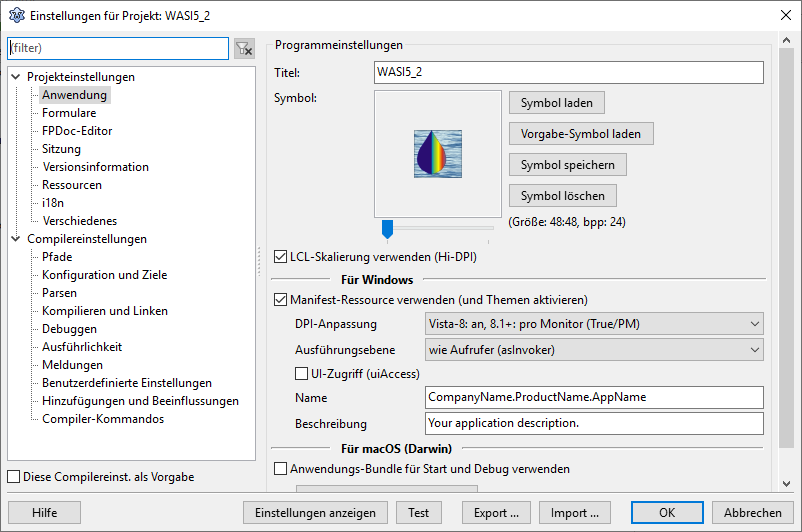
Umstellung auf High DPI

**wichtig**: in der Projekt-Datei (Wasi5\_2.pas) muss im Code „Application.Scaled := true“ gesetzt werden (<https://www.lazarusforum.de/viewtopic.php?f=5&t=11700>). Dann wird alles im kompilierten Programm entsprechend der DPI hoch- oder runter-skaliert. Sonst behalten die Formulare ihre Größe, wie sie in den .lfm Dateien stehen.

**aufpassen**: wenn man in den Projekteinstellungen etwas ändert, wird dieser Wert automatisch entfernt. Danach muss also „Application.Scaled := true“ wieder gesetzt werden. (meine Theorie: ein Bug in der Entwicklungsumgebung)

**weitere Einstellungen in den Projekteinstellungen**: Um die DPI-awareness im kompilierten Programm zu ermöglichen, müssen Haken bei „LCL-Skalierung verwenden“ und „Manifest-Ressource verwenden“ gesetzt werden. Dann bei DPI-Anpassung entweder pro Monitor/V2 oder an(true) auswählen.



Die beiden Varianten unterscheiden sich darin, dass bei „pro Monitor“ die Applikation an die DPI von jedem angeschlossen Monitor angepasst wird. Bedeutet wenn man die Applikation von einem Monitor zum nächsten schiebt, ist es auf beiden Monitoren scharf und an die Größe angepasst (sonst wäre es nur auf dem Hauptmonitor scharf) (<https://docs.microsoft.com/de-de/windows/win32/hidpi/high-dpi-desktop-application-development-on-windows?redirectedfrom=MSDN> ).

Der neue Lazarus Editor (bei mir Version 2.0.6), im Vergleich zur 1.8.0 Version, ist selber DPI aware (<http://free-pascal-lazarus.989080.n3.nabble.com/Lazarus-LCL-High-DPI-td4050986.html> ).

Beim Öffnen des Forms (.lfm-Datei) liest er den „DesignTimePPI“-Wert aus (ist er nicht vorhanden, dann liegt der Standardwert von 96 vor). Anhand von dem Wert und der DPI vom Monitor skaliert der Editor automatisch alle Werte die in der .lfm Datei stehen (mit Rundungsfehlern, da nur Integer-Werte akzeptiert werden).

Diese skalierten Werte werden dann beim Speichern/Kompilieren in die .lfm-Datei übernommen und der DesignTimePPI-Wert wird auf die aktuelle Monitor-Einstellung gesetzt.

**Problematisch** könnte dies sein, wenn zB auf Linux die Skalierung + Manifest-Datei nicht funktioniert. Dann werden im kompilierten Programm alle Werte so angezeigt, wie sie in den .lfm-Dateien stehen. Dies hat zur Folge, dass das angezeigte Formular zu groß oder zu klein (je nach DPI des Monitors, wo das Formular zum letzten Mal geändert wurde).

Maßnahmen:

**erster Schritt**

Alle DesignTimePPI-Werte von den .lfm-Dateien werden so angepasst (auf 96), dass sie auf einem Full-HD Bildschirm im Editor gut dargestellt werden. Full-HD bedeutet eine DPI von 96 und ist das verbreitete Format.

Dann kompilieren und die Werte sind alle so angepasst, dass das Programm mit einem 96DPI Monitor ohne DPI-awareness passend angezeigt wird.

Meine Theorie warum dieser Schritt überhaupt nötig war: die alte Lazarus Version (1.8.0) hat die DPI während dem Designen erkannt und in die .lfm-Datei geschrieben. Die anderen Werte bezüglich Position/Breite… der Komponenten werden nicht angepasst. Der neue Editor (Version 2.0.6) hat dann alles anhand des DesignTimePPI-Wertes skaliert, obwohl alle Werte sich immernoch auf eine DPI von 96 beziehen.

wichtig: wenn etwas in der .lfm-Datei geändert wird, dass sollte das Projekt in Lazarus nochmal geöffnet werden, damit die Änderungen übernommen werden.

Werden nun alle notwendigen Einstellungen (siehe Seite 1) getroffen damit das kompilierte Programm DPI-aware ist, kann das Programm auf Monitoren mit unterschiedlicher DPI ausprobiert werden.

Die Elemente werden teilweise nicht an ihren vorgesehenen Positionen angezeigt und überlappen sich teilweise. Die Haupt-GUI selbst hat hier die größten Probleme. (Dieses Problem tritt auch im Lazarus Editor auf, wenn am Anfang keine 96 beim DesignTimePPI-Wert angegeben ist.)

Der erkennbare Grund für die falschen Positionen bestimmter Elemente liegt an der Schriftgröße. In den aktuellen Einstellungen zum Font ist das „Default\_Charset“ als Charset und „MS Sans Serif“ Name ausgewählt. Die Werte für die Schrifthöhe und die Font-Size (Font-Size wird aus der Schrifthöhe errechnet) werden an die DPI angepasst.

Das erkennbare Problem ist, dass sich die Schrift bei einer Font-Size kleiner 9 nicht mehr ändert (edit: 7 habe ich hinbekommen. Dies hat zur Folge, dass die Schrift mit anderen Elementen der GUI überlappt. Auch einige

**zweiter Schritt**

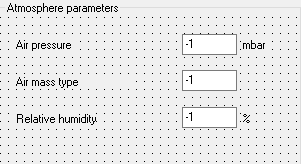
Um das Grundlegende Problem mit der Schrift zu lösen, werden auf einem 96 DPI System alle Schrifthöhen so einstellen, dass die Font-Size nicht kleiner als 9 ist. Die Einstellungen zum Charset (default\_charset) und dem Namen (Ms Sans Serif) werden beibehalten. Wichtig ist auch, dass überhaupt eine Schrifthöhe eingestellt wird. Ist der Wert 0, dann wird die Schrift nicht skaliert und bleibt über alle Bildschirme in derselben Größe.

Dies kann nur bedingt das Problem lösen, da immer Rundungsfehler auftreten werden und nicht jede Schriftgröße im Charset unterstützt wird.

Es könnte sich demnach lohnen mit anderen Charsets auszuprobieren, ob die Skalierung damit besser funktioniert. Beispielsweise im Greek\_charset sind alle Stufen möglich (getestet für die Schriftgröße 1-20). Wenn man also sicher gehen will, dass die GUI auch auf Monitoren kleiner 96DPI läuft, muss der Charset umgeändert werden. Da die Buchstaben mit dem Charset verändert ausschauen, müssen auch andere Schrifthöhen eingestellt werden und eventuell auch Elemente neu positioniert werden.

**dritter Schritt**

Das Problem der Schrift ist nicht völlig beseitigt. Bei normalen Textfeldern könnte es schlimmstenfalls daraus auswirken, dass ein Teil des letzten Buchstabens nicht ganz zu sehen ist. Einige Schriftzüge sind aber durch Lehrzeichen an ihre Position gerückt worden. Dies kommt beispielsweise bei Eingabefeldern vor, wo nach dem TEdit noch eine Einheit angegeben ist (wie im Bild dargestellt; da standen „Air pressure“ und „mbar“ im selben TStaticText).

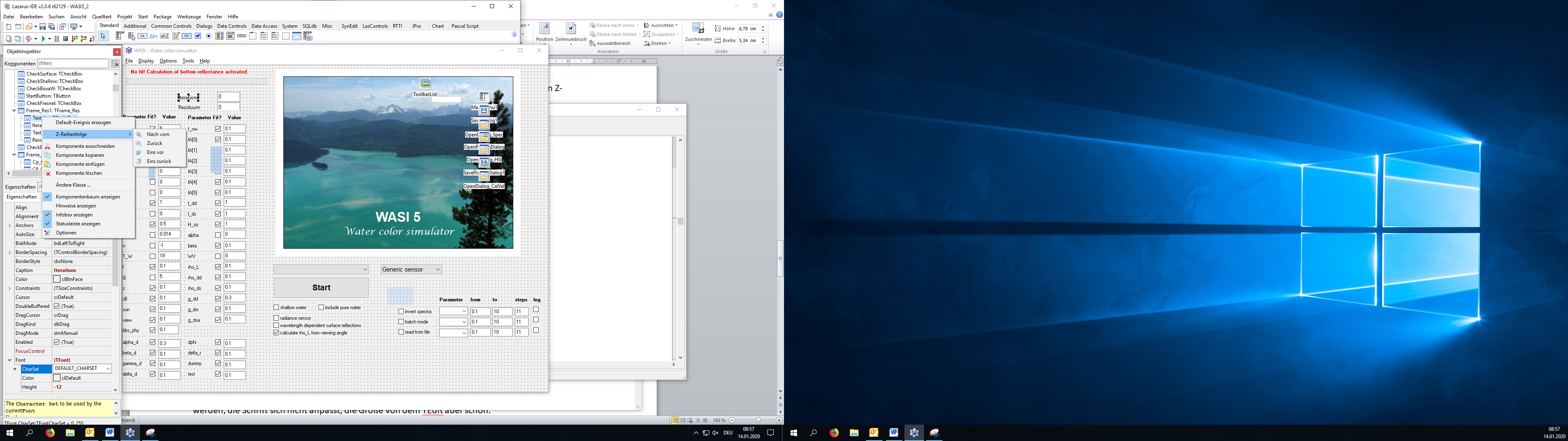


Skaliert die Schrift mal nicht richtig, dann ist die Position von „mbar“ nicht richtig. Entweder sie wird teilweise hinter dem TEdit verschwinden, oder sie geht über die Grenze des TGroupBox hinaus und verschwindet.

Um das zu verhindern werden solche Textfelder in separate Textfelder aufgeteilt.

**wichtig**: Die Position der Textfelder wird anhand der oberen linken Ecke festgelegt. Der Fehler beim Skalieren wird also wahrscheinlich an der rechten Kante auftreten. Beim Design der GUI sollte wenn möglich genügend Platz zwischen Textfeldern und weiteren Komponenten in der horizontalen liegen.

Zusätzlich werden Textfelder mit Autosize=True festgelegt. Dies verhindert, dass Text abgeschnitten wird. Ich habe es meist so eingestellt, dass sich die Textfelder links von TEdits sich in einer geringeren Z-Reihenfolge befinden. Das bedeutet, dass die Textfelder von den Edits überdeckt werden, sollte der Text durch das Skalieren mal zu lang werden. (Reihenfolge änderbar im Objektinspector mit Rechtsklick auf eine Komponente und dann Z-Reihenfolge auswählen) Das was weiter unten steht, überdeckt das was über ihn steht.



ein weiteres Problem mit überlappenden Komponenten entsteht durch den TEdit.

Standardmäßig ist BorderStyle „bsSingle“ eingestellt. Die Komponente

Wenn **Autosize aktiviert** ist, bedeutet das ein nicht-lineares ansteigen der Höhe der TEdit Komponenten.

Ist Autosize nicht aktiviert, wird der Text nicht vollständig in dem TEdit dargestellt.

Ein Problem ergibt sich eigentlich nur beim runterskalieren (bedeutet der Zielmonitor auf dem das kompilierte Programm läuft hat eine geringere DPI als der Monitor auf dem das Programm kompiliert wurde). Momentan sind die Lagen der Elemente auf 96 DPI angepasst. Werden die Positionen der Komponenten im Editor auf einem hochauflösenden Monitor nicht drastisch geändert, dann überlappen sich die Komponenten beim runterskalieren nicht.

Aufpassen sollte man nur, wenn ein komplett neues Formular auf dem hochauflösenden Monitor erstellt wird. Wird genug Platz zwischen den Komponenten über und unter dem TEdit gelassen, sollte es kein Problem geben.

von einem hochauflösenden Monitor zu einem Monitor mit geringerer DPI).

**Möglichkeit 1**: den BorderStyle auf „bsNone“ ändern und die Elemente wieder mittig platzieren. Dann befindet sich aber kein schwarzer Rand um die Eingabefelder.

**Möglichkeit 2:** wähle eine andere Schrift (zB Greek\_charset), die eine Font-size kleiner 9 ermöglicht. Der Abstand vom Rand zum Text ist zwar dann trotzdem kaum verändert und könnte mit einer anderen Komponente überlappen, aber es ist in einem akzeptablen Maß.

Ich habe erstmal keine Änderung an den TEdits vorgenommen. Denn das Problem ist mit dem aktuellen Design erst beim runterskalieren auf Monitore mit einer Auflösung kleiner 96 DPI relevant. Und dort passt die Schriftgröße auch wieder nicht, da eine Font-Size gebraucht wird, die kleiner 9 ist. Für solche Systeme (und Linux-Systeme, da ich glaube es wird dort nicht funktionieren) könnte es nützlich sein eine Version der Software zu haben, die nicht DPI aware ist. **wichtig:** wenn das kompilierte Programm nicht DPI-aware sein soll, sollten alle Formulare vor dem Kompilieren im Editor geöffnet werden, damit sie eine einheitliche Größe haben.

Es sollte dann beim Designen auf Monitoren mit höherer DPI darauf geachtet werden, dass bestimmte Elemente (zB Check-Box) einen Rand von wenigen Pixeln um sich haben und der sich nicht anpasst. Elemente sollten also nie zu nah beieinander stehen.

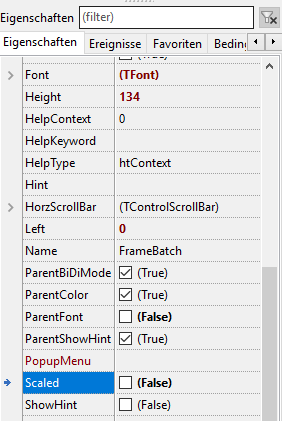
**Problem bei Haupt-GUI beim Öffnen im Editor**

Wird die gui.lfm Datei im Editor auf einem Monitor mit anderer DPI angezeigt, werden die Elemente der 3 Frames (batch, par und resid) nicht richtig dargestellt.

Meine Theorie: der Editor skaliert hier die Elemente doppelt. Beim Öffnen des Projekts erkennt der Editor den Unterschied der Monitor-DPI zur DesignTimePPI und möchte die Elemente skalieren. Da die Elemente von den Frames in die gui integriert werden, wird das Frame skaliert. Dann werden die Elemente (schon skaliert) auf die gui gesetzt. Die gui selbst muss aber auch noch skaliert werden und der Editor nimmt die Änderungen für alle Elemente vor. Also auch für die der schon skalierten Elemente der drei Frames.

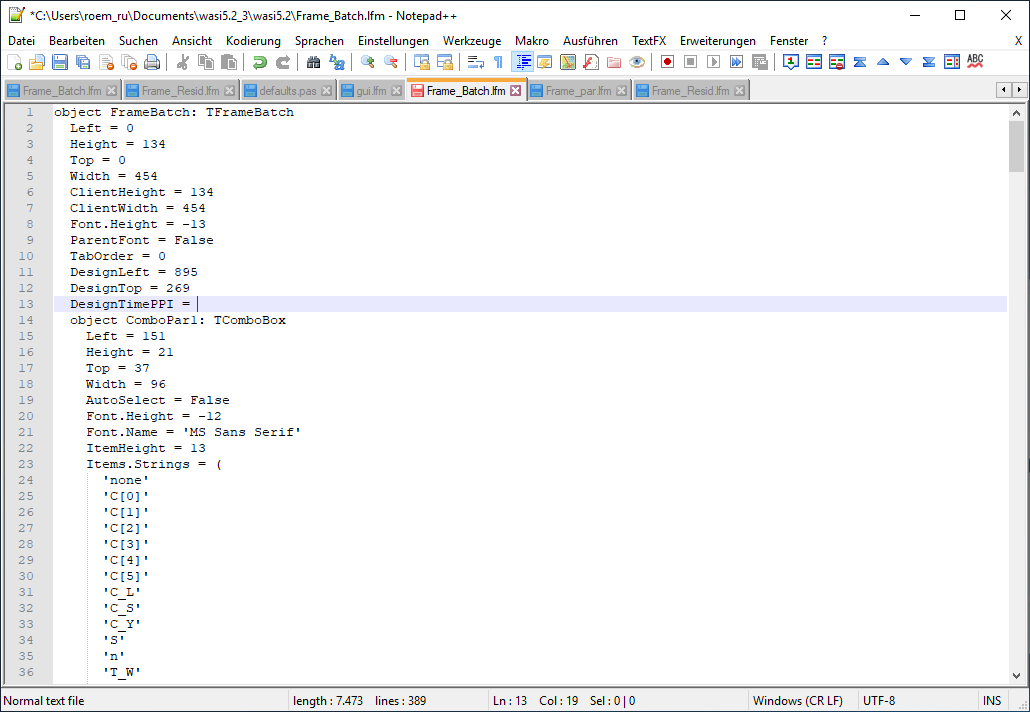
Aber das ist nur eine Theorie, mit der sich aber erklären lässt, warum die Lösung funktioniert.

**Möglichkeit1:** Setze die Eigenschaft Scaled = False für die drei Frames (batch, par und resid), dann werden die Elemente auch nicht mehr im Editor skaliert.



**Möglichkeit2:** Wenn die drei Frames auch skaliert im Editor dargestellt werden sollen, ist es ein bisschen komplizierter.

Man öffnet die .lfm Datei zB in Notepad und füge die Zeile „DesignTimePPI = …“ hinzu oder passe sie an. Der Wert muss die DPI vom aktuellen Monitor sein. Bei der gui.lfm Datei muss nichts manuell geändert werden.



Dann öffnet man das Projekt im Editor und die Hauptgui lädt richtig. Die Hauptgui sollte gespeichert werden (muss vielleicht das Formular ein bisschen bewegen, damit überhaupt erkannt wird, dass sich etwas geändert hat) und den Editor wieder schließen. Den DesignTimePPI Wert der 3 Frames wieder auf dem vorherigen Wert setzen und speichern. Im Editor öffnen und die 3 Frames speichern (wieder ein bisschen bewegen, damit der Editor die Änderung erkennt).

Fehlt der zweite Schritt sieht die GUI beim ersten Öffnen noch richtig aus, aber nach dem Neuzeichnen zB bei Monitor-Wechsel sind die Elemente falsch skaliert.

**Empfehlung**: am besten von den .lfm-Dateien regelmäßig eine Sicherheitskopie anlegen, da vor allem in der Hauptgui schnell etwas nicht richtig wird.

**Problem mit den Konstanten**

Hier liegt das Problem in der Event-Reihenfolge.



<https://wiki.freepascal.org/Event_order/de>

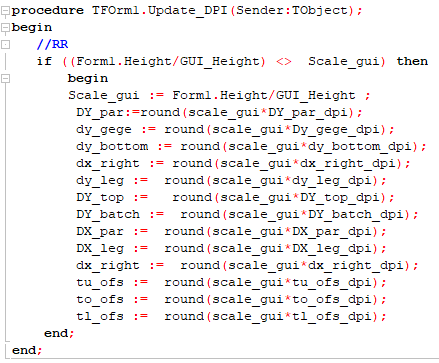
Erst nach OnResize werden die Elemente an die DPI skaliert. Alle Events die vorher aufgerufen werden sehen eine DPI vom vorherigen Monitor (bzw. die DPI beim Designen der Frames).

Das gilt auch für Scale96ToForm. Wird diese Funktion zum Anpassen der Konstanten im FormCreate() verwendet, werden die Konstanten an die DPI zur Design-Zeit angepasst und nicht an die aktuelle DPI des Monitors der das Programm startet.

**Möglichkeit 1**: Pro-Monitor

Ein OnResize Event errechnet sich aus der

Momentan kann die Größe des Fensters im kompilierten Programm nicht per Maus verändert werden und Form1.Height ändert sich demnach nur beim Skalieren. Form1.Height wird mit der Konstanten GUI\_Height (entspricht Form1.Height bei 96 DPI) verglichen und als Skalierungsfaktor für die restlichen Konstanten gesetzt.



Problem: FrameGroesse ist als OnResize festgelegt also wird die Prozedur zum Skalieren der Konstanten innerhalb von FrameGroesse aufgerufen.

Problem: Das Skalieren beim Wechsel zu einem Monitor mit anderer DPI dauert etwas länger. Sollte das ein Problem für den Programmablauf sein, sollte nicht die pro-Monitor Einstellung gewählt werden.

**Möglichkeit 2**: kein pro-Monitor

der obere Code muss trotzdem hinzugefügt werden. Stellt man pro-Monitor aus vermeidet es das lange skalieren bei Wechsel zwischen Monitoren mit unterschiedlicher DPI.

Noch nicht gelöst: Bild auf der Hauptgui skaliert nicht mit.