PySNES –
Ein SNES Emulator in PYTHON

Grafiken erstellen

Inhalt

- Grafiken erstellen
- BMP mit GIMP erstellen
- BMP mit Pythonskript einlesen
- BMP zu SNES Kodierung

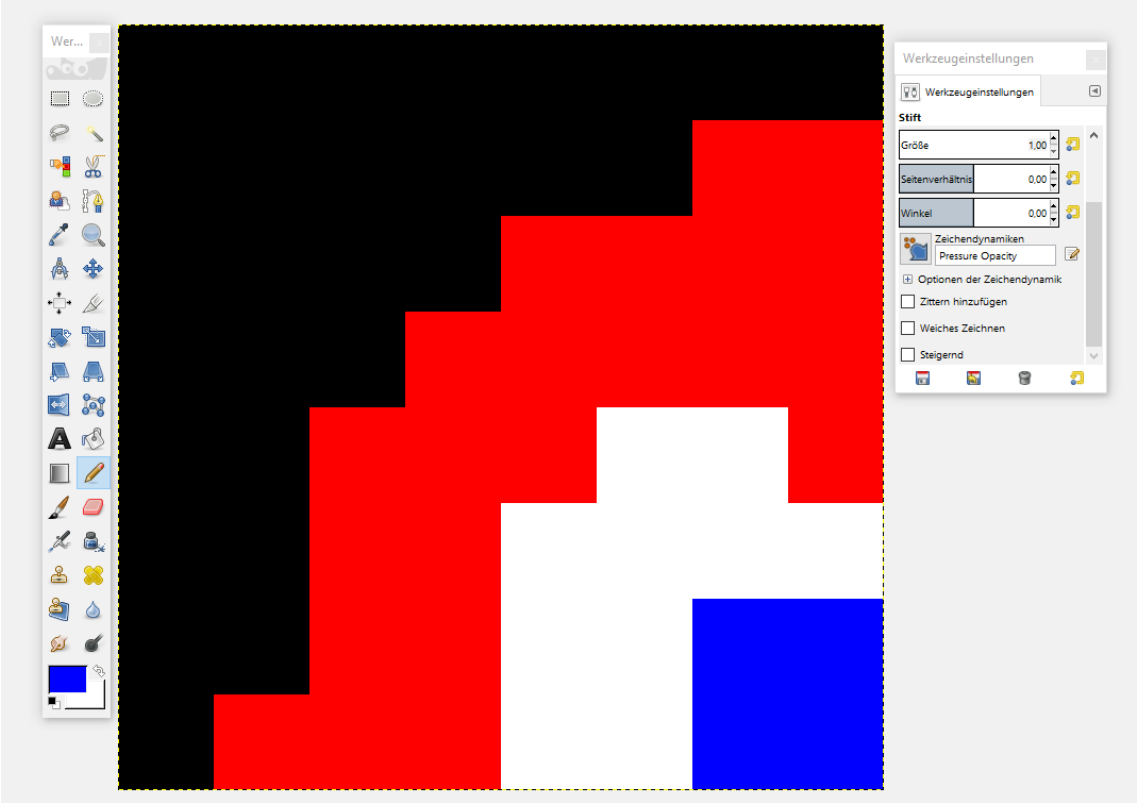
Grafiken erstellen

- Diese Folien erklären wie man SNES Grafiken mit GIMP erstellt und dann in die korrekte Folge von Bytes umwandelt
- Hierzu wird ein 8 mal 8 Pixel Bitmap erstellt: Eine BMP Datei.

  Also nix kompliziertes wie gif oder jpg!
- Dann lesen wir das Bild mit nem Pythonskript ein
- Und am Ende erzeugen wir WLA Code, den wir dann nutzen können.

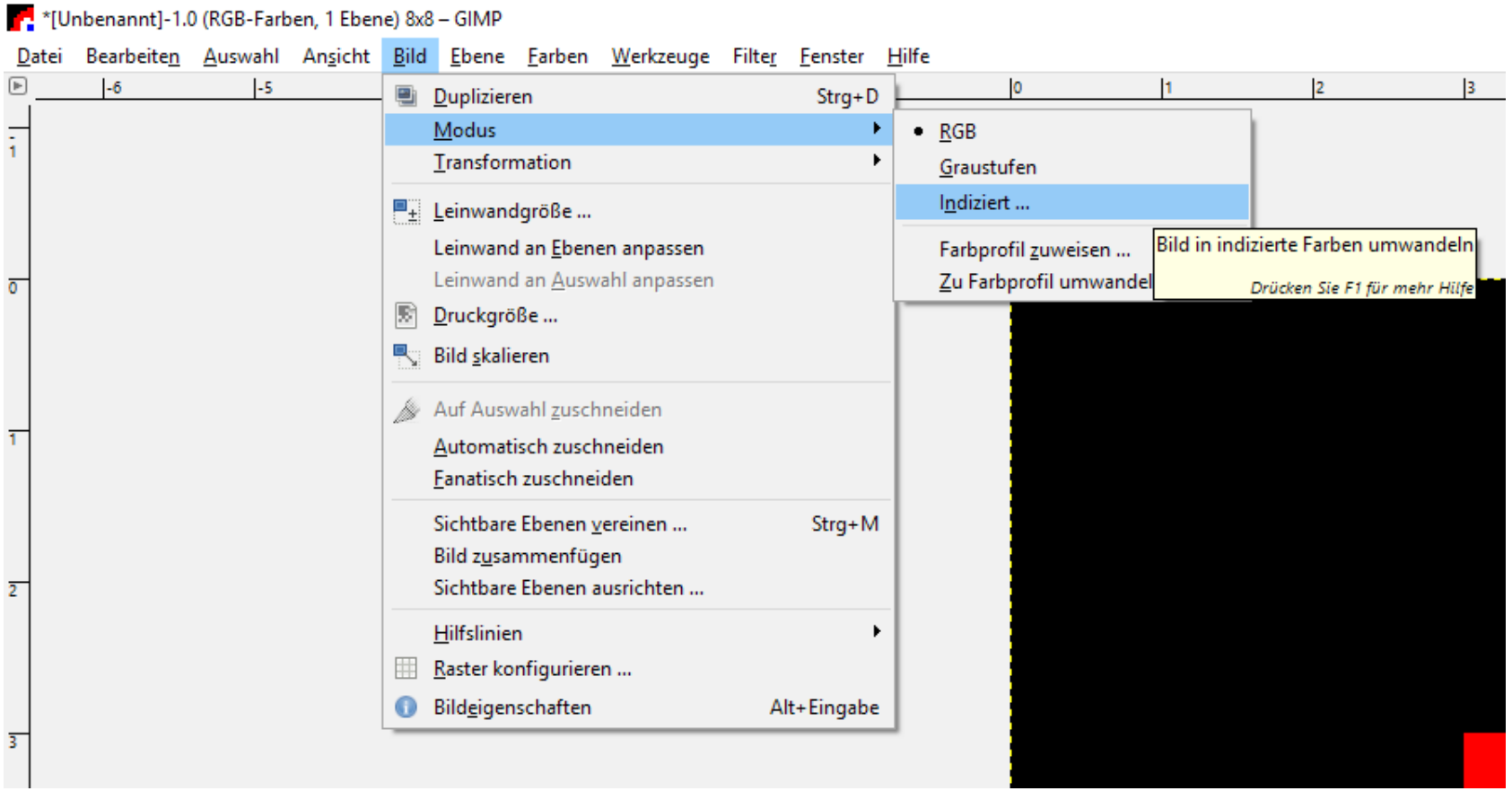
BMP mit GIMP erstellen

- Eine 8x8 Grafik erstellen
- Pixelgröße des Stifts auf 1.00 und zoomen



BMP mit GIMP erstellen

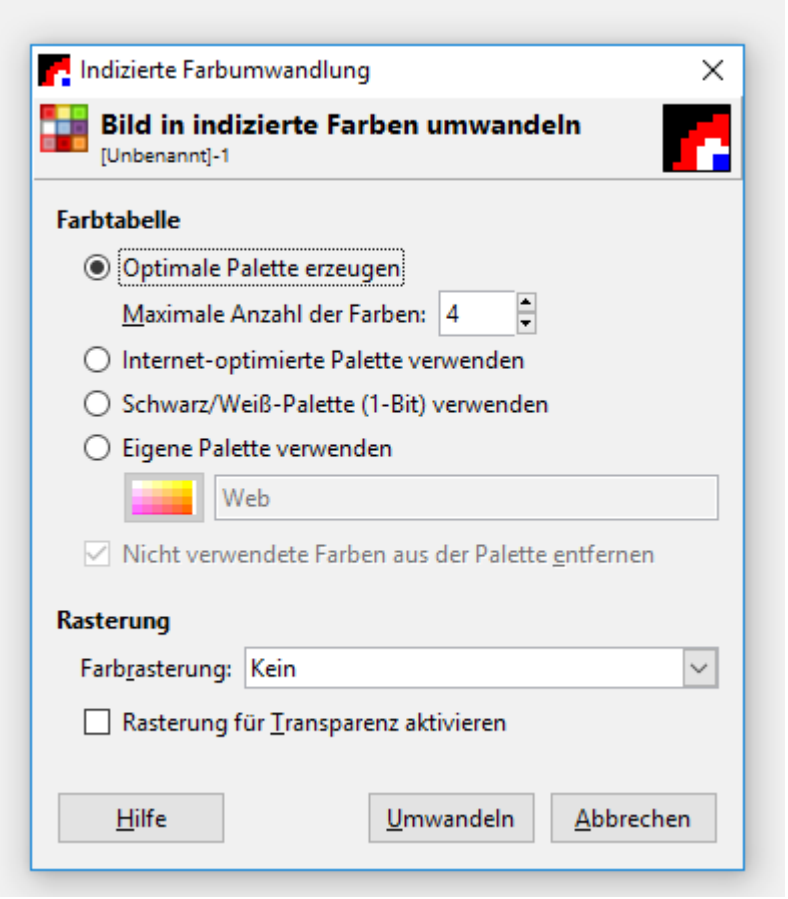
- Den Modus des Bildes auf Induziert stellen



BMP mit GIMP erstellen

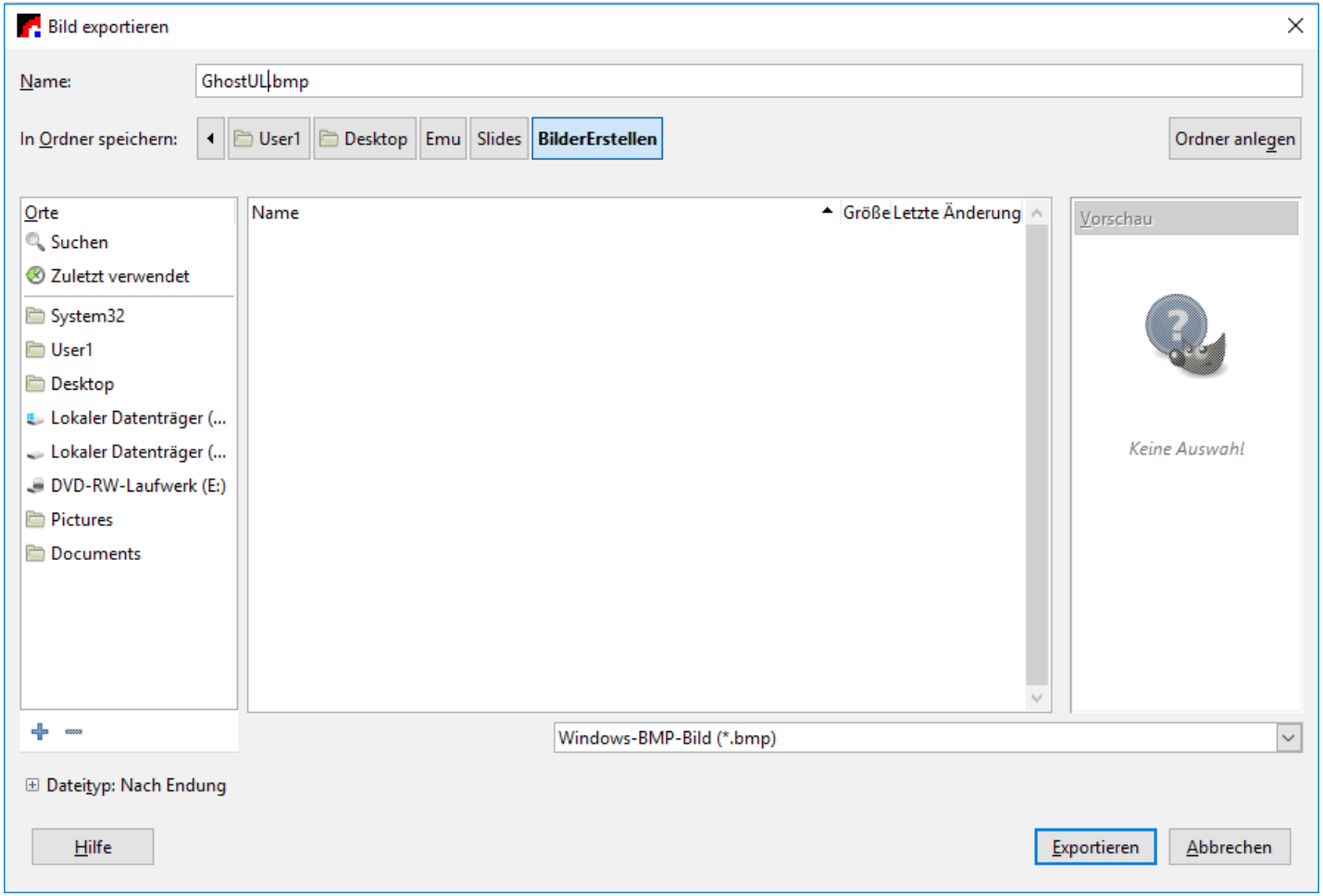
- Auf 4 Farben stellen
- 2BPP(2Bit pro Pixel)

Mögliche Modi:
- 2BPP (2 Bit): 4 Farben
- 4BPP (4 Bit): 16 Farben
- 8BPP (8 Bit): 256 Farben



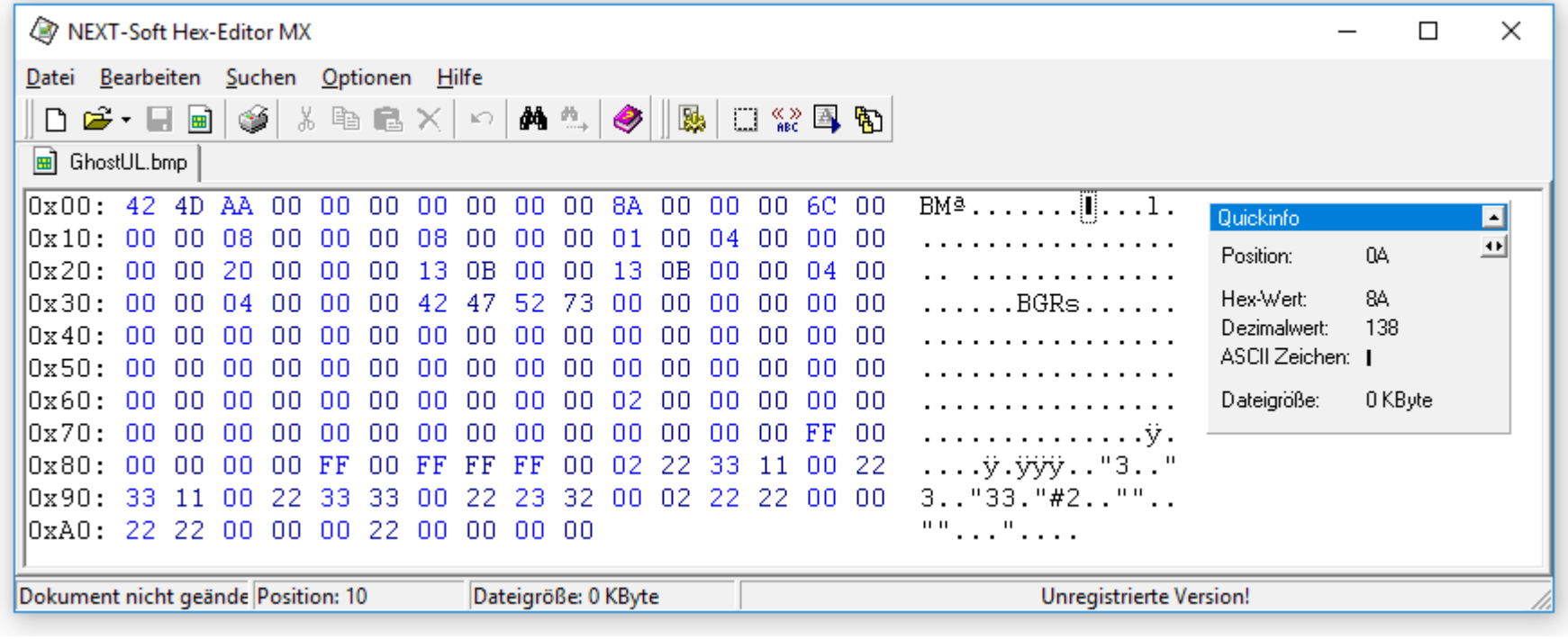
BMP mit GIMP erstellen

- Und dann als BitMap (*.bmp) exportieren



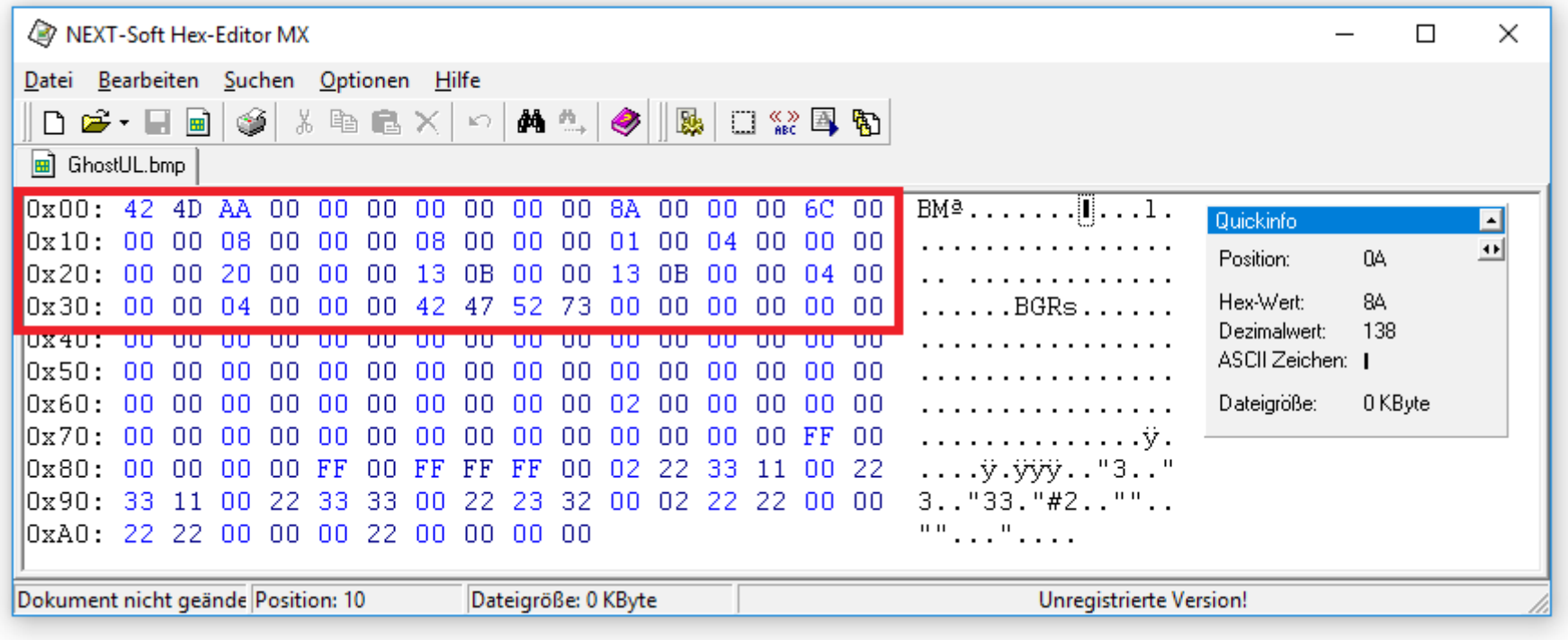
BMP mit Pythonskript einlesen

- Man kann eine Bilddatei auch mit einem Hexeditor öffnen. Unsere Datei sieht so aus:



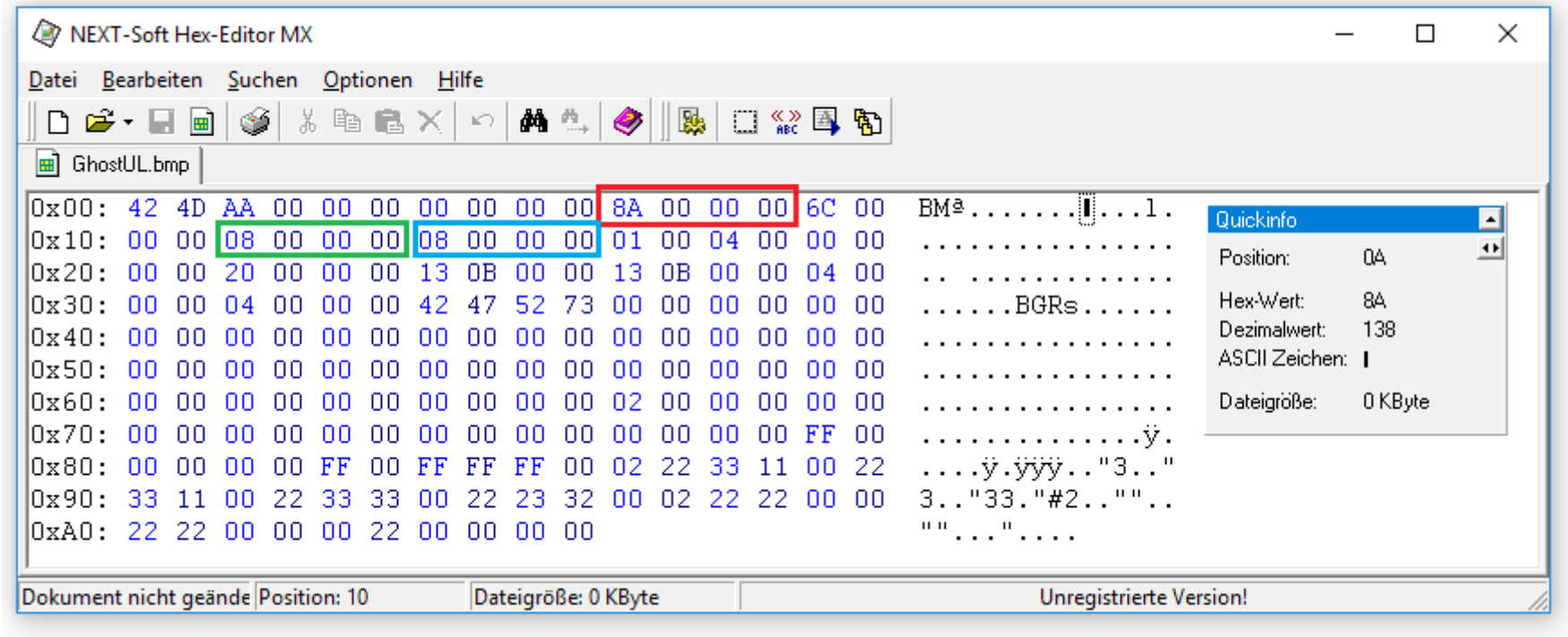
BMP mit Pythonskript einlesen

- Die ersten Vier Zeilen enthalten den Header
- Hier werden nur Bitmaps vom Typ BM (0x424D) besprochen (erste Zwei Bytes)



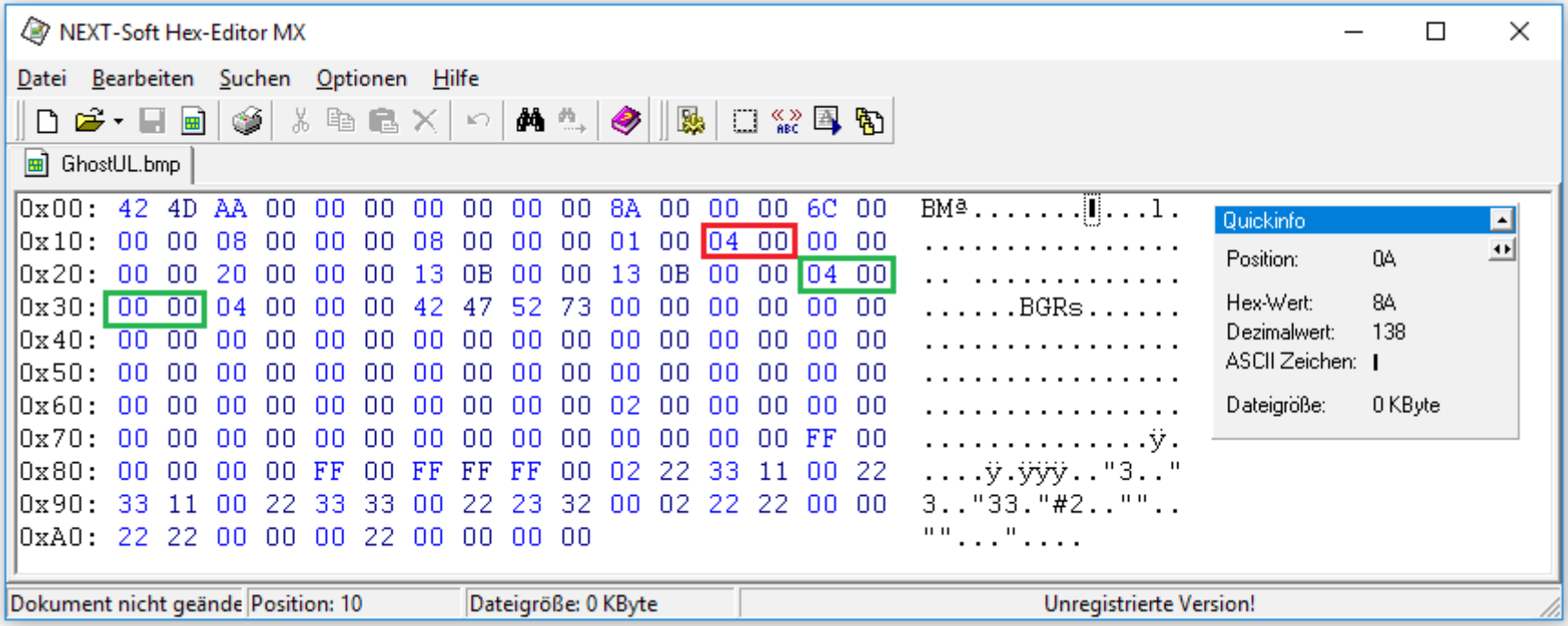
BMP mit Pythonskript einlesen

- Weitere Kopfdaten:
- Byte 10-13: Start des Pixelarrys z.B 0x8A (also 128)
- Byte 18-21: Bildbreite 0x08 Pixel
- Byte 22-25: Bildhöhe 0x08 Pixel



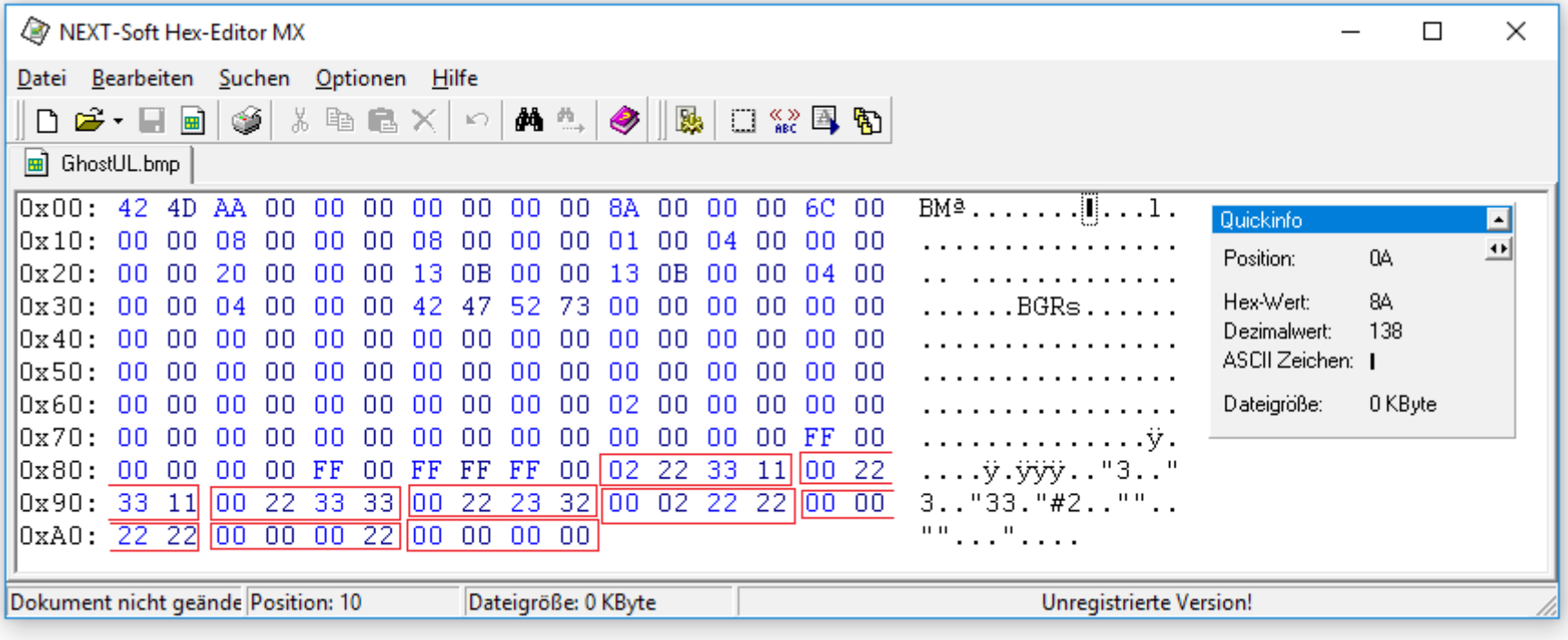
BMP mit Pythonskript einlesen

- Weitere Kopfdaten:
- Byte 28-29: Bits pro Pixel (BPP) z.B. 4
- Byte 46-49: Anzahl der Farben in der Palette z.B. 4
- Andere Werte brauchen wir nicht und werden hier nicht besprochen



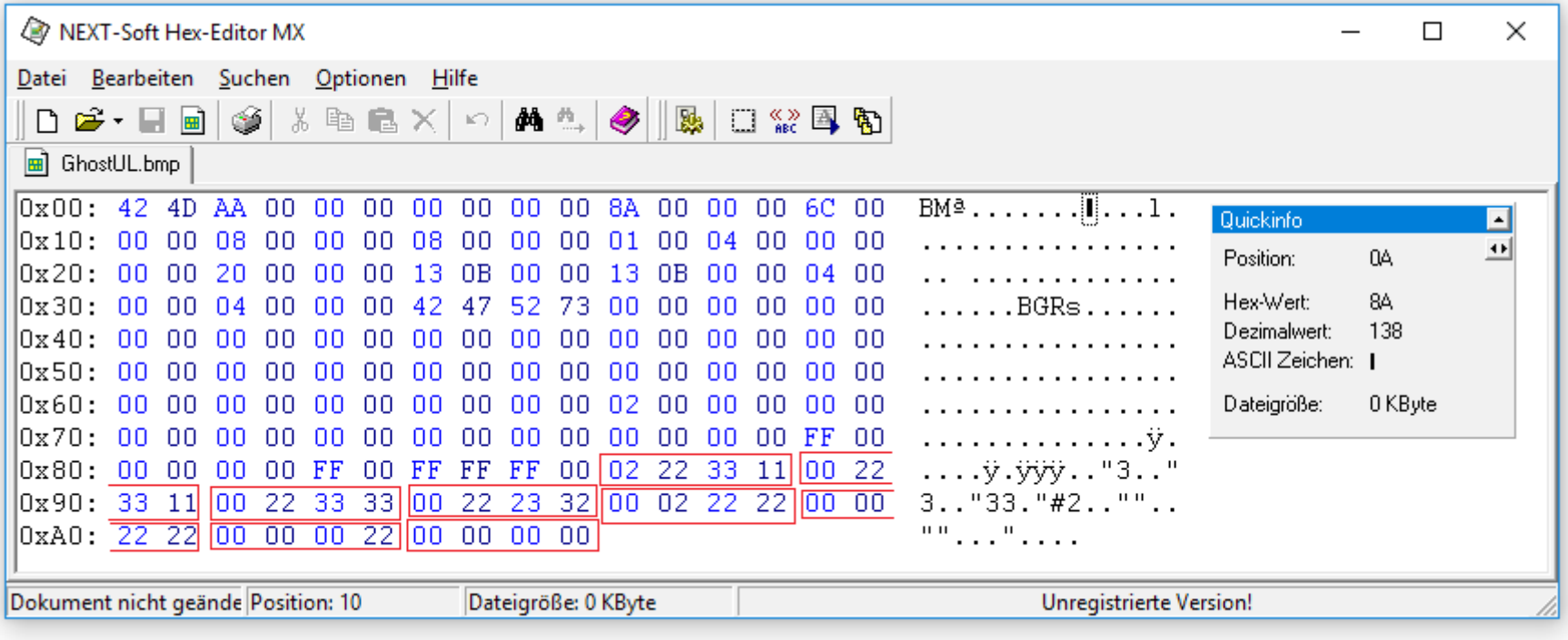
BMP mit Pythonskript einlesen

- Pixel Array bei Byte 138
- Bei 4 Bits Pro Pixel besteht (bei einer 8x8 Grafik) eine Zeile aus 8 Hex-Ziffern



BMP mit Pythonskript einlesen

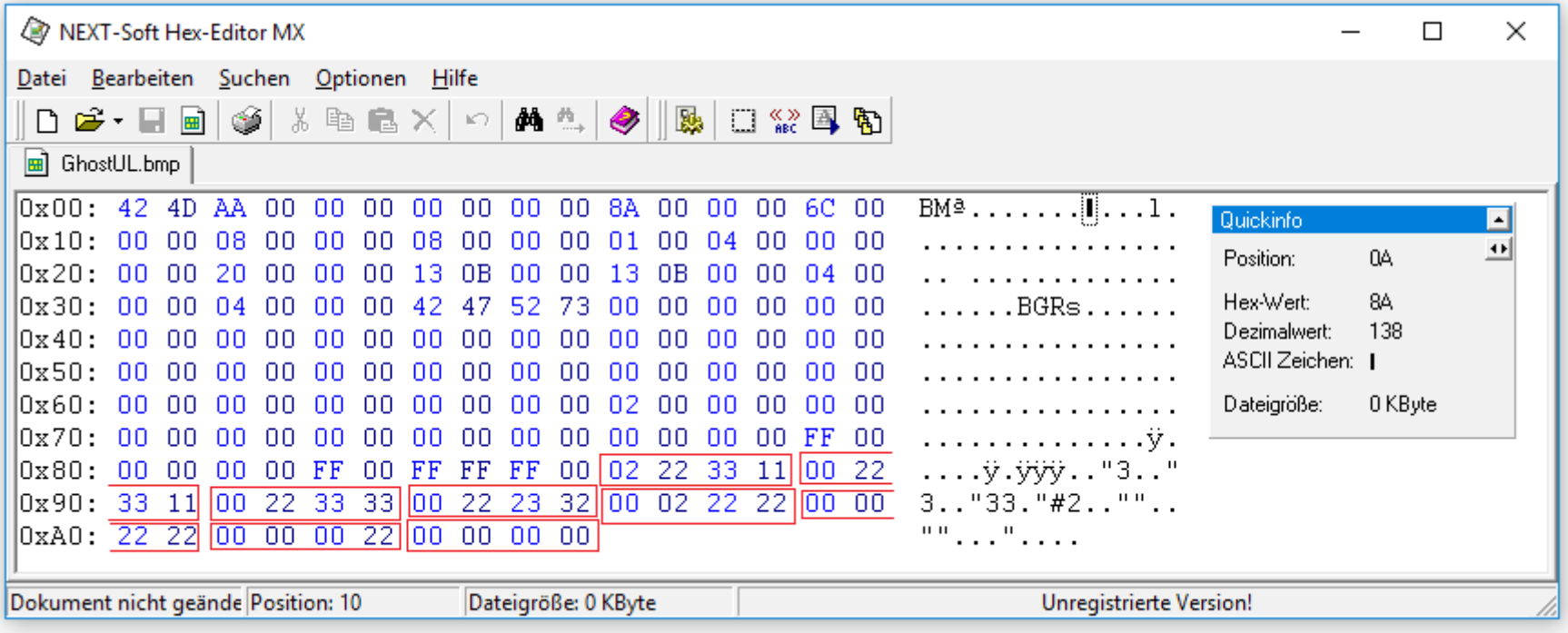
- Achtung! Es wird unten Links mit den Pixeln begonnen.
- Die erste Zeile ist also oben rechts wäre also 00 00 00 00:



BMP mit Pythonskript einlesen

- Das Pixel Array ist also:

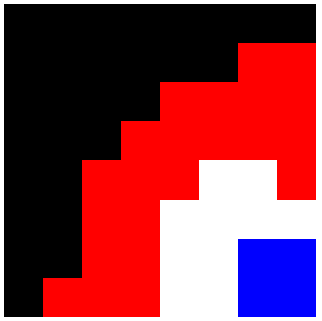
00000000
00000022
00002222
00022222
00222332
00223333
00223311
02223311

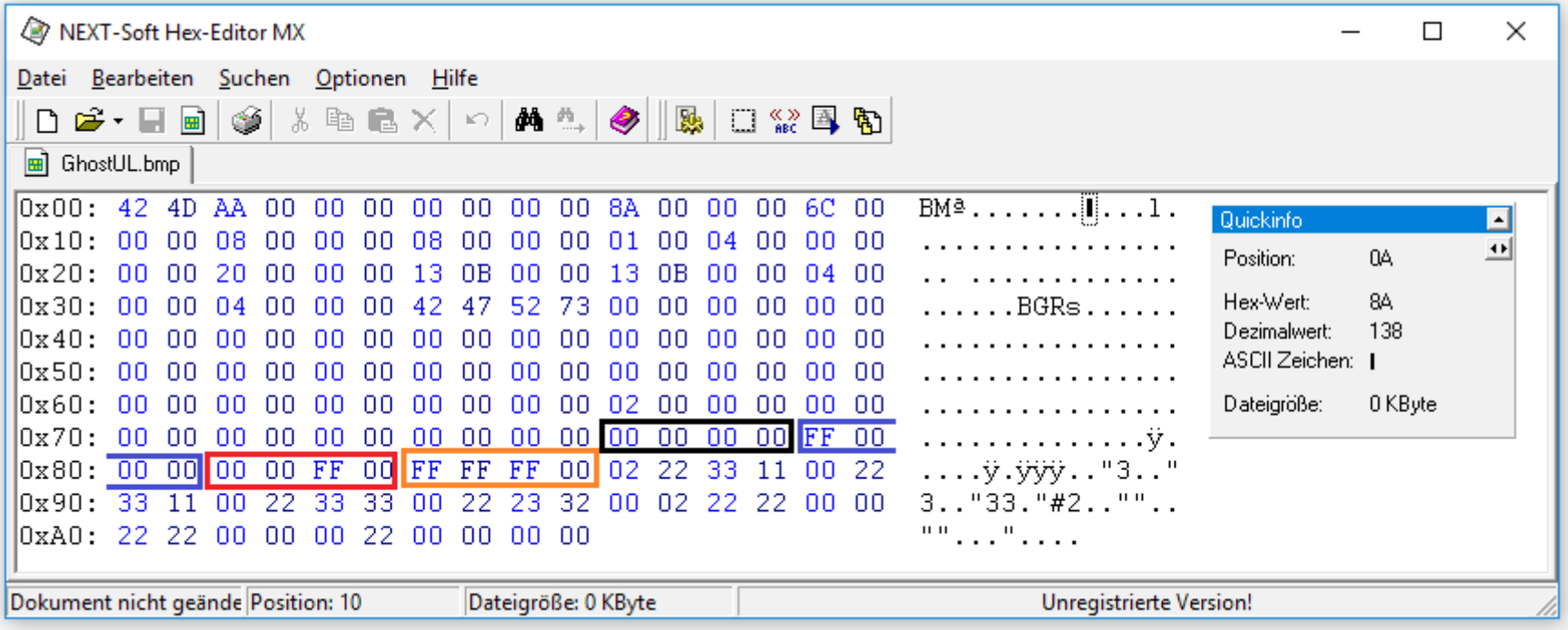


BMP mit Pythonskript einlesen

- Die Zahlen sind die Nummer der Farbe

00000000
00000022
00002222
00022222
00222332
00223333
00223311
02223311

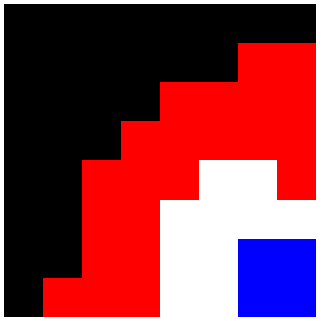




BMP mit Pythonskript einlesen

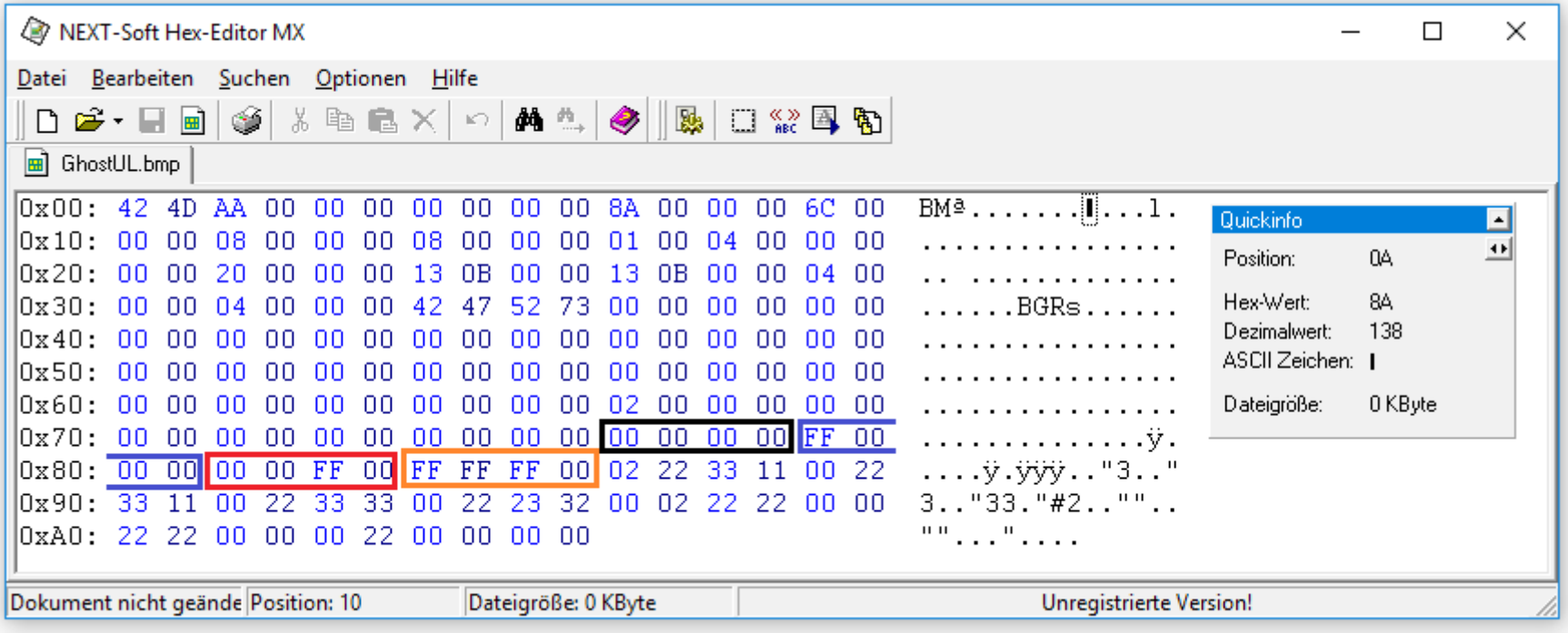
- Die Zahlen sind die Nummer der Farbe

00000000
00000022
00002222
00022222
00222332
00223333
00223311
02223311



Es handelt sich um Blau-, Grün-, Rot-, Alpha- Werte

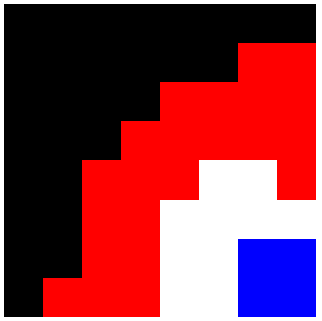
Farbe 0: (00,00,00,00)
Farbe 1: (FF,00,00,00)
Farbe 2: (00,00,FF,00)
Farbe 3: (FF,FF,FF,00)



BMP mit Pythonskript einlesen

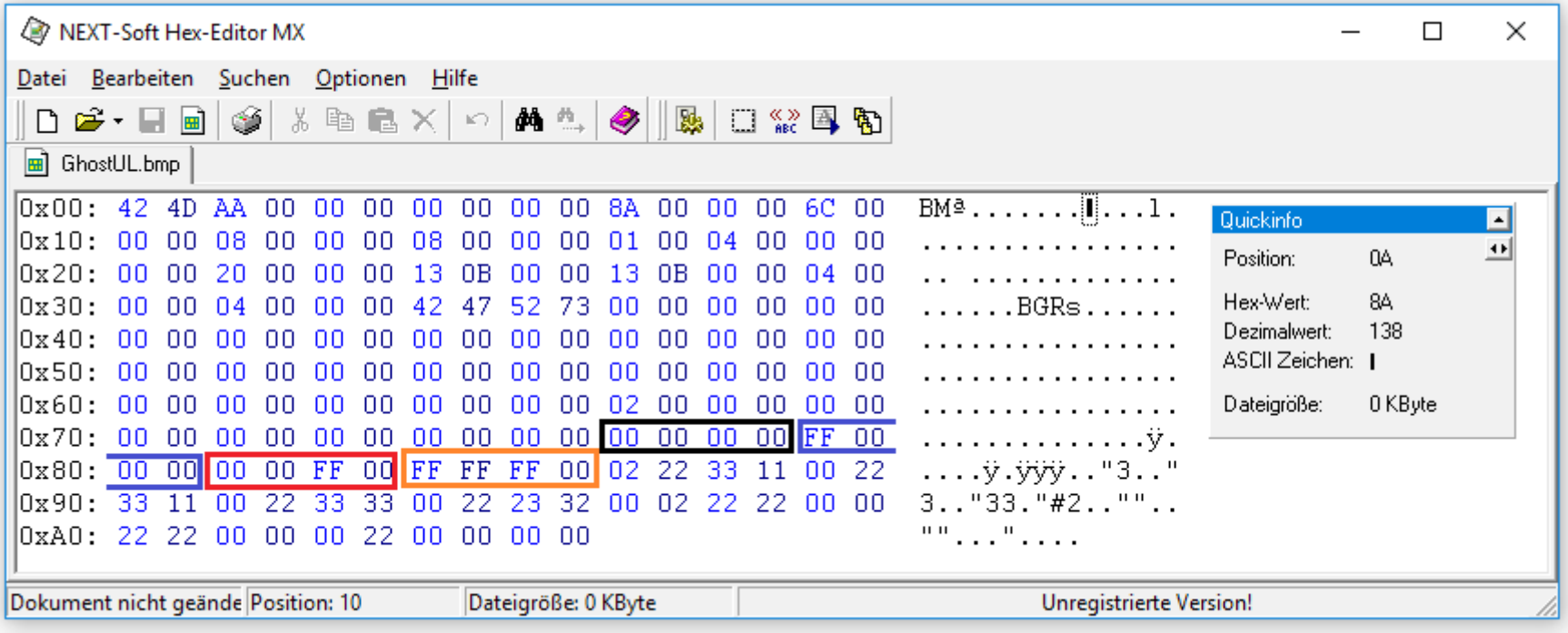
- Die Zahlen sind die Nummer der Farbe

00000000
00000022
00002222
00022222
00222332
00223333
00223311
02223311



Es handelt sich um Blau-, Grün-, Rot-, Alpha- Werte

Farbe 0x00: (00,00,00,00) - **Schwarz**
Farbe 0x01: (255,00,00,00) - **Blau**
Farbe 0x02: (00,00,255,00) - Rot
Farbe 0x03: (255,255,255,00) - Weiß



BMP mit Pythonskript einlesen

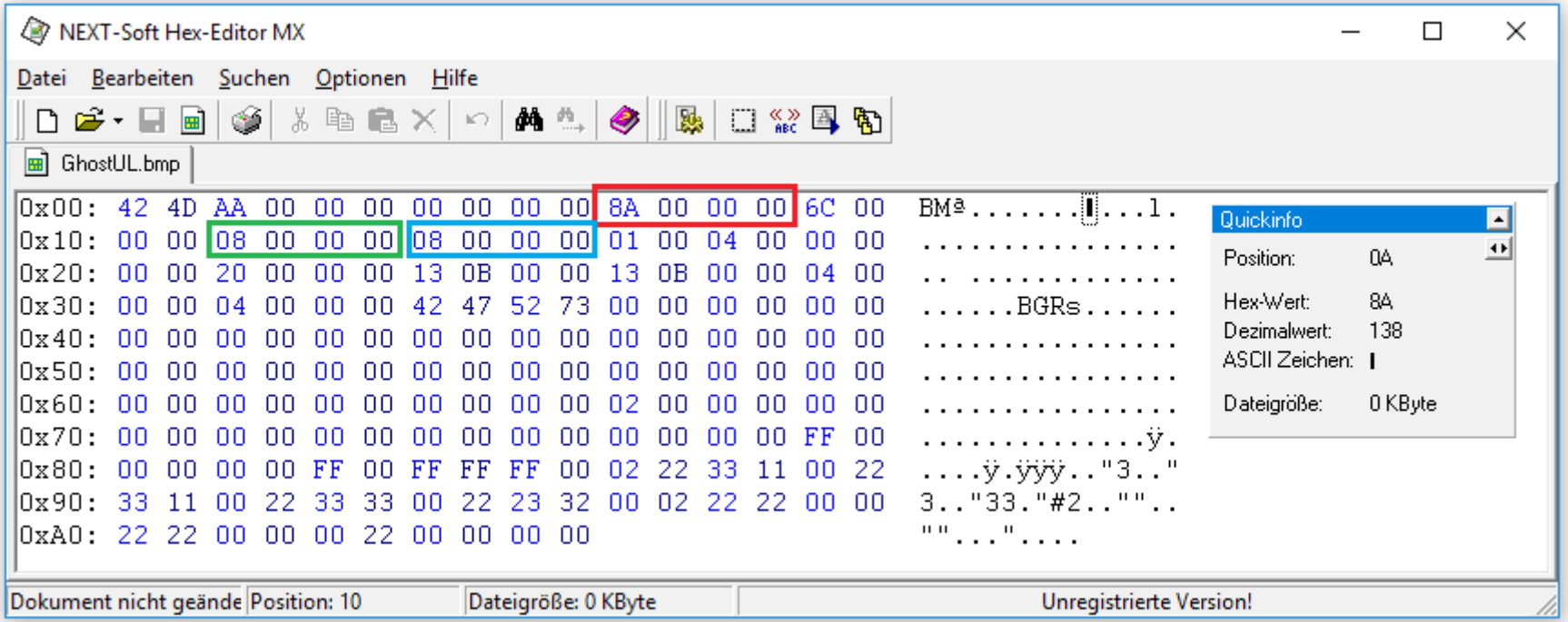
- Mit Python BMP einlesen

```
bytes = []
with open(“GhostUL.bmp“) as file:
    for data in file:
        bytes = data
```

BMP mit Pythonskript einlesen

- Header-Bytes auslesen

```
bytes = []
with open("GhostUL.bmp") as file:
    for data in file:
        bytes = data
pix_array_offset_byte0 = ord(bytes[10]) # 0x0A
pix_array_offset_byte1 = ord(bytes[11]) # 0x0B
...
pixel_array = bytes[pix_array_offset:]
```



BMP zu SNES Kodierung

- Konvertierung der Farben:

  2BPP Bitmap: 4 Byte pro Farbe

  SNES 2BPP: 2 Byte pro Farbe

BMP zu SNES Kodierung

- Lösung: Einfach durch 8 teilen :-)

  Farbe 0:(0, 0, 0, 0): 0 00000 00000 00000

  Farbe 1:(255, 0, 0, 0): 0 11111 00000 00000

  Farbe 2:(0, 0, 255, 0): 0 00000 00000 11111

  Farbe 3:(255, 255, 255, 0): 0 11111 11111 11111

- Unsere Palette in Hex (little Endian):

  (0x00, 0x00), (0x00, 0x7C)

  (0x1F, 0x00), (0xFF, 0x7F)

BMP zu SNES Kodierung

- Pixel Array umwandeln:

00000000
00000022
00002222
00022222
00222332
00223333
00223311
02223311

- Das ist binär:

0000000000000000
0000000000001010
0000000010101010
0000001010101010
0000101010111110
0000101011111111
0000101011110101
0010101011110101

BMP zu SNES Kodierung

- Pixel Array (binär):

0000000000000000
0000000000001010
0000000010101010
0000001010101010
0000101010111110
0000101011111111
0000101011110101
0010101011110101

- Bei 2BPP speichert das SNES zuerst alle low Bits:

0000000000000000
0000000000001010
0000000010101010
0000001010101010
0000101010111110
0000101011111111
0000101011110101
0010101011110101

BMP zu SNES Kodierung

- Pixel Array (binär):

0000000000000000
0000000000001010
0000000010101010
0000001010101010
0000101010111110
00001010111111111
000010101111**0101**
001010101111**0101**

- Bei 2BPP speichert das SNES zuerst alle low Bits:

0000000000000000 = 0000 0000 (0x00)
0000000000001010 = 0000 0000 (0x00)
0000000010101010 = 0000 0000 (0x00)
0000001010101010 = 0000 0000 (0x00)
0000101010111110 = 0000 0110 (0x06)
0000101011111111 = 0000 1111 (0x0F)
0000101011110101 = 0000 1111 (0x0F)
0010101011110101 = 0000 1111 (0x0F)

BMP zu SNES Kodierung

- Pixel Array (binär):

0000000000000000
0000000000001010
0000000010101010
0000001010101010
0000101010111110
00001010111111111
000010101111**0101**
001010101111**0101**

- Bei 2BPP speichert das SNES danach alle high Bits:

0000000000000000 = 0000 0000 (0x00) , 0000 0000 (0x00)
0000000000001010 = 0000 0000 (0x00) , 0000 0011 (0x03)
0000000010101010 = 0000 0000 (0x00) , 0000 1111 (0x0F)
0000010101010101 = 0000 0000 (0x00) , 0001 1111 (0x1F)
0001010101111110 = 0000 0110 (0x06) , 0011 1111 (0x3F)
0001010111111111 = 0000 1111 (0x0F) , 0011 1111 (0x3F)
0001010111101101 = 0000 1111 (0x0F) , 0011 1100 (0x3C)
0101010111101101 = 0000 1111 (0x0F) , 0111 1100 (0x7C)

BMP zu SNES Kodierung

- Pixel Array:

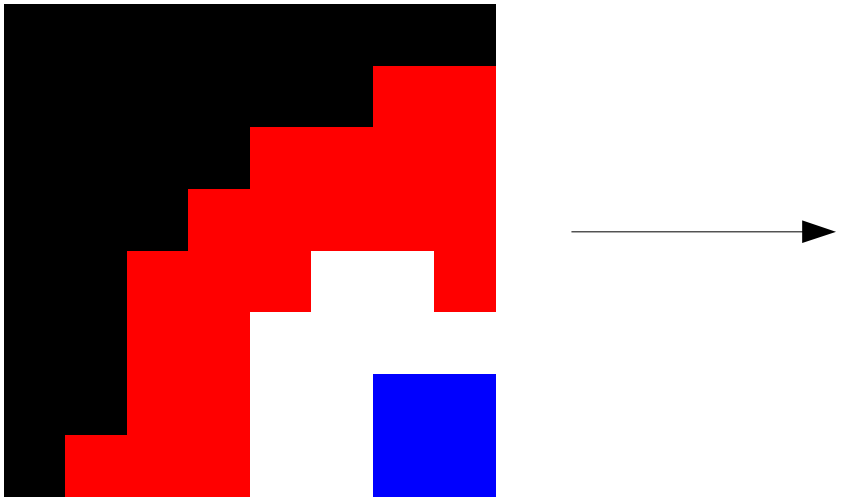
0000000000000000 = 0000 0000 (0x00) , 0000 0000 (0x00)
0000000000001010 = 0000 0000 (0x00) , 0000 0011 (0x03)
0000000010101010 = 0000 0000 (0x00) , 0000 1111 (0x0F)
0000010101010101 = 0000 0000 (0x00) , 0001 1111 (0x1F)
0001010101011110 = 0000 0110 (0x06) , 0011 1111 (0x3F)
0001010101111111 = 0000 1111 (0x0F) , 0011 1111 (0x3F)
0001010111101001 = 0000 1111 (0x0F) , 0011 1100 (0x3C)
0010101011110101 = 0000 1111 (0x0F) , 0111 1100 (0x7C)

- Low und High werden abwechselnd gespeichert. Unser Pixel Array in SNES:

(0x00, 0x00, 0x00, 0x03, 0x00, 0x0F, 0x00, 0x1F,

0x06, 0x3F, 0x0F, 0x3F, 0x0F, 0x3C, 0x0F, 0x7C)

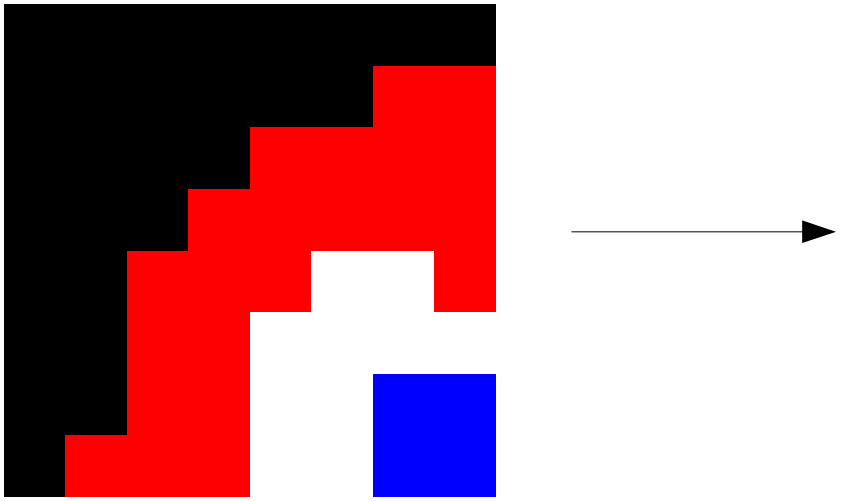
BMP zu SNES Kodierung



(0x00, 0x00, 0x00, 0x03,
0x00, 0x0F, 0x00, 0x1F,
0x06, 0x3F, 0x0F, 0x3F,
0x0F, 0x3C, 0x0F, 0x7C)

(**(0x00, 0x00)**,
(0x00, 0x7C)
(0x1F, 0x00),
(0xFF, 0x7F))

BMP zu SNES Kodierung



(0x00, 0x00, 0x00, 0x03,
0x00, 0x0F, 0x00, 0x1F,
0x06, 0x3F, 0x0F, 0x3F,
0x0F, 0x3C, 0x0F, 0x7C)

(**(0x00, 0x00)**,
(0x00, 0x7C)
(0x1F, 0x00),
(0xFF, 0x7F))

In WLA (z.B. tiles.inc)

```
Data:
    .db $00,$00,$00,$03,$00,$0f,$00,$1f,$06,$3f,$0f,$3f,$0f,$3c,$0f,$7c
Palette:
    .db $00,$00,$00,$7c,$1f,$00,$ff,$7f
```