

# Bitmap Font Integration

## TrueType-Font Konvertierung

bmfont64.exe öffnen und TrueType-Font konvertieren zu einem PNG- und FNT-File.

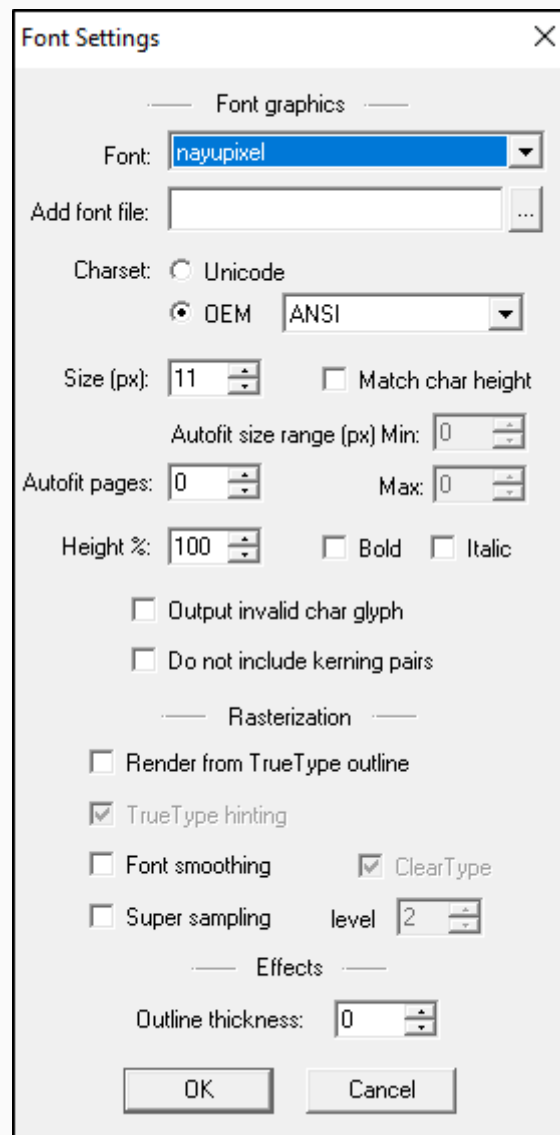


Figure 1: Import Settings.

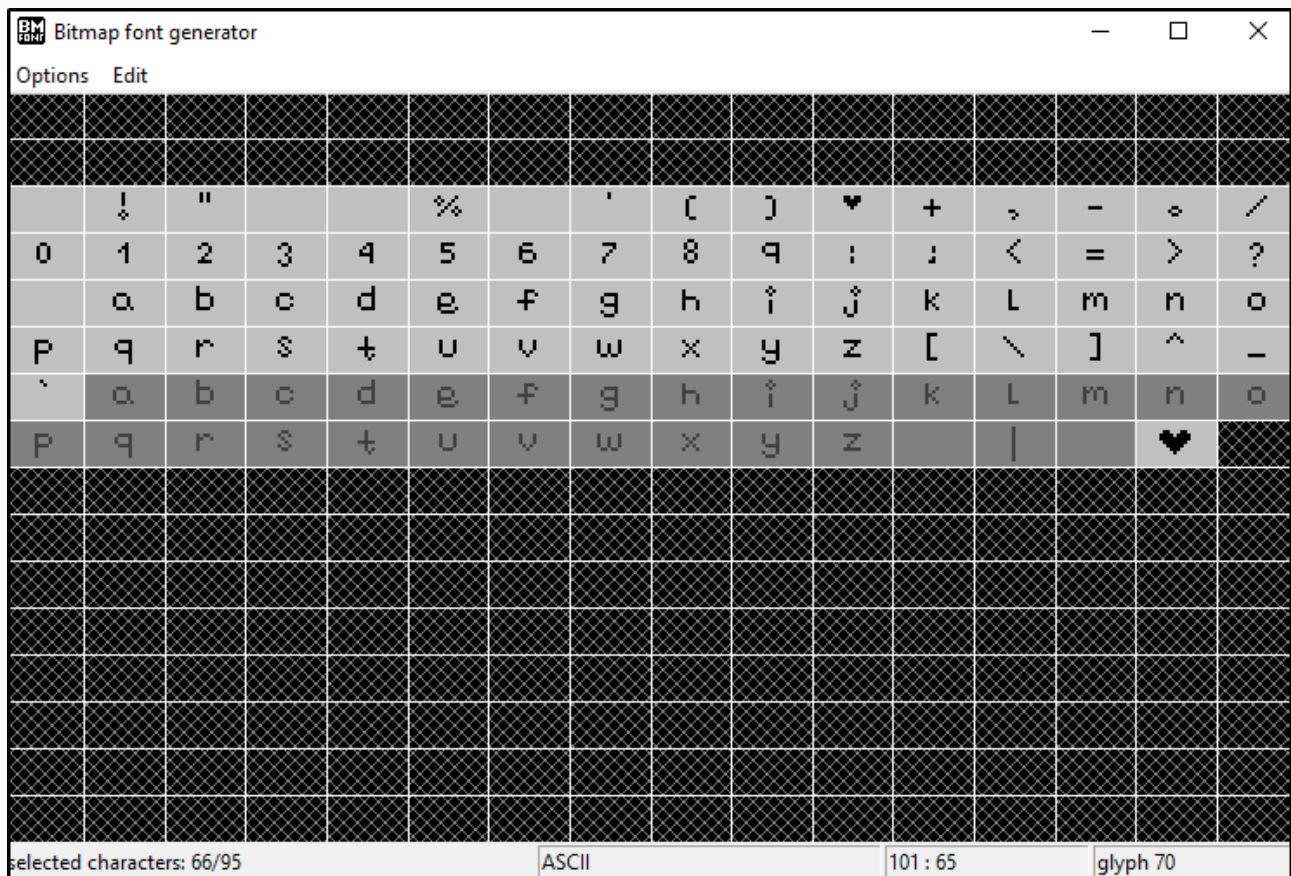


Figure 2: Die zu exportierenden Characters selektieren.

**Export Options**

Layout

Padding: 0 Spacing: A

0 A 0 1

0 B 1 C

☒ Equalize the cell heights

☐ Force offsets to zero

Adaptive padding factor: 0

Texture

Width: 256 Height: 32

Bit depth: ☐ 8 ☒ 32

☐ Pack chars in multiple channels

Chnl	Value	Invert
A:	glyph	<input type="checkbox"/>
R:	one	<input type="checkbox"/>
G:	one	<input type="checkbox"/>
B:	one	<input type="checkbox"/>

Presets: White text with alpha

File format

Font descriptor: ☒ Text ☐ XML ☐ Binary

Textures: png - Portable Network Graphic

Compression: Deflate

OK Cancel

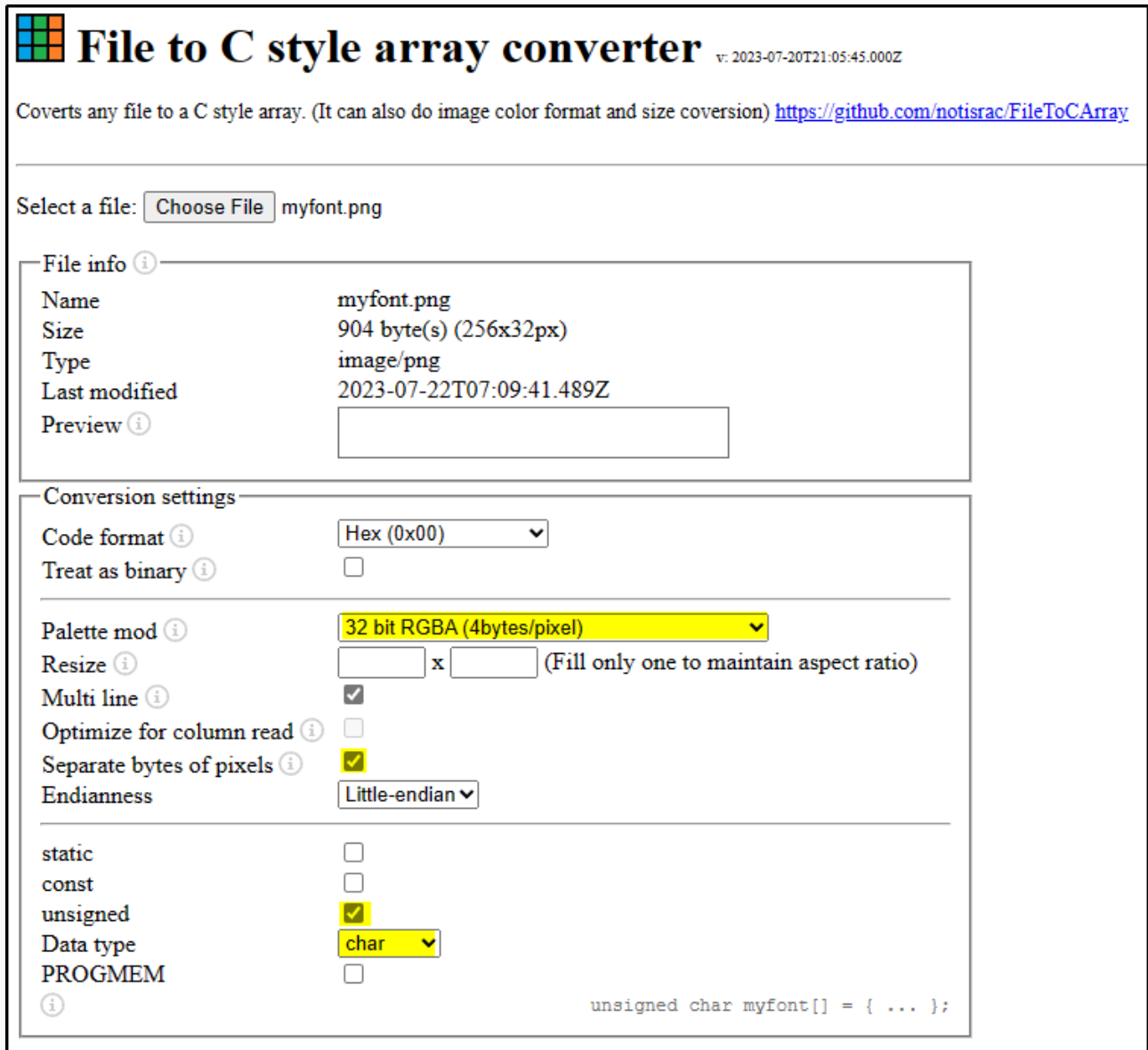
*Figure 3: Export Settings: Texture  
Width/Height möglichst optimal mit  
Power-of-Two-Dimensionen setzen.*

Vor dem Export mittels **Options** → **Visualize** kontrollieren, dass alle Characters auf eine Textur der gewählten Grösse passen.

Danach mit **Options** → **Save bitmap font as...** das Resultat als PNG- und FNT-Datei abspeichern.

## PNG Konvertierung zu C-Header

FileToArray öffnen und das exportierte PNG-File selektieren:



**File to C style array converter** v: 2023-07-20T21:05:45.000Z

Converts any file to a C style array. (It can also do image color format and size conversion) <https://github.com/notisrac/FileToArray>

Select a file:  myfont.png

**File info** ⓘ

Name	myfont.png
Size	904 byte(s) (256x32px)
Type	image/png
Last modified	2023-07-22T07:09:41.489Z
Preview ⓘ	<input type="text"/>

**Conversion settings**

Code format ⓘ

Treat as binary ⓘ ☐

Palette mod ⓘ

Resize ⓘ  x  (Fill only one to maintain aspect ratio)

Multi line ⓘ ☒

Optimize for column read ⓘ ☐

Separate bytes of pixels ⓘ ☒

Endianness

static ☐

const ☐

unsigned ☒

Data type

PROGMEM ☐

ⓘ `unsigned char myfont[] = { ... };`

Figure 4: Conversion Settings.

Auf den Button Convert klicken und das Resultat als Header-Datei im C-Project inkludieren in `resources.c`. Dort muss in der Prozedur `fw_resource_init()` die Font-Textur initialisiert und zugänglich gemacht werden.

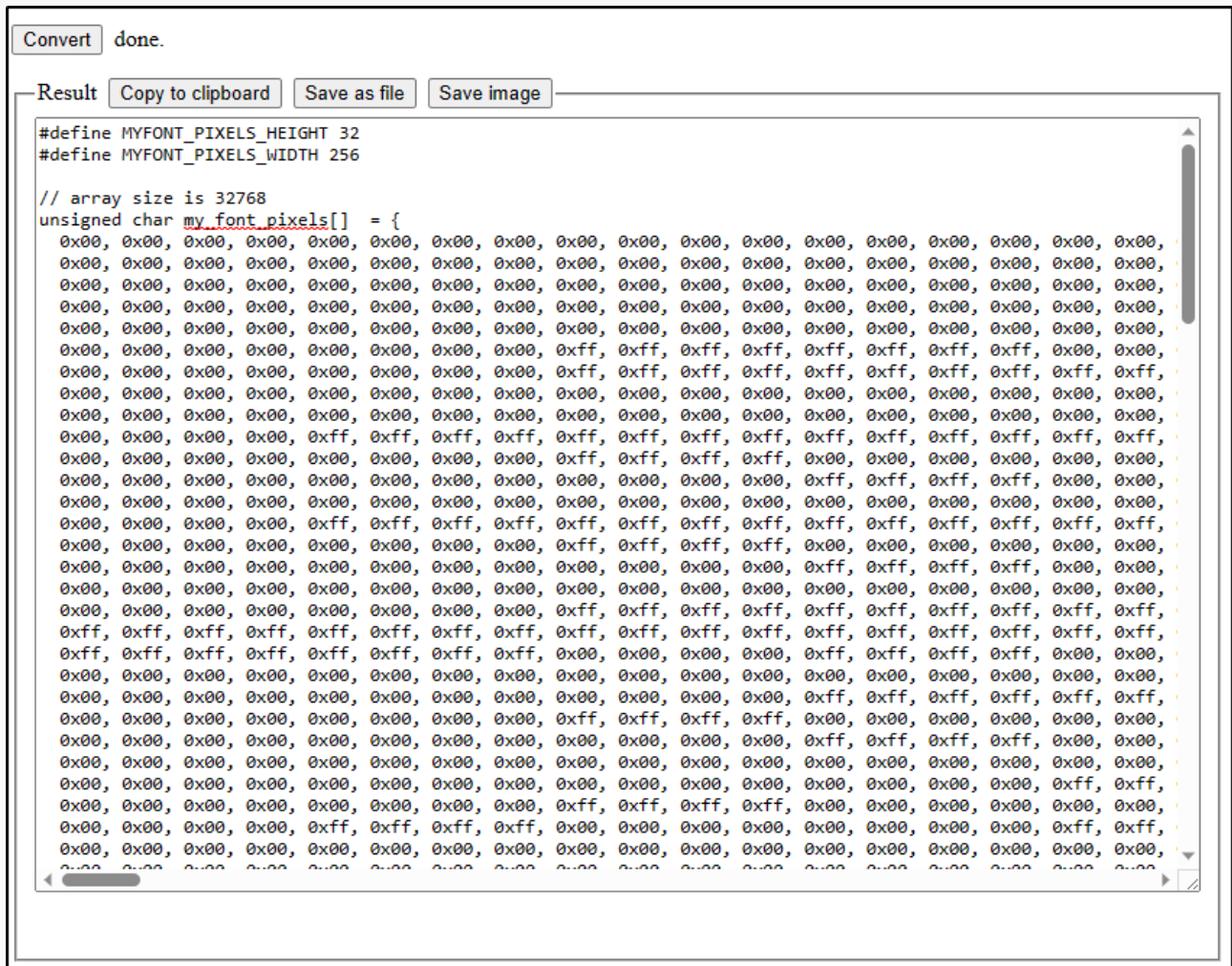


Figure 5: Resultat.

Der Name der generierten Variable sollte my\_font\_data, my\_font\_pixels o.ä. gewählt werden.

## FNT Konvertierung zu C-Struct

Mittels sh/create\_font\_meta.sh die FNT-Datei zu einer C-Header-Datei konvertieren. Diese enthält die Font-Metadaten (Texturkoordinaten, Character-Dimensionen, usw.), welche beim Font-Rendering benötigt werden.

### Usage:

```
./create_font_meta.sh <file> <name>
```

### Arguments:

```
file  Bitmap font metadata in fmt format.
name  Name of the font structs.
```

Beispiel:

```
$ ./create_font_meta.sh myfont.fnt my_font | tee my_font_meta.h  
$ unix2dos my_font_meta.h
```

Es wird dadurch die Datei `my_font_meta.h` erstellt. Diese enthält die beiden Structs `_my_font_fontGlyphs` und `_my_font_fontFace`.

Diese Header-Datei ist in `font.c` zu inkludieren. In der Prozedur `fw_initFonts()` sind die Metadaten des Fonts mittels Aufruf zu `fw_initFont(fw_font_face*)` zu initialisieren.

Die Prozedur `fw_initFonts()` ist im generellen Programm-Ablauf einmalig auszuführen.