritmi di "rescaling" nel momento in cui è richiesta una dimensione di icona che non esiste nella collezione.
3) Dover avere il doppio di icone (come minimo) per gestire una applicazione con tema chiaro / scuro o addirittura un set di icone di colori diversi che si adattino al meglio allo stile VCL selezionato dall'utente.

La soluzione? Font di Icone o SVG

Per poter modernizzare una applicazione Delphi - VCL dobbiamo necessariamente tenere conto e risolvere questi elementi:

- Avere a disposizione un set di icone dal design moderno.
- Avere a disposizione un set di icone in grado di scalare su schermi con DPI diversi.
- Poter adattare facilmente il colore delle icone al tema (chiaro o scuro) selezionato dall'utente.

Un'altro elemento vincolante, nel mio caso, era quello di avere una soluzione compatibile con versioni di Delphi precedenti alla 10.3, dove ImageCollection e VirtualImageList non sono ancora disponibili.
Trovare una soluzione che fosse applicabile anche a versioni precedenti sarebbe risultato un plus per molti sviluppatori Delphi che hanno la necessità di "ringiovanire" vecchie applicazioni, almeno dal punto di vista grafico, senza dover (o poter) cambiare la versione di Delphi.

Un elemento imprescindibile: la "scalabilità"

Il problema principale da risolvere è quello di poter scalare le icone utilizzando una tecnologia che ci permetta di farlo senza passare attraverso algoritmi di interpolazione ma sfruttando la grafica di tipo vettoriale, ed ora che si è affermato il concetto di avere delle icone "stilizzate" è la soluzione ottimale.
Anche nel mondo WEB, dove la qualità delle immagini conta molto, si sta assistendo alla stessa trasformazione e la risposta è contenuta in due tecnologie, per certi versi molto simili:

1) i Font di Icone
2) le Icone SVG

I font di Icone

L'uso di Web-Fonts o Font di Icone è in larghissima espansione. Esistono ormai decine di collezioni molto note e ricche, ne citiamo un paio:
https://github.com/Templarian/MaterialDesign-Font (quasi 5.600 icone free di tipo "Material-Design")
https://fontawesome.com/icons?d=gallery (quasi 8.000 icone di cui 1.600 free)

Utilizzare un Font di Icone offre molti vantaggi:

1) La collezione è praticamente già pronta (tutte le icone hanno un aspetto coerente)
2) Non bisogno distribuire o caricare centinaia o migliaia di file png, basta uno o pochi Font Files che contengono tutte le icone.
3) La scalabilità è perfetta, grazie alla grafica vettoriale.
4) E' possibile cambiare l'attributo di colore dell'icona.

Quest'ultimo, in realtà, è anche il più grosso limite dell'utilizzo di Font di Icone: non è infatti possibile utilizzare più colori nella stessa icona.

Le Icone SVG

Scalable Vector Graphics è la tecnologia in grado di definire immagini "vettoriali" e quindi perfettamente scalabili. Introdotta nel 2001, per tanti anni è rimasta una tecnologia relegata ad un uso limitato e poco utilizzata per le icone, perché richiede un "motore" di rendering: oggi non è più un problema dover "disegnare" al volo le icone grazie alle prestazioni di CPU/GPU, pertanto l'utilizzo di Icone SVG offre anch'essa tanti vantaggi:

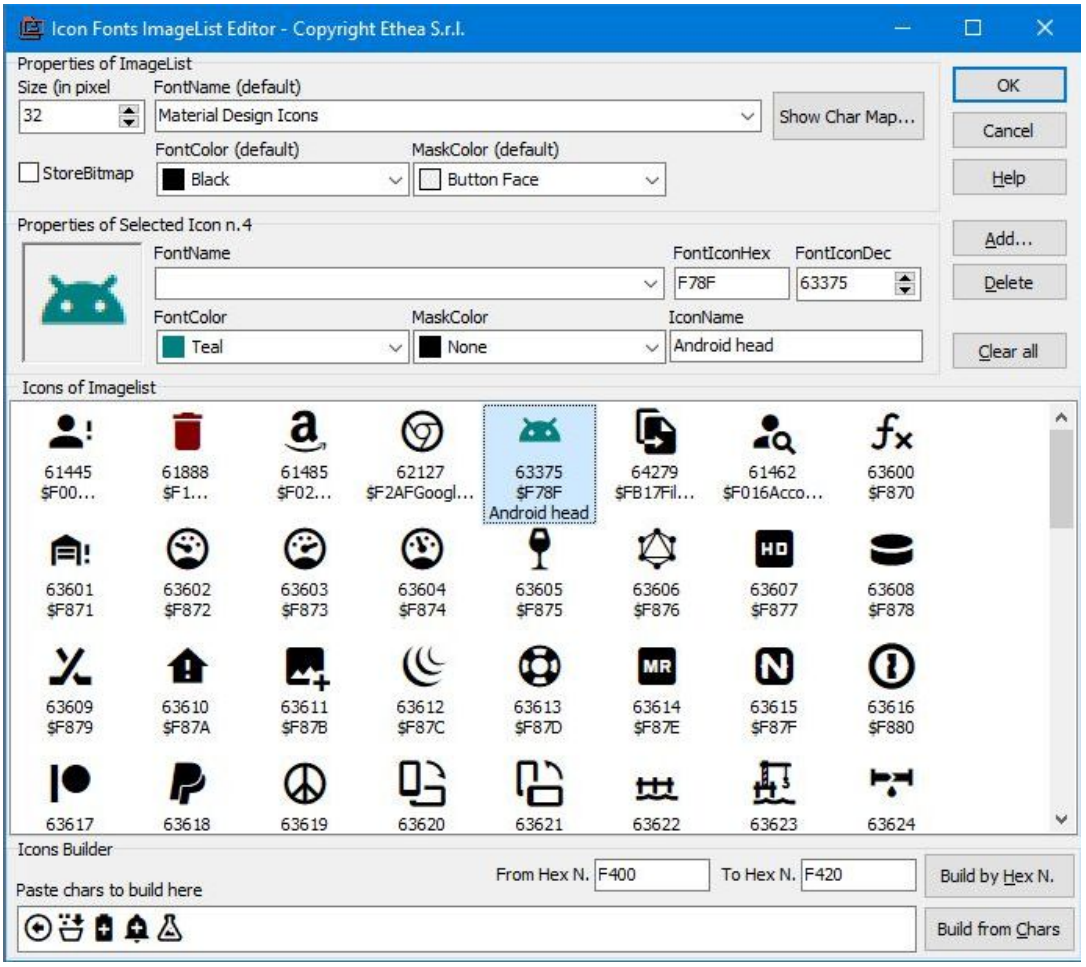
1) Esistono in circolazione tantissime collezioni di icone SVG.
2) Non servono tanti file per ogni risoluzione ma un solo file di testo SVG.
3) La scalabilità è perfetta.
4) E' possibile facilmente manipolare gli elementi xml del testo SVG per cambiare. attributi, come il colore.

L'unico svantaggio è legato alla necessità di avere un "engine" in grado di supportare tutte le caratteristiche dello standard SVG, ma vedremo come anche questo problema è risolvibile.

Non mi restava che applicare queste tecnologie al mondo Delphi...

Evoluzione di IconFontsImageList

Nel mese di Novembre 2019 inizia lo sviluppo: l'idea è semplice, ispirata da Luca Minuti (https://github.com/lminuti/FontIconEditor) che mostra come aggiungere una icona ricavata da un font di icone ad una ImageList, ma la soluzione che immagino fa esattamente il contrario: sostituire una ImageList classica con una ImageList di Icone generate dinamicamente da uno o più font e quindi in grado di "scalare" e di cambiare colore facilmente.
Il primo rilascio Open-Source su GitHub di **IconFontsImageList**, il 25 Novembre, riscuote moltissimo interesse: già dalla prima versione il componente permetteva di generare Icone a partire da Font diversi e di applicare un colore primario ed uno di sfondo. Viene fornita una Demo per illustrare come è semplice ottenere Icone di qualsiasi dimensione perfettamente scalabili e di "adattarsi" al cambio di stile (chiaro o scuro) di una applicazione.
Il componente è corredato da un avanzato editor in grado anche di catturare le icone dalla mappa caratteri del font. (nella figura il "vecchio" component editor).



In seguito ci saranno molti rilasci, con tante novità:

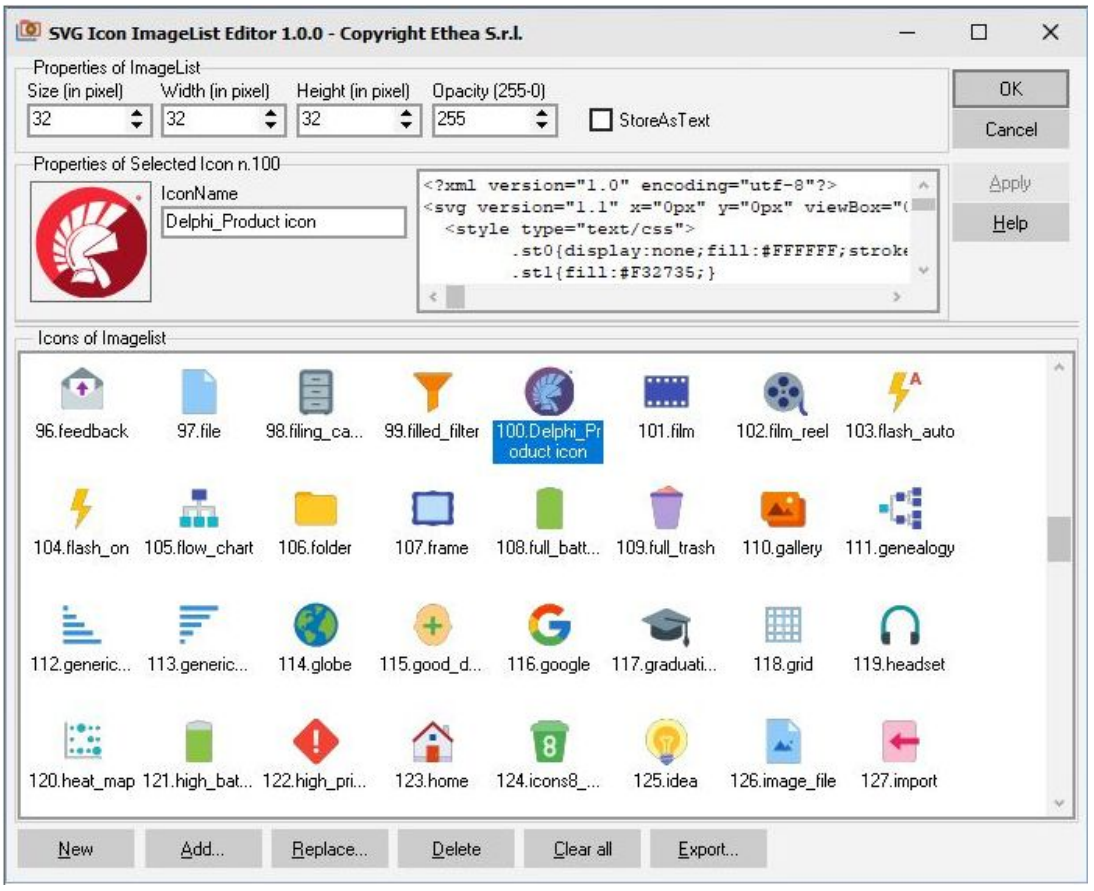
- 30 Gennaio: il componente fornisce il supporto VCL fino a Delphi 7
- 25 Marzo: la versione beta per Firemonkey prende luce
- 28 Marzo: si estende il supporto ai Font che usano i caratteri Unicode "surrogate pair"

- 13 Aprile: si aggiunge una Charmap custom, per mostrare anche i caratteri "surrogate pair" che la charmap di Windows non supporta
- 7 Maggio: la versione per Firemonkey si arricchisce del component editor
- 27 Maggio: vengono aggiunti i metadati dei font "Material Design" e "Font Awesome" per accedere alle icone della CharMap per nome.
- 19 Giugno: la versione 2.0 aggiunge il supporto GDI+ per il rendering e un nuovo componente **IconFontsImage** per mostrare un'icona in una immagine.

In Agosto viene fatto un completo refactoring della libreria e il 27 viene rilasciata la versione 2.2 che contiene la trasformazione in "Collection"+"Virtual": lo vedremo in dettaglio nella sezione dedicata alle funzionalità in dettaglio della libreria.

Evoluzione di SVGIconImageList

Mentre è in corso lo sviluppo di IconFontsImageList, da più parti arrivano anche richieste di poter avere icone colorate e non solo monocromatiche. Mi viene suggerito di costruire un componente simile basato su SVG ma in Delphi non esiste un "engine" SVG nativo.
Esistono in circolazione implementazioni commerciali, ma mi ricordo di una vecchia libreria vista tanti anni fa di Martin Walter, rilasciata gratuitamente (https://development.mwcs.de/svgimage.html).
Dopo alcune ricerche trovo anche un progetto su Git-Hub fermo da tempo (https://github.com/ekot1/DelphiSVG) che aveva recuperato questa libreria ed aveva anche prodotto una ImageList sperimentale (ma con performance terribili), oltre ad una bozza di supporto FMX. Gli elementi c'erano tutti per iniziare a costruire il "gemello" di IconFontsImageList, che viene rilasciato per la prima volta il 24 maggio.
La struttura di **SVGIconImageList** e il component editor ricalcano quella di IconFontsImageList: stavolta le immagini sono definite da un testo SVG e possono essere caricate da files su disco (nella figura il "vecchio" component editor).



Anche se la libreria **TSVG** di Martin Walter presenta alcuni bug di rendering, il primo risultato comunque raccoglie molti consensi e anche in questo caso la libreria cresce col tempo:

- Il 26 Maggio viene resa disponibile la versione per FMX (solo Windows Platform), comprensiva del component-editor.
- Il 28 Maggio viene estesa la retrocompatibilità fino a XE6
- Il 5 Giugno viene aggiunto il componente TSVGIconImage per mostrare un'icona direttamente in una immagine (con possibilità di Stretch / Proportional).
- Il 9 Giugno si aggiungono le proprietà GrayScale e FixedColor, che mostrano la flessibilità di manipolazione delle immagini SVG e permettono quindi di adattare le icone al tema dell'applicazione.
- Il 21 Luglio viene fornito il tool **SVGExplorer** per navigare e vedere le Icone dei file SVG sul filesystem Windows (dato che Windows ancora non è in grado di farlo).

L'interesse cresce fino ad attirare altri sviluppatori come collaboratori: inizia un mese Agostano "frenetico" di sviluppo a 6 mani che porterà alla versione 2.0.

2 motori di rendering SVG

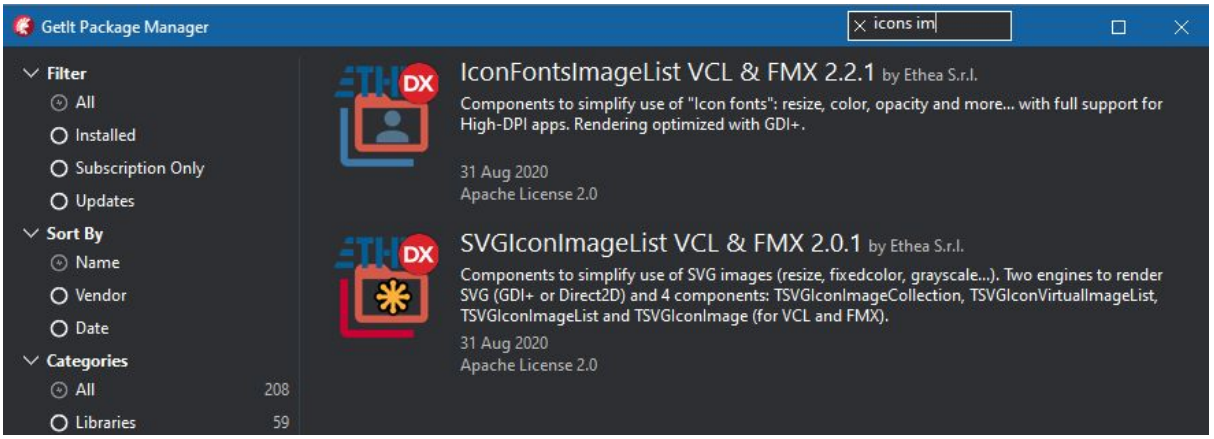
Una grossa novità della libreria SVGIconImageList 2.0 è la possibilità di utilizzare 2 motori di rendering diversi:

1) nativo con Direct2D (fornito da Windows 10): è il default se il S.O. lo fornisce
2) TSVG di Martin Walter con GDI+ (rivisto, corretto e ampiamente migliorato)

N.B. il supporto nativo di Windows 10 non copre il 100% delle funzionalità SVG, pertanto alcune icone non vengono renderizzate correttamente: in tal caso è possibile passare a TSVG cambiando una semplice direttiva di compilazione.

Pubblicazione su Get-It

La “consacrazione” delle librerie avverrà con la pubblicazione su Get-It il 2 Settembre:



IconFontsImageList e SVGImageList: caratteristiche e istruzioni per l'uso

In questa sezione illustreremo le caratteristiche delle due librerie in relazione alle tematiche legate allo sviluppo di applicazioni High-DPI.
Dato che sono molto simili e speculari per diversi aspetti, verranno trattate quasi sempre insieme, evidenziandone le peculiarità.

I Componenti

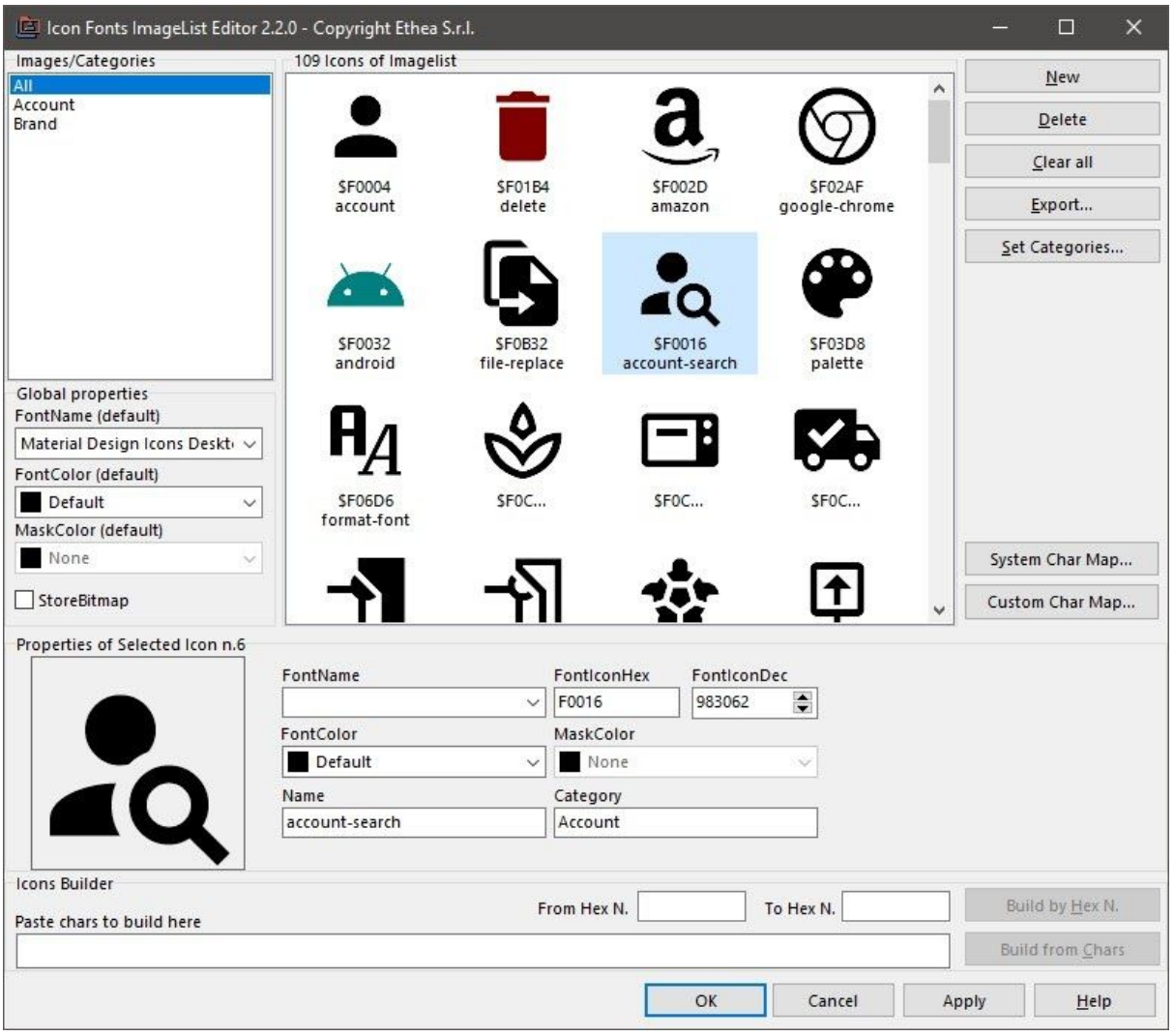
In questa tabella sono riassunti i componenti "gemelli" delle due librerie: in entrambe sono 4 e hanno un funzionamento molto simile:

	TIconFontsImageList TSVGIconImageList	Sono il "vecchio" componente derivato da TCustomImageList con la collezione di Icone al proprio interno.
	TIconFontImage TSVGIconImage	Sono un componente "grafico" per disegnare un'icona su una superficie, scalabile a qualsiasi dimensione.
	TIconFontsImageCollection TSVGIconImageCollection	Sono i "nuovi" componenti che contengono la collezione di icone disponibili e scalabili
	TIconFontsVirtualImageList TSVGIconVirtualImageList	Sono i "nuovi" componenti che si agganciano alla collezione per generare le icone di una certa dimensione.

Collection + VirtualImageList per vecchie versioni Delphi

Come già spiegato nella sezione dedicata, nello sviluppo di applicazioni High-DPI non è possibile fare a meno di utilizzare la logica Collection + VirtualImageList. Dato che in versione di Delphi precedenti alla 10.3 questa cosa non è disponibile, dopo una prima versione "classica" è stato fatto un refactoring per rendere disponibili questi componenti, che possono essere utilizzati a partire da Delphi 7 (IconFontsImageList) o dal Delphi XE4 (SVGIconImageList). Sull'esempio del component editor di ImageCollection e VirtualImagelist sono stati ridisegnati i nuovi component editor.

Il nuovo component editor di IconFontsImageList

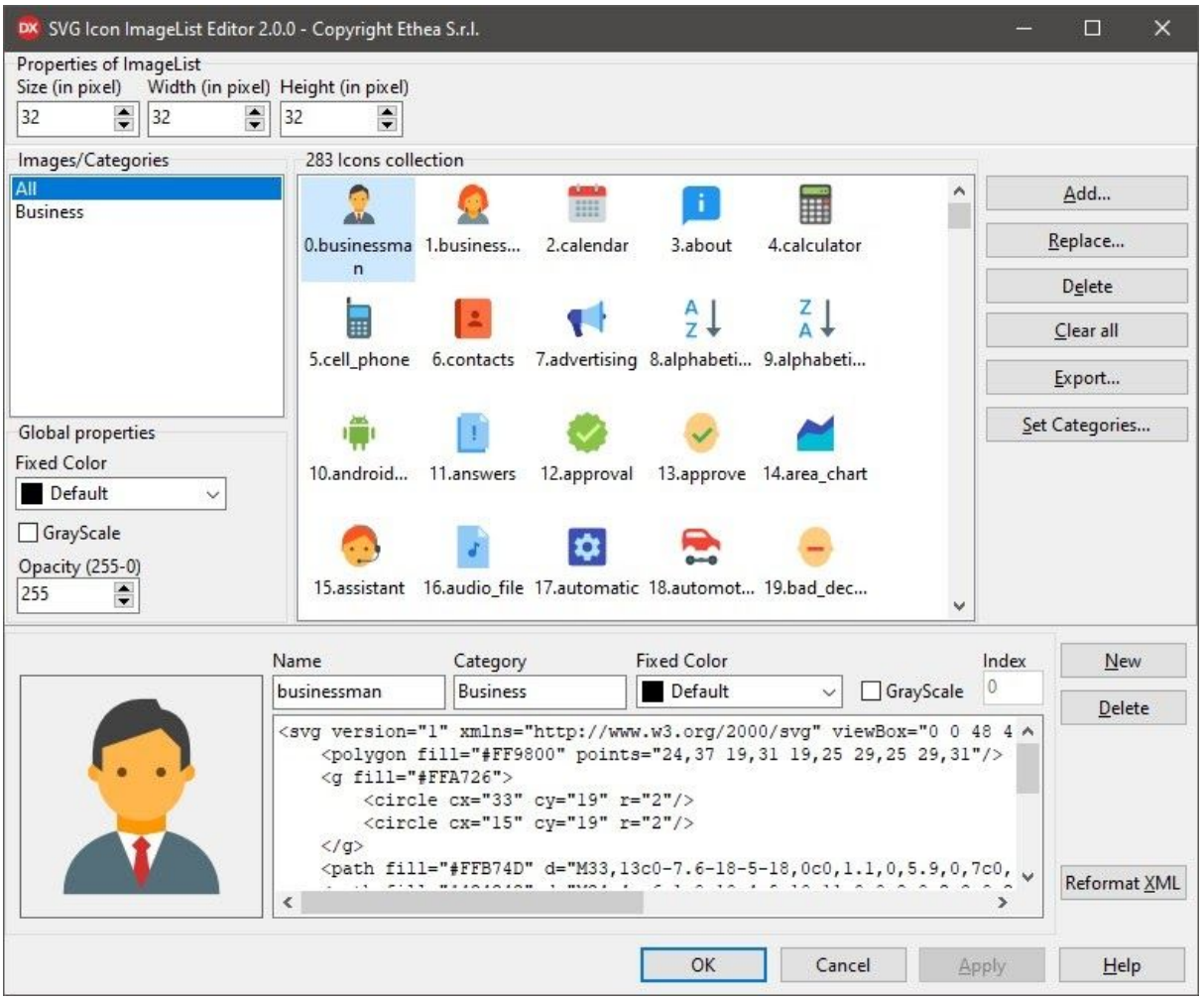


Il nuovo component editor è attivabile sia dal "vecchio" TIconFontsImageList che da TIconFontsVirtualImageList, ma anche da TIconFontsImageCollection: in questo caso non si vedrà il pannello superiore relativo alle dimensioni delle icone (come si vede in questa figura). Dal component editor è possibile impostare i valori globali:

- **Size** (o **Width** e **Height** se sono diversi) qualora lo si attivi dalla ImageList.
- **FontName** di default, da utilizzare per tutte le icone
- **FontColor** (**MaskColor** è abilitato solo per le vecchie versioni di Delphi prima di XE4 che non hanno il supporto GDI+ e che usano le masked-bitmap).
- **"New"**: crea una icona "vuota" dove è possibile indicare FontIconHex (oppure FontIconDec) cioè il "carattere" Unicode dell'icona.
- **"Delete"** e **"Clear All"** cancellano l'icona corrente o tutte.
- **"Set Categories"** permette di assegnare alle icone selezionate una categoria: cliccando nel box "Images/Categories" si filtrano le icone per categoria.
- Per ogni singola icona è possibile anche:
    - specificare un **FontName** alternativo a quello globale (se si usano icone appartenenti a font diversi)
    - specificare un **FontColor** (e **MaskColor**) alternativo a quello globale
    - definire **Name** e **Category**

Nella parte inferiore è sempre disponibile l'**Icons Builder**, per poter "importare" le icone dal Font indicando il range di valori esadecimali o il campo di **"Paste"** dalla CharMap.

Il nuovo component editor di SVGIconImageList

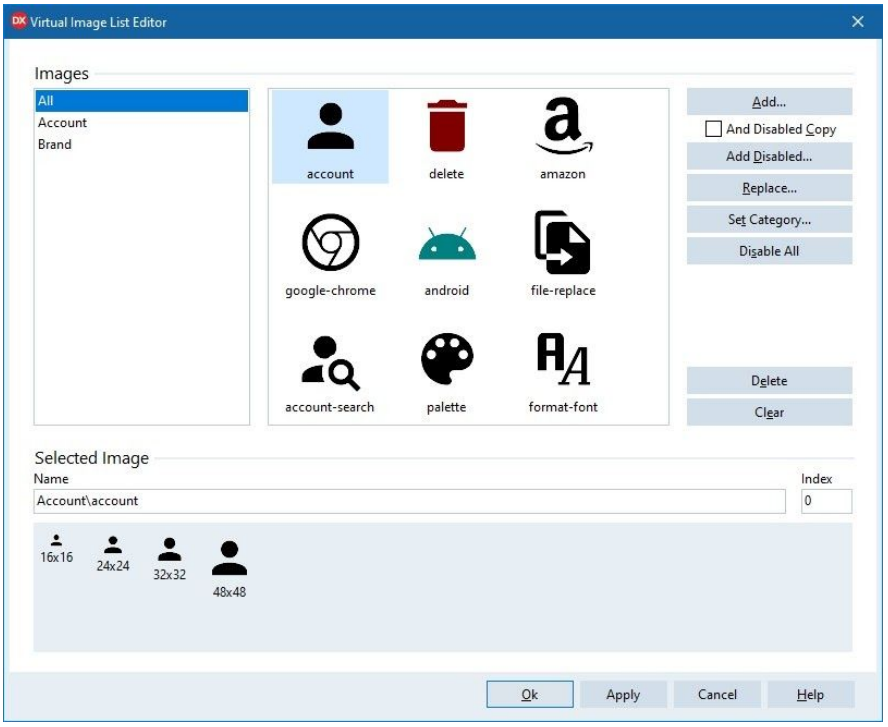


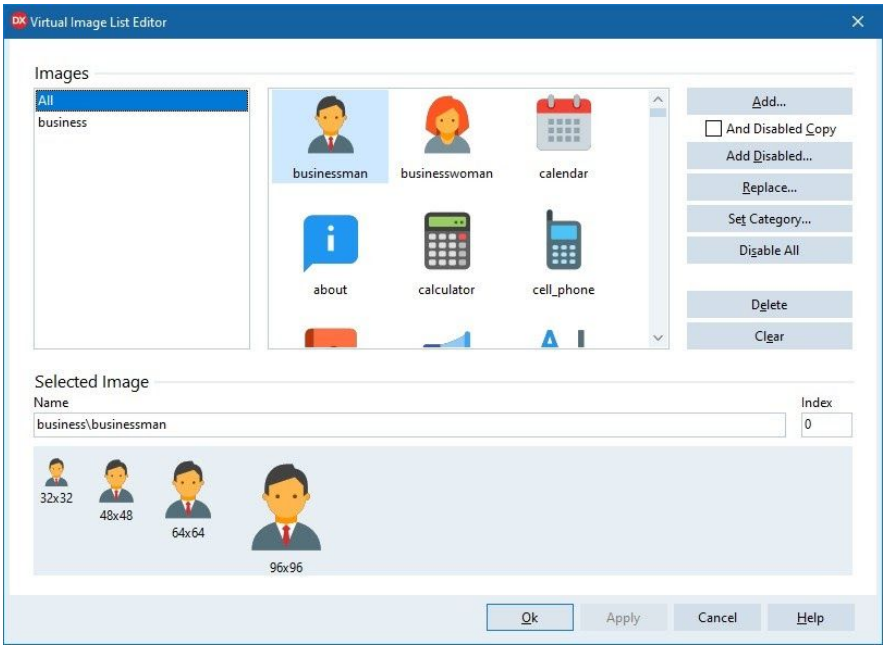
Il nuovo component editor è attivabile sia dal "vecchio" TSVGIconImageList che da TSVGIconVirtualImageList, ma anche da TSVGIconImageCollection: in questo caso non si vedrà il pannello superiore relativo alle dimensioni delle icone (visibile in questa figura).
Dal component editor è possibile impostare i valori globali:

- **Size** (o **Width** e **Height** se sono diversi) qualora lo si attivi da una ImageList.
- **FixedColor**, solo se si vuole "forzare" un unico colore a tutte le icone.
- **GrayScale**, solo se si vuole rendere "grigie" tutte le icone.
- **Opacity**, solo se si vuole applicarlo a tutte le icone.
- **"Add..."**: permette di caricare da files SVG da importare: verranno create le icone ed assegnato il "nome" a partire dal file.
- **"Replace…"** sostituisce le icone selezionate con altre da files.
- **"Delete"** e **"Clear All"** cancellano le icone selezionate o tutte.
- **"Export..."** salva su disco tanti files SVG
- **"Set Categories..."** permette di assegnare alle icone selezionate una categoria: cliccando nel box "Images/Categories" si filtrano le icone per categoria.
- Per ogni singola icona è possibile anche:
    - definire **Name** e **Category**
    - specificare un **FixedColor** alternativo a quello globale.
    - applicare **GrayScale**
    - Editare manualmente il testo SVG nel piccolo editor.
    - Riformattare il testo XML

Supporto alla VirtualImageList nativa

Come già detto, con versioni antecedenti alla 10.3 è possibile definire una collezione e utilizzare il componente IconFontsVirtualImageList o SVGIconVirtualImageList fornito.
Se invece si possiede una versione recente di Delphi è consigliabile utilizzare la VirtualImageList nativa che offre delle funzionalità superiori a quella implementata, come la possibilità di definire solo le icone necessarie alla form stessa (come già spiegato) e la proprietà "PreserveItems" che permette di preservare l'indice dell'icona, anche quando cambia la collezione.
Per questo motivo, la IconFontsImageCollection e la SVGIconImageCollection derivano da TCustomImageCollection (se compilate da 10.3 in poi), in modo tale da poter essere agganciate anche alla TVirtualImageList: in questo modo, il component editor nativo è in grado di selezionare le icone fornite dalla collection, come mostrato in queste figure:



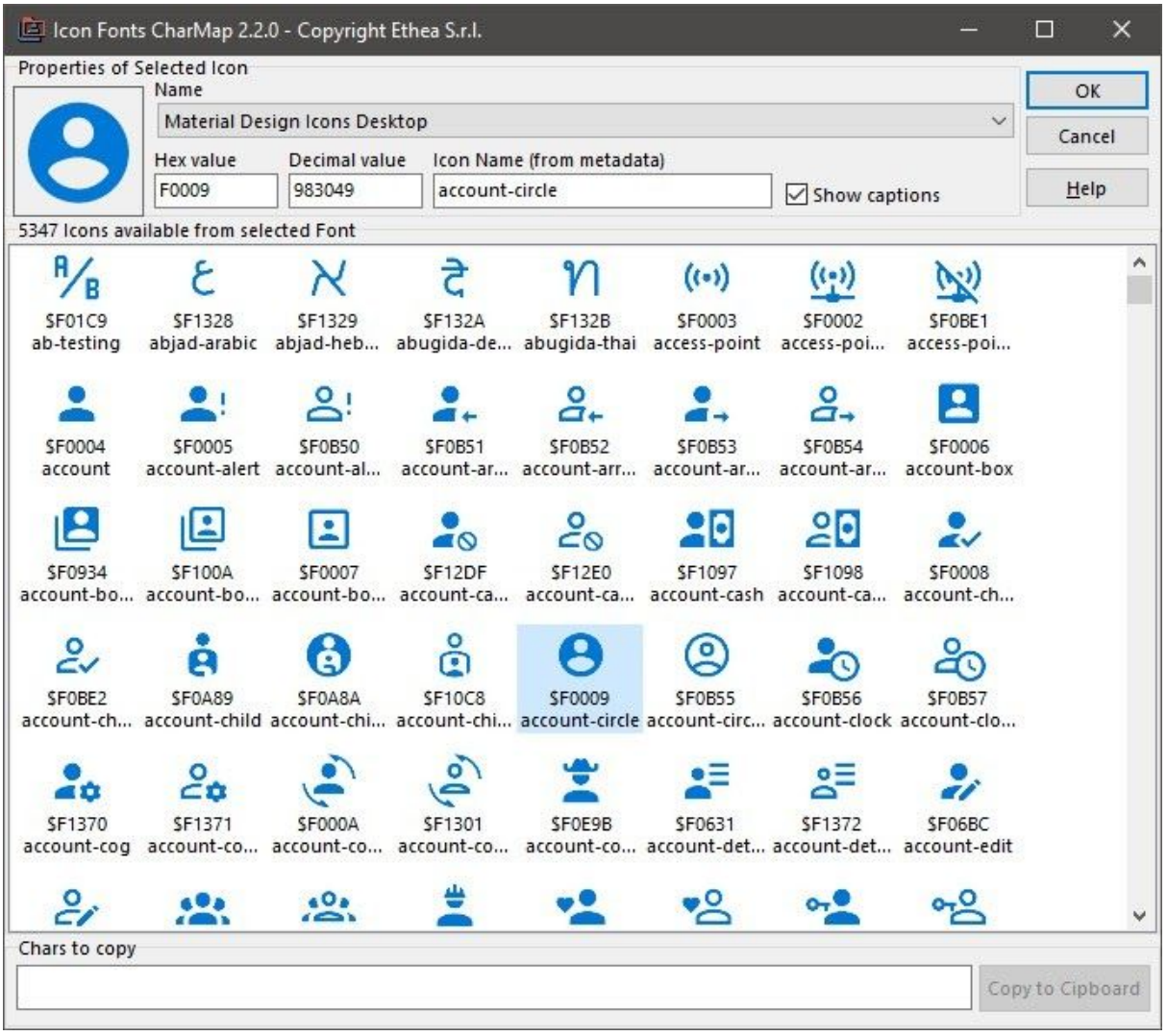


Utilities

In entrambe le librerie ci sono alcune utility per agevolare il lavoro di configurazione e preparazione della collezione di immagini.

IconFonts CharMap e catalogo Icone

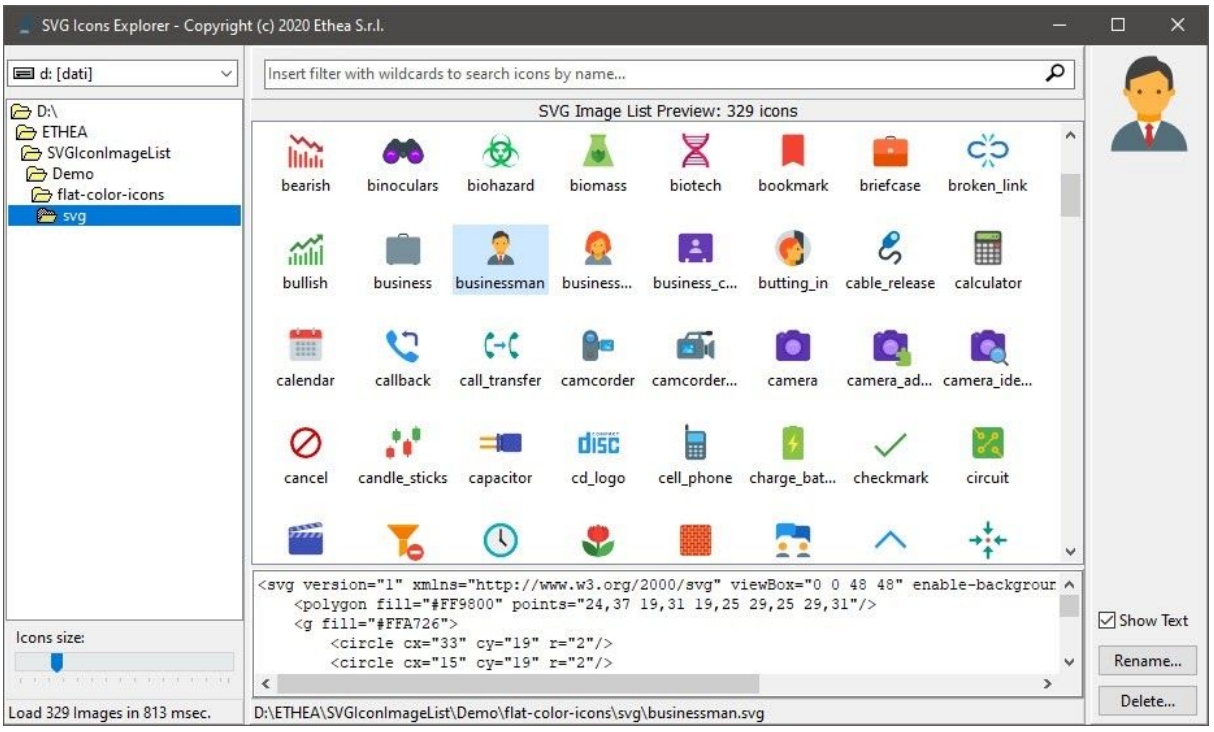
Per agevolare la preparazione delle Icone da collezionare è stata sviluppata una CharMap "custom", in quanto quella di sistema non è in grado di mostrare i caratteri speciali "surrogate-pair".



All'interno della CharMap è possibile cambiare il Font da visualizzare, selezionare le icone che si vogliono importare nella collezione con Ctrl+Click (e premere OK).
Infine esiste anche la possibilità di definire un catalogo di "metadati" delle icone per associarle ad un nome: nella figura, il font "Material Design Icons Desktop.ttf" è stato mappato nella unit unit Icons.MaterialDesign.pas generata con IconFontsImageListUtility di Andrea Magni (https://github.com/andrea-magni/IconFontsImageListUtility).

SVG Icons Explorer

Per agevolare la gestione delle icone SVG è stato sviluppata questa utility che permette di navigare nel file-system e visualizzare i file SVG direttamente come Icone, dato che l'explorer di Windows ancora non è in grado di farlo.



E' possibile anche filtrare le icone, vedere il testo del file e ridimensionarle.
Icons Explorer è di fatto un'esempio di una applicazione Delphi che utilizza proprio SVGIconImageList, il componente singolo, dato che essendo una applicazione "mono-finestra" non è necessario lo sdoppiamento collection+virtual.

Demos e documentazione

Entrambe le librerie sono corredate da una Demo che mostra le principali funzionalità:

- L'utilizzo delle icone tramite TToolbar, TButton, TActionList/TAction, TTreeView e TListView
- La possibilità di riscalare le Icone (attraverso una trackbar)
- La possibilità di adattare il colore allo stile VCL scelto
- L'utilizzo del componente TIconFontImage o TSVGIconImage per mostrare una singola icona anch'essa scalabile
- L'applicazione di attributi particolari a tutte le icone (Colore, GrayScale…)

N.B. quando la demo viene compilata con Delphi 10.3 o superiore, utilizza una VirtualImageList nativa al posto di quella fornita.

Per approfondire altri aspetti, entrambe le librerie hanno una sezione "Wiki" molto completa agli indirizzi:
https://github.com/EtheaDev/IconFontsImageList/wiki

https://github.com/EtheaDev/SVGIconImageList/wiki

Ringraziamenti

Un particolare ringraziamento va a **Vincent Parrett** (autore di FinalBuilder) e **Kiriakos Vlahos** (autore di PyScripter) che hanno contribuito in maniera sostanziale alla crescita della libreria SVGIconImageList: il successo di queste librerie è dovuto anche a loro, che mi hanno aiutato a fare la transizione da “normale” a “Virtual” ed hanno contribuito al test e al bug-fixing.

Personalmente devo anche ringraziare Embarcadero, che con Delphi mi ha permesso di trasformare la mia passione per la programmazione in una professione. La società che ho fondato (Ethea S.r.l. - https://ethea.it/) è Technology Partner e cura anche altri progetti Open-Source come InstantObjects e Kitto (https://github.com/EtheaDev).

FireMonkey version

Si, di entrambe le librerie esiste anche una versione per FireMonkey… ma lo vedremo in un prossimo articolo.

Copyright © 2020 - Ethea S.r.l. - tutti i diritti sono riservati