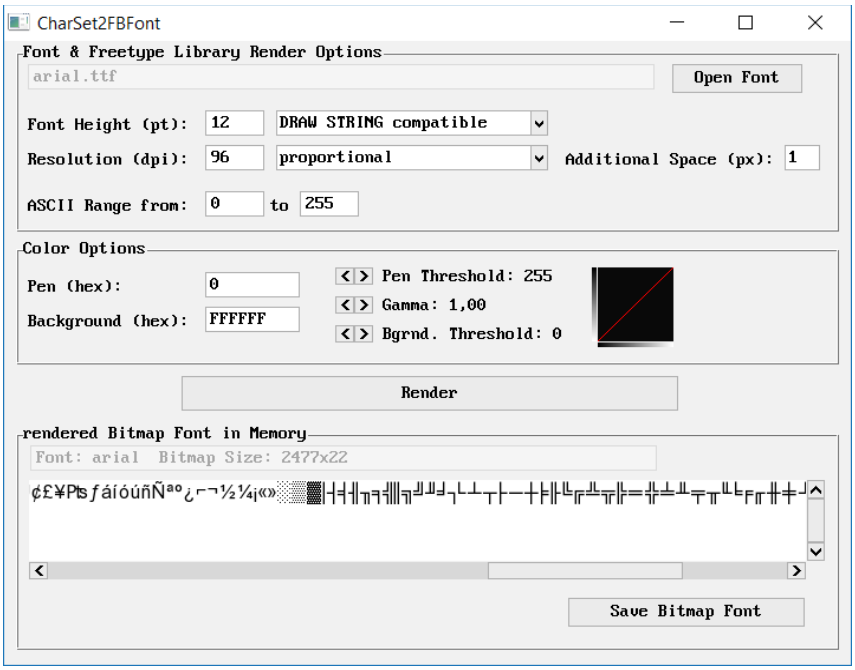
Charset2FBFont v2.03

Dieses Programm wandelt, so gut es geht, einen ausgewählten Vektorfont in eine Bitmap um. Eine zusätzliche Datei liefert alle weiteren Informationen um später Zugriff auf die entsprechenden Teilgrafiken innerhalb der Bitmap zu bekommen.
Dieses Programm ist ein kleines Remake, nun in FreeBASIC geschrieben.
Zum Rendern wird die FreeType Library als „Fremdsoftware“ benutzt. Daraus kann noch eine weitere Abhängigkeit an eine weitere Bibliothek entstehen.
Für Windows Rechner werden die entsprechenden Bibliotheken mitgeliefert.
Für Linux Rechner kann ich leider keine Aussage über Lauffähigkeit und Abhängigkeiten treffen.
Die Vorgängerversion ist erfolgreich unter Linux zum Laufen gebracht worden, ich gehe davon aus dass diese hier es auch tut.

Die neueste Version sollte nun einen Font erzeugen, der sich an die im Screen benutzte Codepage 437 hält, soweit der ausgewählte Font die entsprechenden Zeichen überhaupt bereitstellt.

Bedienung



Zuerst muß mit „**Open Font**“ ein beliebiger Font geöffnet werden. Es öffnet sich ein Dateidialog.
Bei meinen Tests habe ich aber festgestellt, daß dieser Dialog unter Windows nicht den Ordner „Fonts“ (im Ordner „Windows“) anzeigt. Eine andere Lösung als den Pfad per Hand einzugeben habe ich derzeit nicht parat, danach wird der Inhalt normal angezeigt. Möglicherweise ist dies aber auch einfach nur ein Zugriffsrechteproblem, keine Ahnung ;). Unter Windows 10 scheint es nicht zu bestehen.
Ist ein Font erfolgreich geöffnet worden, so wird dieser links oben angezeigt.

Mit „**Font Height (pt):**“ und „**Resolution (dpi):“** läßt sich die Größe (Höhe) des zu rendernden BitmapFonts beeinflussen. Die Fonthöhe wird hier in Pt angegeben. Resolution bezieht sich auf die Auflösung des Ausgabemediums, hier ist es der Monitor.
Der entstandene BitmapFont entspricht leider in den seltensten Fällen exakt der gewünschten Pixelhöhe.
Somit hilft derzeit nur ein „Spielen“ mit diesen beiden Werten und nochmaliges Rendern und auch ein wenig Kompromissbereitschaft ;).

Mit „**ASCII Range from / to:**“ läßt sich der Zeichenumfang einstellen, der im BitmapFont enthalten sein soll.

Mit „**DRAW STRING compatible**“ werden die Zeichen innerhalb der Bitmap so angeordnet, daß eine Nutzung des BitmapFonts mit dem DRAW STRING Kommando der FBGFX Bibliothek möglich ist.
In der darunter befindlichen Combobox kann man nun zwischen „**proportional**“ und

„**monospaced**“ wählen.

Im Falle von **proportional** kann man in der rechten Eingabebox die Anzahl der Pixel **zwischen** den einzelnen Zeichen bestimmen. Also ein zusätzlicher Zwischenraum.
Bei **monospaced** definiert die selbe Eingabebox die einheitliche Breite aller Zeichen! Zu breite Zeichen werden einfach „beschnitten“. Es sollte also jedes Zeichen überprüft werden ob es gut (oder zumindest ausreichend) zu sehen ist.

Die „**smallest Bitmap**“ Variante versucht alle Zeichen auf engstem Raum unterzubringen, um die Bitmap so klein wie möglich zu halten. In diesem Fall ist die Bitmap mehr als ein „Store“ der Glyphs zu verstehen.
Proportional/monospaced wird nicht unterstützt, vielmehr wird versucht, die Zeichen entsprechend so zu setzen, wie es die FreeType Library selbst tun würde.
Diese Variante ist nicht DrawString-kompatibel und benötigt zur Darstellung dann eine spezielle „Plotroutine“. Die mitgelieferte „BitmapFontTest.bas“ zeigt entsprechende Beispiele.

Mit „**Pen (hex):**“ und „**Background (hex):**“ kann der Font und der Hintergrund entsprechend coloriert werden. Die Farbwerte sind hexadezimal anzugeben.

Die Grundlage für alle Farbberechnungen und Manipulationen ist eine 256Grayscale Bitmap, geliefert von der FreeType Bibliothek.
Aber gerade bei recht kleinen Fontgrößen (pt9 bis pt13) wirken die Fonts mitunter recht „verwaschen“. Mit dem kleinen „Kontrastwerkzeug“ ist es in Grenzen möglich, auch bei kleinen Fonts die Lesbarkeit zu steigern.
Einfach mal „herumspielen“ und anschließend auf Render klicken :) !!

Der „**Render**“ Button versucht dann aus allen oben genannten Einstellmöglichkeiten eine Bitmap zu generieren. Im Bereich unterhalb des Renderbuttons (rendered Bitmap Font in Memory) werden nach erfolgreichem Rendern die „Eckdaten“ des erstmal nur im Speicher(!) befindlichen Bitmapfonts angezeigt.

„**Save Bitmap Font**“ erzeugt einen Ordner mit Namen des Fonts und speichert darin die Bitmap als auch eine sogenannte Descriptor Datei, die alle relevanten Daten zum Font enthält.

Descriptor Datei

Diese Datei wird zusätzlich zum Font erzeugt und enthält alle Daten zum Font. Hier ein kleiner Auszug aus einer solchen Datei:

```
7309,124,1,97,32,127
32,0,49,0,0,10,97
33,49,39,19,94,10,6
34,88,63,43,41,10,3
.
.
.
.
125,7046,56,36,106,10,6
126,7102,58,38,16,10,8
127,7160,149,129,65,10,31
```

Die erste Zeile enthält folgende Daten in selbiger Reihenfolge durch Komma getrennt:
Bitmapbreite (px)
Bitmaphöhe (px)
PlotModus 1=DRAW STRING compatible, 2=smallest Bitmap
Baseline (px) Abstand der Grundlinie des Fonts von oben
ASCIICode erstes Zeichen
ASCIICode letztes Zeichen

Ab der zweiten Zeile erfolgt die Beischreibung der einzelnen Zeichen, Daten ebenfalls durch Komma getrennt:
ASCIICode des Zeichens das in dieser Zeile beschrieben wird
OffsetX (px) Abstand des Zeichens vom rechten Rand der gesamten Bitmap
AdvanceX (px) Laufweite des Zeichens innerhalb des Schriftbildes
GlyphWidth (px) Breite des Glyphs
GlyphHeight(px) Höhe des Glyphs
GlyphLeft (px) Abstand des Glyphs vom linken Rand
GlyphTop (px) Abstand des Glyphs vom oberen Rand

BitmapFont.bi

Mit dieser Include-Datei ist es in eigenen Programmen möglich die erzeugten BitmapFonts zu laden und darzustellen.Die BitmapFont.bi stellt dafür zwei Möglichkeiten zur Verfügung:

Zum einen ist ein Loader vorhanden, der den entsprechenden Font im Speicher als FB.Image „installiert“, sodaß er mit dem FreeBASIC Befehl DRAW STRING benutzt werden kann.
Voraussetzung dafür ist allerdings ein entsprechend kompatibel erzeugter BitmapFont.
Die zweite Möglichkeit besteht in der Erzeugung eines „Font Objektes“ mit eigener Plot Routine und einfacheren Manipulationsmöglichkeiten bezüglich Schriftfarbe und Hintergrundfarbe.

Die BitmapFontTest.bas zeigt entsprechende Beispiele