Die Grafikformate GIF und PNG

von Annette Kaudel

Übersicht

- Geschichte
- GIF
  - Eigenschaften
  - Dateistruktur
  - Beispiele
- PNG
  - Eigenschaften
  - Dateistruktur
  - Beispiele
- Literatur

Geschichte

1977 Der Kompressionsalgorithmus **LZ77** wird von Abrahem Lempel und Jacop Ziv erfunden

1983 (20. Juni) Terry A. Welch (Sperry Corporation - später Unisys) patentiert das **LZW**-Kompressionsverfahren (Variante von LZ78)

1987 (15. Juni) CompuServe veröffentlicht **GIF** als freie und offene Spezifikation (Version **87a**)

1989 **GIF 89a** wird vorgestellt

Geschichte

1993 Unisys informiert CompuServe über die Verwendung ihres patentierten LZW-Algorithmus in GIF

1994 (29. Dez.) Unisys gibt öffentlich bekannt, Gebühren für die Verwendung des LZW-Algorithmus einzufordern

1995 (4. Jan.) die **PNG** Gruppe wird gegründet
(7. März) erste PNG Bilder werden ins Netz gestellt
(8.Dez.) die PNG-Spezifikation 0.92 steht im **W3C**

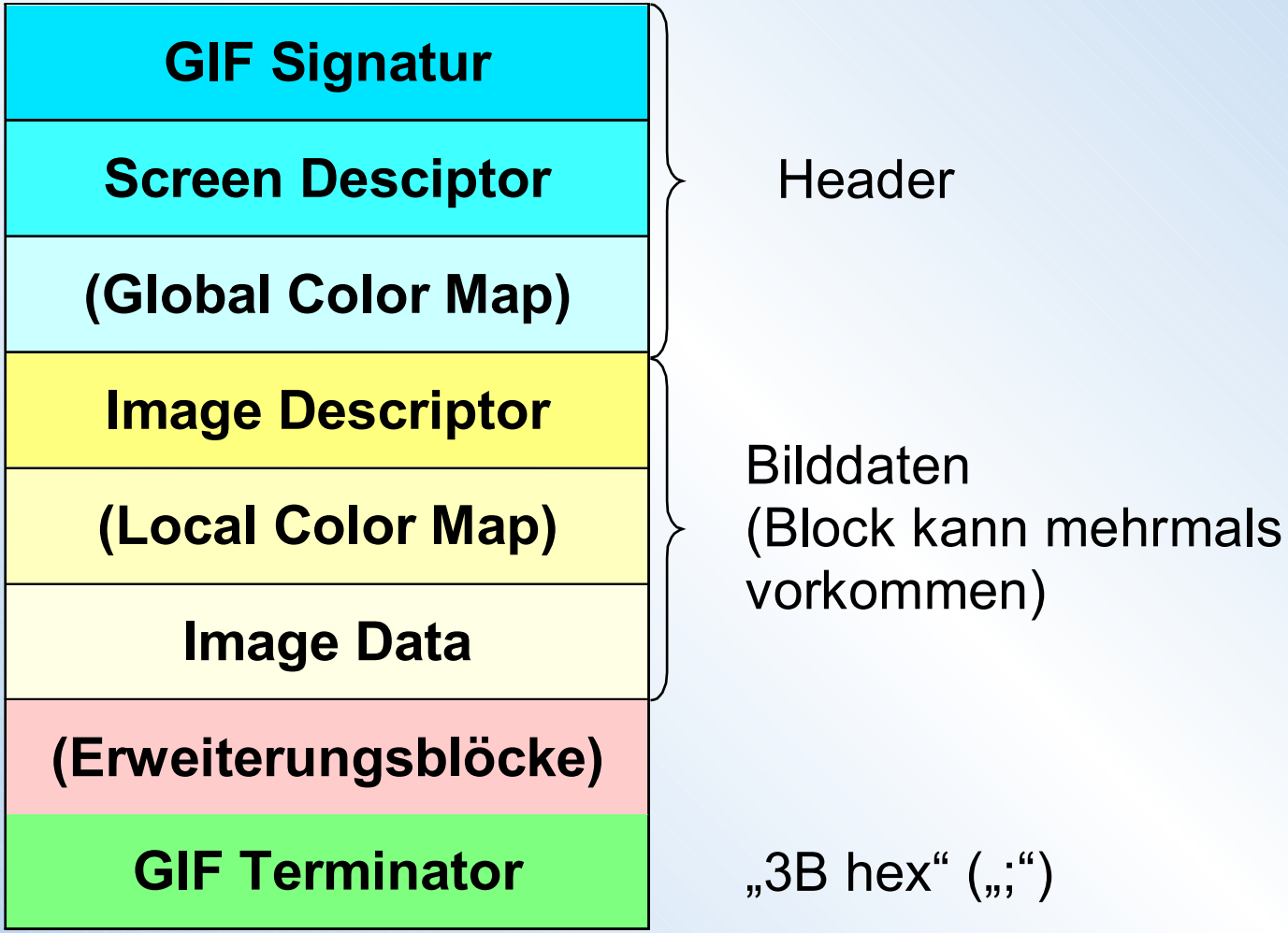
1997 Netscape 4.04 und Internet Explorer 4.0 erscheinen mit PNG Unterstützung

GIF Eigenschaften

Graphics Interchange Format (GIF)

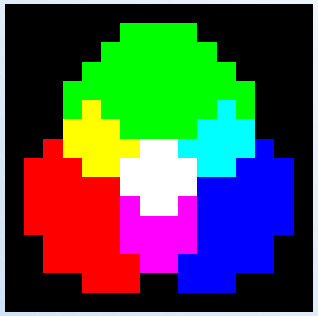
- Raster Grafik
- Farbtiefe 1 bis 8 Bit pro Einzelbild
- eine transparente Farbe
- mehrere Teilbilder in einer Datei (Animationen)
- verlustfreie Komprimierung (LZW)
- interlaced Modus
- hohe Verbreitung

GIF Dateiaufbau



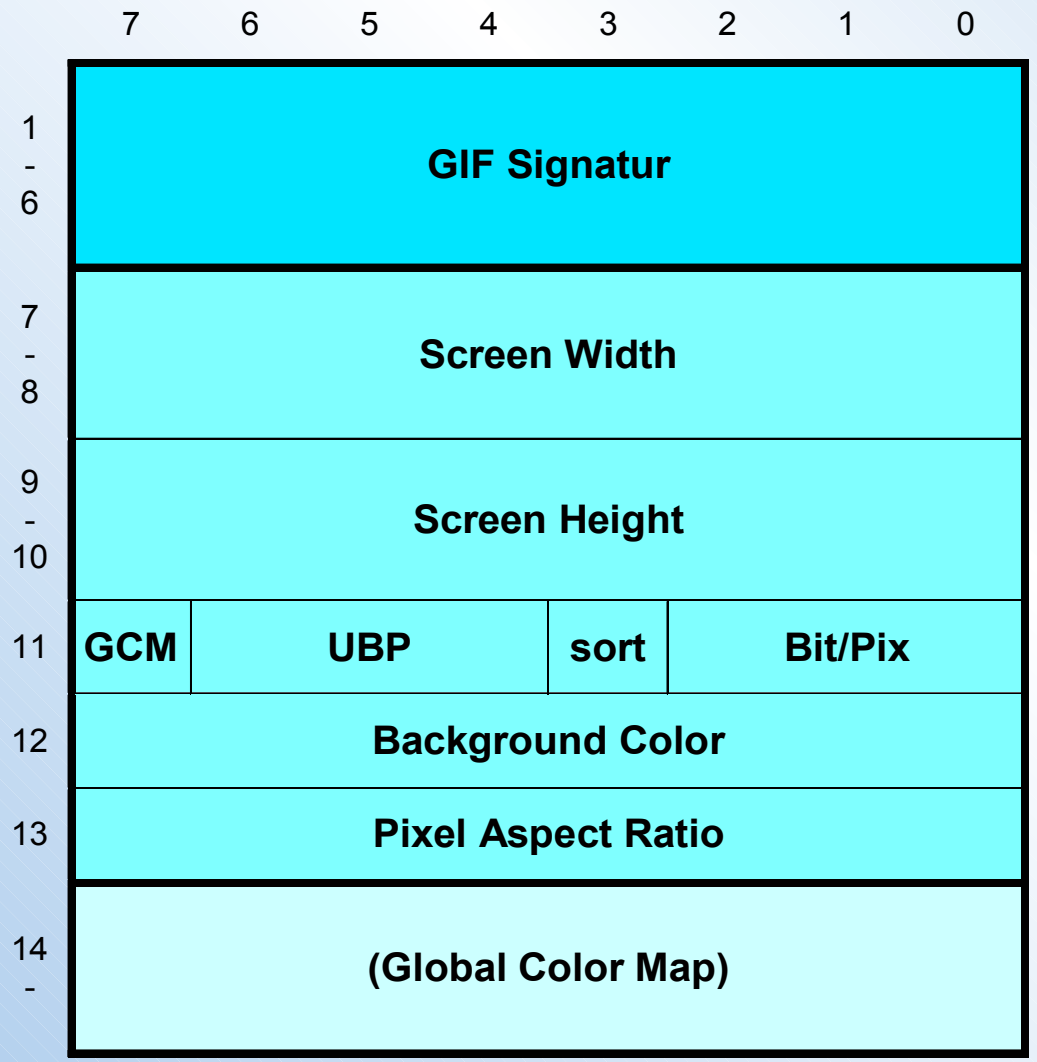
GIF Dateiaufbau

Beispiel:



```
47 49 46 38 39 61 10 00 10 00 A2 00 00
00 00 00 00 00 FF 00 FF 00 00 FF FF FF
00 00 FF 00 FF FF FF 00 FF FF FF 2C 00
00 00 00 10 00 10 00 00 04 46 10 C8 49
AB 05 2A AB 4B B5 E6 98 F7 59 A2 48 2E
A5 B3 55 4B EB 39 70 95 B4 ED F3 C0 30
33 25 7C 6D DF 0E 86 50 C2 2B FE 1E C2
24 B1 98 68 D8 1A C9 21 80 D9 6C 58 A3
BA 25 CF 7A 55 EE 98 5D 2F A5 28 11 83
38 11 00 3B
```

GIF Header



GIF Signature „GIF87a“ oder „GIF89a“

Resolution Flag:

GCM Global Color Map

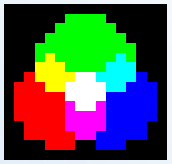
UBP Anzahl der Bits pro Pixel in der Ursprungsgrafik +1

sort Global Color Map absteigend nach der Häufigkeit der Farben sortiert? (ab GIF 89a)

Bit/pix Farbtiefe der Global Color Map + 1

GIF Header

Beispiel:



```
47 49 46
38 39 61
```

= „GIF 89a“

```
10 00
10 00
A2
00
00
```

Gesamt Breite (hier 16)
Gesamt Höhe (hier 16)
= 1 010 0 010
Background Color (hier keine)
Pixel Aspect Ratio (hier 1:1)

1	010	0	010

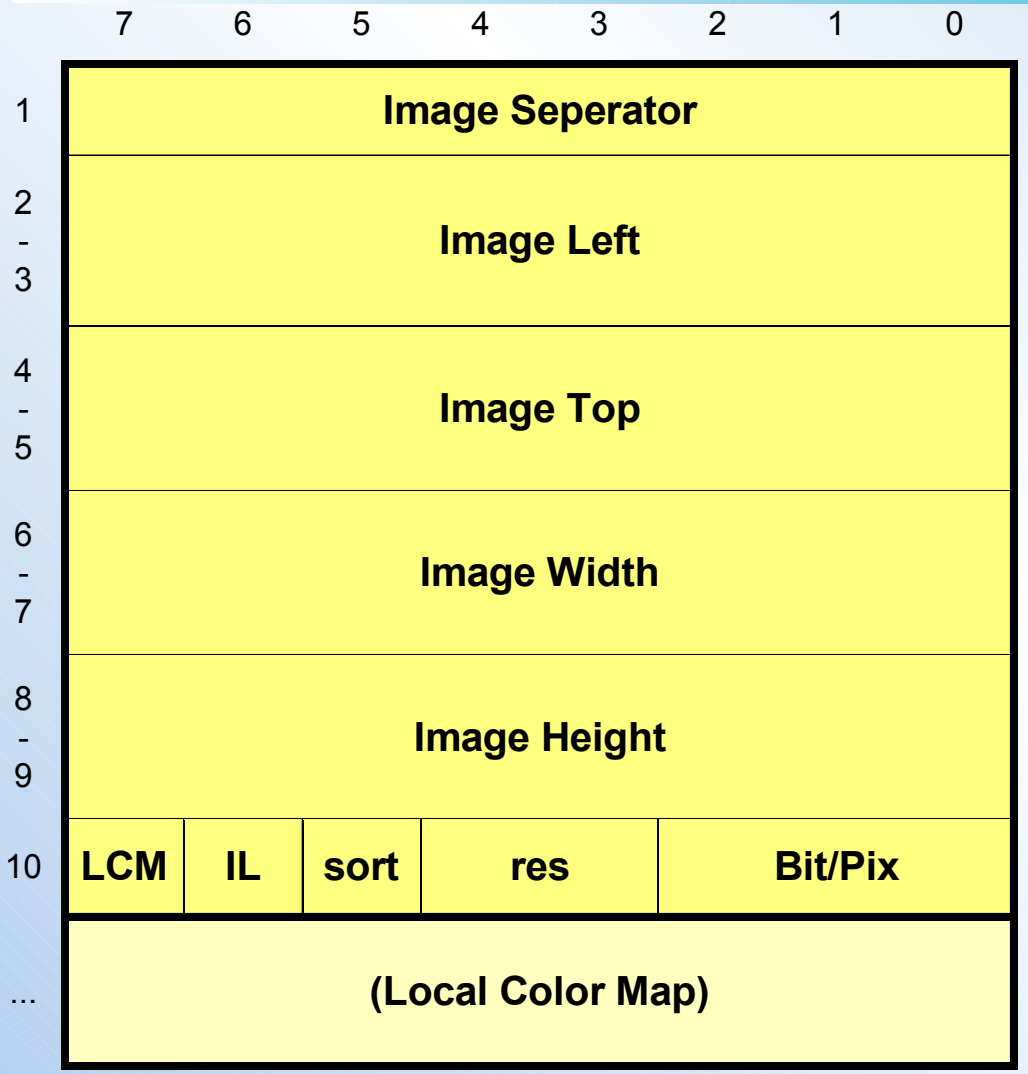
Farbtiefe (hier 3)
Farben sortiert (hier nein)
Farbtiefe der Ursprungsgrafik (hier 3)
globale Farbtabelle (hier ja)

globale Farbtabelle

in RGB Format

```
00 00 00
00 00 FF
00 FF 00
00 FF FF
FF 00 00
FF 00 FF
FF FF 00
FF FF FF
```

GIF Bilddaten

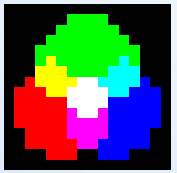


„2C hex“ (“ ,“)

LCM	Local Color Map
IL	Interlaced
sort	Local Color Map absteigend nach der Häufigkeit der Farben sortiert? (ab GIF 89a)
res	Reserviert (Bits müssen 0 sein)
Bit/pix	Farbtiefe der Local Color Map - 1

GIF Bilddaten

Beispiel: (Image Descriptor)



```
2C
00 00
00 00
10 00
10 00
00
```

Image Seperator
Image Left (hier 0)
Image Top (hier 0)
Image Width (hier 16)
Image Height (hier 16)
= **0 0 0** 00 **000**

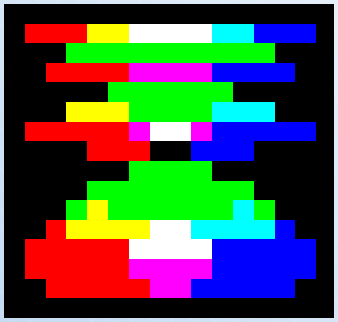
Farbtiefe der LCM
Farben sortiert (hier nein)
Interlaced abgespeichert (hier nein)
lokale Farbtabelle (hier nein)

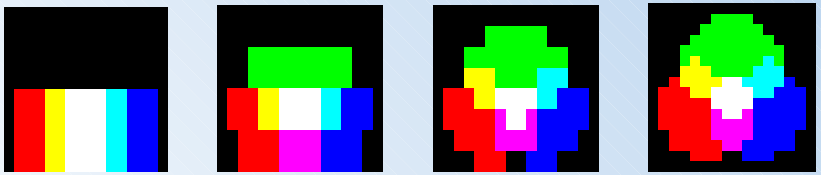
GIF Bilddaten

Interlaced Modus:

(Speicherreihenfolge der Bildzeilen)

- Aufbau zeilenweise
- in 4 Schritten
  1. jede 8. Zeile ab Zeile 0
  2. jede 8. Zeile ab Zeile 4
  3. jede 4. Zeile ab Zeile 2
  4. jede 2. Zeile ab Zeile 1

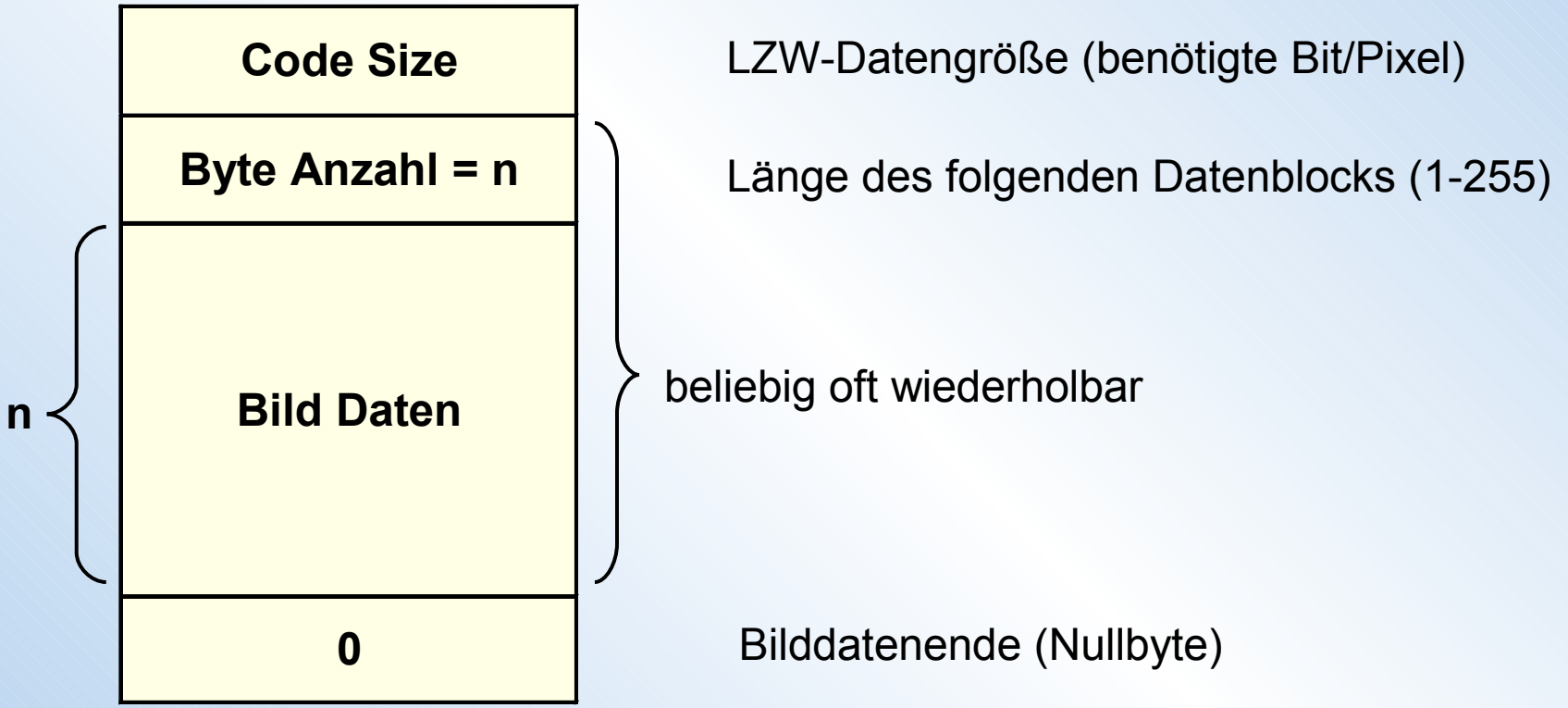




Zeile	Schritt 1	Schritt 2	Schritt 3	Schritt 4
0	1a			
1				4a
2			3a	
3				4b
4		2a		
5				4c
6			3b	
7				4d
8	1b			
9				4e
10			3c	
11				4f
12		2b		
13				4g
14			3d	
15				4h

GIF Bilddaten

Image Data:



GIF Bilddaten (LZW)

Lempel-Ziv Welch-Kompression:

Idee: wiederkehrende Zeichenketten in Codes komprimieren

GIF:

- „Clear Code“ - Codetabelle auf Anfangswert zurücksetzten (2^Code Size)
- „End of Information Code“ (2^Code Size + 1)
- erster freier Code beginnt bei 2^Code Size + 2
- variable Codelänge (3 bis 12 Bits)
  - wenn alle Codes vergeben: Clear Code, neue Codlänge + 1 Bit
- in Bytes gepackt (um Füllbits zu vermeiden)

  Beispiel: (mit 5-Bit-Codes)

	Byte 3	Byte 2	Byte 1
	76543210	76543210	76543210
...	eeeedddd	dcccccbb	bbbaaaaa

LZW Algorithmus

Encode:

```
Codetabelle initialisieren (jedes Zeichen erhält einen Code)
präfix = ″ ″
while (Ende des Eingabedatenstroms noch nicht erreicht) {
        suffix := nächstes Zeichen aus dem Eingabedatenstrom
        muster := präfix + suffix
        if muster in Codetabelle then
                präfix := muster
        else
                muster in Codetabelle eintragen
                LZW-Code von präfix ausgeben
                präfix := suffix
}
if präfix nicht leer then
        LZW-Code von präfix ausgeben
```

LZW Algorithmus

Beispiel: Codetabelle: 0:A 1:B 2:C 3:D

	präfix	muster	suffix	Eingabedatenstrom	LZW Code	Ausgabe
0		A	A	ABCABCABCD		
1	A	AB	B	BCABCABCD	4:AB	0 (A)
2	B	BC	C	CABCABCD	5:BC	1 (B)
3	C	CA	A	ABCABCD	6:CA	2 (C)
4	A	AB	B	BCABCD		
5	AB	ABC	C	CABCD	7:ABC	4 (AB)
6	C	CA	A	ABCD		
7	CA	CAB	B	BCD	8:CAB	6 (CA)
8	B	BC	C	CD		
9	BC	BCD	D	D	9:BCD	5 (BC)
10	D					3 (D)

LZW Algorithmus

Decode:

```
Codetabelle initialisieren (jedes Zeichen erhält einen Code)
präfix = ″ ″
while (Ende der Daten noch nicht erreicht) {
       lese LZW-Code
       muster := dekodiere (LZW-Code)
       gebe muster aus
       neuer LZW-Code := präfix + erstes Zeichen von muster
       präfix := muster
}
```

LZW Algorithmus

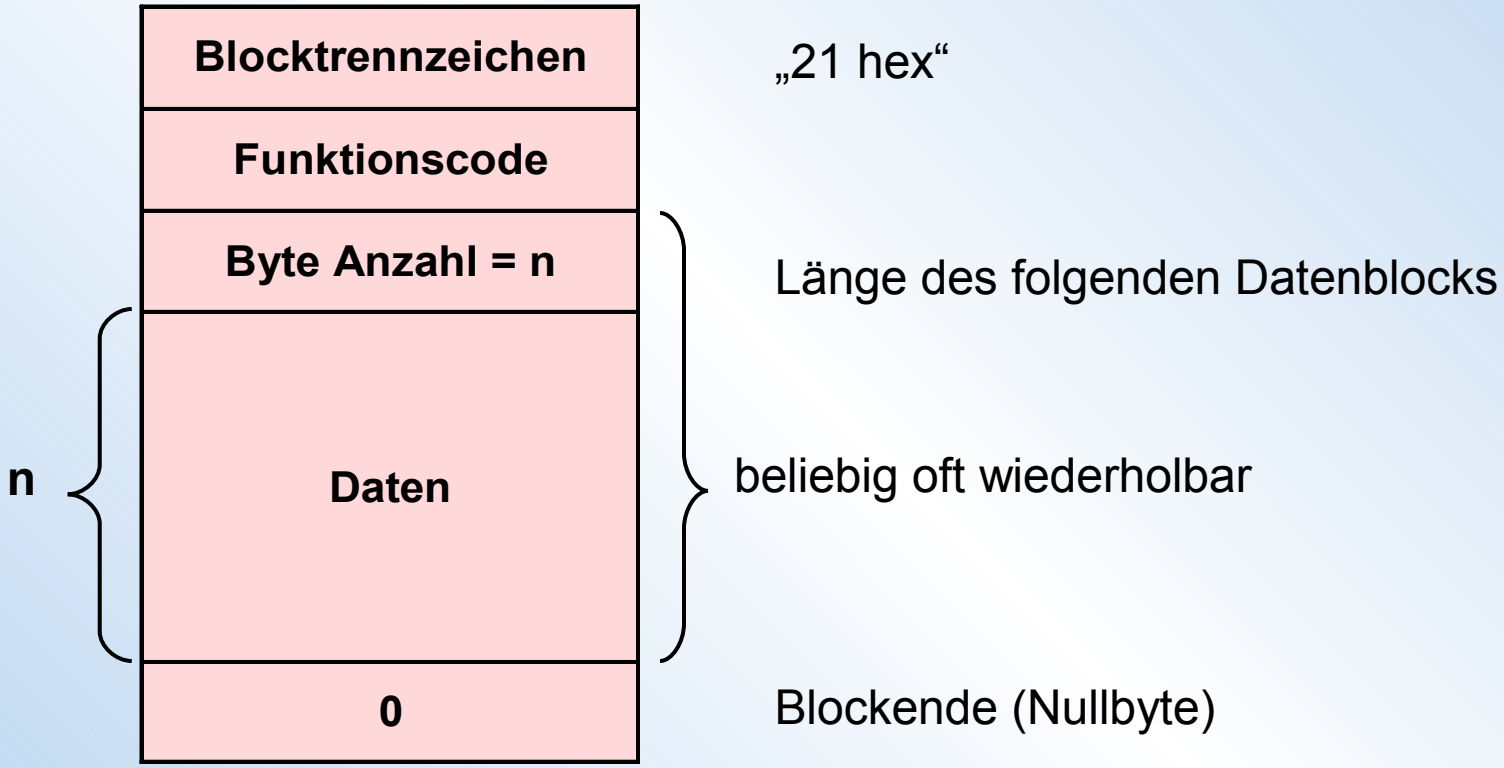
Beispiel: (Dekodierung)

Codetabelle:

0 : A
1 : B
2 : C
3 : D

Code	präfix	muster	neuer Code	Ausgabe
0		A		A
1	A	B	4 = AB	B
2	B	C	5 = BC	C
4	C	AB	6 = CA	AB
6	AB	CA	7 = ABC	CA
5	CA	ABC	8 = CAB	BC
3	ABC	D	9 = BCD	D

GIF Erweiterungsblöcke



Erweiterungsblöcke sind erst ab Version 89a definiert

GIF Erweiterungsblöcke

bisher spezifizierte Erweiterungsblöcke:

- **Plain Text Extension:** (01 hex) — ASCII Test als Grafik auf der Bildfläche darstellen
- **Comment Extension:** (FE hex) — Kommentare (z.B.: Autor ...) als ASCII Text mitspeichern aber nicht im Bild anzeigen
- **Graphic Control Extension:** (F9 hex) — gibt an wie nach der Anzeige des folgenden Bilddatenblocks weiter verfahren werden soll (z.B.: Animationsgeschwindigkeit)
- **Application Extension:** (FF hex) — anwendungsspezifische Daten für eigene Erweiterungsblöcke

PNG Eigenschaften

Portable Network Graphics (PNG)

intern “PiNG is not GIF”

- Raster Grafik
- Farbtiefe: Palette (1 - 8 Bit), Graustufen (16 Bit), Truecolor (48 Bit)
- Alpha-Kanal für stufenlose Tranzparenz (+ 16 Bit)
- verlustfreie Komprimierung (LZ77)
- leicht erweiterbar
- progressive Anzeige (Adam 7)

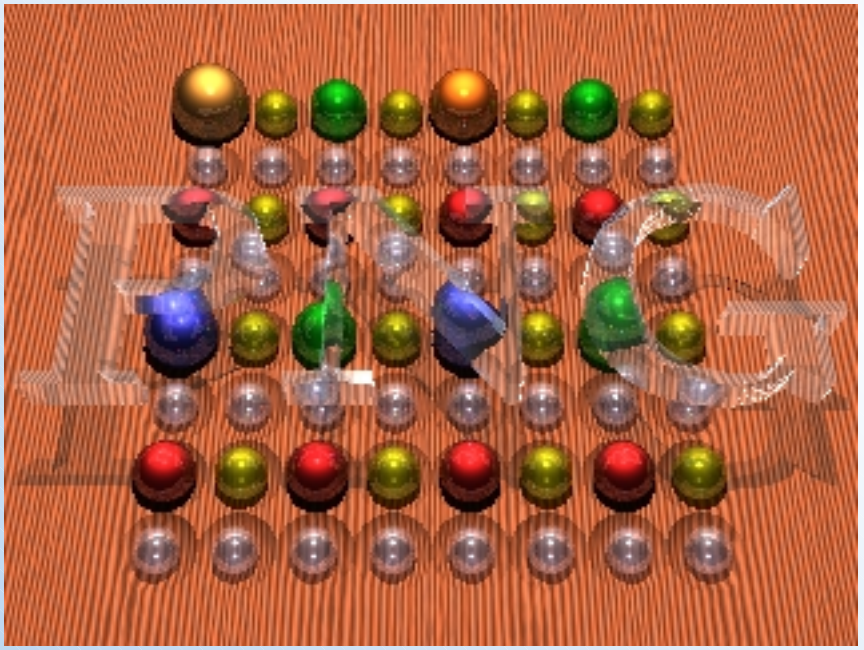
- keine Animationen (⟶ MNG)

PNG Schritt 1

Interlaced Modus:

Adam 7 (progressive Anzeige)

- in 7 Schritten
- Blöcke a 8x8 Pixel



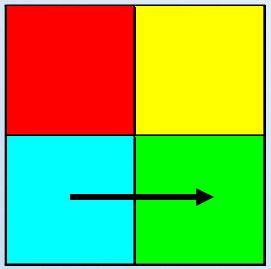
1	6	4	6	2	6	4	6
7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6
7	7	7	7	7	7	7	7
3	6	4	6	3	6	4	6
7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6
7	7	7	7	7	7	7	7

PNG Schritt 2

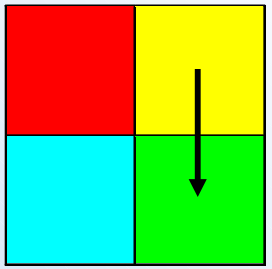
Filter:

- Differenz zu benachbarten Werten speichern statt ganzen Wert
- Daten besser komprimierbar
- Filtermethode kann von Zeile zu Zeile wechseln
  (Filtertypindikator-Byte vor jeder Zeile)

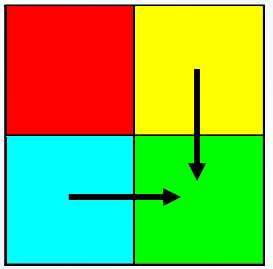
0 = None



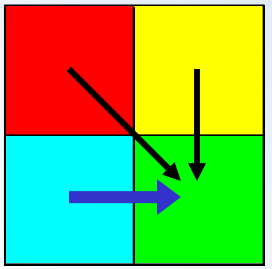
1 = Sub



2 = Up



3 = Average



4 = Paeth

PNG Schritt 3

Kompression:

Deflate-Algorithmus (Variante von LZ77 mit 32 kBit Sliding Window)

zlib Datenstrom

LZ77-Algorithmus:

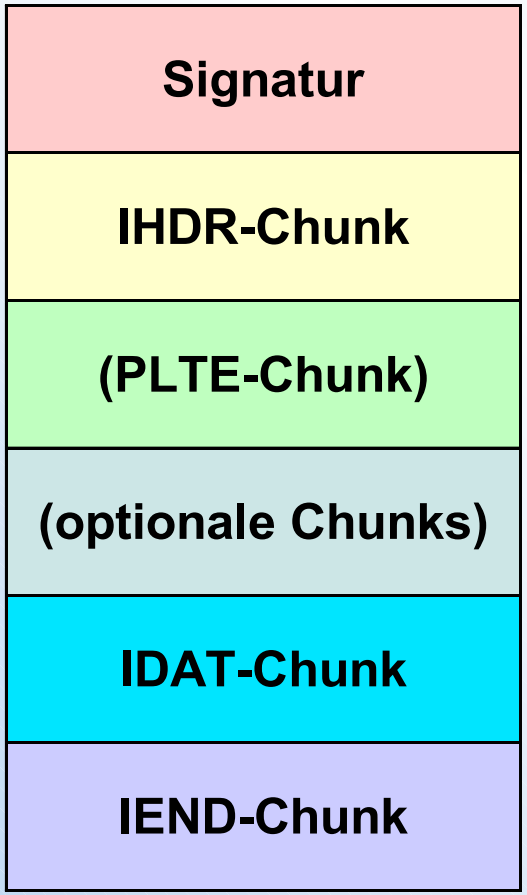
```
Kodierungsposition := Anfang des Eingabedatenstroms
while (Ende des Eingabedatenstroms noch nicht erreicht) {

        finde die längste Übereinstimmung im Datenfenster mit
        der an der Kodierungsposition beginnenden Zeichenkette

        gebe den Zeiger auf die Übereinstimmung Datenfenster und
        das erste nicht passende Zeichen im Vorschaupuffer aus

        Kodierungsposition += Länge der Übereinstimmung + 1
}
```

PNG Schritt 3

Beispiel:

Pos	Sliding Window	Vorschaupuffer	Pos, ÜS	Ausgabe
0		AABCAAABC		(0,0)A
1	A	ABCAAABC	0,A	(0,1)B
3	AAB	CAAABC		(0,0)C
4	AABC	AAABC	0,AA	(0,2)A
7	AABCAAA	BC	3,BC	(2,1)C

PNG Dateiaufbau



critical chunks:

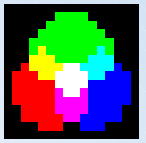
müssen von allen Dekodern und Encodern interpretiert werden

ancillary chunks:

untergeordnete optionale Chunks

PNG Dateiaufbau

Beispiel:



PNG	89 50 4E 47 0D 0A 1A 0A
IHDR	00 00 00 0D 49 48 44 52 00 00 00 10 00 00 00 10 08 03 00 00 00 28 2D 0F 53
tIME	00 00 00 07 74 49 4D 45 07 D2 0C 0A 17 19 3B 0A 0D EF 28
pHYs	00 00 00 09 70 48 59 73 00 00 0B 12 00 00 0B 12 01 D2 DD 7E FC
gAMA	00 00 00 04 67 41 4D 41 00 00 B1 8F 0B FC 61 05
PLTE	00 00 00 18 50 4C 54 45 00 00 00 FF 00 00 00 00 FF FF 00 FF FF FF FF FF FF 00 00 FF FF 00 FF 00 09 B4 2B E0
IDAT	00 00 00 4E 49 44 41 54 78 DA 6D CA 5B 0E 00 21 08 43 D1 82 3C F6 BF 63 51 E2 08 66 FA 77 4F 0A FC CF 63 6F 57 71 EF E2 FE 88 6B A6 1D 50 DD 62 66 D9 14 A0 22 D1 C6 BB 89 56 2F 61 E6 04 A2 05 CC 05 86 C8 E8 10 3B 80 0A C0 BD 7C 87 24 E0 66 DF 04 5A B2 02 38 83 63 D1 5B
IEND	00 00 00 00 49 45 4E 44 AE 42 60 82

PNG Dateiaufbau

Signatur:

89	50	4E	47	0D	0A	1A	0A
\211	P	N	G	\r	\n	\032	\n

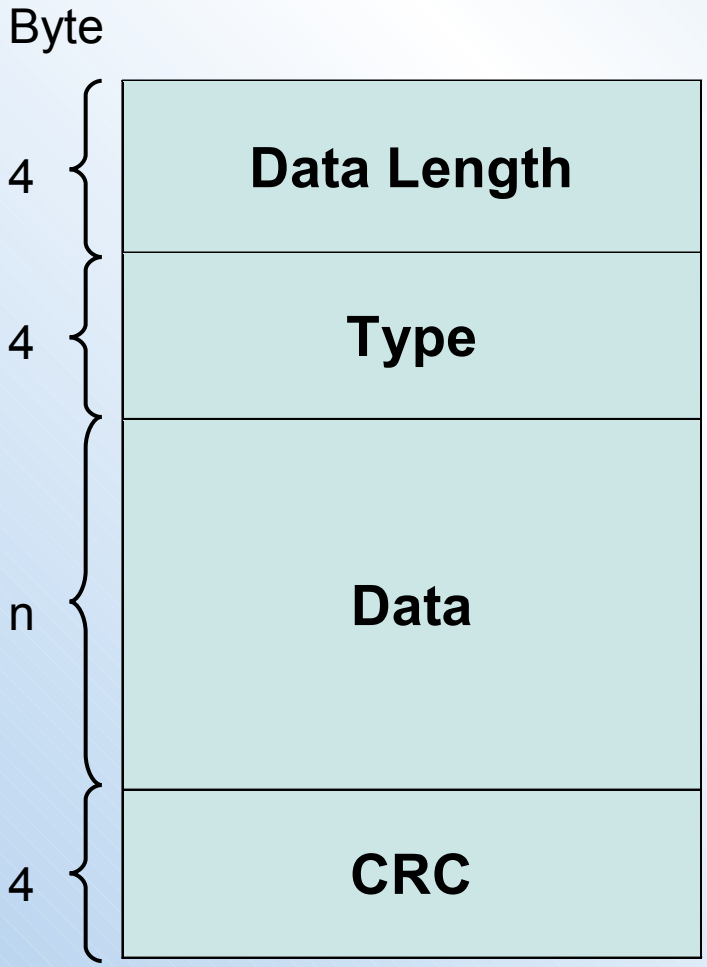
Test auf
8-Bit Transfer

Test ob CR + LF
unverändert
gesendet wird

Ctrl-Z
stoppt
Ausgabe
unter DOS

Test ob
einzelnes LF
unverändert
gesendet
wird

PNG Chunk Layout



Anzahl der Datenbytes (n)

Typkennung (ASCII)

Konvention: (Groß- Kleinbuchstaben)

	1	2	3	4
G	critical	public	reserv.	not copy
k	ancillary	privat	X	copy

Prüfsumme CRC-32 (Type und Data)

Prüfpolynom:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

PNG IHDR Header-Chunk

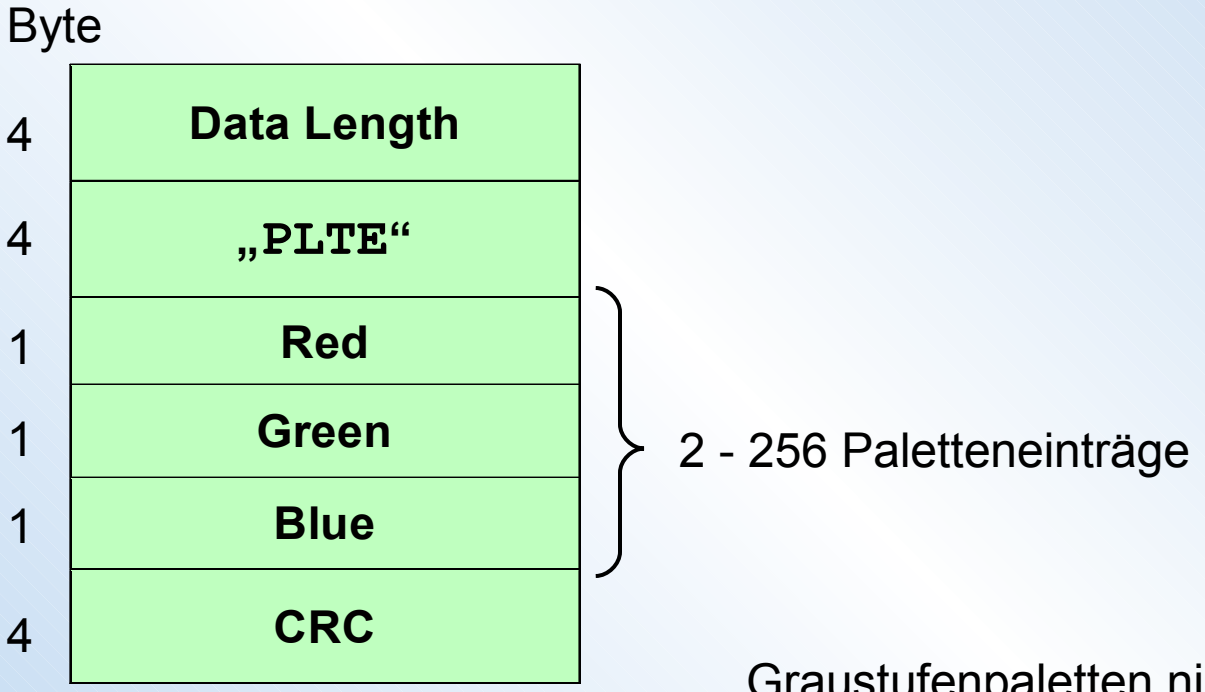
Byte		
4	**Data Length**	= 13 Byte (Länge ohne Data Length, Type, CRC)
4	**„IHDR“**	⇒ Critical Chunk, public
4	**Width**	Breite des Bildes in Pixeln
4	**Height**	Höhe des Bildes in Pixeln
1	**Bit depth**	Länge der Bitmuster (1,2,4,8,16 je nach Farbtyp)
1	**Color Type**	Bit1: Palette, Bit2: Farbbild, Bit3: Alpha-Kanal
1	**Compression method**	0: Deflate Komprimierung
1	**Filter method**	0: adaptive Filterung
1	**Interlace methode**	0: unverschachtelt, 1: Adam7
4	**CRC**	

PNG IHDR Header-Chunk

Beispiel:

`00 00 00 0D`	= 13 Byte (Länge ohne Data Length, Type, CRC)
`49 48 44 52`	„IHDR“
`00 00 00 10`	Breite (16 pixel)
`00 00 00 10`	Höhe (16 pixel)
`08`	Länge der Bitmuster (8)
`03`	`0011` ➔ kein Alpha-Kanal, Farbbild, mit Palette
`00`	0: Deflate Komprimierung
`00`	0: adaptive Filterung
`00`	0: unverschachtelt
`28 2D 0F 53`	CRC

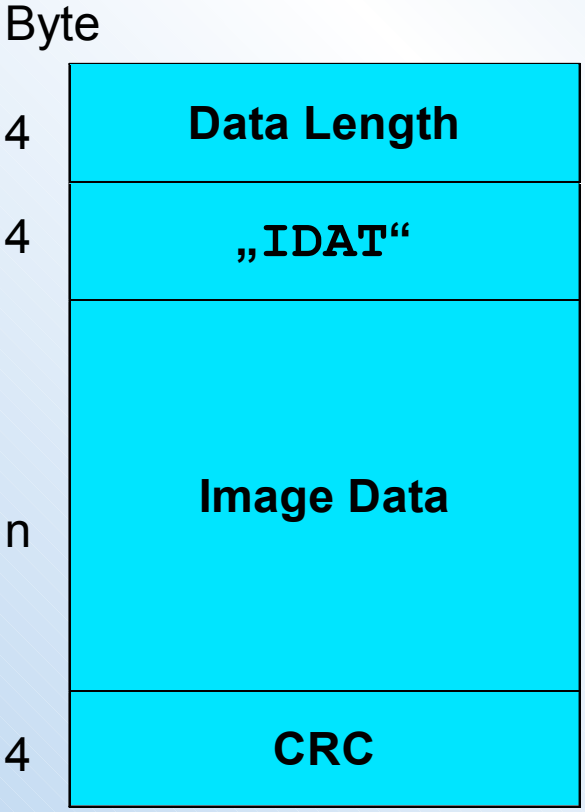
PNG PLTE Palette-Chunk



Graustufenpaletten nicht möglich

Palettenblock kann bei nicht Palettenbildern als Angabe für die Reduktion der Farben dienen (falls nur weniger Farben darstellbar)

PNG IDAT Data-Chunk



Es können auch mehrere IDAT Chunks aufeinander folgen

PNG IEND Trailer-Chunk

Byte	
4	**Data Length**
4	**„IEND“**
4	**CRC**

= 0 Byte

PNG optionale Chunks

tIME	last modification	Datum der letzten Veränderung
bKGD	Background	Hintergrundfarbe (Paletteneintrag, Grau- oder RGB-Wert)
tRNS	Transparency	Alphawerte für einzelne Farben/Paletten
sBit	Significant Bits	Bittiefe vor dem Speichern und nötige Umrechnung
gAMA	Image gama	Heiligkeitswerte beim Erstellen
cHRM	chromaticities	Chromaticity und White Point (RGB Spezifikation)
hIST	histogram	Häufigkeit der Farbwerte (nur bei Palette)
pHYS	physical pixel	Abmessungen der Pixel
sCAL	Physical scale	Größe des abgebildeten Objekts (Karten, Medizin, ...)
tEXt	Textual Data	unkomprimierter Text (Autor, Copyright ...)
zTXt	compr. Text	komprimierter Text (für lange Texte)
gIFg	GIF control Ext.	GIF-Kontrollwerte
gIFx	GIF Aplication E.	GIF-Erweiterungsblöcke

...

PNG Beispiele

Browser Kompatibilität:

MS Internet Explorer 5.5 und 6.0

PNG Beispiele

Browser Kompatibilität:

Netscape Navigator 4.75

PNG Beispiele

Browser Kompatibilität:

Netscape Navigator 6.1 und 7.0, Opera 6.05, Mozilla1.1

Literatur

- GIF Specification (CompuServe Inc.) (http://www.wotsit.org)
- PNG Specification Version 1.0 (http://www.w3.org/TR/REC-png.html)
- Grafikformate (Thomas W. Lipp) 1997
  ISBN: 3-86063-391-0