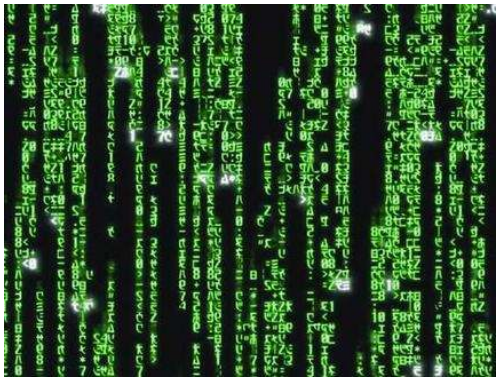
Computergraphik I

Einführung & Displays

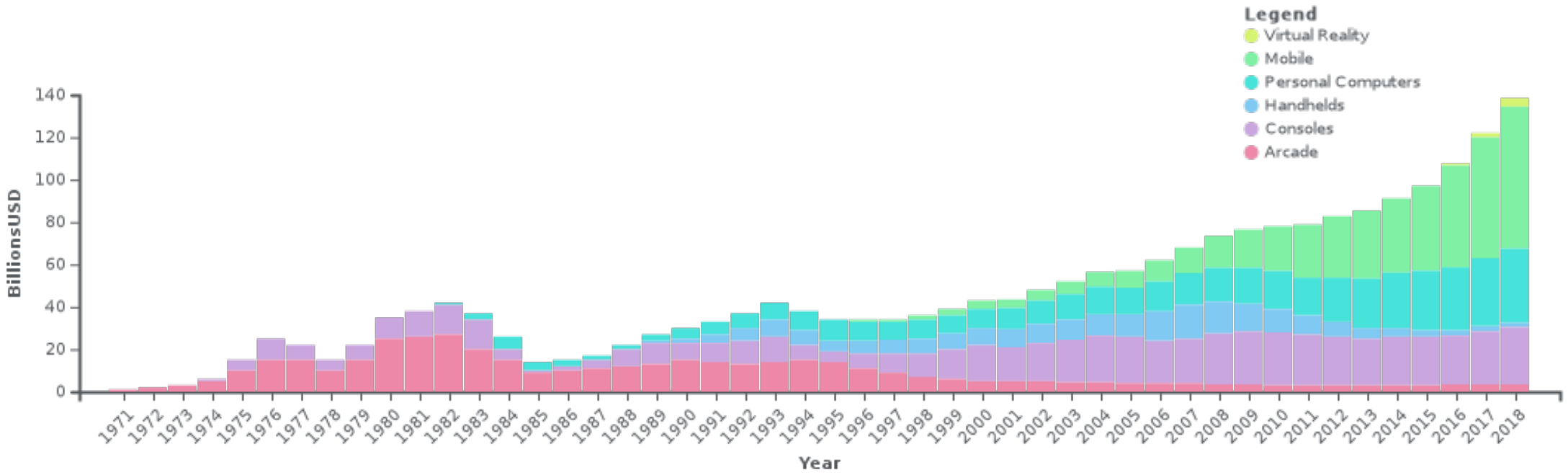


G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

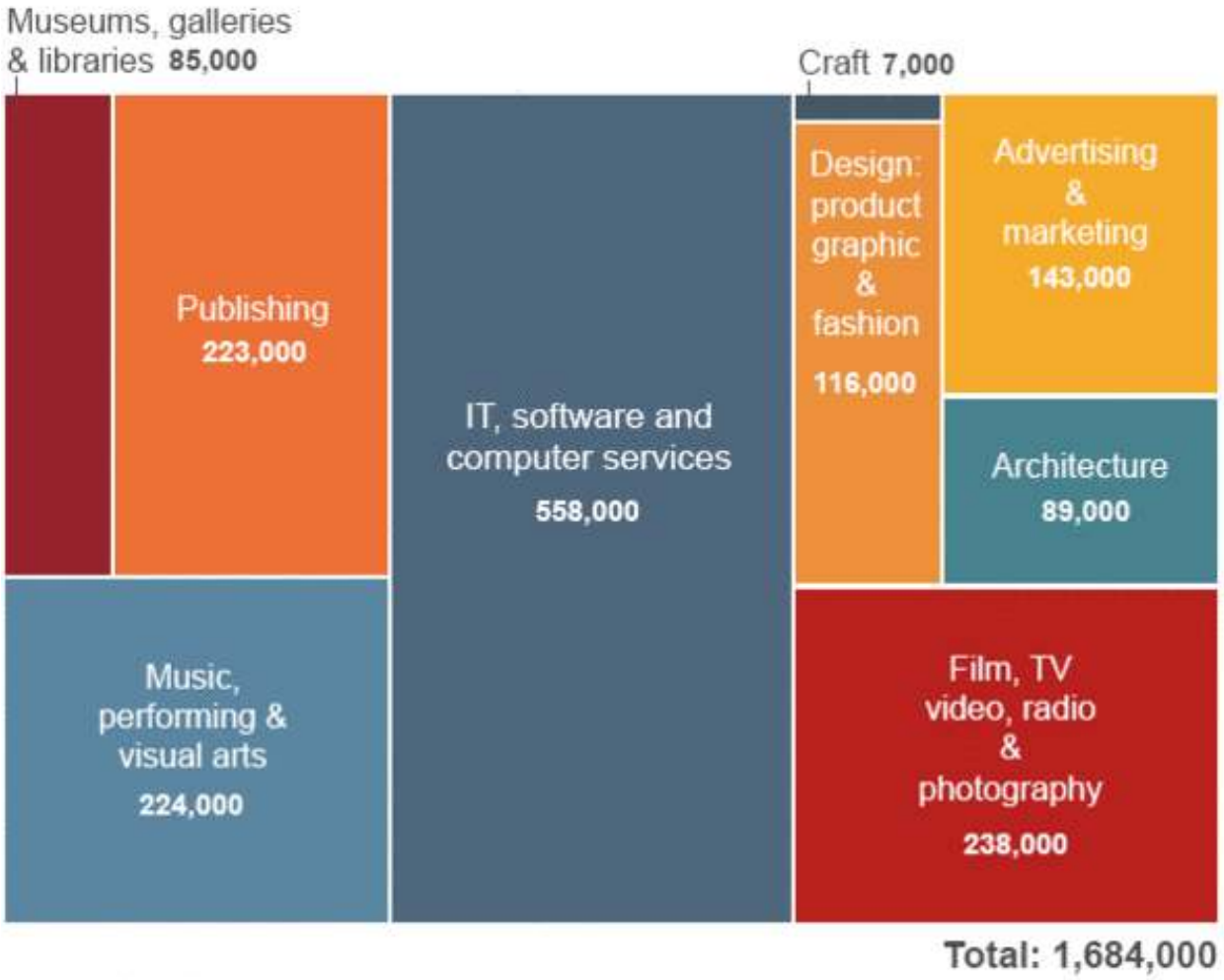
Fragen-Kanal für dieses Kapitel

Oder:
www.menti.com/
qyragw8ezp

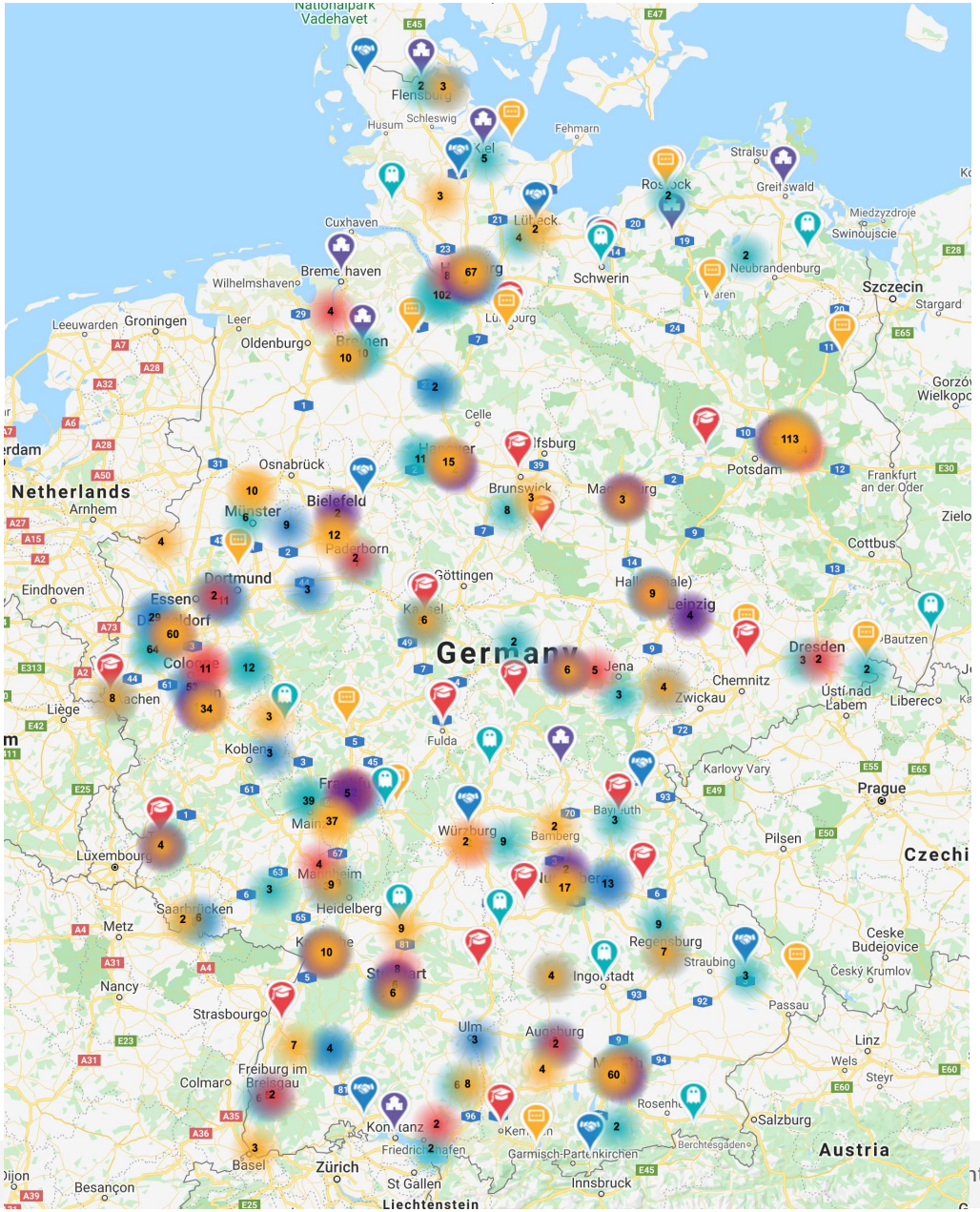
Development of Games Market



Employment in the creative industries in the UK in 2012



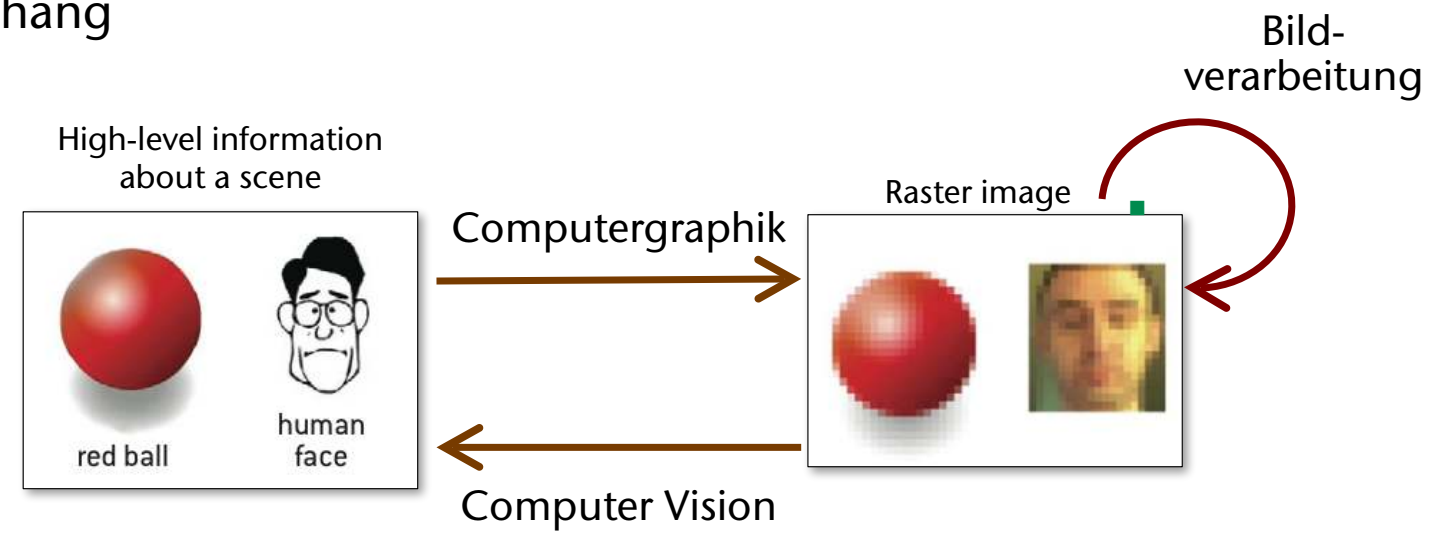
Games Map



https://www.gamesmap.de

Das Gebiet "*Visual Computing*"

- *Visual Computing* = Informatik-Disziplinen "mit Bildern"
- Computergraphik, Maschinelles Sehen und Bildverarbeitung stehen in einem engen Zusammenhang



- Trend: Computergraphik und Computer Vision wachsen immer stärker zusammen ("ProCams")

Anwendungsgebiete der Computergraphik

- Videospiele
- Filme
  - Zeichentrickfilme
  - Computeranimationsfilme
  - Spezialeffekte
- CAD / CAM
- Simulationen
- Medizinische Visualisierung
- Visualisierung von Informationen
- Training (Flug-, Fahr-, Operationssimulator)

Kinofilme

Pixar: Monster's Inc.

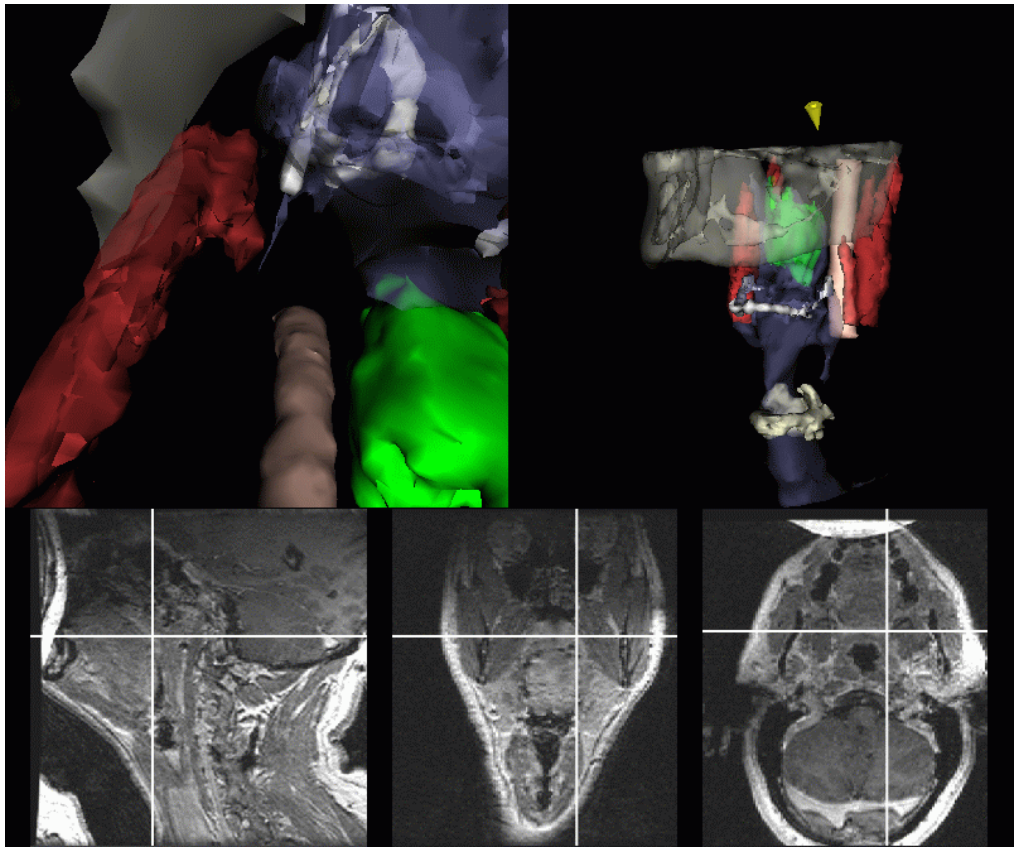
Square: Final Fantasy

Spiele

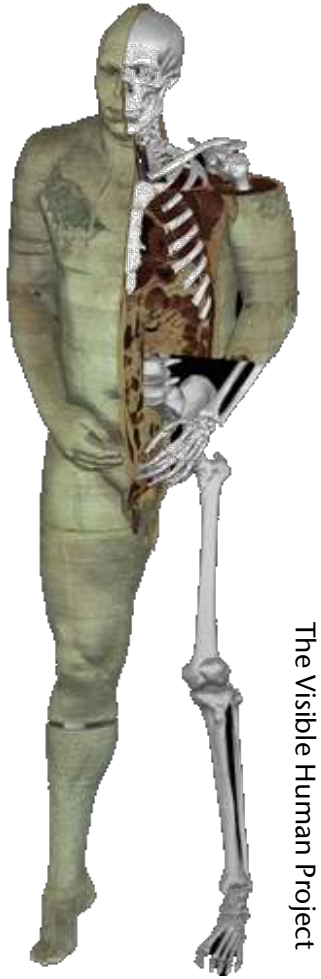

GAMING-UNIVERSE.DE
GRAN TURISMO HD
GAMING-UNIVERSE.DE
GRAN TURISMO HD


SUBARU
GAMING-UNIVERSE.DE
GRAN TURISMO HD
Position
3
Lap
1 / 2
Best Lap
Last Lap
2'08.000
1
63
GAMING-UNIVERSE.DE
GRAN TURISMO HD

Medizinische Visualisierung

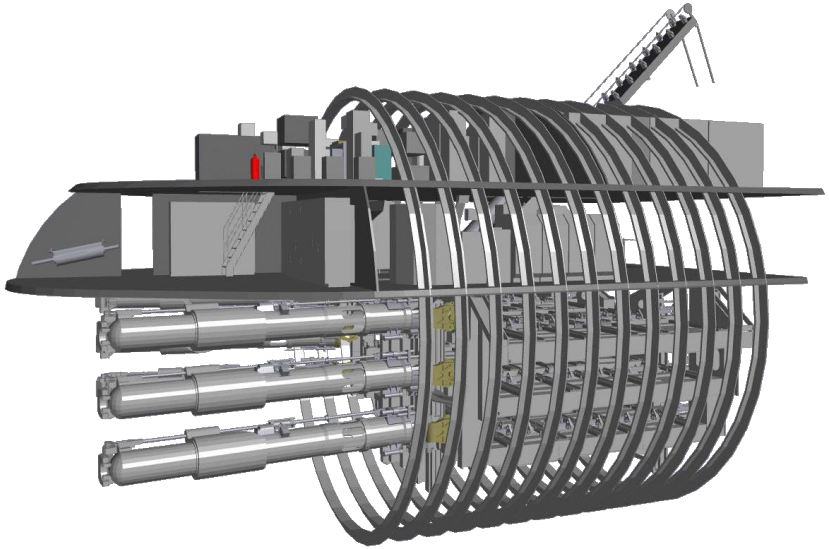


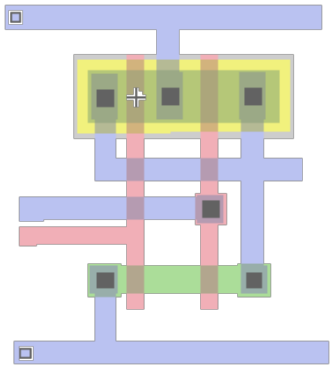
MIT: Image-Guided Surgery Project



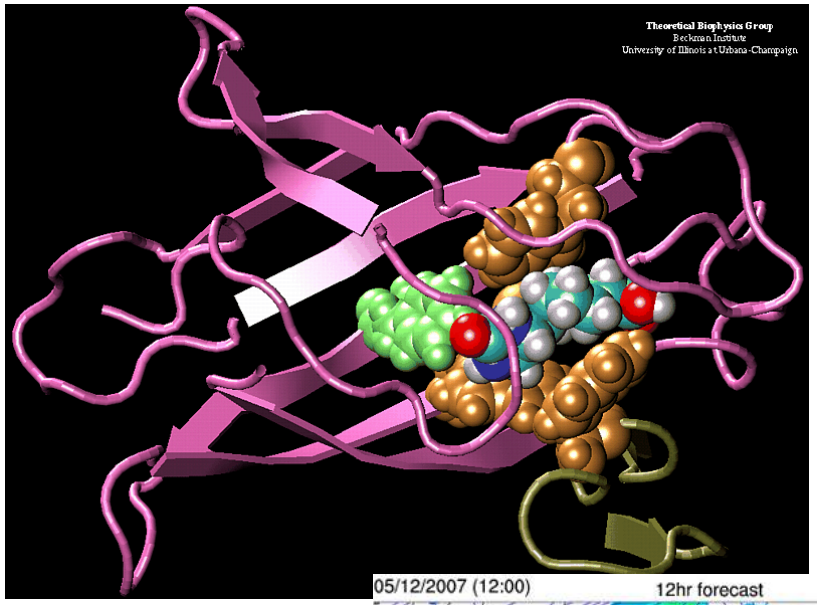
The Visible Human Project

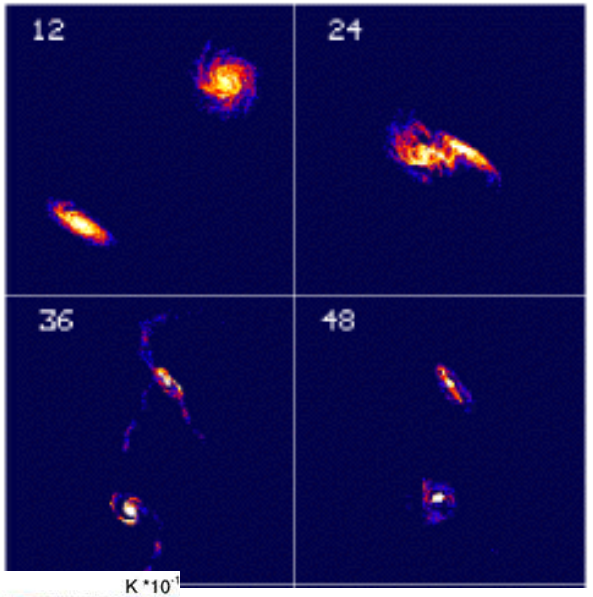
Computer Aided Design (CAD)

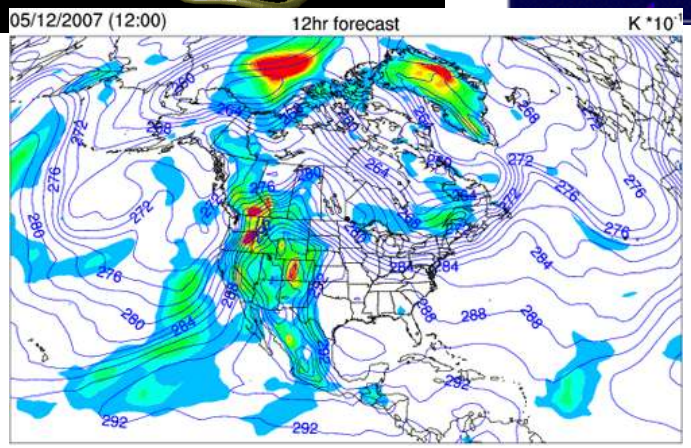




Visualisierung wissenschaftlicher Daten







Trainingssimulatoren

immersivetechnologies.com

campar.in.tum.de

riohondo.edu

news.erau.edu / Embry-Riddle

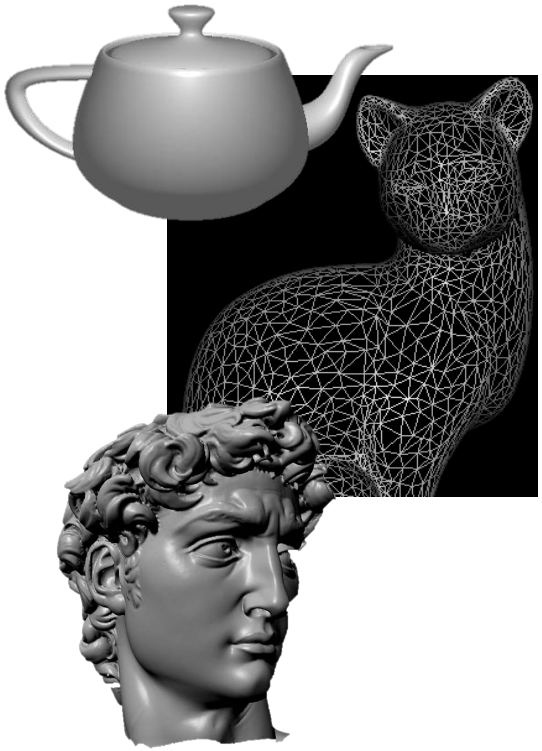
Was ist Computergraphik?

Welche (Teil-)Aufgaben gibt es hierbei zu lösen?

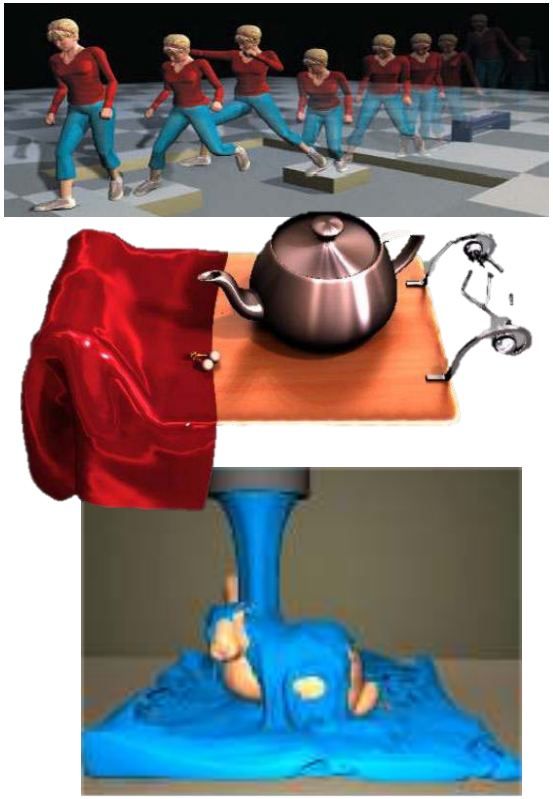
Oder:
www.menti.com/
qyragw8ezp

Die Computergraphik-Prozesskette

Modelling (Beschreibung/Processing der 3D Geometrie)



Animation & Simulation



Rendering (Wiedergabe, inkl. Shading, Lighting, Materials)

Teilgebiete der Computergraphik

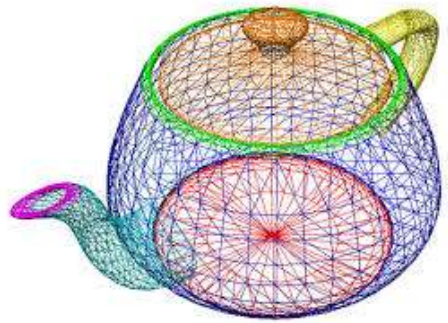
- Die wichtigsten Gebiete:
  - Modellierung
    - Festlegen der Form und Wirkung des äußeren Erscheinungsbildes
    - Interne Datenstrukturen für Objekte, und die ganze Szene, Mesh Processing
  - Rendering
    - Erzeugung des 2D Bildes aus einem 3D Modell
    - Berechnung der Ausbreitung des Lichtes von den Lichtquellen bis ins Auge
  - Animation / Simulation
    - Bewegung der Bilder
    - Implementierung der physikalischen Gesetzte (z.B. klassische Mechanik)
- Weitere Gebiete:
  - Interaktion mit dem Anwender (Human-Computer Interaction - HCI)
  - Virtual Reality (VR)
  - Visualisierung (*scientific / information visualization*)

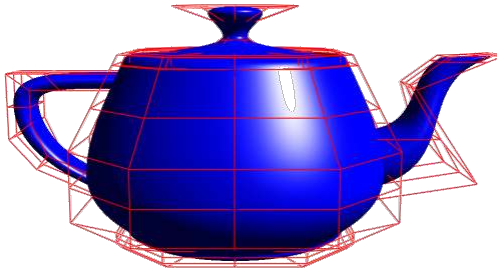
Fragestellungen

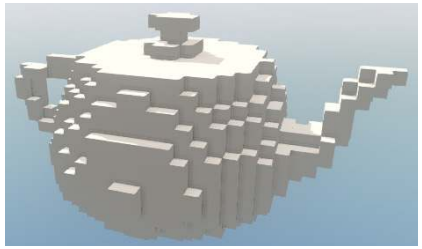
- Wie beschreibt man Objekte einer Szene?
- Wie stellt man die ganze Szene und die einzelnen Objekte schnell dar?
- Was ist mit Lichtquellen?
- Wie erzeugt man Schatten? ... Verdeckungen? ... Tiefeneindruck?
- Was macht man bei "rauhen" Oberflächen?
- Was ist mit kleinen Partikeln wie Nebel, Rauch, Dunst, ... ?
- Physik?
- Animation?

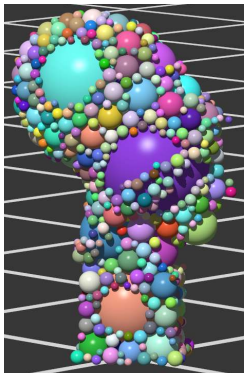
Modellierung – Repräsentation von 3D-Obj.en

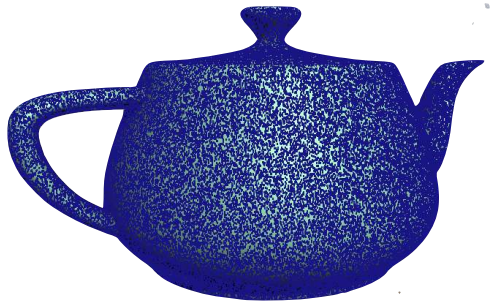
- Boundary Representations:
  - Meshes / "polygon soups"
  - Spline-Surfaces (B-Spline, Bezier, etc.)
- Point Clouds
  - Input von Laser Scanners, Kinect, etc.
- Volumen-Repräsentationen:
  - Voxel
  - Kugeln
- U.v.m.





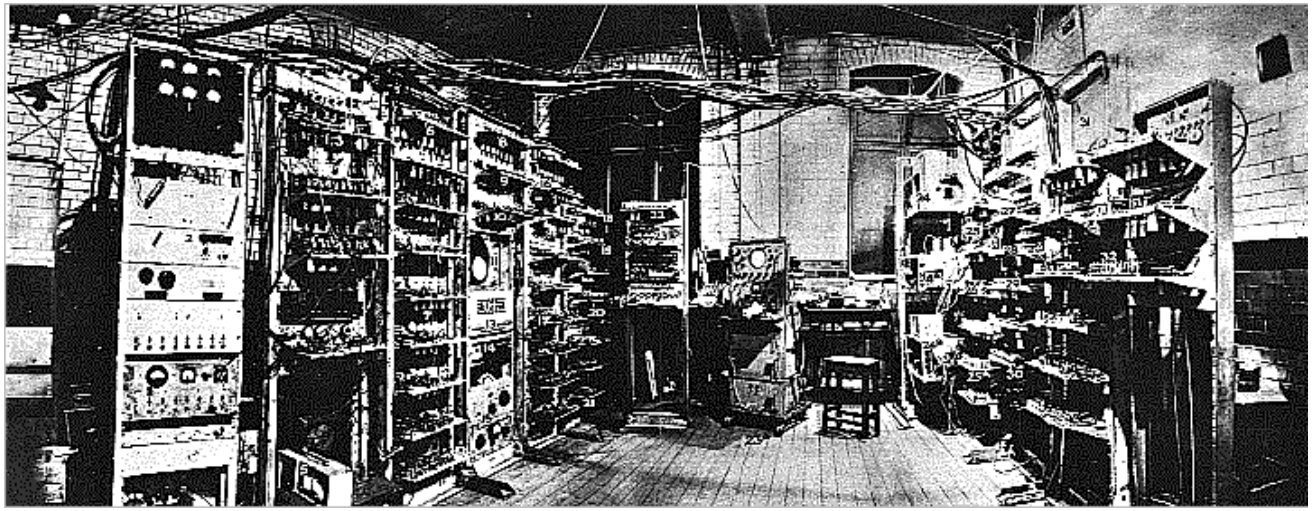






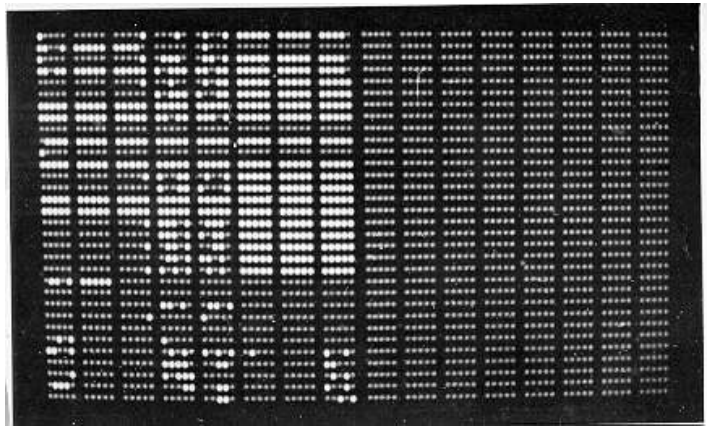
Der *Utah Teapot*: von Hand modelliert von Martin Newell
Mehr zur "Folklore" dazu: http://www.sjbaker.org/wiki/index.php?title=The_History_of_The_Teapot

Kurze Historie der Computergraphik / Rendering



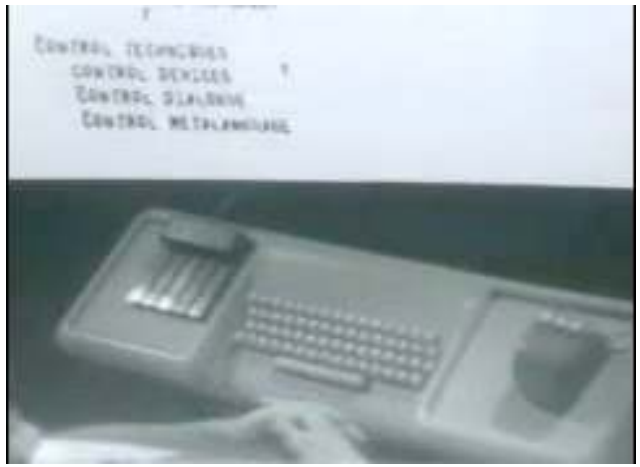
Manchester Mark I

- Am Anfang: noch nicht einmal Text-Displays



Sketchpad (1963) & "The Mother of all Demos" (1968)

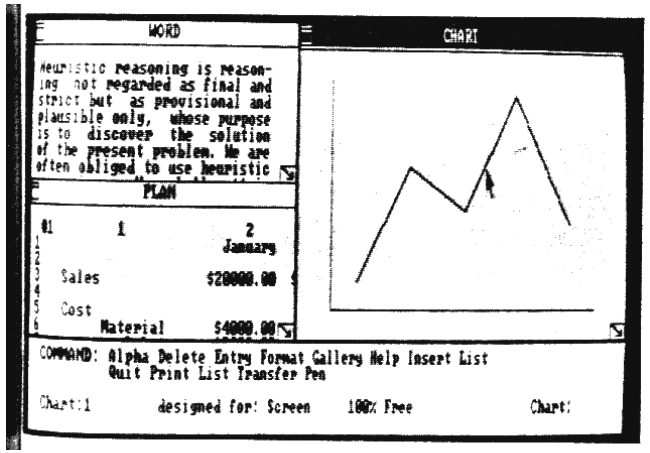
- Ivan Sutherland's Sketchpad
  - MIT, 1963
  - Der Beginn der modernen interaktiven Graphik
  - Sehr teuer!
  - Viele Konzepte findet man in heutigen Zeichensystemen wieder
  - Pop up Menü
  - Hierarchisches Modellieren
- Doug Engelbart
  - Maus
  - Hyperlinks / Hypertext
  - Email, CSCW
  - Telekonferenz, ...



"The Mother of all Demos" Engelbart , 1968

Von Text zu GUIs

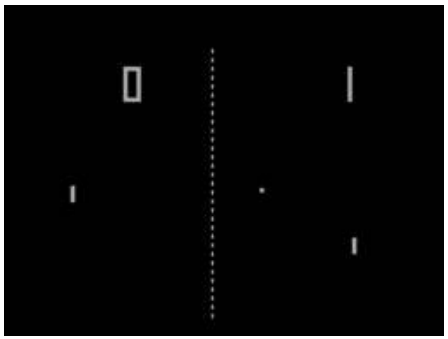
- Ausgedacht vom Xerox PARC etwa 1975
- 1981: "Echtzeit" Vektor-Displays,
  erste bezahlbare Rastergraphik (Apple][ )
- "GUI / Desktop" zuerst kommerziell eingesetzt auf Apple Mac
- Mitte/Ende 80er: C64, IBM PC
  - PCs mit eingebautem Raster-Display
  - Bezahlbare Rastergraphik

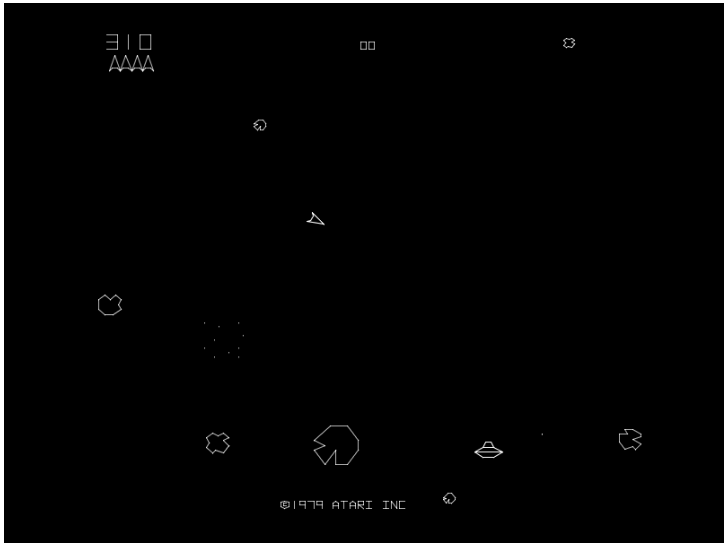


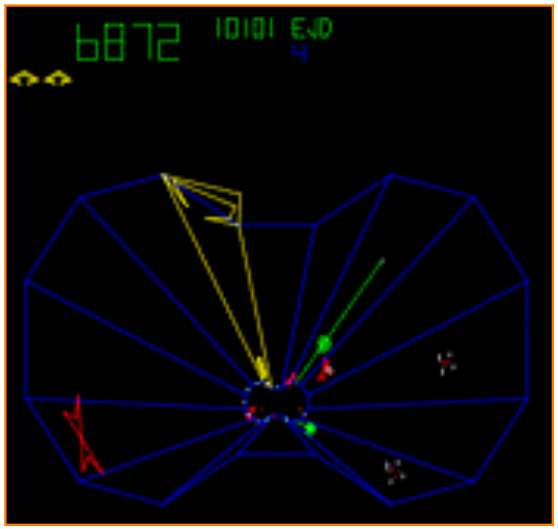
Windows 1.0

Erste Spiele

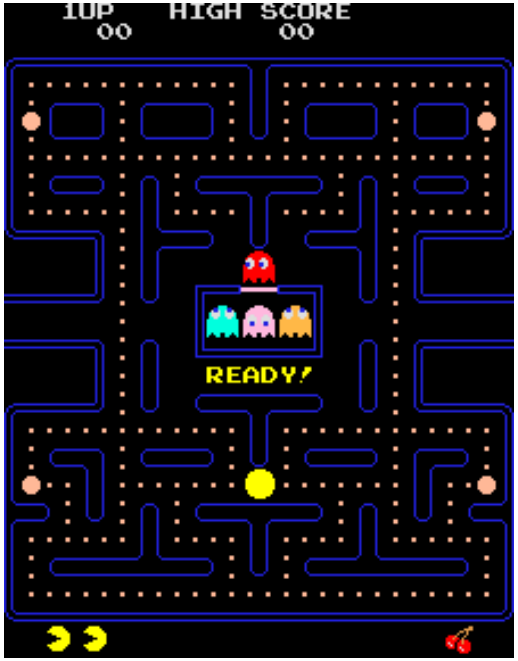
- Zunächst noch reine "Vektorgraphik":
  - Pong
  - Asteroids
  - Star Wars





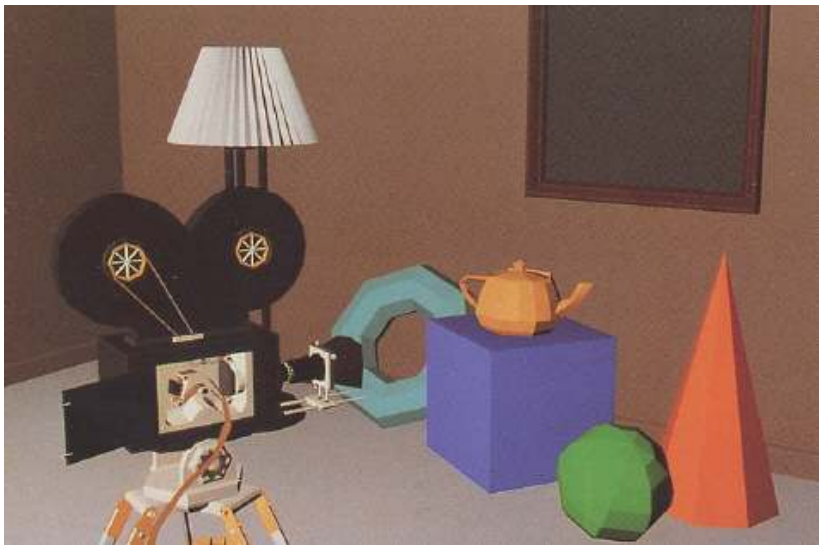


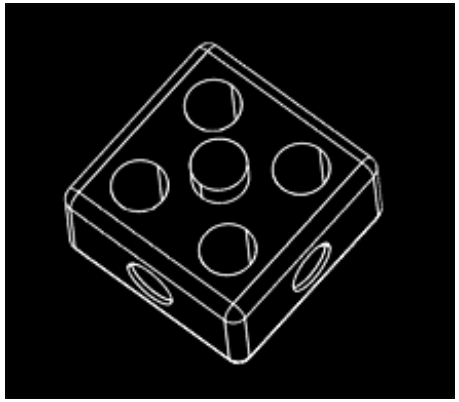
- Dann Rastergraphik:
  - Space Invaders
  - Pac Man



Rendering: 1960 (Sichtbarkeit)

- Roberts (1963), Appel (1967): verdeckte Linien
- Warnock (1969), Watkins (1970): verdeckte Flächen
- Sutherland (1974): Sichtbarkeit = Sortierung



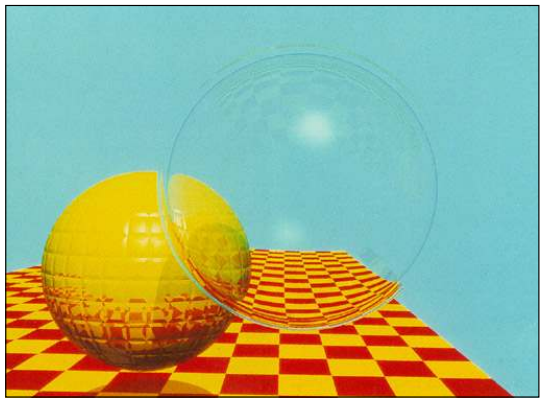
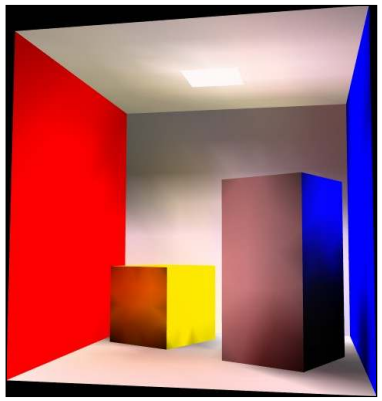
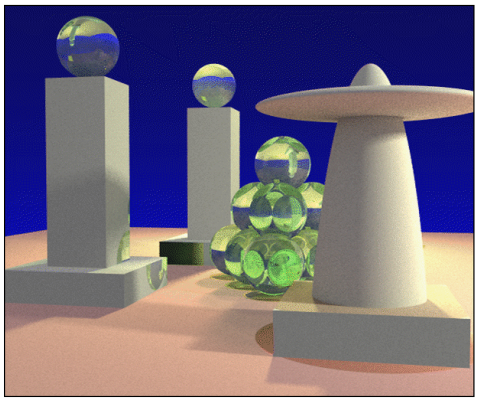


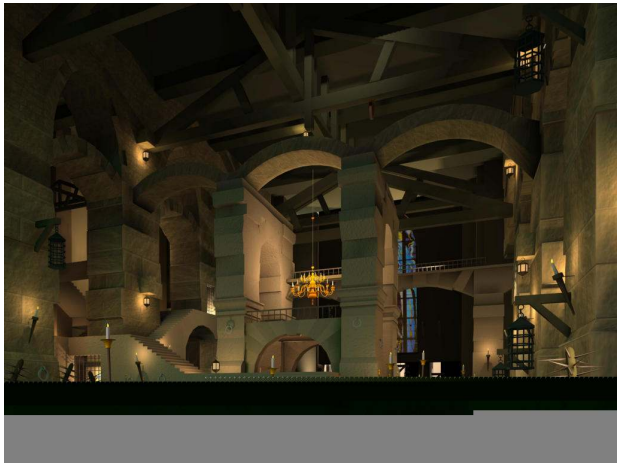
Rendering: 1970 (Beleuchtung)

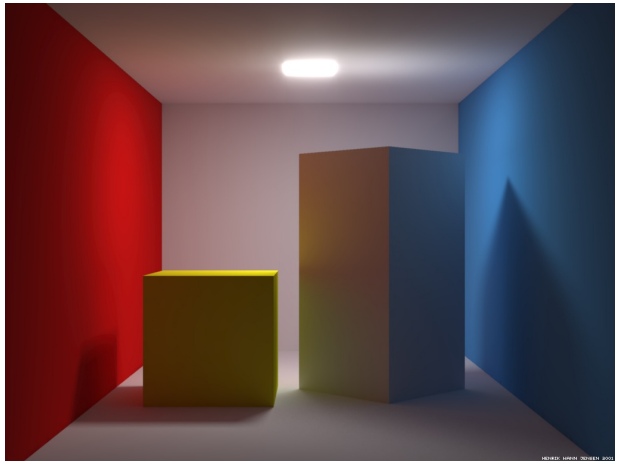
- Raster Graphiken:
  - Gouraud (1971): diffuse Beleuchtung
  - Blinn (1974): gewölbte Oberflächen, Textu
  - Phong (1974): spiegelnde Beleuchtung
  - Catmull / Straßer (1974): verdeckte Fläche

Rendering: 1980, 1990 (Globale Beleuchtung)

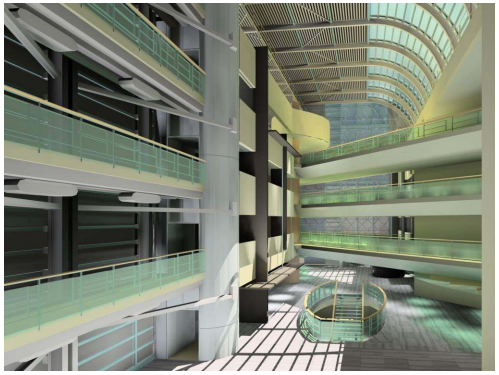
- Whitted (1980) : Ray-Tracing
- Goral, Torrance et al. (1984) : Radiosity
- Kajiya (1986) : Die Rendering-Gleichung

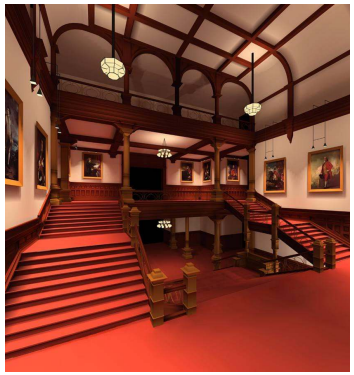
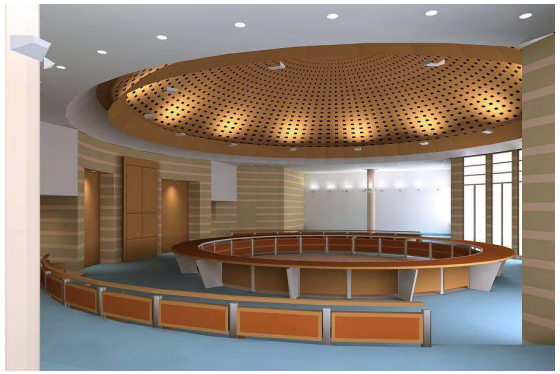












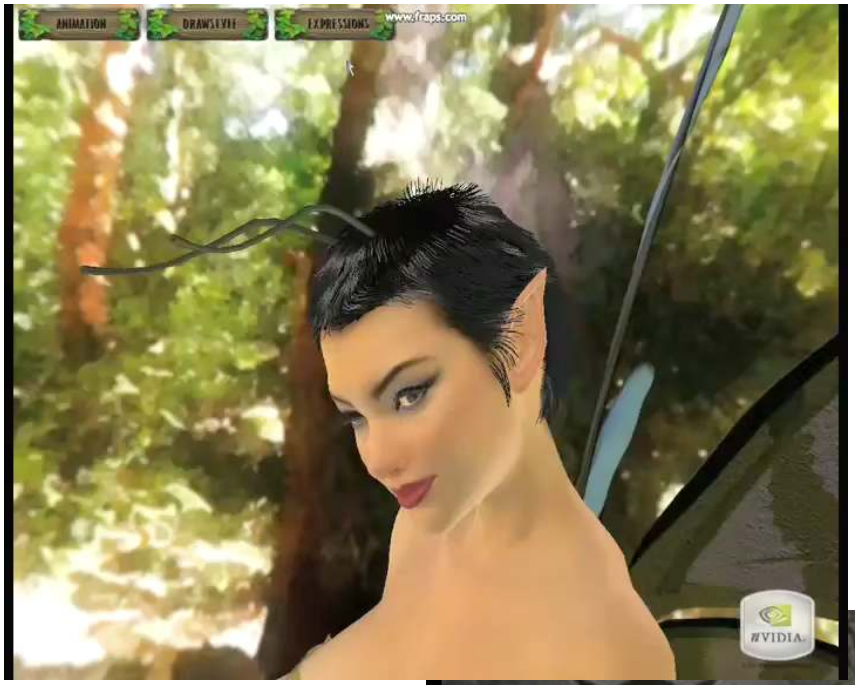



Beleuchtungseffekte bei polygonalem Rendering

+

Polygonales Rendering heute ...

- CG1-Level (Rendering)



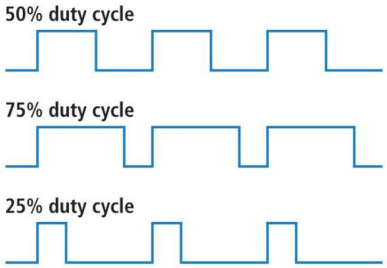
- CG2-Level (Rendering)

Displays – The Ultimate Display?

Avatar, 2009, James Cameron

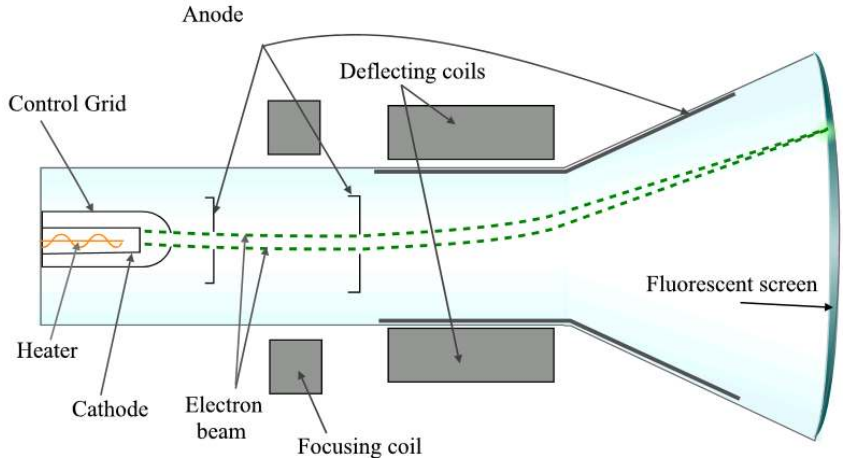
Display-Kategorien

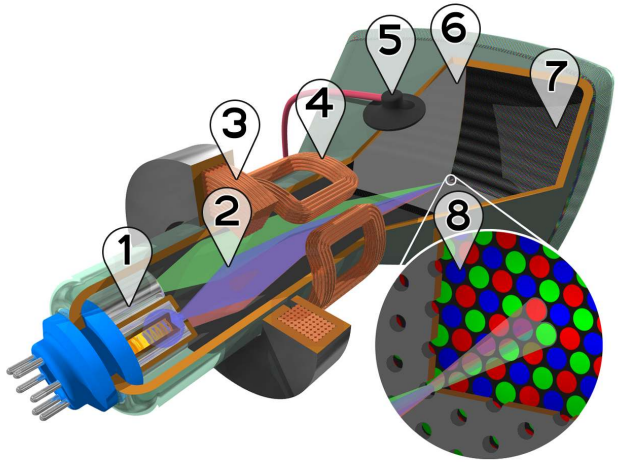
- Emittierende Displays (emissive displays):
  - Keine zusätzliche Lichtquelle nötig
  - Beispiele: Kathodenstrahlröhre (CRT), Laser-Projektion, LEDs
  - I.a. beliebige Helligkeitszwischenstufen durch verschiedene Spannung
- Light Valve Displays ("Lichtventile"):
  - Benötigen separate Lichtquelle, z.B. rückwärtige Beleuchtung
  - Beispiele: LCD, Digital Micromirror Device (DMD), E-Ink,
  - Display lässt Licht passieren / blockiert Licht
  - Manche können nur komplett an/aus, Abstufungen durch sog. Pulse Width Modulation (Duty-Cycles)



Kathodenstrahlröhre (CRT, Braunsche Röhre, 1897)

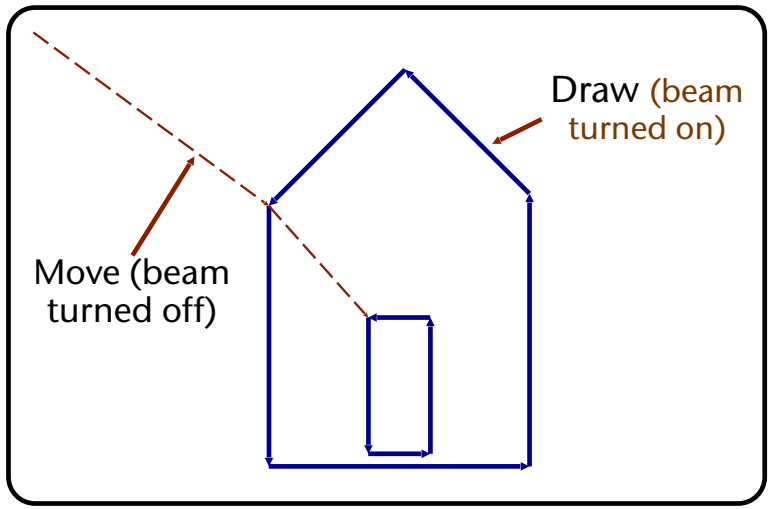
- Elektronen werden
  - erzeugt durch Erhitzung der Glühkathode (1)
  - beschleunigt in Richtung Anode (5)
  - fokussiert (3)
  - abgelenkt (4)
  - gefiltert durch Lochmaske (6)
  - treffen auf Phosphorpunkte (8)
- Lichterzeugung und Farben:
  - Phosphor-Atome "konvertieren" Elektronenbeschuss in Photonen
  - 3 Arten ⟶ rot, grün, blau; dazu drei Elektronenstrahlen



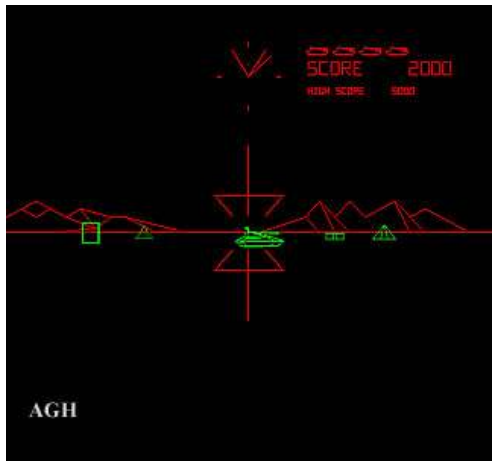


Vektor Displays / Vektorgrafiken

- Bis Anfang / Mitte der 80er
  - Steuere X, Y durch die Spannung an den vertikalen/horizontalen Ablenkspulen
  - Oft wird Intensität durch Z geregelt



Tempest



Battlezone

Vektor-Displays heute

- Hauptsächlich für “Entertainment displays” (Laser Shows, Exhibits, ...)

Chalky

Refresh (Aktualisierung)

- Ein Bild muss aufgefrischt werden, um ein neues Bild darzustellen
  - Aktiviert der Elektronenstrahl eine Region von Phosphoratomen, so verblasst diese nach einer Weile (einige Millisekunden)
- Folge: der Elektronenstrahl muss regelmäßig alle Stellen des Bildes treffen, um Flimmern zu vermeiden
- Kritische Frequenz: 25 Hz (Vollbilder!)
- Max. mögliche Refresh-Rate hängt bei Vektordisplays von Anzahl und Länge der Linien ab → beschränkte Komplexität der Szene

- Persönliche Konvention auf den Folien: Begriffe, die neu definiert werden, werden mit blauer Schrift geschrieben (egal wo auf der Folie)

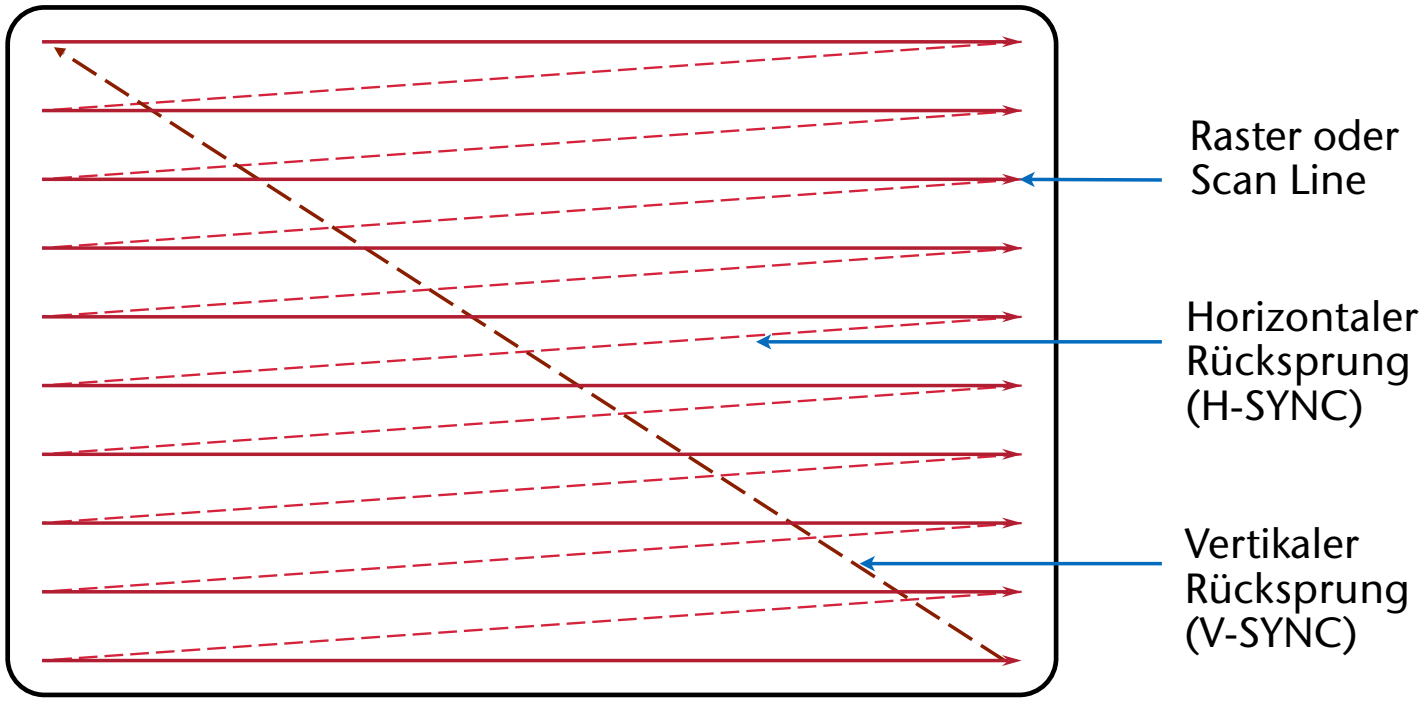

20
21
EXPERIENCE
SIGGRAPH

Bitte spekulieren: woher kommt das "Flackern"?

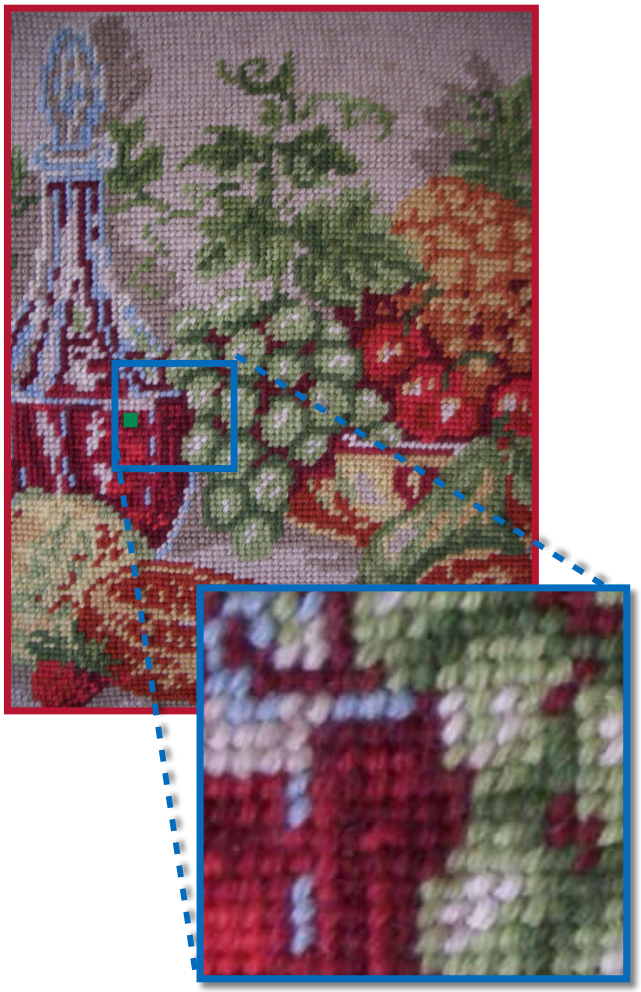
Oder:
www.menti.com/
qyragw8ezp

Rastergraphik (Raster / Scanline Displays)

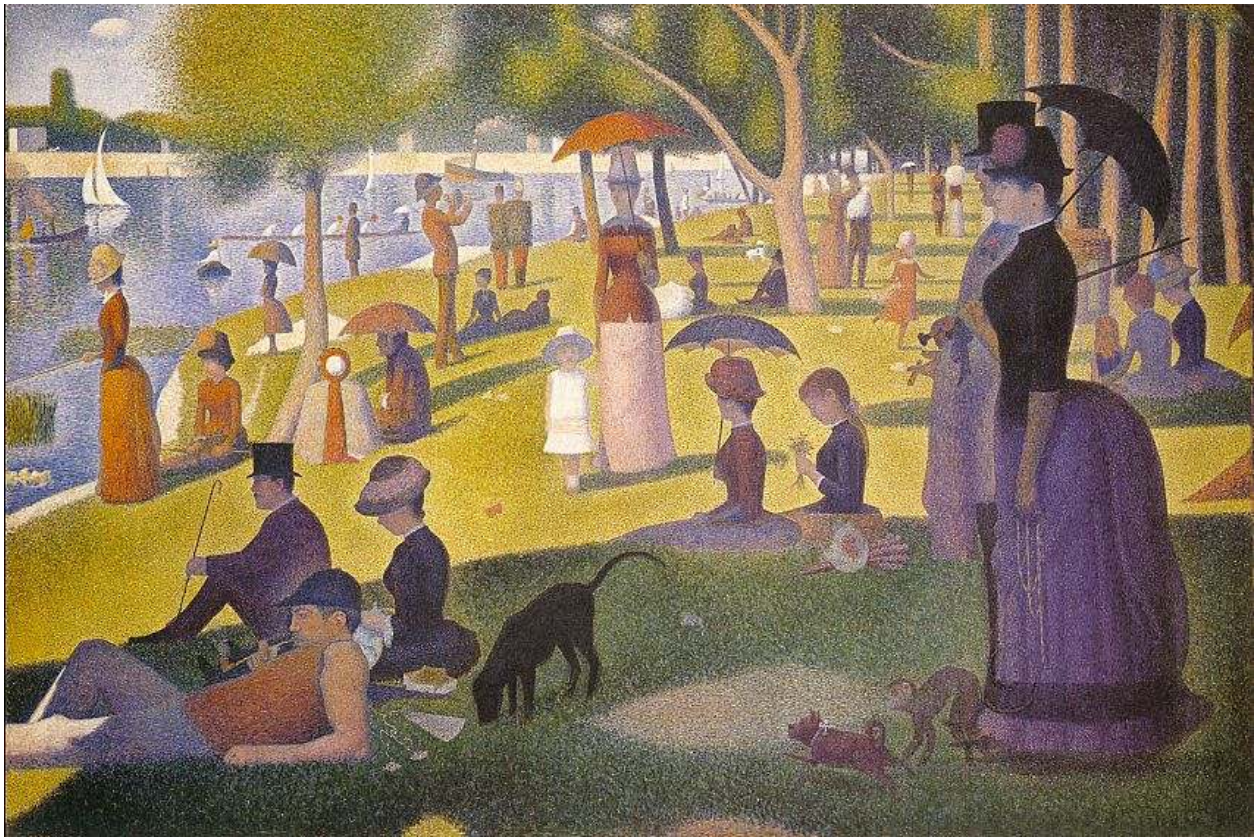
- Heutzutage sind fast alle Displays Raster-basiert
- Speicherung von Bildern als Bildpunktmatrix
  - Feste Informationsmenge pro Bildpunkt (und damit: pro Bild)



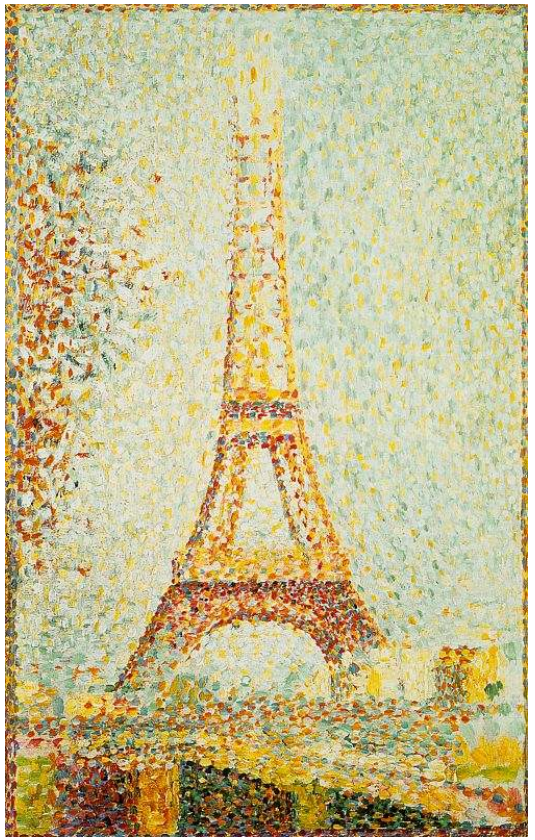
Beispiele aus anderen Bereichen



Die Pointillisten

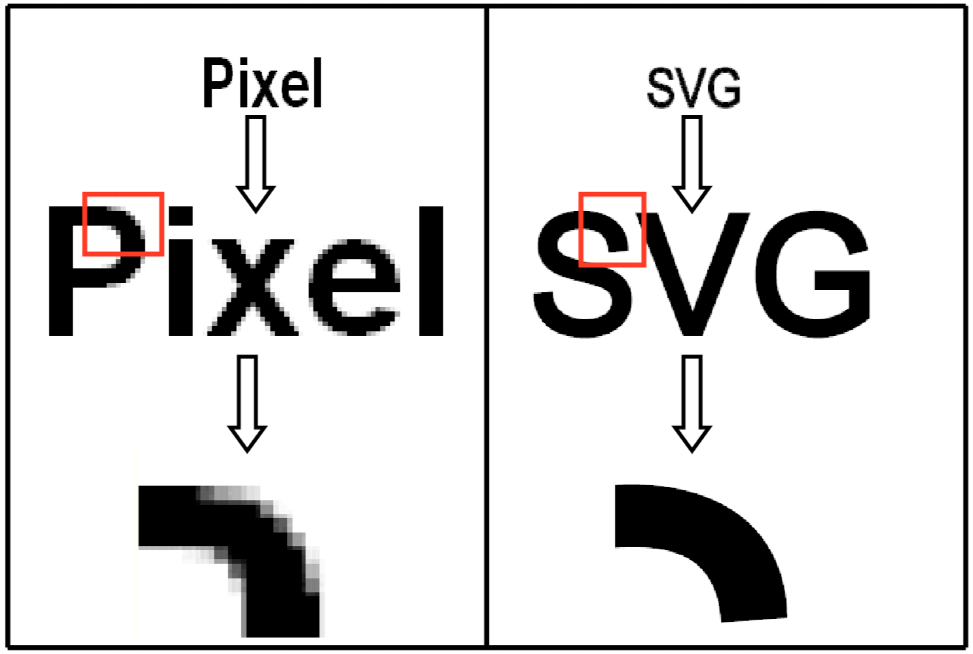


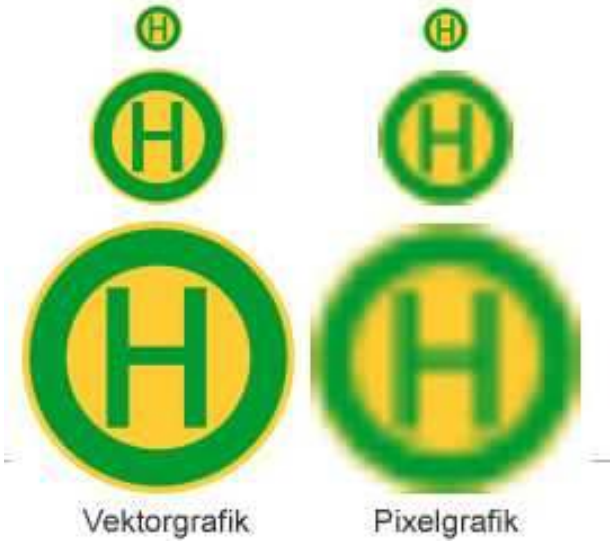
Seurat, A Sunday Afternoon on the Island of La Grande Jatte, 1884-86



Seurat, The Eiffel Tower, 1889

Kleiner Exkurs: Pixelgraphik vs. Vektorgraphik





Beispiele und Vor-/Nachteile für Pixel-/Vektorgraphik

Oder:
www.menti.com/
qyragw8ezp

Bildformate für Pixelgraphik & Vektorgraphik

Vektorgraphik

Linien- und Kurvenbeschreibungen für geometrische Formen, sog. *Paths* (auch Text)

Vorteile

- Skalierung ohne Qualitätsverlust
- Kompression / geringe Dateigröße
- Textbearbeitung

Vertreter: PDF, SVG

Pixelgraphik (aka. Rastergraphik)

Quadratische Bildpunkte im Raster, denen jeweils eine Farbe zugeordnet ist

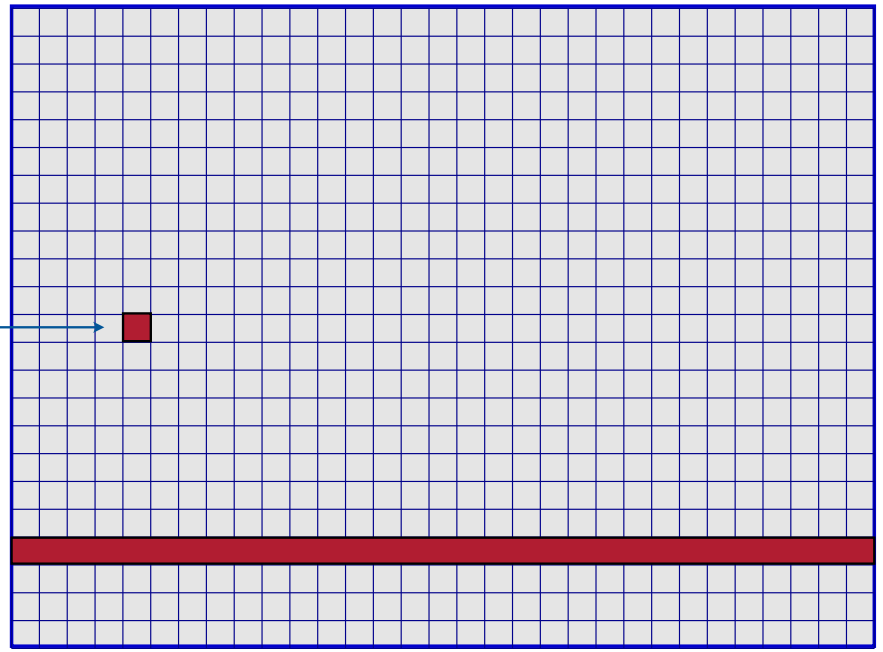
Vorteile

- Programmunabhängig
- Bearbeitung jedes einzelnen Bildpunktes
- Detailreicher

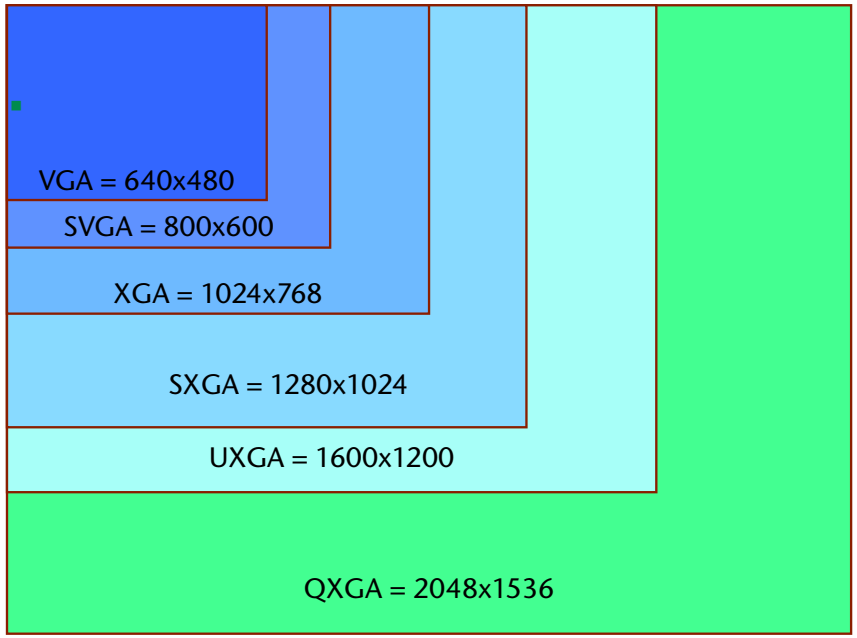
Vertreter: JPG, PNG, BMP, GIF

Einige Fachbegriffe

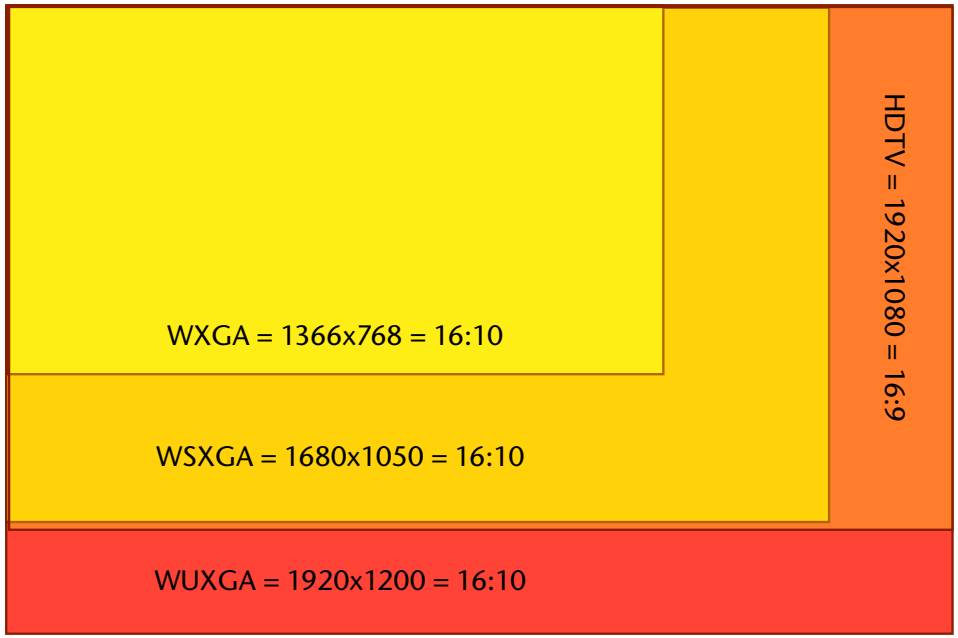
- Frame
  1. ein 2D Array von Pixeln oder Zellen (wird auch so gespeichert)
  2. Einzelbild auf dem Monitor
- Pixel ("picture element"): eine einzelne Zelle eines Frames
  - Speichert nicht nur Farben!
- Scanline: eine Reihe von Pixel
- Auflösung: eigentlich Pixel pro Zoll (ppi); hier Größenbeschreibung von Bildern (z.B. 1280x1024)
- Aspect ratio = Breite : Höhe (früher 4:3, jetzt immer mehr 16:9)



Standardauflösungen

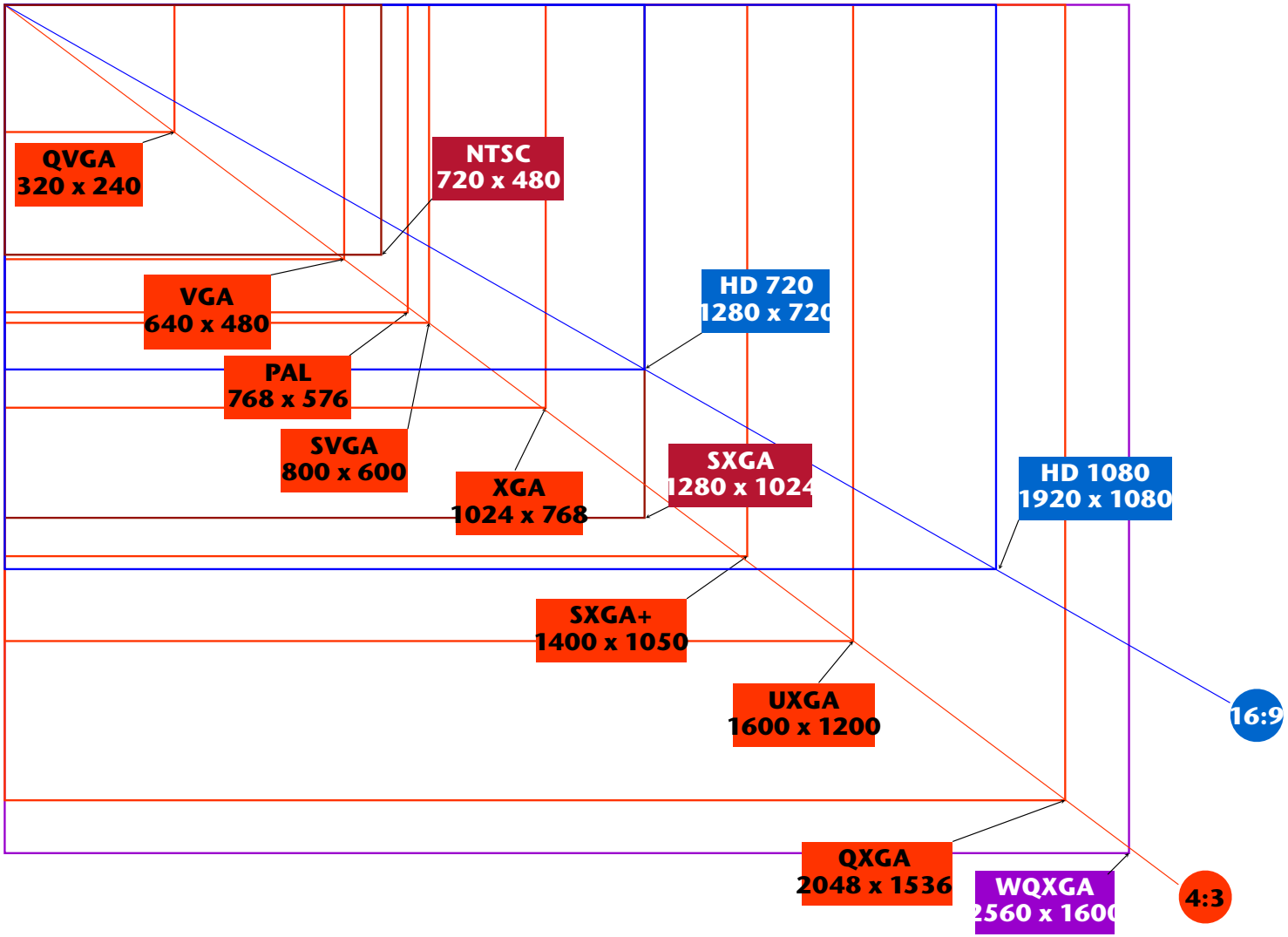


Standardauflösungen:
diese haben eine *aspect ratio* von
4:3 = 1.33:1, außer SXGA mit 1.25:1

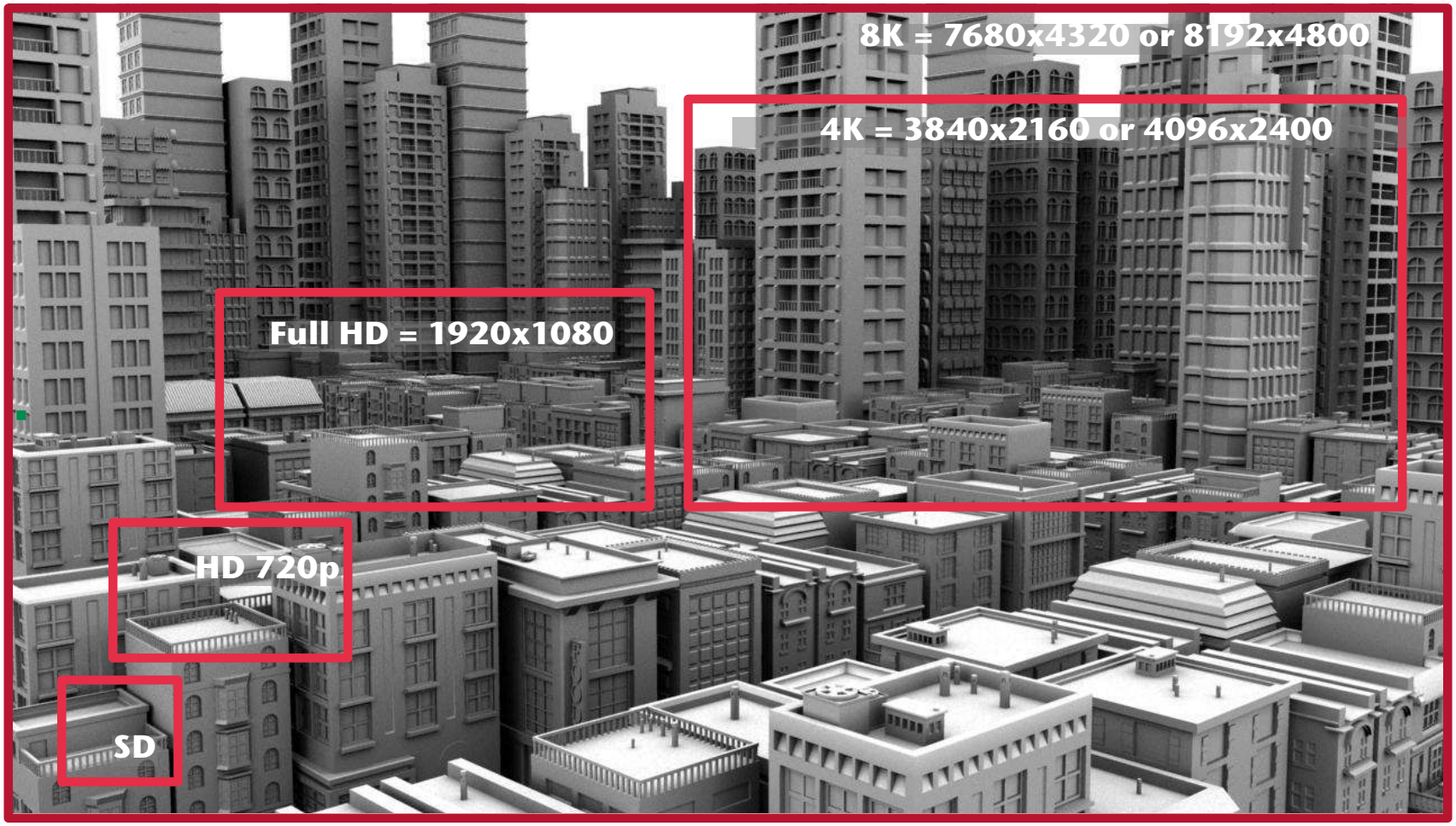


Wide-screen-Auflösungen:
Aspect ratio ≈ 16:9 ≈ 1.78:1.
(Viele Kinofilme sind in 1.85:1
oder 2.35:1 ≈ 21:9 gedreht.)

Standardauflösungen ?



Current Trend: Exponential Pixel Growth

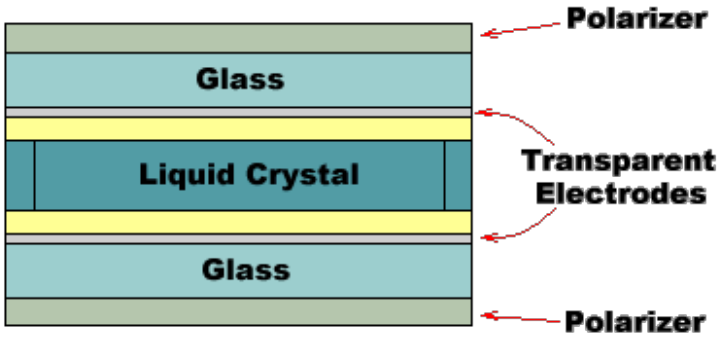


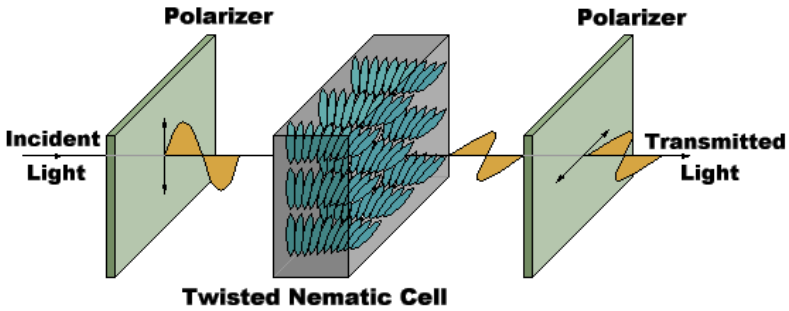
[Rose Adler, Leighana Ginther, Jackie Osterday]

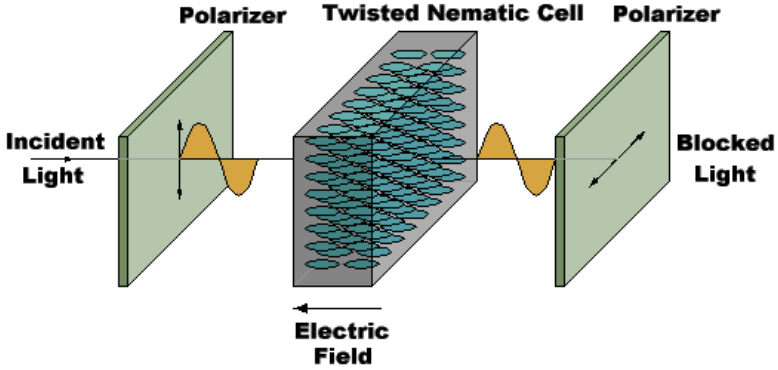
Liquid Crystal Displays (LCDs)

- LCDs lassen das Licht hindurch bzw. nicht und sind somit auf eine externe Lichtquelle angewiesen
- Laptop Bildschirme: von hinten beleuchtet, durchlässige Displays
- Alte Handy's: reflektierende Displays (+ externe Lichtquelle)

Sandwich structure of LCD's

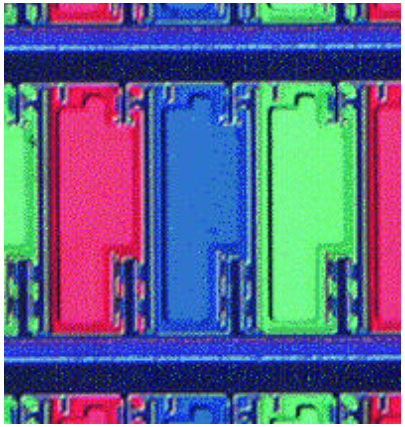


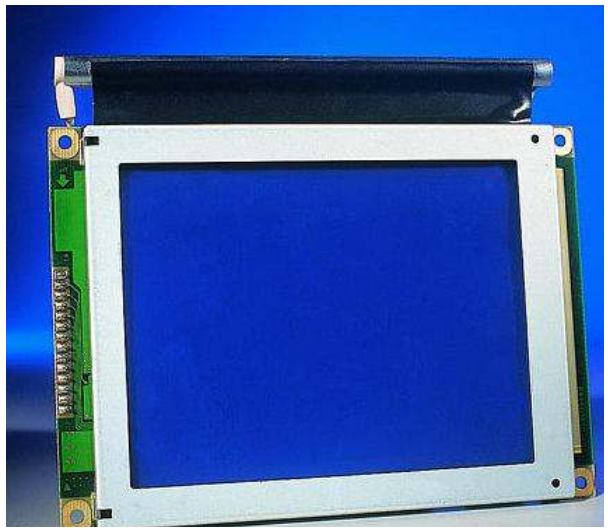




TFT-LCD-Displays

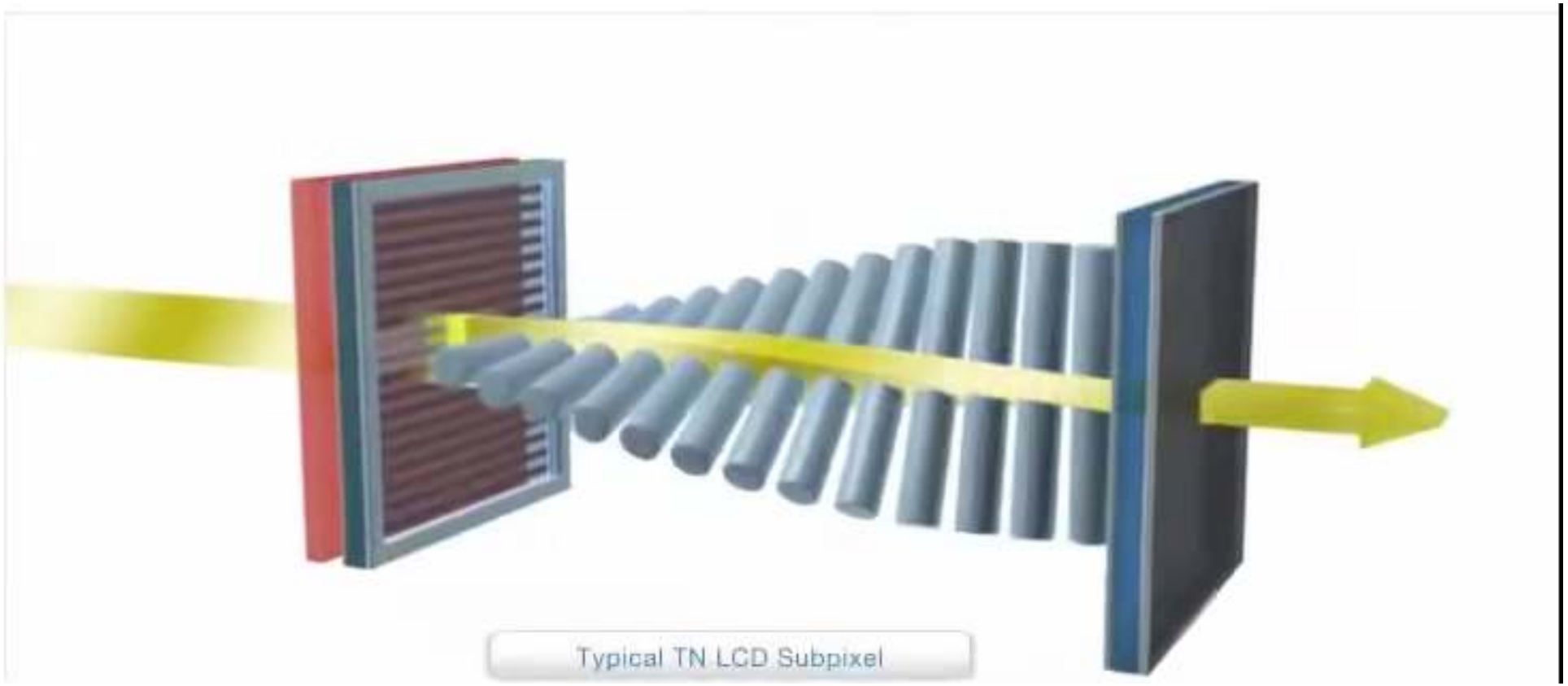
- Pixel besteht aus drei Sub-Pixeln mit R/G/B-Farbfilter
  - Jedes Subpixel ist ein Transistor!
- Leuchtmittel bei transmissiven LCDs
  - schmale Leuchtstofflampe oder LEDs an der Seite
  - Licht wird verteilt durch flachen Lichtleiter + Diffuser-Scheibe
  - Liefert etwas sichtbares Licht, aber vor allem UV-Spektrum
  - Beschichtung an der Innenseite des Glaspanels erzeugt daraus sichtbares Licht
- Reflektive LCDs schalten Hintergrund nur bei Bedarf an





- Das Licht durchdringt den hinteren (vertikalen) Polarisator, wird dabei polarisiert (schwingt nur noch in einer Richtung)
- Nichtaktivierte Flüssigkeitskristalle drehen die Polarisierung um 90° → Licht gelangt durch vorderen (horizontalen) Polarisator
- Angeschaltete Transistoren erzeugen ein elektrisches Feld (in diesem Subpixel)
  - Das führt zu einer Ausrichtung der Flüssigkristalle
  - So ausgerichtete Flüssigkeitskristalle ändern die Polarität des Lichtes nicht
  - Licht wird vom vorderen (horizontalen) Polarisator geblockt
- Die Transistoren werden Zeile für Zeile nach dem Scan-Line-Verfahren aktualisiert
- Die Kristalle müssen eine gewisse Zeit ausgerichtet bleiben, um Flimmern zwischen der Aktualisierung zu verhindern

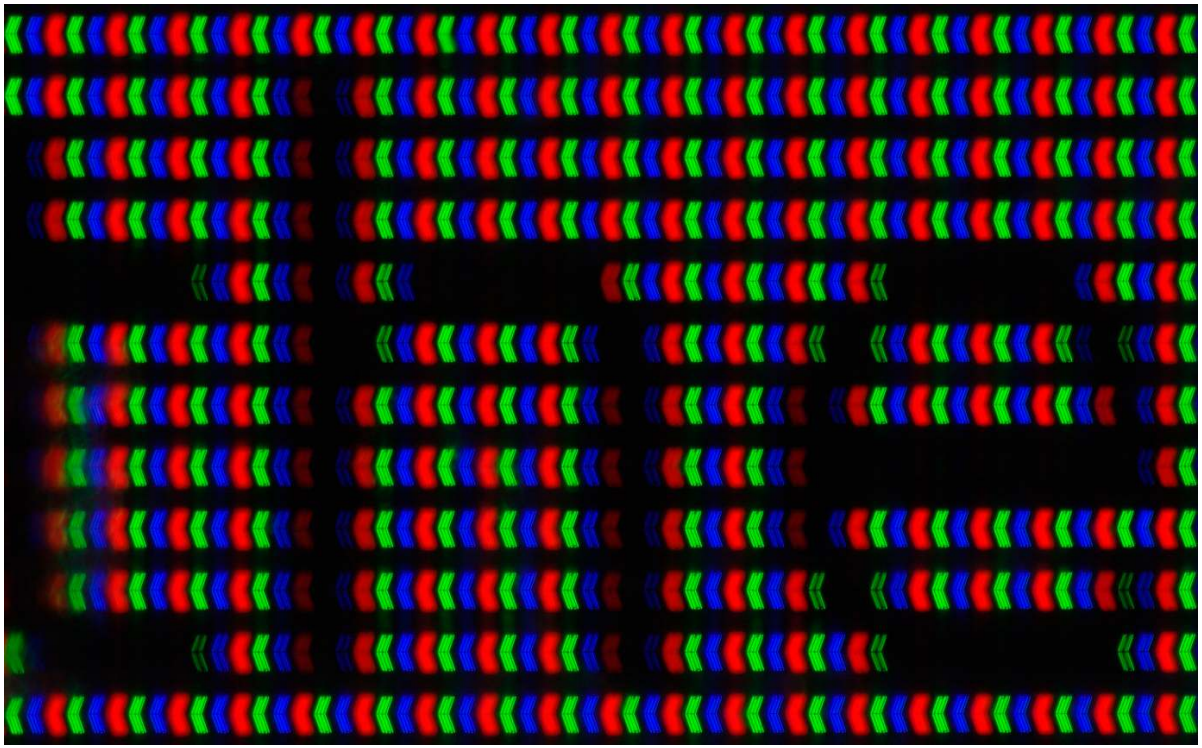
Das Ganze nochmal als Video



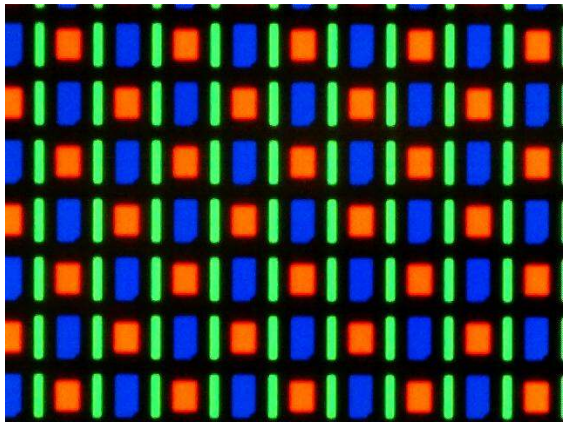
Ausschnitt aus http://solutions.3m.com/wps/portal/3M/en_US/Vikuiti1/BrandProducts/secondary/optics101/

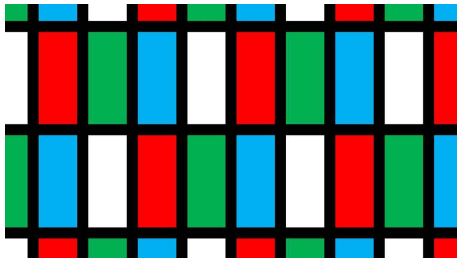
Weiterentwicklungen

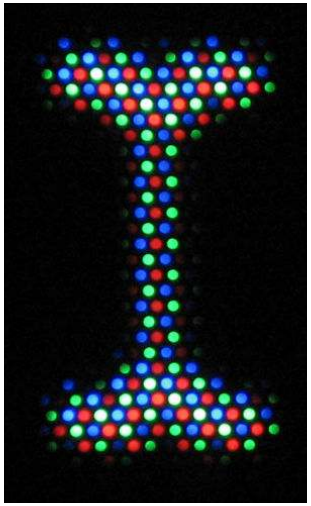
- Einfache LCD-Displays haben das Problem, dass die Farben & Helligkeit vom Viewing Angle abhängen!
- Nicht so IPS-Displays (u.a.):

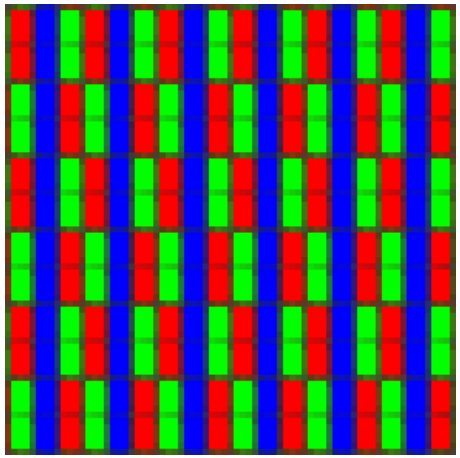


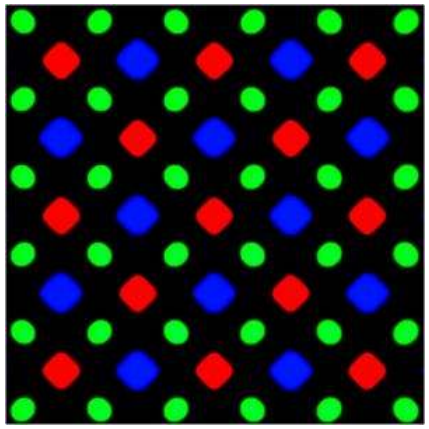
Weitere Sub-Pixel Anordnungen

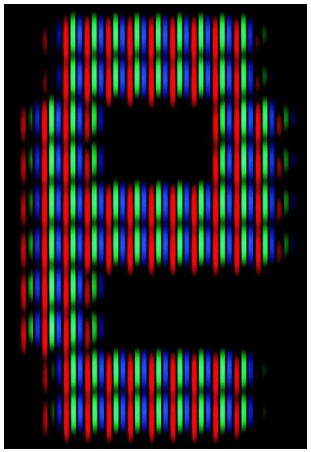






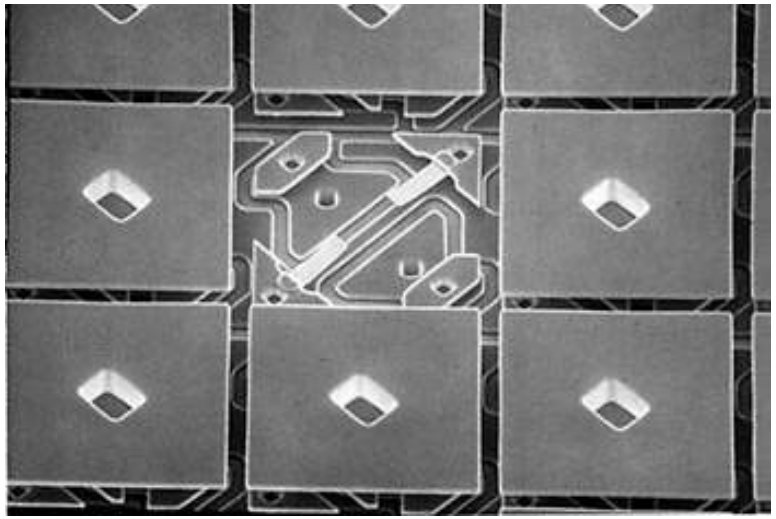


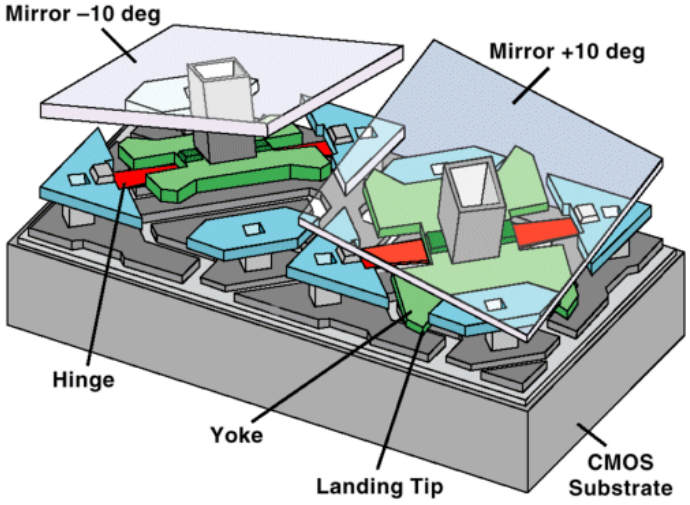




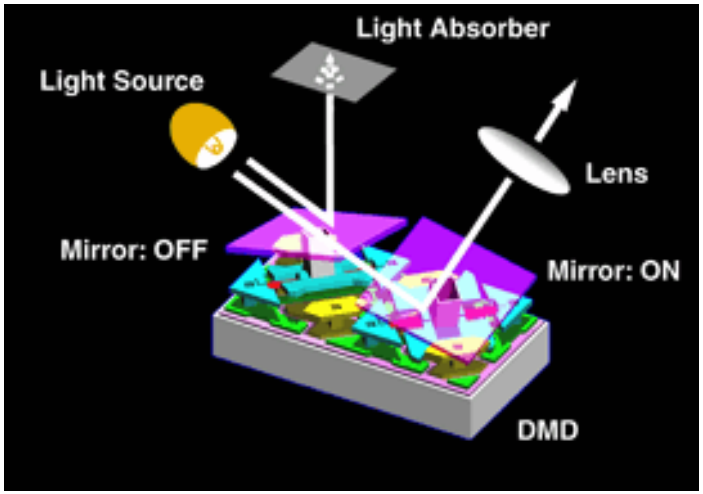
Digital Light Processing Devices

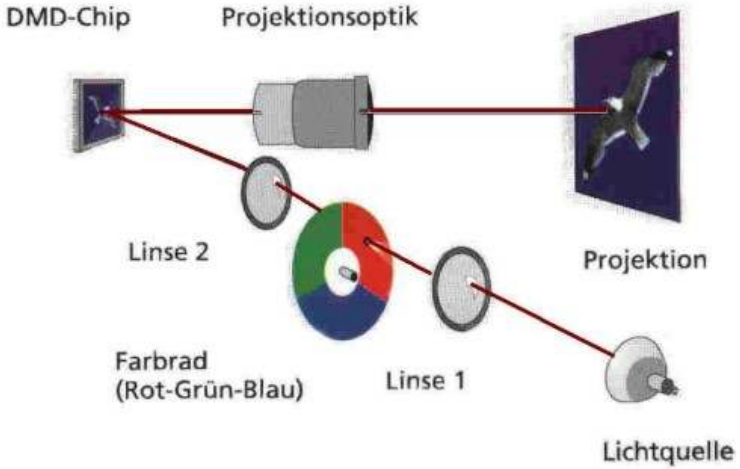
- DMD-Chip (Digital Micromirror Device):
  - Kernstück eines DLP-Projektors
  - Microelectromechanical (MEM) Geräte werden mit VLSI Technik gefertigt
  - Auf 2 cm² über 508.000 reflektierende Mikro-Spiegel, jeder für sich um bis zu 10° kippbar
  - Jeder Spiegel kann einzeln elektrostatisch bewegt werden, schaltet genau ein Pixel an/aus





- DMDs haben „binäre“ Pixel →
  - Verschiedene Grauwerte durch Anpassen der Impulslänge
- Drei Grundfarben per rotierende RGB-Farbfilter-Scheibe oder mehrere Chips
- Vorteile:
  - Hochauflösend
  - Sehr flach
  - Sehr lichtstark
- Problem mit Flimmern

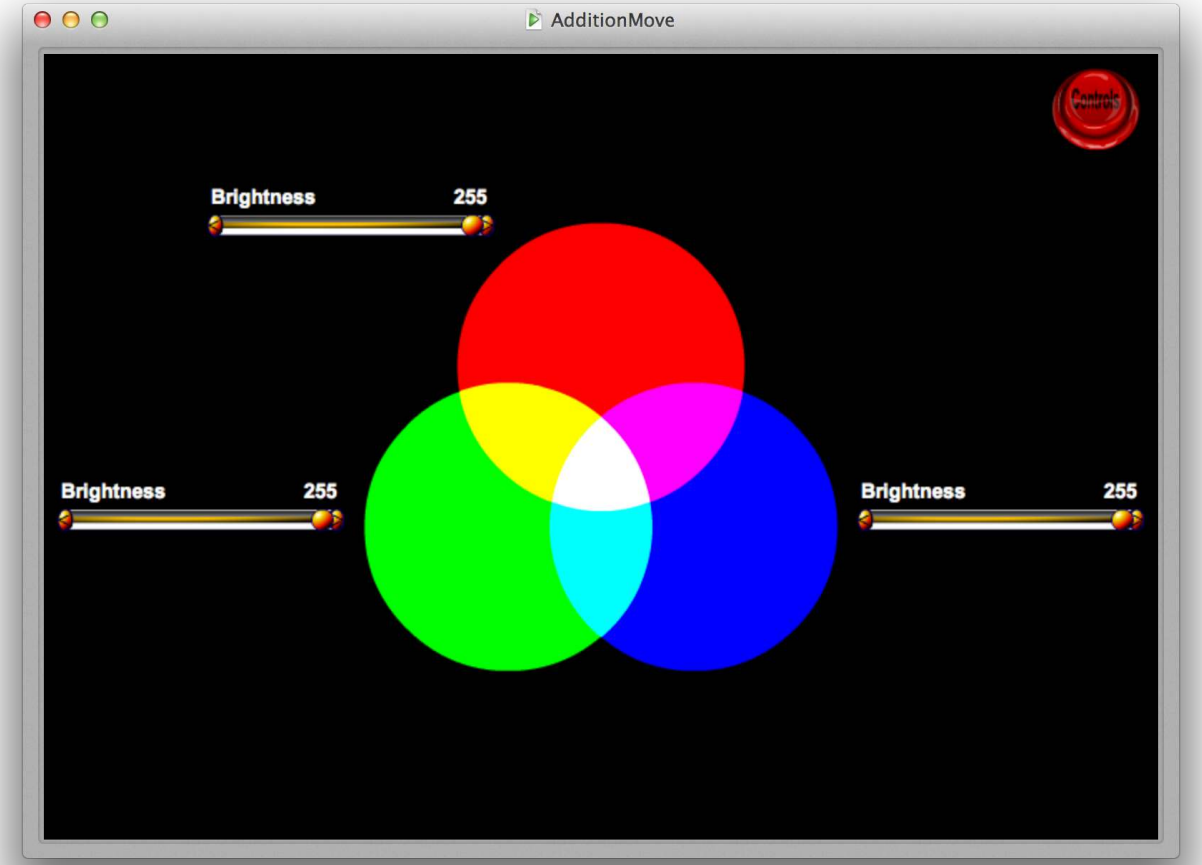




Additive Farbmischung

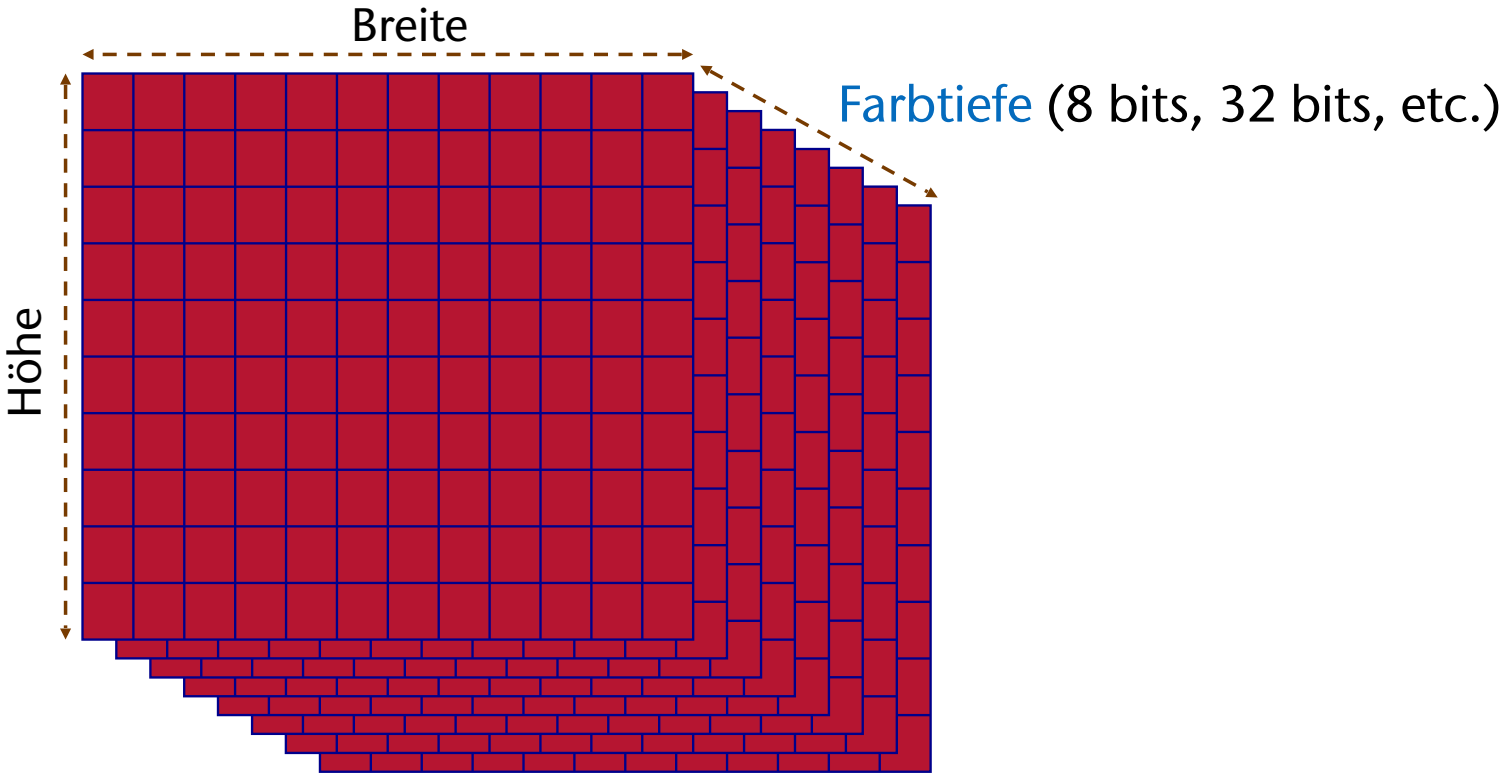
(vorerst oberflächlich)

- Das RGB-Farbmodell:
  - Farbe $= (r, g, b) \in [0,1]^3$
  - (0, 0, 0) schwarz
  - (1, 0, 0) rot
  - (0, 1, 0) grün
  - (0, 0, 1) blau
  - (1, 1, 0) gelb
  - (1, 0, 1) Magenta
  - (0, 1, 1) cyan
  - (1, 1, 1) weiß



Der Frame Buffer

Muss bei Farbdisplays viele Bits pro Pixel spendieren (später noch mehr)



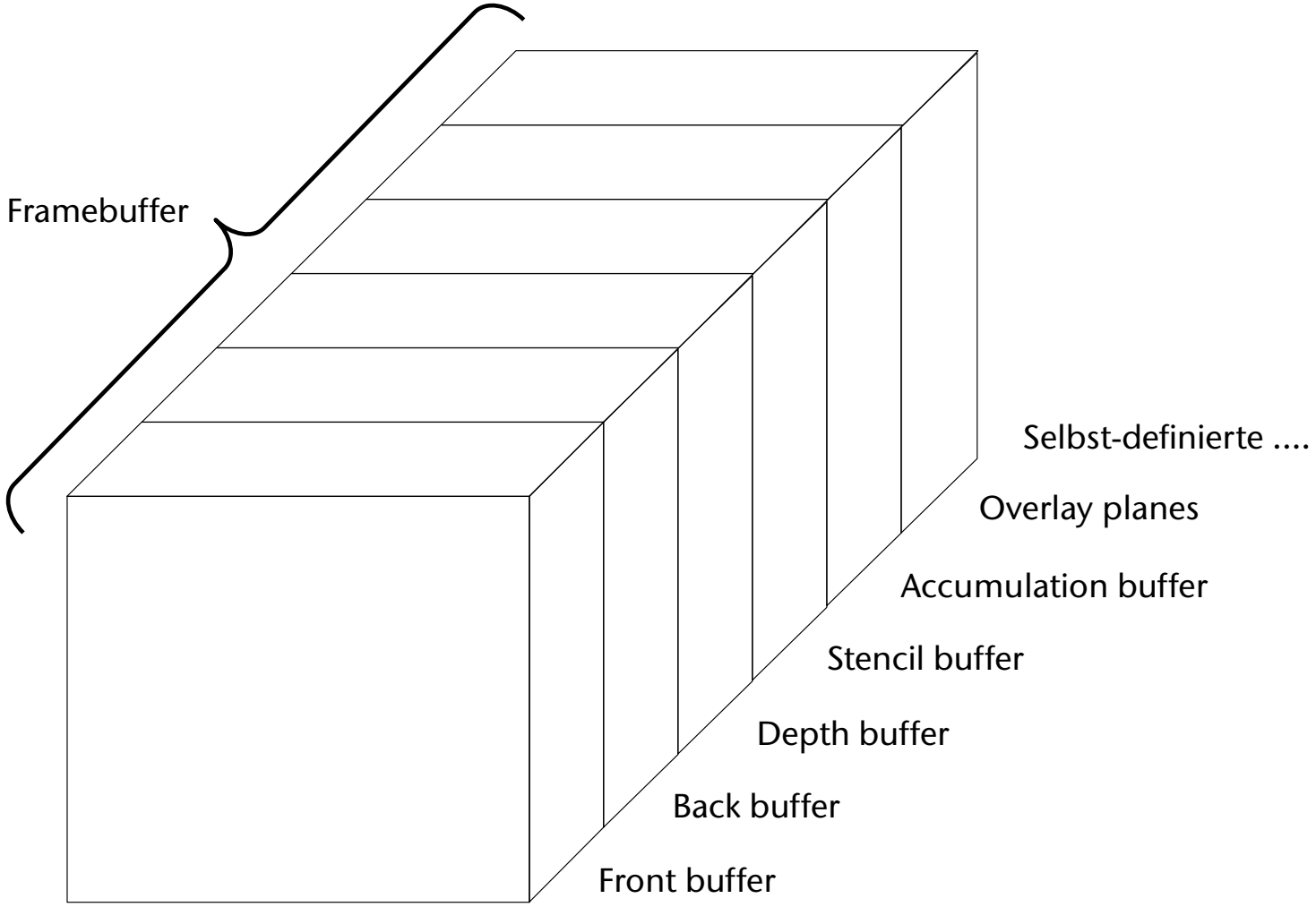
Memory (Bits) = Breite * Höhe * Farbtiefe

Farbtiefen

- Schwarz und Weiß: 1 Bit/Pixel
- Grauskala: 8 Bit/Pixel
- 8-bit Farbe: spart Speicherplatz, 3-2-2 oder Color Lookup Table
- 24-bit Farbe: 8 Bit pro Farbkanal – rot, grün, blau (RGB)
- Wie groß muss der Frame-Buffer für ein 1600x1200 Pixel großes Bild in RGB sein?
  - 8 Bit für jeden Farbkanal
  - Das sind 24 Bit/Pixel
  - Das ergibt $1600 \cdot 1200 \cdot 24$ Bit = 5.76 MBytes
  - Die meisten Graphikkarten reservieren 32 Bit/Pixel bei true color

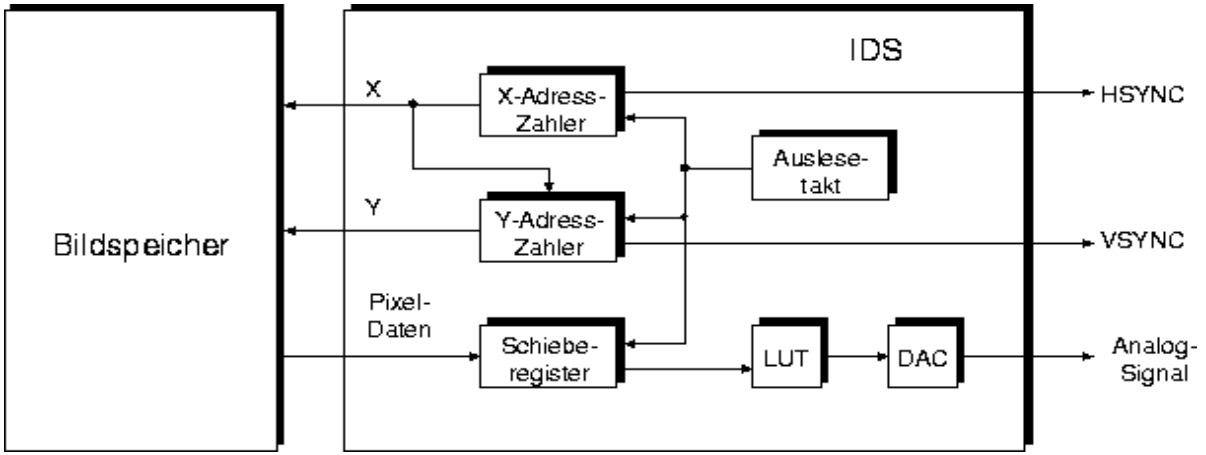
    = 7.68 MBytes
- Datenrate bei 30 frames per second (FPS): 230 Mbytes / sec

Kleine Antizipation: es gibt viele weitere Buffer in einem Framebuffer

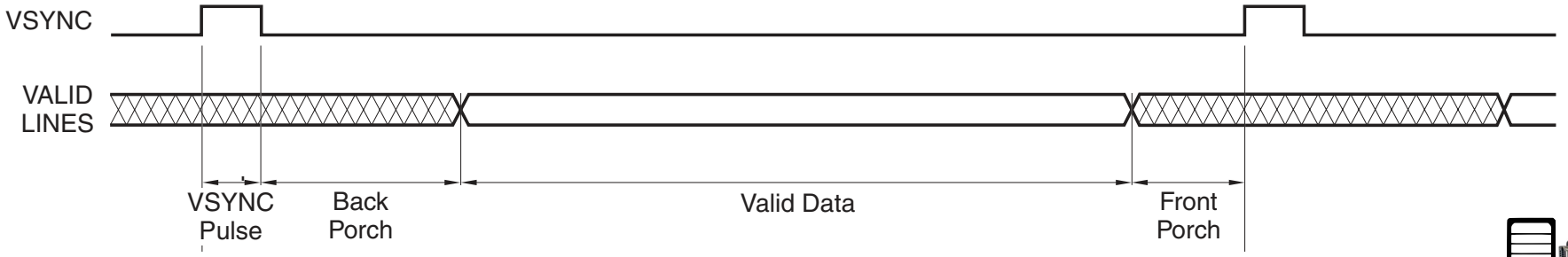


Der Video-Controller

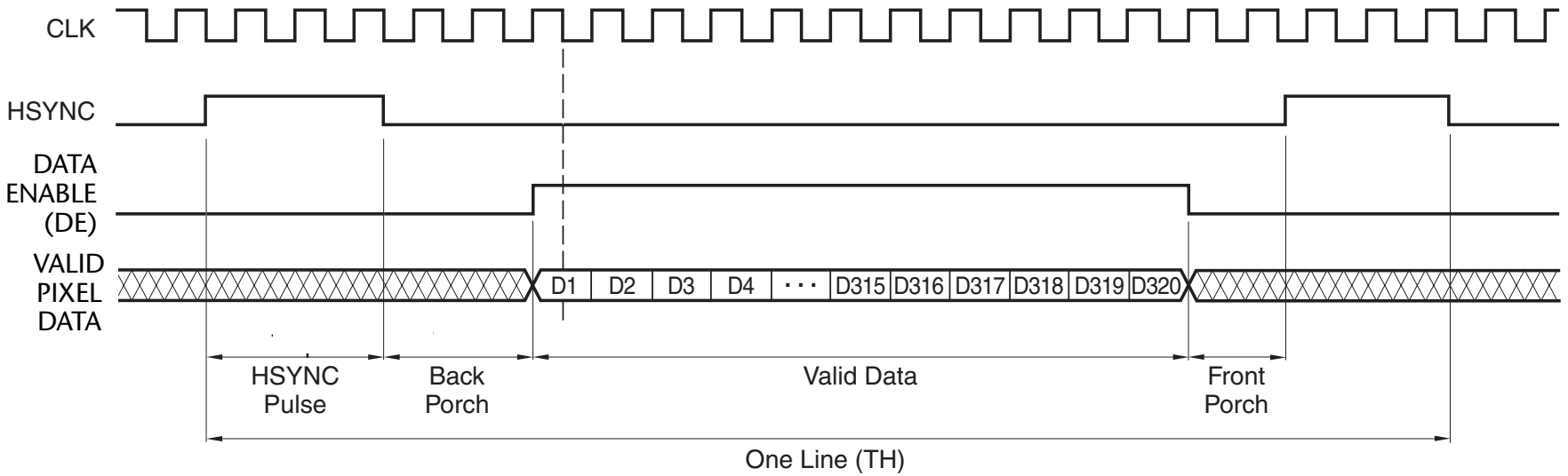
- Erzeugen der horizontalen (HSYNC) und vertikalen (VSYNC) Synchronisationsimpulse für das entsprechende Bildformat
- Adressierung und Auslesen des Bildspeichers
- Ansteuern des Monitors mit entsprechenden Intensitäts-/ Farbwerten, mit Dunkelsignal für H/V-Austastlücke und Digital-Analog-Wandlung (DAC).



Technical Details on Transmitting the Signals

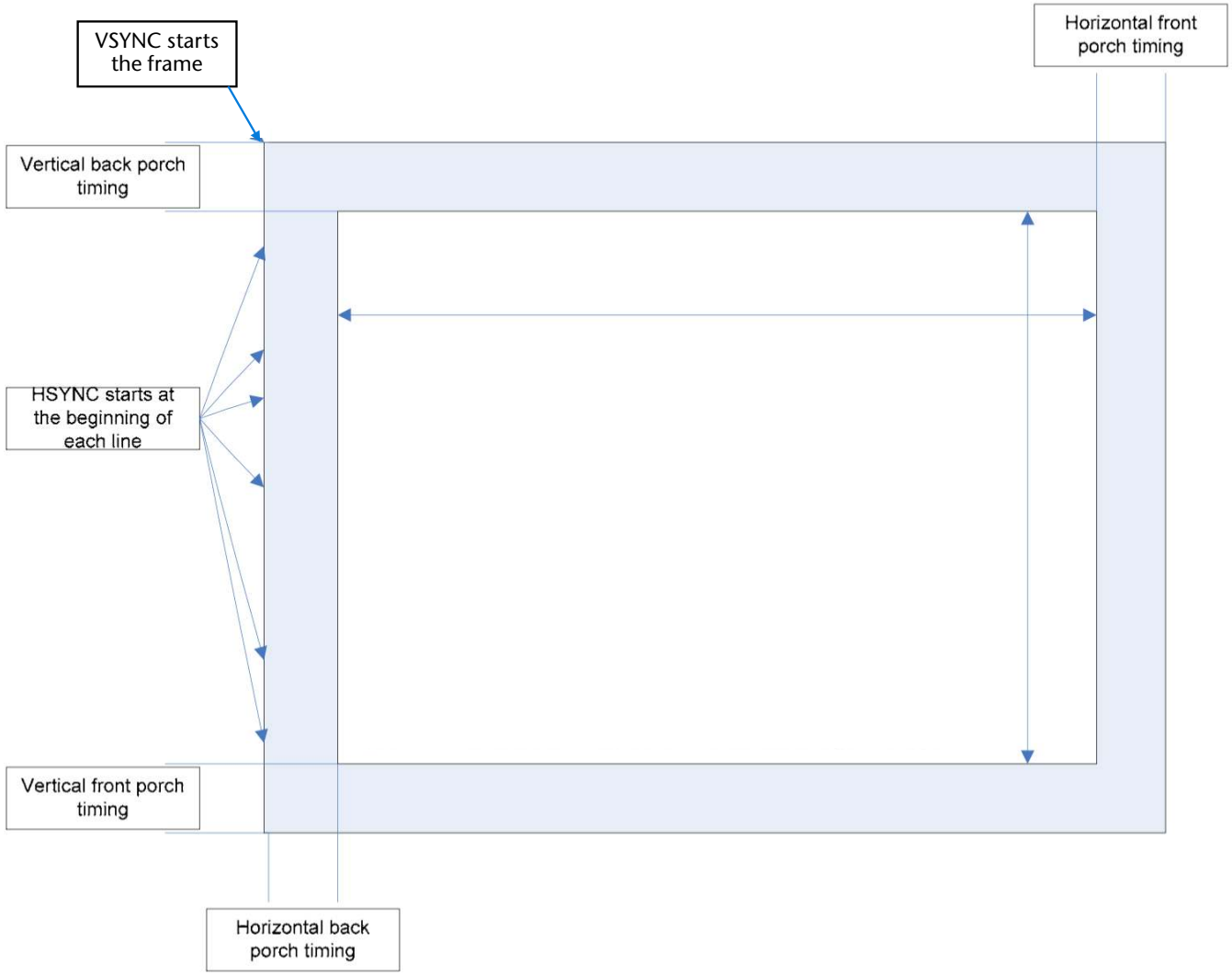


Vertical Signal Timing

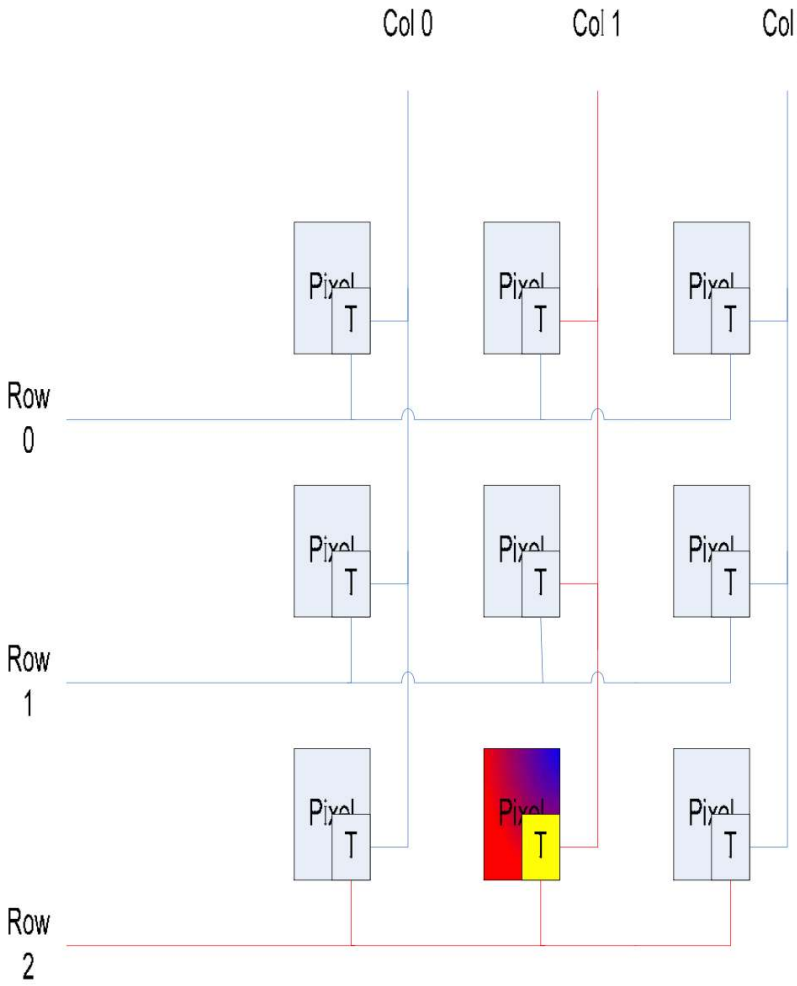


Horizontal Signal Timing

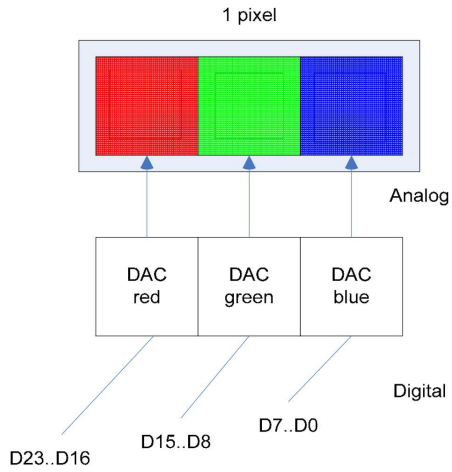
Visualisierung der Timings als "Frame"



Ansteuerung auf der Monitor-Seite (LCD / LED)

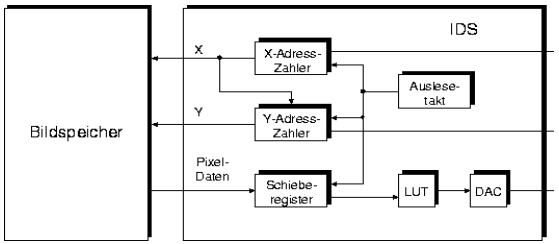


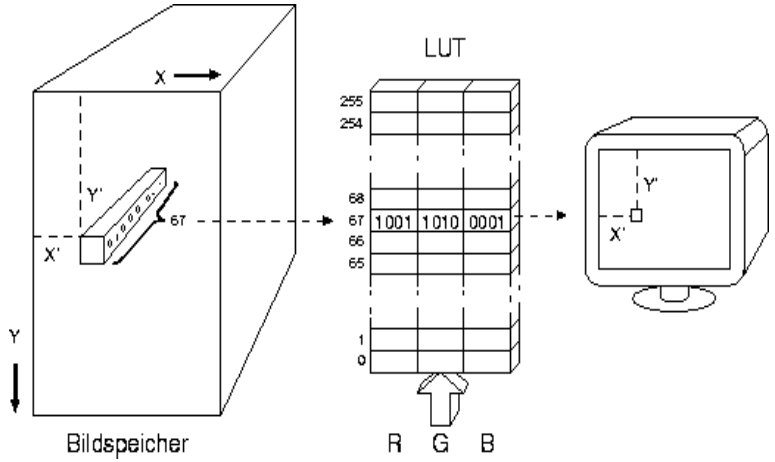
TFT LCD-Displays

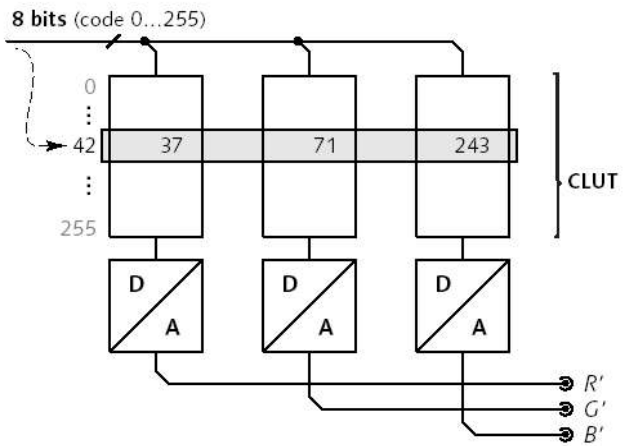


Farbtabelle (Color Lookup Table, Pseudo-Color)

- Eine Idee, um die Datenrate und den Speicheraufwand zu senken:
  - Erstelle eine Color Lookup Table (LUT), welche *alle* im *aktuellen* Bild benötigten Mischfarben enthält
  - Speichere pro Pixel nur einen *Index* in die CLUT (= kleine Anzahl Bits)
  - Vorteil: spart Speicherplatz & Datenrate(!)
  - Spielt heute keine Rolle mehr, das Prinzip aber sehr wohl! (z.B. bei Kompression)

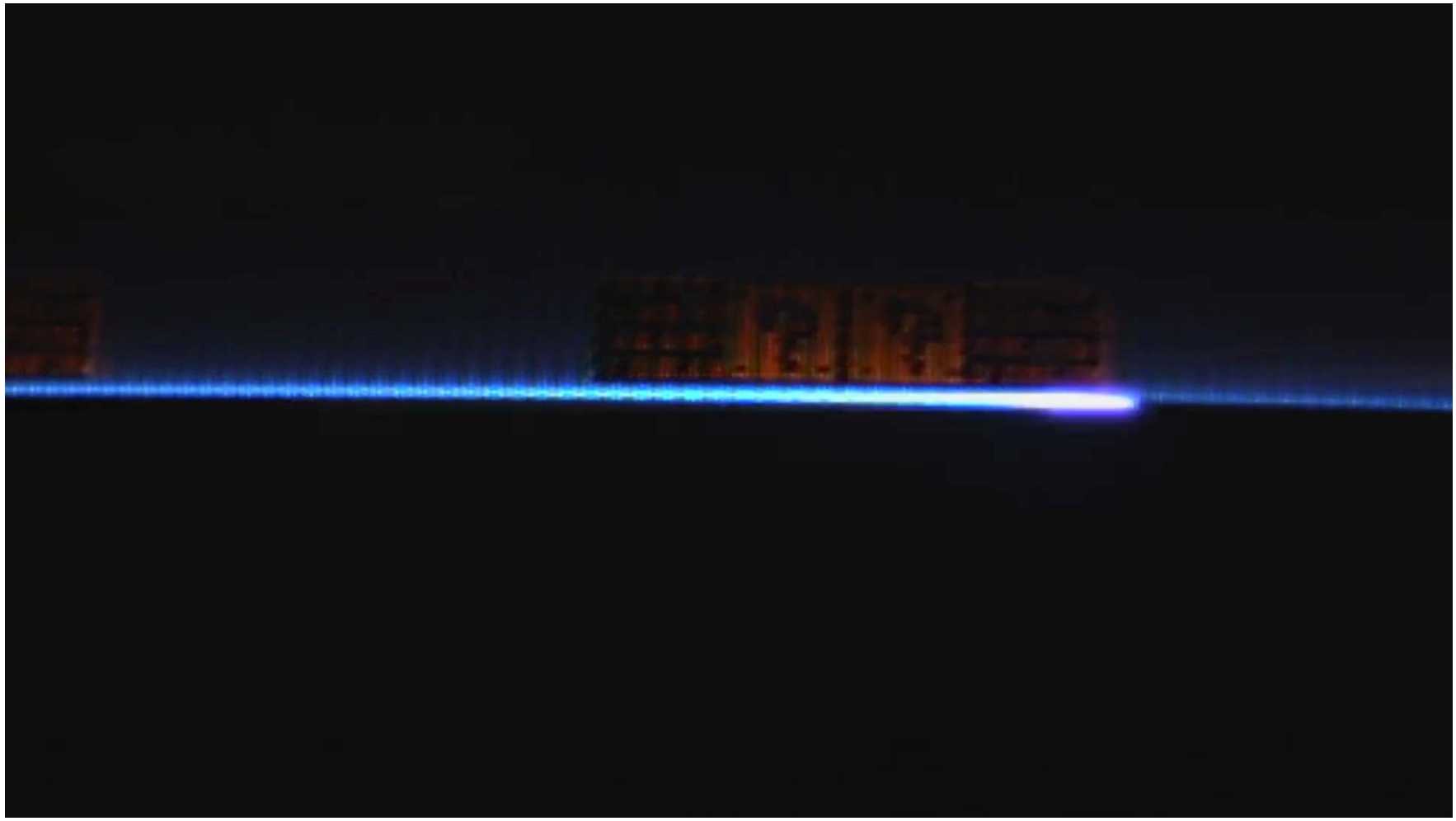






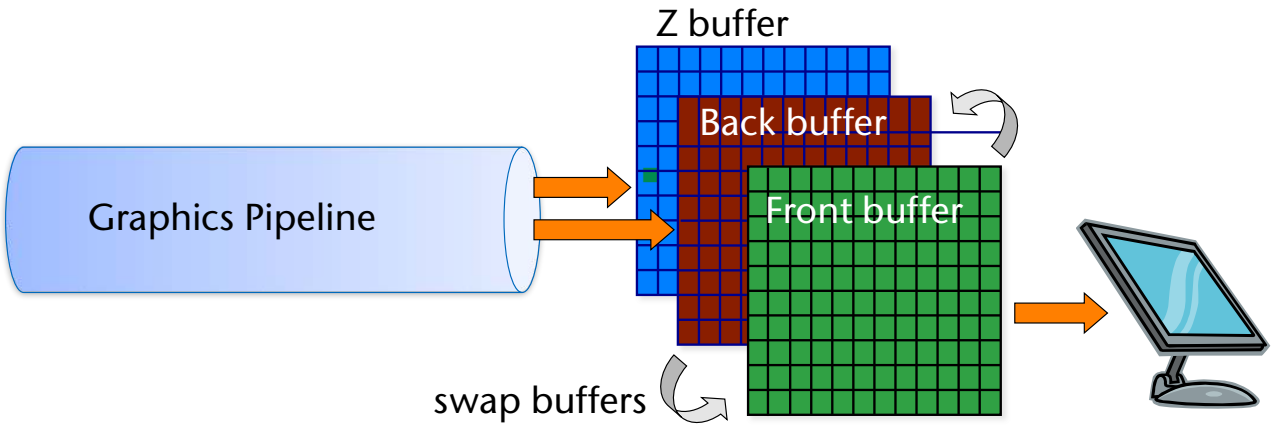
- Beispiel:
  - 8 Bit Farbe pro Pixel
  - 12 Bit breite Color LUT
  - Das ergibt 2^{12} = 4096 unterschiedliche Farben
  - Jeder Pixel durch 8 Bit dargestellt, kann nur 2^8 = 256 Farben verwenden
  - Nehme 256 verschiedene Farben aus den möglichen 4096 und speichere sie in der Color LUT
  - 8 Bit Farbwert eines Pixels indiziert einen Eintrag der Color LUT
  - Die gespeicherte 12 Bit Farbe wird letztendlich angezeigt
- Wird heute nur noch selten gemacht, aber an anderer Stelle (in Algorithmen) taucht dieses Verfahren wieder auf ...

Video Showing Display Scan-Outs



Double-Buffering

- Es gibt noch viele weitere Buffer in einem Framebuffer
- Problem beim Single-Buffering: Flickering
  - Eine Art *race condition*
- Lösung: der Double-Buffer
  - Front Buffer = Color-Buffer, der vom Display gerade angezeigt wird
  - Back Buffer = Color-Buffer, in den gerade gezeichnet ("gerendert") wird



Synchronisation mit Vertical Retrace (VSYNC)

- Man kann zu einem bestimmten Zeitpunkt der Video-Hardware sagen, dass jetzt der Back-Buffer "gelten" soll
  - **`swapbuffers()`** (in OpenGL) = tauscht die Pointers zu den beiden Buffers
  - Die Video-Hardware liest den (neuen) Front-Buffer direkt nach dem Swap aus, beginnend mit dem Pixel im Buffer, wo sie gerade steht
- Wann darf die Applikation tatsächlich den Swap-Buffers durchführen?
- Bei beliebigem Zeitpunkt
  ⟶ screen tearing
- Konsequenz: Swapbuffers muss mit VSYNC synchronisiert werden (außer bei GSYNC/freesync)

Tear point

Tear point

Just FYI (nicht klausurrelevant)

Ein Wort zu "swap buffers" und Synchronisation allgemein

- Die Funktion gibt es nicht in OpenGL, nur in der OS-spezifischen Window-Handler-Library: **`wglSwapBuffers()`, `aglSwapBuffers()`, `glXSwapBuffers`**
- Verwendet man praktisch nie "von Hand" bei Einsatz von high-level GUI-Libraries (Qt, GLUT, GLEW, etc.)
  - Dort liegt die *main loop* (und damit die Kontrolle) immer in der GUI-Library
- Der *Buffer-Swap* muß (normalerweise) mit dem *vertical retrace* (VSYNC) des Monitors synchronisiert werden
- Konsequenz: es kann hohe Zeitverluste durch Synchronisation geben
  - Beispiel: main loop benötigt 1/45 Sekunde = 22 Millisek., Monitor läuft mit 60 Hz → main loop läuft nur mit 30 Hz → die main loop muß am Ende jedes "Applikations-Frames" 33 - 22 = 11 Millisek. warten!

Example of a Video Player on a PC Showing Serious Tearing

Lösungen

1. VSYNC (vertical synchronization):
   Der **`swapbuffers`** erfolgt nur während des VSYNC (früher "Bildaustastlücke"); wartet also bis zum nächsten Auftreten des VSYNC

- Nachteil:
  - Ursprüngliche Framerate F_G des Renderers (z.B. Spiel) wird gedrosselt auf $F'_G < F_G$, so dass $F'_G \mid F_M$, wobei F_M = Framerate des Monitors
  - Judder: bei Film als Input (Film kann nicht "gebremst" werden!)

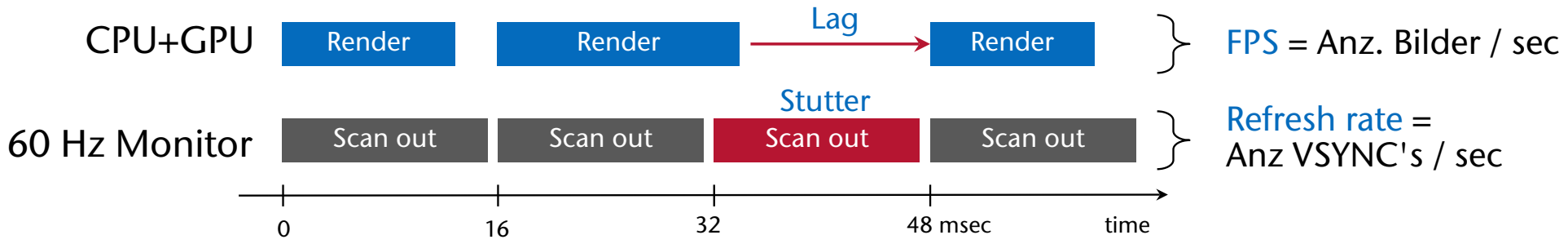
Beispiel für Judder

Which Monitor Showed Stuttering?

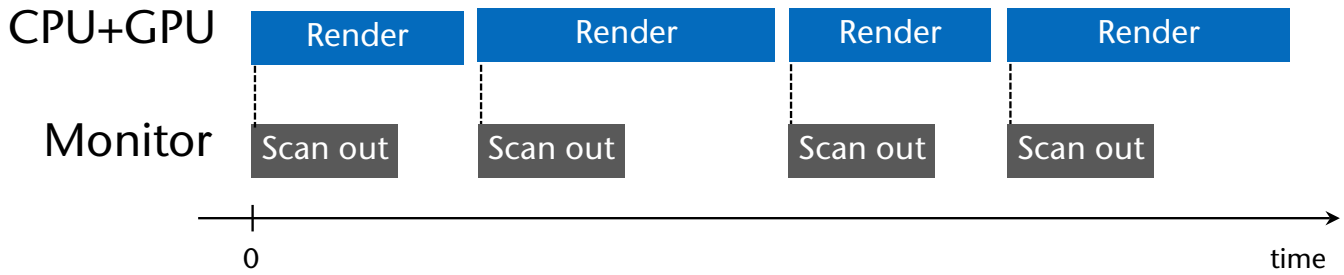
Oder:
www.menti.com/
qyragw8ezp

Fixed Refresh Rate vs Variable Refresh Rate

- Fixed refresh rate (w/ VSYNC): GPU wartet auf VSYNC des Monitors



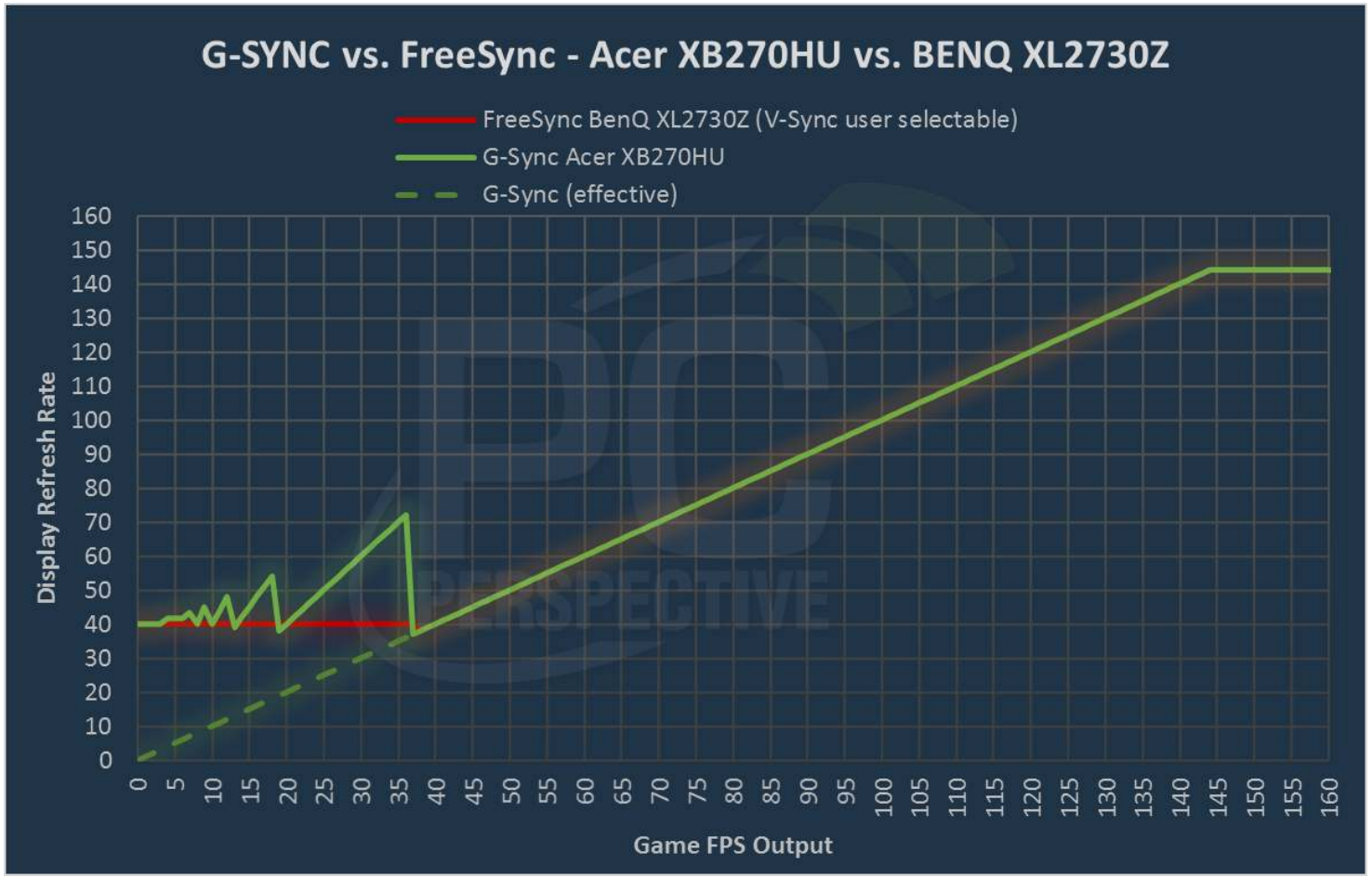
- Variable refresh rate / adaptive sync: Monitor "wartet" auf GPU



Lösung 2

2. Reverse who waits for whom:
   - Monitor waits with vertical retrace, until renderer performs **`swapbuffers`**
   - No problem with LCD's, but spec's for video signals must be adapted!
   - Called G-Sync (Nvidia), FreeSync (used by AMD), Adaptive-Sync, variable refresh rate
   - Seems to be the best solution currently

Messung der tatsächlichen Refresh-Rate

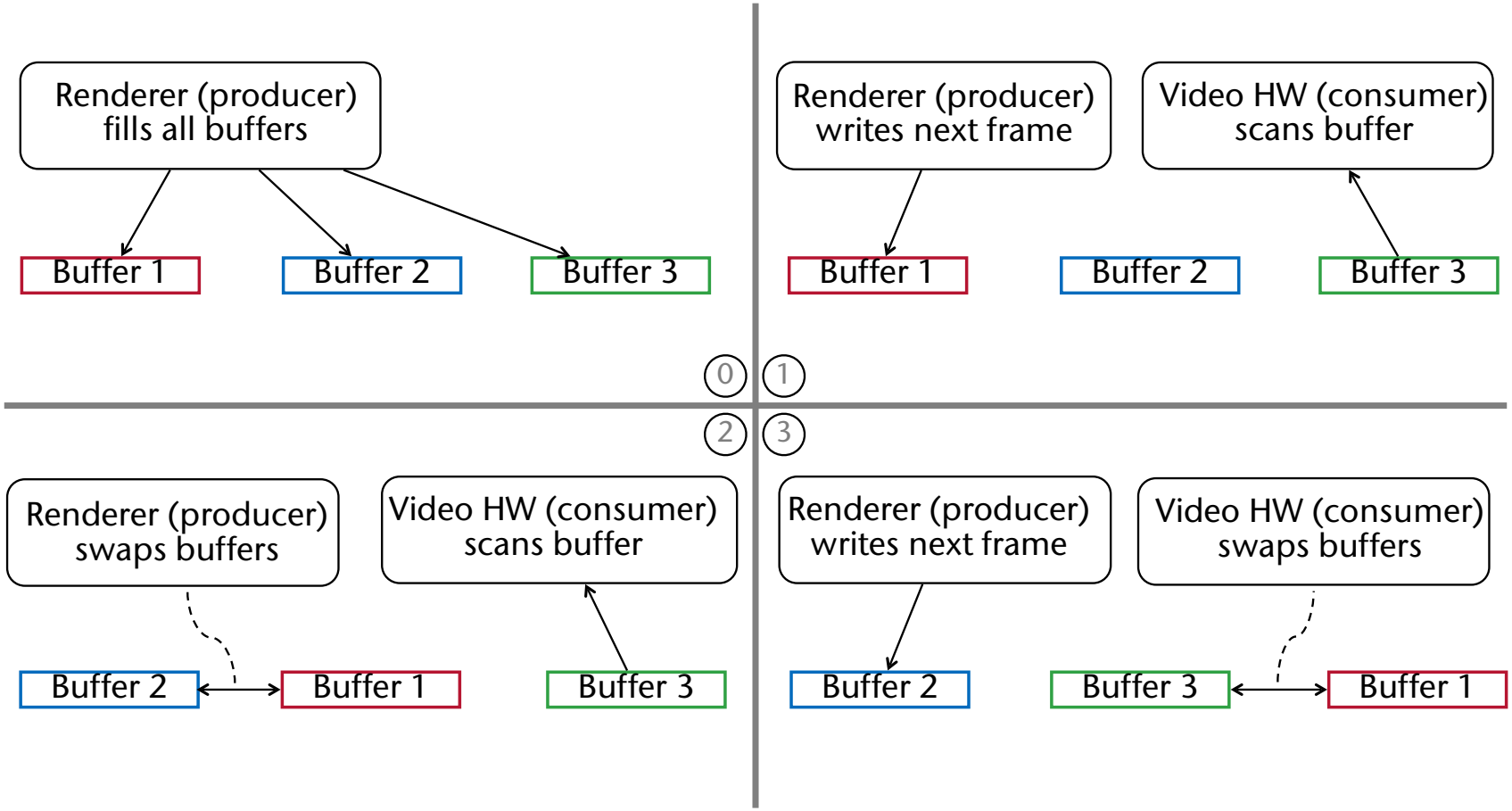


https://www.pcper.com/reviews/Graphics-Cards/Dissecting-G-Sync-and-FreeSync-How-Technologies-Differ

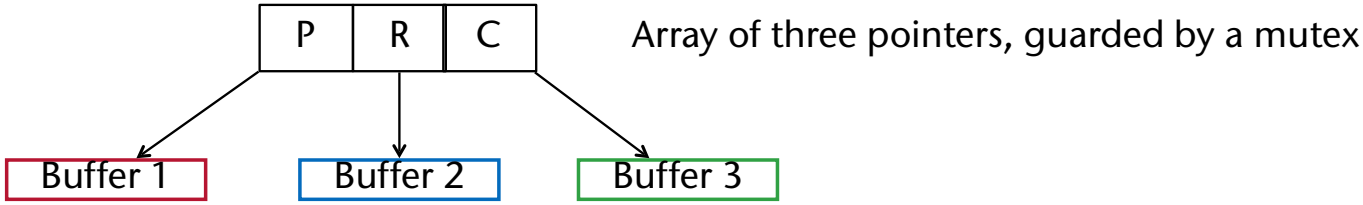
Effects of Tearing and Stuttering vs GSYNC

Lösung 3

3. Triple buffering:



- Array for holding the 3 pointers to the 3 buffers:



- Important: during swapping pointers to the buffers, the renderer (producer) and/or the video HW (consumer) must acquire the lock (mutex) that guards access to the array of 3 pointers!
- Note: the monitor might miss one or more frames, if the renderer is faster than the monitor ⟶ slight stutter?

Weitere Synchronisationen

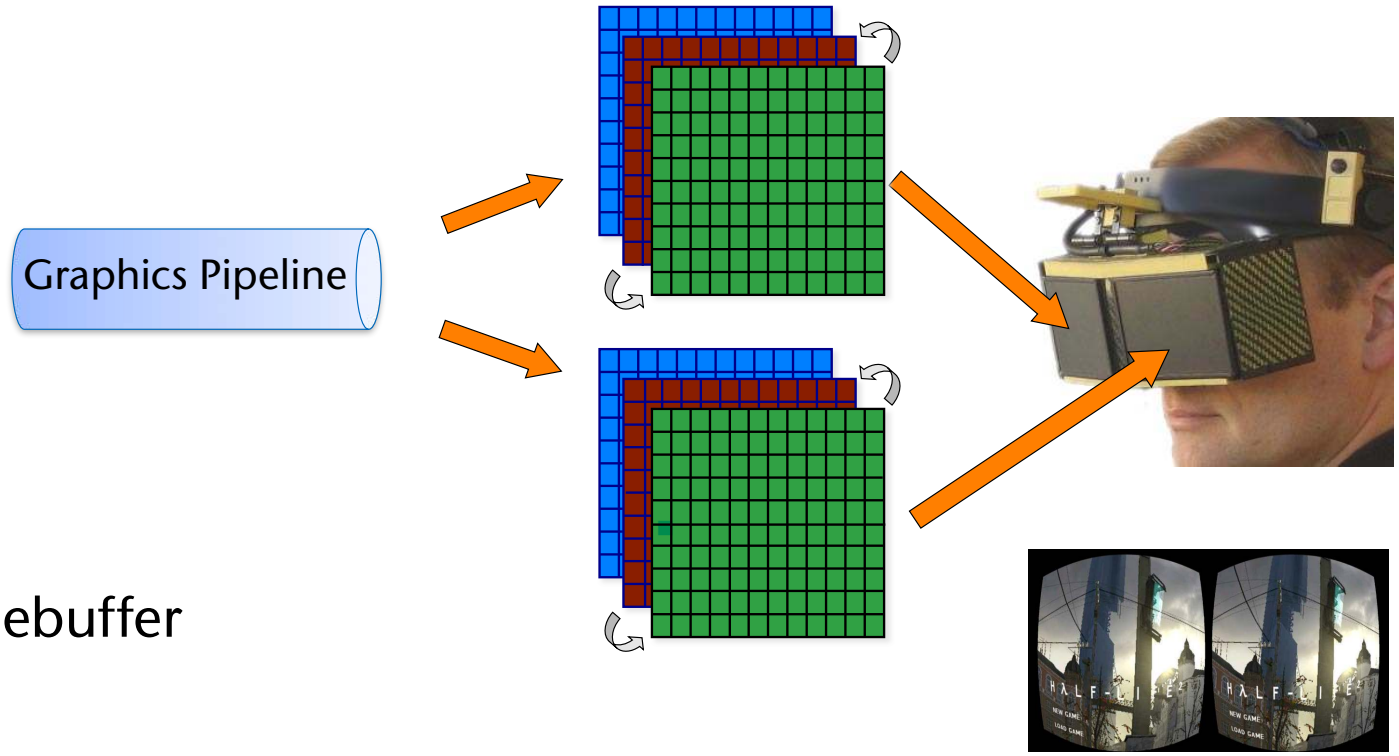
- Bei Setups mit mehreren PCs/Graphikkarten für 1 Display (z.B. Powerwall) muß der Swap-Buffers aller Renderer auf allen PCs miteinander synchronisiert werden → Swaplock
  - Wird typischerweise durch einen Barrier in Software implementiert
- Damit dies das gewünschte Resultat produziert, muß der VSYNC aller Monitore (oder Projektoren) miteinander synchronisiert werden → Genlock
- Fazit: noch mehr Synchronisationsverluste

Ähnliches Problem im Film

- Filme von einer Framerate in eine andere konvertieren
  - Z.B. alte SW-Filme (16 FPS) auf 24 FPS konvertieren
- Exzellentes Beispiel: Peter Jackson's "They Shall Not Grow Old"

Für Stereo-Rendering: Quad Buffers

- Für Stereo- (3D-) Rendering muß man 2 unterschiedliche Bilder generieren: je eines für das linke bzw. rechte Auge
- Eine Lösung:
  2 Front buffers,
  2 back buffers
  (und 2 Z-Buffer!)

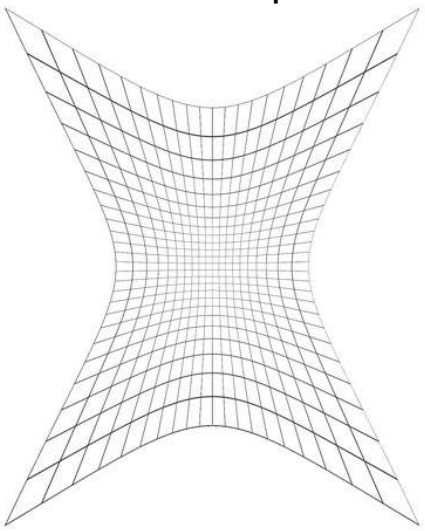


- Andere Lösung:
  "side-by-side stereo" =
  doppelt so breiter Framebuffer

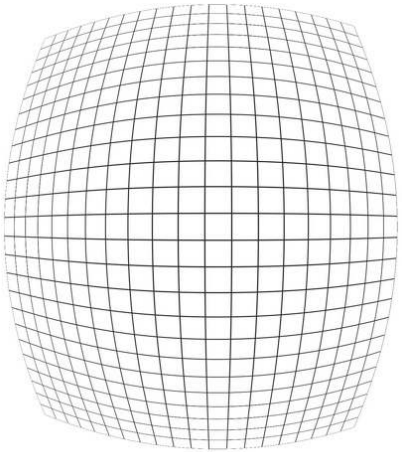
Exkurs: Pre-Warping in HMDs

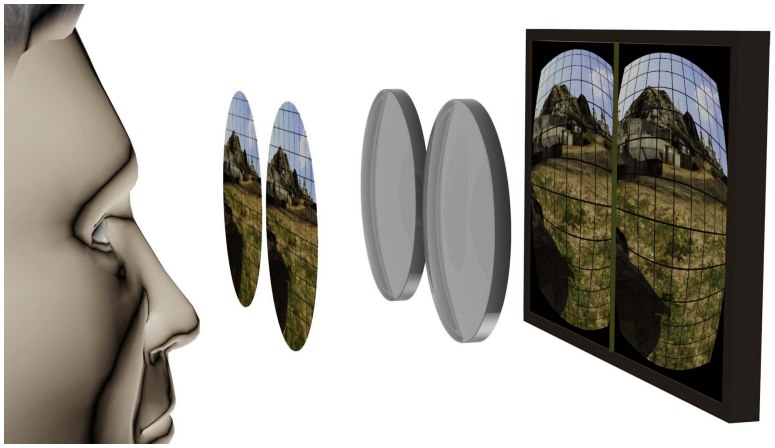
- Heutige Head-Mounted Displays (HMD) verzerren das Bild (*pincushion effect*)
- Lösung: Bild im Framebuffer vorverzerren (*pre-warping*) mittels *barrel distortion*
- Resultat:

What the HMD produces



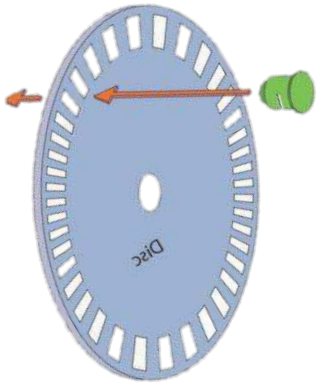
Pre-distortion in renderer





An Important Human Factor: Flicker Fusion

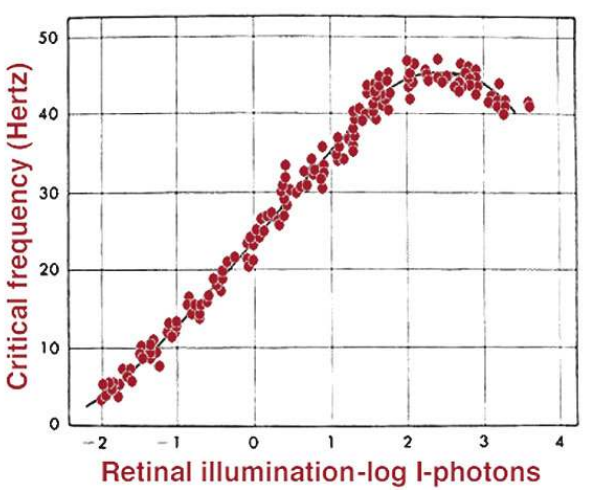
- Experiment:
  - Light source is intercepted by a sectored wheel
  - Low rotation frequency ⟶ sensation = on/off light
  - Medium frequency ⟶ sensation = flicker (rapid fluctuations in brightness)
  - Frequency > critical fusion frequency ⟶ sensation = continuous brightness
    - A.k.a. critical flicker frequency (CFF)
- Factors influencing the critical frequency of fusion:
  - Brightness: higher brightness = higher frequency needed
  - Contrast (high/low brightness)
  - Region on the retina: periphery = lower frequency needed
  - Background/surrounding brightness
  - Arousal, anxiety, exhaustion, age, fatigue (more fatigue = lower frequency)

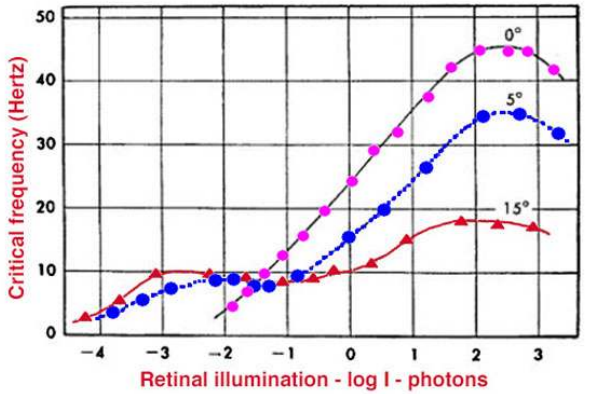


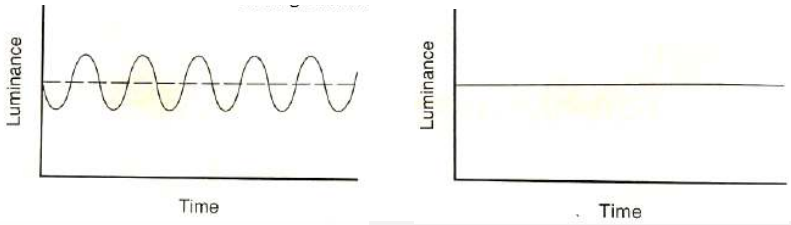
- Ferry-Porter Law: CFF ~ log(luminance)

$$\mathrm{CFF} = a \log(L) + b$$

- Consequence from plot: 60 Hz update rate
- CFF is different for rods and cones :
  - Proportion of rods and cones depends on distance from fovea (see 5° and 15° in plot)
- Talbot-Plateau Law:
  perceived brightness of fused temporally modulated stimulus = perceived brightness of stimulus with *average* luminance







Frameless Rendering

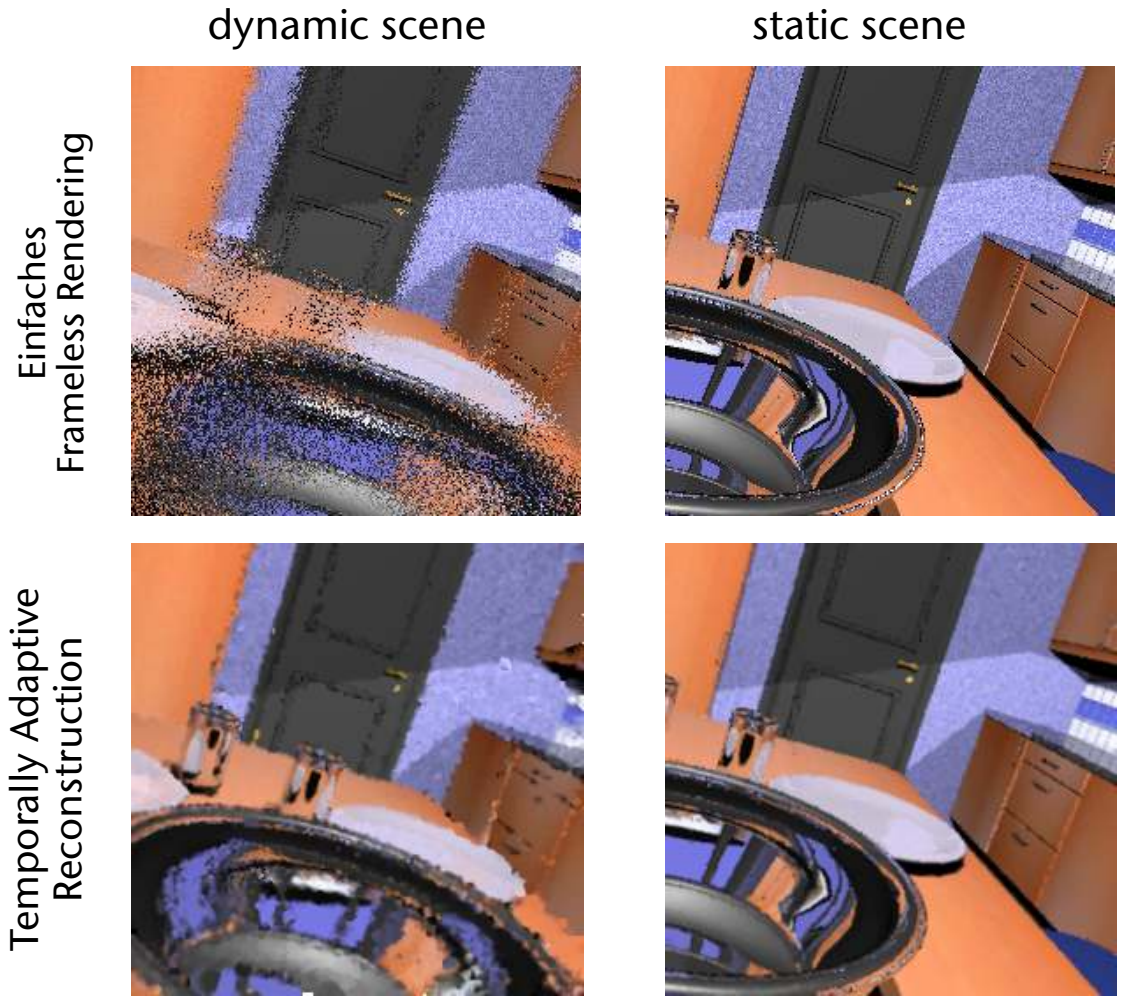
Just FYI

[1994, 2005]

- Annahme: die Anzahl der Pixel im Frame ist der bestimmende Faktor für die Rendering-Zeit (→ "fill limited")
  - Ist z.B. der Fall bei wenigen Polygonen und großem Display; oder bei Ray-Tracing
- Idee: verwende das alte Frame wieder, und erneuere nur einige, zufällig ausgewählte Pixel
  - Konsequenz: es gibt keinen Double-Buffer mehr
  - Wenn die Szene dann statisch wird, werden sukzessive alle Pixel erneuert, und das Bild konvergiert zum "klassisch" gerenderten Bild
- Vorteil: wesentlich geringere Latenz zwischen Kamera-Bewegung und dem Erscheinen eines "neuen Frames" auf dem Display

Beispiele

Just FYI



Bitte nicht durch "look-ahead" spoilern!

Oder:
www.menti.com/
qyragw8ezp

Die Gammakorrektur (hier nur 1 Kanal)

Was ist hier der Unterschied?

Nichtlineare Transferfunktionen

- Begriffe:
  - Die wahrgenommene(!) Helligkeit = eine physiologische Größe
  - Die Intensität I = eine physikalische Größe
  - Dynamikbereich (dynamic range) = Verhältnis max. / min. Intensität

1. Die Nichtlinearität im Auge:
   - Beobachtung: eine Folge von Intensitäten I_0 , $I_{1, \dots,} I_k$ wird als linear wahrgenommen gdw.

$$\forall j : \frac{I_{j+1}}{I_j} \equiv \text{const.}$$

- Aufgabe: *k* Intensitätsstufen I_j so im Intervall I_{min} bis I_{max} verteilen, dass die wahrgenommenen Helligkeitsstufen linear verlaufen
  - In der Praxis oft: $k+1 = 2^8 = 256$ oder $k+1 = 2^{12} = 4096$

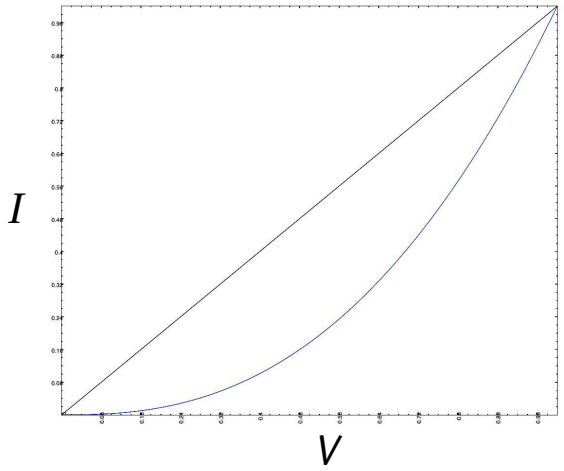
- Lösung: geometrische Folge
  - $I_0 = I_{\text{min}}\,,\ I_1 = r \cdot I_0\,,\ I_2 = r \cdot I_1 = r^2 \cdot I_0\,, \ldots, I_j = r^j \cdot I_0$
  - I_{max} / I_{min} kann man messen $\rightarrow\ r = \left(\frac{I_{\text{max}}}{I_{\text{min}}} \right)^{1/k}$
- Korrektur der wahrnehmungspsychologischen Nicht-Linearität:
  - Gegeben: j = Pixel-Wert im Framebuffer, j = 0, ..., k
  - Bestimme: $I_j = r^j \cdot I_{\text{min}}$
  - Evtl. in Color-LUT ablegen (als Preprocessing / in HW)

2. Die Nichtlinearität im Monitor:

- Beobachtung: bei Eingangsspannung V liefert ein Monitor eine Ausgangsintensität I (an einem Pixel) von

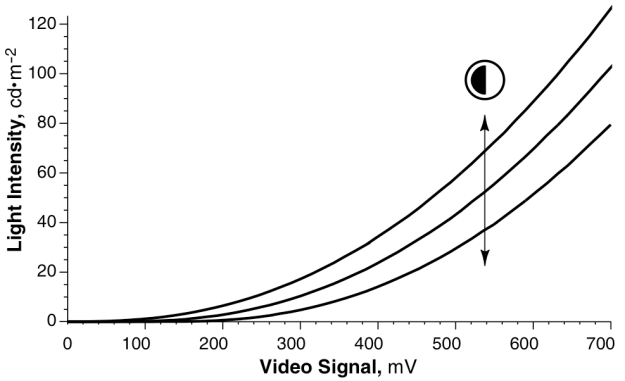
$$I = I_{\max}\left(\frac{V}{V_{\max}}\right)^{\gamma}$$

- Typischer Wert ist $\gamma = 2.5$



- Die Gamma-Korrektur:
  - Gegeben I_j
  - Bestimme: $V_j = \left(\frac{I_j}{I_{\max}}\right)^{1/\gamma} \cdot V_{\max}$
  - Passiert nach der DA-Konvertierung auf Rechnerseite (oder gleichzeitig)

- Bemerkung: "Contrast"-Knopf am Monitor ändert einfach das Gamma des Monitors



- Wahrnehmungspsychologische Korrektur "für Arme":
  - Approximiere $I_j = r^j \cdot I_{\text{min}}$

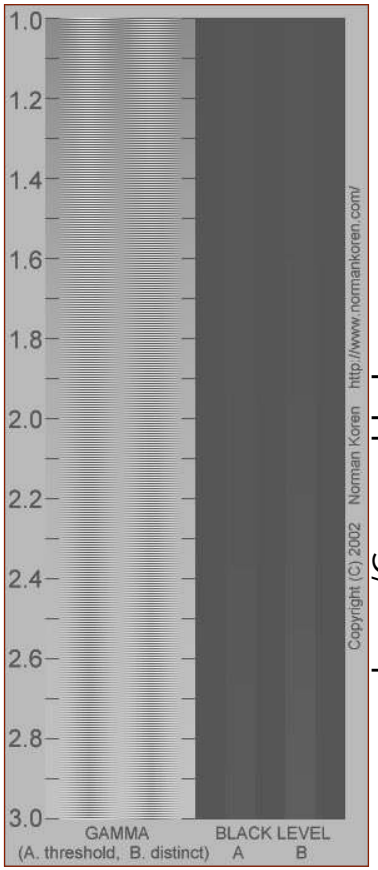
    durch $I_j \approx \left(\frac{j}{k}\right)^{\gamma} \cdot I_{\text{min}}$

  - Damit wird $V_j = \left(\frac{I_j}{I_{\text{max}}}\right)^{1/\gamma} \cdot V_{\text{max}} \approx \left(\frac{\left(\frac{j}{k}\right)^{\gamma} \cdot I_{\text{min}}}{I_{\text{max}}}\right)^{1/\gamma} \cdot V_{\text{max}} = j \cdot c$

Wie bestimmt man das Monitor-Gamma?

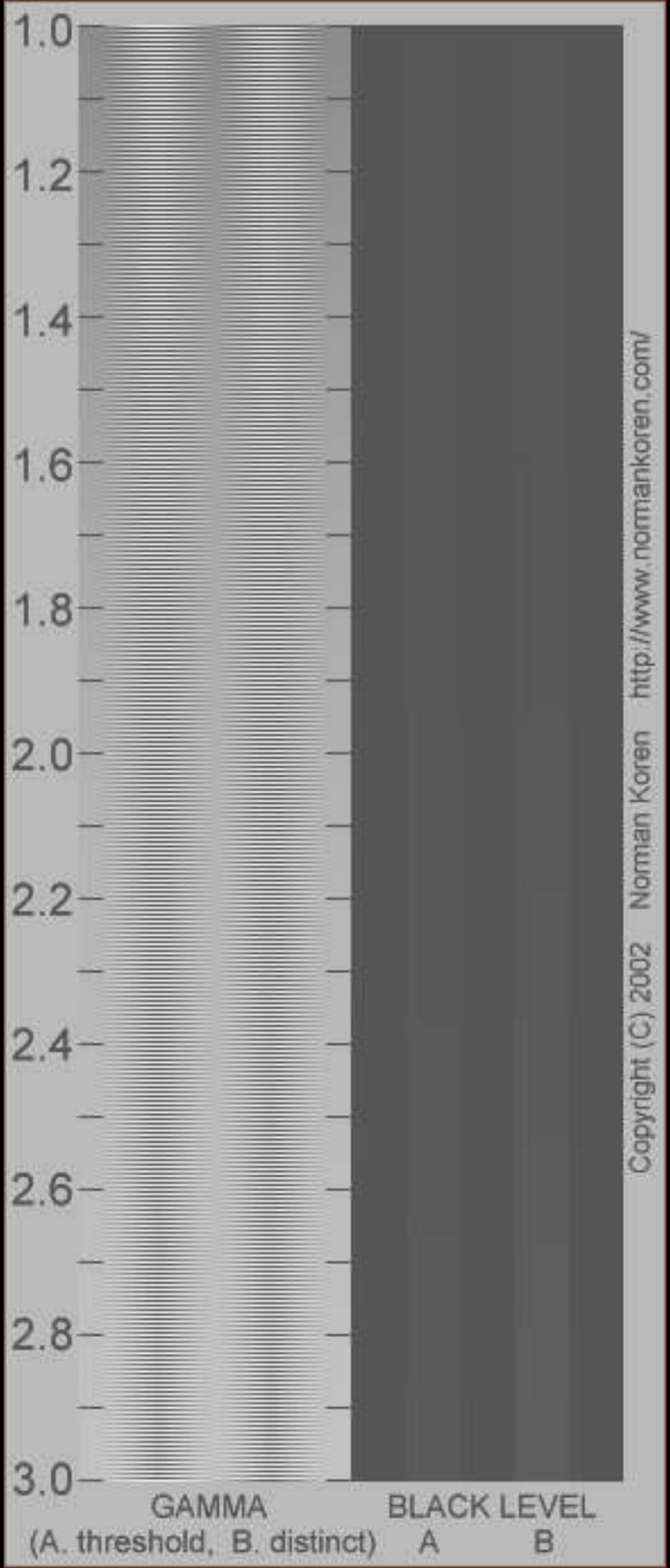
- Achtung: (older / cheap) LCD screens are poorly suited for critical image editing because gamma is extremely sensitive to viewing angle!
- Testbild zur visuellen Bestimmung des aktuellen Gammas des Gesamtsystems
  - Aktuelles Gamma ist dort, wo ein einheitlicher Grau-Level auf einer horizontalen Linie zu sehen ist
  - Die Methode:
    - Schwarze & weiße Pixel werden — unabhängig von γ! — als "Schwarz" bzw. volle Helligkeit wahrgenommen
    - Auge bildet aus der Entfernung daraus einen wahrnehmungspsychologischen(!) Mittelwert
    - Finde den Pixel-Grauwert a , so daß $\frac{1}{2} = a^{\gamma}$

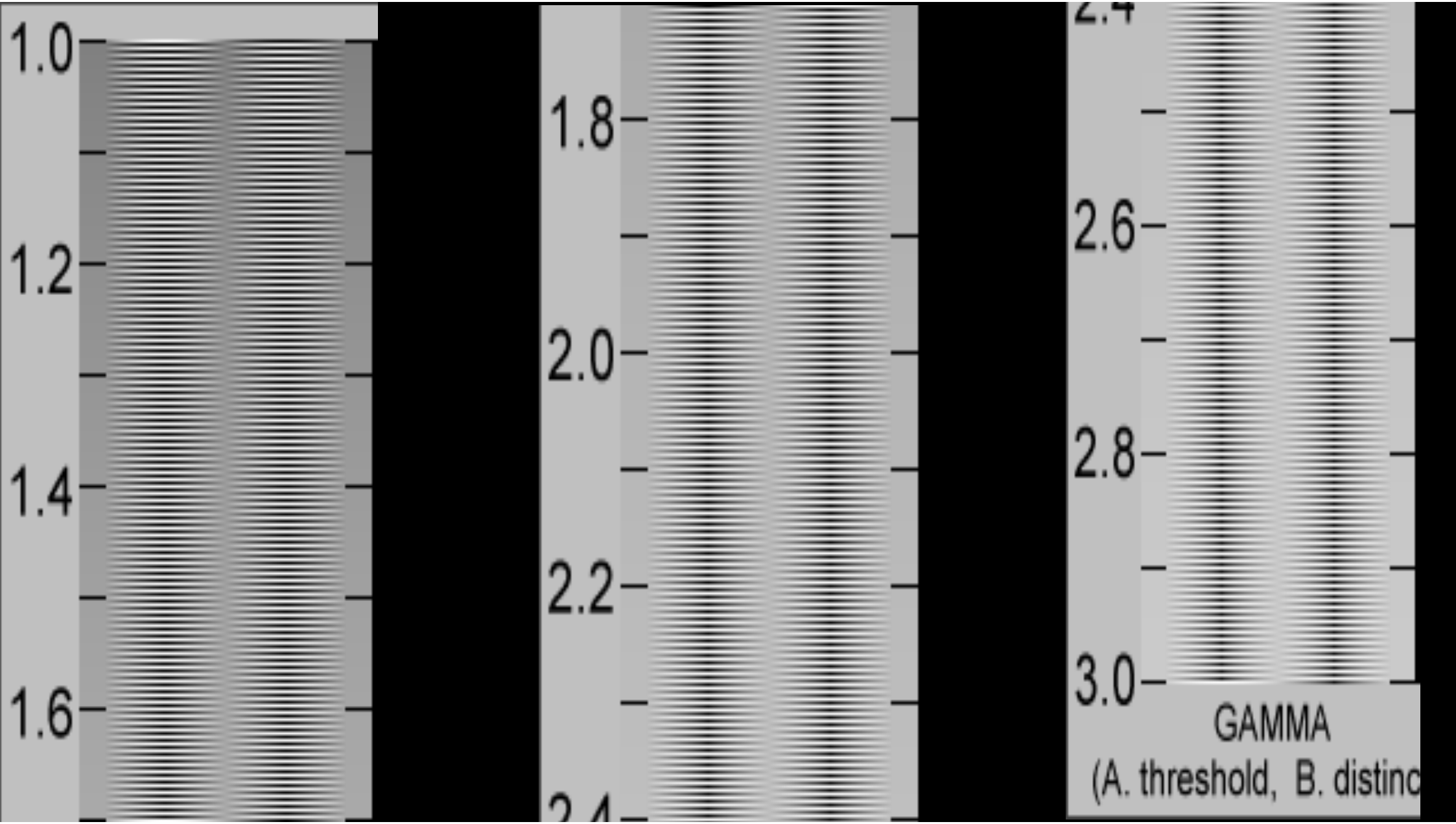
      Führe das für verschiedene γ durch und eiche damit die Skala



demos/Gamma_black_new.png

S.a. VL-Homepage


1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2.8
3.0
GAMMA
(A. threshold, B. distinct)
BLACK LEVEL
A
B
Copyright (C) 2002 Norman Koren http://www.normankoren.com/


1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2.8
3.0
GAMMA
(A. threshold, B. distinc

Das Chaos & die Lösung

- Das Chaos (jahrzehntelang):
  - Verschiedene Monitor- und System-Gammas
  - Unklarheit darüber, an welcher Stelle in der Pipeline die Gamma-Korrektur gemacht werden soll:
    - Im Bild? (manche Tools haben das Bild schon gamma-korrigiert abgespeichert!)
    - In der Software? (= vor dem sog. Frame-Buffer, d.h., im Browser, Video-Spiel, Photoshop, ...)
    - In der Graphikkarte? (= beim Auslesen und Konvertieren des Frame-Buffers)
    - Im Monitor? (= vor der Ansteuerung der CRT-Kanone / der LCD-Transistoren)
- Die Lösung: *ICC Color Profiles*
  - Speichern Info, ob das Bild schon gamma-korrigiert wurde, und – falls ja – mit welchem Gamma (speichern noch viel mehr, u.a. den Farbraum)
  - Kann man in moderne Bildformate einbetten (z.B. JPG, TIF, PNG)