Thema mit Navigation öffnen

Sie sind hier: Release Notes IDM 6 > 1 A.06.03.a

1 A.06.03.a

1.1 HighDPI Unterstützung

Die Auflösung moderner Bildschirme wächst stetig. Damit ändern sich auch die Anforderungen an Design und Qualität der Bedienelemente und Oberflächen. Probleme sind dabei häufig, dass Anwendungen viel zu klein erscheinen und Text und Grafiken unscharf oder pixelig dargestellt werden. Die im System eingestellte Auflösung, DPI und Skalierung sowie Fonts und Größe und Qualität der Bilder beeinflussen das Darstellungsbild einer Anwendung. Damit bestehende Anwendungen auch auf hochauflösenden Bildschirmen detailreich und scharf dargestellt werden, müssen diese auf HighDPI vorbereitet werden.

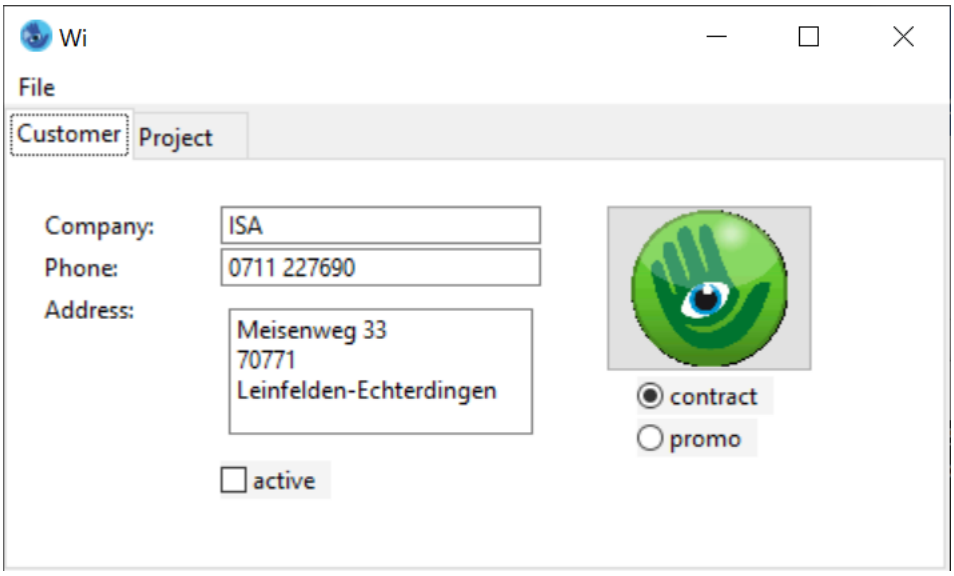

Abbildung 23-1: *ohne HighDPI Unterstützung*

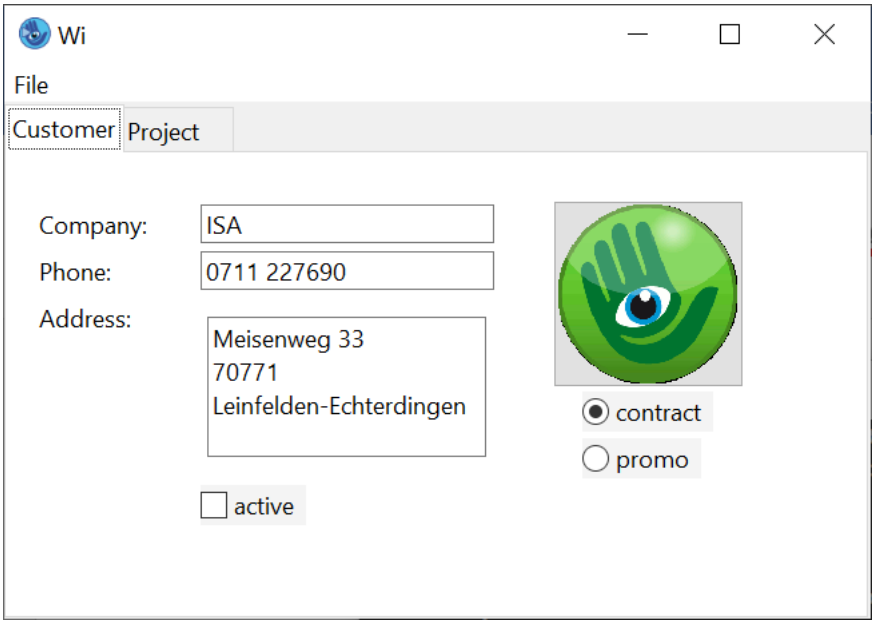

Abbildung 23-2: *mit HighDPI Unterstützung*

Der IDM für MOTIF, QT und MICROSOFT WINDOWS ab Version A.06.03.a (unter MICROSOFT WINDOWS nur der IDM FÜR WINDOWS 11) bietet nun die geeigneten Werkzeuge um hochauflösende Bildschirme zu unterstützen und Anwendungen zukunftssicher zu gestalten. Außerdem wurde das Arbeiten mit mehreren Bildschirmen verbessert.
IDM-Anwendungen beachten nun die DPI-Einstellungen und Skalierungsfaktoren des Systems. Bisherige Probleme wie pixelige, matschige oder zu klein angezeigte Anwendungen oder verringerte Schriftgrößen bei hohen Auflösungen und geringer Bildschirmgröße (die Effekte können je nach System variieren) treten somit nicht mehr auf.
Der IDM passt Geometrie, Layout, Schriftgrößen, Grafiken sowie Cursor nun automatisch an die entsprechende Auflösung und vom System bestimmte DPI an.
Der Skalierungsfaktor basiert dabei auf den Systemeinstellungen der jeweiligen Desktop-Umgebung.
Es steht dem Anwender aber frei durch verschiedene neue Attribute, Optionen und Mechanismen andere Vorgaben zu machen bzw. gestalterisch einzugreifen.

1.1.1 Layout-Ressourcen

Bisher waren die verfügbaren, vordefinierten Ressourcen beim IDM immer WSI- bzw. systemabhängig und bedurften bei Übertragung auf andere Systeme immer aufwändiger Anpassungen. Der IDM A.06.03.a bietet nun erstmals WSI-unabhängige und HighDPI-konforme, vordefinierte Layout-Ressourcen an. Dazu gehören folgende Ressourcen, die auf allen Systemen in ähnlicher Ausprägung bzw. Bedeutung verfügbar sind: **font (Zeichensatz)**, **color (Farbe)**, **cursor**.
Das bietet den enormen Mehrwert, dass Dialoge, die komplett auf die neuen UI-Ressourcen umgestellt wurden, nun einfach und ohne Aufwand auf andere Systeme übertragen werden können. Erkennbar sind diese Ressourcen über das Namensschema ***UI*_FONT***, ***UI*_COLOR*** bzw. ***UI*_CURSOR***.

Beispiel:

```
module Resc

font FontNormal "UI_FONT";
font FontBig "UI_FONT", 16;
font FontFixed "UI_FIXED_FONT";

cursor CurStop "UI_STOP_CURSOR";

!! Background color for group objects
color ColGrp "UI_BG_COLOR";
!! Background color for input objects
color ColInp "UI_INPUTBG_COLOR";
```

Die UI_-Ressourcennamen sind ebenfalls – wie alle anderen vordefinierten Ressourcennamen - am ***setup***-Objekt über die Attribute *.colorname[integer]*, *.fontname[integer]*, *.cursorname[integer]* abfragbar.

Cursor

Die einheitlichen Cursor-Ressourcen haben das Namensschema ***UI*_CURSOR*** und sind wie folgt definiert:

UI_CURSOR	Standardcursor meist Pfeil
UI_IBEAM_CURSOR	Edittext-Einfügemarke
UI_WAIT_CURSOR	Anzeige, dass die Anwendung beschäftig ist, „Eieruhr“
UI_CROSS_CURSOR	Kreuz
UI_UP_CURSOR	Pfeil nach oben
UI_SIZEDIAGF_CURSOR	Sizing-Pfeil von links oben nach rechts unten
UI_SIZEDIAGB_CURSOR	Sizing-Pfeil von links unten nach rechts oben
UI_SIZEVER_CURSOR	Sizing-Pfeil von oben nach unten
UI_SIZEHOR_CURSOR	Sizing-Pfeil von links nach rechts
UI_MOVE_CURSOR	Verschiebe-Pfeil in alle Richtungen
UI_STOP_CURSOR	Verbotszeichen (durchgestrichener Kreis)
UI_HAND_CURSOR	Handsymbol (Zeigefinger dient zum Zeigen)
UI_HELP_CURSOR	Pfeil mit Fragezeichen (für Kontexthilfe-Modus)

Fonts

Die einheitlichen Font-Ressourcen haben das Namensschema ***UI*_FONT*** und sind wie folgt definiert:

UI_FONT	Schriftart, die standardmäßig für Oberflächenelemente verwendet wird (Default oder Dialog Font).
UI_FIXED_FONT	Schriftart mit einer festen Buchstabenbreite.
UI_MENU_FONT	Schriftart, die für Menüs verwendet wird.
UI_TITLE_FONT	Schriftart, die in der Fensterüberschrift verwendet wird.
UI_SMALLTITLE_FONT	Schriftart, die in einer kleinen/niedrigen Fensterüberschrift verwendet wird.
UI_STATUSBAR_FONT	Schriftart, die in der Statusbar verwendet wird.
UI_NULL_FONT	Verhalten als wenn keine Font-Vorgabe gemacht würde (null). Das WSI benutzt den für das Oberflächenelement vorgesehenen Standard-Systemfont oder einen Systemfallback-Font.

Color

Die Farb-Ressourcen haben das Namensschema ***UI*_COLOR*** und sind wie folgt definiert:

UI_BG_COLOR	Farbe, die standardmäßig für allgemeine Hintergründe von (meist Gruppierungs-) Objekten verwendet wird. Verwendung bei Objektklassen mit Attribut .bgc/.bgc[], z.B. Fenster, Groupbox, Notepage, Statictext, Pushbutton...
UI_FG_COLOR	Farbe, die standardmäßig für allgemeine Vordergründe/Textfarbe von (meist Gruppierungs-) Objekten verwendet wird. Verwendung bei Objektklassen mit Attribut .fgc/.fgc [], z.B. Fenster, Groupbox, Notepage, Statictext, Pushbutton...

UI_INPUTBG_COLOR	Farbe, die standardmäßig bei Hintergründen von Eingabe-/Auswahl-Objekten verwendet wird. Verwendung z.B. bei Edittext, Listen, Tabellen, Poptext, Treeview…(z.B. Objektklassen mit Attributen .bgc/.textbgc)
UI_INPUTFG_COLOR	Farbe, die standardmäßig bei Vordergründen von Eingabe-/Auswahl-Objekten verwendet wird, wie z.B. Edittext, Listen, Tabellen, Poptext, Treeview (z.B. Objektklassen mit Attributen .fgc/.textfgc)
UI_BUTTONBG_COLOR	Farbe, die standardmäßig beim Hintergrund von Button-artigen Objekten, z.B. Pushbutton oder Image, verwendet wird (z.B. Objektklassen mit Attribut .bgc/.imagebgc).
UI_BUTTONFG_COLOR	Farbe, die standardmäßig beim Vordergrund von Button-artigen Objekten, z.B. Pushbutton oder Image, verwendet wird (z.B. Objektklassen mit Attribut. fgc/.imagefgc).
UI_BORDER_COLOR	Farbe, die standardmäßig als Rahmen, Rand oder Begrenzung verwendet wird (z.B. an .bordercolor).
UI_ACTIVEBG_COLOR	Farbe, die standardmäßig als Hintergrundfarbe des aktiven Elements z.B. bei Listen, Poptext, Tabellen, ggf. Progressbar, Markierungen etc. verwendet wird. Meist Invertierung der normalen Vorder- und Hintergrundfarben des Widgets (z.B. Objektklassen mit Attribut .bgc).
UI_ACTIVEFG_COLOR	Farbe, die standardmäßig als Vordergrund-/Textfarbe des aktiven Elements z.B. bei Listen, Poptext, Tabellen, ggf. Progressbar, Markierungen etc. verwendet wird. Meist Invertierung der normalen Vorder- und Hintergrundfarben des Objekts (z.B. Objektklassen mit Attribut .fgc).
UI_TITLEBG_COLOR	Farbe, die standardmäßig als Hintergrundfarbe bei Titelleisten oder Umrandungen von Fenstern/Dialogen verwendet wird, sofern unterstützt (z.B. an Attribut .titlebgc).
UI_TITLEFG_COLOR	Farbe, die standardmäßig als Vordergrund-/Textfarbe bei Titelleisten oder Umrandungen von Fenstern/Dialogen verwendet wird, sofern unterstützt (z.B. an Attribut .titlefgc).
UI_MENUBG_COLOR	Farbe, die standardmäßig als Hintergrundfarbe bei Menüs verwendet wird, sofern unterstützt (z.B. an Attribut .titlebgc).
UI_MENUFG_COLOR	Farbe, die standardmäßig als Vordergrund-/Textfarbe bei Menüs verwendet wird, sofern unterstützt (z.B. an Attribut .titlefgc).
UI_HEADERBG_COLOR	Farbe, die standardmäßig als Hintergrundfarbe bei Tabellenheadern – sofern unterstützt – verwendet wird.
UI_HEADERFG_COLOR	Farbe, die standardmäßig als Vordergrund-/Textfarbe bei Tabellenheadern – sofern unterstützt – verwendet wird. Verwendung bei Tablefield (z.B. an Attribut .rowheadfgc, .cloheadfgc).
UI_NULL_COLOR	Verhalten, als wenn keine Farbe (null) gesetzt wäre. WSI zeichnet nach seiner Standardpalette.

Es kann vorkommen, dass sich nicht alle UI*COLORS voneinander unterscheiden. Das liegt daran, dass die meisten Desktopfarben aus wenigen Basisfarben "berechnet" werden.

Hinweise Motif:

Neben den ***UI*_COLOR*** Farben wurden die Systemnamen für Farb-Ressourcen vervollständigt. Neu hinzugekommen sind:

- INACTIVE_BACKGROUND
- INACTIVE_FOREGROUND
- INACTIVE_TOP_SHADOW
- INACTIVE_BOTTOM_SHADOW
- INACTIVE_SELECT
- SECONDARY_BACKGROUND
- SECONDARY_FOREGROUND
- SECONDARY_TOP_SHADOW
- SECONDARY_BOTTOM_SHADOW
- SECONDARY_SELECT

Hinweise Windows:

Die ***UI*_COLOR*** Farben greifen auf die Windows Systemfarben zu. Hierbei ist zum einen zu beachten, dass mit Einführung der Visual Styles von den einzelnen Objekten nicht mehr die Systemfarben, sondern die durch das Theme definierten Farben verwendet werden. Je nach ausgewähltem Theme kann es somit starke Diskrepanzen geben. Deshalb sollten unter Windows nach Möglichkeit keine Farben (oder besser die ***UI_NULL_COLOR***) gesetzt werden.
Zum anderen ist zu beachten, dass mit Windows 10 die Farbvielfalt verringert wurde. Die Farben ***UI_HEADERBG_COLOR***, ***UI_HEADERFG_COLOR***, ***UI_MENUBG_COLOR, UI_MENUFG_COLOR***, ***UI_TITLEBG_COLOR*** und ***UI_TITLEFG_COLOR*** werden zum Beispiel nicht mehr unterstützt. Das heißt, dass diese Farben für den Anwender zwar noch verfügbar und nutzbar sind, sie aber aktuell von keinem Objekt defaultmäßig verwendet werden.

Hinweise Qt:

Die ***UI*_COLOR*** Farben greifen auf die von Qt angebotenen Farben der Standard-Palette zu. Für die Farbausprägung wird dabei die ColorGroup Normal bzw. Active zugrunde gelegt. Die Farben der Standard-Palette wiederum werden durch den aktuell eingestellten Systemstyle vorgegeben.

Neben den ***UI*_COLOR*** Farben wurden die vordefinierten Farben um weitere vom Qt-Toolkit angebotenen ColorRoles vervollständigt. Neu hinzugekommen (in allen Ausprägungen der ColorGroups) sind:

- BRIGHTTEXT
- ALTERNATEBASE
- TOOLTIPBASE
- TOOLTIPTEXT
- LINK

- LINKVISITED
- PLACEHOLDERTEXT

1.1.2 Erweiterung an der Tile-Ressource

Die bestehenden Skalierungsoptionen an der Tile-Ressource wurden erweitert. Es kann nun ein *.scalestyle* angegeben werden, der bestimmt wie ein Tile skaliert werden soll. So können Muster (tiles) nun nach folgenden Merkmalen skaliert werden:

- automatisch
- um einen bestimmten Faktor
- proportional
- in alle Richtungen frei
- an den DPI-Wert anpassen und keine Skalierung

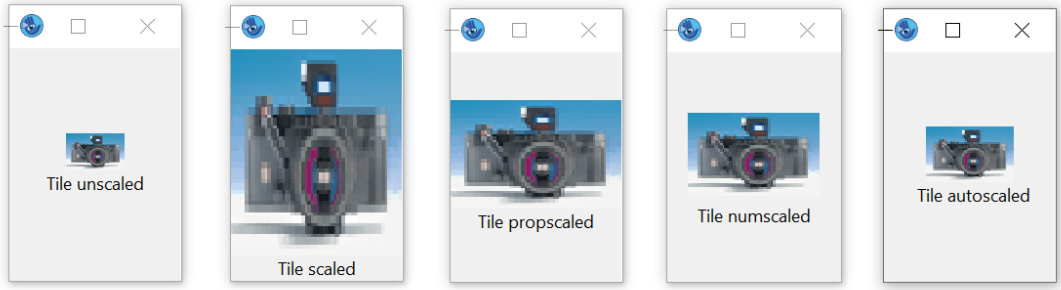

Abbildung 23-3: *das gleiche Bild mit verschiedenen Ausprägungen des Attributs .scalestyle bei einer Systemskalierung von 150%*

In der Regelsprache stehen dabei folgende Werte zur Auswahl:

Definition am Tile	Dynamische Setzung	Bedeutung
noscale	scalestyle_none	Das Muster bzw. Bild wird nicht skaliert., *.tiledpi* hat keine Auswirkungen.
scale	scalestyle_any	Höhe und Breite des Musters bzw. Bildes werden voll auf die verfügbare Fläche vergrößert.
propscale	scalestyle_prop	Höhe und Breite des Musters bzw. Bildes werden auf die verfügbare Fläche vergrößert, wobei Höhen- und Breitenproportionen des Musters bzw. Bildes auf jeden Fall erhalten bleiben. D.h. es können oben und unten bzw. links und rechts Freiflächen entstehen.
numscale	scalestyle_num	Die Skalierung des Musters bzw. Bildes wird durch einen numerischen Skalierungsteiler vorgenommen. Dabei wird die Skalierung in Viertel-Schritten vorgenommen, also 1.25-fach, 1.5-fach, 1.75-fach, 2-fach, 2.25-fach, 2.5-fach, usw. Eine Verkleinerung finded bis maximal 0.25-fach statt.
dpi	scalestyle_dpi	Das Muster bzw. Bild wird immer entsprechend der eingestellten Bildschirmskalierung skaliert.
Keine Angabe	scalestyle_auto	Das Muster bzw. Bild wird entsprechend der eingestellten Bildschirmskalierung skaliert. Eine zur Vorgängerversion kompatible Skalierung findet statt. **Defaultwert**

Der ***scalestyle*** kann dabei entweder direkt bei der **Definition des Tiles** oder aber als **dynamisches Attribut** angegeben werden.

Tile-Definition:

```
tileSpec ::= <tileBitmap> | "<Dateipfad>" | "<Grafikressource>" { scale | noscale | numscale | propscale | dpiscale} ;
```

Dynamisches Attribut:

	Bezeichner		Datentyp
Regelsprache	**.scalestyle**		enum
C	AT_scalestyle		DT_enum
COBOL	AT-scalestyle		DT-enum
Klassifizierung	Objektspezifisches Attribut		
Objekte	***tile***		
Zugriff	get, set	Standardwert	scalestyle_auto
changed-Ereignis	nein	Vererbung	–

Beispiel:

```
dialog D
tile Ti_Rect_Pattern 5,5,
  "#####",
  "# #",
  "# #",
  "# #",
```

```
  "#####" noscale;          // entspricht scalestyle_none
tile Ti_BG „bg.gif“ scale;  // entspricht scalestyle_any
tile Ti_Logo „logo.gif“;    // Default: scalestyle_auto
...
on dialog start {
  if setup.scale > 100 then
  Ti_Logo.scalestyle := scalestyle_prop;   // dynamisch
endif
}
```

Siehe auch Kapitel „tile (Muster)“ im „Ressourcenreferenz“-Handbuch.

1.1.3 Erweiterung der Font-Ressource

Die Font-Ressource hat eine neue Eigenschaft *.propscale*. Dies steuert, ob das horizontale und vertikale Raster proportional auf den maximalen Wert gesetzt werden soll, welcher durch die Werteberechnung von xraster und yraster des Font-Rasters ermittelt wird.

Die ***propscale***-Eigenschaft kann dabei entweder direkt bei der **Definition des Fonts** oder aber als **dynamisches Attribut** angegeben werden.

Der Divisor dient der Verfeinerung des Rasters.

Font-Definition:

```
fontspec ::=
...
...
{ x:= * <xscale> % + <xoffset> | / <xdivider>} { y:= * <yscale> %
+ <yoffset> | / <ydivider>}
{ propscale }
{ r:= "<RefString >" } ;
```

Dynamisc hes Attribut:

	Bezeichner		Datentyp
Regelsprache	**.propscale**		boolean
C	AT_propscale		DT_boolean
COBOL	AT-propscale		DT-boolean
Klassifizierung	Objektspezifisches Attribut		
Objekte	***font***		
Zugriff	get, set	Standardwert	false
changed-Ereignis	nein	Vererbung	–

Siehe auch Kapitel „font (Zeichensatz)“ im „Ressourcenreferenz“-Handbuch.

1.1.4 Erweiterung am Setup-Objekt

	Bezeichner		Datentyp
Regelsprache	**.tiledpi**		integer
C	AT_tiledpi		DT_integer
COBOL	AT-tiledpi		DT-integer
Klassifizierung	Objektspezifisches Attribut		
Objekte	***setup***		
Zugriff	get, set	Standardwert	96
changed-Ereignis	nein	Vererbung	–

Dieses Attribut bestimmt für welche Auflösung die Bilder/Muster entworfen wurden. Die Größe eines Bildes/Musters wird ausgehend von diesem Wert auf den aktuell geltenden DPI-Wert umgerechnet.

Siehe auch Kapitel „setup“ im „Objektreferenz“-Handbuch.

1.1.5 Erweiterungen am Image-Objekt

Für eine bessere Ausrichtung und erweiterte Layoutmöglichkeiten können nun über die folgenden Attribute am Image-Objekt die Ränder bzw. Abstände gesteuert werden:

	Bezeichner	Datentyp
Regelsprache	**.xmargin**	integer
C	AT_xmargin	DT_integer

	Bezeichner		Datentyp
COBOL	AT-ymargin		DT-integer
Klassifizierung	Objektspezifisches Attribut		
Objekte	***image***		
Zugriff	get, set	Standardwert	0
changed-Ereignis	nein	Vererbung	ja

Dieses Attribut bestimmt den horizontalen Abstand zwischen Rand und Darstellungsbereich.

	Bezeichner		Datentyp
Regelsprache	**.ymargin**		integer
C	AT_ymargin		DT_integer
COBOL	AT-ymargin		DT-integer
Klassifizierung	Objektspezifisches Attribut		
Objekte	***image***		
Zugriff	get, set	Standardwert	0
changed-Ereignis	nein	Vererbung	ja

Dieses Attribut bestimmt den vertikalen Abstand zwischen Rand und Darstellungsbereich.

Siehe auch Kapitel „image (Bild)“ im „Objektreferenz“-Handbuch.

1.1.6 Unterstützung von HiDPI Bild-Varianten

Für eine optimale Gestaltung der Oberfläche bzgl. Bilder und Applikations-Icons durch Ausnutzung der höheren Auflösung sind für die Skalierungsstufe angepasste Bilder sinnvoll.
Bisher gab es hier nur für die Fenstersysteme WINDOWS und QT eingeschränkte Möglichkeiten. Windows bietet hier bislang zur Unterstützung den ICON/BITMAP-Mechanismus und Qt den Icon-Ressource-Mechanismus. Beide Mechanismen ermöglichen das Vorhalten mehrerer Auflösungsstufen, sind jedoch begrenzt was Bildtyp oder Verwendung angeht. Einen solchen Mechanismus gibt es für MOTIF nicht.
Daher wurde mit dem IDM A.06.03.A eine Möglichkeit zum Laden mehrerer Auflösungsstufen geschaffen, die gleichermaßen für WINDOWS, QT und MOTIF angewandt werden kann und somit auch systemunabhängig ist.
Die Tile-Verwaltung vom IDM war bisher nur in der Lage eine Bilddatei zu laden (bspw.: tile Ti „smiley.gif“) und konnte nicht mehrere Auflösungsstufen der Datei vorhalten (bis auf o.g. Ausnahmen). Liegt nun eine Skalierung der Anwendung vor, versucht der neue Lademechanismus für Tile-Dateien nun zuerst die Bilddaten für die entsprechende Skalierung zu laden. Dazu wird in Anlehnung an den QT-Mechanismus die Endung ***@nx*** an den Dateinamen und vor dem Suffix angehängt. Dabei steht ***n*** für die Skalierungsstufe zwischen *2* und *9*. Die Zahl *2* entspricht dabei einer Skalierung von *150%-249%*, *3* von *250%-349%*, usw.
Sinnvollerweise sollten die Größenänderungen der Varianten Bilder der Skalierungsstufe entsprechen, wobei eine Überprüfung durch den IDM nicht erfolgt. Also z.B. Orginalbild „smiley.gif“ (32x32 Pixel), smiley@2x.gif (64x64 Pixel), smiley@3x.gif (96x96 Pixel), smiley@4x.gif (128x128 Pixel).

Beispiel:

```
tile Ti „smiley.gif“;  // bei einer Skalierung von 200% wird – sofern
                       // vorhanden – vom IDM automatisch das Bild
                       // smiley@2x.gif geladen (Windows, Motif, Qt)

                       // bei einer Skalierung von 300% wird – sofern
                       // vorhanden – vom IDM automatisch das Bild
                       // smiley@3x.gif geladen (Windows, Motif, Qt)

tile Ti2 „smiley.ico“  // nur Windows! ICON-Mechanismus von Window nimmt
                       // das für die Auflösungsstufe am besten Bild aus
                       // der .ico-Datei

tile Ti3 „:/smiley“    // nur Qt! Der Resource-Mechanismus von Qt nimmt das
                       // für die Auflösungsstufe am besten passende Bild
                       // mit dem hier angegebenen Alias aus der Ressource-
                       // Datei
```

1.1.7 Startoptionen

-IDMscale <integer>

Über die Startoption **-IDMscale <integer>** kann die Skalierung durch den IDM aktiviert oder deaktiviert werden. Unter QT und MOTIF gibt der Wert die Skalierung in % an.

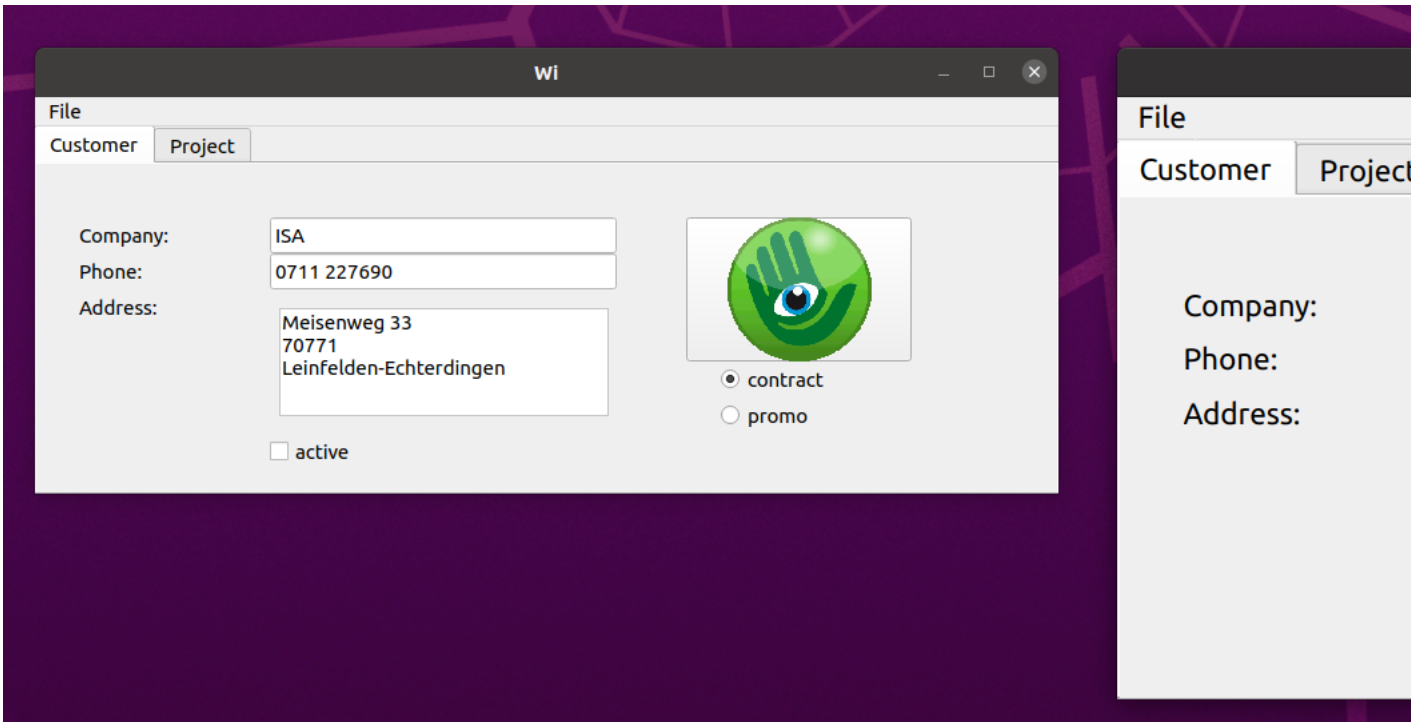

Abbildung 23-4: *links normal gestartet (Systemskalierung 100%), rechts mit 150% Skalierung gestartet*

Hinweis

Diese Option sollte unter Microsoft Windows nicht verwendet werden. Die DPI-Awareness ist eine Eigenschaft der Anwendung und sollte in einer Manifest-Datei spezifiziert werden. Für Testzwecke kann die DPI-Awareness eingeschalten (Wert: *1*) oder ausgeschalten (Wert: *0*) werden.

Achtung:

Es wird empfohlen eine Skalierung > 0 und < 100% nicht zu verwenden, da es hier zu Einschränkung in der Darstellung und Bedienung von Objekten kommen kann.

-IDMtiledpi <integer>

Grafiken werden automatisch entsprechend des eingestellten Vergrößerungsfaktor skaliert. Es wird hierbei davon ausgegangen, dass die Bilder für einen DPI-Wert von *96* entworfen wurden. Sind die Bilder der Anwendung für eine höhere Auflösung entworfen worden, kann diese über die Startoption **-IDMtiledpi <integer>** eingestellt werden. Die Größe eines Bildes wird dann ausgehend von diesem Wert auf den aktuell geltenden DPI-Wert umgerechnet und dementsprechend skaliert.

Ebenso kann diese Einstellung direkt am ***setup***-Objekt mit dem Attribut *.tiledpi* (siehe **Erweiterung am Setup-Objekt**) vorgenommen werden.

1.1.8 Installationshinweise

Um eine Anwendung für verschiedenen Auflösungen vorzubereiten, kann es nötig sein die Grafiken für die verschiedene Skalierungsstufen vorzuhalten. Wer dies in Anspruch nehmen möchte und nicht den ICON-Mechanismus von Windows oder den Resource-Mechanismus von Qt nutzt, der sollte darauf achten die Bilderdateien entsprechend der @-Namensgebung (siehe Kapitel „Unterstützung von HiDPI Bild-Varianten“) mit seiner Anwendung mit auszuliefern bzw. diese ggf. in die Installationspakete mit aufzunehmen.

1.1.9 Geometrie und Koordinaten

Koordinaten und Größenangaben werden vom IDM passend zum Skalierungsfaktor des Systems transformiert um eine zu den Systemeinstellungen konsistente Darstellung zu gewährleisten.

Die Geometrieeinheiten werden dabei wie bisher in **IDM-Pixelwerten** oder *Raster* gesetzt.

Die **IDM-Pixelwerte** werden intern entsprechend dem Skalierungsfaktor in **reale Pixelwerte** hochgerechnet und umgesetzt. Dadurch wird die Anwendung als Ganzes skaliert und Verhältnisse und Proportionen der Objekte zu- und untereinander beibehalten.

Diese Konvertierung geschieht in beide Richtungen. Beim Abfragen von Größe und Position von Objekten wie auch bei der Event-Verarbeitung werden die **realen Pixelwerte** vom IDM ebenso wieder bereinigt in **IDM-Pixelwerten** angegeben. Der Vergrößerungsfaktor ist dann wieder herausgerechnet. Dies betrifft alle pixelorientierten Geometrieattribute.

Rasterwerte sind entweder an den verwendeten *(HighDPI-fähigen) Zeichensatz* geknüpft oder werden ebenfalls in **IDM-Pixelwerten** gesetzt und dann hoch skaliert. Die *Rastereinheiten* können basierend auf der zugrundeliegenden Schriftart ebenfalls zu **IDM-Pixelwerten** ermittelt werden.

1.1.10 Unterstützung hoher Auflösungen unter Motif

Das Motif-Toolkit selbst enthält zwar keine besondere Technologie zur Unterstützung hoher Auflösungen, wenn aber das verwendete X-Display die Xrender-Extension implementiert hat und eine Unterstützung von XFT-Fonts mit den entsprechenden Schriftarten vorliegt, können IDM-Anwendungen für Motif auf HiDPI-Bildschirmen, unter Berücksichtigung des Skalierungsfaktors des Desktops, betrieben werden.

Wie kann geprüft werden, ob auf dem verwendeten X-Display/Desktop eine Unterstützung vorliegt? Dazu kann die Serverinformation des X-Displays über die Option **-IDMserver:** erfragt werden.

```
$ idm -IDMserver
ServerVendor The X.Org Foundation
VendorRelease 12013000
Motif at compiletime @(#)Motif Version 2.3.8
Motif at runtime 2003
XRenderVersion 11 (active)
```

```
XFT Support yes (active)
Scaling 200% (active)
Screen dpi 96 (tile dpi: 96, scaled dpi: 192)
```

Die Version der Xrender-Extension und deren Nutzbarkeit (active) wird darin ebenso aufgelistet wie das Vorhandensein von XFT und seiner Nutzbarkeit. Die „Scaling“-Zeile zeigt den Skalierungsfaktor an, der in den Desktop-/Anzeigeeinstellungen vom Anwender gesetzt wurde.

Die DPI-Information welche X liefert, ist zumeist der Standardwert von 96 und spiegelt damit nicht unbedingt den real am Monitor vorhandene DPI-Wert wider. Die optional ausgegebenen „tile dpi“ und „scaled dpi“ dienen zur Kontrolle der für die Bilder und Bedienelemente zugrunde gelegten DPI-Werte.

Ein dynamisches Wechseln der Skalierung ist für Motif-Anwendungen nicht vorgesehen. Ebenso kann eine Umsetzung der Skalierung im Desktop nicht erkannt werden.

Unterstützung von XFT / Font-Handhabung / Motif-Font

Die Basis für eine GUI-Anwendung für unterschiedliche Desktop-Skalierungen stellen die verwendeten Schriften dar. Im Gegensatz zu den bisher unter Motif/X verwendbaren Schriftarten, die meist über einen XLFD (X Logical Font Descriptor) angeben wurden, werden XFT Fonts entsprechend der gesetzten Skalierung des Desktop mitskaliert.

Die bisher unter dem IDM FÜR MOTIF verwendbaren Schriftarten hatten sich nicht automatisch an unterschiedliche Desktop-Skalierungen angepasst.

*Die XFT-Unterstützung ist im IDM nur auf Linux-Plattformen verfügbar. Da die meisten Desktops auf Linux-Plattformen den Standard-Font für Motif-Widgets nur unzureichend setzen, benutzt der IDM **Liberation Sans** als Standard-Schrift und **Liberation Mono** als Festbreitenschriftart, wenn kein Font im Dialog gesetzt wurde. Dadurch skalieren sich Texte in IDM-Dialogen auch ohne gesetzte Schriftart.*
Durch explizites Ausschalten des Scaling (über die Startoption -IDMscale=0), kann man die alten Default-Schriftarten erhalten.

Zum setzen eines XFT Fonts muss wie bei bisherigen Fontdefinitionen im IDM nur der Fontname und ggf. weitere Modifier gesetzt werden. Ist der Font als XFT Font auf dem System vorhanden, dann wird dieser geladen. Die Liste der auf dem System verfügbaren XFT Fonts kann im IDM EDITOR eingesehen werden.

Unter CDE-konformen Plattformen wie AIX und HP-UX verwendet der IDM als Standard-Schrift „-dt-interface_system-medium-r-normal-*“ und „-dt-interface_user-medium-r-normal-*“ als Festbreitenschrift, sofern keine Schrift gesetzt wurde. Je nach Skalierungsfaktor werden dabei unterschiedliche Größenausprägungen des Fonts herangezogen. Zwischen *1%...75%* wird **xs** (extra small) verwendet, zwischen *75%-150%* wird **m** (medium) verwendet, zwischen *150%-250%* wird **l** (large) verwendet und ab *250%* wird **xxl** (extra extra large) verwendet. Dadurch kann für solche Anwendungen eine Skalierbarkeit erreicht werden auch wenn keine XFT-Fonts verfügbar sind.

1.1.11 Unterstützung hoher Auflösungen beim IDM für IDM for Windows 11

Es gibt jetzt einen ***IDM für Windows 11*** und einen für ***Windows 10***. Der IDM FÜR WINDOWS 11 unterstützt DPI-Awareness und High DPI.

Der IDM FÜR WINDOWS 11 unterstützt hohe Auflösungen des Modells „PerMonitor V2“. Da die Unterstützung hoher Auflösungen eine Eigenschaft der Anwendung ist, muss diese im Anwendungsmanifest entsprechend gekennzeichnet werden. Entsprechende Setzung können in den IDM-Beispielen entnommen werden. Der IDM erkennt dies dann selbständig und rechnet die Koordinaten entsprechend dem eingestellten Vergrößerungsfaktor um. Mit anderen Worten, die Pixelkoordinaten des IDM sind virtuelle Pixel, die entsprechend dem eingestellten Vergrößerungsfaktor umgerechnet werden.

Die Manifest-Dateien der IDM-Beispiele kennzeichnen die gebauten Anwendungen als „High DPI-Aware“. Um eine Anwendung zu bauen, die nicht „DPI-Aware“ ist, muss der gesamte Block `<asmv3:application>` aus der Manifest-Datei entfernt werden.

Hinweise IDM für Windows 11

Alle Funktionen, die Microsoft Windows Nachrichten verarbeiten: Wenn die Anwendung hohe Auflösungen unterstützt, dann sind alle Koordinaten, die über Microsoft Windows Nachrichten erhalten werden, die realen hochauflösenden Koordinaten. Der IDM benutzt dagegen um den Vergrößerungsfaktor bereinigte Koordinaten. Es muss also berücksichtigt werden, dass im Falle der Unterstützung hoher Auflösungen, die IDM-Koordinaten im Allgemeinen nicht mehr identisch zu den Microsoft Windows Koordinaten sind. Betroffen hiervon sind zum Beispiel Inputhandler-, Canvas-, Monitor- und Subclassing-Funktionen, sowie auch die USW-Implementierungen.

Hinweise IDM für Windows 10 und 11:

Die Unterstützung hoher Auflösungen erfordert eine andere Behandlung von Windowsnachrichten. Dadurch ist es leider nicht mehr möglich Fenster während des Verschiebens oder Vergrößerns an Rahmenbedingungen wie zum Beispiel Rasterkoordinaten anzupassen. Erst nach dem Loslassen der Maustaste wird das Fenster auf die nächste Rasterkoordinate angepasst.

Intern definierte Cursor werden jetzt auf die vom System geforderte Größe vergrößert bzw. verkleinert, um diese automatisch auf den eingestellten Vergrößerungsfaktor anzupassen. Hierdurch werden die Cursor richtig dargestellt, wenn die Anwendung hohe Auflösungen unterstützt. Bisher wurden die Cursor entweder mit durchsichtigen Bereichen aufgefüllt oder einfach abgeschnitten, wenn die Größe nicht passte.

Bilder (***tile*** Ressource) werden automatisch entsprechend des eingestellten Vergrößerungsfaktor vergrößert. Es wird hierbei davon ausgegangen, dass die Bilder für einen DPI-Wert von *96* entworfen wurden. Sind die Bilder der Anwendung für eine höhere Auflösung gestaltet, dann kann dies am setup Objekt mit dem Attribute *.tiledpi* eingestellt werden. Eine automatische Anpassung gilt nur für die ***tile***-Ressource. Bilder die direkt über den Dateinamen am *.picture* Attribute des image-Objektes angegeben werden, werden nicht automatisch angepasst.

Als Standardschriftart wird nun die in den Systemparameters definierte Dialog- bzw. Message-Schriftart verwendet, da diese sich an die Auflösung anpasst. Hierdurch kann es zu Änderungen der Darstellung von Objekten kommen, die keine Schriftart gesetzt haben. Die erwähnte Schriftart kann über Windows Systemeinstellungen konfiguriert werden.

Die alten Systemschriftarten sind nicht DPI-Aware. Nur die neuen ***UI*_FONT*** Schriftarten passen sich der eingestellten Auflösung an.
Die alten Windows-Schriftarten „ANSI_FIXED_FONT“, „ANSI_VAR_FONT“, „DEVICE_DEFAULT_FONT“ und „SYSTEM_FONT“ lassen sich nicht an verschiedene Auflösungen anpassen, daher wird von der Verwendung abgeraten. Von der Verwendung von Schriftarten ohne Größenangabe wird abgeraten, da in einem solchen Fall vom Windows Font Mapper eine schriftartspezifische Standardgröße gewählt wird, die sich nicht an unterschiedliche Auflösungen anpasst.

1.1.12 Unterstützung hoher Auflösungen unter Qt

Für die Unterstützung hoher Auflösungen aktiviert der IDM das Qt-seitige HighDpi-Scaling sowie das mögliche Rendern von Bildern über ihre eigentlich angeforderte Größe hinaus. D.h. die Skalierung wird auf Basis der Pixeldichte des Monitors vorgenommen.

Unter Linux Desktop-Umgebungen ist die Unterstützung für HighDPI leider sehr unterschiedlich ausgeprägt und fortgeschritten. Da die Werte für logische und tatsächliche Pixeldichte sowie Skalierungsfaktoren vom Qt-Framework bezogen werden, kann es sein, dass weniger populäre Desktop-Umgebungen hier unzureichende Werte liefern.

Hinweise zu HighDPI Einstellungen

Qt bietet noch einige Umgebungsvariablen an, mit denen die Darstellung bei hohen Auflösungen gesteuert werden kann. Bei Einsatz solcher Umgebungsvariablen sollte jedoch bewusst sein, dass somit das Verhalten des IDM überschrieben wird und es dadurch zu ungewollten Darstellungseffekten kommen kann.

Anmerkungen zur Skalierung von Tiles

Objekte werden standardmäßig mit den vom Desktop-Style vorgegebenen Werten durch das WSI/Toolkit dargestellt. Es kann daher in bestimmten Situationen vorkommen, dass die gewünschten Einstellungen bezüglich Größe und Skalierung von Tiles nicht umgesetzt werden. Dies betrifft die Verwendung von Tiles bei Objekten mit Tabs wie dem Notebook bzw. den Notepages sowie dem Menu mit Menuitems. Der IDM setzt zwar die gewünschten Darstellungsoptionen, jedoch unterliegt die finale Anzeige der

Darstellungsrichtlinie der jeweiligen DrawingEngine des gesetzten UI-Styles. Das bedeutet, dass die Anzeige von den gesetzten Optionen Style-abhängig abweichen kann oder Optionen vollständig ignoriert werden.

1.1.13 Erweiterungen/Änderungen im IDMED

Eine beliebige Auswahl der Schriftart für die Bedienoberflächen oder Regelcode ist nicht mehr möglich. Dafür kann die Schriftgröße zwischen Klein (Small) – Normal – Groß (Large) – Extra Groß (Extra Large) ausgewählt werden. Diese hat dann natürlich Einfluss auf die Fenstergröße.

Im Schriften- bzw. Font-Einstellungsbereich werden unter Motif auch XFT-Fonts aufgelistet. Außerdem kann die Liste der Auswählbaren Schriften zum Zwecke der Übersichtlichkeit auf UI-Fonts, X-Fonts und XFT-Fonts reduziert werden.

Der IDMED, TracefileAnalyzer sowie der Debugger setzen nun UI-Ressourcen ein.

1.1.14 Unterstützung hoher Auflösungen bei der USW-Programmierung

Die Funktion **DM_GetToolkitDataEx** wurde zur Unterstützung von HighDPI verbessert. Die **ToolkitDaten-Struktur DM_ToolkitDataArgs** wurde zu diesem Zweck erweitert und liefert nun detaillierte Informationen zu DPI, Skalierungsfaktor und Bild. Diese können über die Attribute *AT_DPI* und *AT_XWidget* erfragt werden.

Hinweis Motif

Der USW-Programmierer muss die Koordinaten zwischen dem IDM (dies betrifft Attribute sowie Events) und den Motif-Werten/Strukturen/Funktionen (*X…/ Xt…/ Xm…*) umrechnen. Die für eine Umrechnung benötigten Daten können über Funktion **DM_GetToolkitDataEx** mittels AT_DPI und die **ToolkitDaten-Struktur DM_ToolkitDataArgs**-Struktur (*tile.scale.factor*) erfragt werden. Je nach Bildart muss u.U. auch eine Skalierung vorgenommen werden.

Hinweis für Windows 11

Der IDM FÜR WINDOWS 11 unterstützt hohe Auflösungen. Wenn eine Anwendung für hohe Auflösungen freigegeben wird, dann müssen auch die von ihr verwendeten USW-Objekte hohe Auflösungen unterstützen. Das heißt dann, dass die USW-Objekte von Windows reale Koordinaten erhalten (um den Vergrößerungsfaktor vergrößert), während der IDM mit virtuellen Koordinaten (nicht vergrößert) arbeitet. Praktisch wird sich nicht viel ändern, da der IDM die Koordinatenberechnungen für das USW Objekt ausführt. Aber es ist zu berücksichtigen, dass der **control** Aufruf für „UC_C_PrefSize“ reale Koordinaten erwartet, da diese im Normalfall aus Toolkitwerten berechnet werden. Sind hier fixe Zahlen wie im *ucarrow* Beispiel eingesetzt, dann müssen diese um den aktuellen DPI-Wert korrigiert werden (siehe ucarrow Beispiel).

Wenn die Anwendung als DPI-Aware gestartet wurde, werden nun von der Task „UC_I_clientarea“ der Funktion **UC_Inquire** auch die hochauflösenden Werte erwartet.

Ergänzungen der C-Schnittstelle für DM_GetToolkitData() und DM_GetToolkitDataEx()

Bei den C-Schnittstellen Funktionen "DM_GetToolkitData()", "DM_GetToolkitDataEx()" und Toolkitdaten-Struktur "DM_ToolkitDataArgs" gibt es zur Unterstützung von hohen Auflösungen die folgenden Erweiterungen:

Attribute AT_DPI

Gibt den DPI-Wert des Systems oder des angegebenen Objektes zurück. Die Funktion "DM_GetToolkitDataEx()" muss mit einem Zeiger auf eine Toolkitdaten-Struktur "DM_ToolkitDataArgs" aufgerufen werden. In dieser Struktur werden weitere DPI-Informationen zurückgeliefert. Sie ist hierzu um das Datenfeld "dpi" und die Unterdatenstruktur "scale" erweitert.

Attribute "AT_Tile" bzw. "AT_XTile"

Dieses Attribut gab es schon für die "tile" Ressource. Jetzt kann die Funktion "DM_GetToolkitDataEx()" mit einem Zeiger auf eine Toolkitdaten-Struktur "DM_ToolkitDataArgs" aufgerufen werden, die um die Unterdatenstruktur "tile" erweitert wurde. Diese enthält unter anderem den DPI-Wert, für den die "tile" Ressource entworfen wurde.

Attribute "AT_IsNull"

Liefert einen Wert ungleich "0" zurück, wenn die angegebene "color" oder "font" Ressource auf "UI_NULL_FONT" bzw. "UI_NULL_COLOR" definiert war.

Siehe auch Kapitel„ToolkitDaten-Struktur DM_ToolkitDataArgs“ im „C-Schnittstelle - Grundlagen“-Handbuch.

Siehe auch Kapitel „DM_GetToolkitDataEx“ im „C-Schnittstelle - Funktionen“-Handbuch.

Siehe auch Kapitel „DM_GetToolkitDataEx“ im „C-Schnittstelle - Funktionen“-Handbuch.

1.2 Multi-Monitor Support unter Windows

Unter Windows ist es jetzt möglich, die Anzahl der Monitore (*.screencount*) und die Koordinaten der Monitore (*.screen_x[integer]*, *.screen_y[integer]*, *.screen_width[integer]* und *.screen_height[integer]*) über das ***setup***-Objekt abzufragen. Die erfragten Werte sind dabei dynamisch, da man jederzeit die Skalierung eines Monitors ändern oder auch Monitore hinzufügen oder wegnehmen kann.

Werden mehrere Monitore verwendet, werden diese in einem virtuellen Desktop positioniert. Die Koordinaten dieses virtuellen Desktop sind mit *.vscreen_x*, *.vscreen_y*, *.vscreen_width*, *.vscreen_height* abfragbar. Es werden immer die Koordinaten des Arbeitsbereichs, also ohne die Taskleiste, geliefert. Zur Berechnung des virtuellen Desktops werden auch die Arbeitsbereiche herangezogen.

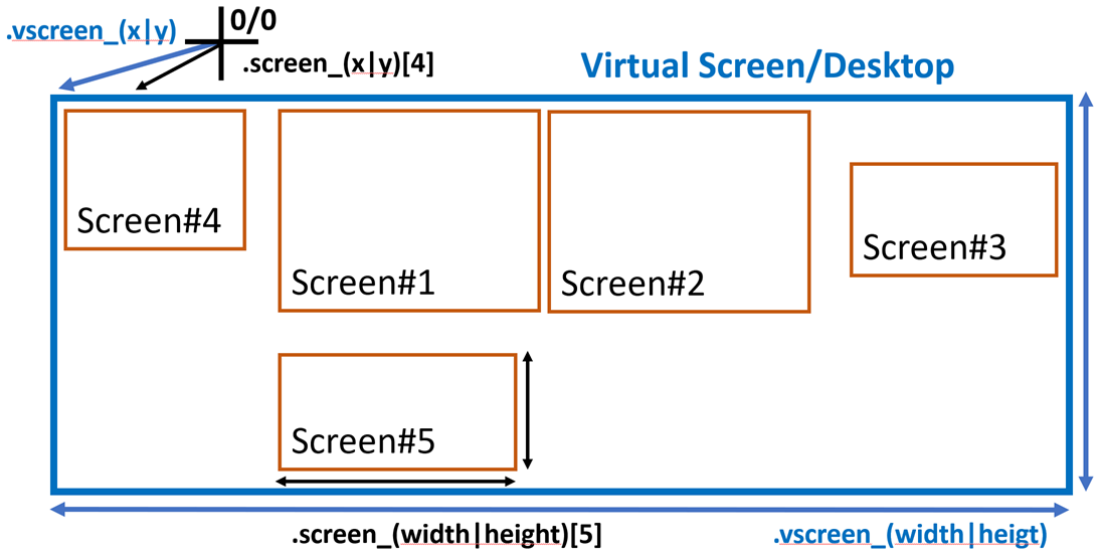

Abbildung 23-5: *Relation der .screen_*[] und .vscreen_* Attribute*

Achtung:

Bei unterschiedlichen Vergrößerungseinstellungen der Monitore ist zu beachten:

IDM für Windows 11

Die Werte erscheinen nicht konsistent. Ausgehend von der für Fenster empfohlenen Verwendung von *.xauto = 1* und *.yauto = 1* wird die Position mit dem Systemvergrößerungsfaktor umgerechnet, während die Größe mit dem Monitorvergrößerungsfaktor umgerechnet wird. Hierdurch lässt sich ein Fenster auf eine Position und Größe setzen, die den gesamten Monitor ausfüllt.
Die Position und die Größe des virtuellen Desktops werden mit dem Systemvergrößerungsfaktor umgerechnet. Zu welchem Monitor ein Fenster, das sich über mehrere Monitore erstreckt, zugeordnet wird, kann von dem Monitor abhängen, auf dem sich das Fenster gerade befindet. Für ein definiertes Verhalten sollten deshalb Koordinaten nur im unsichtbaren Zustand geändert werden.

IDM für Windows 10

Microsoft Windows skaliert Anwendungen, die DPI-Unaware sind, intern. Hierbei führen die unterschiedlichen Vergrößerungsfaktoren zu Falschdarstellungen. Um dies zu vermeiden sollten entweder alle Fenster nur auf dem primären Monitor geöffnet werden, oder alle Monitore denselben Vergrößerungsfaktor besitzen.

Anmerkung zu Windows 10 und 11 mit mehreren Monitoren:

Unter Windows kann es nach dem Hinzufügen/Entfernen eines Monitors oder dem Ändern der Anzeigeeinstellungen zu folgenden Problemen kommen:

- Es erscheint kein Verschieberahmen beim Verschieben eines **toolbar** Objektes. Es kann auch vorkommen, dass ein Rechteck in der Größe des **toolbar** Verschieberahmens mit einem falschen Hintergrund auf einem anderen Monitor dargestellt wird.
- Ausgedockte **toolbar** Objekte passen sich nicht an Änderungen der DPI Wertes an. Sowohl beim Verschieben auf einen anderen Monitor oder auch beim Ändern der Vergrößerung.

Das Problem liegt an Windows, das zum einen die Koordinaten beim Zeichnen auf den Desktop falsch umsetzt und zum anderen wichtige Nachrichten (*WM_DPICHANGED*) nicht an sogenannte Tool-Fenster versendet und den Window DPI Wert nicht ändert.

Nachdem der Bildschirmschoner aktiv war oder man sich neu angemeldet hat (und nach einem Neustart), tritt das Problem nicht mehr auf.

1.3 Kompatibilität

1.3.1 Motif

Der IDM für Motif stellt eine Anwendung vergrößert dar, wenn eine Systemskalierung aktiv ist. Mittels der Startoption `-IDMscale=0` kann dieses Verhalten ausgeschaltet werden kann.

1.3.2 Windows

Wenn eine Bildschirmvergrößerung aktiv ist, kann das Raster des IDM für Windows 11 nicht proportional zur Vergrößerung sein, da die Schriftgröße in logischen Points angegeben wird. Je mehr Pixel zur Verfügung stehen umso besser entspricht der logische Wert dem realen.

1.3.3 Qt

Beim IDM für Qt ist nun HighDPI-Skalierung eingeschaltet. D.h. bei aktiver Systemskalierung wird die Anwendung nun unweigerlich und automatisch angepasst, was nun nicht mehr nur Fonts sondern auch Geometrie betrifft.

1.4 Kern

IDM-13192: Ein *finish*-Ereignis von Applikations-Objekten hatte Einfluss auf die Behandlung von finish-Ereignissen von Dialog/Module-Objekten. Dies führte zur doppelten Ausführung von `finish`-Regeln am Dialog/Modul und konnte das Verlassen, bzw. nicht-Verlassen der Ereignisschleife beeinflussen. Dieses Fehlverhalten ist nun behoben.

IDM-13191: Es kommt nun nicht mehr zu ungültigen Speicherzugriffen im Regel-Resolver bei der Auflösung von Pfaden in Werte-Ausdrücken. Auch ein u.U. weggelassener Attributszugriff sollte nicht mehr passieren.

IDM-13072: Das „Sharing“ von importieren Modulen, welches mittels Umgebungsvariable **DM_SHARED_MODULES** aktiviert werden kann, um Imports mit gleichem Bezeichner vom Vorhandensein des gleichen Imports im Supermodule zu befreien, ist eingeschränkt um die Konsistenz bzgl. `Start/Stop/:init/:finish`, der Dialogzugehörigkeit und dem Sichtbarkeitszustand von Objekten zu gewährleisten. Es können nur Module im gleichen Dialog geshared werden!

Achtung:

Das Sharing von Modulen über **DM_SHARED_MODULES** ist kein Standard-Feature des IDM. Sinnvoller ist die Verwendung von „use“ um Module zu importieren und so das mehrfache Laden von Modulen in komplexen Dialogen sicher zu unterbinden.

Neu: Attribut *.hinttext* an den Objekten ***edittext*** und ***poptext*** eingefügt, mit dessen Hilfe es möglich ist einen Platzhalter- oder Hinweistext anzeigen zu lassen, wenn das Objekt noch keinen Inhalt gesetzt hat. Am ***poptext***-Objekt muss muss dabei der Style *edittext* gesetzt sein..

	Bezeichner		Datentyp
Regelsprache	**.hinttext**		string
C	AT_hinttext		DT_string
COBOL	AT-hinttext		DT-string
Klassifizierung	Objektspezifisches Attribut		
Objekte	***edittext***, ***poptext***		
Zugriff	get, set	Standardwert	–
changed-Ereignis	ja	Vererbung	ja

Achtung:

Bei einem Leerzeichen als Inhalt wird der Platzhalter - oder Hinweis-Text nicht angezeigt, dies betrifft ebenso einige Formate.

Besonderheiten:

Dieses Attribut wird nur von Windows und Qt unterstützt.

Anmerkung für Windows:

Unter Microsoft Windows wird das Attribut im Moment an folgenden Objekten unterstützt:

- ***edittext***-Objekt, wenn die Attribute *.multiline* und *.options[opt_rtf]* den Wert *false* besitzen
- ***poptext***-Objekt, wenn das Attribut *.style* den Wert *edittext* oder *listbox* besitzt

Der Hinttext wird angezeigt, wenn das Objekt den Fokus nicht besitzt und der Inhalt leer ist.

Anmerkung für Qt:

Wenn beim ***edittext***-Objekt das Attribut *.multiline* den Wert *true* besitzt und Layoutattribute (z.B. *.alignment*, *.xmargin*, *.ymargin* etc.) gesetzt sind, so werden diese bei der Anzeige des Hinweis- oder Platzhaltertexts nicht beachtet. Der Hinweis- oder Platzhaltertext ist immer links oben angebunden.

1.5 Windows

Neu: ***Achtung:*** Es gibt jetzt einen IDM für Windows 11 und einen IDM für Windows 10. Der IDM für Windows 11 unterstützt ***DPI Awareness*** und ***High DPI***.

Neu: Mit dem IDM für Windows 11 wird nun auch Microsoft Visual C++ 2022 unterstützt.

IDM-13194: Nach einer Drag und Drop Aktion erschien die Fehlermeldung "WSI error processing clipboard.value". Voraussetzung war, dass eine :setclip Methode existierte, die das Clipboard änderte.

IDM-13215: Unter Windows wurden die Geometrieattribute nicht vererbt, wenn diese interaktiv am Modell geändert wurden.

IDM-12999: Es trat eine Exception in einer **UIAutomation** Testanwendung auf, wenn an einem ***poptext*** Objekt die Option **opt_hscroll** gesetzt war. Dies wird jetzt vermieden, indem das **UIAutomation** Objekt für die Poptextliste erst wieder freigegeben wird, wenn das Poptext **UIAutomation** Objekt nicht mehr gebraucht wird. Seither wurde das **UIAutomation** Objekt für die Poptextliste freigegeben, wenn diese nicht mehr gebraucht wurde.

IDM-12967: Es konnte vorkommen, dass die erste Tracemeldung der **UIAutomation** leer war. Das Problem trat dann auf, wenn das erste Objekt erst nach dem Einschalten des **UIAutomation** Tracings sichtbar geschalten wurde.

IDM-13188: Wenn ein Fenster aus dem minimierten Zustand heraus maximiert wurde, dann wurden unten angebundene Objekte falsch positioniert, sie erschienen zu weit unten.

IDM-11065: Wenn die Anwendung unter Terminal Services lief, ertönte kein Warnton. Jetzt wird unter Terminal Services die Windows-Funktion **Beep** verwendet. Achtung, die Änderung betrifft nur die Töne, die der IDM erzeugt.

IDM-10294: **Mnemonics** werden nun nicht mehr auch an Objekten auf inaktiven ***notepage*** Seiten ausgelöst.

Neu: Die Attributsnamen *AT_Font* und *AT_XFONT* wurden als Alternative zu *AT_wsidata* für die **ToolkitData**-Funktionen an der ***font*** Ressouce eingeführt. Die Verwendung ist identisch zu *AT_wsidata*.

Neu: Die Attributsnamen *AT_Color*, *AT_Tile* und *AT_Widget* als Alternative zu *AT_XColor*, *AT_XTile* und *AT_Widget* für die **ToolkitData**-Funktionen eingeführt. Die Verwendung ist identisch zu dem entsprechenden Attribut mit dem „X“.

IDM-12697: Bei einer Installation auf ein Netzwerklaufwerk kam es zu Problemen. Achtung, die Installaltion auf ein Netzwerklaufwerk ist nicht vorgesehen. In der Regel wird der Versuch zu einem Abbruch führen, nachdem bereits alle Dateien entpackt sind. Falls kein Abbruch geschieht, findet das Entsperren der Dateien statt. Dies kann nur in der letzten Phase der Windows Installation geschehen. Diese Phase wird vom Windows Installer unter dem "SYSTEM" Konto ausgeführt und hat in der Regel keinen Zugriff auf Netzwerk-Resourcen.

Erweitert: Für den IDM Editor werden jetzt mehr Informationen zu den vorhandenen Schriftarten angezeigt, mögliche Optionen werden verdeutlicht. Außerdem wird bei der Selektion jetzt der *type* Parameter berücksichtigt.

Neu: Die Option **opt_show_ticks** wird nun vom Scrollbar-Objekt unterstützt.

Behoben: Das ***setup*** Objekt wurde im WSI nicht angelegt/initialisiert. Hierdurch wurde ein initial gesetzter *.overridecursor* nicht übernommen.

Neu: Die ***font*** Ressource um die neuen Optionen für *black*, *demibold* und *light* erweitert. Außerdem wird jetzt auch die Fontfamilie *roman* unterstützt.

IDM-13057: Der Verschieberahmen eines ***toolbar*** Objektes erschien nicht über dem ***toolbar*** Objekt, wenn ein Vergrößerungsfaktor eingestellt war. Dieses Problem ist jetzt behoben.

Anmerkung Windows 10 und 11 mit mehreren Monitoren:

Unter Windows kann es nach dem Hinzufügen/Entfernen eines Monitors oder dem Ändern der Anzeigeeinstellungen zu folgenden Problemen kommen:

- Es erscheint kein Verschieberahmen beim Verschieben eines **toolbar** Objektes. Es kann auch vorkommen, dass ein Rechteck in der Größe des **toolbar** Verschieberahmens mit einem falschen Hintergrund auf einem anderen Monitor dargestellt wird.
- Ausgedockte **toolbar** Objekte passen sich nicht an Änderungen der DPI Wertes an. Sowohl beim Verschieben auf einen anderen Monitor oder auch beim Ändern der Vergrößerung.

Das Problem liegt an Windows, das zum einen die Koordinaten beim Zeichnen auf den Desktop falsch umsetzt und zum anderen wichtige Nachrichten (*WM_DPICHANGED*) nicht an sogenannte Tool-Fenster versendet und den Window DPI Wert nicht ändert.

Nachdem der Bildschirmschoner aktiv war oder man sich neu angemeldet hat (und nach einem Neustart), tritt das Problem nicht mehr auf.

Neu, IDM-11904, IDM-11753: Der IDM für Windows bietet nun Multi-Monitor Support. Siehe Kapitel 1.2 „Multi-Monitor Support unter Windows“

Damit ein Fenster in einer Multi Monitor Konfiguration richtig maximiert wird, muss die Berechnung der maximalen Größe Microsoft Windows überlassen werden. Hierdurch können Fenster von der Taskbar überdeckt werden.

Behoben: **Achtung:** Änderung der Berechnungsmethode für die Rasterbreite.

Mit der IDM-Version A.06.03.a wurde die Berechnung der Rasterbreite grundlegend umgestellt. Wenn kein Referenzstring angegeben ist, wird die Rasterbreite jetzt aus einem internen Referenzstring („M“) berechnet, um ein übermäßiges Breitenwachstum durch überbreite Buchstaben innerhalb einer Schriftart zu vermeiden.

Aus Kompatibilitätsgründen kann jedoch durch die Option *opt_fontraster_compat*, die Startoption **-IDMfontraster_compat** oder die Umgebungsvariable `IDM_FONTRASTER_COMPAT` vorübergehend die veraltete Berechnung der Rasterbreite wieder reaktivieren werden (verbunden mit dem Nachteil, dass es wieder zu übermäßiges Breitenwachstum kommen kann).
Bei Verwendung der Option *opt_fontraster_compat* basiert die Größenberechnung zum Teil auf dem Systemfont, der nicht High DPI fähig ist, daher sollten High DPI fähige Anwendungen, die mit IDM für Windows 11 erstellt wurden, dies Option nicht verwenden.

Hinweis: Die **DM_GetToolkitData**/**DM_GetToolkitDataEx** Attribute *AT_Tile*, *AT_XTile* und *AT_wsidata* wurden um **ToolkitDaten-Struktur DM_ToolkitDataArgs** erweitert und sind wie folgt belegt:

- `tile.gfxtype`
  Wird auf den *GFX Type* gesetzt, der am besten passt.
- `tile.datatype`
  Wird auf den Wert gesetzt, der bisher über *AT_DataType* abgefragt werden musste.
- `tile.data`
  Wird auf die Daten gesetzt, der genaue Typ ist abhängig von `tile.gfxtype` bzw. `tile.datatype`:
  HICON: wenn *DM_GFX_ICO* bzw. *DMF_TlkDataIsIcon*
  HBITMAP: wenn *DM_GFX_BMP* bzw. *0*
  HMETAFILE: wenn *DM_GFX_WMF* bzw. *DMF_TlkDataIsWMF*
  HENHMETAFILE: wenn *DM_GFX_EMF* bzw. *DMF_TlkDataIsEMF*
- `tile.mask`
  Wird auf die monochrome Maske gesetzt, falls eine vorhanden ist. Dies kann nur bei *DM_GFX_BMP* vorkommen.
- `tile.palette`
  Wird auf die Farbpalette gesetzt, falls eine vorhanden ist.
- `tile.dpi`
  Der DPI Wert für den die vom IDM geladenen Bilder entworfen wurden.
- `rectangle`
  Wird auf die Größe gesetzt, die zum angeforderten DPI Wert passt. Siehe Bemerkung.
- `dpi`, `scale.dpi` und `scale.factor`
  Werden passend zum angeforderten DPI Wert gesetzt. Siehe Bemerkung und Hinweise zu *AT_DPI*.

Bemerkung zum angeforderten DPI Wert:

Der angeforderte DPI Wert wird in der angegebenen Reihenfolge aus folgenden Angaben ermittelt:

- Ist in der `argmask` das *DM_TKAM_handle* Bit gesetzt, dann wird der DPI Wert des Microsoft Windows Control ermittelt dessen Windows Handle (HWND) im `handle` Element steht.
- Ist in der `argmask` das *DM_TKAM_scaledpi* Bit gesetzt, dann wird der System DPI Wert ermittelt (entspricht dem Vergrößerungsfaktor des primären Monitors).
- Sonst wird die originale Bildgröße, also der DPI Wert für den die Bilder entworfen wurden, ermittelt.

Achtung: Dies bedeutet nicht, dass die Bilddaten die entsprechende Größe haben, diese müssen eventuell skaliert werden.

Der Rückgabewert bleibt wie gehabt, also entspricht `tile.data`.

Hinweis: Die Elemente der **ToolkitDaten-Struktur DM_ToolkitDataArgs** werden beim Abfragen des Attributs *AT_DPI* in **DM_GetToolkitData**/**DM_GetToolkitDataEx** folgendermaßen belegt:

- `dpi`
  Wird auf den Standard DPI Wert gesetzt, den Anwendungen die nicht DPI Aware sind verwenden. Dieser Wert wird von Microsoft Windows als Konstante *USER_DEFAULT_SCREEN_DPI* (Wert: *96*) definiert. Diesen Wert verwendet der IDM für seine Pixelkoordinaten.
- `scale.dpi`
  Wird auf den aktuellen DPI Wert des Objektes gesetzt, er entspricht dem Rückgabewert von *AT_GetDPI*. Dieser Wert ist beim IDM für Windows 11 dynamisch, wenn die Anwendung DPI Aware ist. Sonst, DPI Unaware oder IDM für Windows 10, ist der Wert fest auf *96* definiert.
- `scale.factor`
  Ganzzahliger Prozentwert, der aus `scale.dpi` und `dpi` berechnet wird. Der Rückgabewert (auf int gewandelt) entspricht `scale.dpi`.
  Um den DPI Wert für ein Microsoft Windows Control zu erhalten, kann in der **ToolkitDaten-Struktur DM_ToolkitDataArgs** Struktur das `handle` Element auf den Windows Handle (HWND) des Controls gesetzt werden. In der `argmask` muss dann das Bit *DM_TKAM_handle* gesetzt sein und als Objekt muss *0* oder das ***setup*** Objekt im **DM_GetToolkitDataEx** Aufruf angegeben werden.

Neu: Mit **DM_GetToolkitData** bzw. **DM_GetToolkitDataEx** kann unter Microsoft Windows der aktuelle DPI Wert mit *AT_GetDPI* abgefragt werden. Abhängig vom *objectID* Parameter werden folgende Werte geliefert:

- ID eines sichtbaren Objektes:
  Der DPI Wert, der für dieses Objekt gilt. Wert ist abhängig vom Monitor auf dem das Objekt erscheint.
- NULL-ID oder ID des ***setup*** Objektes:
  Der System DPI Wert oder, wenn im *data* Parameter der Microsoft Windows Window Handle (HWND) übergeben wird, der DPI Wert dieses Objektes.
- sonst:
  Entweder ein zufälliger alter Wert (wenn Objekt nicht sichtbar) oder 0 und eine Fehlermeldung.

Der Rückgabewert muss auf "int" gewandelt werden. Achtung, alle Werte sind dynamisch und können durch den Anwender über die Systemsteuerung geändert werden. Ist die Anwendung nicht DPI Aware (zum Beispiel IDM FÜR WINDOWS 10) dann wird immer der Standard DPI Wert von *96* geliefert.

Änderung: Linkeroption zu Subsystemversion entfernt, so dass die Anwendung nicht mehr auf XP-Komatibilität gesetzt wird.

Behoben: Das ***scrollbar*** Objekt hatte die *.real_width* und *.real_height* Attribute nicht aktualisiert, wenn keine Größe gesetzt war und das Attribut *.arrows* den Wert false besaß.

Änderung: Ergänzungen zur Einstellung der fontraster-Kompatibität für das Windows-WSI. Diese ist normalerweise auf false/inactive. Diese Kompatibitätseinstellung kann über folgende Wege gesetzt/aktiviert werden:

- Startoption **-IDMfontraster_compat**
- Umgebungsvariable `IDM_FONTRASTER_COMPAT` ist gesetzt
- Option *setup.options[opt_fontraster_compat]*

Für Hinweise bzgl. Rasterberechnung siehe entsprechende Kapitel im Windows WSI.

Neu:**DM_GetToolkitData** bzw. **DM_GetToolkitDataEx** unterstützt unter MICROSOFT WINDOWS am setup Objekt die Attribute *AT_Font* und *AT_XFont*. Es wird der Font-Handle der verwendeten Standardschriftart zurückgeliefert (der Rückgabewert muss auf *HFONT* gewandlet werden). Die Schriftart ist für den System-DPI-Wert skaliert. Im *data* Parameter kann ein anderer DPI-Wert als Integer angegeben werden (Wandlung: (DM_Pointer) (size_t) dpi).

Neu:**DM_GetToolkitData** bzw. **DM_GetToolkitDataEx** unterstützt unter MICROSOFT WINDOWS an den IDM Objekten die Attribute *AT_Font* und *AT_XFont*. Es wird der Font-Handle der verwendeten Schriftart zurückgeliefert (Rückgabewert muss auf *HFONT* gewandelt werden). Der Font ist für den aktuellen DPI-Wert des Objektes skaliert.

Neu:**DM_GetToolkitData** bzw. **DM_GetToolkitDataEx** unterstützt unter MICROSOFT WINDOWS an der font und color Resource das Attribute *AT_IsNull*. Hiermit kann abgefragt werden, ob die Resource auf *NULL* definiert wurde (*UI_NULL_FONT* bzw. *UI_NULL_COLOR*). Die Resource wurde auf NULL definiert, wenn der Rückgabewert nicht *0* ist.

Änderung: Die Funktion **DM_GetToolkitDataEx** liefert unter MICROSOFT WINDOWS bei den font- und color Resourcen einen **NULL** Wert zurück, wenn die Resource auf **UI_NULL_FONT** bzw. **UI_NULL_COLOR** definiert wurde. Dies betrifft folgende Attribute:

- *AT_wsidata*, *AT_Font*, *AT_XFont*:
  Der Rückgabewert ist bei *UI_NULL_FONT* dann *(HFONT) 0*.
- *AT_Color*, *AT_XColor*:
  Der Rückgabewert ist bei *UI_NULL_COLOR* dann *(COLORREF) -1L*.
- *AT_Tile*, *AT_XTile*:
  Der Rückgabewert ist bei *UI_NULL_COLOR* dann *(HBRUSH) 0*.

Neu: Das *.scalestyle* Attribut an der ***tile*** Ressource wird jetzt auch von den Objekten ***treeview***, ***notebook***/***notepage***, ***menuitem*** und ***window*** beachtet.

Neu: Die beiden Attribute *.real_x* und *.real_y* entsprechend *.real_width* und *.real_height* eingeführt.

Änderung: An der ***tile*** Ressource wurden die vordefinierten Namen *OLD_UPARROW*, *OLD_DNARROW*, *OLD_RGARROW*, *OLD_LFARROW*, *OLD_REDUCE, OLD_ZOOM* und *OLD_RESTORE* entfernt, es müssen die OEM Ressource-Namen ohne *OLD_* verwendet werden.

Behoben: Wurde am ***toolbar*** Objekt das *.reffont*, bzw. *.xraster* / *.yraster* Attribut geändert, dann wurde die Größe des toolbar Objektes nicht angepasst, wenn *.sizeraster = true* war.

Behoben: Wenn am ***poptext*** Objekt das *.format* Attribute dynamisch geändert wurde, dann wurde der Listeninhalt nicht angepasst.

Behoben: Wenn am ***image*** Objekt das Attribut *.imagefgc* gesetzt war, dann wurde diese Farbe fälschlicherweise auch als Textfarbe verwendet.

Änderung: Es gab bei manchen Objekten einen visuellen Unterschied zwischen einer gesetzten Schriftart, die es nicht gibt, und einer auf *null* gesetzten Schriftart. Dieser Unterschied tritt jetzt nicht mehr auf.

Neu: DPI-Awareness Informationen zum Tracefile-Header hinzugefügt. Sowie weitere Warnungen und Fehlermeldungen bei der DPI Initialisierung eingefügt, die auf nicht unterstützte Einstellungen hinweisen.

1.6 Motif

Behoben: Die Layoutbox hat beim Verkleinern die Kinder nicht sofort angepasst.

Behoben: Die Umrechnung der Positionen von Fenster und Toolbars (*dock_window*) erfolgt möglichst unter Berücksichtung von realen Fensterdekorationen um beim zweifachen *.visible*-Tooglen das Fenster/Toolbar wieder an die gleichen Stelle zu positionieren.

Behoben: Insensitiver Text beim ***image*** Objekt wird ab einschließlich MOTIF-2.3 ausgegraut und mit Schattenwurf analog zu anderen MOTIF-Widgets gezeichnet.

Neu: Durch neuere Desktop- und Window-Manager vorgegebene WorkArea-Bereiche (z.B. durch Menüs oder Dock-Bereiche) werden vom IDM berücksichtigt und verkleinern somit die nutzbare Screen-Größe.

1.7 Qt

IDM-13186: Beim Umschalten von Hintergrund-Tiles in der ***groupbox*** wurde nicht beachtet, ob die ***groupbox*** eine virtuelle Größe hat.

Neu: Die Option **opt_show_ticks** wird nun vom Scrollbar-Objekt unterstützt.

Neu: Weitere bisher nicht beachtete System-Farben zur color-Ressource hinzugefügt.

Neu: **DM_GetToolkitData** unterstützt nun *AT_XTile* und *AT_XWidget*. *AT_XTile* liefert dabei immer den Typ *DM_GFX_QPIXMAP* mit QPixmap zurück.

Behoben: Wenn ein Tile nachträglich dynamisch an der ***splitbox*** gesetzt wurde, konnte es vorkommen, dass dieses nicht gezeichnet wurde.

Behoben: Wenn ein Tile nachträglich dynamisch an der ***layoutbox*** gesetzt wurde, konnte es vorkommen, dass dieses nicht gezeichnet wurde.

Behoben: Das ***window*** Objekt beachtete eine dynamische Änderung von *.reffont* nicht und hat in Folge dessen seine Größe nicht angepasst.

Behoben: Die Methode **:openpopup()** hatte das Kontextmenü unter Qt nur am Mauszeiger und nicht direkt am Objekt angezeigt.

Behoben: Beim Ändern der Richtung der ***scrollbar*** im Sichtbaren wurde die Größe der Scrollbar nicht angepasst.

1.8 DDM (Netz)

Neu: Das Schema ***ssh://*** unterstützt nun auch das OpenSSH Kommando. Dieses Schema findet Anwendung beim *.exec* Attribut um zu bestimmen wie die Serverseite gestartet wird. Es wird zuerst gesucht, ob die **libssh** DLL verfügbar ist. Ist diese nicht verfügbar wird überprüft, ob ein `ssh` Kommando aufrufbar ist. Zusätzlich gibt es das Schema ***sshlib://*** um nur **libssh** zu verwenden und ***sshcmd://*** um nur OpenSSH zu verwenden. Der Nachteil des Kommandos ist, dass kein Passwort verwendet werden kann. OpenSSH muss so konfiguriert sein, dass eine Verbindung ohne Passwort möglich ist (siehe Manpages von `ssh`).

Neu: Es wird jetzt auch OpenSSL Version 3 unterstützt.

Behoben: Alte OpenSSL DLLs konnten unter Umständen nicht geladen werden, da die Funktion **SSL_CTX_set_options** nicht verfügbar war. Das Problem konnte auftreten, wenn Bau- und Einsatzsystem unterschiedliche OpenSSL Versionen besaßen. Jetzt wird ggf. optional die alte Funktion **SSL_CTX_ctrl** verwendet.

Neu: Attribute *.publickeyfile* und *.privatekeyfile* hinzugefügt. Für *.options[enum]* die Optionen **opt_verify_peer**, **opt_cert_required** und **opt_no_ssl_v2**, sowie die entsprechenden Startoptionen hinzugefügt.

Behoben: Die Netzwerk-Seite verhielt sich zwischen Unix und Windows unterschiedlich wenn ein geöffneter Port von einem zweiten Prozess verwendet werden sollte. Jetzt wird unter Windows die Socketoption **SO_REUSEADDR** nicht mehr gesetzt, so dass es zu einem Fehler kommt wenn der Port bereits verwendet wird.

Behoben: Eine Verifizierung der SSL Zertifikate konnte scheitern, wenn Standardsuchpfade nicht geladen wurden.

Behoben: Die Anwendungsseite reagierte nicht mehr, wenn es zu SSL Fehlern kam. Jetzt wird bei Fehlern, die während dem Anlegen des SSL Kontexts geschehen, der Prozess nicht mehr beendet, sondern die Verbindung wird weiter aufgebaut und somit die entstehenden Fehler auf beiden Seiten mitprotokolliert.

1.9 Debugger

IDM-13002: Wenn eine zu große Tracedatei im IDM Tracefile Analyzer abschnittweise geladen wurde, konnte es dazu kommen, dass die Anwendung nicht mehr reagiert hat. Dies wird nun verhindert.

Neu: Das abschnittweise Laden von zu großen Tracedateien im IDM Tracefile Analyzer ist nun möglich.

Behoben: Der IDM Tracefile analyzer unter Microsoft Windows hatte zum Teil die Darstellungsfläche nicht voll ausgenutzt.

1.10 Editor

Änderung: Die Preview eines Tiles ermittelt zuerst den realen Pfad vom editierten Modul, um so auch die Anzeige von Bildnamen mit „~“ besser zu unterstützen.

Änderung: Editiermöglichkeiten des *.scale*-Attributes bei ***tile***-Ressourcen umgestellt auf Enum/Auswahl für *.scalestyle*.

Neu: Propscale-Eigenschaft bei font-Ressourcen nun editierbar.

Neu: Der IDM TraceView analyzer unter Microsoft Windows hatte zum Teil die Darstellungsfläche nicht voll ausgenutzt.

1.11 USW

IDM-13057 (Keine Änderung): Hinweis zur High DPI Unterstützung des IDM für Windows 11: Wenn die Anwendung als DPI Aware gestartet wurde, erwartet der IDM für Windows 11 von der Task **UC_I_clientarea** der Funktion **UC_Inquire** die hochauflösenden, realen Werte.

Siehe auch Kapitel „HighDPI Unterstützung“ im Handbuch „Programmiertechniken“„Programmiertechniken“

Behoben: Die USW-Beispiele für Windows hatten schon geraume Zeit kein vernünftiges Bild in ***uctable*** dargestellt. Die Beispiele zeigen jetzt Icon in der richtigen Größe an.

Behoben: Tasten, die der IDM für die Navigation verwendet (bspw. `TAB`, `Cursor links` usw.), wurden unter Microsoft Windows nicht an Kindobjekte des USW-Widgets weitergeleitet obwohl im **UC_I_querykey** Inquire die Taste als benötigt gekennzeichnet wurde.

Behoben: Das umgebende IDM Objekt von USW Widgets besitzt jetzt keinen Rahmen mehr. Das USW Widget sollte für seinen Rahmen selbst zuständig sein.

1.12 IFD

Änderung: Die IFD Option wird nicht mehr unterstützt.

1.13 Demos

IDM-13189: In der README Datei der Beispiele stand nur, dass das Project „usw\usw“ nicht gestartet werden kann. Ein Hinweis auf das Projekt „usw\ucsample“ fehlte.

Neu: USW Beispiel ***ucgrid*** eingefuehrt um *HiDPI*-Support für USW Widgets aufzuzeigen.

Behoben: Das **ucframe** Widget der USW-Beispiele hatte unter Windows bei der Größenberechnung die gesetzte Schriftart nicht berücksichtigt, dadurch wurde Überschrift abgeschnitten.

1.14 Dokumentation

Neu: Kapitel „HighDPI Unterstützung“ im Handbuch „Programmiertechniken“ hinzugefügt.

Änderung: Dokumentation zu **DM_GetToolkitData** und **DM_GetToolkitDataEx** grundlegend überarbeitet. Dokumentation für **ToolkitDaten-Struktur DM_ToolkitDataArgs** im Handbuch „C-Schnittstelle - Grundlagen“ erstellt.

IDM-13198: Wichtige Hinweise, dass die Option **opt_hscroll** am ***poptext*** unter Microsoft Windows problematisch ist, nun in die entsprechenden Kapitel des Attribute Handbuch *.options[enum]* sowie Objekte Handbuch ***poptext*** übernommen.