Version 2023b

BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

Grundlagen der Web-Programmierung

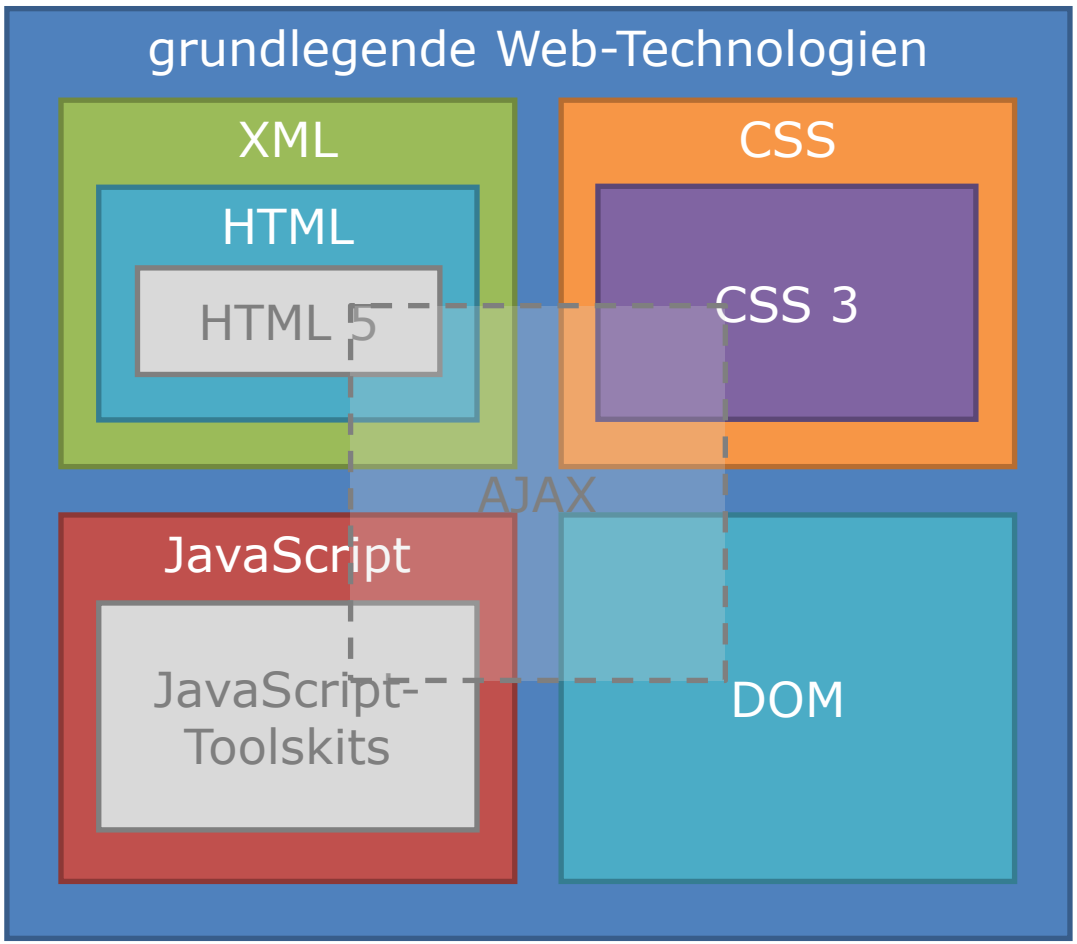
Inhalte von Web-Seiten

Prof. Dr.-Ing. Tenshi Hara
tenshi.hara@ba-sachsen.de

Aufbau der Lehrveranstaltung

Progressive Web Apps

Search Engine Optimization



Proprietäre Standards

Web-Inhalte

- i.d.R. für Menschen gemacht
  - Texte
  - Bilder
  - Videos
  - Tonaufnahmen
  - Kombinationen
- organisiert in definierter Abfolge
  - räumlich nacheinander (z.B. durch Scrollen)
  - zeitlich nacheinander (z.B. durch Klicken oder automatisiert)
- statisch, quasi-statisch oder dynamisch generiert
- im DOM eingebettet oder freistehend

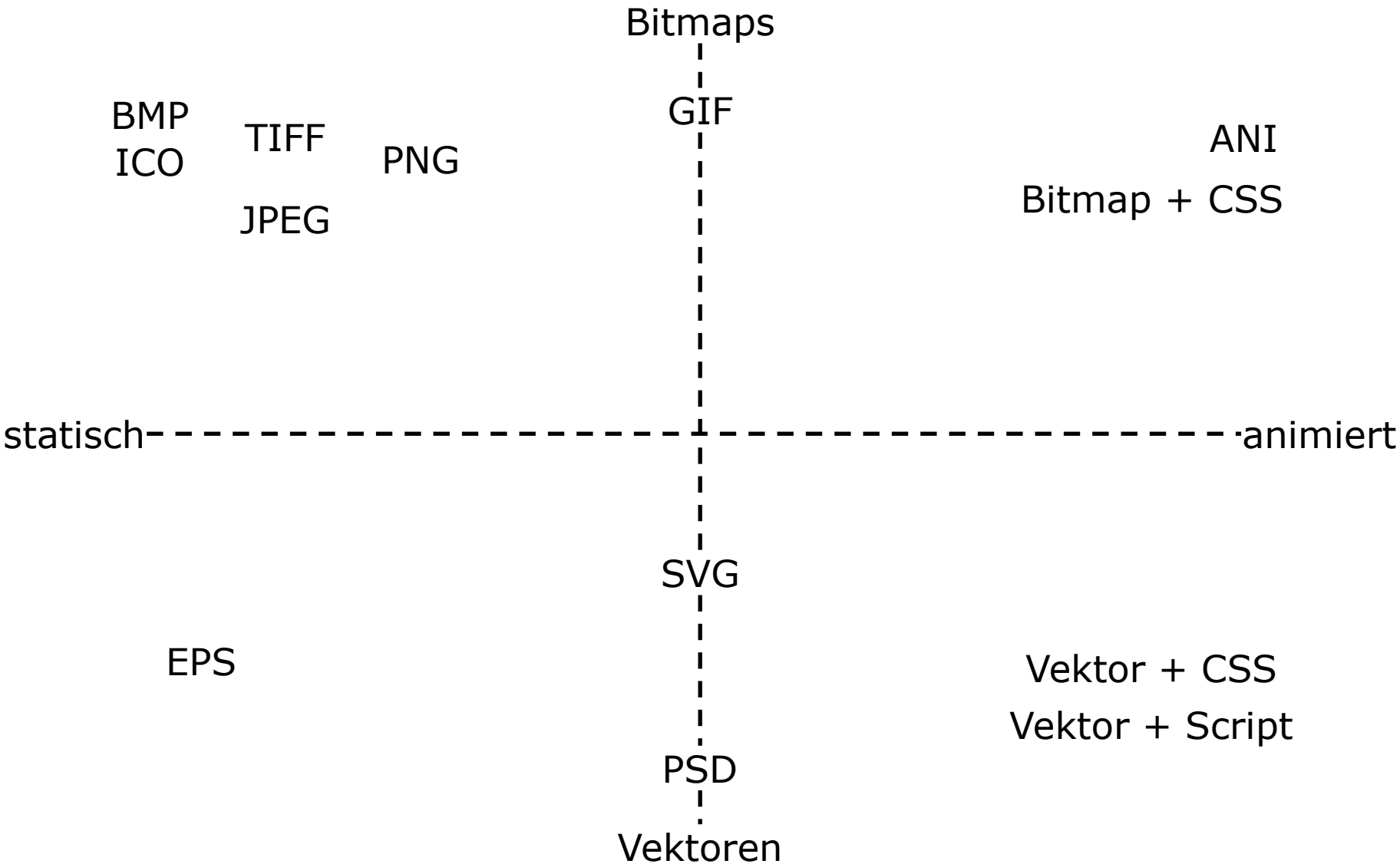
In a Nutshell: Bilder

BILDDARSTELLUNG

- pixeltreu gemäß Bildgröße in Pixel (unabhängig von Auflösung)
- größentreu gemäß Bildmaßen
  - auflösungstreu (Resolution Matching)
  - pixelreduzierend (Oversampling)
  - pixelergänzend (Undersampling)
  - skalierend (Scaling) ⇝ `Scalable Vector Graphics`
- inhaltstreu gemäß umgebender Inhalte
  - absolute Maße
  - relative Maße

⇒ i.d.R. inhaltstreu wegen Einfachheit der Umsetzung (`width="10rem"`)

Bildquellen (1/2)

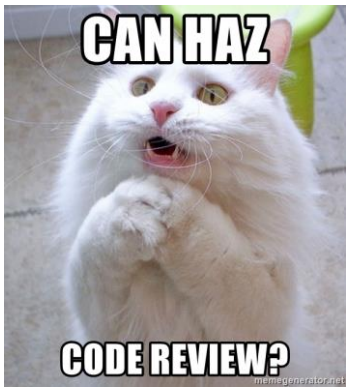


Bildquellen (2/2)

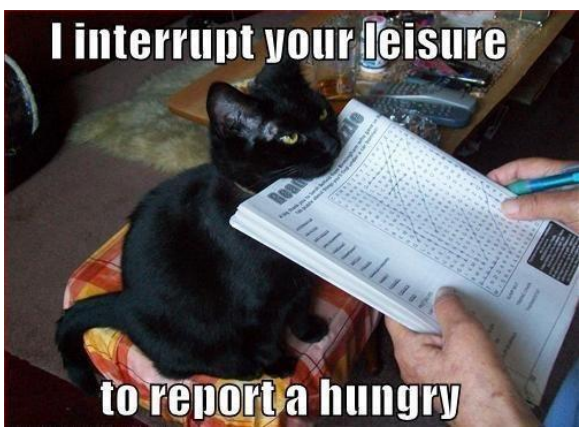
- statische Dateien
  (siehe vorherige Folie)
  - Unterstützung hängt von Fähigkeiten des Browsers ab
  - Server-seitig keine weiteren Vorkehrungen notwendig
- dynamisch generiert
  - auf Basis existierender statischer Dateien
  - von Grund auf generiert
  - Unterstützung hängt von Fähigkeiten des Browsers ab
  - Server muss
    - notwendige MIME-Typen unterstützen
    - Grafikbibliothek nutzen

BEISPIELBILDER

statische Bilder

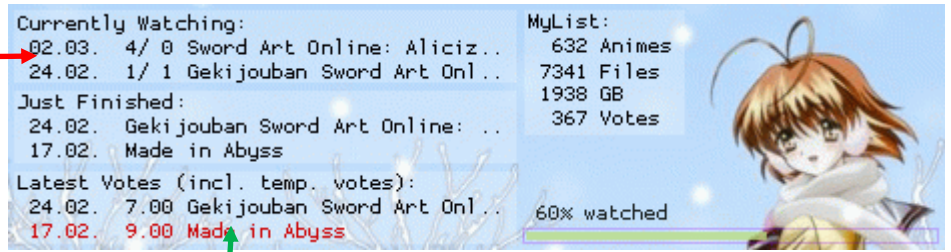


animierte Bilder



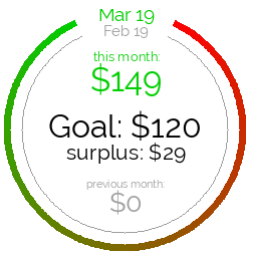
alle Bilder in Public Domain

DYNAMISCH GENERIERTES BILD (XML-BESCHREIBUNG)




```
<signature>
<variables>
    <default element="text" name="face" value="proggytiny" />
    <default element="text" name="size" value="12" />
    <default element="text" name="line-space" value="1" />
    <!-- mylist viewed percentage graph position -->
    <set name="graph_ovp_pos_x" value="455" />
    <set name="graph_ovp_pos_y" value="5" />
    <set name="graph_ovp_height" value="112" />
    <set name="graph_ovp_width" value="4" />
    <set name="graph_ovp_color" value="#0000ff" />
    <set name="graph_ovp_alpha" value="40" />
    <!-- background shading -->
    <set name="textbg_alpha" value="70" />
</variables>

<layout>
    <shape type="rectangle" position="3x3" size="251x36" color="#ffffff" alpha="25"></shape>
    <text position="5x4" align="top-left">
        <line>Currently Watching:</line>
        <line> {{date("d.m.",$watching_id1_viewdate);}} {{padding($watching_id1_state_seenepcount,2)}}/{{padding($watching_id1_eps,2)}} {{truncate($watching_id1_aname,26,"..")}}</line>
        <line> {{date("d.m.",$watching_id2_viewdate);}} {{padding($watching_id2_state_seenepcount,2)}}/{{padding($watching_id2_eps,2)}} {{truncate($watching_id2_aname,26,"..")}}</line>
    </text>
    <shape type="rectangle" position="3x43" size="251x36" color="#ffffff" alpha="25"></shape>
    <text position="5x43" align="top-left">
        <line>Just Finished:</line>
        <line> {{date("d.m.",$finishedwatching_id1_viewdate);}}  {{truncate($finishedwatching_id1_aname,31,"..")}}</line>
        <line> {{date("d.m.",$finishedwatching_id2_viewdate);}}  {{truncate($finishedwatching_id2_aname,31,"..")}}</line>
    </text>
    <shape type="rectangle" position="3x83" size="251x36" color="#ffffff" alpha="25"></shape>
    <text position="5x83" align="top-left">
        <line>Latest Votes (incl. temp. votes):</line>
        <line color="{{if($latestvote_id1_vote>8,"#cc0000","#000000")}}"> {{date("d.m.",$latestvote_id1_date);}} {{padding(precision($latestvote_id1_vote,2),5,"right");}} {{truncate($latestvote_id1_aname, 26, "..")}}</line>
        <line color="{{if($latestvote_id2_vote>8,"#cc0000","#000000")}}"> {{date("d.m.",$latestvote_id2_date);}} {{padding(precision($latestvote_id2_vote,2),5,"right");}} {{truncate($latestvote_id2_aname, 26, "..")}}</line>
    </text>
    <shape type="rectangle" position="258x3" size="80x62" color="#ffffff" alpha="25"></shape>
    <text position="260x4" align="top-left">
        <line>MyList:</line>
        <line>{{padding($stat_mylist_acount, 5, "right");}} Animes</line>
        <line>{{padding($stat_mylist_fcount, 5, "right");}} Files</line>
        <line>{{padding(round($stat_mylist_fsize/1024), 5, "right");}} GB</line>
        <line>{{padding($stat_anidb_votecount, 5, "right");}} Votes</line>
    </text>
    <shape type="rectangle" position="257x111" color="#9000FF" size="208x8" hollow="true" alpha="25"></shape>
    <shape type="rectangle" position="259x113" color="#BBDD77" size="{{round($stat_anidb_ownviewedp * 2)}}x4" alpha="75"></shape>
    <text position="264x107" alpha="85">
        <line>{{padding($stat_anidb_ownviewedp,2,"right");}}% watched</line>
    </text>
</layout>
</signature>
```

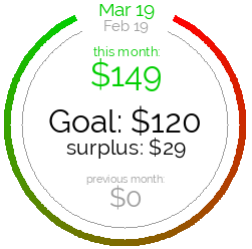
DYNAMISCH IN PHP GENERIERTES BILD (1/2)




```php
<?php
header ("Cache-Control: no-cache, must-revalidate, max_age=7200");
header("Content-type: image/png");

$theFile = realpath(".")."/../tmp/badges/funds.png";

$fromcache = file_exists($theFile) && (filemtime($theFile) < (time()-7200)) && !$debug;

if (!$fromcache) $img = imagecreatetruecolor(241, 241);
else {
    $img = @imagecreatefrompng($theFile);
    if(!$img) $img = imagecreatetruecolor(121, 121);
}
$white  = imagecolorallocate($img, 255, 255, 255);
$black  = imagecolorallocate($img, 0, 0, 0);

$noerrors = true;
if (!$fromcache) {
    $img    = imagecreatetruecolor(241, 241);
    putenv('GDFONTPATH='.realpath("."));
    $font["raleway"] = "badges/raleway";
    $curColor   = array();
    for ($i=0; $i<310; $i++)
        $curColor[$i]   = imagecolorallocate($img, 255-round((255/310)*$i), round((207/310)*$i), 0);
    $prevColor      = imagecolorallocate($img, 159, 159, 159);
    $curColor[310]  = imagecolorallocate($img, 0, 207, 0);
    imagefill($img, 0, 0, $white);

    include_once(getcwd()."/../~/db_data.php");
    $previous_date = gmdate("M y",strtotime("-1 month"));
    $current_date = gmdate("M y",time());
    $previous_month = 0;
    $current_month = 0;
    $previous_month_query = "SELECT SUM(`amount`) FROM `".$_DB["prefix"]."_donations` WHERE MONTH(`date`) = MONTH(CURRENT_DATE() - INTERVAL 1 MONTH) AND
                             YEAR(`date`) = YEAR(CURRENT_DATE() - INTERVAL 1 MONTH);";
    $current_month_query = "SELECT SUM(`amount`) FROM `".$_DB["prefix"]."_donations` WHERE MONTH(`date`) = MONTH(CURRENT_DATE()) AND YEAR(`date`) = YEAR(CURRENT_DATE());";

    $query_result = db($previous_month_query);
    if ($query_result === false)
        $noerrors = false;
    else {
        $previous_month = $query_result[0]["SUM(`amount`)"];
        $previous_month = ($previous_month != "") ? floor($previous_month) : 0;
    }

    $query_result = db($current_month_query);
    if ($query_result === false)
        $noerrors = false;
    else {
        $current_month = $query_result[0]["SUM(`amount`)"];
        $current_month = ($current_month != "") ? floor($current_month) : 0;
    }
```

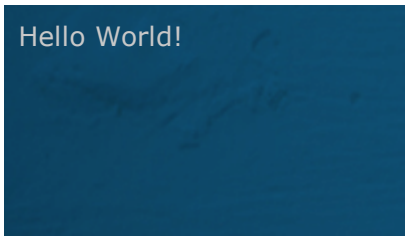
DYNAMISCH IN PHP GENERIERTES BILD (2/2)

```php
    if ($noerrors) {
        $required = 120;
        $previous_quota = $previous_month/$required;
        $previous_quota = ($previous_quota > 1) ? 1 : $previous_quota;
        $previous_total = floor(310*$previous_quota);
        $current_quota = $current_month/$required;
        $current_quota = ($current_quota > 1) ? 1 : $current_quota;
        $current_total = floor(310*$current_quota);

        imagearc($img, 120, 120, 236, 236, -65, 245, $curColor[$current_total]);
        $curMonthText = imagettfbbox(14, 0, $font["raleway"], $current_date);
        imagefttext($img, 14, 0, (122-floor($curMonthText[4]/2)), (5-floor($curMonthText[5]/2)), $curColor[$current_total], $font["raleway"], $current_date);

        imagearc($img, 120, 120, 206, 206, -65, 245, $prevColor);
        $preMonthText = imagettfbbox(11, 0, $font["raleway"], $previous_date);
        imagefttext($img, 11, 0, (122-floor($preMonthText[4]/2)), (21-floor($preMonthText[5]/2)), $prevColor, $font["raleway"], $previous_date);

        for ($i=0; $i<12; $i++) {
            for ($j=0; $j<311; $j++) {
                if ($current_total > $j) {
                    imagearc($img, 120, 120, (230+$i), (230+$i), (-65+$j), (-64+$j), $curColor[$j]);
                    imagearc($img, 120, 120, (231+$i), (230+$i), (-65+$j), (-64+$j), $curColor[$j]);
                    imagearc($img, 120, 120, (230+$i), (231+$i), (-65+$j), (-64+$j), $curColor[$j]);
                }
            }
            if ($previous_total > 0) {
                imagearc($img, 120, 120, (202+$i), (202+$i), (-65), (-64+$previous_total), $prevColor);
                imagearc($img, 120, 120, (203+$i), (202+$i), (-65), (-64+$previous_total), $prevColor);
                imagearc($img, 120, 120, (202+$i), (203+$i), (-65), (-64+$previous_total), $prevColor);
            }
        }

        imagefttext($img, 10, 0, 89, 52, $curColor[$current_total], $font["raleway"], "this month:");
        $curMonthAmount = imagettfbbox(25, 0, $font["raleway"], "\$".$current_month);
        imagefttext($img, 25, 0, (122-floor($curMonthAmount[4]/2)), (68-floor($curMonthAmount[5]/2)), $curColor[$current_total], $font["raleway"], "\$".$current_month);

        imagefttext($img, 24, 0, 44, 128, $black, $font["raleway"], "Goal: \$120");
        if (120-$current_month > 0) {
            $missingAmount = imagettfbbox(16, 0, $font["raleway"], "missing: \$".(120-$current_month));
            imagefttext($img, 16, 0, (122-floor($missingAmount[4]/2)), (140-floor($missingAmount[5]/2)), $black, $font["raleway"], "missing: \$".(120-$current_month));
        }
        else {
            $missingAmount = imagettfbbox(16, 0, $font["raleway"], "surplus: \$".($current_month-120));
            imagefttext($img, 16, 0, (122-floor($missingAmount[4]/2)), (140-floor($missingAmount[5]/2)), $black, $font["raleway"], "surplus: \$".($current_month-120));
        }

        imagefttext($img, 8, 0, 82, 178, $prevColor, $font["raleway"], "previous month:");
        $prevMonthAmount = imagettfbbox(20, 0, $font["raleway"], "\$".$previous_month);
        imagefttext($img, 20, 0, (122-floor($prevMonthAmount[4]/2)), (190-floor($prevMonthAmount[5]/2)), $prevColor, $font["raleway"], "\$".$previous_month);
    }
    imagepng($img, $theFile);
}

imagecolortransparent($img, $white);

imagepng($img);
imagedestroy($img);
?>
```



DYNAMISCH IN PHP GENERIERTE ANIMATION

```php
<?php
header('Content-type:image/gif');
include('GIFEncoder.class.php'); // aGIF-Bibliothek

$text = "Hello World!";

$image = imagecreatefrompng('blau.png');
$text_color = imagecolorallocate($image, 200, 200, 200);
imagestring($image, 5, 5, 5, $text, $text_color);

ob_start();
imagegif($image);
$frames[]=ob_get_contents();
$framed[]=1000; // Animation verzögern
ob_end_clean();

$image = imagecreatefrompng('gruen.png');
$text_color = imagecolorallocate($image, 200, 200, 200);
imagestring($image, 5, 5, 5, $text, $text_color);

ob_start();
imagegif($image);
$frames[]=ob_get_contents();
$framed[]=1000; // Animation verzögern
ob_end_clean();

$gif = new GIFEncoder($frames,$framed,0,2,0,0,0,'bin');
echo $gif->GetAnimation();
?>
```



Einbinden von Bildern – img-Tag

```
<img src="relativer/oder/absoluter/pfad/beispiel.png" />
```

neben HTML-Standardattributen (`class`, `style`, ...) weitere Attribute möglich:

`align`	Ausrichtung (`top`, `bottom`, `middle`, `left`, `right`; nicht in HTML5)
`alt`	Alternativtext für Screen-Reader
`border`	Rahmendicke in Pixel (nicht in HTML5)
`crossorigin`	Einbettung externer Bilder (`anonymous`, `use-credentials`; ab HTML5)
`height`	Höhe in Pixel
`hspace`	horizontaler Elementabstand in Pixel (nicht in HTML5)
`ismap`	gibt an ob Bild server-seitiges Image-Map ist (`ismap`)
`longdesc`	URL mit detaillierter Bildbeschreibung
`sizes`	Angabe von Bildgrößen für unterschiedliche Seiten-Layouts
`src`	URL des anzuzeigenden Bildes
`srcset`	URLs des Bildes in unterschiedlichen Situationen
`usemap`	Angabe einer client-seitigen Image-Map (`#mapname`)
`vspace`	vertikaler Elementabstand (nicht in HTML5)
`width`	Breite in Pixel

IMAGE-MAP (1/2)

- früher sehr beliebt , um grafische Menüs zu erzeugen
- Jetzt bitte nicht mehr benutzen!
  → gefährdet die Barrierefreiheit
- Bild und Koordinaten müssen auf Pixeln basieren
  → skaliert schlecht
- nur drei Standardformen: Kreis, Rechteck, Polygon

```
<img src="/kari.png" border="0" usemap="#kari" />
<map name="kari" id="kari">
   <area shape="circle" coords="45,45,7" href="/eyes/left" />
   <area shape="circle" coords="65,46,7" href="/eyes/right" />
   <area shape="rect" coords="44,75,68,82" href="/mouth" />
</map>
```

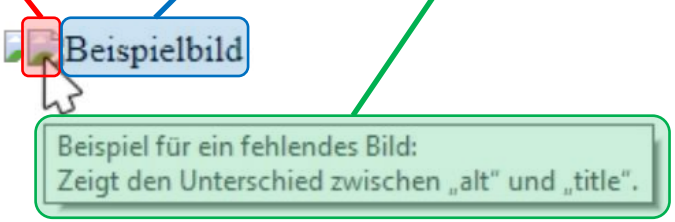
Image-Map (2/2) – Attribute des area-Tag

`alt`	Alternativtext für die Area (notwendig, falls `href` gesetzt)
`coords`	Koordinaten der Area in Pixel
`download`	Dateiname falls Ziel gespeichert werden soll (ab HTML5)
`href`	URL des Hyperlink-Ziels
`hreflang`	Sprachkode der Sprache des Ziels (ab HTML5)
`media`	gibt an, für welchen Medientyp das Ziel optimiert wurde (CSS `media query`; ab HTML5)
`nohref`	gibt an, dass die Area kein Ziel hat (nicht in HTML5)
`rel`	definiert Beziehung zwischen aktuellem Dokument und Ziel (`alternate`, `author`, `bookmark`, `help`, `license`, `next`, `nofollow`, `noreferrer`, `prefetch`, `prev`, `search`, `tag`; ab HTML5)
`shape`	geometrische Form der Area (`default`, `circle`, `poly`, `rect`)
`target`	Angabe, wo das Ziel geöffnet werden soll (`_blank`, `_parent`, `_self`, `_top`, `framename`)
`type`	Medientyp des Ziels (ab HTML5)

Unterschied alt vs title

- alt-Tag definiert alternativen Text für
  - fehlende Bilder (insbesondere bei 404er Status)
  - Bildschirmleser (barrierefreie Dokumente)
- title-Tag definiert Titel für HTML-Elemente
  - bei Mouse-Over anzuzeigen
  - bei Aktivierung anzuzeigen

```
<img src="/dev/null" alt="Beispielbild"
     title="Beispiel f&uuml;r ein fehlendes Bild:&#10;
            Zeigt den Unterschied zwischen &bdquo;alt&ldquo;
            und &bdquo;title&ldquo;." />
```

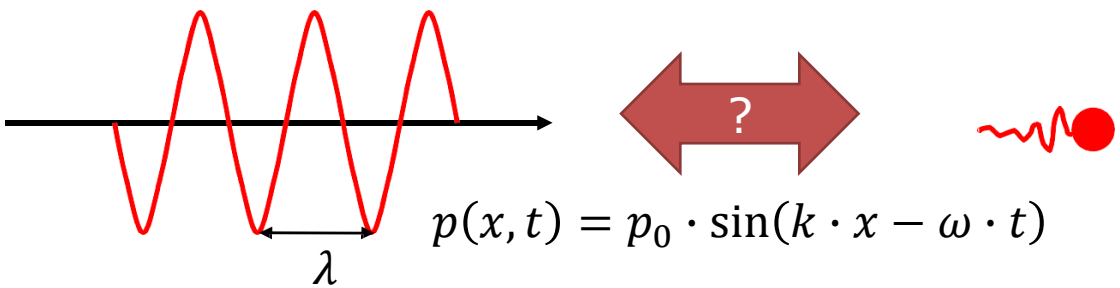


BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

Bilder: Licht und Farbmodelle

Licht – Beschaffenheit

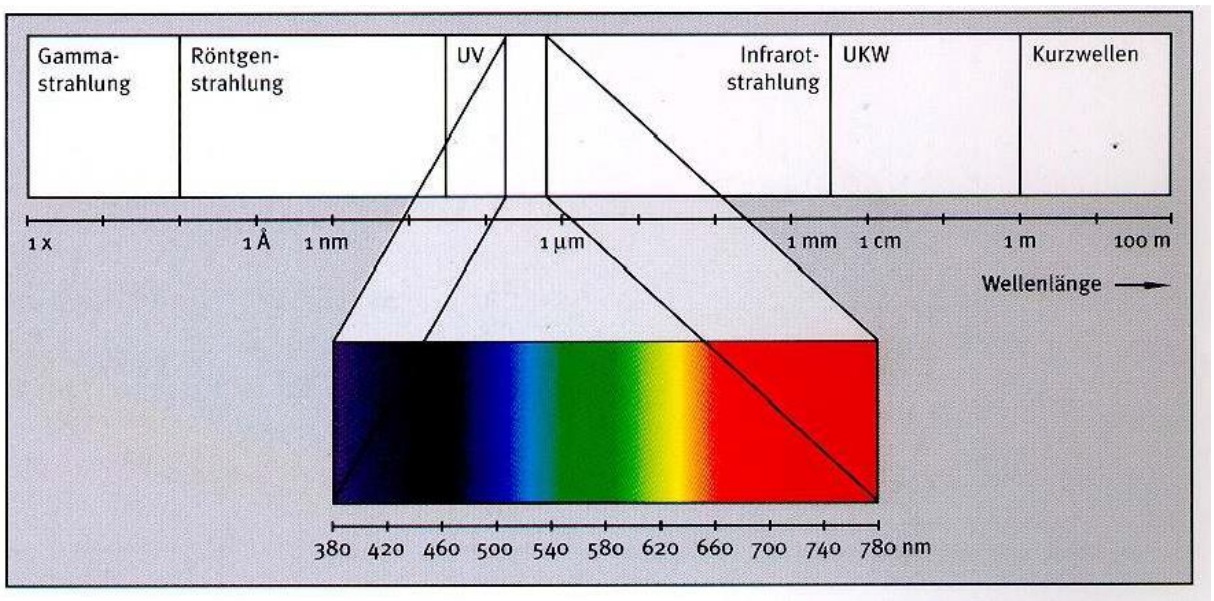
- Licht : in Photonen gequantelte transversale elektromagnetische Welle

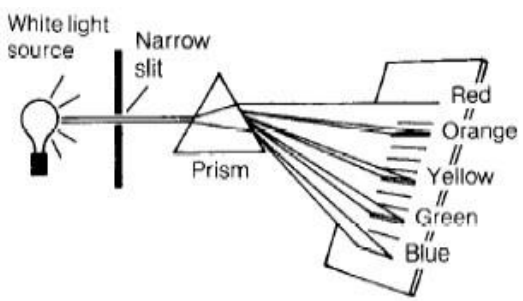


- Lichtgeschwindigkeit: $c = \lambda \cdot f$
- c nur vom Medium abhängig,
  Frequenz ist Maß für Energie: $E = h \cdot f = \frac{h \cdot c}{\lambda}$

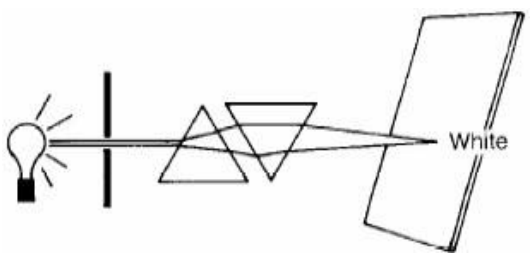
$h = 6{,}63 \cdot 10^{-34}\text{Js}$ (Planck'sches Wirkungsquantum)

LICHT – BESCHAFFENHEIT





Newton Experiment: Aufspaltung



Rekombination

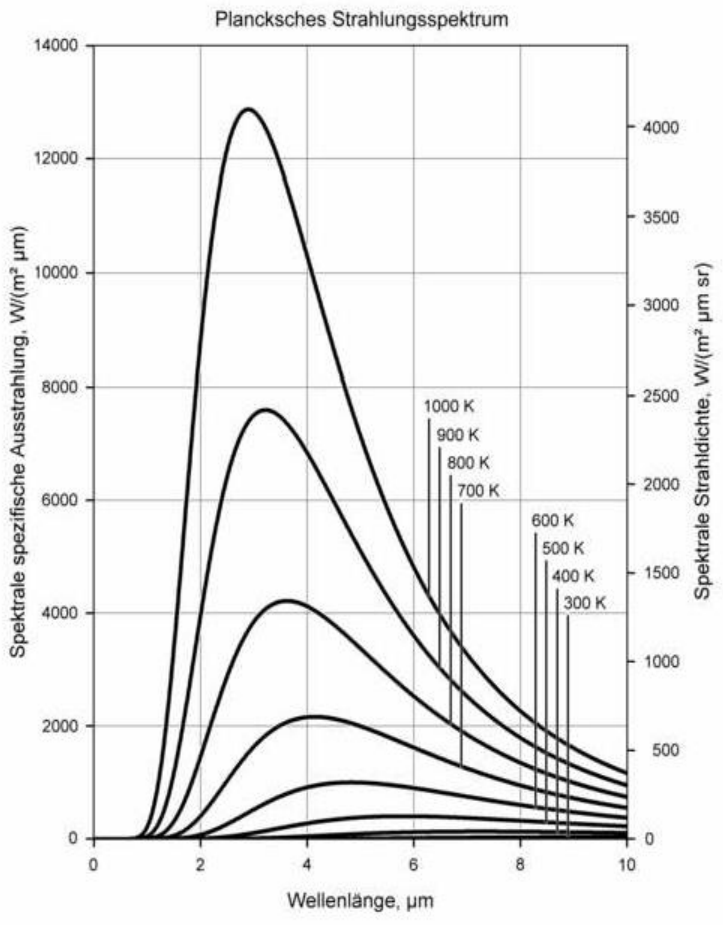
Licht – Emission

Glühen: Flüssigkeiten oder Feststoffe emittieren Licht bei Erhitzung über 1000K

- Theoretisches Model (Planck, 1900) der Schwarzkörperstrahlung beschreibt spektrale Leistungsverteilung:

$$M(\lambda) = \frac{2\pi \cdot h \cdot c^2}{\lambda^5 \cdot \left(e^{\frac{c \cdot h}{\lambda \cdot k \cdot T}} - 1\right)}, \qquad [M] = \frac{\mathrm{W}}{\mathrm{m}^3}$$

- wobei:
  - h … planck'sches Wirkungsquantum
  - c … Lichtgeschwindigkeit
  - k … Boltzmannkonstante
  - T … Absoluttemperatur
- basiert auf der Annahme von Quanten (Photonen)



Licht – Emission

- *Gasentladung*
  bei Stromdurchfluss leuchten Gase (i.d.R. geringer Dichte)
  → Wellenlänge des emittierten Lichts hängt vom Gas und der Dichte ab

- *Fotolumineszenz*
  einfallendes Licht hebt Elektronen in Phosphoren auf höhere Energieniveaus; bei Rückfall ins ursprüngliche Energieniveau wird für Phosphor charakteristisches, spektrales Licht emittiert

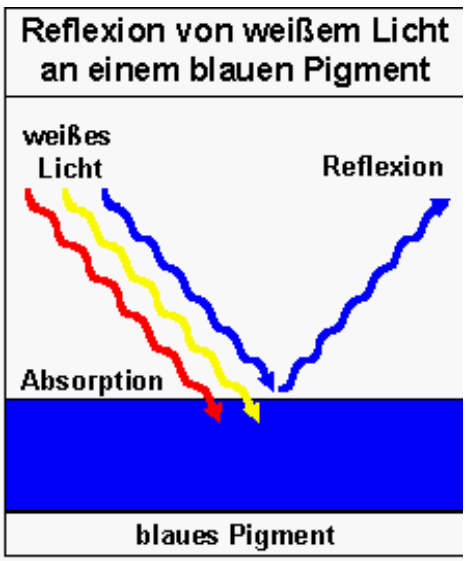
- *chemische Reaktion*
  bei Verbrennung wird chemische Energie in der Flamme in Licht umgewandelt
  (genauer: es werden Photonen charakteristischer Wellenlänge emittiert)

- *elektromagnetische Umwandlung*
  Strom und Elektronenbeschuss bringen Phosphore zum Leuchten
  (bspw. Röhrenbildschirm; Prinzip ähnlich Fotolumineszenz)

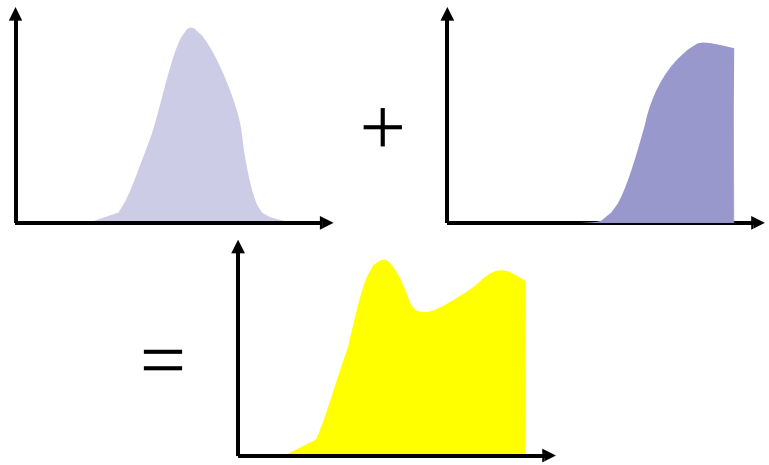
Licht – Reflexion

- *Absorption*
  beim Durchgang durch eine farbige Schicht, wird ein Teil des Spektrums absorbiert

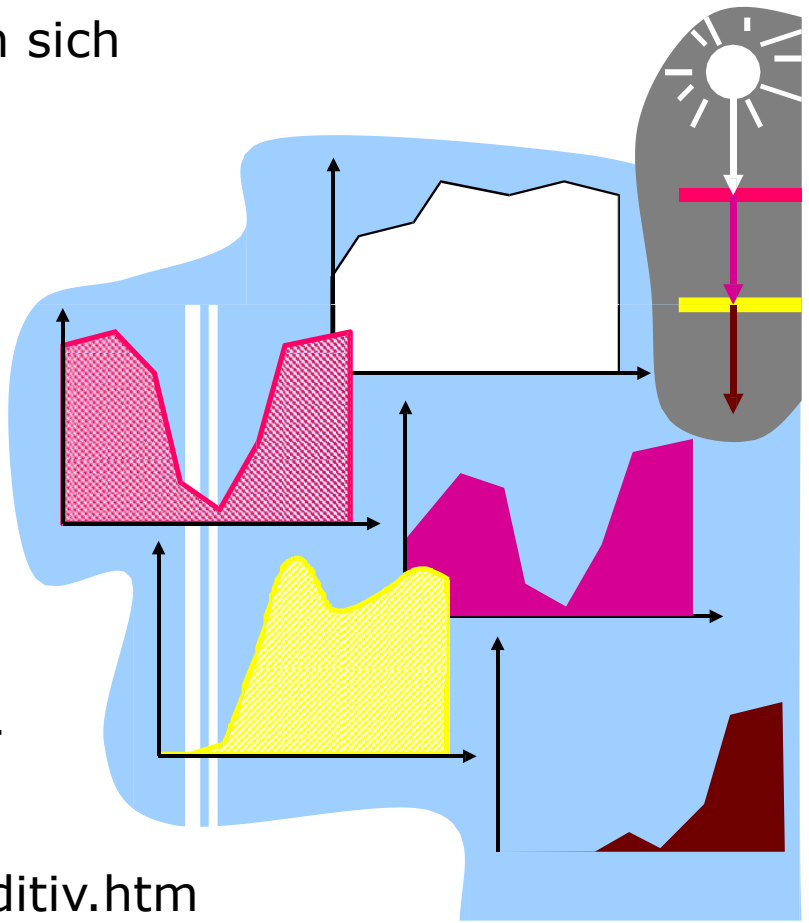
- *Reflexion*
  fällt Licht auf eine farbige Oberfläche wird ein Teil des Spektrums absorbiert und ein Teil reflektiert



Licht – Mischung



- *additive Farbmischung*
  Überlagerung von Licht mehrerer Emissionsquellen: Spektren addieren sich



- *subtraktive Farbmischung*
  Licht fällt auf Absorptions-/Reflektions-Filter: Spektren multiplizieren sich unter Filterreduktion

⇒ siehe auch: https://www.farbe.com/additiv.htm

Licht – Standardbeleuchtung (1/2)

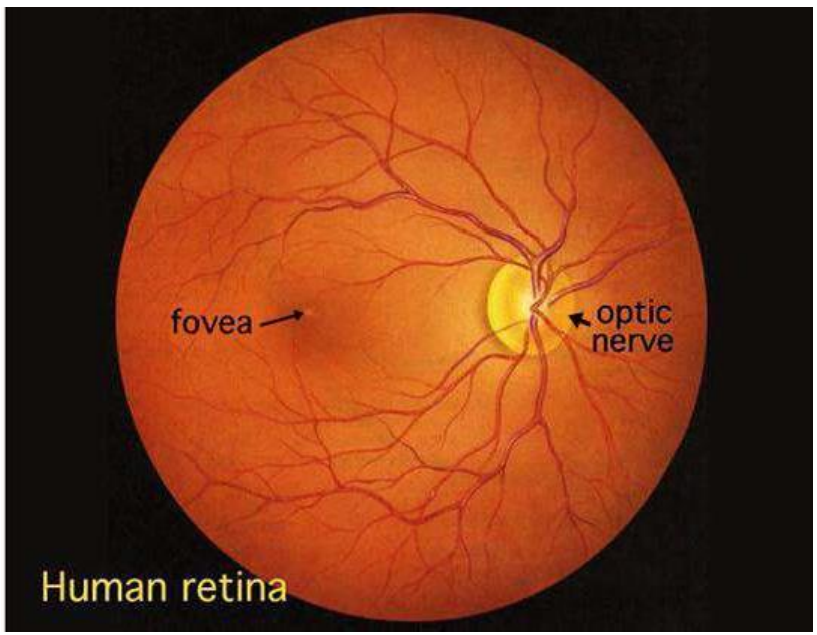
- gedruckte Farben erscheinen je nach Beleuchtung unterschiedlich (bspw. erscheint Gelb in magentaner Beleuchtung als Rot)
- definiere Standardbeleuchtung:
  - Illuminant A 2856K Wolfram Glühwendel
  - Illuminant B direktes Sonnenlicht
  - Illuminant C mittleres Sonnenlicht
  - Illuminant D55 5500K bewölktes Tageslicht
  - Illuminant D65 6500K Tageslicht auf Nordhalbkugel
  - Illuminant TL84 4000K 3-Band-Fluoreszenzlampe

Licht – Standardbeleuchtung (2/2)

Illuminant A | Illuminant B | Illuminant C

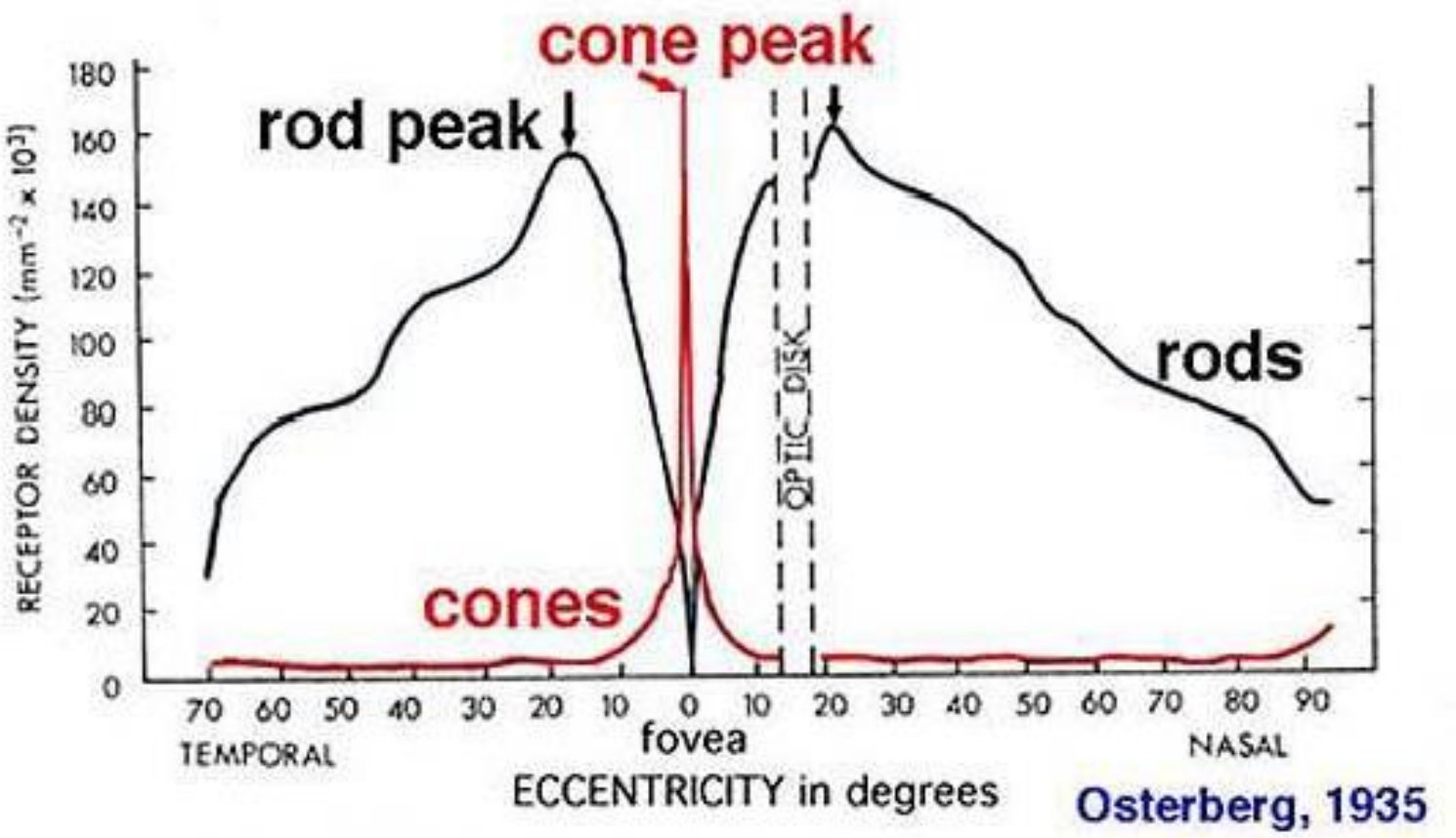
Gleiche Szene mit unterschiedlicher Beleuchtung

Lichtwahrnehmung – Retina (1/4)



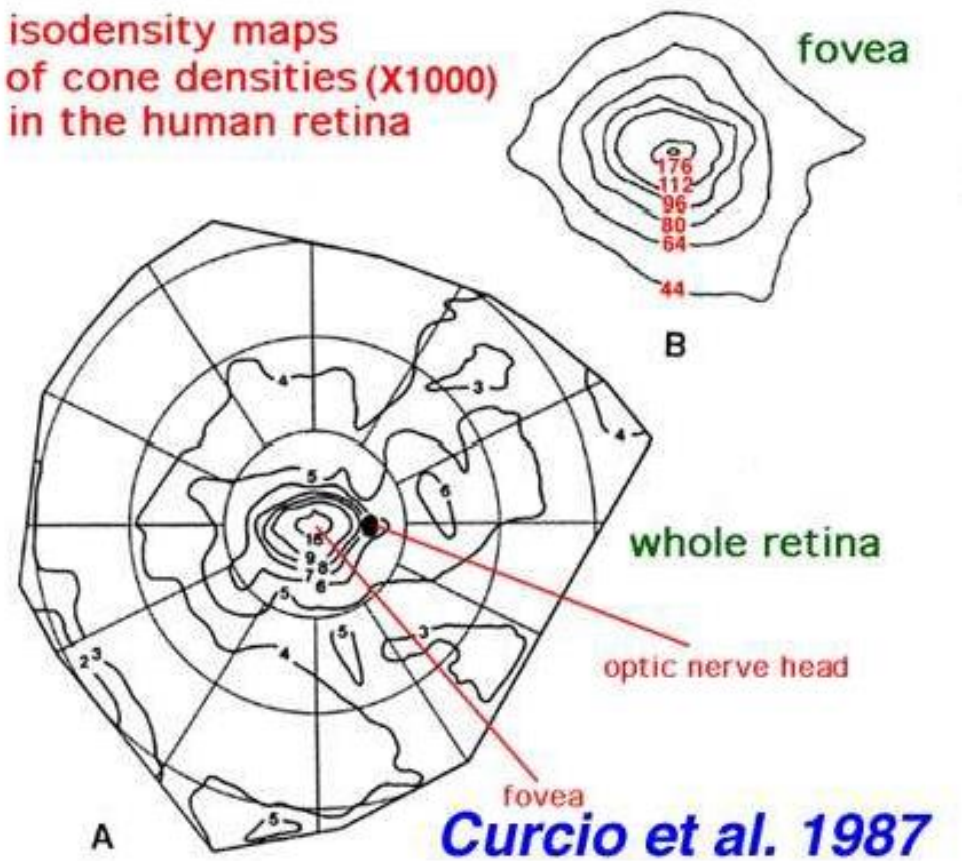
- *Retina*
  0,4mm dick,
  mit Fotorezeptoren besetzt
- *Fovea*
  Durchmesser ~1,5mm,
  zentral gelegene Region mit höchster Zäpfchenkonzentration aber ohne Stäbchen
- *Blinder Fleck*
  misst ~1,75mm im Durchmesser

LICHTWAHRNEHMUNG – RETINA (2/4)



Dichteverteilung der Stäbchen und Zäpfchen um die Fovea

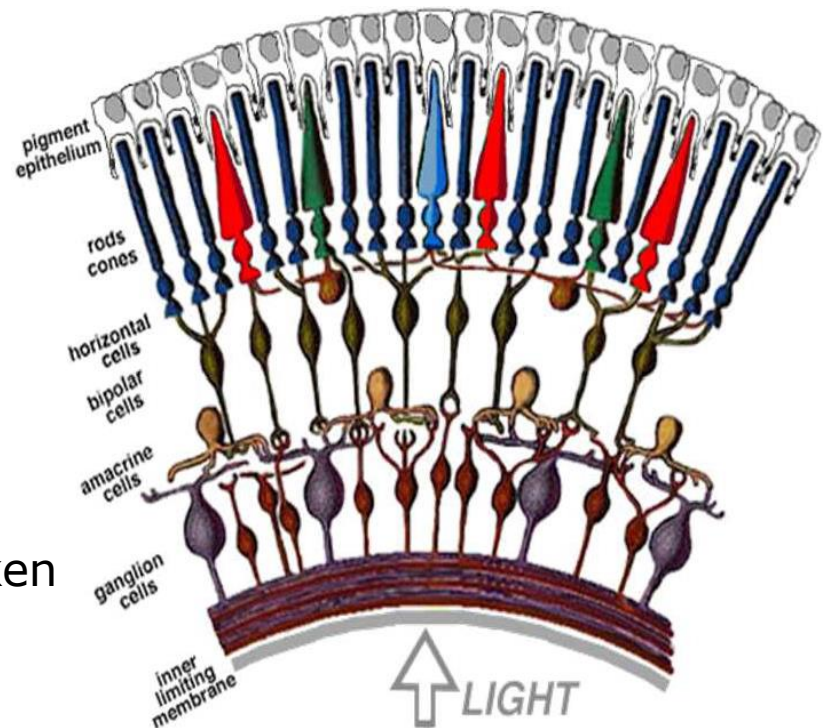
LICHTWAHRNEHMUNG – RETINA (3/4)



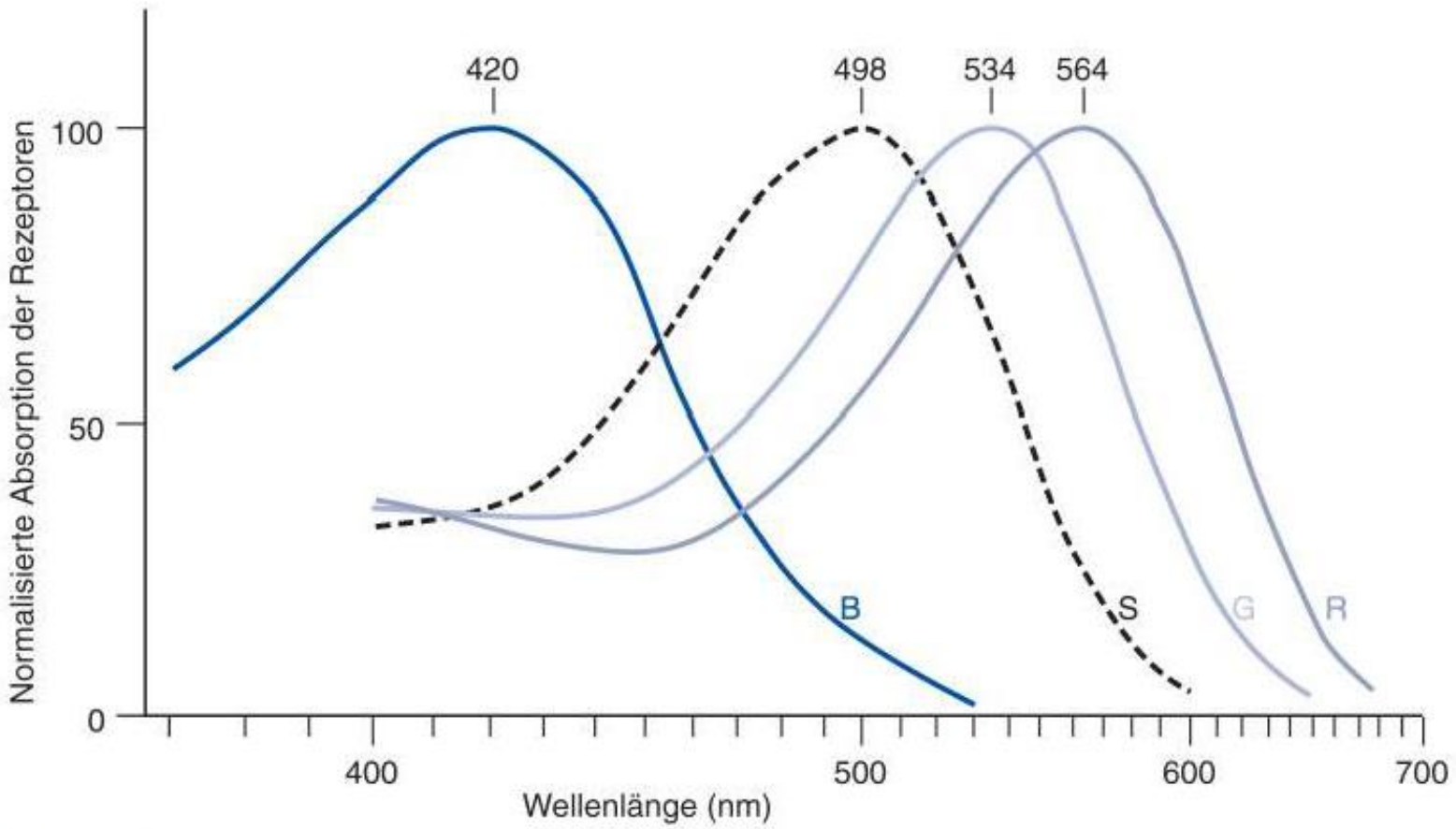
Dichteverteilung der Stäbchen und Zäpfchen um die Fovea

Lichtwahrnehmung – Retina (4/4)

- *Zapfen*
  drei Arten mit unterschiedlicher spektraler Empfindlichkeit
- *Stäbchen*
  sehr hohe Empfindlichkeit
- *Horizontalzellen*
  Zusammenführung von Rot/Grün, Blau/Gelb und Stäbchensignale in Kontrastsignale
- *Bipolarzellen*
  On- und Off- Zellen die Kontraste verstärken
- *Amakrinzellen*
  zusätzlich Empfindlichkeit auf Reizänderung
- *Ganglienzellen*
  sehr komplex, reagieren auf Farbe, Größe, Richtung und Geschwindigkeit



FARBWAHRNEHMUNG – REIZE

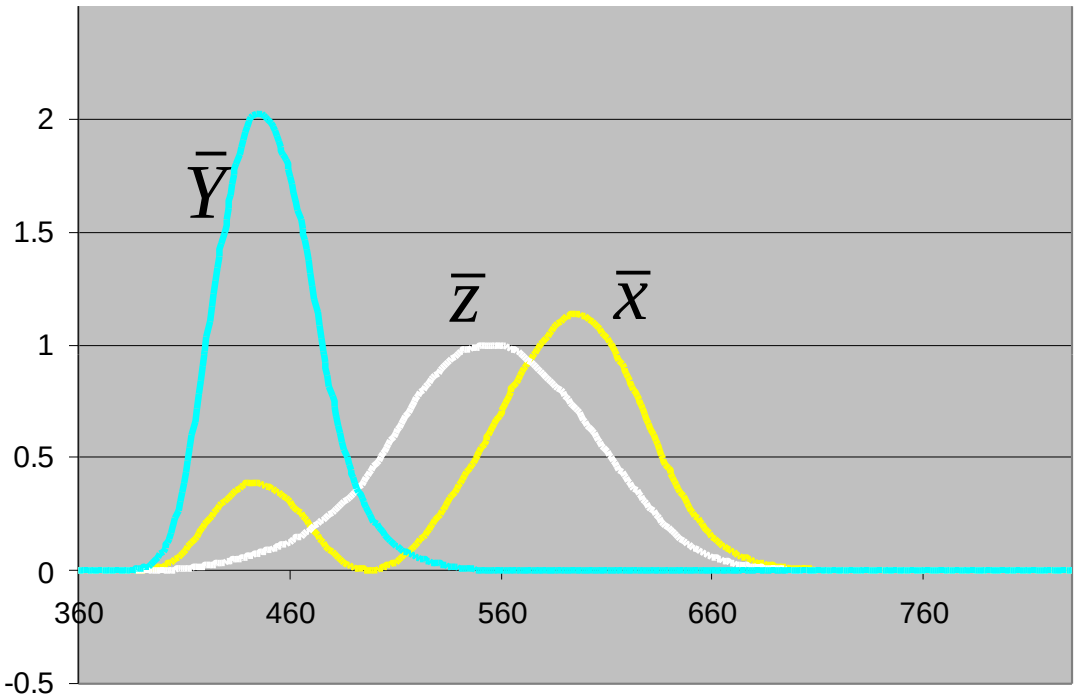


Antwortspektren von Stäbchen und Zapfen (R,G,B), Maximalwerte normalisiert auf 100

Farbmodelle – CIE 1932 (1/3)

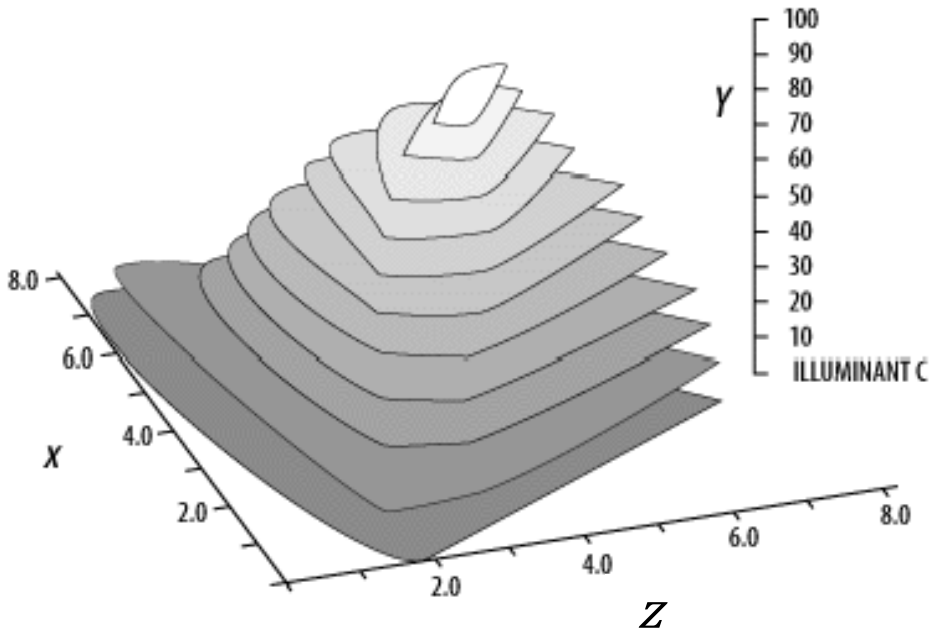
- Problem bei RGB: negative Farbvalenzen
- Lösungsansatz: chromatisches Farbsystem mit den Primärfarbvalenzen xzY der Commission Internationale de l'Éclairage (CIE)
- alle drei Farbabgleichsfunktionen sind positiv
- Y entspricht der Summe der spektralen Empfindlichkeiten der SML-Zapfen und somit der im Auge detektierten Helligkeit
- x und z sind ungefähr Rot/Grün- und Blau/Gelb-Unterschied

FARBMODELLE – CIE 1932 (2/3)

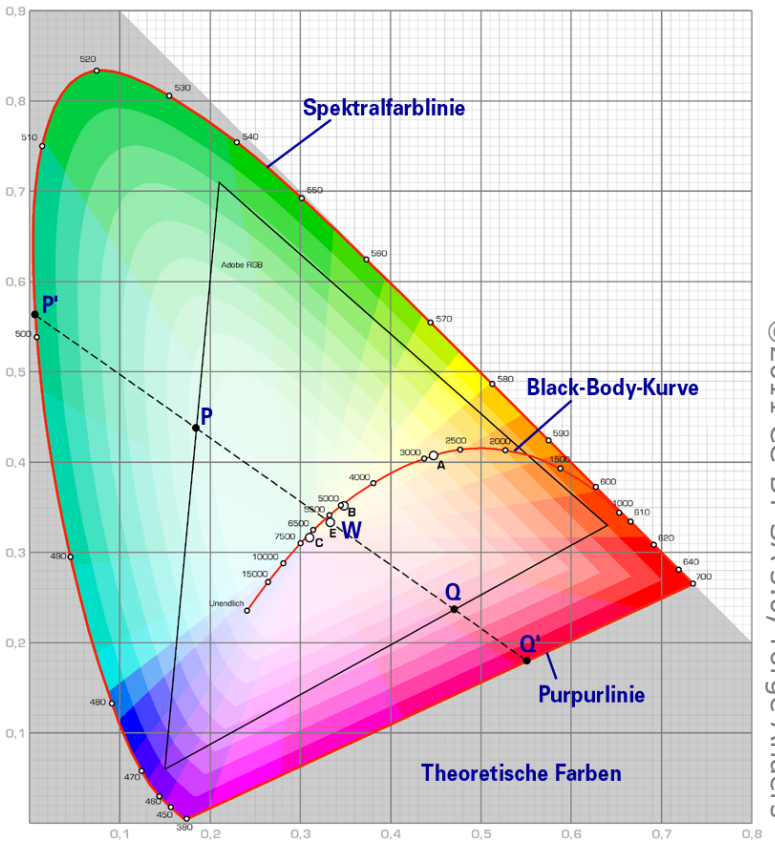


Farbableichsfunktionen (Colormatching Functions)
für x, z und Y (CIE 1932) → Tristimulusmodell

FARBMODELLE – CIE 1932 (3/3)



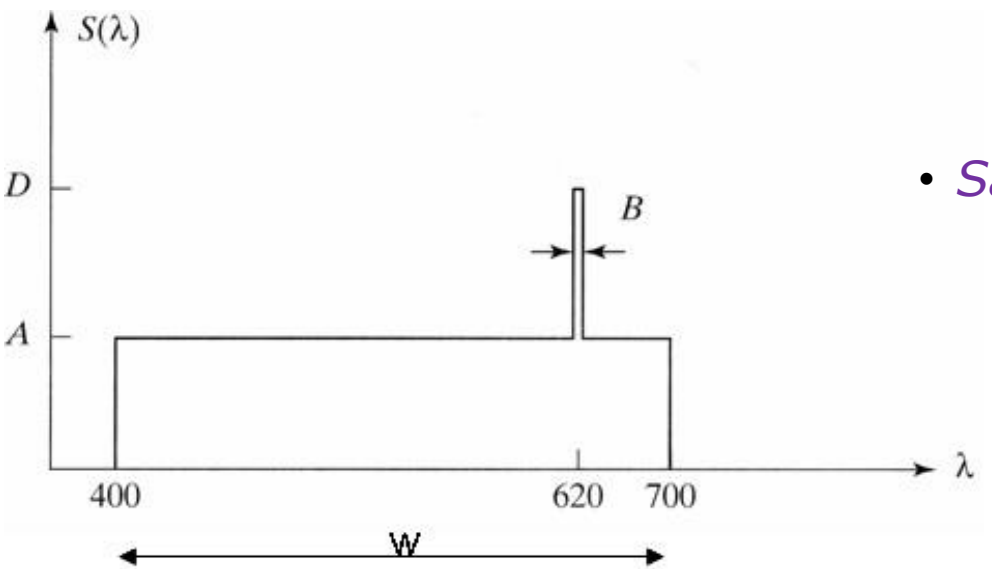
3D-Darstellung des xzY-Systems



Ebene maximaler Sättigung mit Verschiebung des Weißpunktes in Abhängigkeit von der Farbtemperatur

FARBWAHRNEHMUNG (1/2)

- Hauptmerkmale neben Helligkeit:
  - Farbton
  - Sättigung
- Berechnungsvorschriften (Hill, 2001) auf Grund des folgenden vereinfachten Modells des Spektrums:



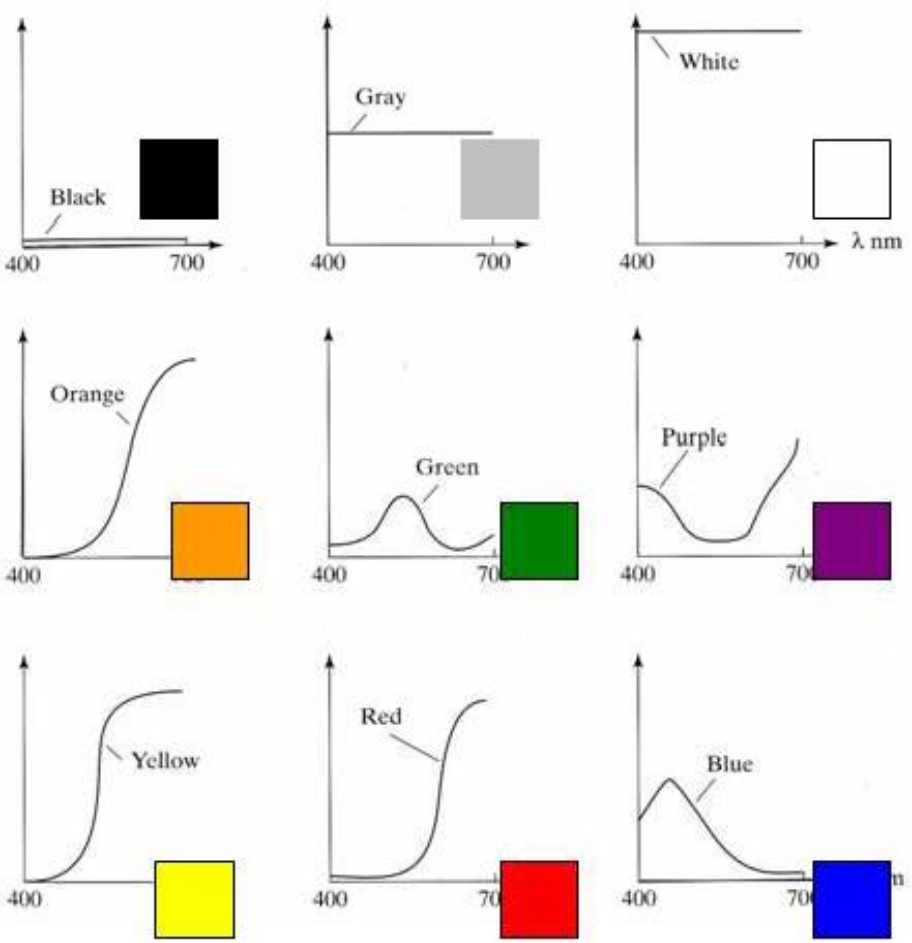
- *dominante Wellenlänge* ergibt den *Buntton* oder *Farbton* (*Hue*)
- *Helligkeit* (*Luminance*)
  → Gesamtleistung des Lichtes

$$L = (D - A) \cdot B + A \cdot W$$

- *Sättigung* (*Saturation*)

$$S = \frac{(D - A) \cdot B}{L} \cdot 100\%$$

FARBWAHRNEHMUNG (2/2)



aus F.S. Hill: Computer Graphics Using Open GL, 2001

- zusätzlich werden empfundene Farben in Bereiche eingeteilt und benannt
- als Bereiche ergeben sich weiß, grau, schwarz und verschiedene Farbtöne
- viele Farbsysteme basieren auf dieser Zerlegung der Farbtöne
- menschliche Wahrnehmung:
  - 128 Farbtöne
  - 130 Farbsättigungen
  - 16 (blau) bis 26 (gelb) Helligkeitswerte
  - ~7 Mio. wahrnehmbare Farbnuancen

USABILITY – FARBEN

- Einsatz von Farben in Text
  - Hervorhebung
  - Bewertung von Worten
- wichtige Grundregeln
  - pro Text soll jede Farbe mit genau einer festen Bedeutung verbunden sein
  - Alltagsbedeutungen von Farben sollten beachtet werden (Kulturkreis beachten, bspw. Europa: rot = Warnung)
  - Farben mit großer spektraler Distanz können nicht gleichzeitig vom Auge fokussiert werden

  - auch Farbenblinde sollten uneingeschränkt das System nutzen können (Simulation möglich)
  - beste Effekte mit sparsam eingesetzten, leuchtenden Farben, die zwischen eher matteren Tönen eingesetzt werden

Bilder: Farbräume

MUNSELL-FARBSYSTEM (1/3)

- im Sinne der Kunst: *describe color in a rational way*
- Farbglobus nach menschlicher Unterscheidungsmöglichkeit abgestuft:
  - Farbton (Hue) um Äquator
    - 100 Schritte
    - 10 Buchstabenkategorien zu je 10 Unterteilungen
  - Helligkeit (Value) in 10 Unterteilungen von unten nach oben

    1 = schwarz

    …

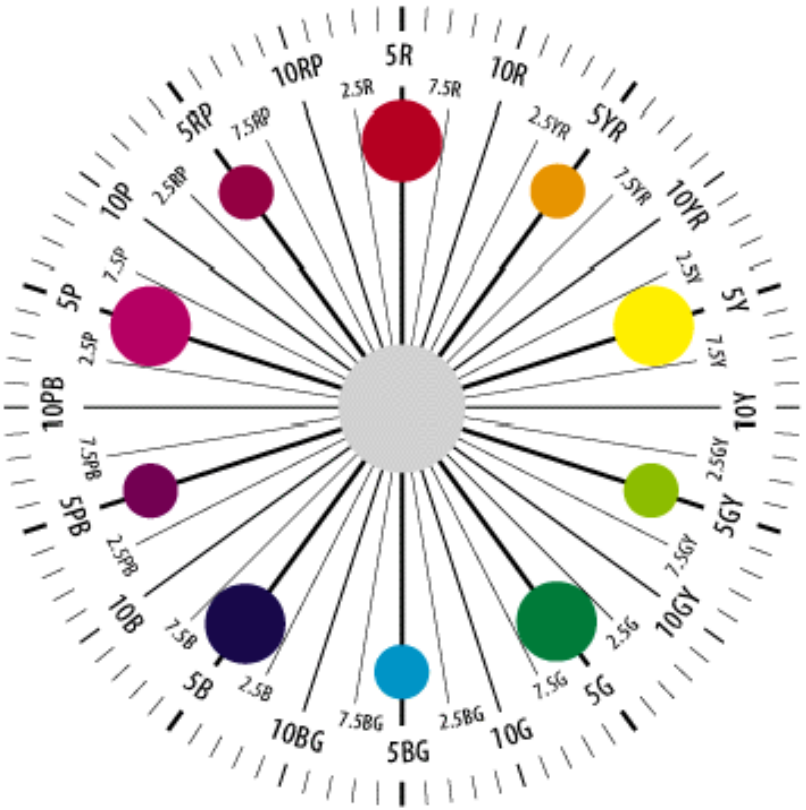
    10 = weiß
  - Sättigung (Chroma) von Achse horizontal nach außen

    0

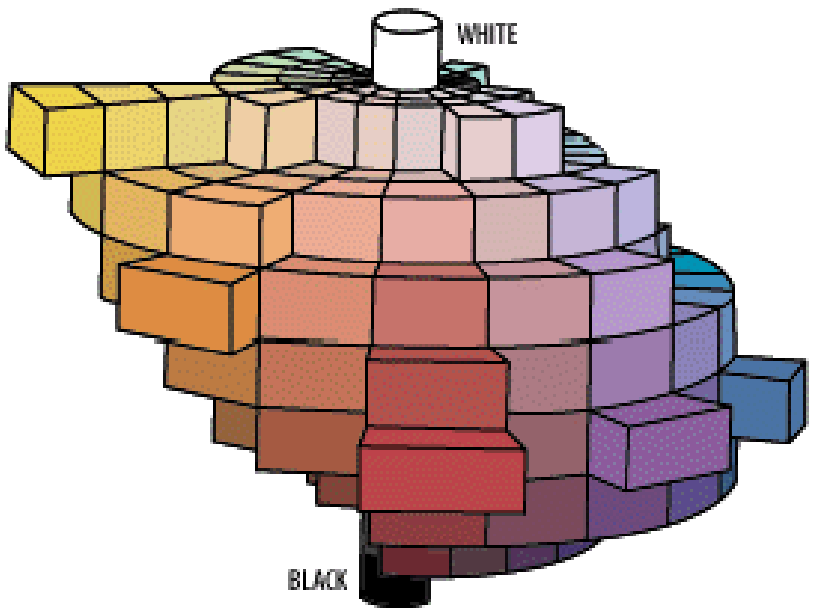
    …

    von Helligkeit abhängiges Maximum

MUNSELL-FARBRAUM (2/3)

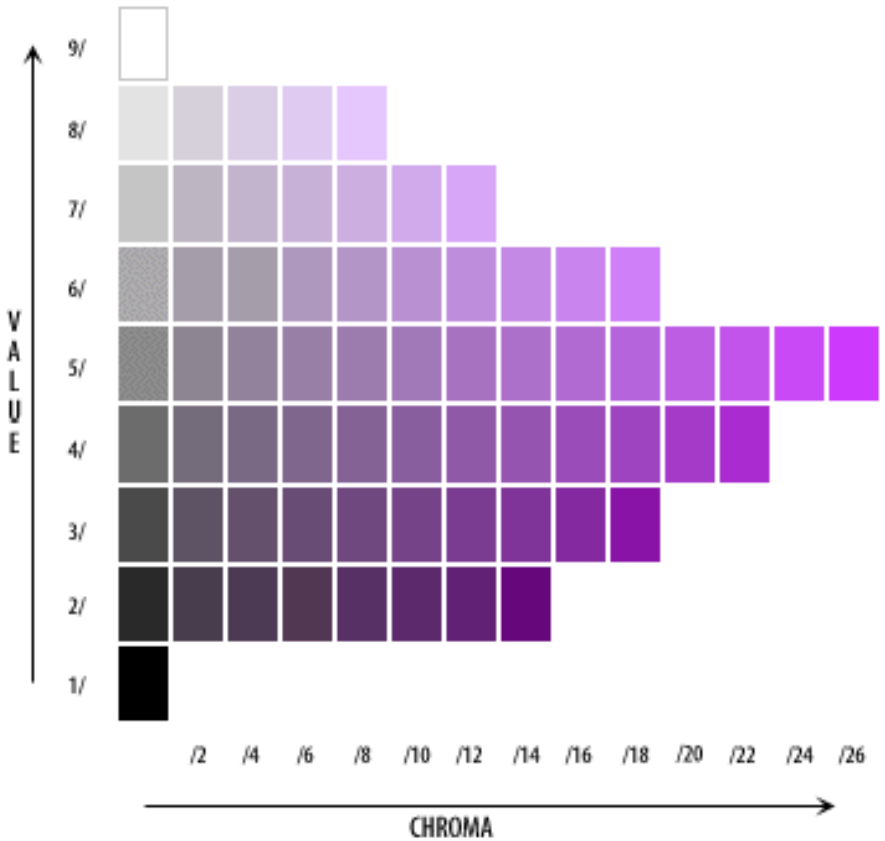


Farbtonabstufungen um den Äquator

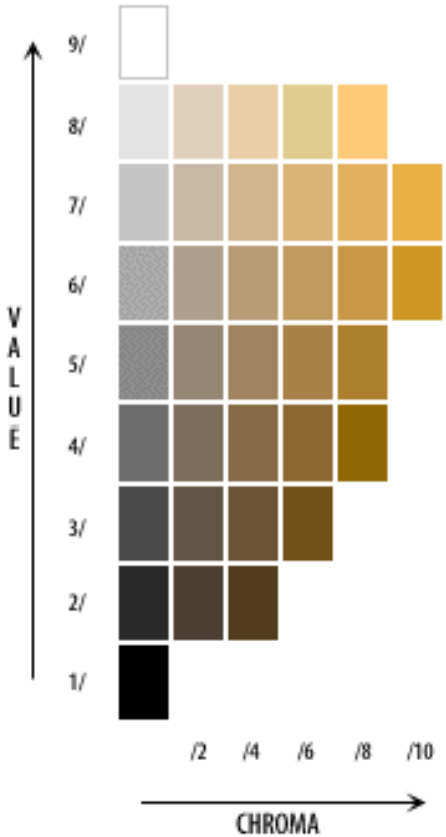


Munsell-Farbglobus

MUNSELL-FARBRAUM (3/3)



Sättigungs- und Helligkeitsabstufungen des Farbtons 5RP



Sättigungs- und Helligkeitsabstufungen des Farbtons 10YR

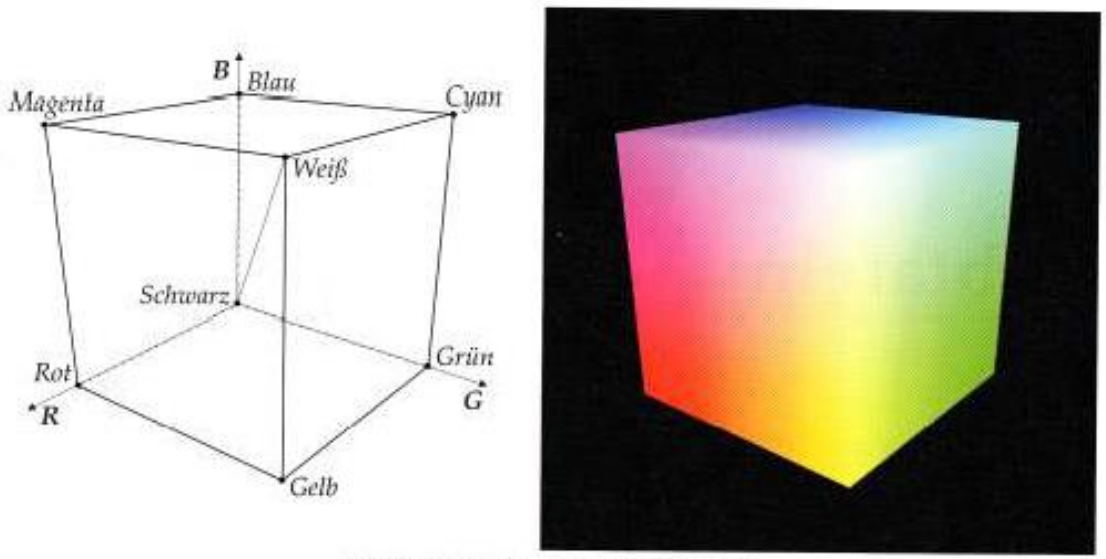
Druckfarbräume

- Pantone (pantone.com)
  - Farbstandard für über 3000 Volltonfarben
  - Vielseitigkeit für Graphiker
  - schlechte Mischbarkeit (Farbseparation)
  - Definition über Farbmuster

- Truematch (trumatch.com)
  - System zum Farbabgleich
  - CMYK-Farben aus >2000 Prozessfa

- Focoltone (focoltone.com)
  - kalkulierbare Prozessfarben
  - 763 Farben aus CMYK

RGB (1/2)

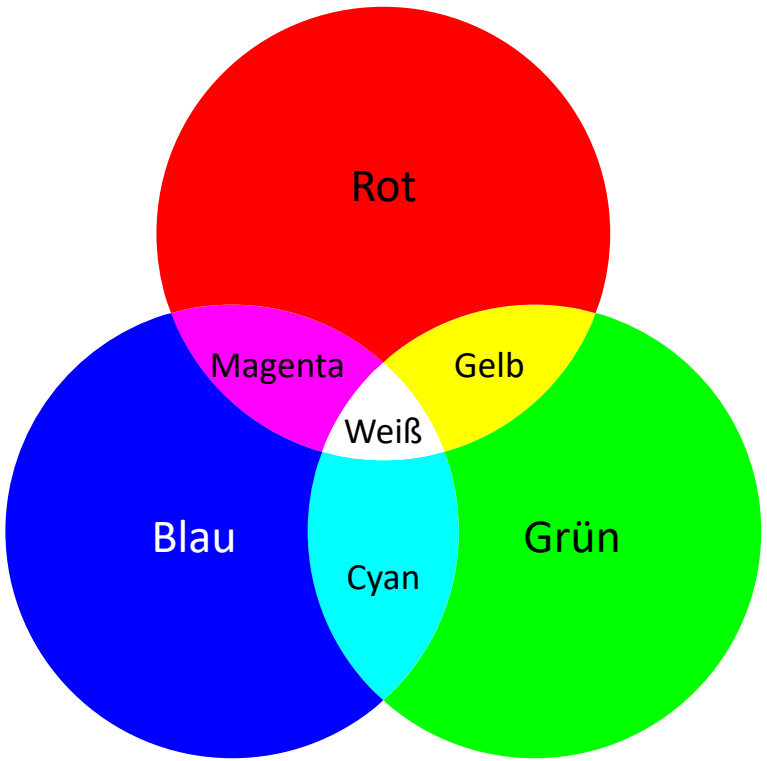


- Abspeichern der Primärfarbanteile in 1 Byte (256 Abschnitte)
- RGB-Modell: $256^3 \sim 16{,}7$ Millionen Farben (True Color)
- Darstellung bei Monitoren, Projektoren, …

RGB (2/2)

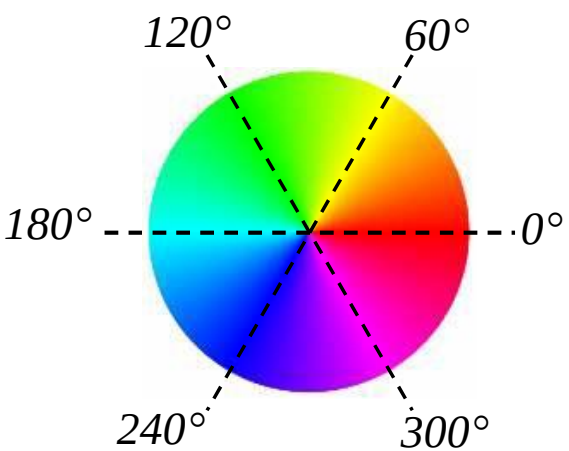
- additive Farbmischung
- Primärfarben
  - Rot $R = (1,0,0)$
  - Grün $G = (0,1,0)$
  - Blau $B = (0,0,1)$
- Mischfarben

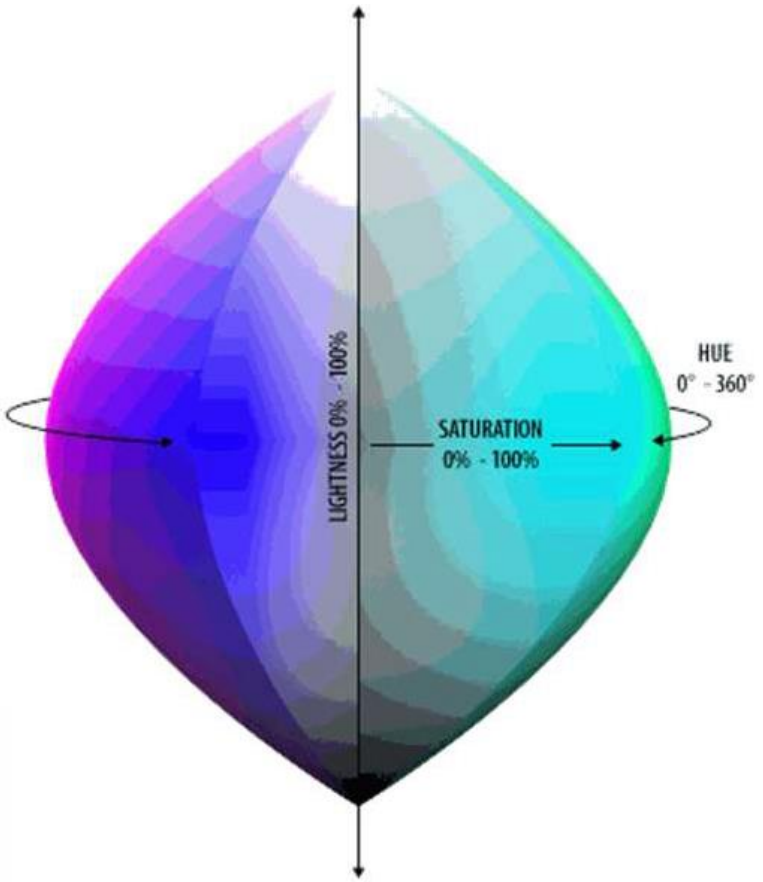
$$\forall 0 \leq x \leq 1, x \in \{a, b, c\}: F = aR + bG + cB$$



HLS

- Hue, Luminance, Saturation
- $H \in [0,360]$ $(0 = 360 = \text{rot})$
- $L \in [0,100]$ (dunkel…hell)
- $S \in [0,100]$ (matt…brillant)





Umrechnung RGB vs HLS

```
void hlsToRgb() {
    H /= 60; L /= 100; S /= 100;
    I = round(H);
    F = H-I;
    if (L <= 0.5) max = L*(1+S)
    else max = L*(1-S)+S;
    min = 2*L - max;
    DM = max - min;
    if (S == 0) [R,G,B] = [L,L,L];
    if (I==0) [R,G,B] = [max, min + F*DM, min]
    if (I==1) [R,G,B] = [min+(1-F)*DM,max,min]
    if (I==2) [R,G,B] = [min,max,min+F*DM]
    if (I==3) [R,G,B] = [min,min+(1-F)*DM,max]
    if (I==4) [R,G,B] = [min+F*DM,min,max]
    if (I==5) [R,G,B] = [max,min,min+(1-F)*DM]
}
```

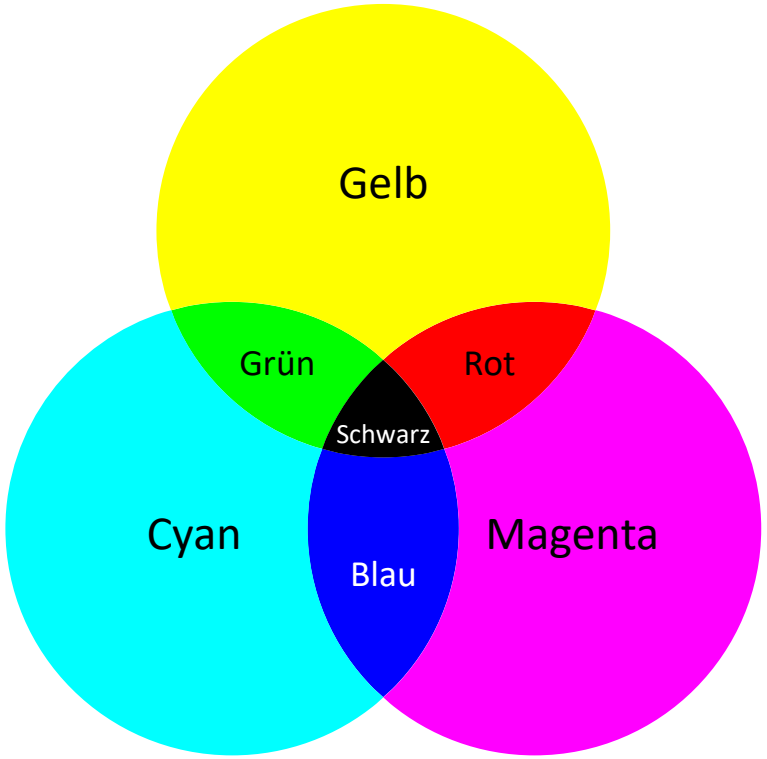
```
void rgbToHls() {
    max = getMax(R,G,B);
    min = getMin(R,G,B);
    L = (max+min)/2;
    if (min == max) [H,L,S] = [0,L,0];
    else {
        DM = max-min;
        SM = max+min;
        S = (L <= 0.5) ? DM/SM : DM/(2-SM);
        if (R == max) H = (G-B)/DM;
        else
            if (G == max) H = 2+(B-R)/DM;
            else H = 4+(R-G)/DM;
        if (H < 0) H += 6;
    }
    H *= 60;
    L *= 100;
    S *= 100;
}
```

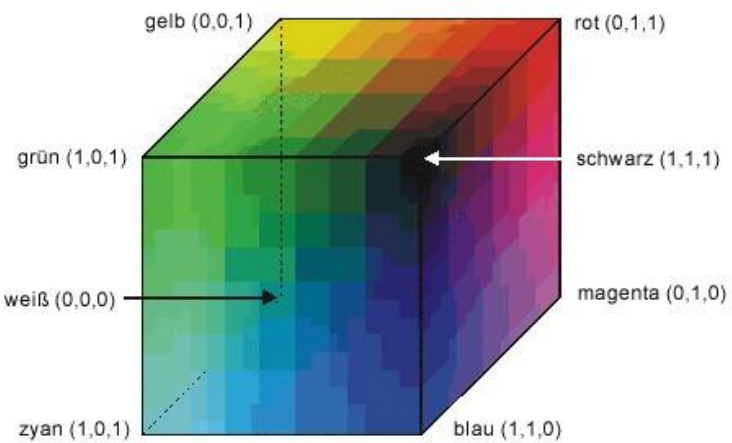
CMY

- CMY-Farbmodell eignet sich für die Körperfarbmischung / subtraktive Farbmischung
  (d.h. Lasurfarben, Druckfarben)
- Umrechnung vom RGB-Modell ins CMY-Modell:

$$(r, g, b) = (1,1,1) - (c, m, y)$$

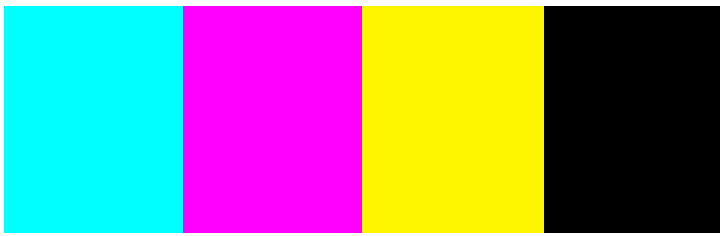
- Darstellung für Drucker macht nur bei gleichem Farbgamut Sinn

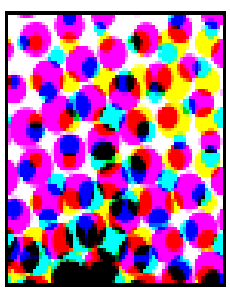




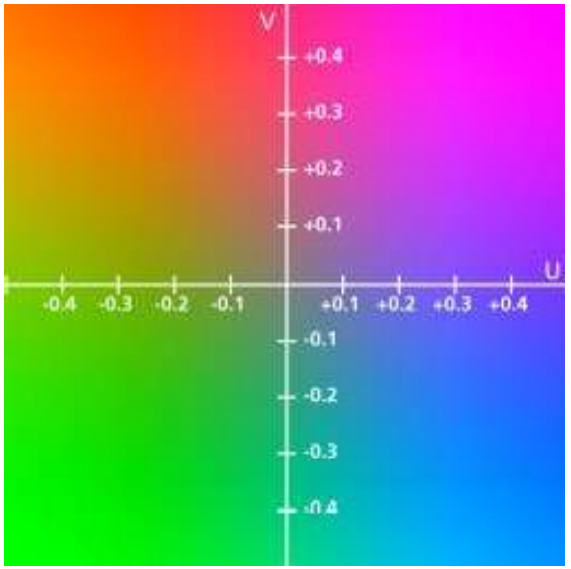
CMYK

- K steht für Black
- Schwarz braucht dann nicht mit Buntfarben gemischt werden
  → Ausnahme: fettes Schwarz
- Anwendung in der Druckindustrie (Prozess-Farben-Modell)





WEITERE LINEARE FARBRÄUME



- Lab:
  physiologisch korrekt und umfassend RGB, CMYK
- YUV (PAL):
  - Y … Helligkeit (Schwarz-Weiß-Kanal)
  - U,V … Farbbalance
- YC_bC_r (digitale Bildformate):

$$f_{YC_bC_r} = \begin{pmatrix} f_Y \\ f_{C_b} \\ f_{C_r} \end{pmatrix} = \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}168736 & -0{,}331264 & 0{,}5 \\ 0{,}5 & -0{,}418688 & -0{,}081312 \end{pmatrix} \begin{pmatrix} f_R \\ f_G \\ f_B \end{pmatrix}$$

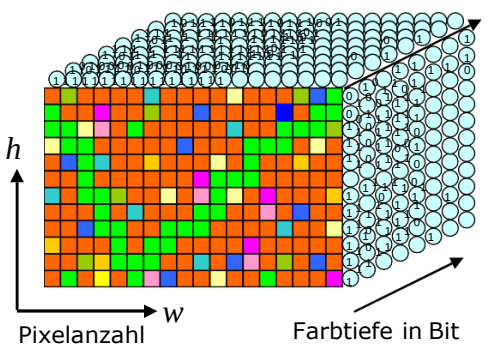
- YIQ (NTSC bis 1970):
  - I, Q … Farbbalance

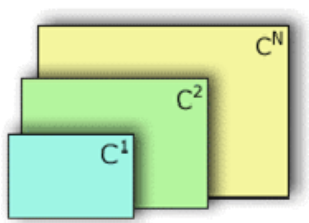
Bilder: Haupteigenschaften

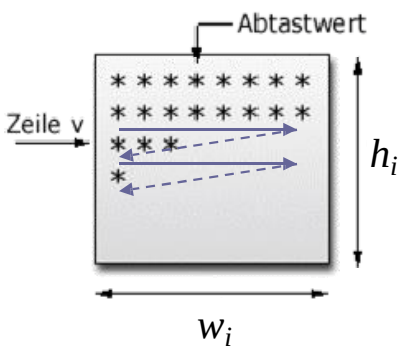
AUFBAU

- Digitales Bild: Sequenz von Pixeln
  - $w \cdot h$ Pixel (w Pixel/Zeile × h Zeilen)
  - je Pixel ein Farbwert mit b Bit kodiert (Farbtiefe)
  - Farbwert besteht aus n Farbebenen (RGB: $n = 3$)
  - jede Farbebene c_i besteht aus $w_i \cdot h_i$ Bildpunkten, die mit b_i Bit kodiert sind (pro Ebene variabel)
- Größe von Bilddateien in Byte:

$$G = \sum_{i=1}^{n} (w_i \cdot h_i \cdot b_i)$$







QUALITÄT

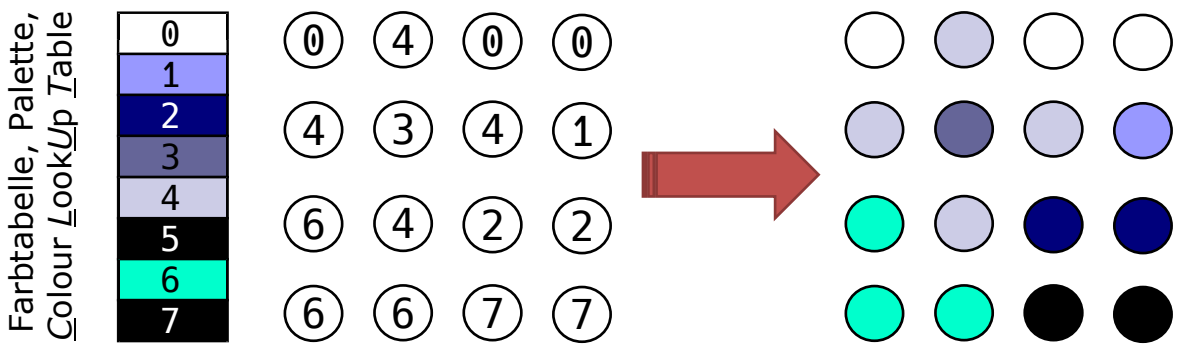
- *Bildqualität* ist eine Funktion über die Pixel- und Farbanzahl
- *Speicherbedarf* ist ebenfalls eine Funktion über Pixel- und Farbanzahl
- *subjektive Bildqualität* ist eine Funktion über die Gestaltung und Bildbearbeitung
- gesuchtes Optimum: Speicher vs. Qualität

Farbtiefe	1 Bit	2 Bit	4 Bit	8 Bit	16 Bit	24 Bit	30 Bit	32 Bit
Farbanzahl	2	4	16	256	65k	16m	1g	4g

True Color

INDIZIERTE FORMATE: INDEXED COLOURS („PSEUDOFARBEN“)

- wenn in einem Bild wenig verschiedene Farben auftreten, ist es effizienter pro Pixel einen Index in eine Farbtabelle zu speichern
- bei 8 Farben werden beispielsweise nur 3 Bit pro Pixel benötigt



FARBREDUKTION

24 Bit - True Colour

8 Bit – 256 Farben

4 Bit – 16 Farben

3 Bit – 8 Farben

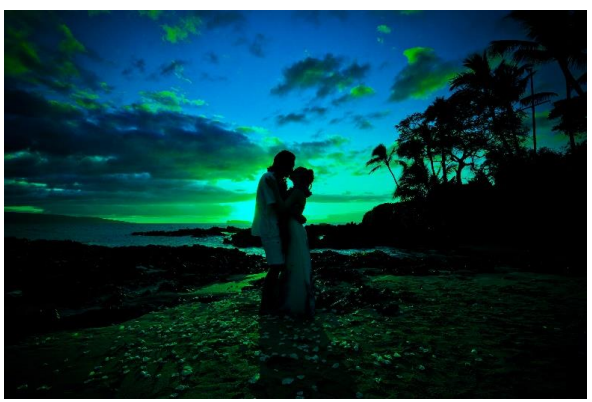
RGB

R G B

YC_BC_R

Y

C_b

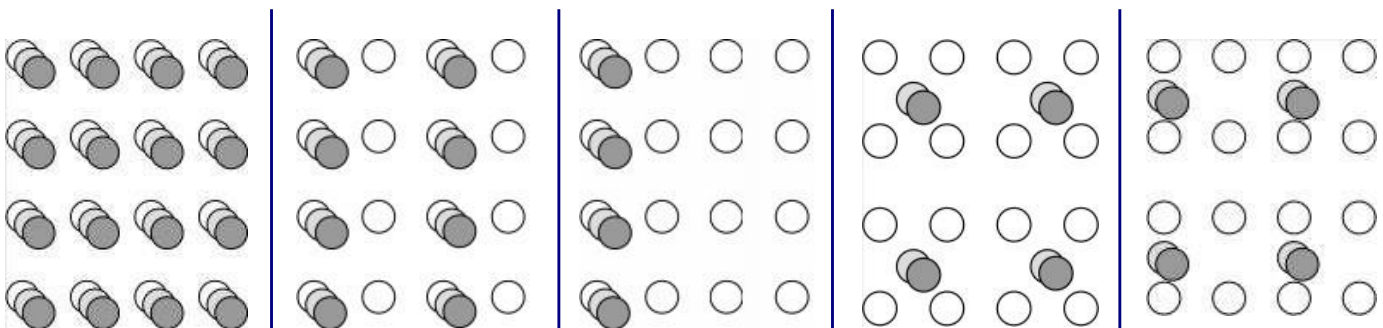


C_r

Chroma Subsampling

in YUV- und YC_bC_r-Bildern werden die Farbigkeitskomponenten (Chroma) meist mit kleiner räumlicher Frequenz abgetastet:

- 4:4:4 … gleiche Abtastung in Leuchtkraft und Chroma
- 4:2:2 … horizontal halbe Chroma-Frequenzen
- 4:2:0 … horizontal und vertikal halbe Chroma-Frequenz



ALPHA-KANAL

- zusätzlich zu den Farbkanälen kann ein α-Kanal definiert werden, in dem pro Pixel ein Opazitätswert zwischen 0% und 100% gespeichert wird
- ist c_P die Farbe des Bildpixels, c_H die Farbe des Hintergrunds und α der alpha-Wert, so berechnet sich die Ausgabefarbe zu:

$$c_A = \alpha \cdot c_P + (1 - \alpha) \cdot c_H$$

ALPHA-KANAL – BEISPIEL

Bild mit α-Kanal

Integration des Bildes

Speicherbedarf

- Beispiel 1: Full-HD RGB-Monitor in 24-Bit True Colour
  - geg.: $n = 3$, $w_i = 1.920$, $h_i = 1.080$, $b_i = 8\text{bit} = 1\text{B}$
  - Rechnung: $G = (1.920 \cdot 1.080)\text{px} \cdot 3 \cdot 8\frac{\text{bit}}{\text{px}} = 49.766.400\text{bit} = 6.220.800\text{B}$ $\cong 5{,}9\text{MiB}$
- Beispiel 2: YUV 4:2:2-HDTV in 24-Bit True Colour
  - geg.: $n = 3$, $w_1 = 1.280$, $w_{2,3} = 640$, $h_i = 720$, $b_i = 8\text{bit} = 1\text{B}$
  - Rechnung: $G = (2 \cdot 1.280 \cdot 720)\text{px} \cdot 8\frac{\text{bit}}{\text{px}} = 14.745.600\text{bit} = 1.843.200\text{B}$ $\cong 1.800\text{KiB}$
- bei 4K-TV ergibt sich
  - $G_{RGB} \cong 25{,}3\text{MiB}$
  - $G_{YUV(4:2:2)} \cong 16{,}9\text{MiB}$

Kenngrößen der Bildauflösung

px … Pixel = Picture Element (1 Farbwert)

ppi … Pixels per Inch: Auflösung von Bildern in Bildpunkten

dpi … dots per Inch: Auflösung auf Tintenstrahl-/Laser-Druckern

lpi … lines per Inch: Anzahl Zeilen pro Inch

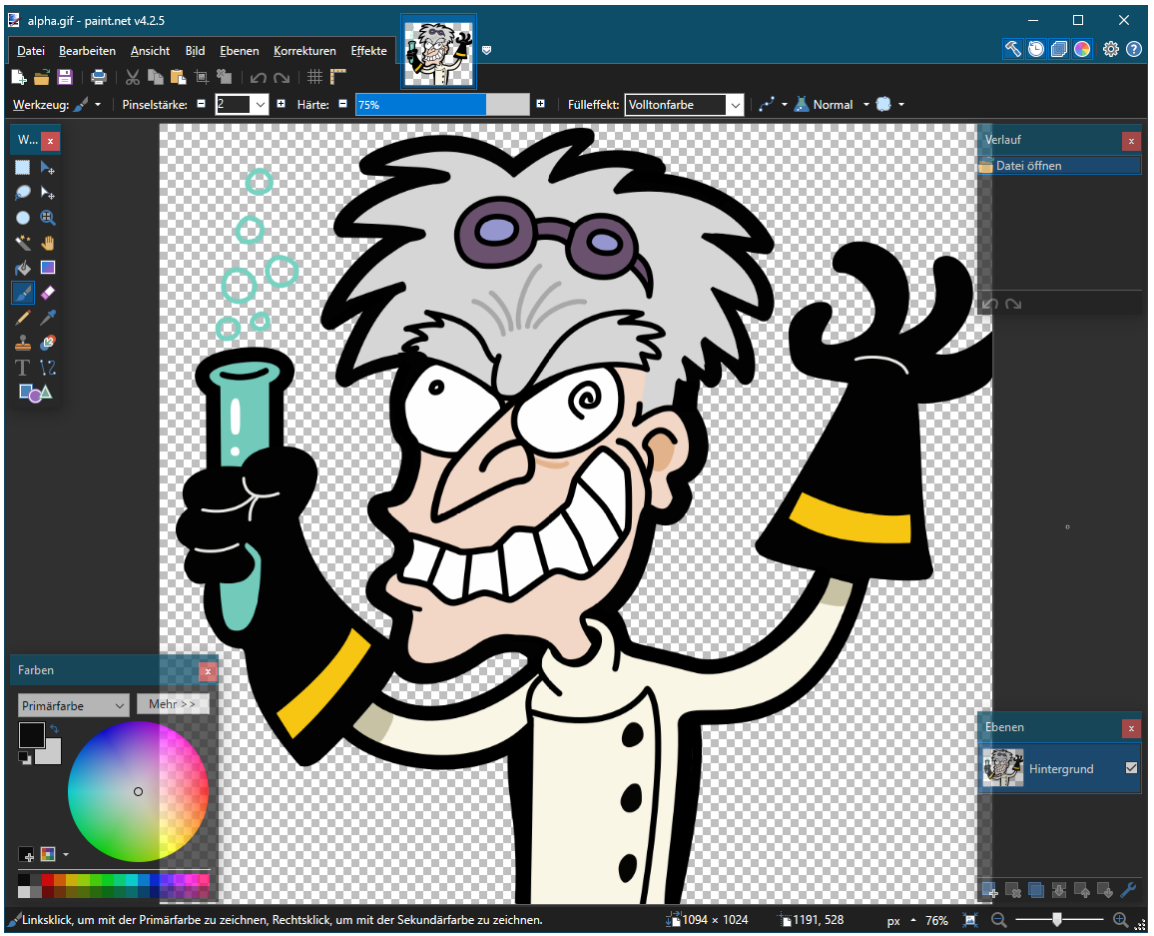
Bilder: Bildformate

BILDFORMATE

- wichtige Web-Formate
  - GIF Graphics Interchange Format
  - PNG Portable Network Graphic
  - JPG/JPEG Joint Photographic Experts Group
- weitere Formate
  - TIF/TIFF Tagged Image File Format
  - PSD Photoshop Document

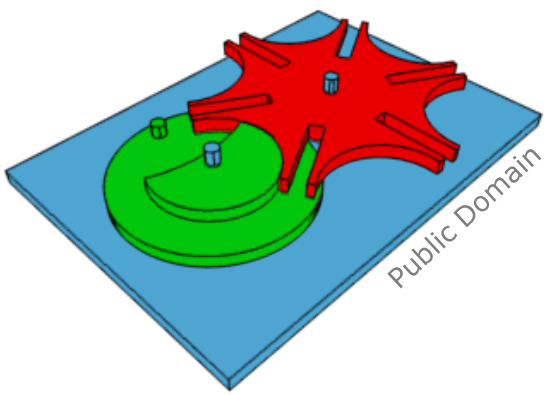
Graphics Interchange Format (GIF)

- indiziertes Format, mit Farbpaletten von bis zu 8 Bit Farbtiefe
- ursprünglich kein Alphakanal möglich (erst ab GIF89a)
- durch Einteilung in Bildblöcke mehr als 256 Farben möglich (256^3 (True Colour) möglich)



Einsatz von GIF

- Strichzeichnungen
- große Landkarten
  - maximale erlaubte GIF-Bildgröße:
    65.535×65.535 = 4.294.836.225px
  - entspricht bei bspw. A0
    - einem Bildformat von 46.354×65.535px
    - einer Auflösung von 1.400dpi
- Diagramme
- Animationen
  (Bildsequenz mit festgelegtem zeitlichen Abstand)



Portable Network Graphic (PNG)

- 1994 entwickelt (aufgrund Lizenzproblemen bei GIF)
- bessere Kompression als GIF und verlustfrei
  - Farbtiefe bis zu 48 Bit
  - Graustufenbildern mit 1, 2, 4, 8 oder 16 Bit
  - Farbbildern (RGB) mit 8 oder 16 Bit pro Kanal
- Interlacing für geringere Downloadzeit bis zur ersten Anzeige
- Unterstützung von mehreren Alphakanälen
- Gammakorrektur der Bildhelligkeit (auf jeder Plattform genau gleiche Darstellung)

Nachteile von PNG

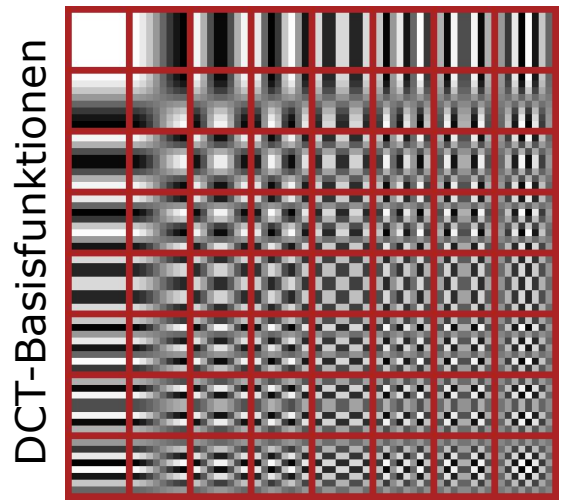
- Animation nur mit Erweiterungen (APNG oder MNG) möglich
  → wenig Unterstützung (bspw. nicht in PowerPoint)
- Alphakanal zum Teil fehlerhaft angezeigt in Browsern
  → Ja-Nein-Transparenzen jedoch möglich
- keine Unterstützung des Ladens von Bildteilen
- verlustfreie Kompression bei Fotos nicht effizient genug
- höhere Kompressionszeiten als bspw. Lauflängenkodierung oder LZW
- PNG unterstützt zwar eingebettete Metainformationen, die aber weder dem EXIF- noch dem IPTC-Standard entsprechen

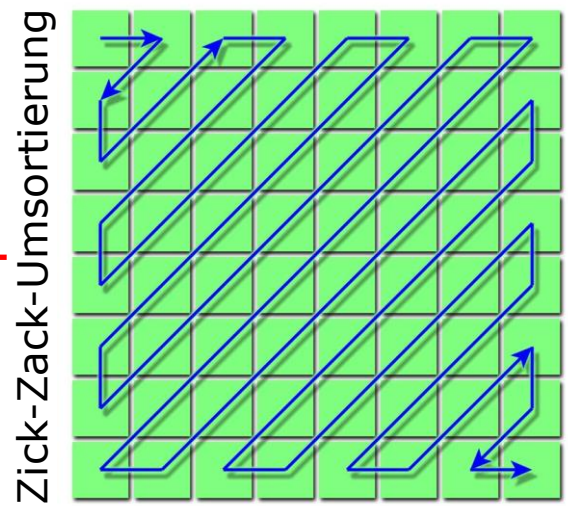
Joint Photographic Experts Group (JPEG)

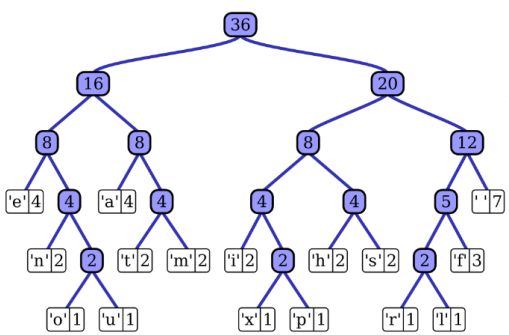
- 1986 gegründetes Gremium der ITU
- Kompression durch verlustbehaftete Verarbeitungsschritte in Zusammenwirken mit Huffman-Kodierung
- Blockbildung, stufige Übergänge, Farbeffekte an Graukeilen
- Nachfolger JPEG-2000 ist wesentlich weniger anfällig für diese Art von Artefakten und kann ICC Profile einbetten

JPEG-Kompression

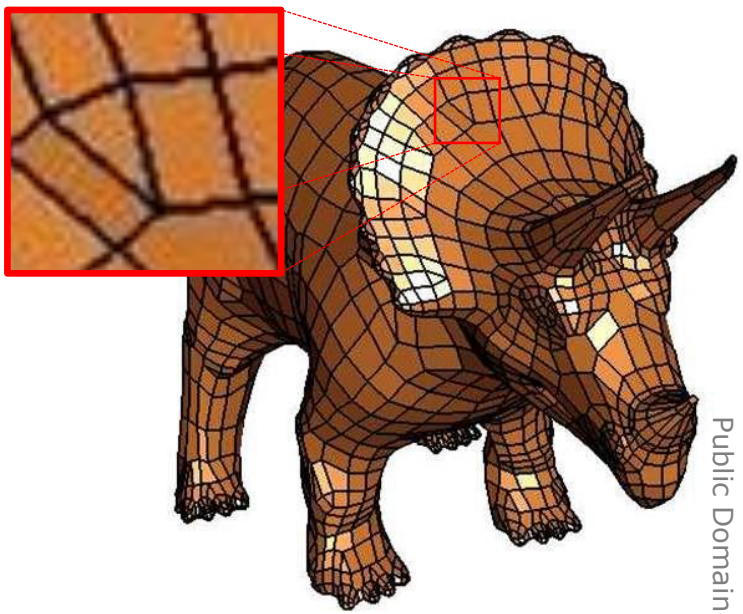
- Farbraumumrechnung in den YUV (IEC 601)
- verlustbehaftete Tiefpassfilterung und Unterabtastung der Farbdifferenzsignale U und V
- Einteilung in 8x8 Blöcke mit diskreter Kosinustransformation (DCT)
- verlustbehaftete Quantisierung
- Umsortierung
- verlustfreie Entropiekodierung (Huffman-Kodierung)







Einsatz von JPEG



Public Domain

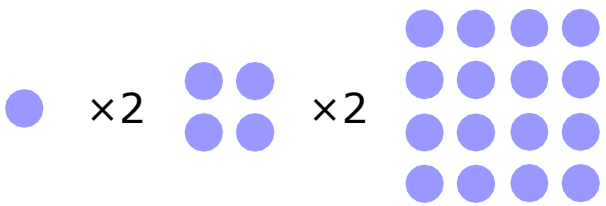
- geeignet für Fotos
- ungeeignet für
  - Strichzeichnungen
  - Schwarzweißbilder
    (nur Schwarz und Weiß, d.h. 1 Bit pro Bildpunkt)
  - gerasterte Bilder (bspw. im Zeitungsdruck)

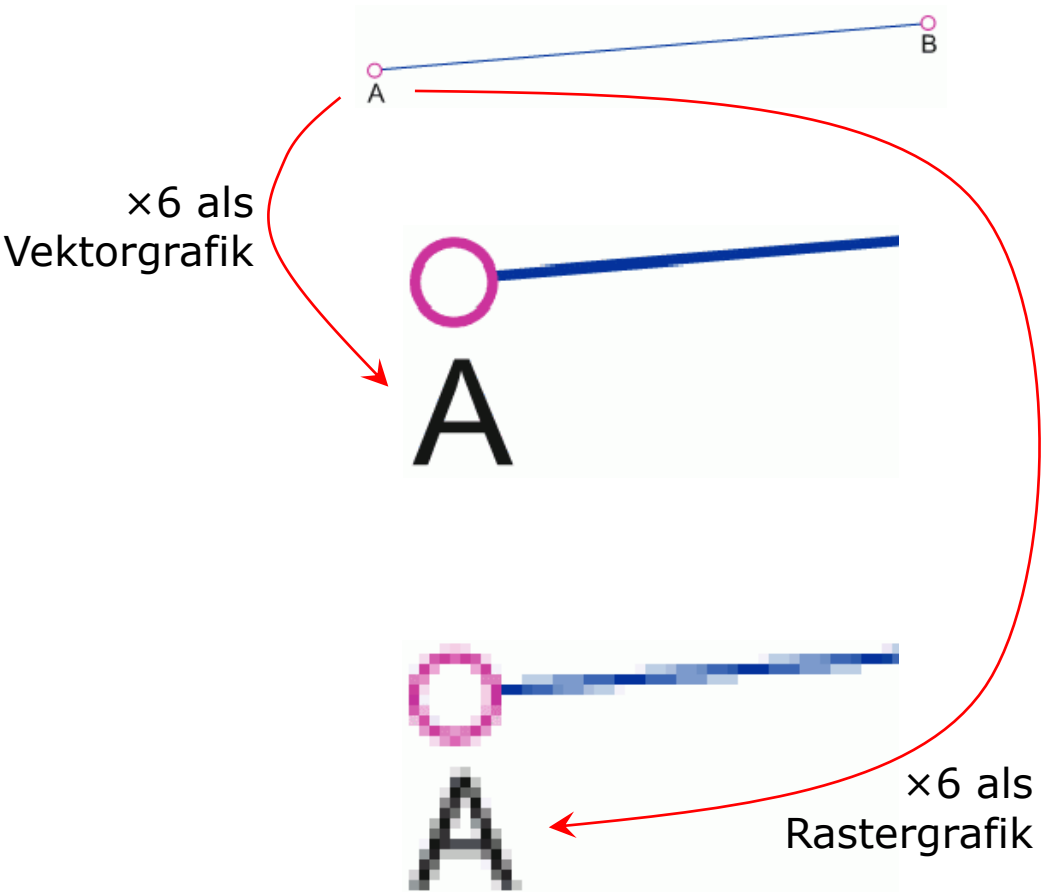
JPEG 2000, JP2

- verlustfreie oder verlustbehaftete Komprimierung
- Format basiert auf XML
- kann Metadaten enthalten, die das Verwalten und Auffinden von Bildern im Internet erleichtern sollen
- Möglichkeit, bestimmten Bildregionen von Interesse in höherer Qualität zu komprimieren und zu dekomprimieren
- Interlacing ähnlich PNG
- Unterstützung in Browsern unterschiedlich/durchwachsen

Bilder: Vektorgrafik und Vektorformate

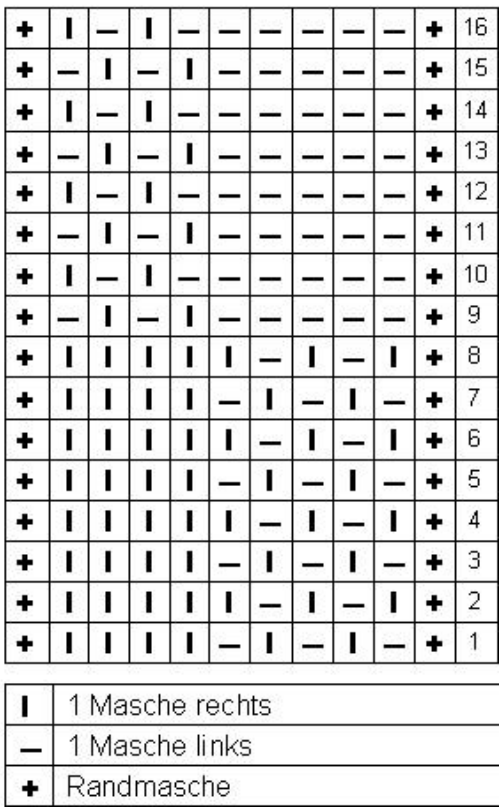
Motivation und Beispiel





BILDBESCHREIBUNG

- beschreibt Grafiken anhand von Zeichenkommandos, ähnlich einer Strickschrift (Beschreibung durch Linien, Kurven und Verhältnisse)
- ermöglicht Verschieben, Skalieren, Umformen und Farbänderung ohne, dass Qualität beeinträchtigt wird
- benötigt wenig Speicherplatz, da nicht Pixel sondern die Konstruktionsweise gespeichert wird
- ab 1970 in CAD Systemen, heute per Scalable Vector Graphic (SVG) in vielen Anwendungen

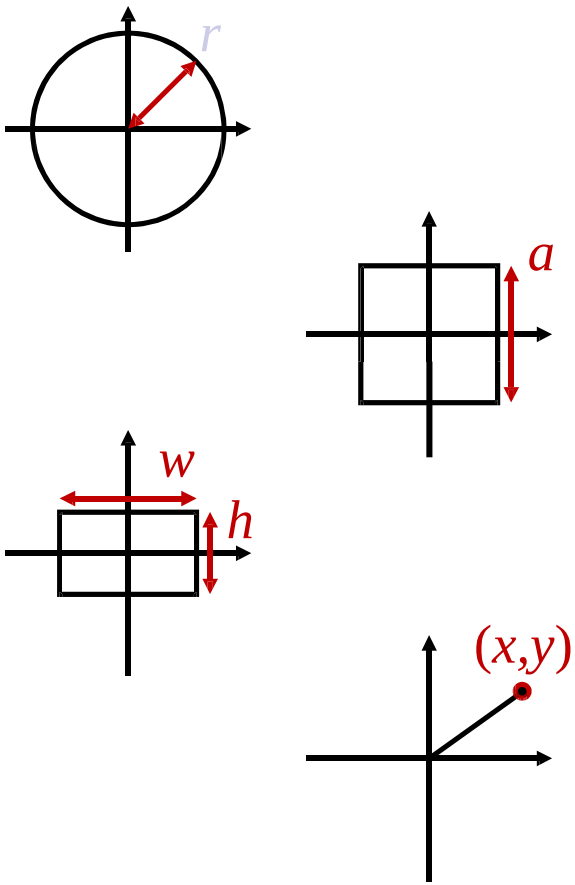


I	1 Masche rechts
–	1 Masche links
+	Randmasche

Strickschrift für den Wollfaden

PARAMETRISIERTE PRIMITIVE

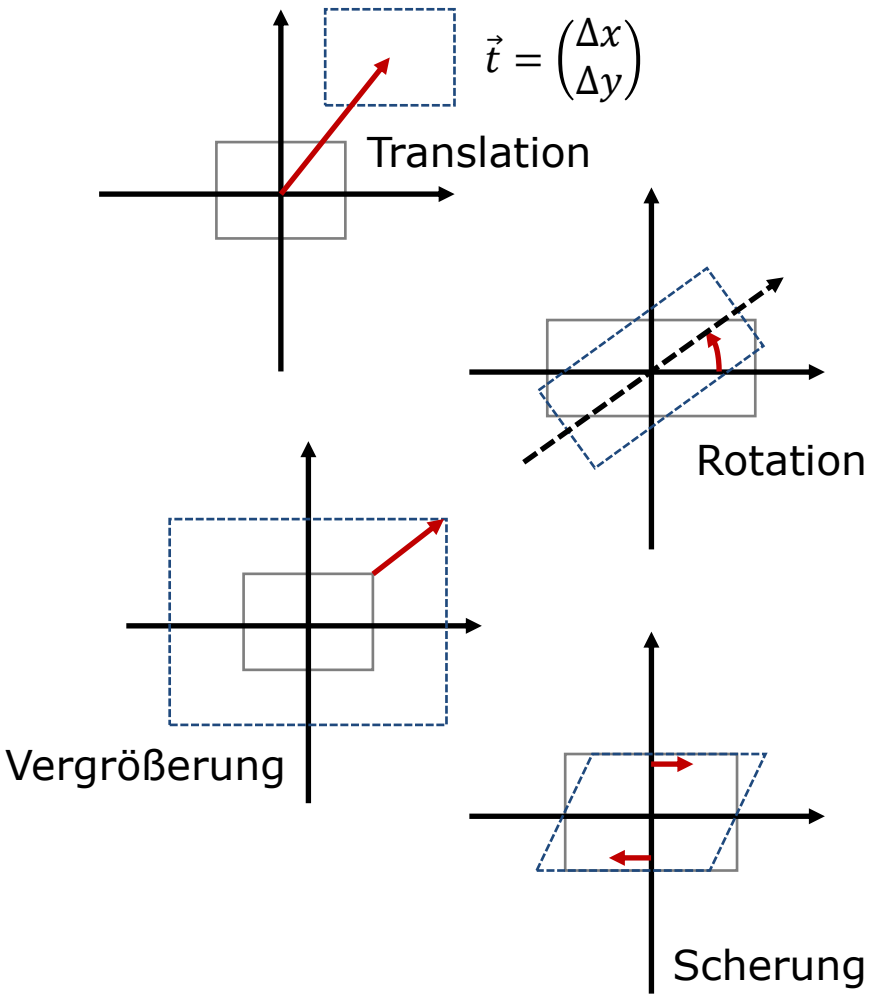
- Kreis im Ursprung ist gegeben durch Radius
- achsenausgerichtetes Quadrat um Ursprung ist definiert durch Kantenlänge
- achsenausgerichtetes Rechteck ist definiert durch Höhe und Breite
- Liniensegment von Ursprung ist definiert durch Position des Endpunktes



TRANSFORMATIONEN

- Transformationen erlauben es die Primitive zu
  - Positionieren (Translation)
  - Skalierung, Spiegelung
  - Drehen (Rotation)
  - Scheren (Scherung)
- bei allen Transformationen können die 2D-Positionen von Punkten auf dem Primitiv mit einer Matrixmultiplikation und einer Vektoraddition umgerechnet werden:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$



TRANSFORMATIONSFORMELN

- Translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

- Scherung

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \vec{0}$$

- Rotation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \vec{0}$$

- Skalierung

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \vec{0}$$

- Spiegelung an der y-Achse

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \vec{0}$$

- Punktspiegelung am Ursprung

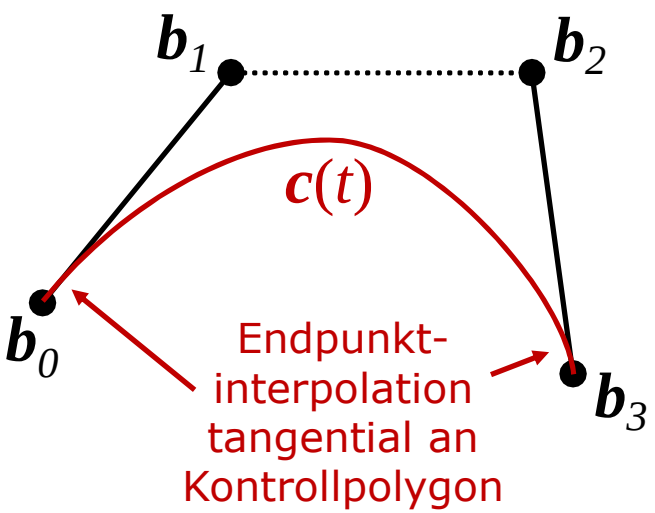
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \vec{0}$$

KURVEN

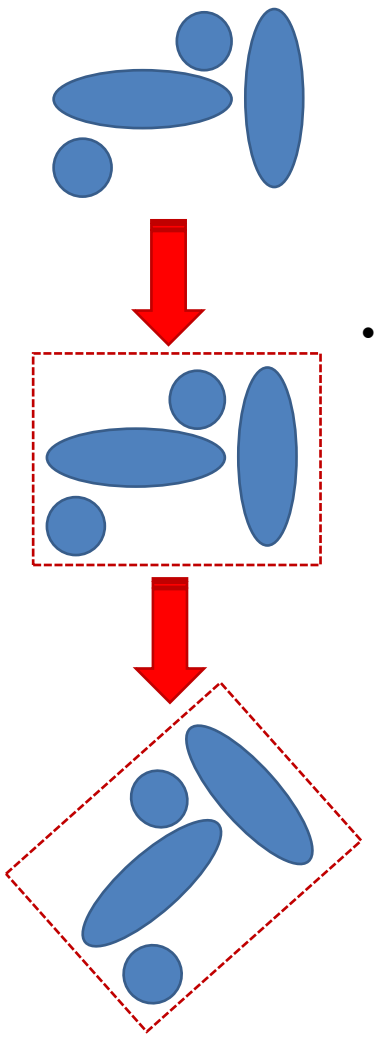
- meistens werden Bezier-Kurven dritten Grades für die Definition von Kurven verwendet
- Kurve durch vier Punkte definiert: b_0, …, b_3
  - b_0 und b_3 definieren Start- und Endpunkt
  - b_1 und b_2 definieren Punkte auf Tangenten durch Start- und Endpunkt

Bezier-Kurve 3.Grades (mit $t \in [0,1]$):

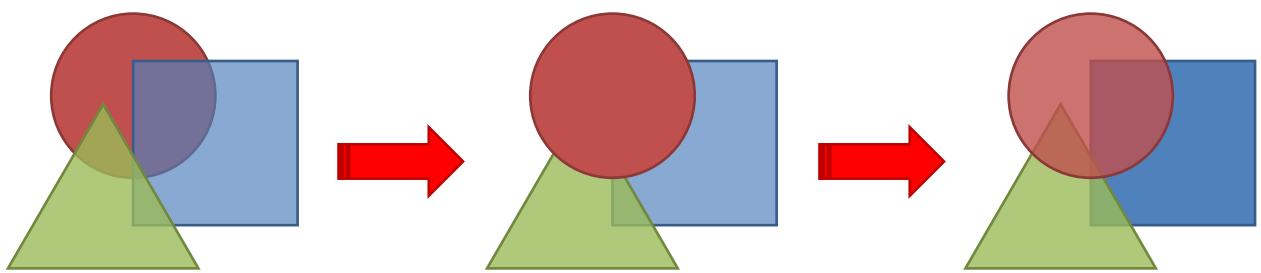
$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = (1-t)^3 \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + 3t(1-t)^2 \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + 3t^2(1-t) \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} + t^3 \begin{pmatrix} x_3 \\ y_3 \end{pmatrix}$$



Gruppierung und Tiefe

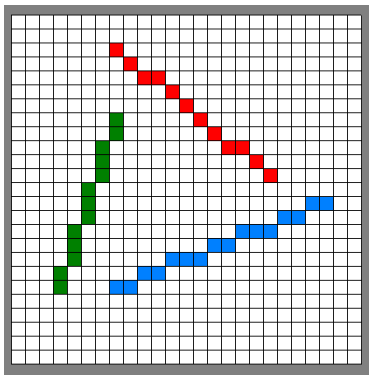


- Gruppierung
  - mit verschiedenen Selektionsmechanismen können mehrere Primitive ausgewählt werden und Gruppen bilden
  - Transformationen wirken dann auf die ganze Gruppe
- Tiefe
  - sich schneidende Primitive erhalten eine Tiefenkoordinate zur Definition, welches Primitiv das andere verdeckt, bzw. in welcher Reihenfolge transparente Primitive zu blenden sind
  - benötigt Transformationen, welche die Tiefe ändern können

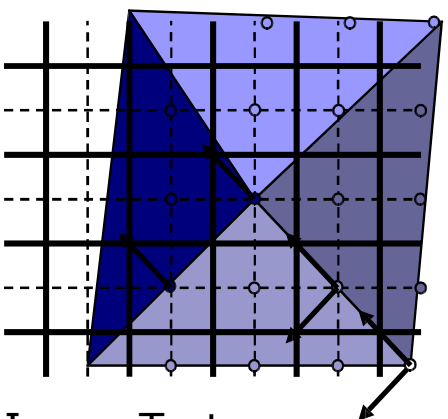


Rasterisierung / Scan-Line Conversion

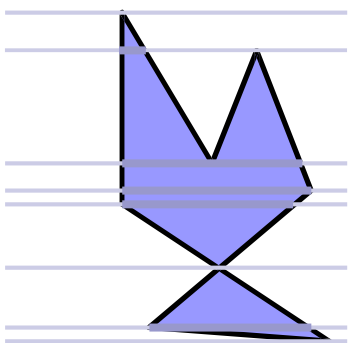
- Umwandlung von einem Vektorprimitiv in ein Rasterbild
- Ansätze
  - Test, ob ein Punkt im Innern des Primitives liegt
  - inkrementell
  - Sweepline



Rasterisieren von Linien

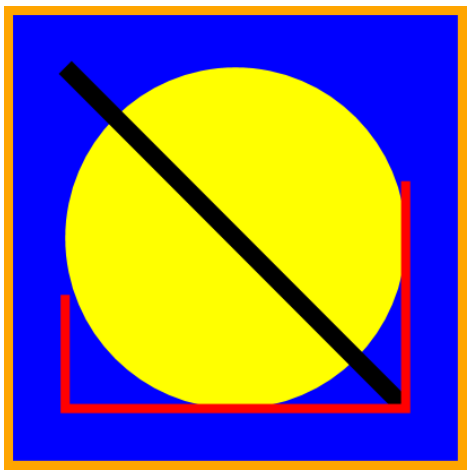


Innen-Test:
Was machen wir mit Pixeln auf Kanten?



Sweepline

Vektorgrafik-Formate: SVG-Beispiel



```
<?xml version="1.0" encoding="UTF-8" ?>
<svg width="391" height="391"
     viewBox="-70.5 -70.5 391 391"
     xmlns="http://www.w3.org/2000/svg">
  <rect x="25" y="25" width="200" height="200"
    fill="blue" stroke-width="4" stroke="orange" />
  <circle cx="125" cy="125" r="75" fill="yellow" />
  <line x1="50" y1="50" x2="198" y2="198"
    stroke="black" stroke-width="8" />
  <polyline points="50,150 50,200 200,200 200,100"
    stroke="red" stroke-width="4" fill="none" />
</svg>
```

Vektorgrafik-Formate: SVG-Funktionalität (1/4)

- Pfade
  - simple und zusammengesetzte
  - gerade oder gebogen
  - gefüllt oder umrissen
  - subtraktiv, additiv oder multiplizierend
- Basisformen
  - einzelne gerade/gebogene Pfade oder zusammengesetzt (Polyline)
  - Kreise, Ellipsen, Rechtecke, Quadrate und Squircle sowie sonstige geschlossene Polygone
- Texte
  - Unicode-Zeichen als XML Character Data
  - visuelle Effekte mittels CSS
  - Unterstützung der gängigen Textrichtungen (LtR, RtL, TRtBL, TLtBR)
  - pfadgebundene Texte (z.B. gebogener Text um Wappen)

VEKTORGRAFIK-FORMATE: SVG-FUNKTIONALITÄT (2/4)

- Füllung
  - Farben, Gradienten, Muster
  - deckend, freistehend oder beliebiger Prozentwert dazwischen
- Dekoration von Linienenden (Pfeilspitzen, Symbole, etc.) (Marker)
- Farben werden wie in CSS2 angegeben
- Gradienten und Muster
  - lineare und radiale Gradienten beliebiger Farbkombinationen
  - Transparenzgradienten sind auch möglich
  - Muster basieren auf vordefinierten Objekten und werden in x- und/oder y-Richtung wiederholt
  - Gradienten und Muster sind animierbar und interaktionsfähig

Vektorgrafik-Formate: SVG-Funktionalität (3/4)

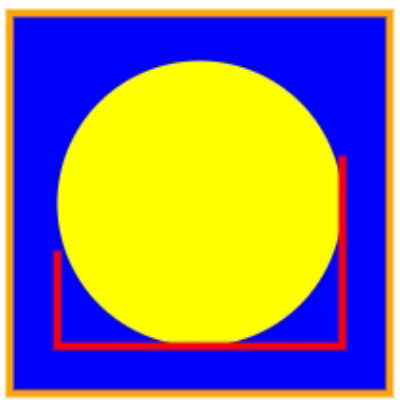
- Interaktivität
  - echte Hyperlinks
  - Event-Handler (Fokuswechsel, Mausklicks, Bildlauf, Zoom, …)
  - Hinzufügen, Starten, Pausieren, Fortsetzen, Stoppen, Entfernen von Objekten, Eigenschaften und Animationen
  - erreicht durch Scripting (SMIL, ECMAScript oder JavaScript)
- Schriftarten
  - wie bei HTML und CSS extern einzubinden
    → Darstellung auf Endgerät weicht ggf. von Erzeugergerät ab
  - in SVG eingebettete Glyphen
    → überall korrekte Darstellung, aber größere Datei
- Metadaten
  - angegeben in `<metadata>`-Elementen
  - folgen den „Dublin Core Metadata“-Eigenschaften
  - zusätzlich: `<title>`- und `<desc>`-Tags

Vektorgrafik-Formate: SVG-Funktionalität (4/4)

- Filtereffekte (SVG Filter Effects)
  - Reihe von Grafikoperationen angewendet auf Vektorgrafik
  - zur Erzeugung einer Rastergrafik
  - Kacheln, Übergänge, Farbmatrizen, Kompositionen, Diffusion, Gauß'sche Unschärfe, …
- Animation
  - `<animate>`, `<animateColor>`, `<animateMotion>`, `<animateTransform>`, …
  - Manipulation des DOM
  - einmalig, kontinuierlich, Schleife, als Reaktion auf Ereignisse

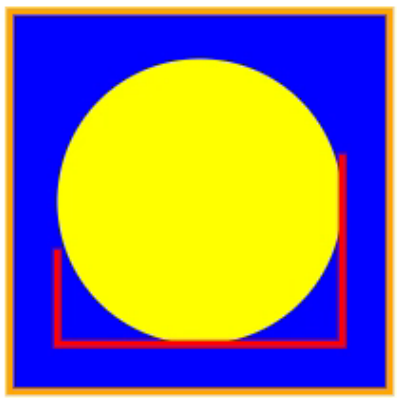
Vektorgrafik-Formate: SVG Animation mit XML

```
<svg width="391" height="391" viewBox="-70.5 -70.5 391 391"
   xmlns="http://www.w3.org/2000/svg">
   <rect x="25" y="25" width="200" height="200" fill="blue"
      stroke-width="4" stroke="orange" />
   <circle cx="125" cy="125" r="75" fill="yellow" />
   <line x1="50" y1="50" x2="198" y2="198" stroke="black"
      stroke-width="8">
      <animateTransform
         attributeName="transform"
         attributeType="XML"
         type="rotate"
         from="0"
         to="360"
         begin="0s"
         dur="1s"
         repeatCount="indefinite“ />
   </line>
   <polyline points="50,150 50,200 200,200 200,100" stroke="red"
      stroke-width="4" fill="none” />
</svg>
```



Vektorgrafik-Formate: SVG Animation mit CSS

```
<svg width="391" height="391" viewBox="-70.5 -70.5 391 391"
   xmlns="http://www.w3.org/2000/svg">
   <style type="text/css">
      @keyframes rot_kf {
         from { transform: rotate(0deg);   }
         to   { transform: rotate(360deg); }
      }
      .rot {
         animation: rot_kf 1s linear infinite;
      }
   </style>
   <rect x="25" y="25" width="200" height="200"
      fill="blue" stroke-width="4" stroke="orange" />
   <circle cx="125" cy="125" r="75" fill="yellow" />
   <line class="rot" x1="50" y1="50" x2="198" y2="198"
      stroke="black" stroke-width="8" />
   <polyline points="50,150 50,200 200,200 200,100"
      stroke="red" stroke-width="4" fill="none" />
</svg>
```



Vektorgrafik-Formate: SVG mit Gauß'scher Unschärfe

```
<?xml version="1.0" encoding="UTF-8" ?>
<svg width="340" height="200"
     xmlns="http://www.w3.org/2000/svg">
     <rect x="7" y="7" width="320" height="180"
          rx="15" style="fill:#cccc33; stroke:black;
               stroke-width:3px;" />
     <text x="30" y="106" style="font-family:verdana;
     font-size:32px; font-weight:bold;">
          feGaussianBlur
     </text>
</svg>
```


```
<?xml version="1.0" encoding="UTF-8" ?>
<svg width="340" height="200"
     xmlns="http://www.w3.org/2000/svg">
     <defs>
          <filter id="f1">
               <feGaussianBlur in="SourceGraphic"
                    stdDeviation="3" />
          </filter>
     </defs>
     <g filter="url(#f1)">
          <rect x="7" y="7" width="320" height="180"
               rx="15" style="fill:#cccc33;
               stroke:black; stroke-width:3px;" />
          <text x="30" y="106"
               style="font-family:verdana;
               font-size:32px; font-weight:bold;">
               feGaussianBlur
          </text>
     </g>
     <filter id="f2">
          <feTurbulence type="fractalNoise"
               baseFrequency=".2" />
     </filter>
</svg>
```

Video/Audio

VIDEO (1/2)

- schnelle Abfolge von Bildern
  - Einzelbilder ab ca. 25 fps nicht mehr unterscheidbar
  - Wahrnehmungsschwelle des Menschen bei ca. 40ms
- i.d.R. mit Ton
  - synchronisiert mit Video
  - im Bereich 50…150ms lippensynchron
  - im Bereich bis ca. 300ms inhaltssynchron

VIDEO (2/2)

- Einzelbilder benötigen viel zu viel Speicherplatz!
  - AVI oft als unkomprimiertes Echtvideo-Rohformat
  - GIF als teilkomprimiertes Meme-Video-Format (als Bildanimation ohne Ton)
- standardisierte Kodierer/Dekodierer-Algorithmen (Codec)
  - H.264 und H.265 weit verbreitet (Kompression bis 1:500)
  - MPEG und MP4 als kommerzielle Codecs (Kompression bis 1:250)
- oft mit Ton-, Kapitel- und Untertitel-Daten in Container-Formaten zusammengefasst, bspw. Matroška/Матрёшка

Einbettung von Videos in HTML

```
<object width="1280" height="720" data="beispiel.mpeg"></object>
```

- wichtige Attribute

`width`, `height`	Dimensionen des Videos (in Pixel)
`data`	URL der Videodatei
`declare`	ob Video bei Bedarf geladen werden soll (`declare`)
`standby`	während des Ladens anzuzeigender Text
`type`	Medientyp (MIME-Type)

- Probleme
  - feste Größe
  - fester Codec
  - kein Streaming
- Problemlösung: HTML5 `<video>`-Tag (nächstes Kapitel)

Ton

- schnelle Abfolge Schwingungszuständen von Schallwellen
  - im Frequenzbereich 16Hz bis 20kHz vom Menschen wahrnehmbar
  - Wahrnehmungsschwelle für Einzeltöne bei ca. 100ms
  - Unterscheidung ca. 270 gleichzeitiger Töne (mind. 1,8Hz Abstand)
- Vollspektrum-Audio benötigt sehr viel Speicherplatz!
  - WAV oft als unkomprimiertes, wellenbasiertes Rohformat
  - MIDI als komprimiertes Abstraktionsformat
- standardisierte Codecs
  - MP3 und FLAC weit verbreitet (Kompression bis 1:20)
  - Spezial-Codecs mit Kompressionsfaktoren jenseits 1:20

Einbettung von Ton in HTML

- ähnlich Video-Einbettung mit `<object>`-Tag, aber sehr unschön
- ähnliche Probleme wie bei Videos
  - feste Rate
  - fester Codec
  - kein Streaming
- Problemlösung: HTML5 `<audio>`-Tag

Video: Formate

Anforderung: Zeitliche Auflösung

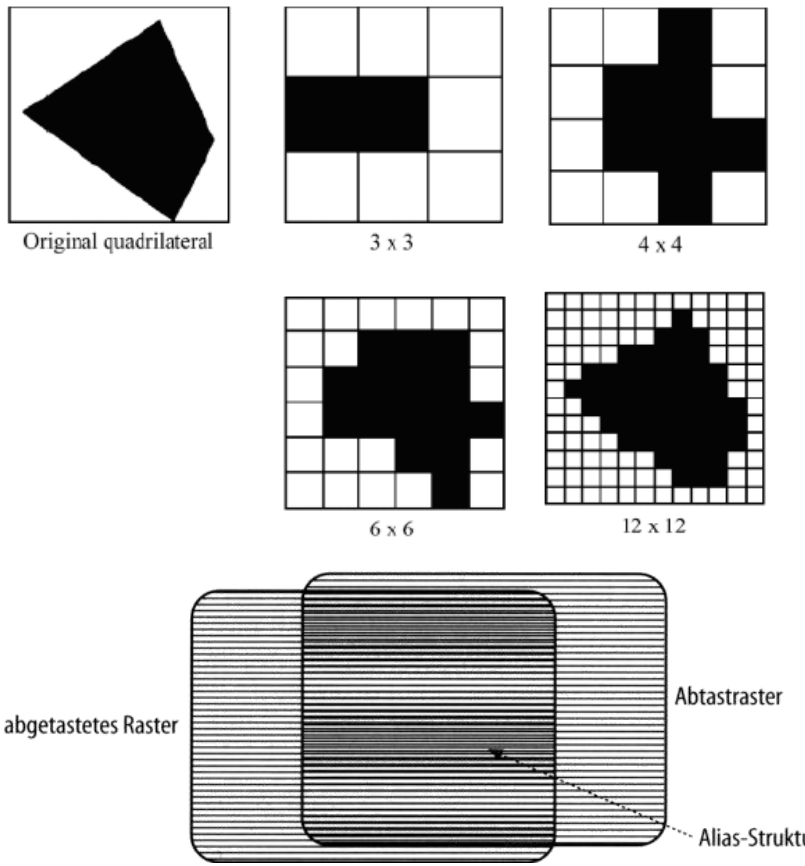
- flüssige Bewegung
  - ab ca. 20 Bilder/Sek.
  - Filmbildfrequenz: 24Hz
- Flimmern
  - durch Dunkelpausen zwischen Bildern wiederholter Erregung und Hemmung der Neuronen
    → unangenehmes Flackern
  - Flimmerverschmelzungs-frequenz bei ca. 50Hz; steigt mit Bildhelligkeit
- Film
  - Bilder 2x projiziert
  - Verdopplung der Dunkelpausen
  - Flimmerfrequenz 48Hz
- Computermonitore
  - hohe Helligkeit, hohe Aufmerksamkeit, großer horizontaler Sehwinkel
    → Bildwiederholfrequenz >72Hz
- TFT/LED/OLED-Displays
  - keine Dunkelphasen
    → kein Flimmern

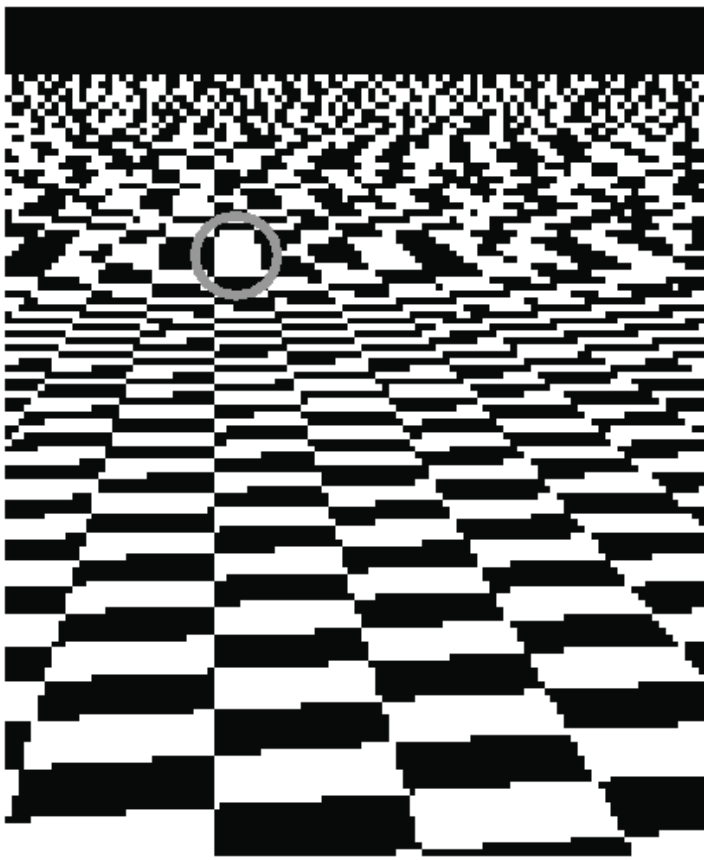
Anforderung: Helligkeitsempfindung unterstützen

- Abhängig vom Adaptionszustand des Auges und der Umfeld-Beleuchtung
- Auge kann sich an Leuchtdichteunterschiede anpassen, dauert aber bis zu 30 Minuten (mondlose Nacht → Sonnen-beschienenes Schneefeld)
- adaptierten Augen können ca. 200 Helligkeitswerte unterscheiden
- Fernsehen
  - Streulichter schränken Kontrastempfindung ein
  - Umfeld-Leuchtdichte sollte ca. 10% der Spitzenleuchtdichte des Bildschirms betragen

ANFORDERUNG: ALIASING VERMEIDEN

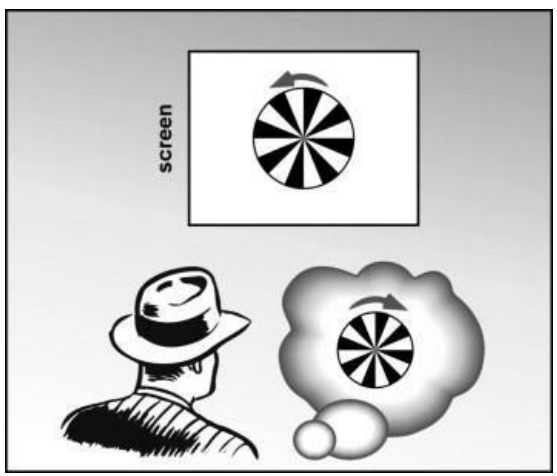
Treppenstufen und Geisterfrequenzen

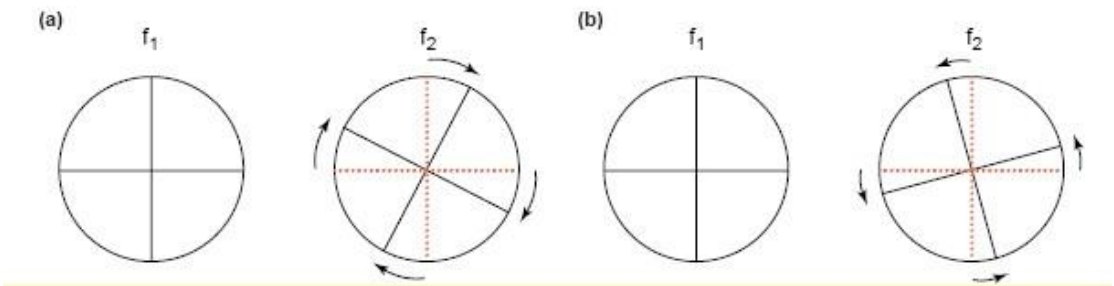




ZEITLICHES ALIASING

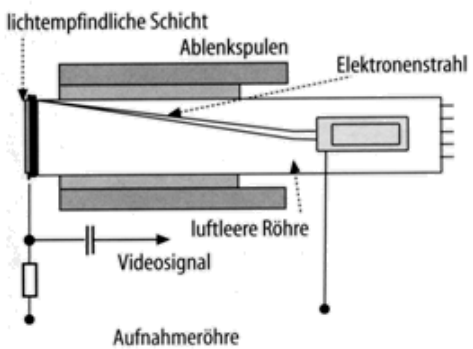
- Wagenrad scheint sich rückwärts zu drehen
  → die Abtastfrequenz zu gering
- Stillstand falls genau um 90° verdreht
- wird vermutlich durch neuronale Adaption verursacht

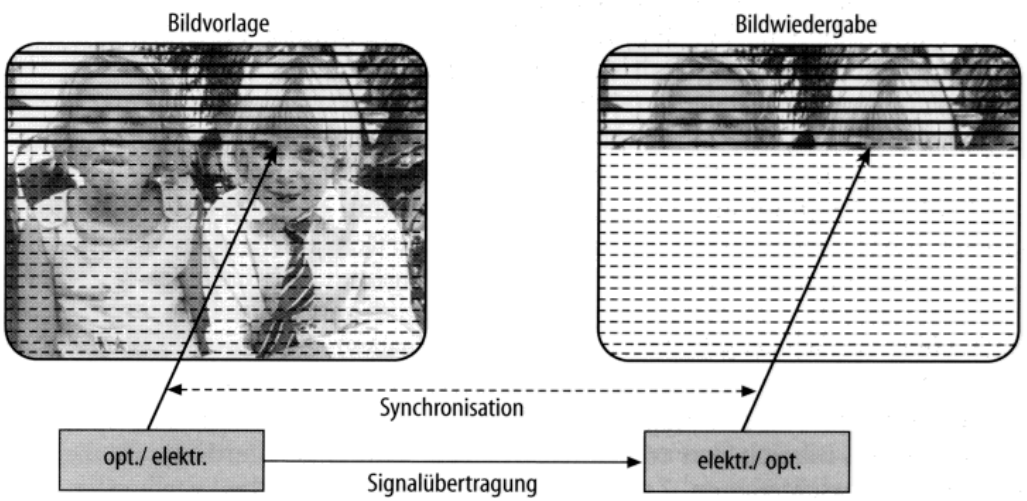




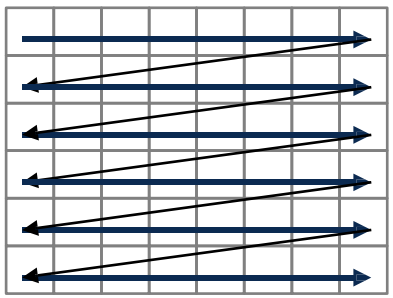
VIDEOSIGNAL (SCHWARZ/WEIß ANALOG):

- zeilenweise Abtastung der Helligkeitswerte
- serielle Übertragung der Bildpunkte
- synchrone Darstellung der Punkte auf Monitor



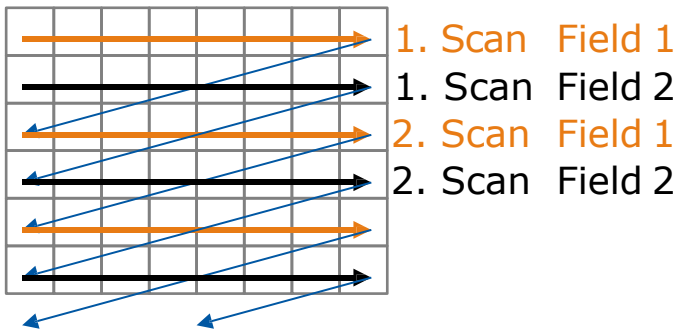


PROGRESSIV/INTERLACED



Progressiv

Vollbilder – Frames



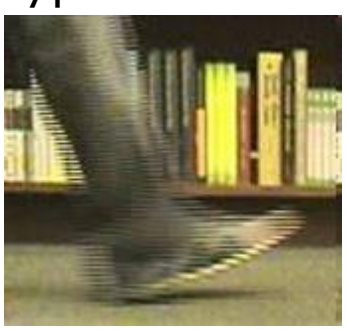
Interlaced

2 Halbbilder

(Fields in doppelter Frequenz)

typische Unschärfe:

typische Artifakte:



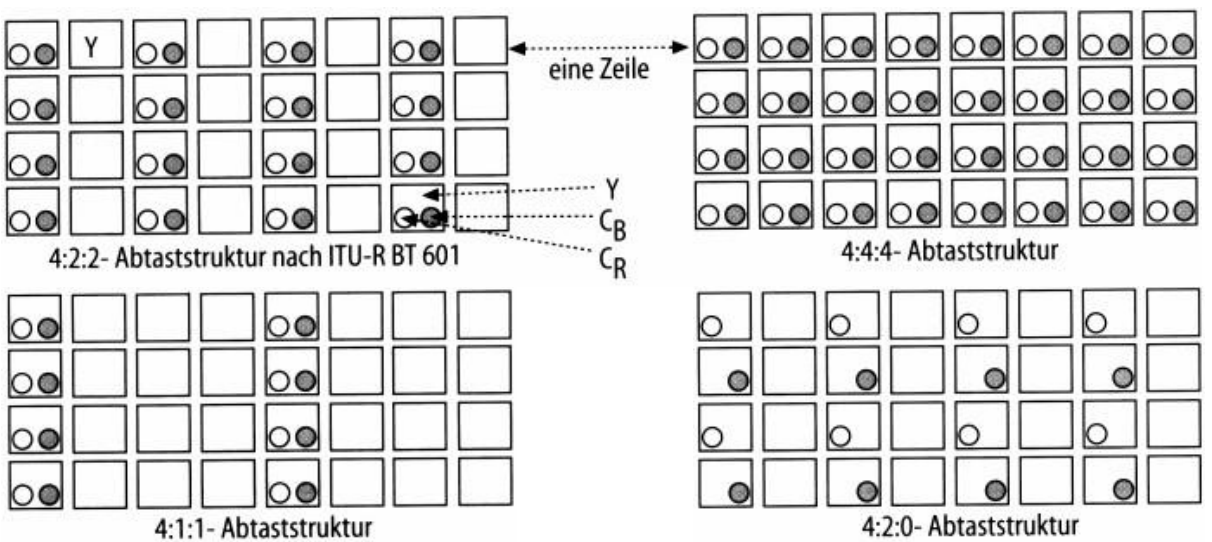
De-Interlacing

- beim Abspielen von TV-Signalen auf 100Hz Röhren bzw. allen LCD/TFT-Panelen und Plasmabildschirmen muss von interlaced auf progressiv umgerechnet werden
- **Weave**
  - gleichzeitige Darstellung der Halbbilder
  - i.O. falls zum gleichen Zeitpunkt aufgenommen
  - bei TV jedoch verschiedene Zeitpunkte → führt zu Kammeffekt
- **Unschärfe**: Weave mit anschließendem Weichzeichnen
- **Bobbing**: jedes Halbbild wird durch Interpolation in Vollbild umgewandelt um so 50Hz zu erreichen
  - jede Zeile mit Nachbarinformationen auffüllen
  - Bild „springt" auf und ab (bobbing), da erste und letzte Zeile jeweils ohne Nachbar
- **Blending**: wie Bobbing aber Mittelung der Halbbilder zu einem Vollbild, das 2mal dargestellt wird
- **Motion Compensation**: wie Weave, ungerade Zeilen werden jedoch an gerade Zeilen angepasst

DIGITALE FORMATE (1/2)

- farbwahrnehmungsbezogenes Chroma-Subsampling $Y:C_R:C_B$
- Y-Komponente wird mit 3,375 MHz abgetastet
- erste Zahl ist ein Vielfaches der Y-Abtastrate
- zweite Zahl ist auch ein Vielfaches der Y-Abtastrate aber beschreibt die Abtastraten für C_R und C_B
- dritte Zahl deutet die räumliche Position an

Haupt-format (CCIR 601)



(NTSC-DV)

nach H.261 (PAL-DV)

DIGITALE FORMATE (2/2)

digitaler Bildstandard CCIR 601/ITU-R BT 601

- 50/60Hz, 13,5MHz Pixel
- 720x625x8Bit, 4:2:2,

Camcorder Standards

- D1/2/3/5/6/9, DCT, Digital Betacam
- DV (Digital Video), DVCpro, DVCAM, DVCAM50, ...

Digital Video Broadcasting

- DVB-S, DVB-C, DVB-T, DVB-H (www.dvb.org)
- Kompression mit MPEG2

Audio Video Interleave (AVI): Containerformat für beliebige Codecs

Motion by JPEG (MJPEG)

- JPEG basierte Kompression der Halb-/Vollbilder
- kein Standard → Probleme beim Austausch

Motion Pictures Expert Group (MPEG)

- weit verbreiteter Standard in verschiedenen Versionen (1, 2, 4, 7, 21)

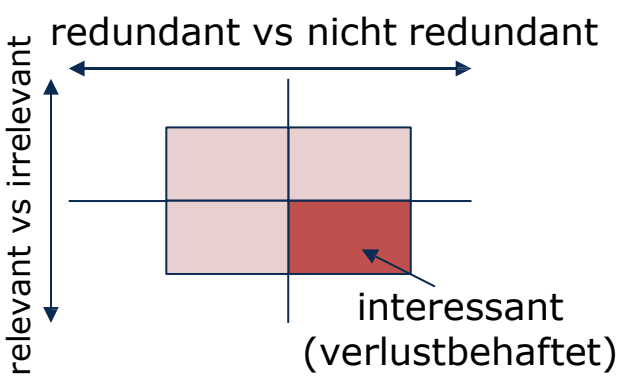
Vergleich der Videoformate

Abkürzung	Name	Auflösung	Seitenverhältnis	physisches Seitenverhältnis	Bildfrequenz	interlaced
SQCIF	Semi QCIF	128 × 96	4:3		29,97	
QCIF	Quarter CIF	176 × 144	4:3	11:9	29,97	
SIF	Source Input Format	352 × 240	4:3, 16:9	11:7,5		
SIF (NTSC)		360 × 240	4:3	3:2	29,97	
SIF (PAL)		360 × 288	4:3	5:4	25	
CIF	Common Intermediate Format	352 × 288	4:3, 16:9	11:9	29,97	
4CIF	2×2 CIF	704 × 576	4:3	11:9	29,97	
9CIF	3×3 CIF	1056 × 864	4:3	11:9	29,97	
16CIF	4×4 CIF	1408 × 1152	4:3	11:9	29,97	
TV		720 × 576	4:3, 16:9	5:4		
PAL	Phase Alternating Line					
D-1 PAL	D-1 PAL	768 × 576	4:3	1:1	25	ja
(PAL)		352 × 576	4:3, 16:9	11:18	25	ja
(PAL)		480 × 576	4:3, 16:9	5:6	25	ja
(PAL)		544 × 576	4:3, 16:9	17:18	25	ja
PAL-Wide	PAL Widescreen	1024 × 576	16:9		25	ja
NTSC	National Television Standards Committee	720 × 480	4:3, 16:9	3:2	29,97	ja
D-1 NTSC	D-1 NTSC	720 × 540	4:3		29,97	ja
(NTSC)		352 × 480	4:3, 16:9	11:15	29,97	ja
(NTSC)		480 × 480	4:3, 16:9	1:1	29,97	ja
(NTSC)		640 × 480	4:3		29,97	ja
(NTSC)		544 × 480	4:3, 16:9	17:15	29,97	ja
HDTV 720	High Definition Television 720 lines	1280 × 720	16:9		50/60	nein
HDTV 1035	HDTV 1035 lines	1920 × 1035	16:9	> 1,855	50/60	nein
HDTV 1080	HDTV 1080 lines	1920 × 1080	16:9		50/60	nein
HDTV 1152	HDTV 1152 lines	1440 × 1152		5:4	50/60	nein
2K		2048 × 1080		> 17:9		
		2048 × 1536		4:3		
DCI 2K	Digital Cinema Initiatives 2000	2048 × 1556		< 4:3		
		3840 × 2160	16:9			
4K		4096 × 2160		> 17:9		
DCI 4K	DCI 4000	4096 × 3112		< 4:3		
UHDV	Ultra High Definition Video	7680 × 4320	16:9		60	nein

Kodierung und Komprimierung

Redundanz-Reduktion

- Verzicht auf mehrfach vorhandene Information(en)
- auf Signalstatistik beruhend
- i.d.R. verlustfrei
- Beschreibung von Gruppeneigenschaften (Muster)

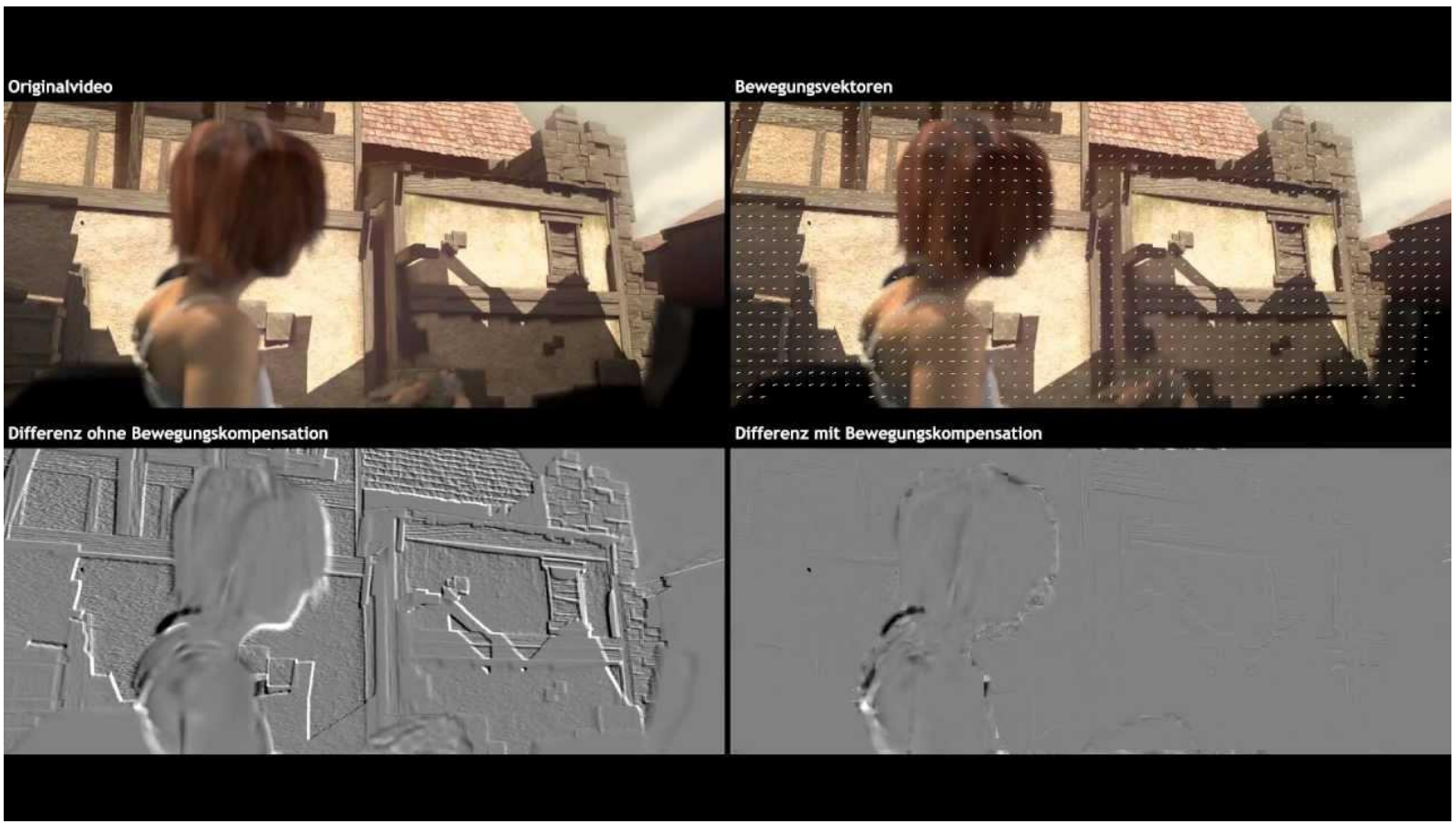


Irrelevanz-Reduktion

- Verzicht auf kaum erkennbare Signalanteile
- auf Wahrnehmungsphysiologie beruhend
- verlustbehaftet

KODIERUNG

Wie bei Audiodaten, ist es auch bei Videodaten sinnvoll, das nächste Bild aus den vorigen Bildern vorherzusagen und nur die Differenz zur Vorhersage zu kodieren.



H.261 und H.263 (1/2)

H.261 (1984-1990)

- Videokompression für Video-Telefonie über ISDN-Kanalbündelung (je Kanal 64 kBit/s)
- weniger als 150ms Verzögerung durch Kompression und Dekompression
- Videoformat: CIF/QCIF, YC_RC_B 4:2:0
- Kompression auf 1,5 MBit/s ≈ 24 gebündelte ISDN-Kanäle

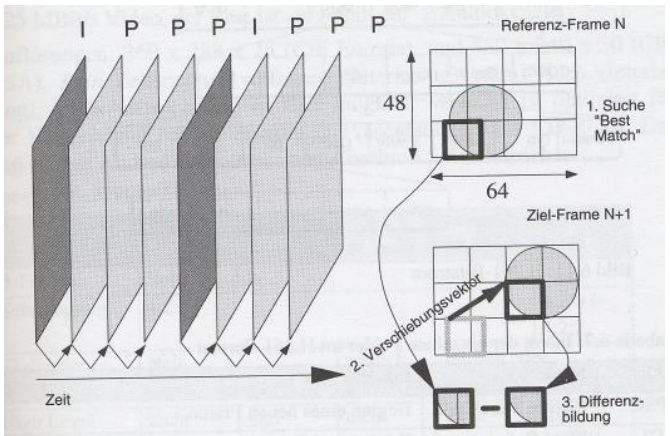
H.263

- Effizienzsteigerung durch
  - Weglassen unwichtiger Teile und
  - Verringerung der Bittiefe von Zusatzinformation
- ergibt ungefähr halbe Datenrate

H.261 und H.263 (2/2)

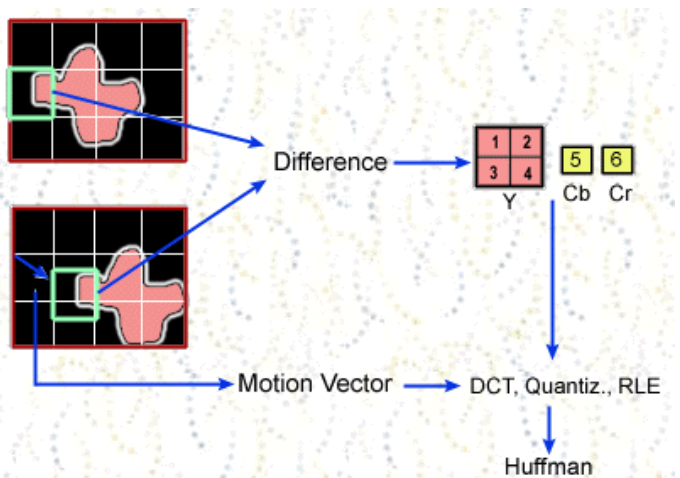
I
Intra-Frames (JPEG)

- Unterteilung in 16x16 px große Blöcke
- darin vier Subblöcke für Helligkeit
- zwei Blöcke für Chroma
- DCT, anschließend Huffman



P
Predicted Frames

- Unterteilung in 16x16 px Blöcke
- für jeden Block Suche nach „Best Match“ in vorherigem dekodierten Frame
- dann Differenzkodierung



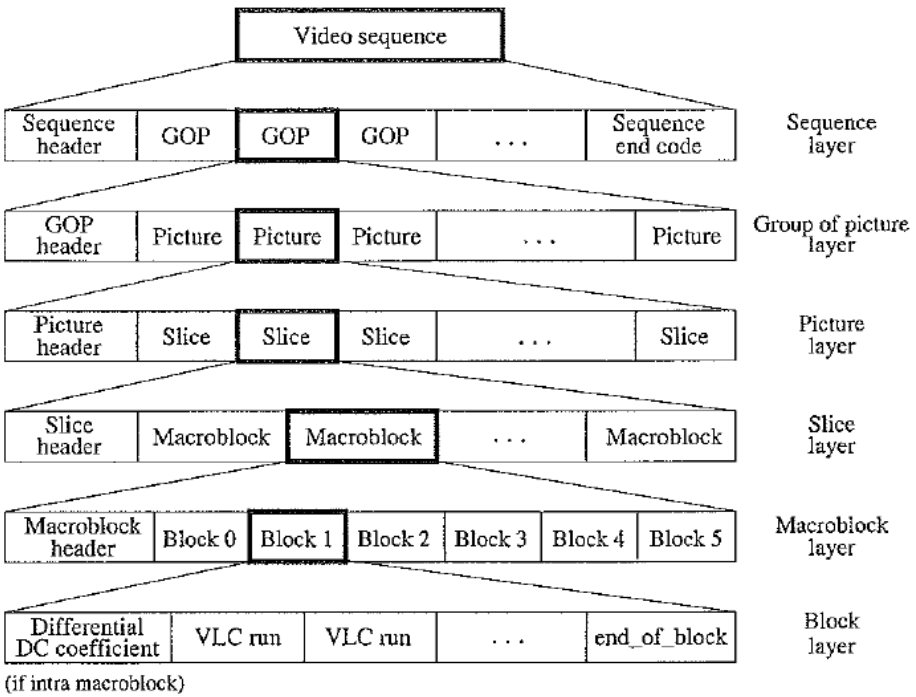
Moving Pictures Experts Group (MPEG)

weit verbreitete Standards für die komprimierte Speicherung von digitalem Video mit Audio

- MPEG-1 (1992): Video auf Speichermedien (nur progressiv)
- MPEG-2 (1993): MPEG-1 und TV sowie HDTV
- MPEG-3 (1993): HDTV (in MPEG-2 aufgenommen)
- MPEG-4 (1998): Interaktives Multimedia, Audiovisuelle Objekte
  (seit 1998 in 33 Parts laufend erweitert, zuletzt 2015; enthält auch MPEG-4 AVC in Part 10 (H.264))
- MPEG-7 (2001): MM Description Interface
- MPEG-21 (2003): Digital Multimedia Framework
- MPEG-H (2013): High Efficiency Coding and Media Delivery in Heterogeneous Environments
  (enthält in Part 2 MPEG-4 HEVC (H.265); zuletzt 2019 aktualisiert)
- MPEG-I (2021): Coded Representation of Immersive Media
  (enthält in Part 2 MPEG-4 VVC (H.266))

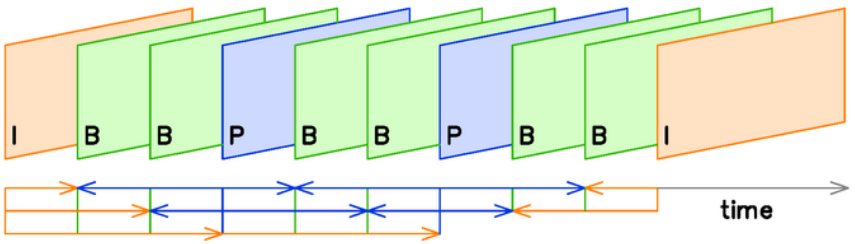
VOKABULAR

- **Sequence**:
  alle Pictures vom *Sequence Header* bis zum *Sequence End Code*
- **Group of Pictures** (GOP):
  alle Pictures zwischen 2 GOP-Headern (12/25 bzw. 15/30 fps)
- **Picture**:
  Field (Halbbild) oder Frame (Vollbild)
- **Macroblock**:
  16x16 Pixel
- **Slice**:
  16 Zeilen x (n x 16 Pixel)
- **Block**:
  8x8 Elemente (Luminanz, Chrominanz oder DCT-Koeffizient)
- **Sample**:
  einzelner Helligkeits- oder Farbwert eines Pixels



MPEG-1 (H.262)

- Zusätzlich zu I- und P-Frames werden B-Frames eingeführt, die bidirektional vorhergesagt werden
- I-Frames wieder nach JPEG-Prinzip
- P-Frames werden aus Vorgänger I/P-Frame wie bei H.261 kodiert
- B-Frames werden aus Vorgänger I/P-Frame und übernächstem Nachfolger B/P-Frame oder Nachfolger I-Frame kodiert (Zweirichtungsdeltakodierung)
- dadurch Umsortierung bei Streaming notwendig



Darstellungsreihenfolge													
Bild	1	2	3	4	5	6	7	8	9	10	11	12	13
Typ	I	B	B	P	B	B	P	B	B	P	B	B	I
Übertragungsreihenfolge													
Bild	1	4	2	3	7	5	6	10	8	9	13	11	12
Typ	I	P	B	B	P	B	B	P	B	B	I	B	B
	GOP				GOP			GOP			GOP		

MPEG-2

Unterschiede zu MPEG-1

- Best Match-Suche nicht nur nach Frames, sondern auch nach Fields
- Makroblöcke mit Farb-Subsampling von 4:2:2 oder 4:4:4
- Frame-Größe bis zu 16.383x16.383 Pixel
- nichtlineare Quantisierungstabelle
- verschiedene Profile
  - räumliche Skalierbarkeit bedeutet, dass derselbe Datenstrom in verschiedenen Auflösungen angezeigt werden kann
  - zeitliche Skalierbarkeit bedeutet, dass der Datenstrom adaptiv auf Veränderungen der Bandbreite reagiert

Überblick der Standards

	MPEG-1	MPEG-2	MPEG-4
Haupt-anwendung	CD-ROM, Digital Storage Media	Broadcast, DVD, HDTV	Webauthoring, Multimedia, Compression, Wireless Mediaphone
Video-Bitraten	1,5 Mbps	4,6 Mbps	20 Kbps - 6 Mbps
Bildauflösung	352x240 720x480	176x144 352x288 1280x720	176x144 352x288 720x480 1280x720
Audio-Qualität	Stereo CD	Surround Sound	Sprache, Musik, CD, HD-Surround

H.264

H.264 (MPEG-4 AVC)

- Integertransformation statt DCT auf 8×8-Blöcken nun auf 4×4-Blöcken
- neben Huffman-Kodierung auch leistungsfähigere arithm. Kodierung
- Makroblöcke in unterschiedliche große Subblöcke unterteilbar
  → eigener Bewegungsvektor für jeden Subblock
  → schärfer abgegrenzte, komplexe Bewegungen besser kodierbar
- Bildvorhersage auch innerhalb von I-Frames möglich
- P- und B-Frames können auf bis zu 16 vorhergehende (beliebig weit zurückliegende) Frames referenzieren (Long-Term Prediction)
  → effizientere Kodierung von periodischen Bewegungen

H.265 UND H.266

H.265 (HEVC, MPEG-4 VVC)

- variable Block- und Macroblock-Größe
- bessere Kodierung von Transformationen

H.266 (Versatile Video Coding)

- entwickelt vom JVET (vereinigtes Video Experten-Team von MPEG & ITU)
- seit 6. Juli 2020 finalisierter Standard
- bei UHD-Auflösung (4K & 8K) bei gleicher Bildqualität Datenreduktion bis 50% im Vergleich zu H.265 (HEVC)
- offizielle Unterstützung von 360°/Panorama-Videos
- Unterstützung komplexer Farbräume und HDR