Grundlagen der Informatik

Grafische Benutzeroberflächen und Grafikprogrammierung

Prof. Dr.-Ing. Thomas Wiedemann
Fachgebiet Informatik / Mathematik

Überblick

- Einführung und Motivation zu grafischen Systemen
  - Historie
  - Anforderungen und Voraussetzungen
  - Generelle Prinzipien der Hardware (Grafikkarten etc.)
- Grafisch-interaktive Betriebsysteme
  - Windows-Programmierung - Grundlagen
  - Überblick zu MFC und Windows-API
  - Komponentenbasierte Programmierung
- Grafikprogrammierung
  - Darstellung der grafischen Grundelemente (Linie, Kreis, etc.)
  - Komplexe 2D- und 3D-Visualisierung (CAD/Multimedia)

Quellen :– Computergrafik - Lehrbücher

-http://de.wikipedia.org/wiki/Kategorie:Computergrafik

Einführung zu grafischen Systemen

Historie :

- bis etwa 1985 kein umfassender Grafikeinsatz infolge zu geringer Hardwareleistung
  - Arbeit mit alphanumerischen Anzeigen (Terminals) mit 80 Zeichen auf 40 Zeile (= 3200 Bytes pro Bildschirm) - entspricht dem MS-DOS-Textmodus
  - eine gleiche Grafikanzeige hätte bei bei einem 16x8 Punktraster pro Buchstabe das 128-fache an Speicherplatz und Übertragungszeit benötigt
- für CAD-Aufgaben extrem teure Spezialrechner (ab 50.000 DM)
- Speziallösungen mit sogenannten Vektorgrafikrechnern (Prinzip des Oszilloskops – Darstellung von Linien und Kurven durch Spannungsverläufe – begrenzt in der Anzahl der Linien , keine Texte möglich)

Start der Massenproduktion von grafikfähigen Rechnern ab 1980

- ab Anfang der 80er Experimentalrechner von Xerox
- erster echter Rechner mit Grafik von APPLE 1982 (Apple II) , Apple war lange Zeit der Vorreiter in der Grafik-EDV und ist deshalb auch heute noch der Maßstab von Designern und Grafikern
- die ersten IBM-PC´s konnten nur mit speziellen Grafikkarten einfache Grafiken von 320x200 Punkten darstellen

Motivation zu grafischen Systemen

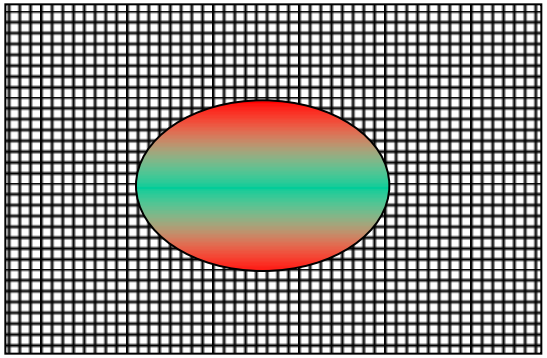
Warum sind Grafiken besser als reine Textdarstellungen ?

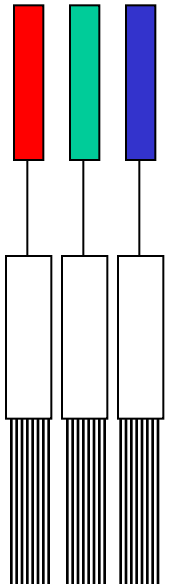
- höhere Flexibilität der Darstellung (der DOS-Zeichensatz beschränkt die mögliche Anzahl von Zeichen auf 256 oder weniger)
- grafische Symbole können vom menschlichen Gehirn leichter erfaßt und verarbeitet werden
- komplexe Zusammenhänge sind als grafische Diagramme leichter darstellbar als in Tabellenform
- bestimme Anwendungen wie Technisches Zeichen oder Anlagenplanung sind auf Grafiken angewiesen+
- Wunsch nach Multimedia : Grafik + Musik + Video

Anforderungen und grundlegende Funktionsweise

Hardware (Grafikkarte)

- Grafikmodul abgetrennt als Grafikkarte - übliche Abk. VGA - Video Graphics **A**dapter
- einzelne Adressierung jedes einzelnen Bildpunktes auf dem Bildschirm
- pro Bildpunkt Helligkeit oder Farbe in Abhängigkeit von der verfügbaren Bitzahl pro Bildpunkt:
  - 1 bit - nur schwarz/weiß
  - 4 bit - 16 Graustufen oder max. 16 Farben
  - 8 bit - 256 Graustufen oder max. 256 Farben
  - 16 bit - 64k Farben
  - 24 bit - Echtfarben mit jeweils 8bit pro Primärfarbe ROT/GRÜN/BLAU (RGB-Kodierung)
  - bei wenigen Bits erfolgt auch eine Übersetzung von 4 oder 8 bit in spezifische Farbwerte

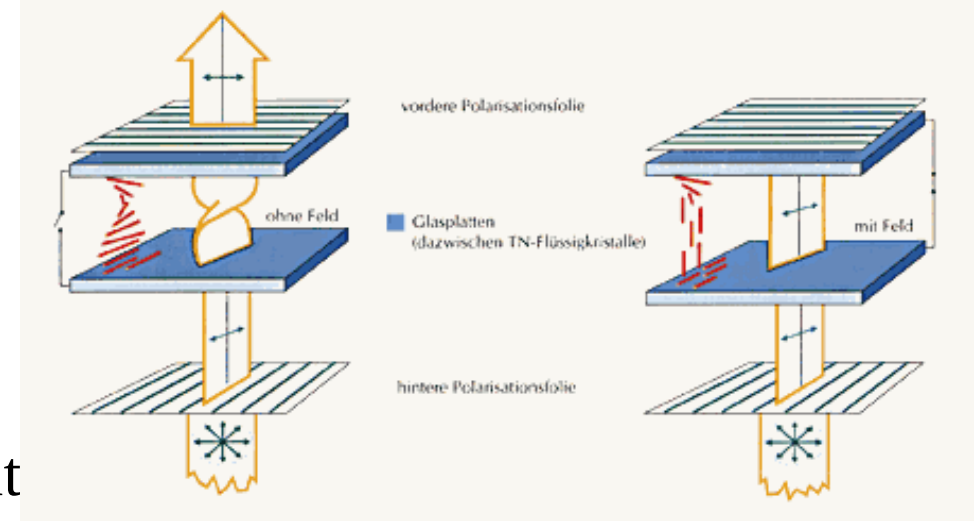


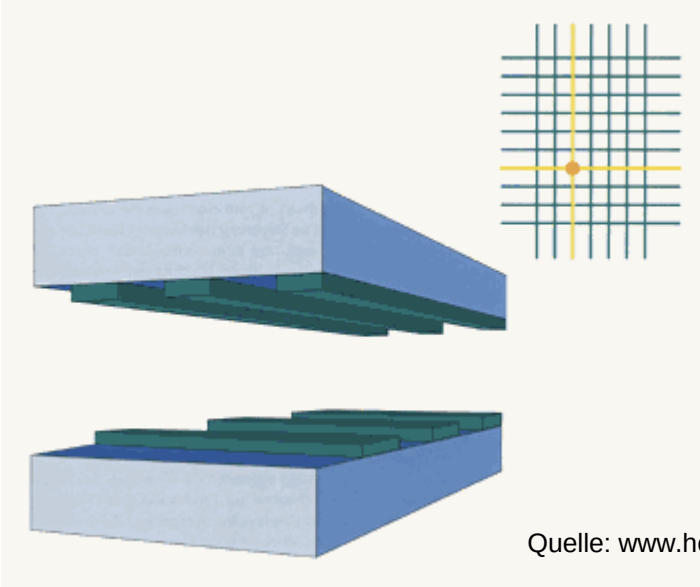


Anforderungen und grundlegende Funktionsweise

Hardware LCD-Monitor

- Ausnutzung vom Polarisations-effekten in Flüssigkristallen bei elektrischer. Spannung
- bessere Farbechtheit durch mehrere Schichten (*Double-Super-Twist-Technik* - DSTN) mit Kontrast bis ca. 15:1
- Spannungsansteuerung entweder über einfache, gekreuzte Leit-ungen (Probleme mit Kontrast oder Geschwindigkeit) oder durch aktive Elemente (Thin Film Tran-sitors =TFT=> Kontrast bis 70: 1)
- Helligkeitssteuerung über Spannungshöhe oder Pulsbetrieb



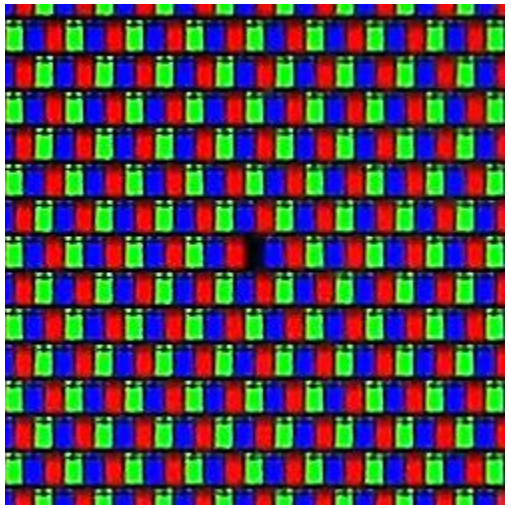


Quelle: www.heimkino.net

Farbdarstellung auf LCD-Bildschirmen

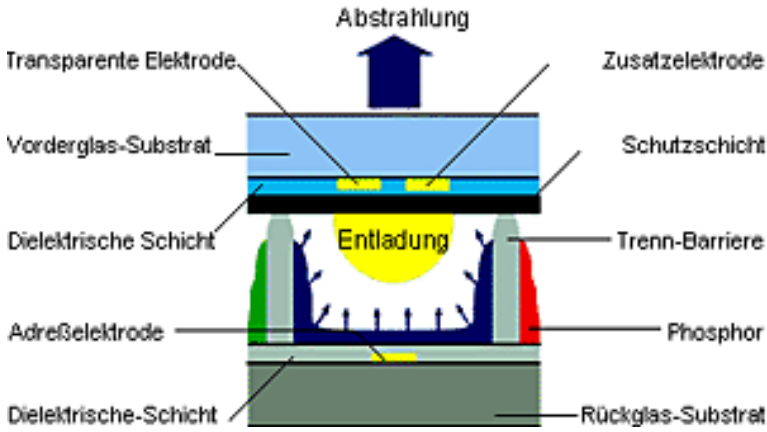
Hardware LCD-Monitor

- Anordnung mehrer Pixel mit Farbfiltern in Delta- (Bsp.) oder Längsanordnung
- Einzelne Pixelfehler (Aus, oder immer leuchtend) sind je nach Preisklasse möglich



Andere Optionen zur Bilderzeugung:

- Plasmabildschirm erzeugt direkt Licht durch elektr. Entladung in Kammern mit Leuchtstoffen
- Beamer mit LCD-Filter oder Spiegel-Array´s
- Elektrisches Papier mit ferroelektrischen Kügelchen und nur einmaliger Ansteuerung

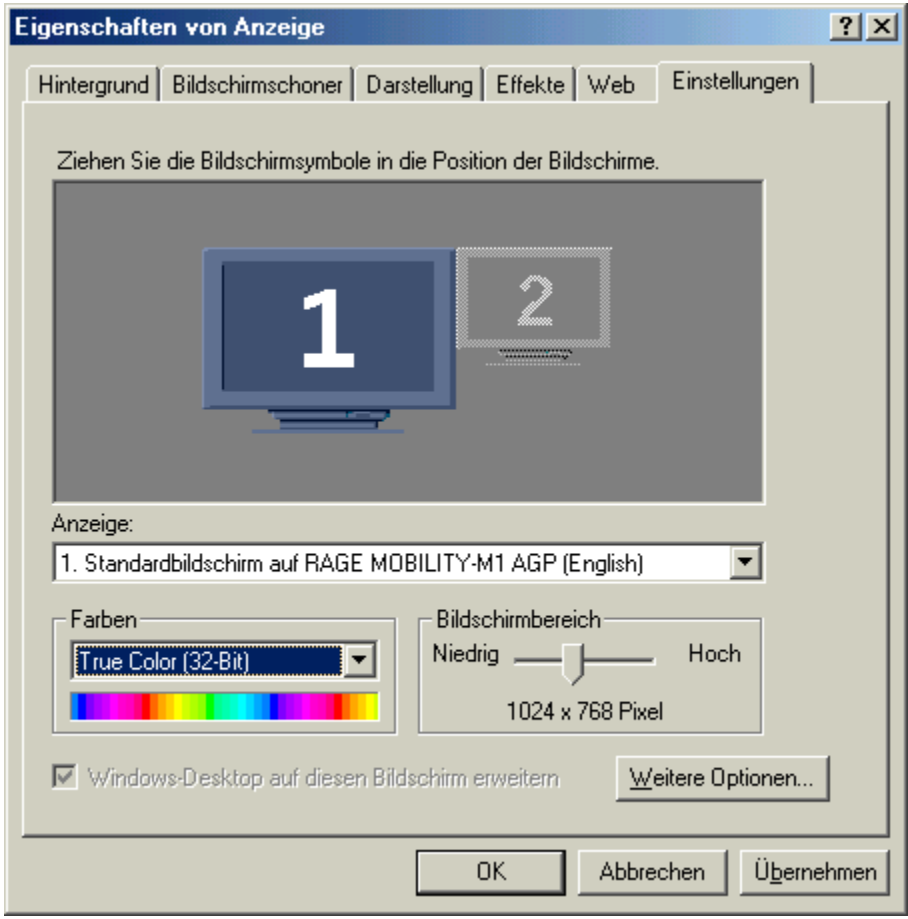


Quelle: www.heimkino.net

Anforderungen und grundlegende Funktionsweise II

Hardware (Grafikkarte)

- Anzahl der Bildpunkte in X und Y ergibt die **Auflösung (AL) :**
  - 320 x 200 - veraltet
  - 640 x 480 - Minimale VGA-AL
  - 800 x 600 - Standard VGA
  - 1024 x 768 - Super VGA (SVGA)
  - 1280 x 1024 - XVGA
  - höhere Auflösungen in Abhängigkeit von der Grafikkarte
- Einstellung über Bildschirm-eigenschaften in starker Abhängigkeit von den Grafikkartentreibern



Bildwiederholfrequenz - Anzahl der Bilder pro Sekunden

- 25 (50) Hz beim Fernsehen (flimmert und ermüdet), bei LCD kein Flimmern
- größer als 72 Hz für ergonomisches Arbeiten

- Merke: Höhere Frequenz erfordert schnellere Bauteile und ist damit teuerer !

Grafisch-interaktive Betriebsysteme

Bis Ende der 80 er Jahre:

- grafische Anwendungen nur durch spezielle Berücksichtigung der konkreten Grafikkarte (Laden spezieller Treiberprogramme)
- **keine Unterstützung durch das Betriebssystem**
- **keine Standards zur Darstellung von Menüs und Bedienelementen**
- **kein Druck von Grafikdaten möglich (nur mit speziellen Treibern für jedes Programm !)**

Situation wurde Mitte der 80er Jahre erkannt :

- erstes echtes grafisches Betriebssystem durch Apple (Macintosh)
- GEM und andere Versuche auf MS-DOS-Basis (erfolglos)
- Windows 1.x und Windows 2.x von Microsoft waren Flops
- erst Windows 3.x brachte sinnvolle Ergebnisse
- parallele Entwicklungen im UNIX-Bereich mit X-Windows
- Konkurrenzprodukte wie OS/2 von IBM waren nicht erfolgreich

Eigenschaften grafisch-interaktive Betriebsysteme

Vorteile:

- einheitliche Ansteuerung von Grafikkarten, Druckern und Eingabegeräten
- grafische Objekte (Symbole, Kästen, etc.) statt Textkommandos
- Einsatz der Maus (u.ä.) für die Bedienung
- einheitliches Bedienkonzept : gleiche Grundstruktur der Menüs und der Bediendialoge, gleiche Bedienelemente
- übersichtlichere Darstellung von komplexen Datenstrukturen
- "What you see is what get" - Ansatz bei der Textverarbeitung und beim Grafikdesign

Nachteile

- mehrfach höhere Anforderungen an die Hardware (bis Faktor 10^3)
- sehr komplexes und relativ fehleranfälliges Betriebssystem
- starke Abhängigkeit vom Hersteller des BS (Microsoft / Apple)

Programmierung grafisch-interaktiver Betriebsysteme

bei traditioneller Programmierweise :

- um Faktor 2 bis 10 aufwendiger
- bis zu 300.000 Quellcodezeilen für einfachste Programme
- extrem fehleranfällig und schlecht zu warten

Erste Lösungsoption:

- größere Bibliotheken der Softwarehersteller
- die wichtigste: **MFC - Microsoft Foundation Classes**
  - umfaßt ALLE Windowsfunktionen
  - erlaubt auch den Zugriff auf andere Programme (z.B. Daten von Excel oder Winword direkt holen)
- MFC-Programmierung stellt immer noch hohe Ansprüche
  - relativ komplizierte Funktionsaufrufe
  - komplexe Abläufe
- Vereinfachung durch objektorientierte Programmierung (siehe VL 3.Sem.)

Anforderungen an eine grafische Benutzungsschnittstelle

Folgende Merkmale werden von grafischen BN gefordert :

- **Aufgabenangemessenheit**
- **Selbstbeschreibungsfähigkeit**
- **Steuerbarkeit**
- **Erwartungskonformität**
- **Fehlertoleranz**
- **Individualisierbarkeit**
- **Lernförderlichkeit**

- In der EU sind die Anforderungen in der Norm EN ISO 9241 geregelt.
- Relevant für Pflichtenhefte und Softwarebewertungen
- Zusätzliche Regeln gelten für die Umsetzung von Benutzungs-schnittstellen für Web-Applikationen und deren Evaluation im Rahmen der Benutzbarkeit.

Überblick zu grafischen Benutzungsschnittstellen

- **BS OS/2 von IBM -** zu Beginn sehr gute Konkurrenz zu Windows, danach infolge Mangel an Anwendungsprogrammen erfolglos

- **X-Window-Systeme (X oder X11) für UNIX-Betriebssysteme –** in den 80er Jahren für verteilte grafische Anzeige unter Unix: - X-Windows-Clients können Informationen von einem entfernten X-Windows-Server anzeigen
- **K Desktop Environment** (**KDE**) für Unix-ähnliche Betriebssystem (in fast allen LINUX-Distributionen enthalten) mit eigenem Set an Verwaltungsprogrammen und Hilfstools

- **BS MS Windows (aktuell Version Win 7..10) :** stark auf Grafikkartenbeschleunigung und 3D-Effekte ausgerichtet, erstmalig vollständig vektororientierte Darstellung der Bildschirmelemente

Typische Algorithmen der Computergrafik

Vorwort:

- Computergrafik wird teilweise als eigenständiger Kurs gelehrt
- insbesondere 3D-Computergrafik erfordert sehr große Softwarelösungen und erlebt gegenwärtig noch eine sehr rasche Entwicklung
- für 3D-Grafiken werden meist bereits vorhandene Subsysteme oder Bibliotheken eingesetzt (Eigenentwicklung kaum sinnvoll)
- am Beispiel der 2D-Grafiken sollen einige typische Algorithmen vorgestellt werden
- 2D-Grafikalgorithmen finden häufig auch direkte Anwendung in technischen Systemen und Maschinen
  - CNC-gesteuerte Bearbeitung von Teilen nutzt zur Steuerung des Werkzeugs die gleichen Algorithmen (+Geschwindigkeitssteuerung)
  - Zielsuchsysteme bei Militär verwenden geometrische Algorithmen zur Kursbestimmung und Zielerkennung

Zeichnen von Punkten

- einfachste Operation
- alle nachfolgenden Computergrafiken bauen darauf auf

Grundprinzip

- im Speicher der Grafikkarte wird die Bitdarstellung eines Punktes entsprechend der gewünschten Farbe gesetzt bzw. geändert
- je nach Art und Auflösung der Grafikkarte kann dies eine Bit-Operation oder eine mehrere Byte umfassende Operation sein
- der Status von Punkten kann auch abgefragt werden

Zusätzliche Optionen

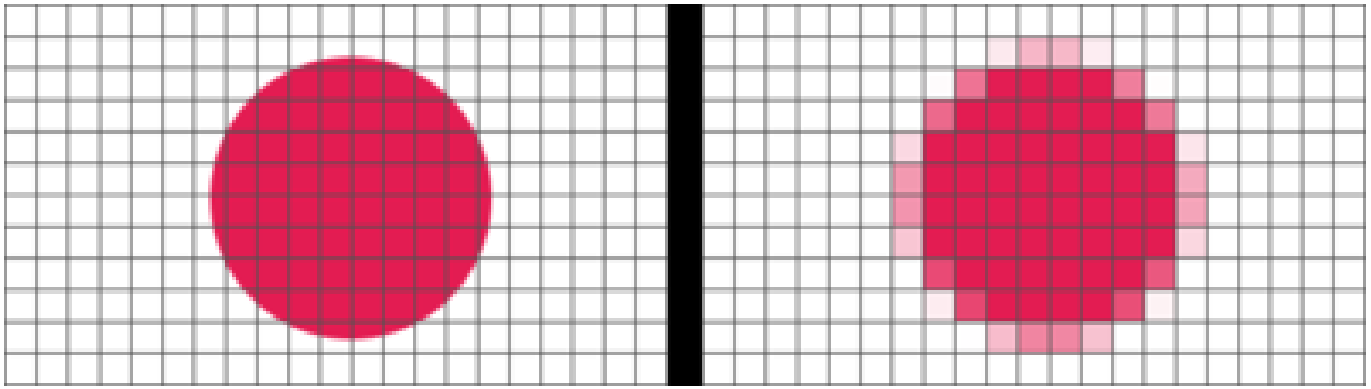
- Invertierung des Hintergrundes durch XOR-Verknüpfung für Mauszeiger oder andere, temporäre Anzeigen
- Definition transparenter Grafikanteile durch Verwendung einer zusätzlichen Filteroperation (Bsp.: in GIF-Grafiken kann eine Farbe als Transparenzfilter definiert werden, Punkte mit dieser Farbe werden nicht gezeichnet sondern es wird der Hintergrund beibehalten)
- moderne Grafikkarten führen das Punktsetzen autonom durch

Probleme bei der Rasterdarstellung

- die endliche Anzahl der Bildpunkte erzeugt Probleme bei nicht rechtwinklig verlaufenden Objekten.
- bei einfacher JA/Nein-Entscheidung bzgl. des Setzens eines Bildpunktes entstehen sehr grobe Verläufe

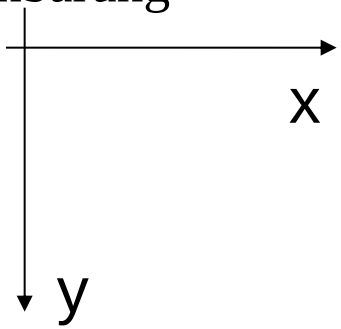
Lösungen

- Durch Helligkeits- oder Farbabstufungen am Rand wird der Übergang abgestuft und es entsteht ein besserer Bildeindruck (Anti-Alias) im Speicher der Grafikkarte wird die Bitdarstellung eines Punktes entsprechend der gewünschten Farbe gesetzt bzw. geändert



Koordinatensysteme

- Punkte und alle nachfolgenden Grafikobjekte benötigen eine Vereinbarung zur zahlenmäßigen Abbildung der Positionen
- **Zielsystem** ist das Raster der Grafikkarte und dessen Maße ändern sich mit der Auflösung !!!
- Problem: Y-Achse der Grafikkarte zeigt nach unten (historisch entstanden durch Auslesen des Grafik-speichers von kleinen (oben) zu großen Adressen



- Anwendungssysteme definieren beliebige Darstellungsmaßstäbe
- Wandlung von Anwendungsmaßstab = "Weltkoordinatensystem" zu Grafikkartensystem notwendig
- bei allen nachfolgenden Beispielen soll diese Berechnung bereits vorab durchgeführt wurden sein:

$$X_{Grafik} = (X_{Welt} - X_{Welt0}) * Zoomfaktor_x + X_{Grafik0}$$
$$Y_{Grafik} = Y_{Grafik0} - (Y_{Welt} - Y_{Welt0}) * Zoomfaktor_y$$

- $X_{Grafik0}$ und $Y_{Grafik0}$ verschieben das Anzeigefensters ($Y_{Grafik0}$ = Ymax)
- $Zoomfaktor_x$ und $Zoomfaktor_y$ sind abhängig von der Auflösung und der gewünschten Darstellungsgröße (meist gleich groß, sonst Verzerrung)

Zeichnen von Linien

- Linien sind die am häufigsten verwendeten Grafikprimitive
- gegeben meist als Punktpaar oder Anfangspunkt mit Winkel+Länge
- auch Kreise und Kurven verwenden Linienstücke (siehe Kreise)
  - müssen besonders performant sein
  - müssen trotz Raster ein möglichst gutes Linienbild zeigen
  - müssen auch bei verschiedenen Winkeln möglichst gleich dick aussehen

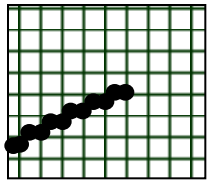
1. Lösungsansatz mit Geradengleichung

- Ausgehend von der Geradengleichung und x_1,y_1,x_2,y_2,

  $y = m * x + a$

  werden a und m berechnet.
- Ausgehend vom Startpunkt x0 werden die y-Werte schrittweise berechnet :
- for $(x = x_1 ; \; x <= x_2 ; \; x = x+ 1)$

  { y = m * x + a ; setze_punkt(x,y); }



- Problem: Bis zu einem Winkel von 45° sieht das Ergebnis gut aus, danach ergeben sich Lücken ! Lösung ? Wie verhält sich der Algor. in anderen Quadranten ?

Zeichnen von Linien – 1. Verbesserung

- Multiplikation ist relativ aufwendig

Verbesserter **Lösungsansatz :**

- erster Algorithmus ändert den X-Wert beim Zeichnen in Grafikspeicher immer nur um 1
- damit erhöht sich der Y-Wert aber auch nur immer um m !
- Algorithmus kann vereinfacht werden zu :

```
y =  a ;
for  (x = x1 ;   x <= x2 ;   x = x+ 1)
     {   y = y + m ;   setze_punkt(x, trunc(y) );  }
```

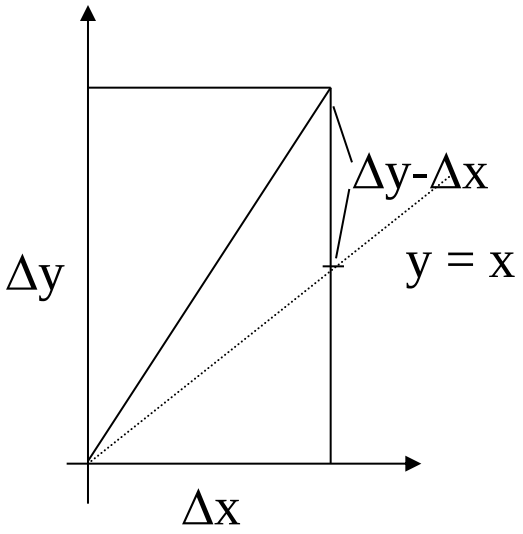
- dadurch Einsparung einer Multiplikation
- diskutierte Fallunterscheidungen in Quadranten bleiben bestehen

Zeichnen von Linien mit Bresenham-Algorithmus

- Standardverfahren arbeitet mit float-Werten und ist dadurch relativ langsam
- Grafikkarte verwendet generell nur ganze Zahlen
- Bresenham schlug 1965 Linienalgorithmus allein mit ganzen Zahlen vor

Lösungsansatz :

- Bresenham schlug 1965 einen Linienalgorithmus allein mit ganzen Zahlen vor
- statt einer echten Berechnung der Funktionswerte von y wird geprüft, welcher Punkt den geringsten Fehler im Vergleich zu gewünschten Linie aufweist
- es werden aus den x und y-Werten die Differenzen Δx und Δy berechnet
- ein Wert Abweichung dient zur maßstäblichen Berechnung der Abweichung von der gewünschten Linie
- Der Algorithmus läuft dann wie folgt ab:
  1. Abweichung = 0 als Startwert
  2. Falls Abweichung < 0 dann gehe zu 5.
  3. Subtrahiere Δx von Abweichung
  4. Erhöhe X um 1 - gehe zu 7.
  5. Addiere (Δy - Δx) zu Abweichung
  6. Erhöhe X um 1 und Y um 1
  7. Zeichne Punkt bei X, Y
  8. Fall x < Xend dann gehe zu 2.



Zeichnen von Kreisen

- seltener als Linien (in der Technik bei CAD/CAM-Systemen jedoch oft)
- Analog übertragbar auf Ellipsen

Erster Lösungsansatz :

- Anwendung der bekannten Formeln aus der Trigonomie

$$x = r * \cos(\beta)$$

$$y = r * \sin(\beta)$$

- β läuft von 0 bis 360° (oder anderes Intervall für Teilkreise)

Nachteile:

- Berechnung erfordert cos/sin und muß dazu Reihen entwickeln (vgl. Summenformeln aus 1. Semester !)
- dadurch sehr langsam
- die Schrittweite von β muß an die Größe angepasst werden (je größer der Kreis um so kleiner die Schrittweite !), andernfalls ergeben sich Lücken (oder Zeichnen von Linienstücken notwendig bei Gefahr des Entstehens von Vielecken...)
- Einziger Vorteil: relativ einfache Berechnung von Kreisbögen - wird meist nur zur Bestimmung des ersten Punktes bei den anderen Algorithmen verwendet

Zeichnen von Kreisen – 1. Verbesserung

- Bei Anwendung der Kreisgleichung

  $x^2 + y^2 = r^2$

- läßt sich y bei gegebenen x ermitteln aus

  $y^2 = r^2 - x^2$

  $y = \text{sqrt}(r^2 - x^2)$ $\quad r^2 = \text{const}$

- diese Gleichung gilt nur im 1. Quadranten des Kreises
- Die Punkte in der anderen 3 Quadranten können jedoch durch Vertauschen der Vorzeichen (=Spiegeln) gewonnen werden.
- Problem: Lücken bei kleinem X (Behebung durch Linien !??)

- Dieser Algorithmus ist schneller als der erste, jedoch durch das Wurzelziehen immer noch recht langsam !
- Das Wurzelziehen kann vermieden werden, wenn man bei

  $x^2 = r^2 - y^2$

  verbleibt und die zwei nur möglichen Y-Werte (entweder nur hoch oder hoch und nach links) einsetzt und nach der kleinsten Differenz vorgeht...

Zeichnen von Kreisen mit Bresenham-Algorithmus

Analog zum Bresenham-Algorithmus geht man hier auch fallweise vor:

- Bei einer Beschränkung auf den Bereich 90-> 45 Grad kann der nächste Punkte entweder nur 1 nach rechts (Q) oder 1 nach rechts und 1 nach unten platziert (R) werden

- Die Abstände dieser beiden Punkte vom Mittelpunkt lassen sich einfach berechnen und miteinander vergleichen :

  Punkt Q : $DQ = ((X+1)^2 + y^2) - r^2$

  Punkt R : $DR = ((X+1)^2 + (y-1)^2) - r^2$

- Falls abs(DQ) > abs(DR) liegt Q weiter weg vom Kreisbogen und ist damit ungünstiger !
- Somit wird als nächster Punkt R , andernfalls Q gewählt.

Zeichnen beliebiger Funktionen

- Bei Vorhandensein einer leistungsfähigen Linienfunktion lassen sich auch beliebige Kurven und Linien zeichnen

Berechnung und Anzeige von y = f(x)

- x wird als realer Wert (Weltkoordinaten) in f(x) eingesetzt und y wird berechnet
- Nach Transformation der Werte in Bildschirmkoordinaten wird geprüft,
- ob der neue Y-Werte sich um mehr als 1 vom letzten Wert unterscheidet
- Falls ja, dann wird eine Linie vom letzten Punkt zum nächsten Punkte gezeichnet
- Fall nein, dann wird nur der nächste Punkt gezeichnet

Sonderfall : Nicht stetige Funktionen (mit Sprungstellen)

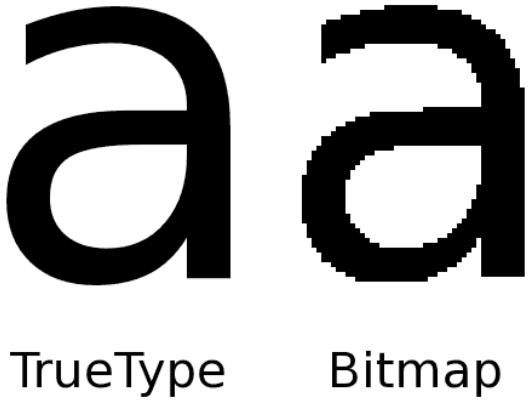
- Bei Erkennung von Unstetigkeiten bei der f(x)-Berechnung muss statt einer Linie ein neuer Punkt gezeichnet werden !

Zeichnen von Text

Zwei Arten der Textgenerierung :

Bitmap-Schriften

- Bitmap-Schriften werden als Pixelraster gespeichert für fest definierte Schriftgrößen (früher 8x8 oder 16x16)
- moderne Grafikkarten verwalten Bitmap-Daten und kopieren Bitmuster eigenständig in den Bildspeicher



Vektorschriftarten („TrueType“) :

- Schrift wird als Grafik beschrieben (B-Splines oder Bezier-Kurven)
- Schriften sind beliebig skalierbar bei konstanter Qualität
- Das Ausrechnen und Setzen der Pixel erfolgt erst NACH der Definition der Schriftparameter zur LAUFZEIT durch den Prozessor oder die Grafikkarte und ist relativ rechenintensiv.
- Schriftarten können durch Copyright geschützt sein !

Schriftsatzparameter

Zur Gestaltung der Textausgabe sind verschiedene Parameter zu beachten :

- **Zeichenbreite**
  - **Monospace**-Schriften haben pro Zeichen eine FESTE Breite (Courier) -> sinnvoll für einfache Spaltenformatierungen

    mime
  - **Proportional**-Schriften verwenden genau den nötigen Platz pro Zeichen -> besser lesbar und heute meistens eingesetzt

    mime
- **Zeichenhöhe** : meist gemessen in DTP-Punkten (= 0,3527 mm)
- **Laufweite** – Abstand der Zeichen voneinanden – bei Computerdarstellungen meist fix, in TYPOGRAFIE anpassbar
- **Serifen -** kleine waagerechte Striche an den Zeichenfüßen
  - Serifen-Schriften sind im Fliesstext besser lesbar

    Times
  - Serifenlose Schriften sind klarer und prägnanter für technische Zwecke

    Arial

Angabe der Auflösung von Grafiken

- Bei der Druckausgabe dots-per-inch (dpi).
- Bei Bildschirmen analog auch pixel per inch (ppi).
- **= Punkte Pro inch = Punkte / 2,54 cm**
- **Typische Auflösungen**
  - 82-cm-Fernsehbildschirm (1366 x 768 Bildpunkte) - 50 dpi
  - Monitore 50 –100 dpi (bei 72 dpi genau 1:1 zu Typografie-Pixel)
  - Poster DIN A0 im gerasterten Plakatdruck 50 dpi
  - Zeitschriften 300 dpi
  - Fotos (Laborabzüge) 300 dpi
  - Laserdrucker 300 bis 1200 (effektiv)
  - Tintenstrahldrucker 300 bis 1200
  - Flachbettscanner 600 bis 1200
  - hochauflösende 35-Millimeter-Filme bis 3000 dpi
  - optische Computermäuse 400 bis 4000 dpi

Speicherung von Grafiken

Verschiedene Grafikformate

- Generelle Parameter : Abmessungen in x/y, Farbtiefe, Kompression

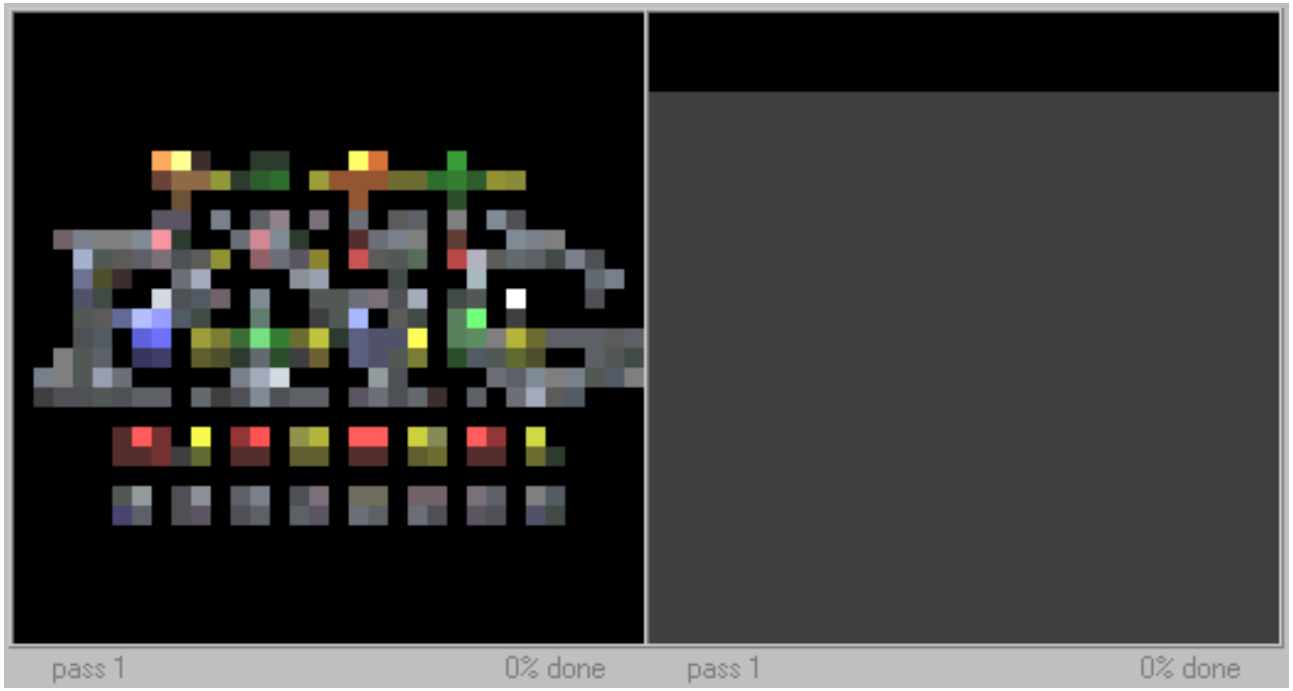
Formate

- **BMP-Windows-Bitmap** : nicht oder nur sehr einfach komprimierte Bildablage mit 1 ... 32 Bit Farbtiefe
- TIFF -**Tagged Image File Format** – sehr komplexes, professionelles Format für die Druckproduktion (verlustfrei, mehrere Bilder /Datei)
- GIF - **Graphics Interchange Format** – ehemals von Compuserve entwickelt, Patentrechtliche Probleme bis 2006, nun frei
  - nur 256 Farben pro Datei, jedoch sehr effiziente Kompression
  - Transparenz und Quasi-Animation möglich (Hauptgrund)
  - Bei der Druckausgabe dots-per-inch (dpi).
- PNG-Ersatz für GIF, leistungsfähiger, jedoch nicht durchgesetzt
- JPEG - verlustbehaftete Kompression in verschiedenen Qualitätsstufen

Techniken zur effizienten Grafikanzeige

Speziell für das Internet sind spezielle Verfahren zur optimalen Anzeige auch bei langsamen Verbindungen verfügbar :

- Interlacing : Bild wird schrittweise aufgebaut – entweder aus Linien oder Blöcken mit schrittweiser Verfeinerung

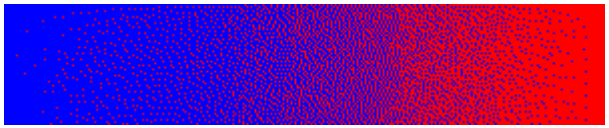


Weitere 2D-Algorithmen

- In Analogie zu den betrachteten Algorithmen werden auch weitere Aufgaben durch meist punktweises Vorgehen gelöst :

Flächen füllen bei vorhandener Begrenzung mit anderer Farbe

- rekursiver Aufruf einer Funktion zum Setzen eines Punktes falls noch nicht gesetzt und Aufruf dieser Funktion für alle 4 Nachbarn (relativ speicherhungrig)
- Zeilenweises Füllen der Fläche und Rekursion über nicht gesetzte Nachbarzeilen
- Schraffur von Rechtecken und Polygonen zum Teil relativ aufwendig (mit Anwendung des Clippings – siehe Folgepunkt)
- Dithering zum Erstellen von Farbübergängen bei geringer Farbtiefe



Abschneiden von überlagerten Grafiken (Clipping)

- Falls alle Grundoperationen auf Punkte zurückgeführt werden, läßt sich ein Clipping durch einfache Intervalltests erreichen
- Bei Linien als Basiselementen erfolgt eine Berechnung der Schnittpunkte mit den umgebenden Kasten und die Linie wird auf die Schnittpunkte begrenzt