2009

Computergraphik 1 (186.461)

Ausarbeitung für die mündliche Prüfung

Ziele der LVA: Vermittlung eines Verständnisses und Überblickes über die Grundbegriffe und Grundmethoden der Computergraphik.

Inhalt der LVA: Einführung, User Interfaces und Eingabegeräte, Datenstrukturen, Mathematische Grundlagen, Abbildungen und Clipping, Rastergraphik, 2D- und 3D Transformationen und Objektrepräsentationen, Sichtbarkeitsberechnung, Beleuchtungsmodelle, Oberflächenschattierung, Farbe, graphische Programmierung und OpenGL.

Martin Tintel
09.01.2009

Kapitel 1 – A Survey of Computer Graphics (S. 2)

1) Was ist Computergraphik?

- Computergraphik ist der Vorgang, der aus einer Beschreibung ein Bild generiert.
- Mustererkennung (pattern recognition) ist Umkehrung: Bild ⇒ Beschreibung.
- Bildverarbeitung (image processing): „schlechtes“ Bild ⇒ „besseres“ Bild.

2) Welche Anwendungsbereiche von Computergraphik gibt es?

Computer Aided Design (CAD)

Ing.-technische Entwicklung wird durch Computer(graphik) unterstützt.

Presentation Graphics

Repräsentation von Geschäftsdaten (z.B. Säulen / Linien / Torten / Balken / Punkt,… Diagramme).

Computerkunst

Paintbrush Programme, kommerzielle Kunst (z.B. Logos, Werbung), Animation, Videoclips,…

Unterhaltung

Filme, Musikvideos, TV-Shows,…

Ausbildung und “Training”

Computer-generierte Modelle von physikalischen, finanziellen und ökonomischen Systemen, Simulatoren (Militär, Fahrschule,…).

Visualisierung

(Wissenschaftliche) Informationsvisualisierung: Dabei sind die Algorithmen von Daten-Charakteristika und dem Untersuchungsziel abhängig. Techniken: Color coding, contour plots, volume visualization.

Bildverarbeitung

Modifikation oder Interpretation von Bildern. Ziele: Qualität verbessern bzw. Maschinenunterstützte Verarbeitung visueller Information.

Graphical User Interfaces (GUIs)

WIMP (windows, icons, menus, pointer).

Kapitel 2 – Overview of Graphics Systems (S. 34)

3) Welche video display devices werden in der Computergraphik verwendet und welche Eigenschaften haben sie? (S. 35) bzw. „Wie funktioniert eine (Farb)-Kathodenstrahlröhre (CRT)? (S. 42) bzw. „Was ist ein Plasmabildschirm und wie funktioniert er? (S. 45)“

CRT Monitore (Röhrenmonitore, S. 42)

4 Magnete links und rechts sowie oben und unten, dazwischen liegt der Elektronenstrahl, dieser trifft dann gelenkt durch die Magneten auf der Oberfläche, welche mit einer phosphoreszierenden Schicht

beschichtet ist, auf und beginnt in diesem Punkt zu leuchten. Ein Bild wird von links nach rechts und von oben nach unten Punkt für Punkt aufgebaut. Bei Farbmonitoren hat man für jede Farbe einen separaten Elektronenstrahl (RGB-fokussieren alle am selben Punkt am Bildschirm).

Raster-Scan Monitore (S.39, S.52)

Zeichnen Punkt für Punkt und Zeile für Zeile.

Random-Scan Monitore (S. 41, S.56)

Zeichnen das Objekt direkt auf den Bildschirm.

Flachbildschirme

Man kann Flachbildschirme in zwei Arten unterteilen:

- Emmisive displays = Emitter (S.44): konvertieren elektrische Energie in Licht, können also Licht selbst erzeugen.
    - Plasmabildschirme: 2 Konduktoren werden elektrische geladen, dazwischen befindet sich lumineszierendes Gas, am Schnittpunkt der Konduktoren/Leiter beginnt das Gas zu leuchten. Die Trennung zwischen den Bildpunkten wird durch das elektronische Feld der einzelnen Konduktoren erreicht.
    - Thin-film-electroluminsecent-displays: sind ähnlich aufgebaut wie die Plasmabildschirme, jedoch zwischen den Glasplatten befindet sich Phosphor. Nachteil: braucht mehr Strom als Plasmabildschirmen und gute Farb/Graudarstellungen sind nur schwer zu erreichen.
- Nonemissive displays = Nonemitter. Erzeugen das Licht nicht selber sondern nutzen Licht aus anderen Quellen zur graphischen Darstellung (z.B.: LCD-Bildschirme)

LED (light emmiting diodes, Flachbildschirm)

Es gibt schon blaue Leuchtdioden ⇒ sehr teuer in der Produktion. In jedem Bildpunkt eine Leuchtdiode.

LCD (liquid crystal displays, Flachbildschirm)

Verwendung bei Notebook-Displays. Verwenden polarisierendes Licht entweder aus der Umgebung oder von einer anderen Quelle, welches durch die Flüssigkeitskristalle geschickt wird, um ein Bild zu erzeugen.

- Passive-matrix-LCDs: Die Flüssigkeitskristalle polarisieren das Licht (S. 46). Man hat 2 Glasplatten jeweils mit einem Polarisierer, die im rechten Winkel zueinander liegen, durch die Flüssigkeitskristalle wird das hereinkommende Licht gedreht ⇒ nur wenn sie unter Strom stehen, ansonsten sind sie in Längsrichtung angeordnet und lassen das Licht nicht durch.
- Active-matrix-LCDs: bei jedem Bildpunkt ein Transistor der die Spannung kontrolliert.

4) Was versteht man unter interlacing? (S. 41)

Interlacing wird bei Raster-Scan Monitoren verwendet. Es ist eine Methode zur Aufnahme von Bewegtbildern, der Übertragung von Bewegtbildern und der Darstellung von Bildern auf sequenziellen Darstellungsgeräten (Bildröhren, Laser-Projektoren). Das Verfahren findet Verwendung bei Fernsehgeräten (dort noch üblich) und Computerbildschirmen (dort mittlerweile eher unüblich).

Dabei bilden zwei Halbbilder (engl. Fields) erst ein vollständiges Bild mit voller vertikaler Auflösung. Beim Bildaufbau des 2:1-Interlacings wird jeweils eine Zeile übersprungen, daher der Name. Die übersprungenen Zeilen wurden beim vorhergehenden und werden beim nächsten Aufbau dargestellt.

Der Vorteil des Verfahrens besteht in der Verdopplung der Bildwiederholfrequenz bei gleicher vertikaler Auflösung. Der Nachteil ist ein erhöhtes Flimmern des Bildes an dünnen waagerechten Linien und anderen feinen Strukturen, Artefakte an langsam bewegten Objekten sowie Inhomogenität der Darstellung von strukturarmen Flächen.

5) Was ist Virtual Reality? (s. 48)

Ist Stimulation der menschlichen Sinne, um ihnen eine fiktive Umgebung vorzutäuschen.

Die 3 Ebenen

- Augen: privat eye, head mounted display (HMD)
- Ohren: Kopfhörer, Lautsprecher in HMD
- Tastsinn: data glove (Datenhandschuh), data suit (Sensorenanzug), Laufband

6) Was ist ein "head-mounted display" (HMD) bzw. headmounted display? (s. 51)

Ist ein (VR-)Helm mit zwei LCD-Bildschirmen und Lautsprechern. Er wird verwendet um eine Virtual-Reality-Umgebung vorzutäuschen. Außerdem verfügen HMDs oft über einen Tracker, der die momentane Blickrichtung und Position des Benutzers erfassen kann.

7) Was ist ein "data-glove" und wozu wird er verwendet? (S. 60)

Ein „data-glove“ ist ein mit Sensoren besetzter Handschuh, der es dem Benutzer erlaubt Objekte in einer VR-Umgebung anzufassen und zu bewegen. Durch die Sensoren werden auch die Bewegungen der einzelnen Finger erfasst. In Kombination mit einem HMD ist es auch möglich, die eigene Hand in der VR-Umgebung zu sehen.

8) Welche Arten von Scannern mit CCD-Elementen gibt es?

Flachbrett-Scanner

CCD-Elemente: Lichtsensoren, die bei einfallendem Licht einen elektrischen Strom erzeugen. Die Lichtsensoren sind farbenblind, das heißt, es wird nur die Intensität der Farbe aber nicht die Farbe selbst erfasst.

Die zeilenweise Abtastung kann je einmal pro Farbe erfolgen, oder einmal mit 3-fach-Zeile und je einem Farbfilter. Durch die CCD-Elemente kann es durch abweichende Lichtempfindlichkeiten der Fotozellen zu Bildfehlern kommen. Ein Trommelscanner ist deshalb beim momentanen Stand der Technik immer noch qualitativ besser, als ein Flachbettscanner.

Dokumenten-Scanner

Die Scanner sind meist als Durchzugsscanner ausgeführt und können ja nach Ausführung Dokumentenstapel von 50 bis 1000 Seiten selbstständig abarbeiten. Die meisten Modelle sind in der Lage mittels zweier CCD-Sensoren-Leisten die Vorder- und die Rückseite gleichzeitig in einem Scan zu

erfassen (Duplex). Einige Modelle werden als Kombination von Durchzugs- und Flachbettscanner ausgeführt, um auch gebundene Dokumente ohne Zerstörung scannen zu können.

Die meisten der am Markt befindlichen Dokumentenscanner sind reine schwarz/weiß-Geräte (mit grün als Blindfarbe). Neuerdings kommen aber immer mehr Geräte auf den Markt, die auch farbig in guter Qualität scannen können.

9) Welche Drucker gibt es und wie funktionieren sie generell? (s. 72)

Impact Drucker

Impact-Drucker sind von der Funktionsweise vergleichbar mit einer Schreibmaschine. Sie drucken vorgefertigte Zeichen über ein Farbband auf Papier.

Nonimpact Drucker

Verwenden keine fixvorgefertigten Zeichen:

- Laserdrucker: Man hat eine Rolle, welche mit einem photoelektrischen Material ummantelt ist. Diese wird durch einen Laser elektrisch aufgeladen. Auf dieser Rolle befinden sich verschiedene Ladungen. Anschließend wird der Toner auf die Rolle aufgetragen und an den geladenen Stellen bleibt der Toner haften, daraufhin wird der Toner auf das Papier übertragen.
- Tintenstrahldrucker: Dabei wird die Tinte durch elektronische Ladungen unterschiedlich abgelenkt um die Tinte genau zu positionieren.
- Electric Static Devices: Kopierer: Papier wird negativ geladen, Toner wird positiv geladen, dort wo das Papier neg. geladen ist, bleibt der Toner haften.

10) Wie arbeiten Sublimationsdrucker?

Sublimationsdrucker arbeiten mit Farbbändern oder Farbfolie. Pro Grundfarbe (cyan, magenta, yellow) wird ein Druckdurchlauf benötigt. Die Farbe wird mit Heizelementen von den Farbbändern gelöst und auf das Papier aufgedampft. Die Temperatur bestimmt dabei die Farbstoffmenge, die auf das Papier übertragen wird und steuert damit die Helligkeit des Bildpunktes. Die meisten Sublimationsdrucker arbeiten mit Farbfolien in der Größe des zu bedruckenden Mediums. Aus diesem Grund sind alle Ausdrucke gleich teuer, egal ob nur ein Punkt gedruckt wird oder eine vollflächige Seite.

Kapitel 3 – Output Primitives (S. 84)

11) Welche line drawing Algorithmen kennen Sie und wie funktionieren sie?

DDA (digital differential analyzer, S. 94)

Die Linie wird in Einerkoordinaten-Einheitsintervalle unterteilt $y_{k+1} = y_k + m$

- k … geht von 0 (Startpunkt) bis Endpunkt => nimmt nur Integerwerte an und wird immer um 1 erhöht
- m … ist die Steigung, jeder Wert zw. 0 und 1

- y … muss zur nächsten Pixelposition gerundet werden

Ist die Steigung größer als 1, werden die x- und y-Rollen vertauscht.

Nachteil: es wird mit float gerechnet und ist daher rechenaufwändiger, aber immer noch schneller, als wenn man die Pixelpositionen direkt implementieren würde. Er ist schneller als normale mathematische Berechnungen für Geraden y=kx+d, da er sich die Eigenschaft vom Rastersystem zu Nutze macht und dadurch die Multiplikation durch Addition ersetzen kann.

Bresenham Line Algorithmus (S. 95)

Verwendet Integerzahlen => weniger rechenaufwändig; weiters können auch Kreise und Kurven dargestellt werden.

Die vertikale Achse definiert die Scanlinienposition und die horizontale Achse die Pixel-Spalten. Es wird überprüft welche Pixelposition der Linie näher ist (obere oder untere).

Berechnung mit Hilfe des Entscheidungsparamter p_k. Ist $p_k < 0$, dann wird y_k geplottet, andernfalls y_{k+1}.

Horizontale Linien (mit delta y=0), vertikale Linien (mit delta x=0) und diagonale Linien (|delta x|=|delta y|) können direkt gezeichnet werden.

12) Welche zwei grundsätzlichen Arten von Polygonen kennen Sie? Wie geht die Umwandlung vor sich?

Polygon ist eine ebene Figur mit mind. 3 Koordinatenpunkten (vertices), welche durch Geraden (edges) miteinander verbunden sind. Edges eines Polygons haben außer den Eckpunkten keine gemeinsamen Punkte ⇒ dürfen sich nicht schneiden

Konvexe Polygone

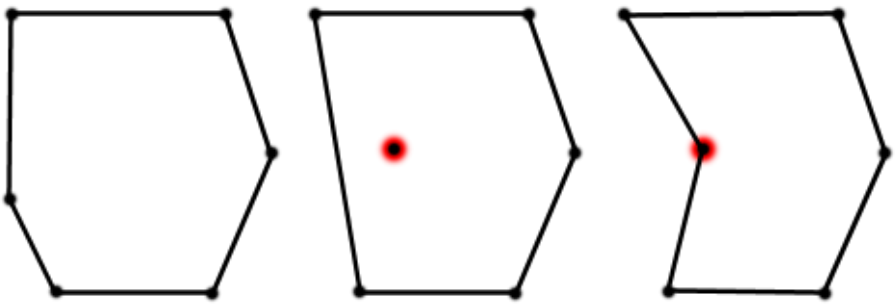
Alle Winkel innerhalb des Polygons sind < 180°

Konkave Polygone

Mindestens ein Winkel > 180°. Verlängerungen mancher Kanten können andere Polygonkanten schneiden

Beispiel für konvexes und konkaves Polygone Quelle

Links sieht man ein konvexes Polygon, in der Mitte ein konkaves Polygon, wobei der rot markierte Eckpunkt das Polygon konkav macht. Wird er übersprungen beim zeichnen, liegt er im neu entstandenen Polygon (rechts zu sehen).



Erkennen konvexer Polygone

- Vektormethode: Alle Vektor-Kreuzprodukte haben das gleiche Vorzeichen => konvex
- Rotationsmethode: Drehe die Polygon-Kanten auf die x-Achse. Immer gleiche Drehrichtung => konvex

Aufspaltung konkaver Polygone

Da konvexe Polygone viel weniger Sonderfälle erzeugen, sind viele Algorithmen für konvexe Polygone ausgelegt. Daher braucht man Methoden um konkave Polygone in mehrere konvexe Polygone zu zerteilen:

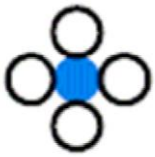
- Vektormethode: Man geht gegen den Uhrzeigersinn vor, hat man irgendein Kreuzprodukt mit neg. z-Komponente, kann man das Polygon aufteilen: man verlängert den Kantenvektor des ersten Punktes des negativen Kreuzprodukt-Paares.
- Rotationsmethode: Man geht wieder gegen den Uhrzeigersinn vor. Der Knoten Vk wird in den Ursprung gebracht und das Polygon wird rotiert bis der Knoten Vk+1 auf der x-Achse ist. Ist der Knoten Vk+2 unter der x-Achse, dann ist das Polygon konkav und wird entlang der x-Achse in zwei konvexe Polygone aufgeteilt.

Man macht den jeweiligen Test dann für die beiden neuen Polygone, solange bis man alle Knoten getestet hat.

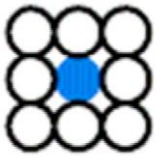
13) Welche Füll-Algorithmen kennen Sie und wie funktionieren Sie? (S. 123 ff.) Welche Möglichkeiten gibt es zum Füllen von Polygonen? (S. 126)

Boundary-Fill Verfahren (S. 201)

Startet bei einem „inneren“ Punkt und füllt von innen nach außen, von Pixel zu Pixel, bis die Polygonränder erreicht sind. Diese Methode ist bei Polygonen mit komplexen Grenzen und interaktiven Zeichnungssystemen sinnvoll.



4-way-stepping



8-way-stepping

Über 4er-Nachbarschaft bzw. 8er-Nachbarschaft

Fülle rekursiv alle Nachbar-Pixel bis Grenze erreicht ist

Bei 4er-Nachbarschaft muss die Grenze die 8er-Nachbarschaft erfüllen und umgekehrt

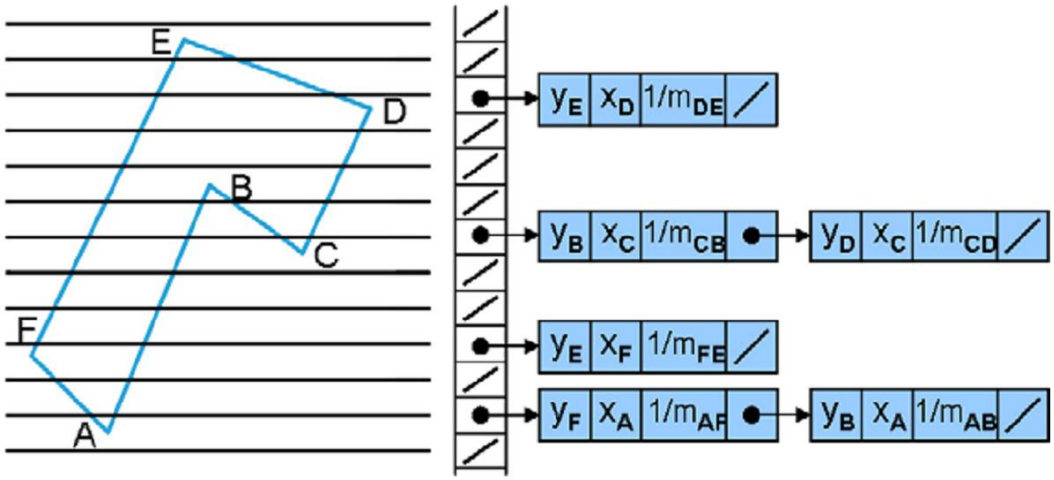
Rekursive Boundary-Fill Algorithmen füllen Flächen nicht immer korrekt, wenn ein Pixel schon in der Füllfarbe dargestellt wird. Stößt man auf ein Pixel, das schon die Füllfarbe hat, kann es passieren, dass das Programm beendet wird, ohne das alle Pixel gefärbt sind. Dem kann man vorbeugen, indem man zuerst die Farbe aller innern Pixel ändert, die schon zu Beginn die Füllfarbe haben, bevor man Boundary-Fill anwendet.

Anwendung siehe Buch S. 204

Da diese Methode bedenkliche Häufungen von benachbarten Punkten benötigt werden meistens effizientere Methoden verwendet.

Scan-Line Fill Verfahren (S. 196)

Berechnet alle Schnittpunkte der Scanlinie mit den Polygonrändern. Danach werden alle Pixel, die in diesem Intervall liegen, auf die entsprechende Farbe gesetzt.



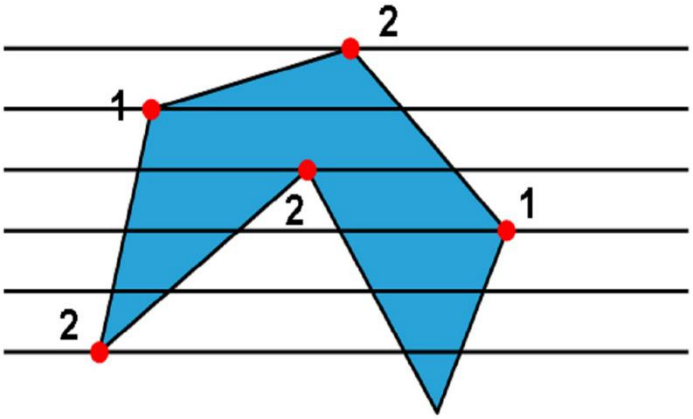
Das Verfahren funktioniert folgendermaßen: Die Kanten werden nach dem kleinsten y-Wert sortiert (Bucketsort). Man speichert für jede Kante: [max. y-Wert, Anfangs-x-Wert, Steigung].

Aus dieser Datenstruktur erzeugt man für jede Scanlinie von unten nach oben eine Liste der „aktiven" Kanten, das sind jene, die die Scanlinie schneiden. Dann wird einfach die Scanlinie vom 1. zum 2. Schnittpunkt gefüllt, vom 3. zum 4., vom 5. zum 6., …

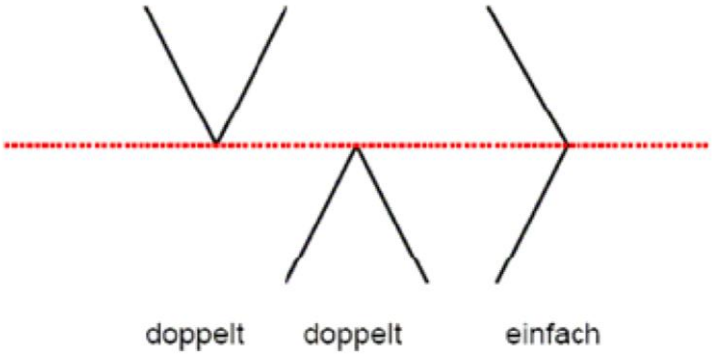
Die aktiven Kanten erzeugt man inkrementell. Am Beginn sind keine Kanten aktiv. Aus der sortierten Kantenliste sieht man für jeden y-Wert, ob dort eine neue Kante beginnt, diese wird zur aktiven Liste hinzugefügt. Gleichzeitig werden alle Kanten, deren maximales y überschritten wurde, entfernt. Die Liste selbst ist immer nach ihren Schnittpunkten von links nach rechts sortiert, sodass das Zeichnen unmittelbar erfolgen kann.

Die Schnittpunkte kann man ebenfalls inkrementell erzeugen. Ausgehend vom (exakten) Schnittpunkt (x_k, y_k) einer Scanlinie mit einer Kante erhält man den nächsthöheren Schnittpunkt (x_{k+1}, y_{k+1}) durch $x_{k+1} = x_k + \frac{1}{m}$ und $y_{k+1} = y_k + 1$. Man sieht nun, warum der Anstieg 1/m in den Knoten mitgespeichert wird.

Probleme: Geht eine Scan-Line genau durch einen Eckpunkt des Polygons, so muss je nachdem wie der Eckpunkt ausschaut, eine Unterscheidung getroffen werden, wie oft dieser Eckpunkt gezählt wird.



Die Unterscheidung funktioniert folgendermaßen: Wenn die beiden Kanten auf derselben Seite liegen (oben oder unten) dann wird zweimal gezählt, wenn sie auf unterschiedlichen Seiten liegen zweimal.



Um diesem Problem aus dem Weg zu gehen werden häufig die Punktkoordinaten um einen kleinen Wert ε nach oben oder unter verschoben. Dieser Wert ist so klein, dass man es nicht sieht, aber groß genug, dass der Punkt neben der Scanlinie liegt.

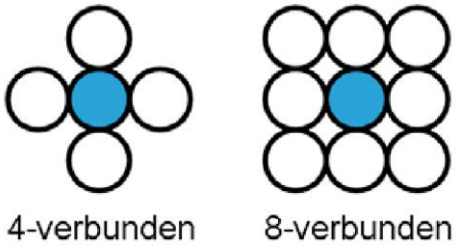
Flood Fill Verfahren (S. 205)

Will man eine Fläche füllen (oder umfärben), deren Ränder nicht in einer Farbe gehalten sind, kann man diese Flächen färben, indem man eine festgelegte innere Farbe ersetzt, anstatt auf die Randfarbe zu achten. Man nimmt wieder einen Punkt im Inneren und ersetzt die Farbe der Pixel, die eine bestimmte Farbe haben. Hat die Fläche mehr als eine Farbe, so werden die Pixelwerte zuerst auf eine Farbe gesetzt. Dies geschieht mittels 4- oder 8-Nachbar-System.

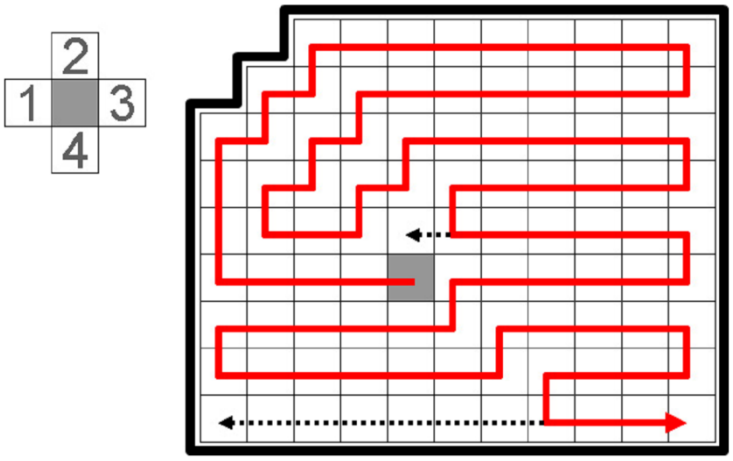
Ausgehend von einem Startpunkt (Referenzpunkt, Saatpunkt) wird in alle Richtungen gefüllt, bis man an eine Grenze stößt. Diese Grenze kann entweder explizit definiert sein, zum Beispiel durch eine Umrandung in einer bestimmten Farbe, oder aber implizit, z.B. dadurch, dass eine bereits gefärbte Fläche umgefärbt wird. Mischvarianten kommen auch vor. Diese Unterscheidung verändert aber lediglich das Abbruchkriterium beim Füllen. Wir wollen davon ausgehen, dass die zu füllende Fläche in einer „alten“ Farbe bereits gezeichnet ist und umgefärbt werden soll.

Grundlage für das Weitergehen in alle Richtungen ist die Definition der erlaubten Richtungen. 4-verbunden heißt, dass eine Verbindung nur über die 4 Hauptrichtungen definiert ist, 8-verbunden sieht auch diagonale Pixel als verbunden an. Man kann sich leicht überlegen, dass für eine 4-

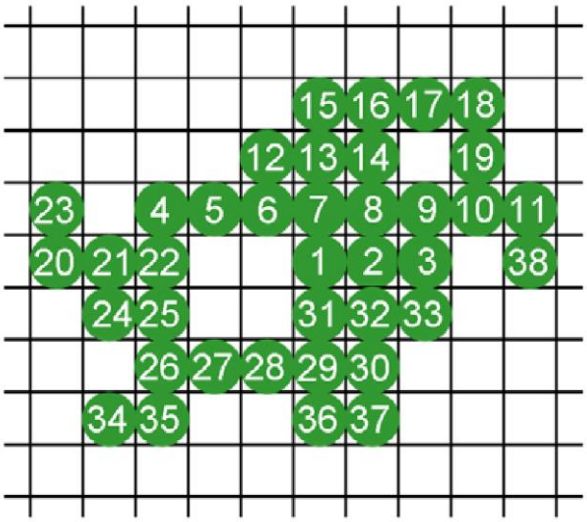
verbundene Fläche eine 8-verbundene Grenze reicht, eine 8-verbundnee Fläche aber eine 4-verbundene Grenzlinie benötigt.



Diese Prozedur erzeugt jedoch eine Füllreihenfolge, die zu einer sehr hohen Rekursionstiefe führt, im allgemeinen Fall bis zur Anzahl der zu füllenden Pixel. In der Abbildung unten ist links (oben) die Rekursionsreihenfolge angegeben, und der durchgezogene Pfeil im Polygon (rechts) zeigt, wie tief die Rekursion in diesem Fall geht.



Um das zu verhindern, füllt man meist in horizontaler Richtung iterativ und wendet die Rekursion nur nach oben und unten an. Natürlich muss man dabei aufpassen, dass alle Spans oberhalb und unterhalb rekursiv aufgerufen werden (ein Span ist eine horizontale nicht unterbrochene Folge von Pixeln, die gemeinsam behandelt werden).



Das obenstehende Beispiel demonstriert die Füllreihenfolge, wobei eines der Pixel 1, 2 oder 3 als Startpixel gewählt wurde. Die Rekursion geht zuerst nach oben und dann nach unten. Bei jedem Aufruf müssen alle Spans in diese Richtung erwischt werden. Nach dem Span 4-11 wird nach oben von links nach rechts aufgerufen und danach nach unten ebenfalls von links nach rechts. Teile, die bereits gefüllt wurden, werden erkannt, und die Rekursion bricht dort ab (z.B. von 4-11 nach unten ist 1-3 schon fertig). Die Gesamtrekursionstiefe kann zwar theoretisch auch bei diesem Verfahren sehr hoch sein, in der Praxis ist sie aber proportional der Anzahl der Pixelzeilen (Scanlinien) des Polygons.

14) Welche "inside-outside tests" kennen Sie, und wie funktionieren sie? (S. 127) Was ist die „nonzero winding number rule“ bzw. Non-zero-winding-number-rule? (S. 128)

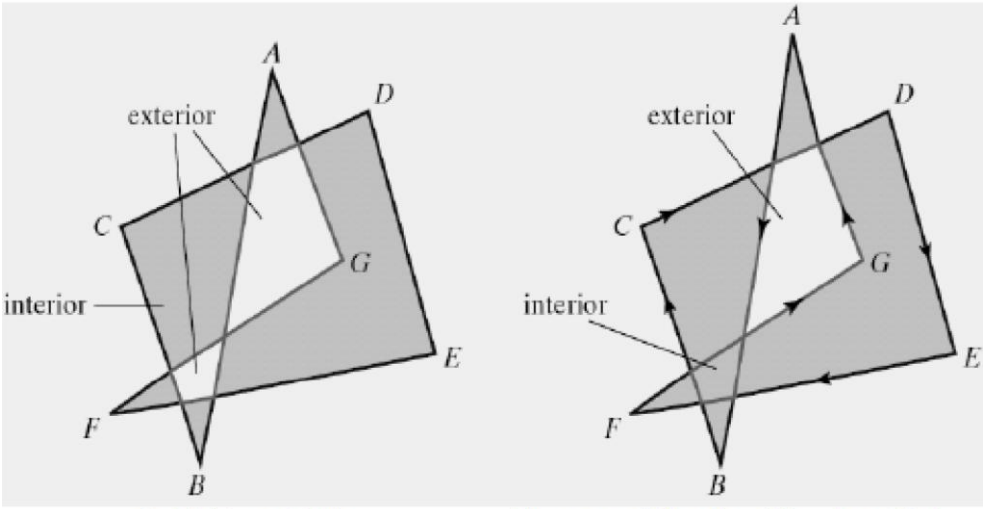
Odd-even rule (auch odd-parity rule oder even-odd rule)

Man zeichnet einen Punkt P und davon ausgehend eine Linie zu einem Punkt außerhalb der Polygongrenzen (nur in eine Richtung). Schneidet diese Linie dabei eine ungerade Anzahl an Kanten, so liegt der Punkt innerhalb, andernfalls außerhalb.

Man muss aufpassen, dass die gezeichnete Linie keinen Eckpunkt kreuzt, da dieser 2x gezählt wird.

Nonzero Winding Number

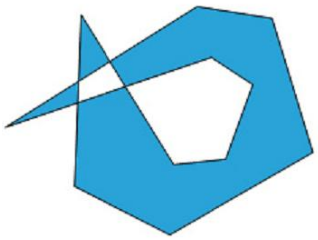
Es wird gezählt, wie oft sich die Objektgrenze um einen bestimmten Punkt dreht (gegen den Uhrzeigersinn). Diese Anzahl wird winding-number genannt. Ein innerer Punkt hat eine winding-numer != 0, also eine nonzero winding number. Man nimmt einen beliebigen Punkt P und zeichnet eine Linie zu einem Punkt außerhalb des Objekts (darf keinen Eckpunkt schneiden). Man geht entlang der Linie von Punkt P zum entfernten Punkt und zählt dabei die Objektgrenzen, die diese Linie schneiden (in jede Richtung). Man zählt +1 wenn es eine Linie von rechts nach links ist und -1, wenn eine Linie von links nach rechts schneidet. bestimmt manche Flächen als innerhalb, die die odd-even-Rule als außerhalb erkennt.



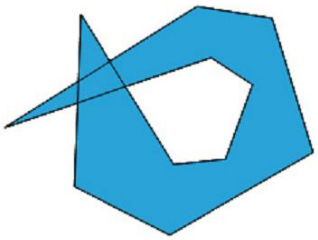
Odd-Even Rule Nonzero Winding Number Rule

All-In

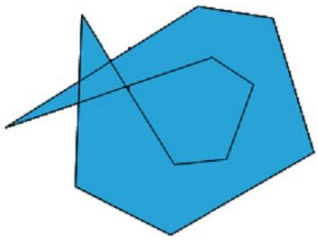
Alles, was irgendwie umschlossen ist, ist innen.



Odd- Even- Rule



Nonzero- Winding- Number Rule



All- In- Rule

15) Welche Möglichkeiten (Verfahren) der character generation (Buchstabenerzeugung) gibt es, welche Vor- und Nachteile ergeben sich daraus? (S. 147)

Bitmap font

Zeichen = Bitmuster auf einem rechteckigen Raster

- Vorteile: Einfach zu definieren und darzustellen
- Nachteile: Benötigt viel Speicherplatz, da jede Variation im Cache gespeichert werden muss Variationen (verschiedene Formate) können oft nur unter Verlust der Qualität dargestellt werden

Outline font

Zeichen werden durch gerade Linien und Kurven dargestellt

- Vorteile: Benötigt weniger Speicherplatz, Zeichengröße und Format (fett, kursiv, usw.) können einfacher verändert werden
- Nachteil: Darstellung dauert länger, weil die Zeichen in den frame buffer geschrieben werden müssen

16) Welche Buffer werden in OpenGL verwendet? (S. 145) Quelle

OpenGL verwaltet mehrere Buffer zum Erstellen von Graphiken:

- Color buffers: front-left, front-right, back-left, back-right (und evtl. weitere)
- Depth buffer (z-buffer)
- Stencil buffer
- Accumulation buffer

Color buffer:

Im color buffer wird das Bild erzeugt. Er ist ein ganz gewöhnliches Pixelarray, meist in rgba (RGB + Alpha). OpenGL verwaltet mindestens 4 davon, um tatsächliche 3D-Bilder mit double-buffering erzeugen zu können.

Depth buffer:

Optional kann man in OpenGL zum rendern einen z-buffer verwenden. In den meisten Fällen wird er auch verwendet werden, OpenGL lässt einem hier aber freie Hand, wenn man z.B. mit BSP's rendern will, kann man den z-buffer auch abschalten: glDisable (GL_DEPTH_TEST) bzw. glEnable (GL_DEPTH_TEST)

17) Was ist ein Accumulationbuffer und wie wird er in OpenGL verwendet? (S. ???) Quelle

Der Accumulationbuffer ist ein RGBA-Buffer (RGB + Alpha) genau wie der Framebuffer. Typischerweise wird er benutzt, um eine Reihe von generierten Bildern zwischen zu speichern und sie dann als ein kombiniertes Bild in den Framebuffer zu kopieren.

Mit Hilfe des Accumulationbuffers kann man leicht Effekte wie z.B. Antialiasing, Motion Blur oder Soft Shadows rendern.

In den Accumulationbuffer wird nicht direkt geschrieben, sondern die einzelnen Bilder werden im Colorbuffer berechnet und dann mit dem Accumulationbuffer auf die eine oder andere Art und Weise verknüpft. Ist das Bild im Accumulationbuffer fertig, wird es dann in den Framebuffer transferiert.

Beim Antialiasing wird das Bild mehrmals leicht versetzt (in x und y) in den Accumulationbuffer gezeichnet. Dadurch entstehen an allen Kanten die gewünschten Farbwerte.

Kapitel 4 – Attributes of Output Primitives (S. 172)

18) Welche Linien-Attribute kennen Sie? (S. 183 ff.)

Eine Linie kann mit 3 Basiselementen dargestellt werden: Farbe, Breite und Style. Die Farbe wird wie auch für alle anderen Primitive gesetzt, wobei Breite und Style in anderen Funktionen ermittelt werden.

Linien-Breite (S. 183)

Die Implementierung hängt davon ab, wo es dargestellt werden soll. Bei Videomonitoren kann man eine dicke Linie mittels paralleler Linien darstellen, wobei ein Tintenplotter eine breite Feder benötigt.

Bei einem Raster wird als Standard eine 1-Pixel-breite Linie gezeichnet. Dickere Linien werden als eine Vervielfältigung der Standardlinie dargestellt, indem einfach Pixel dazu addiert werden und dadurch eine parallele Linie gezeichnet wird. Hat man eine Steigung <=1, kann man dadurch Linien zeichnen, dass man Pixel in vertikale Richtung plottet und diese geplotteten Pixel dann für jede Spalte (x-Wert) darstellt. Soll die Linie 2 Pixel breit sein, so plottet man die Pixel an Stelle (x,y) und (x,y+1). Bei drei und mehr Pixel breiten Linien, werden die Pixel über und unter der Linie geplottet.

Ist die Steigung >1, wird einfach in horizontale Richtung geplottet, und das für jeden y-Wert. Pixel werden links und rechts zur Standardlinie hinzugefügt. Eine weitere Möglichkeit um dicke Linien zu zeichnen ist, selbige als ausgefüllte Rechtecke darzustellen. Um das Rechteck zu zeichnen, bestimmen wir die Vertices in senkrechter Richtung zur Linienrichtung, so dass die Rechteckvertices von den Originallinienendpunkten um die halbe Linienbreite entfernt positioniert werden.

Line caps (S. 184)

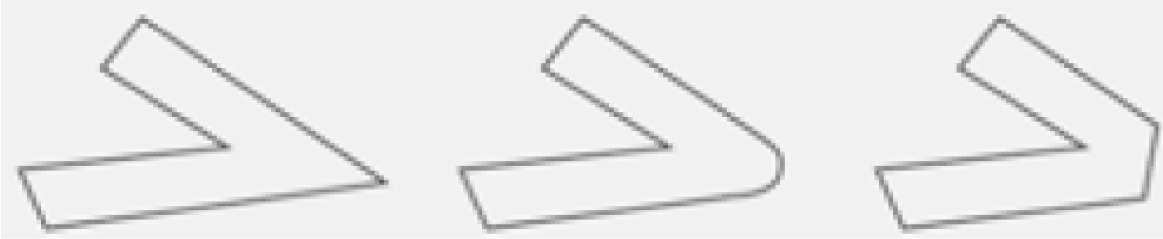
Problem: durch Verbreitung einer Linie können horizontale und vertikale Enden entstehen ⇒ bei sehr dicken Linien fällt das auf. Dies kann man durch line caps lösen.

- Butt cap: hat quadratische Enden, die senkrecht zur Linienrichtung stehen. Hat die Linie die Steigung m, so haben die Enden eine Steigung von -1/m. Jede parallele Linie wird dann zw. den 2 Endlinien dargestellt.
- Round cap: wird dargestellt, indem zur Linie ein ausgefüllter Halbkreis gehängt wird. Der Kreismittelpunkt wird genau in der Mitte der dicken Linie festgelegt und der Kreis hat einen Durchmesser = der Breite der Linie
- Projecting Square cap: dabei wird die Linie um butt caps erweitert.

Linien-Typ

Durchgezogen, gestrichelt, punktiert, …

Linien-Verbindungen (S. 149)



Bei Polylines ist das Problem, dass bei unterschiedlichen Steigungen der Linien Pixel an den Rändern zw. zwei Linien überbleiben, vor allem beim Übergang von horizontalen Pixelspans zu vertikalen. Man kann dicke Polylines zeichnen, indem man an den Endpunkten etwas hinzufügt.

- Miter join: dabei werden die Enden solange erweitert, bis sie einander treffen (Spitze)
- Round Join: dabei wird die Verbindung zwischen den beiden Linien mit einem Kreis abgerundet. Der Kreisdurchmesser entspricht der Linienbreite.
- Bevel Join: dabei werden die Liniensegmente mit butt caps dargestellt, und die 3-eckigen Verbindungsstücke aufgefüllt.

19) Auf welche zwei grundsätzlichen Arten können Farben im Framebuffer gespeichert werden?

Direktspeicherung

RGB-Farbwerte werden direkt im Buffer gespeichert wann immer eine spezielle Farbe im Programm gewünscht wird, wird diese Farbinformation im Framebuffer auf die zugehörige Pixelposition (die dargestellt werden soll) gesetzt.

Ein Minimum an Farbe kann durch 3 Bits Speicher pro Pixel erreicht werden. Jede der 3 Bit-Positionen kontrolliert die Intensität (entweder on oder off). Je mehr Bits/Pixel umso mehr Farben hat man zur Auswahl.

6Bits/Pixel: 2 Bits können für jeden Kanal verwendet werden, das erlaubt 4 versch. Intensitäten für jeden der 3 Farbkanäle (64 Farbwerte für jedes Pixel möglich). Bei einer 1024x1024 Auflösung benötigt man 3 MB Speicher.

Color lookup tables (CLUTs)

Die Farbwerte werden in einer separaten Tabelle gespeichert und an jeder Pixelposition referenziert ein Index auf den speziellen Farbwert. Im Framebuffer werden nur die Indizes auf die CLUT gespeichert.

20) Erklären Sie den Begriff soft-fill (tint-fill) (S. 195)

Ein Muster kann mit einer Hintergrundfarbe konstruiert werden, indem man den Transparenz- Faktor ermittelt, der angibt, in welchem Verhältnis der Hintergrund mit der Objektfarbe gemischt werden soll, oder man verwendet logische bzw. Entscheidungs-Operatoren. (AND, OR, XOR…)

Eine Verwendung ist z.B. um die Füllfarbe weicher zu gestalten wenn die Kanten unscharf sind, eine andere, dass erneute einfärben einer Fläche, die zuvor mit einer semitransparenten Farbe gefüllt wurde, die aus einer Mischung von Vorder- und Hintergrund bestand.

21) Welche Text-Attribute kennen Sie? (S. 211)

Serif (mit „Schnörkseln“)

Ganze Textblöcke sind leichter lesbar.

Sans serif

Einzelne Buchstaben sind leichter lesbar ⇒ besser für Beschriftungen und kurze Überschriften (z.B. Arial, Times New Roman, …)

Variationen/ Stil

Unterstrichen, fett, kursiv, mit Outline, mit Schatten.

Textgröße

Durch Veränderung von Höhe und/oder Breite (in points angegeben), Verschiedene Schriften mit der gleichen Pointsgröße, können verschiedene Textgröße haben. Die Distanz zwischen bottomline und topline ist für alle gleich, aber die Breite ist unterschiedlich.

Abstand

Proportionally spaced fonts: Kleinbuchstaben schmäler als Großbuchstaben.

Charakterhöhe

Wird definiert durch baseline und capline.

Kerned Characters

Stehen über das Limit hinaus. Z.B. f ist breiter als normale Kleinbuchstaben, oder g, j, p, q stehen unten aus der baseline.

Ausrichtung

Die Schrift kann vertikal, horizontal oder diagonal dargestellt werden. Dazu benötigt man den character up vector. Die Buchstaben werden dann so dargestellt, dass die Orientierung von baseline zu capline der Vektorrichtung entspricht.

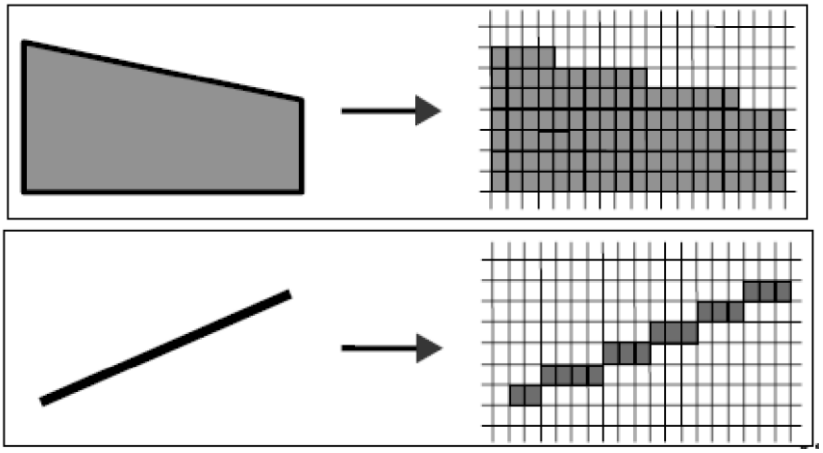
Ausrichtung im Absatz

Rechts-, linksbündig, zentriert oder Blocksatz.

Farbe

22) Was versteht man unter (Anti-)Aliasing? (S. 214)

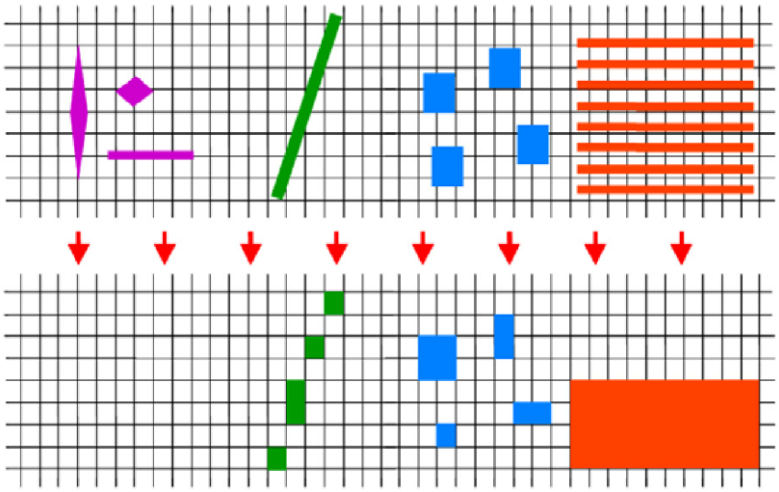
Durch die Darstellung von Linien in einem Raster, müssen die Koordinaten der Punkte auf Integerwerte gerundet werden. Der Stufeneffekt der dabei entsteht wird auch „jaggies“ genannt. Diese Informationsverzerrung aufgrund des low-frequency-samplings (undersampling) heißt aliasing. Sichtbare Aliasingeffekte haben z.B. folgende Ursachen: zu geringe Auflösung, zu wenige verfügbare Farben, zu wenige Bilder/sec, geometrische Fehler, numerische Fehler.



Anti-Aliasing nennt man Methoden zur Reduktion unerwünschter Aliasing-Artefakte. Antialiasing wird z.B. verwendet, um das undersampling auszugleichen.

Da eine Verbesserung der Hardware meist unrealistisch ist, werden hauptsächlich Software-Methoden eingesetzt.

Einige bekannte Effekte neben dem Treppeneffekt sind: Verschwinden kleiner Objekte, unterbrochene schmale Objekte, unterschiedliche Größe gleicher Objekte, völlige Zerstörung feiner Texturen.



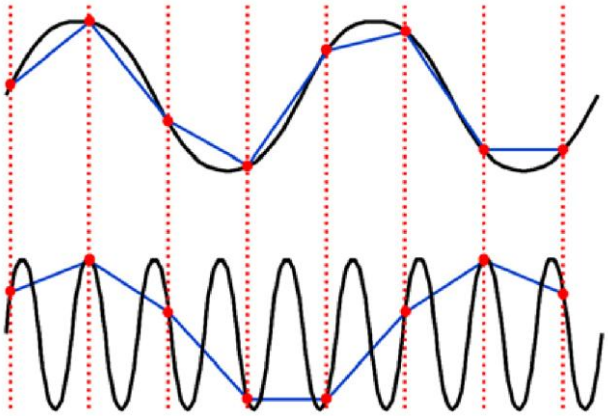
Die Ursache von Aliasing ist ungenügend feine Abtastung des wahren Bildes. Die theoretische Grundlage dazu bildet das Shannon'sche Abtasttheorem. Demnach kann eine Information nur dann korrekt rekonstruiert werden, wenn eine Abtastfrequenz (sampling rate) verwendet wird, die mindestens doppelt so hoch ist wie die höchste zu übertragende Informationsfrequenz. Diese Grenze heißt Nyquist-Limit bzw. Nyquist sampling frequency:

$$f_s = 2 * f_{max}$$

Eine andere Art um dies auszudrücken ist, dass das Sampling-Intervall nicht größer als die Hälfte des Cycle-Intervalls sein darf:

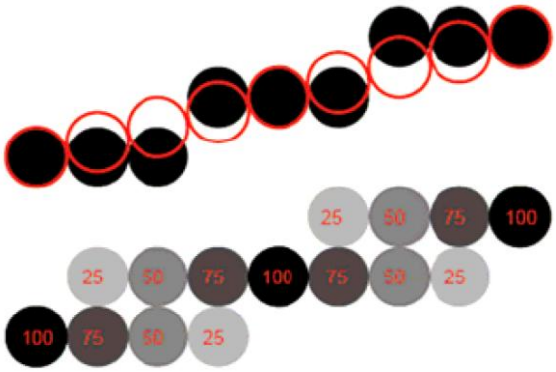
$$\Delta x_s = \frac{\Delta x_{cycle}}{2} \text{ mit } \Delta x_{cycle} = \frac{1}{f_{max}}$$

Die Abbildung unten zeigt wie eine zu grobe Abtastrate eines Signals (Kurve) zu einer völlig falschen Rekonstruktion (Polygonzug) führen kann. Reduzieren lassen sich solche Fehler entweder durch Vorfilterung des Signals oder durch eine Nachbearbeitung des fertigen Bildes. Eine Nachbearbeitung ist der Vorfilterung aber jedenfalls unterlegen. Die zentrale Strategie beim Vorfiltern ist Supersampling (auch Oversampling).



23) Welche Anti-Aliasing Strategien gibt es und wie funktionieren Sie? (S. 215 ff.)

Bei Rastersystemen, die mehr als 2 Intensitätslevels darstellen können, können wir Antialiasingmethoden anwenden, um die Intensitäten der Pixel zu verändern. Hierbei werden die Farbintensitäten der Pixel an den Rändern der darzustellenden Objekte vermindert und dadurch weichere Kanten und eine weniger treppenförmige Darstellung erreicht.

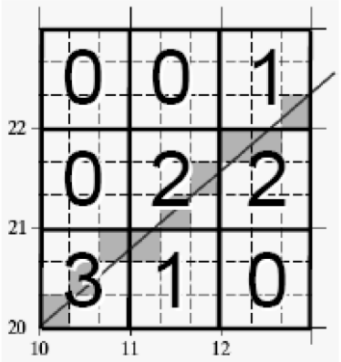


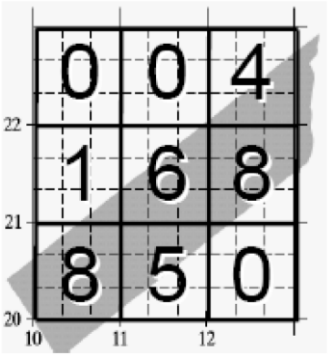
Supersampling

Pixel, die von einer Linie weiter durchkreuzt werden, sollen mehr Linienfarbe bekommen, als Pixel, die von einer Linie nur leicht gestreift werden. Dazu unterteilt man jedes Pixel in Subpixel, zählt, wie viele Subpixel auf der Linie liegen, und wählt eine Intensität die proportional zu dieser Anzahl ist.

Für breitere Linien berechnet man den Prozentsatz der Überdeckung des Pixels durch die Linie und wählt danach die Intensität der Linienfarbe. Aufbauend auf der Erkenntnis, dass die Mitte eines Pixels wichtiger ist als dessen Rand, werden auch manchmal die Subpixel in der Mitte stärker gewichtet als die am Rand („weighted oversampling“).

Anders gesagt: Hierbei wird die Samplingrate erhöht und die Anzeige behandelt, als ob sie ein viel feineres Raster hätte. Nun kann man die Vielzahl von Samplingpoints entlang des feineren Rasters dazu nutzen, um einen entsprechenden Intensitätslevel für jeden Pixel der Anzeige zu berechnen. Objekte werden also in einer hohen Auflösung gesampled und in einer niedrigeren Auflösung dargestellt.





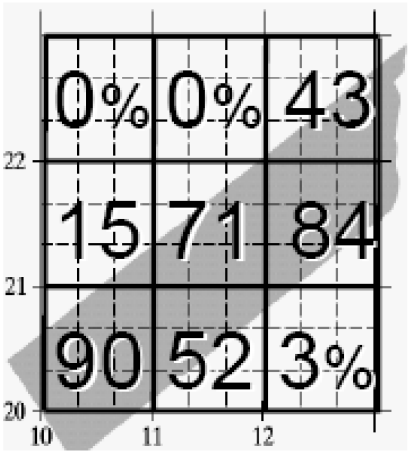
Pixel-Weighting Masks

Supersampling-Algorithmen werden oft so implementiert, dass sie Subpixeln, die näher am Zentrum des Pixelfeldes liegen, mehr Gewichtung geben, da man annehmen kann, dass diese Subpixel wichtiger für die Bestimmung der Gesamtintensität eines Pixels sind. Dies wird auch als Maske für Subpixel-Gewichte bezeichnet. Diese Masken werden oft erweitert, um die Subpixel der Nachbarpixel mit einzubeziehen, damit die Intensitäten über angrenzenden Pixel gemittelt werden können.

1	2	1
2	4	2
1	2	1

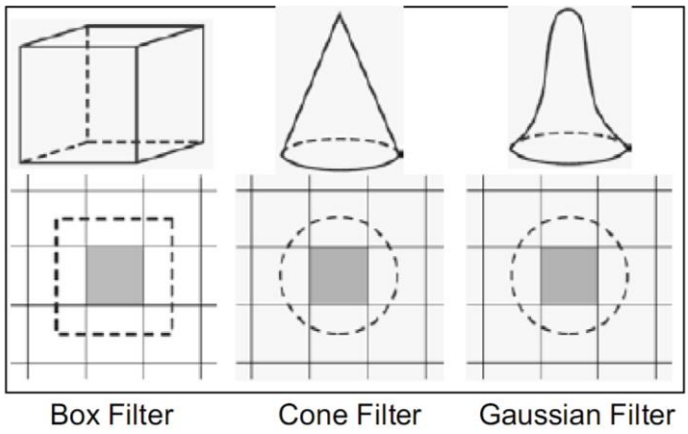
Area Sampling

Man berechnet die Pixelintensität, indem man die Überschneidungsflächen von jedem Pixel mit den Objekten, die angezeigt werden sollen, ermittelt. Diese Überschneidungsflächen bekommt man, indem man bestimmt, wo sich die Objektgrenzen mit den Pixelgrenzen schneiden.



Filtering Techniques

Eine exaktere Methode zum Antialiasen von Linien ist die Anwendung von Filtertechniken. Diese Methode ist ähnlich der Anwendung von gewichteten Pixelmasken, aber jetzt stellen wir uns eine durchgehende „Gewichtungsoberfläche“ vor, die die Pixel bedeckt. Es wird über die Pixeloberfläche Integriert, um die gewichtete Durchschnittsintensität zu erhalten. Um die Berechnungen zu reduzieren, werden üblicherweise Table lookups zur Evaluierung des Integrals verwendet.

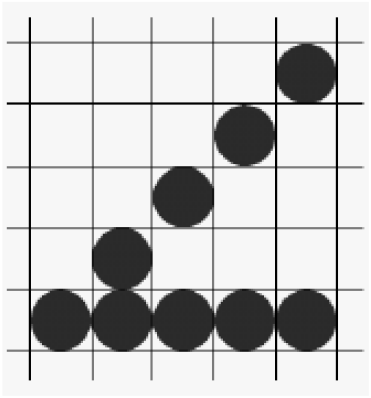


Pixel Phasing

Bei Rastersystemen die Subpixelpositionen innerhalb des Anzeigegitters adressieren können, kann Pixelphasing zum Antialiasen eines Objekts verwendet werden. Stufen entlang eines Linienpfades oder Objekts werden geglättet, indem der Elektronenstrahl zu einer besser passenden Position („Mikropositionierung“), die durch die Objektgeometrie spezifiziert wird, verschoben wird. Ein Elektronenstrahl wird normaler Weise um ¼, ½, oder ¾ eines Pixeldiameter verschoben, um Punkte näher am echten Pfad einer Linie oder Objektkante auszugeben.

Compensating for Line Intensitiy Differences

Das Glätten kompensiert auch einen anderen Rastereffekt: Linien mit unterschiedlichen Steigungen erscheinen unterschiedlich intensiv, da z.B. bei einer horizontalen Linie mehr Pixel ausgegeben werden, als bei einer diagonalen. Dadurch hat die Diagonale eine geringere Intensität pro Einheit als die Horizontale. Dies kann behoben werden, indem der Zeichenalgorithmus die Intensität abhängig von der Steigung der Linie anpasst. Hierbei würden horizontale und vertikale Linien am schwächsten, Linien mit 45° Steigung am stärksten dargestellt werden. Wenn aber Antialiasingtechniken verwendet werden, werden die Intensitäten automatisch kompensiert.



Antialiasing Area Boundaries

Die schon diskutierten Antialiasing-Methoden können auch auf Objekt- und Flächengrenzen angewandt werden. Wir können diese Prozeduren in einen Scanline-Algorithmus integrieren, der die Kanten der Objekte beim Zeichnen glättet. Wenn das System die Repositionierung von Pixeln unterstützt, kann das Antialiasing mit der Neupositionierung der Randpixel erreicht werden. Andere Methoden passen die Intensität jedes Pixels entsprechend der Prozent der Pixelfläche, die innerhalb der Grenzen liegen, an. Supersamplingmethoden können angewandt werden, indem die gesamte Fläche sub-unterteilt wird und die Subpixel innerhalb der Flächengrenzen bestimmt werden.

24) Wofür verwendet man super-sampling und area-sampling, und was ist der Unterschied?

Beides sind Konzepte für das Antialiasing

Supersampling (S. 215)

Für eine Grauwertdarstellung eines Liniensegments können wir jedes Pixel in eine bestimmte Anzahl von Subpixeln aufteilen und die Subpixel zählen, die entlang des Linienpfades liegen. Der Intensitätslevel des Pixels wird dann auf einen Wert gesetzt, der proportional zu dieser Subpixelzählung ist. So wird die Intensität der Linie auf eine größere Anzahl an Pixel aufgeteilt und der Treppeneffekt wird durch die etwas verschwommene Darstellung des Linienpfades geglättet. Für mehr unterschiedliche Intensitätslevels zum Antialiasing, erhöht man die Anzahl von Samplingpositionen entlang jedes Pixels. Diese Berechnungen basieren auf der Annahme, dass die Linie keine Dicke aufweist. Üblicherweise haben aber Linien bei der Darstellung ungefähr eine Dicke von 1 Pixel.

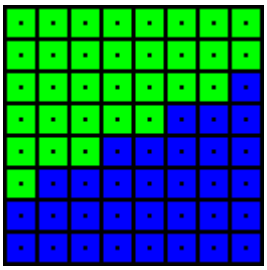
Wenn wir nun eine endliche Dicke der Linie mit einbeziehen, können wir die jeweilige Pixelintensität proportional zu der Anzahl der Subpixel innerhalb des Polygons, das die Linienflache repräsentiert, setzten. Ein Subpixel kann dabei als innerhalb der Linie angesehen werden, wenn die untere, linke Ecke sich innerhalb des Polygons befindet. Der Vorteil dieser Methode ist, dass die Anzahl der Intensitätslevels für jedes Pixel gleich der Anzahl der Subpixel innerhalb der Pixelfläche sind. Ein weiterer Vorteil von Supersampling mit einer Liniendicke ist, dass die totale Linienintensität über mehr Pixel verteilt wird.

Für ein Farbdisplay können wir diese Methode so modifizieren, dass sie die Hintergrundfarbe mit einbezieht. Eine Linie könnte mehrere verschiedene Hintergrundfarben schneiden und wir können den Durchschnitt der Subpixelintensitäten verwenden um die korrekte Farbe zu berechnen.

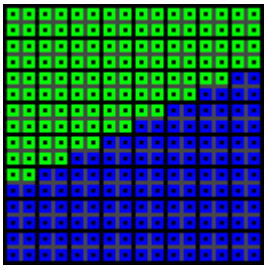
Ein Nachteil ist der hohe Berechnungsaufwand für die Identifizierung der Subpixel entlang des Linienpfades. Dies wird zusätzlich durch die Positionierung der Liniengrenzen in Relation zum Linienpfad erschwert. Diese Positionierung hängt von der Steigung der Linie ab. Bei einer Linie mit 45° ist der Linienpfad zentriert zum Polygonfeld; aber bei jeder horizontalen/ vertikalen Linie soll der Pfad die Polygongrenze sein. Für Linie mit einer Steigung $|m|<1$ ist der mathematische proportional näher an der unteren Polygongrenze positioniert, für Linien mit $|m|>1$ näher an der Oberen.

Supersampling Beispiel Quelle

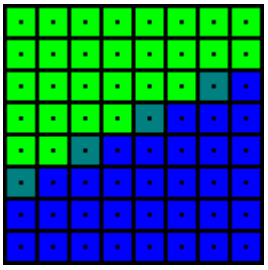
In diesem Beispiel sollen zwei Polygone dargestellt werden. Bei der normalen Darstellung wird die Farbe des Pixels durch die Farbe des Polygons im Mittelpunkt des Pixels bestimmt. An den Kanten der Polygone entsteht dadurch der Treppeneffekt.



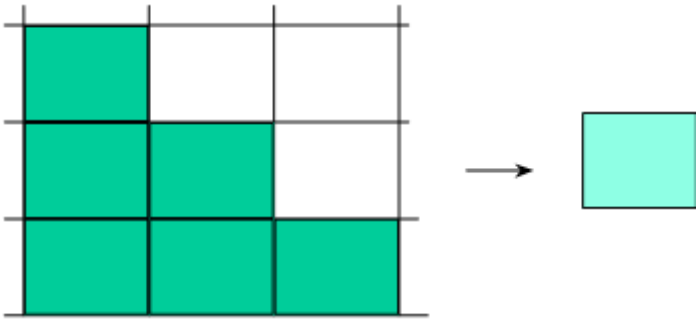
Wird nun jedes Pixel in ein 2x2 Raster unterteilt und die Farbe entsprechend den neuen Mittelpunkten berechnet, dann ergibt sich folgendes Bild:



Die endgültige Farbe ergibt sich nun aus dem Mittel jedes 2x2 Rasters. Dabei entstehen entlang der Trennungslinie zwischen den Polygonen Mischfarben.



Ein anderes Beispiel, bei dem links die „Ergebnisse“ der höher aufgelösten Subpixel zu sehen sind. Rechts das Ergebnis (und somit der „Durchschnittswert“ der links abgebildeten Subpixel) für das reale Pixel welches man mit Supersampling berechnete.



Area Sampling (S. 217)

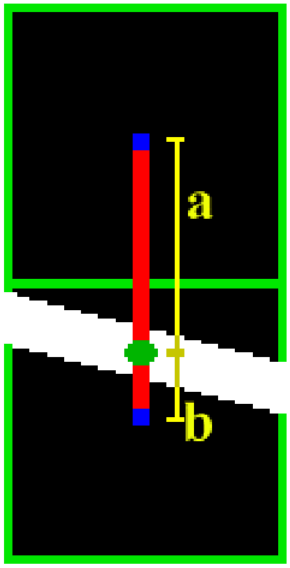
Sampling wird durch das Setzten der Pixelintensität proportional zur Fläche der Überlappung des Pixels mit der begrenzt-dicken Linie durchgeführt. Die Linie kann dabei als Rechteck behandelt werden, und die Sektion der Linienfläche zwischen 2 adjazenten (angrenzend, anliegend) vertikalen/horizontalen Bildschirmgitterlinien als Trapezoid. Nun können die Überlappungsflächen der Pixel dadurch berechnet werden, wie viel jedes Pixel in der vertikalen/horizontalen Linie vom Trapezoid überdeckt wird.

Eine Methode um die Überlappungsfläche zu schätzen ist, mit Supersampling die Anzahl der Subpixel innerhalb der Liniengrenzen zu zählen, da diese ca. der Fläche entsprechen. Diese Schätzung wird durch verfeinern des Pixelgitters verbessert.

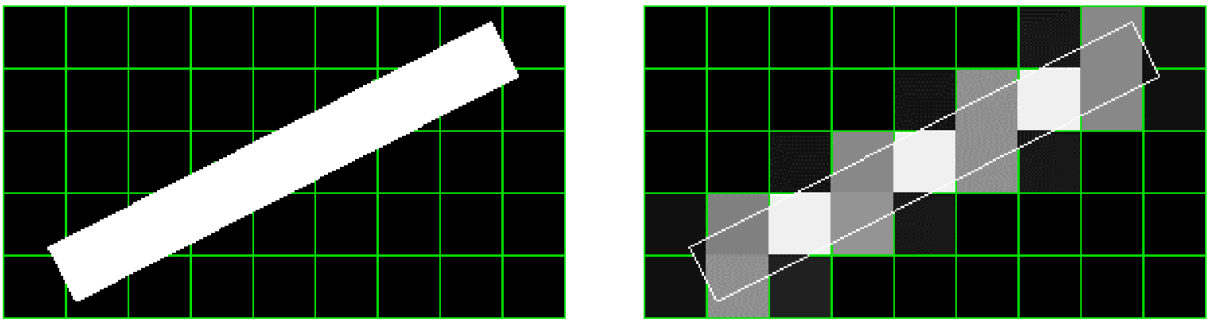
Bei Farbanzeigen werden die Flächen der Pixelüberlappung mit anderen Farbregionen berechnet und für die endgültige Pixelfarbe das Mittel über die verschiedenen Überlappungsflächen genommen.

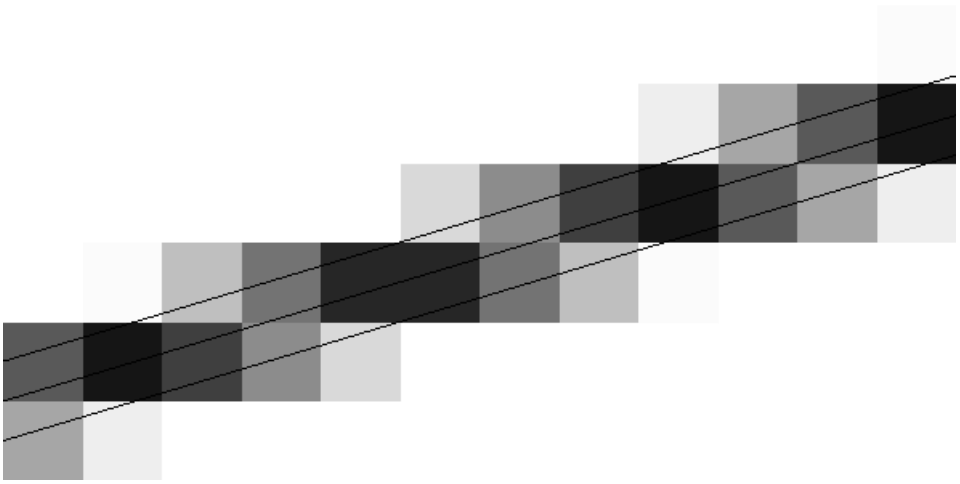
Area Sampling Beispiel Quelle1 Quelle2

Die Intensität ist proportional zu dem Bereich (area) in dem das Objekt sich befindet. Vereinfacht: Intensität proportional zu (1-Distanz zum Pixel):



Beispiel einer Linie die links gegeben ist, und rechts das entsprechende Ergebnis:





25) Was versteht man unter Oversampling? (S. 214) Quelle

In der digitalen Signalverarbeitung spricht man von Überabtastung oder engl. Oversampling, wenn ein Signal mit einer höheren Abtastrate bearbeitet wird, als für die Darstellung der Signalbandbreite benötigt wird.

Eine Überabtastung eines Signals kann applikative Vorteile haben. Einige dieser Applikationen sind:

- Digital-Analog-Wandlung
- Analog-Digital-Wandlung
- SC-Filter

Praxis

In der Praxis werden beim Oversampling ganzzahlige Frequenzverhältnisse, vorzugsweise Zweierpotenzen verwendet. Das reduziert den Rechenaufwand. Bei Oversampling höherer Ordnung $(k \geq 8)$ wird häufig das notwendige Resampling mehrstufig durchgeführt.

Höhere Abtastraten werden hierbei dadurch erreicht, dass im Frequenzbereich die Summen- und Differenzbänder bei ungeradzahligen Vielfachen der Abtastfrequenz entfernt werden. Dadurch treten im Zeitbereich doppelt so viele Abtastwerte auf, die Abtastrate ist also verdoppelt. Dieses Verfahren nennt man *Zweifach-Oversampling*. Bei *Vierfach-Oversampling* werden die Summen- und Differenzbänder auch bei geradzahligen Vielfachen, außer bei 4*n, der Abtastfrequenz entfernt.

Entsprechend dem Nyquist-Shannon Abtasttheorem muss die Abtastrate über dem Doppelten der höchsten vorkommenden Signalfrequenz liegen, um eine fehlerfreie Rekonstruktion zu erlauben. Das Theorem setzt **ideale** Antialias- und Rekonstruktionsfilter voraus.

In der Praxis sind hierzu Filter nötig, die eine hohe Flankensteilheit und eine hohe Dämpfung haben (z. B. muss bei einem CD-Player der Filter zwischen 20 kHz und 22,05 kHz um ca. 100 dB fallen). Mit analoger Technik sind Filter mit solchen Anforderungen nicht sinnvoll möglich. Oversampling erlaubt es hingegen, die Filterung vom analogen in den digitalen Bereich zu verschieben. Die Filterung erfolgt mit einem Digitalfilter, am Ausgang ist dann nur noch ein sehr einfaches analoges Filter notwendig.

Oversampling führt **nicht** zu höheren Datenraten und höherem Speicherplatzverbrauch. Dieses Verfahren findet beim Auslesen und nicht beim Schreiben von Daten Anwendung.

Ein angenehmer Nebeneffekt ist, dass durch Oversampling der Störabstand, beispielsweise bei CD-Wiedergabe, verbessert wird. Die Rauschleistung wird durch Überabtastung gleichmäßig auf ein größeres Frequenzintervall verteilt.

Häufig wird Antialiasing auch falsch als *Oversampling* bezeichnet!

Kapitel 5 – Two-Dimensional Geometric Transformations (S. 230)

26) Welche Arten von Transformationen kennen sie (2D Transformationsmatrizen)?

Translation (Verschiebung): $\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$

Rotation (Drehung): $\begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Skalierung (Vergrößerung bzw. Verkleinerung): $\begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Scherung: $\begin{pmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Reflexion (Spiegelung): $\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ um x Achse, $\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ um y Achse

27) Welchen Vorteil hat die Matrizenschreibweise bei Transformationen? (S. ???)

Meist werden größere Teile (Objekte, Bilder) als Ganzes transformiert, daher auf jeden Punkt dieser Gebilde wird die gleiche Folge von Transformationen angewendet. Dies entspricht einer sequenziellen Multiplikation eines Punktes P mit Matrizen M_1, M_2, M_3 … : $P' = M_1 * P$, $P'' = M_2 * P'$, $P''' = M_3 * P''$, …

Nun kann man sich die Assoziativität der Matrizenmultiplikation [also $((M_1 * M_2) * M_3 = M_1 * (M_2 * M_3)$] zunutze machen und den Rechenaufwand damit massiv reduzieren.

Statt $P^{(n)} = M_n * (M_{n-1} * \dots (M_3 * (M_2 * (M_1 * P))) \dots)$ schreibt man $P^{(n)} = (M_n * M_{n-1} * \dots * M_3 * M_2 * M_1) * P$

Nun kann man $M = (M_n * M_{n-1} * \dots * M_3 * M_2 * M_1) *$ **vorher** ausrechnen und diese eine Gesamtmatrix dann auf alle Punkte anwenden.

28) Was sind homogene Koordinaten und warum verwendet man sie? (S. 238)

Homogene Koordinaten dienen häufig zur Umrechnung von Koordinatensystemen. Um Rundungsfehler zu vermeiden und Berechnungsaufwand massiv zu reduzieren, ist es vorteilshaft, mehrere aufeinander folgende Matrixtransformationen (Drehung, Skalierung, Scherung, Translation) zu einer einzigen Transformationsmatrix zusammenzufassen (möglich auf Grund der Assoziativität von Matrizen). Hinderlich ist jedoch, dass die drei erstgenannten Operationen eine Matrizenmultiplikation erfordern, die Translation jedoch eine Addition.

Um eine Translation ebenfalls als Multiplikation berechnen zu können, wird der Raum um eine weitere Dimension erweitert. Eine Translation im dreidimensionalen Raum lässt sich nun durch eine Matrizenmultiplikation mit einer 4x4-Matrix beschreiben. Zu beachten ist, dass die Matrizen zwar assoziativ sind, aber nicht kommutativ. Daher es ist nicht egal, ob ich ein Objekt zuerst drehe und dann verschiebe, oder umgekehrt.

Aus „Skript“: Damit auch die Translation in Matrixschreibweise angegeben werden kann, verwendet man *homogene Koordinaten*. Jedem Punkt wird eine zusätzliche Koordinate h zugeordnet, wobei die Umrechnung in 2D Koordinaten durch Division der x- und y-Komponente durch h erfolgt. Daher verwendet man meist h=1. Für den Punkt (x, y) schreiben wir daher (x, y, 1). Die Transformationsmatrizen werden um eine Zeile und Spalte mit Einheitswerten erweitert.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D-Rotation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D-Skalierung

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D-Translation

Siehe auch Frage zuvor!

29) Was ist eine affine Transformation, und welche Eigenschaften hat sie? (S. 283)

Affine Transformation wird auch als affine Abbildung oder lineare Transformation bezeichnet. Dies ist eine Transformation zwischen zwei Vektorräumen die Kollinearitäten und Abstandsverhältnisse bewahrt. Bewahrung der Kollinearität bedeutet, dass die Bilder von Punkten, die auf einer Geraden liegen (daher kollinear sind), wieder auf einer Geraden liegen. Ebenso sind die Bilder paralleler Geraden wieder parallel.

Beispiele für affine Transformationen sind Translation, Rotation, Skalierung, Reflexion und Scherung. Jede generelle Transformation lässt sich durch Komposition aus diesen fünf Grund-Transformationen zusammenstellen. Affine Transformationen, die nur aus Translation, Rotation und Reflexion bestehen, bewahren auch die Winkel und die Längen.

Aus Textblättern: „Alle behandelten Transformationen sind affine Transformationen, daher die Koordinaten lassen sich durch lineare Funktionen plus einer Translation ineinander überführen. Affine Abbildungen erhalten Kolinearität, daher (je) 3 Punkte auf einer geraden Linie sind auch nach der Abbildung auf einer geraden Linie, und Proportionalität von Abständen entlang einer gerade Linie, daher das Verhältnis von Längen auf einer Geraden bleibt erhalten. Weiters bleiben parallele Linien immer parallel und endliche Punkte bleiben im Endlichen. Alle affinen Transformationen lassen sich aus Skalierung, Rotation und Translation zusammensetzen (auch die Scherung!). Affine Transformationen, bei denen nur Rotation, Translation und Spiegelung verwendet werden, sind überdies längen- und winkelerhaltend."

Kapitel 6 – Two-Dimensional Viewing (S. 296)

30) Was versteht man unter der Grafikpipeline (Viewing Pipeline) und aus welchen Komponenten besteht sie? (S. 299)

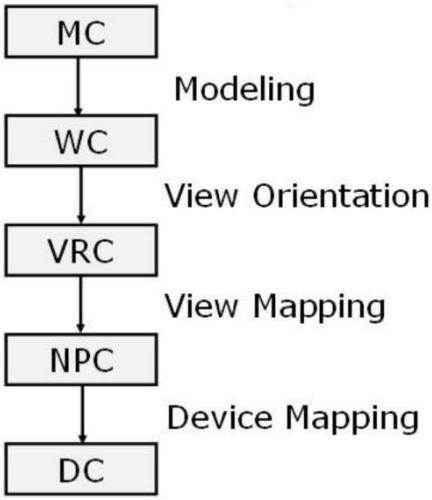
Allgemein

Die Viewing Pipeline ist die Abbildung des Vorganges, der ein Modell einer realen Welt auf einem Gerät (Bildschirm) abbildet.

- Modeling: MC (Modellkoordinaten) ⇒ WC (Weltkoordinaten): Konstruiere eine Weltkoordinaten-Szene unter Verwendung von Modell-Koordinaten-Transformation – Beschreiben der Szene in Weltkoordinaten
- View Orientation: WC ⇒ VC (Viewingkoordinaten): Wandle Welt-Koordinaten in Viewing-Koordinaten um – Szene aus der Sicht der Kamera

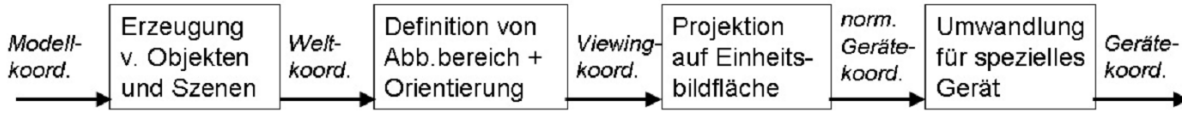
- View Mapping: VC ⇒ NVC (normalisierte Viewingkoordinaten): Wandle VCs in normalisierte VCs mittels Window-Viewport-Transformation – Überführe die Szene in einen Einheitswürfel
- Device Mapping: NVC ⇒ DC (gerätespezifische Gerätekoordinaten): Wandle die NVCs in Device Koordinaten um – Projeziere die Szene auf den Bildschirm

Bei der 3D-Graphik-Pipeline kommt noch hinzu, dass man das „Verstecken“ von Objekten (nicht sichtbare Objekte können außer Acht gelassen werden) sowie die Licht- und Schatteneffekte (unter Berücksichtigung der vorhandenen Lichtquellen) miteinbeziehen muss.



2D Viewing Pipeline

Unter Viewing-Pipeline versteht man die Folge von Umwandlungen, die geometrische Daten durchlaufen um schließlich als Bilddaten auf einem Gerät dargestellt zu werden. Die 2D-Viewing-Pipeline beschreibt diesen Vorgang für 2D-Daten:



Man hat einen Viewport durch den man eine „Szene“ sieht. Dieser Viewport entspricht einem Canvas das man „bewegen“ kann um auch andere Teile der „Szene“ zu sehen. Die Größe des Viewports könnte man auch ändern. Deswegen muss man die Weltkoordinaten in Viewport Koordinaten umwandeln, Objekte die außerhalb des Viewports liegen korrekt clippen (z.B. mithilfe des Sutherland-Hodgman-Algorithmus), sodass man innerhalb des Viewports korrekte Kanten hat.

Aus „Textblätter“‘: Die Koordinaten, in denen einzelne Objekte konstruiert werden, nennt man **Modellkoordinaten**. Aus diesen Objekten werden Szenen zusammengestellt, diese befinden sich in **Weltkoordinaten**. Diese nennt man nach der Transformation in das Kamerakoordinatensystem die **Viewingkoordinaten**. Die Abbildung eines Fensters der Szene (bei uns sind zum Beispiel die „Screen Coordinates“ zwischen [-1,... 1 und -1,... 1]) erfolgt in geräteunabhängiger Weise in **normalisierte Koordinaten**. Schließlich werden diese normalisierten Werte in **gerätespezifische Gerätekoordinaten** abgebildet (zum Beispiel könnte das Canvas 200x300 Pixel haben).

3D Viewing Pipeline

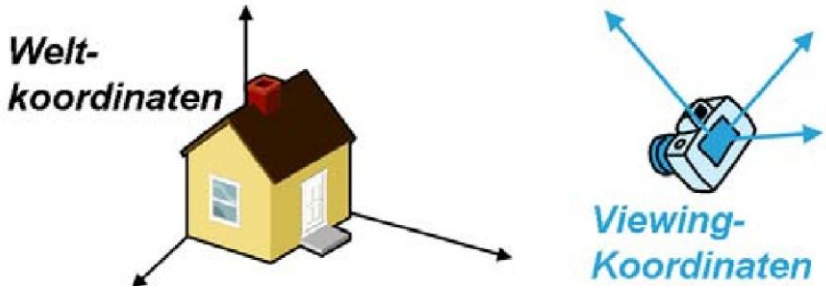
Die Viewing-Pipeline im 3-Dimensionalen ist mit der 2D-Viewing-Pipeline fast ident. Lediglich nach der Definition der Blickrichtung und Orientierung (also der Kamera) erfolgt noch ein zusätzlicher Projektions- Schritt, also die Reduktion der 3D-Daten auf eine Projektionsebene:

Dieser Projektionsschritt kann beliebig aufwändig sein, je nachdem welche der 3D-Viewing-Konzepte dabei einfließen sollen.

Viewing-Koordinaten:

Ähnlich wie beim Fotografieren hat man beim Festlegen der Kamerawerte mehrere Freiheitsgrade:

1. Position der Kamera im Raum
2. Blickrichtung von dieser Position aus
3. Orientierung der Kamera (wo ist oben?)
4. Größe des Bildausschnittes (entspricht der Brennweite bei einem Fotoapparat)



Mit diesen Parametern legt man das *Kamerakoordinatensystem* fest (Viewing-Koordinaten). Normalerweise ist die xy-Ebene dieses Viewing- Koordinatensystems normal auf die Hauptblickrichtung, und man *blickt in die Richtung der negativen z-Achse*.

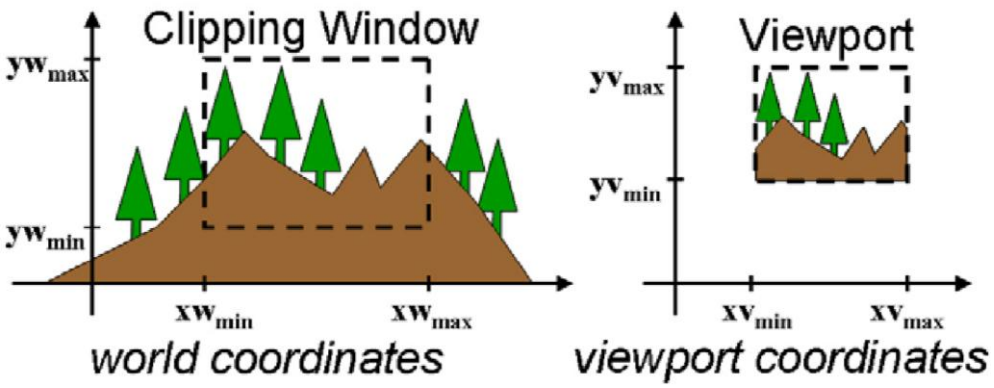
31) Was ist die window-viewport-Transformation? (S. 217)

Bei der window-viewport-Transformation wird ein Teil (Ausschnitt) der in Welt-Koordinaten vorhandenen Objekte in die viewport-Koordinaten transformiert und dargestellt. Dabei bestimmt das clipping-window welcher Teil angezeigt wird und der viewport, welcher Ausschnitt transformiert wird, wo das Ergebnisfenster liegt, sowie wie das Fenster verschoben, skaliert oder verdreht wird.

Die Transformation wird in zwei Schritten erreicht:

1. Translation des view-Ursprung in den Welt-Ursprung
2. Rotation, sodass die Achsen sich überlappen

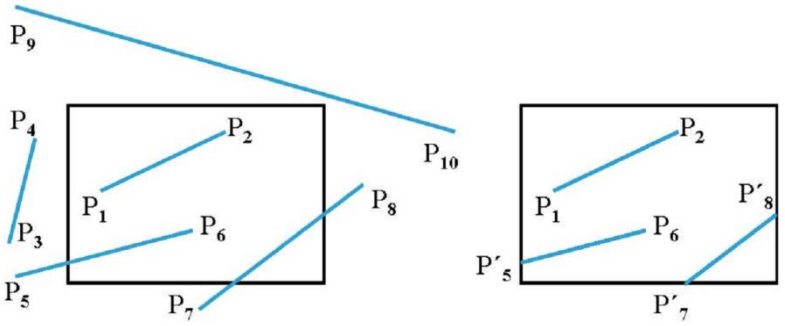
Bei der Transformation der Punkte bleiben die relativen Positionen erhalten.



32) Welche grundsätzlichen Möglichkeiten für das Clipping gibt es?

Clipping stellt sicher, dass nur die Teile der Objekte dargestellt werden müssen, die noch sichtbar sind. Dazu gibt es grundsätzlich 3 Möglichkeiten:

- analytisch = in Welt-Koordinaten (reduziert den Aufwand bei WC ⇒ DC Transformation). Zum frühestmöglichen Zeitpunkt.
- bei der Rasterkonvertierung = als Teil des Rasterisierungs- Algorithmus (effizient für komplexe Primitive).
- auf Pixel-Ebene = primitivster Algorithmus mit größtem Aufwand. Nach allen Berechnungen, erst unmittelbar vor der Zeichnung.



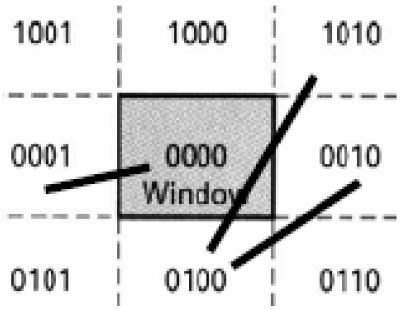
33) Was sind die wesentlichen Schritte beim Cohen-Sutherland line clipping Algorithmus? (S. 317)

Algorithmus zum Clippen von Linien nützen i.A. die Tatsache aus, dass jede Linie in einem rechteckigen Fenster höchstens einen sichtbaren Teil besitzt. Weiters gilt es Grundprinzipien der Effizienz auszunutzen, etwa häufige einfache Fälle früh zu eliminieren und unnötige teure Operationen (Schnittpunkt-Berechnungen) zu vermeiden.

Vorgehensweise beim Cohen-Sutherland Algorithmus

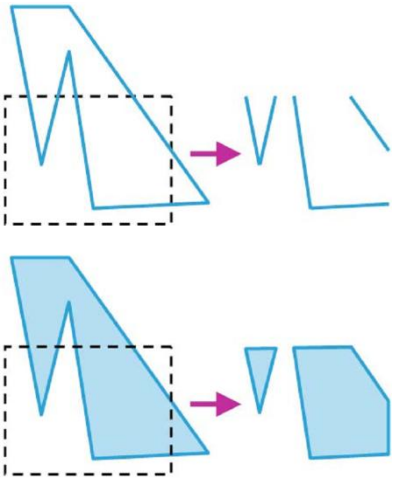
- Jedem Endpunkt des clip windows wird eine 4-stelliger region code zugeordnet
- Entscheiden, welche Linien komplett außerhalb des clip windows und welche komplett innerhalb liegen ⇒ können vernachlässigt werden. Entscheidung kann anhand einer „OR“ und „AND“-Verknüpfung der Bitmuster der region codes in denen die Endpunkte der Linie liegen, durchgeführt werden

- Alle verbleibenden Linien müssen mit den Randlinien des clip windows (links, rechts, unten, oben) geschnitten werden. Dabei wird der Teil der Linie außerhalb des clip windows vernachlässigt.
- intersection test (Punkt zuvor) wird bis zu 4x durchgeführt (da 4 Randlinien des clip windows)



34) Wie passiert das Clippen an beliebigen Polygonen? (S. 316)

Die Linien des Polygons werden nacheinander gegen das clip window geschnitten (links, rechts, unten, oben) und die entstehenden Schnittpunkte als neue Punkte in das Polygon aufgenommen. Dabei entstehen auch neue Kanten, die die neuen Punkte entlang des Randes des clip windows miteinander verbinden. Ein Weg hierfür ist jeweils an jeder Clipping-Grenze eine neue Vektorenliste zu generieren, und dann diese an die nächste Clipping-Grenze weiterzugeben. Hierbei sollte zum Schluss die Vektorenliste des geclippten Polygons heraus kommen.

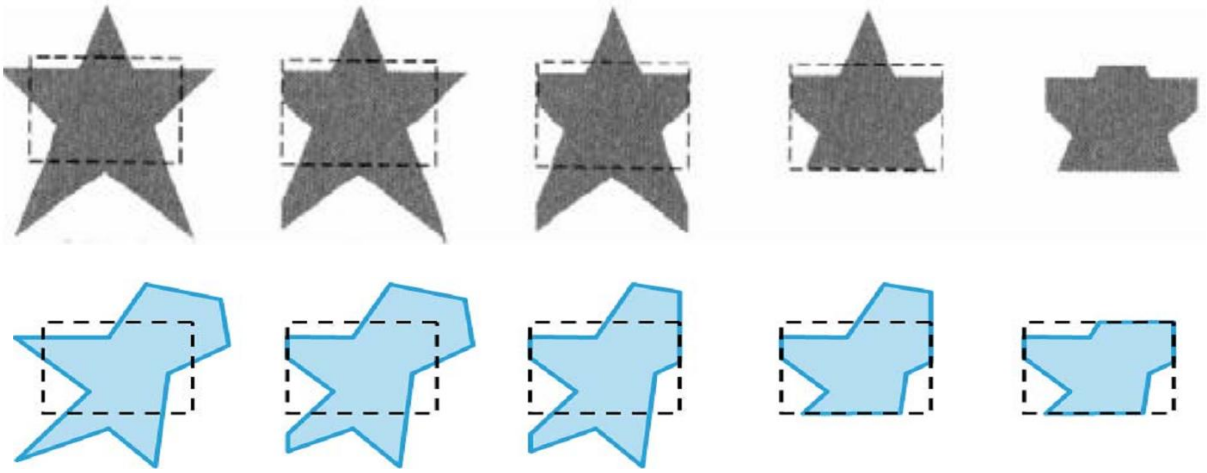


Das Clippen von Polygonen hat zu berücksichtigen, dass nach dem Clippen als Resultat wieder ein Polygon erzeugt wird, auch wenn durch den Clipping-Vorgang mehrere Teile entstehen. Die obere Abbildung zeigt ein Polygon, dass mit einem Linien-Clipping-Algorithmus geclippt wurde. Es ist nicht mehr erkennbar, was innen und was außen ist. Das untere Bild zeigt das Ergebnis eines korrekten Polygon-Clipping-Verfahrens. Das Polygon zerfällt in mehrere Teile, die alle korrekt gefüllt werden können.

35) Was sind die wesentlichen Schritte beim Algorithmus von Sutherland und Hodgman? (S. 238)

Das Polygon wird gegen die 4 clip-window-Kanten (links, rechts, unten, oben) getrennt geschnitten. Um Zwischenergebnisse zu verhindern, werden die Schnittpunkte rekursiv durch die Tests geschickt

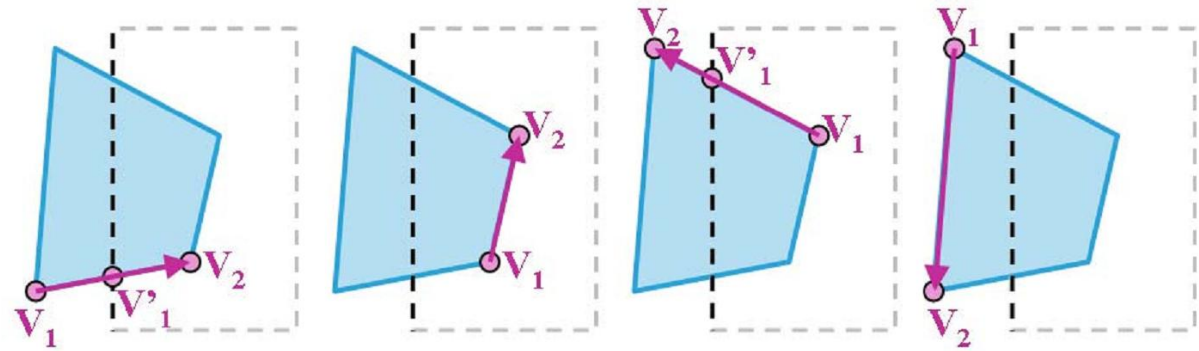
(boundary clipping pipeline) und somit entsteht nur eine Ergebnisliste. Die Grundidee kommt von der Erkenntnis, dass beim Clippen an nur einer Kante keine gröberen Komplikationen entstehen.



Beim Clippen eines Vektorenpaars gibt es 4 Mögliche Situationen, die unterschieden werden müssen:

I. Beide Punkte sind innerhalb der Fenstergrenze: Nur der 2. Eckpunkt wird zum nächsten Clipper weitergegeben (Grafik unten Bild 2).
II. Beide Punkte sind außerhalb der Fenstergrenze: Kein Eckpunkt wird zum nächsten Clipper weitergegeben (Grafik unten Bild 4).
III. Der 1. Punkt ist innen, der 2. Punkt außen: Nur der Schnittpunkt des Polygons mit der Fenstergrenze wird weitergegeben (Grafik unten Bild 3).
IV. Der 1. Punkt ist außen, der 2. Punkt innen: Sowohl der Schnittpunkt des Polygons mit der Fenstergrenze, als auch der 2. Eckpunkt werden weitergegeben (Grafik unten Bild 1).

Der letzte Clipper in dieser Serie generiert anschließend die neue Liste der Eckpunkte, die das geclippte Polygon beschreiben.



1. out→in (output V'_1, V_2) 2. in→in (output V_2) 3. in→out (output V'_1) 4. out→out (kein output)

Problem beim Clippen konkaver Polygone

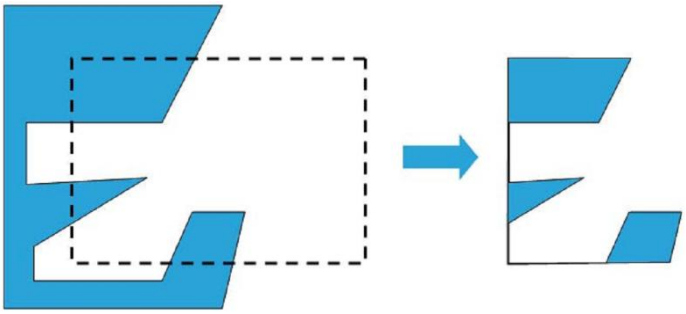
Ein Problem (S. 335) bei dieser Methode stellt das clippen konkaver Polygone dar, da die durch das clippen entstehenden separaten Polygone durch eine Linie verbunden werden. Es gibt zwei Möglichkeiten um dies zu verhindern:

I. Aufteilung der Polygone vor dem Clippen in konvexe Polygone

II. Überprüfung der finalen Eckpunktliste auf multiple Schnittpunkte entlang irgendeiner Fenstergrenze. Wenn mehr als 2 Schnittpunkte entdeckt werden, können wir die Liste in 2 oder mehrere einzelne Listen aufteilen, um so eine korrekte Beschreibung des geclippten Polygons zu erreichen.

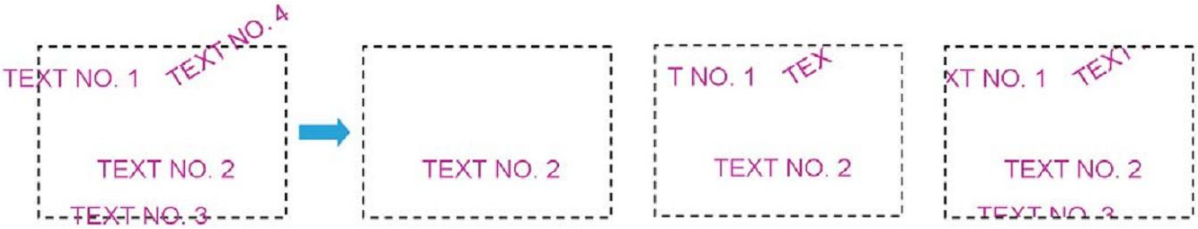
Problem wenn Polygon in mehrere Teile zerfällt

Wenn ein Polygon beim Clippen in mehrere Teile zerfällt, dann erzeugt dieses Verfahren Verbindungskanten entlang des Clippingfensters. Eine nachträgliche Kontrolle und eventuelle Nachbearbeitung ist in solchen Fällen notwendig.



36) Welche Möglichkeiten des Text-Clipping kennen Sie, und was sind die Unterschiede?

Das Clippen von Text erscheint auf den ersten Blick trivial, es muss allerdings eine kleine Feinheit beachtet werden. Je nach Erzeugungsweise der Buchstaben kann es passieren, dass nur Texte angezeigt werden, die komplett lesbar sind (wo also alle Buchstaben komplett im Fenster liegen), dass Text nur buchstabenweise geclippt wird (also alle Buchstaben ganz verschwinden, die nicht ganz im Fenster sind), oder dass Text korrekt abgeschnitten wird (also auch halbe Buchstaben erzeugt werden).



Es gibt mehrere Methoden um Buchstaben zu clippen. Diese hängen vor allem davon ab, wie die Buchstaben in der jeweiligen Applikation generiert werden und was für Anforderungen wir für die Anzeige selbiger haben.

„all-or-none string-clipping"-Methode

Einfachste Methode die eine bounding box um den Text benutzt. Wenn der String zur Gänze im sichtbaren Bereich liegt wird er dargestellt, ansonsten nicht. Dies wird durch Überprüfung der Koordinatenmaxima der bounding box erreicht (Grafik oben Ergebnisbild 1).

„all-or-none character-clipping"-Methode

Sie ist gleich wie die obige, nur das die bounding box um einzelne Buchstaben gelegt wird (Grafik oben Ergebnisbild 2).

„individual character“-Method

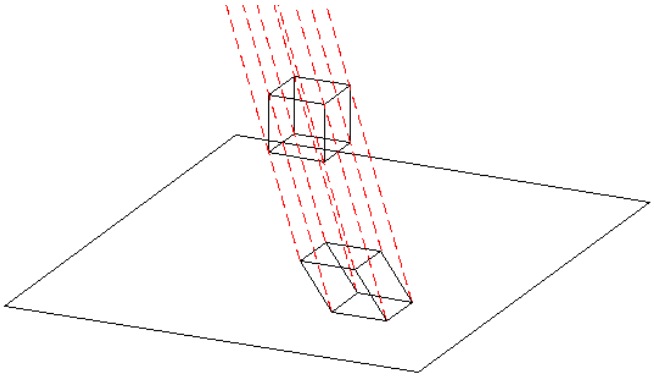
Bei dieser Methode werden die Buchstaben ähnlich wie Polygone behandelt. Buchstaben die durch ihre Außenlinien definiert sind (outline font) werden gleich wie Polygone geclippt. Bei Buchstaben die als Bitmaps vorliegen (bitmap font), werden die relativen Positionen der Pixel in den Buchstaben mit den Clipping-grenzen verglichen (Grafik oben Ergebnisbild 3).

Kapitel 7 – Three Dimensional Viewing (S. 344)

37) Was ist eine Parallelprojektion und welche Eigenschaften hat sie? (S. 346)

Bei der Parallelprojektion sind eine Projektionsebene und eine Projektionsrichtung gegeben. Den Bildpunkt eines beliebigen Punktes im Raum erhält man dadurch, dass man die Parallele zur Projektionsrichtung durch diesen Punkt mit der Projektionsebene zum Schnitt bringt.

Geraden werden durch eine Parallelprojektion im Allgemeinen wieder auf Geraden abgebildet. Das gilt jedoch nicht für Parallelen zur Projektionsrichtung, da diese in Punkte übergehen. Die Bildgeraden von parallelen Geraden sind - soweit definiert - ebenfalls parallel zueinander. Die Länge einer Strecke bleibt nur dann erhalten, wenn diese parallel zur Projektionsebene verläuft; in allen anderen Fällen erscheinen Strecken in der Projektion verkürzt. Auch die Größe eines projizierten Winkels stimmt normalerweise nicht mit der Größe des ursprünglichen Winkels überein. Aus diesem Grund wird ein Rechteck im Allgemeinen auf ein Parallelogramm abgebildet, aber nur in Ausnahmefällen auf ein Rechteck. Ähnliches gilt für Kreise, die im Allgemeinen in Ellipsen übergehen.

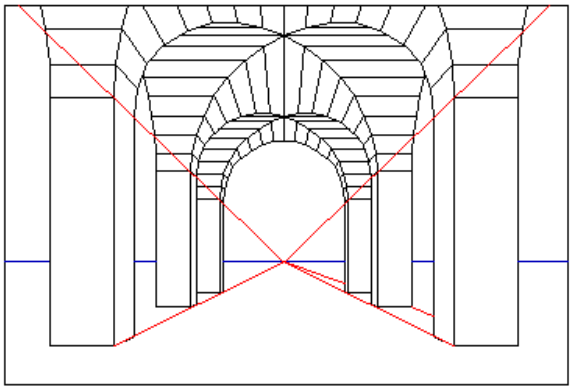


Ein wichtiger Spezialfall ist die orthogonale (senkrechte) Parallelprojektion. Sie ist dadurch gekennzeichnet, dass Projektionsrichtung und Projektionsebene zueinander senkrecht sind. Die orthogonale Parallelprojektion entspricht einer Fotografie mit einem starken Teleobjektiv.

Unterschied zur perspektivischen Projektion

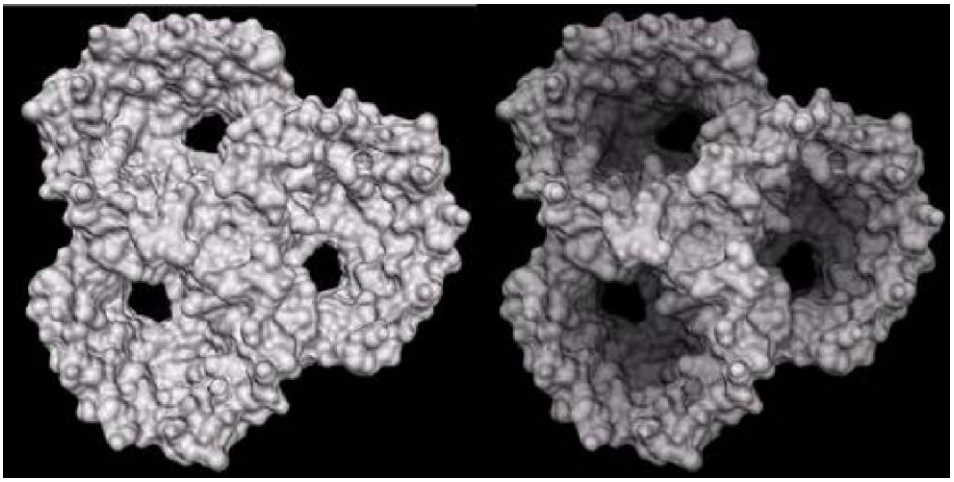
Bei der Parallelprojektion findet keine Entfernungsverzerrung statt, somit sind weiter weg liegende Objekte gleich groß wie vorne liegende (gleicher Größe).

Bei der perspektivischen Projektion hat man eine Entfernungsverzerrung. Dadurch können Objekte die im Hintergrund sind, viel größer sind als Objekte die im Vordergrund sind, viel kleiner sein/ wirken.



38) Was versteht man unter "depth cueing", und wofür wird es verwendet? (S. 346)

Die Information für die „Tiefe“ ist essentiell für das räumliche Erfassen einer Szene. Aufgrund der Blickrichtung kann man berechnen welche Seite dem Blickpunkt zugewandt ist (front face) und welche abgewandt (back face). Objekte die näher am Blickpunkt liegen, verdecken Objekte die weiter entfernt sind. Genauso verdecken einzelne nahe Polygon-Flächen solche, die weiter entfernt sind.



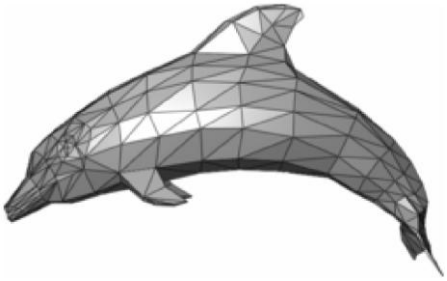
Vor allem werden bei Depth- Cueing Kanten oder Teile, die näher zum Betrachter liegen, intensiver dargestellt (heller, breiter, gesättigter), Kanten, die weiter weg liegen, weniger intensiv (dunkler, dünner, grauer).

Kapitel 8 – Three-Dimensional Object Representations (S. 402)

39) Was versteht man unter Polygonflächen (B-Rep-Listen)? (S. 403)

Unter Polygonfläche versteht man eine Boundary Representation („B-Rep“), also ein Set von Polygonen, die ein Objekt- Inneres einschließen.

Datenstrukturen für B-Reps enthalten neben geometrischer Information auch Attribute (Eigenschaften). Die Geometrie besteht aus Punktlisten (Vertex Table), Kantenlisten (Edge Table), Flächenlisten (Surface Table) und muss auf Konsistenz und Vollständigkeit überprüft werden.



Speicherung

Die Speicherung kann in polygon tables (sog. „B-Rep" Listen) erfolgen.

- Haben geometrische Listen (beinhalten vertex-Koordinaten, sowie Parameter zur räumlichen Identifizierung des Polygons und werden in 3 Listen gespeichert ⇒ vertex table, edge table und surface-facet table) und Attribut-Listen (Parameter für den Grad der Transparenz, Reflexion der Oberfläche)
- Ermöglichen Konsistenz- und Vollständigkeit-Checks

Ebenengleichung

Polygonflächen können durch eine Ebenengleichung dargestellt werden:

- Ax+By+Cz+D = 0
- Kartesische Koeffizienten: A,B,C,D
- Normale auf die Ebene = N (A,B,C)
- Distanz zum Ursprung = D

Polygon-Meshes

Effiziente Datenstruktur für zusammenhängende Polygone (Einsparung von Kanten und Ecken). Kann entweder aus Drei- oder Vierecken bestehen (triangle strp bzw. quadrilateral mesh)

40) Was versteht man und einem front- und back-face eines Polygons? Wie kann man diese mathematisch berechnen?

Die Objektseite, die in das Innere eines Objektes schaut, nennt man back-face und die sichtbare, nach aussen zeigende Seite heißt front-face.

Alles, was vor dem Polygon liegt, ist für das front-face sichtbar; alles, was hinter der Polygon-Fläche liegt, ist für das back-face sichtbar

Man kann dies für jeden Punkt mittels der Ebenengleichung berechnen.

- Ax+By+Cz+D = 0 ⇒ Punkt (x,y,z) liegt in der Ebene
- Ax+By+Cz+D < 0 ⇒ Punkt liegt hinter der Ebene
- Ax+By+Cz+D > 0 ⇒ Punkt vor der Ebene

- Die Orientierung einer Polygonoberfläche im Raum kann durch den Normalvektor bestimmt werden. Der Normalvektor zeigt in die Richtung vom Inneren eines Objekts nach außen ⇒ vom back-face in Richtung front-face.

41) Was sind Splines und welche Eigenschaften haben sie? (S. 420)

Ein Spline ist ein Begriff aus der numerischen Mathematik und bezeichnet ein stückweises Polynom, das für Polynome n-ten Grades an den Punkten, an denen jeweils zwei Polynome aufeinandertreffen, n-1 Mal stetig differenzierbar ist. Sind die einzelnen Polynome alle linear, so nennt man den Spline linear, analog gibt es quadratische, kubische usw. Splines. Der Begriff stammt aus dem Schiffbau: eine lange dünne Latte (Straklatte), die an einzelnen Punkten durch Nägel fixiert wird, biegt sich genau wie ein kubischer Spline mit natürlicher Randbedingung.

42) Was ist der Unterschied zwischen Achsenabhängigkeit und Achsenunabhängigkeit bei Kurven? (S. ???)

- Achsenabhängige Darstellung: Drehung des Koordinatensystems verändert Kurve
- Achsenunabhängige Darstellung: Drehung des Koordinatensystems verändert Kurve nicht

43) Was ist der Unterschied zwischen Interpolation und Approximation bei Splines? bzw. „Was ist der Unterschied zwischen interpolierenden und approximierenden Kurven?“ (S. 420)

Kontroll-Punkte

- Set von Punkten, die die generelle Form der Spline (Kurve) vorgeben
- Parametrisierung der Spline durch Veränderung der Position der Kontroll- Punkte
- Transformation der Spline durch Transformation der Kontroll-Punkte

Interpolation

- Die Kontroll-Punkte liegen auf der Spline (Kurve)

Approximation

- Die Kontroll-Punkte geben die ungefähre Form der Spline (Kurve) vor liegen aber nicht (unbedingt) auf ihr.

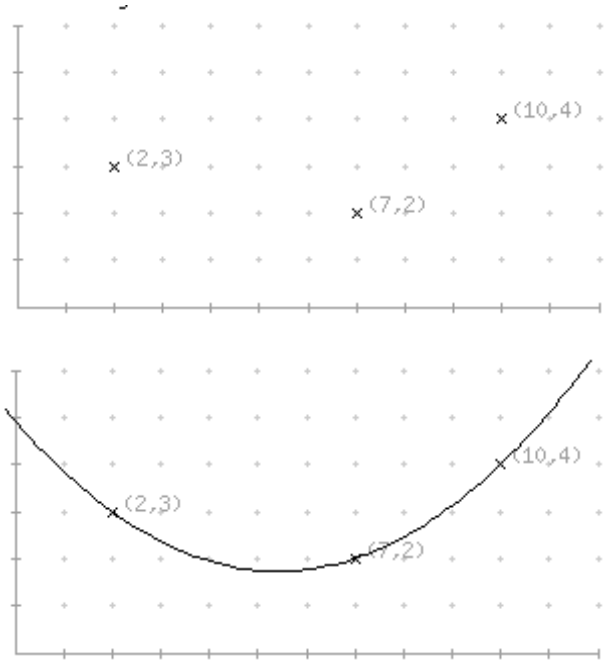
44) Was ist der Unterschied zwischen Stütz- und Kontrollpunkten? (S. 420)

Kurven, die durch Stütz- oder Kontrollpunkte definiert sind, nennt man Splines.

- Stützpunkte (interpolierend): Kurve verläuft durch diese Punkte
- Kontrollpunkte (approximierend): Punkte liegen neben der Kurve

45) Was ist die Lagrange-Interpolation durch Polynome? (S. ???) Quelle

Der eigentliche Sinn der LAGRANGE-Interpolation ist es, ein Polynom zu finden, welches durch die vorgegebenen Punkte läuft. Diese Aufgabenstellung wird auch das Interpolationsproblem genannt, welches unter anderem. mit der Lagrange-Interpolation zu lösen ist.



46) Welche Arten der Kontinuität bei Splines kennen Sie? Wie sieht der Unterschied zwischen C1- und G1-Stetigkeit aus? (S. 421 ff.)

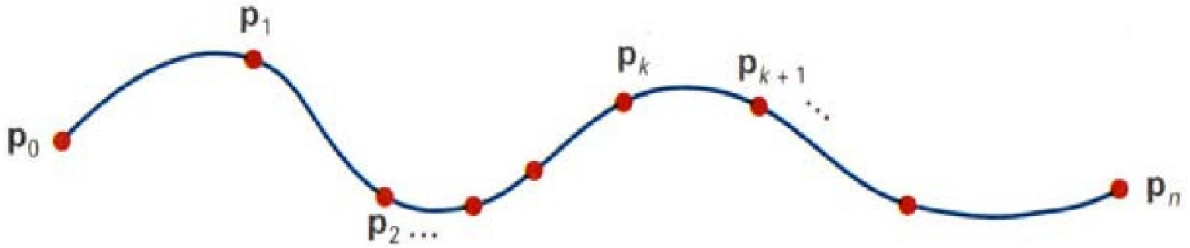
Parametrische Kontinuitäts-Bedingungen (Cn)

- Ableitungen an den Verbindungspunkten sind gleich
- C0 (zero order): die Kurven müssen sich treffen
- C1 (first order): die erste Ableitung (Tangente) muss im Verbindungspunkt gleich sein
- C2 (second order): die zweite Ableitung muss gleich sein

Geometrische Kontinuitäts-Bedingungen (Gn)

- Ableitungen an den Verbindungspunkten sind proportional
- G0 (zero order) = C0
- G1 (first order): die Richtung der Tangenten aber nicht unbedingt ihre Steigungen müssen gleich sein
- G2 (second order): auch die zweite Ableitung muss gleich sein

47) Was sind kubische Splines und welche Eigenschaften haben sie? (S. 425)

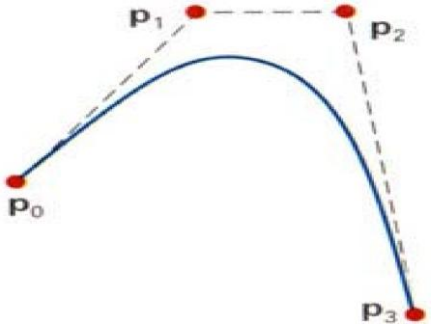


- Grad des Kontrollpolynoms = 3

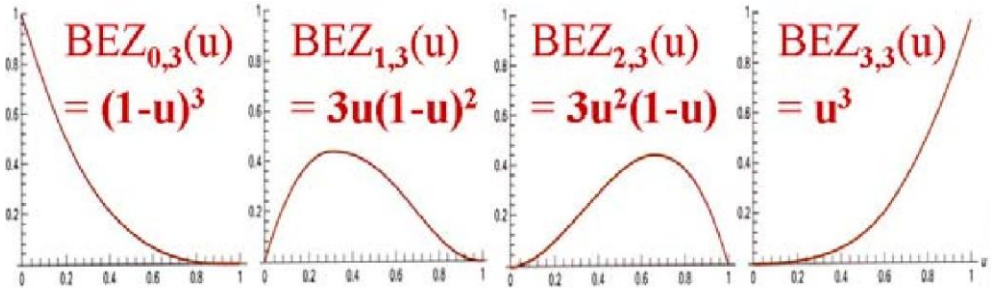
- nach der Interpolation und Erfüllung der restlichen Bedingungen bleiben noch zwei Freiheitsgrade, die für gewöhnlich dafür genutzt werden, Randbedingungen anzugeben. Je nach Art der Bedingungen heißt der Spline dann:
  - natürlich (freie Ränder, Krümmung gleich Null),
  - periodisch (linker und rechter Rand gehen stetig ineinander über),
  - vollständig (Steigung an den Rändern wie bei interpolierter Funktion) oder allgemein.

- Interpolation wird zur Generierung der Kurve verwendet
- C2-Kontinuität zwischen den Teilstücken muss gegeben sein
- Nachteil: Wenn sich die Position einer der Kontroll-Punkte verändert, verändert sich automatisch die ganze Kurve (keine „lokale Kontrolle“)

48) Was sind Bézier-Kurven und welche Eigenschaften haben sie? (S. 432)



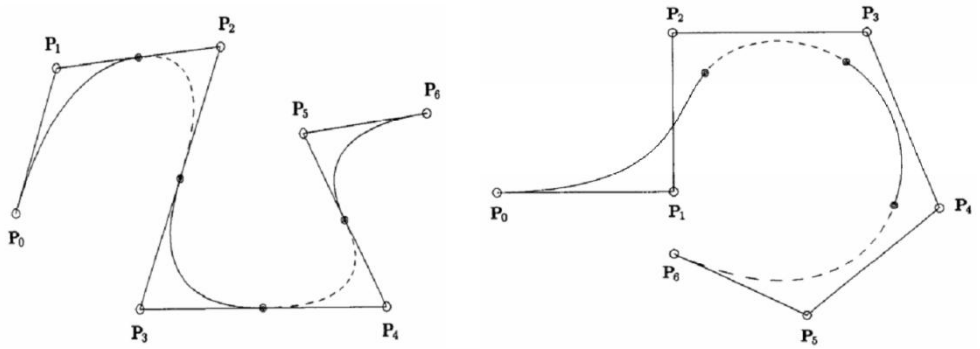
- Grad des Kontrollpolynoms = n-1, wobei n die Anzahl der Kontroll-Punkte angibt
- Kontroll-Punkte werden approximiert, wobei der erste und der letzte Kontroll-Punkt immer auf der Kurve liegt (= Eckpunkte werden interpoliert)
- C1-Kontinuität zwischen den Teilstücken muss gegeben sein
- Kurve liegt innerhalb der convex hull, die durch die Kontroll-Punkte aufgespannt wird
- Bézier-Kurven können geschlossen sein (P0 = Pn)
- Die Kurve kann näher an einen Punkte „gezogen“ werden indem man an dieser Stelle einen multiplen Punkt erstellt (Pi = Pj)
- Kontroll-Punkte haben (wie bei den kubischen Splines) globale Auswirkungen



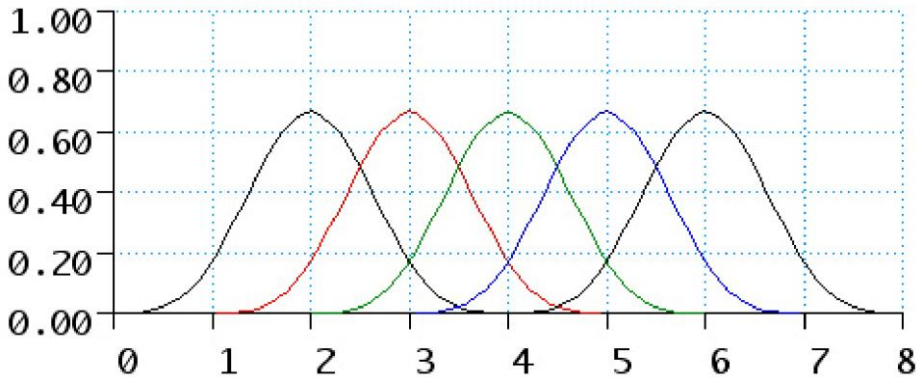
49) Welche Eigenschaften haben B-Splines und was sind die Unterschiede zu Bézier-Kurven? (S. 432)

Der Hauptnachteil der Bezierkurven ist der globale Einfluss der Kontrollpunkte auf die ganze Kurve. Dies hat zwei Hauptnachteile: (1) jede Veränderung der Kontrollpunkte (Einfügen, Entfernen,

Verschieben) verändert das Aussehen der Kurve an allen Stellen, und (2) die Rechenzeit für große Kontrollpunktmengen ist höher. Die Ursache liegt in der Form der Gewichtsfunktionen. Die sogenannten B-Splines sind ebenso wie die Bezier-Splines approximierende Kurven, jedoch sind die Bernsteinpolynome durch B-Spline-Polynome $B_{k,d}$ ersetzt. Diese beschränken die Anzahl der Kontrollpunkte, die einen Kurvenpunkt beeinflussen, auf d. Die Berechnung der $B_{k,d}$ ist etwas komplexer und erfolgt rekursiv, für das Verständnis reicht es allerdings, die Form der B-Spline-Polynome zu sehen. Man erkennt, dass jede Kurve nur in einem begrenzten Bereich ungleich null ist, dass also jeder Punkt über weite Bereiche keinen Einfluss auf die Kurve hat.



Eine wichtige Eigenschaft der B-Spline-Gewichtsfunktionen ist die Tatsache, dass für jeden Kurvenpunkt die Summe genau 1 ist. Jeder Kurvenpunkt ist also ein gewichteter Mittelwert aus den Kontrollpunkten.



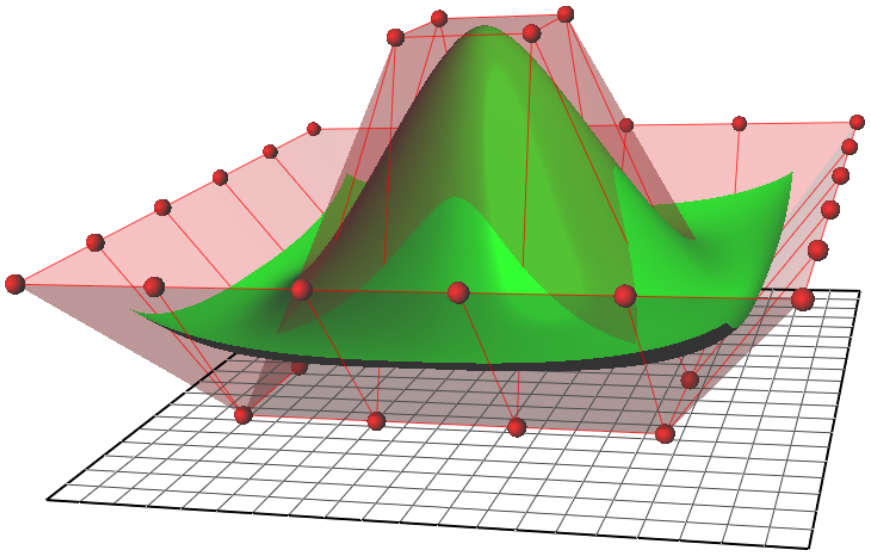
Unterschied B- Spline zur Bezierkurve

- Werden am öftesten zur Approximierung von Splines verwendet
- Vorteile gegenüber Bézier-Kurven:
  - Der Grad des Trägerpolynoms ist unabhängig von der Anzahl der Kontroll-Punkte
  - B-Splines erlauben „lokale Kontrolle“ über die Form der Kurve bzw. der Fläche
  - Aus der „lokalen Kontrolle“ ergibt sich, dass der Aufwand um eine B-Spline zu zeichnen linear von n abhängig ist (statt quadratisch bei Bezierkurven) ⇒ es ist nicht notwendig Splines mit großer Anzahl von Kontroll-Punkten zu teilen.
- Nachteil: B-Splines sind komplexer als Bézier-Kurven

50) Was sind NURBS (= Non-Uniform Rational B-Splines)? (S. 454)

- Non-uniform: der Abstand zwischen den Kontroll-Punkten ist nicht gleich.

- Rational: die Punkte sind gewichtet (mittels hi)
- Vorteile rationaler (gegenüber non-rationalen) Splines:
    - Ermöglichen eine genaue Repräsentation von quadratischen Kurven (conics). Non-rationale Splines können solche Kurven nur approximieren
    - Sie sind invariant im Bezug auf perspektivische viewing-Transformationen ⇒ allein aus der Transformation der Kontroll-Punkte ergibt sich die korrekte transformierte Ansicht der Kurve.

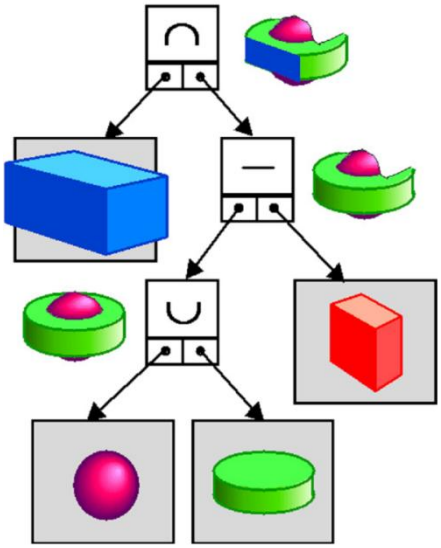


51) Was sind CSG-bäume? (S. 474)

CSG = Constructive Solid Geometry. Beruht auf binären Mengenoperationen (Vereinigung, Schnitt, Differenz) von 3D-Objekten. Jedes Objekt kann mittels einfacher geometrischer Strukturen durch die Mengenoperationen dargestellt werden.

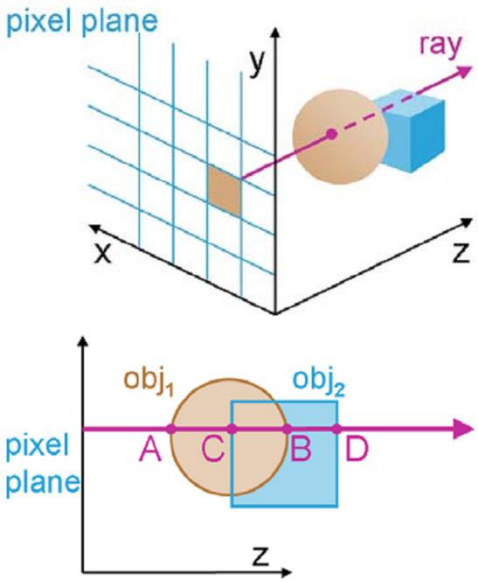
Sie werden so angeordnet, dass sie in einer hierarchischen Datenstruktur angeordnet werden können, die obwohl es sich eigentlich nur um einen kreisfreien Graphen handelt, normalerweise CSG-Baum genannt wird. Als Primitive dienen einfach geometrische Formen wie Kugel, Treträder, Würfel, Zylinder, die mit den Operatoren Vereinigung, Durchschnitt und Differenz verknüpft werden. Da alle Primitive trivialerweise konsistent sind und die Operatoren aus konsistenten Teilen nur konsistente Objekte erzeugen, sind bei CSG alle Objekte immer konsistent (keine Löcher in der Oberfläche, wohldefiniertes Inneres).

Zusätzlich enthält jeder Knoten eines CSG-Baums noch Transformationen (in Form von Matrizen), die angeben, welche Transformationen auf den darunter befindlichen Teilbaum angewendet werden. Primitive (z.B. achsenparalleler Einheitswürfel) erhalten dadurch ein weites Spektrum an Formen (z.B. beliebig im Raum positionierter Quader), und auch jedes komplexere Objekt kann noch verschoben, skaliert, gedreht usw. werden.



Vorgehensweise

Die gebräuchlichste Methode um CSG-Objekte abzubilden ist das Ray-Casting, bei dem das Bild pixelweise berechnet wird. Mittels Ray-Casting werden von einer sogenannten „firing plane“ Strahlen auf die Objekte „abgeschossen“, die modelliert werden sollen. Daraus werden die Schnittpunkte zwischen den einzelnen Flächen bestimmt und die Objekte nach ihrer Entfernung sortiert.



Für jedes Pixel wird in Blickrichtung ein Strahl (Ray) gelegt („auswerfen“ = to cast) und mit allen Objekten der Szene geschnitten. Der vorderste dieser Schnittpunkte gibt an, welches Objekt in diesem Pixel zu sehen ist, und das Pixel erhält dessen Farbe. Bei einem CSG-Baum erfolgt diese Berechnung rekursiv:

- bei Endknoten ist die Berechnung aller Schnittpunkte einfach
- bei Zwischenknoten werden die Schnittpunktlisten der beiden Nachfolger entsprechend dem Operator verknüpft: aus den Listen (A,B) und (C,D) im Beispiel oben entsteht
    - bei Vereinigung die Liste (A,D),

    - bei Durchschnitt die Liste (C,B),
    - bei Differenz die Liste (A,C).
- bei der Baumwurzel wird der erste Punkt der verknüpften Schnittpunktliste ausgewählt.

Um das Volumen eines (zusammengesetzten) Objekts zu bestimmten, wird die „firing plane“ in Teilflächen A_{ij} unterteilt. Das Volumen des Objekts kann anschließend aus der Überlappung des Objekts mit den Teilflächen und dem Abstand der Schnittpunkte der Strahlen mit dem Objekt geschätzt werden.

Datenstruktur zur Speicherung

Binär-Baum, der rekursiv abgearbeitet wird.

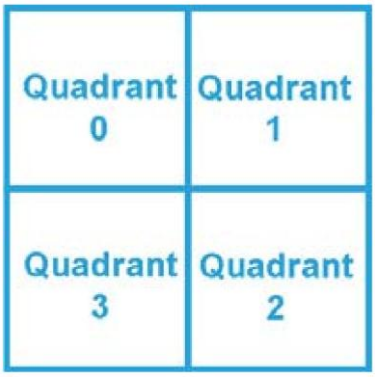
Vorteile und Nachteile

- Vorteile: exakte Repräsentation (eine Kugel ist wirklich eine Kugel!), niedriger Speicherverbrauch und triviale Kombinationen und Transformationen.
- Nachteile: wesentlich aufwändigere Berechnung von Bildern, also das kompliziertere Rendering. Dazu muss man entweder die Datenstruktur in eine BRep-Repräsentation umwandeln und auf herkömmliche Art rendern, oder man verwendet Ray-Casting oder Ray-Tracing zur direkten Bilderstellung.

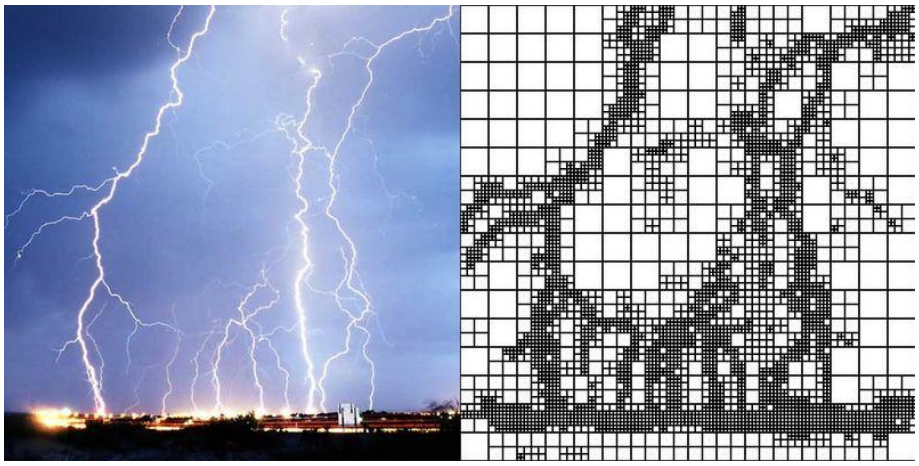
52) Was ist ein Quadtree? (S. 477) Quelle

Ist eine Datenstruktur (in Baum Form), die für das quadtree encoding verwendet wird, zur Repräsentation beliebiger zweidimensionaler Strukturen geeignet ist und zur Organisation zweidimensionaler Daten im Bereich der Computergrafik eingesetzt wird.

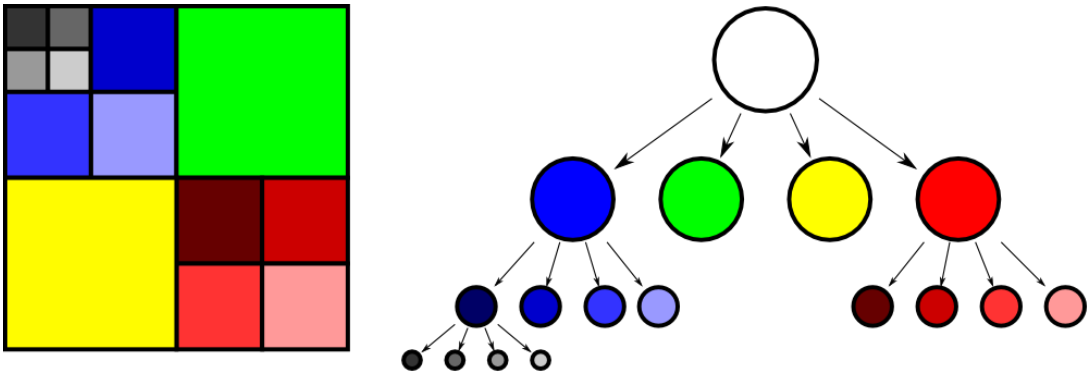
Die Wurzel („Wurzelknoten“) des Baumes repräsentiert dabei eine quadratische Fläche. Der relevante Bereich wird überall dort in vier (gleich große) Viertel (Quadraten) geteilt, wo die Information noch zu kompliziert ist um einfach abgelegt zu werden, andernfalls wird die einfache Information abgelegt (in einem „Blatt“). Jedem Bildbereich entspricht ein „Knoten“ eines Baumes, in dem jeder Knoten (maximal) vier Nachfolger hat („Quadtree“). Durch rekursive Anwendung dieser Zerteilung kann die vom Wurzelknoten repräsentierte Fläche beliebig fein aufgelöst werden. Für dreidimensionale Daten verwendet man gewöhnlich Octrees.



Das Bild unten zeigt einen aus den Konturwerten (links) berechneten Quadtree (rechts). Zu beachten ist, dass bei starken Konturen (hier die Blitze) die Quadrate sehr klein sein müssen, was sich im Speicherverbraucht niederschlägt.



Da ein Blatt unter Umständen eine verhältnismäßig große Fläche abdecken kann, ist die Datenstruktur relativ speichersparend und schnell nach einem Blatt, das einen bestimmten Punkt beinhaltet, zu durchsuchen.



Die Farbe eines Quadranten in der grafischen Darstellung (links) entspricht der Farbe des zugehörigen Blattknotens im Baum (rechts).

Vorteile

- effiziente Speicherung von Daten
- lässt große Traversal- Schritte in den leeren Regionen zu

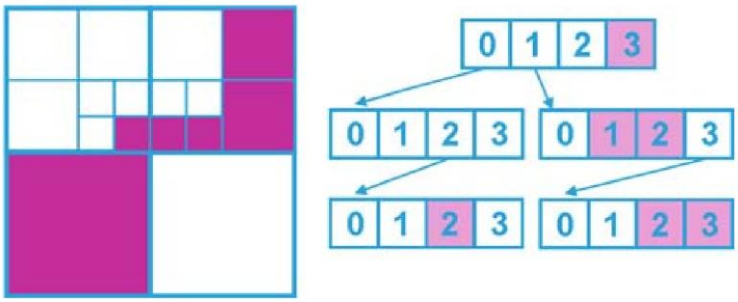
Nachteile

- Relativ komplizierte Traversalalgorithmen
- Benötigt manchmal sehr viele Unterteilungen zur Auflösung verschiedener Objekte.

Beispiel

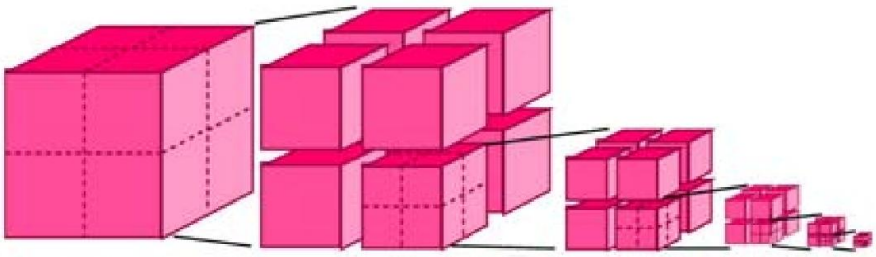
Das untenstehende Beispiel zeigt einen einfachen Quadtree, der eine zweifärbige einfache Graphik

repräsentiert. Der Wurzelknoten entspricht dem ganzen Bild, die Knoten in der zweiten Reihe entsprechen den vier Vierteln des Bildes und die letzten zwei Knoten entsprechen den zwei Bereichen, die am feinsten aufgelöst sind.

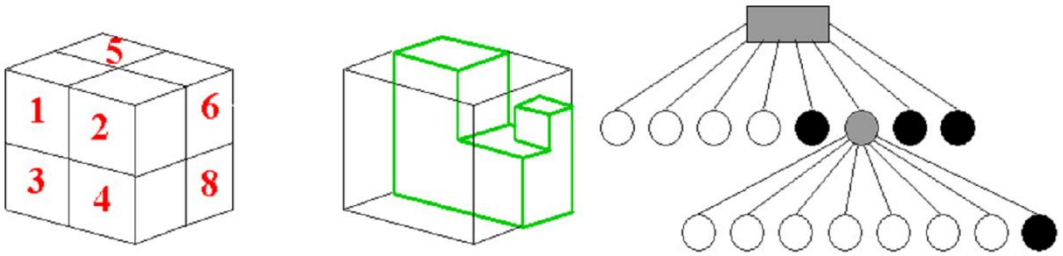


53) Was ist ein Octree? (S. 477)

Ein Octree ist die Erweiterung dieses Konzeptes (Quadtree) auf drei Dimensionen. Ein beliebig geformtes Objekt (oder auch eine ganze Szene) innerhalb eines Würfels wird dadurch repräsentiert, dass einfache Teilwürfel (leer oder ganz innerhalb eines Objektes) durch Endknoten beschrieben werden, und kompliziertere Teilwürfel (alle anderen!) in acht kleinere Teilwürfel (Oktanten) unterteilt werden, auf die wieder dieselben Regeln angewendet werden (rekursiv). Dadurch entsteht ein Baum, in dem jeder Knoten 8 Nachfolger hat („Octree“). Man hört mit der Unterteilung auch dann auf, wenn die Teilwürfel eine bestimmte Mindestgröße unterschreiten (z.B. ein Tausendstel der Gesamtgröße); in diesem Fall erhält der Knoten des Baumes die bestmögliche einfache Information. Das passiert zumindest bei allen schrägen Oberflächen irgendwann, und dann müssen diese Randwürfel entweder als innerhalb oder als außerhalb deklariert werden.



Octrees werden genauso wie Quadtrees rekursiv bearbeitet. Mengenoperationen sind so ganz einfach, dafür sind geometrische Transformationen (von Ausnahmen abgesehen) sehr aufwändig, weil der Octree komplett neu generiert werden muss. Das Rendering von Octrees ist dagegen bei Verwendung eines überschreibbaren Speichers einfach.



Vorteile

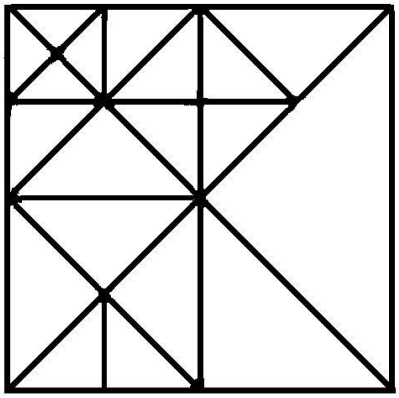
- Mengenoperationen sind einfach durchzuführen
- beliebige Formen repräsentieren
- Schnelles Rendering
- Räumliches Suchen ist möglich
- schnell untersuchbar, was sich an einer bestimmten räumlichen Position befindet

Nachteil

- Transformationen sind (bis auf einige wenige) schwierig durchzuführen
- Niedrige Bildqualität da ungenaue Repräsentation
- Hohe Memory-Kosten/ Speicherbedarf

54) Was ist ein Bintree? (S. ???) Quelle

Ein Bintree wird als Datenstruktur in der Computergrafik eingesetzt und funktioniert ähnlich einem Quadtree. Er ist ein Binärbaum, also hat jeder Knoten genau zwei Kinder. Bei der Landschaftsvisualisierung wird ein Bintree folgendermaßen eingesetzt: Man weist jedem Knoten des Baums genau ein Dreieck zu. Die erste Stufe (zwei Knoten) besteht dabei aus zwei Dreiecken, welche zusammen das ganze Terrain bedecken. Jedes Dreieck wird dann in zwei weitere geteilt. Diese Unterteilungen ermöglichen verschiedene Auflösungsstufen für die Landschaft, wodurch sich große Einsparungen beim Speicherbedarf und der Rechenzeit ergeben. Da die Dreiecke gemeinsame Kanten haben, müssen beim Unterteilen Auswirkungen auf Nachbarknoten immer berücksichtigt werden.



Kapitel 9 – Visible Surface Detection Methods (S. 528)

55) Welche zwei Kategorien von visible surface detection methods kennen Sie (S. 529)

Object- space Methoden

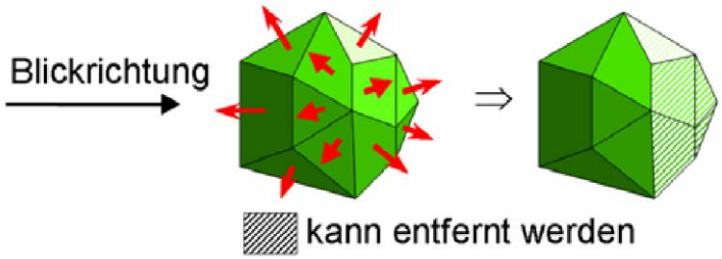
Vergleichen Objekte und Teile von Objekten miteinander im Rahmen der Szenen-Definition als Ganzes um festzustellen welche Flächen (als Ganzes) als sichtbar eingestuft werden sollen.

Image- space Methoden

Hierbei wird die Entscheidung über die Sichtbarkeit pixel für pixel getroffen für jeden Bildpunkt auf der Projektions-Ebene.

56) Was ist „back-face detection“ bzw. backface culling und auf was für Objekte darf man es anwenden? (S. 530)

Back-face detection ist eine schnelle und simple object-space Methode. Backface Culling ist kein vollständiges Sichtbarkeitsverfahren. Es werden lediglich alle Polygone, deren Oberflächennormale vom Betrachter weg zeigen und die daher ganz sicher nicht sichtbar sein können, eliminiert, um den Aufwand nachfolgender Arbeitsschritte zu reduzieren. Dadurch werden im Durchschnitt 50% der Polygone entfernt.



Man berechnet den Normalvektor N für die Polygonebene. Ist Vview der Vektor in Blickrichtung von unserer Kameraposition ausgehend, dann ist das Polygon ein back-face, wenn Vview * N > 0 ist und somit unsichtbar. Wenn also der Vektor N in die gleiche Richtung wie unser Sichtvektor zeigt, dann ist die Fläche ein Backface, daher der Normalvektor der Fläche muss in unsere Richtung zeigen, damit wir eine Fläche sehen können.

Darf nur auf konvexe Polyhedra angewendet werden (bei konkaven ist entweder eine Umwandlung notwendig, oder zusätzliche Methoden)

57) Dot Product vs. Cross Product beim Backface Culling

Das Dot Product, auf Deutsch Skalarprodukt (auch inneres Produkt) genannt, wird oft verwendet um Winkel zwischen zwei Vektoren und die Länge von Vektoren zu bestimmen. Man berechnet es durch komponentenweises Multiplizieren der Koordinaten der Vektoren und anschließendes Aufsummieren.

$$\begin{pmatrix}1\\2\\3\end{pmatrix} \cdot \begin{pmatrix}-7\\8\\9\end{pmatrix} = 1 \cdot (-7) + 2 \cdot 8 + 3 \cdot 9 = 36$$

Cross Product, auf Deutsch Kreuzprodukt (auch äußeres Produkt) genannt, entspricht einem Vektor, der senkrecht auf der von den beiden Vektoren aufgespannten Ebene steht. Die Länge dieses Vektors entspricht dem Betrage nach der Fläche des Parallelogramms mit den Seiten $\vec{a}$ und $\vec{b}$.

$$\begin{pmatrix}1\\2\\3\end{pmatrix} \times \begin{pmatrix}-7\\8\\9\end{pmatrix} = \begin{pmatrix}2\cdot 9 - 3\cdot 8\\3\cdot(-7) - 1\cdot 9\\1\cdot 8 - 2\cdot(-7)\end{pmatrix} = \begin{pmatrix}-6\\-30\\22\end{pmatrix}$$

Info am Rande: Kreuz- und Skalarprodukt sind über das Spatprodukt miteinander verknüpft.

58) Was sind die wesentlichen Schritte beim "depth-buffer" (z-buffer)- Algorithmus? Was ist ein z-Puffer und wozu dient er? (S. 532)

Der z-Buffer-Algorithmus ist eine image-space Methode, die mittels einer Hardware-Implementierung umgesetzt wird. Es gibt keine Sortierung. Beim z-Buffer-Algorithmus werden die z-Werte für jeden Pixel einer Oberfläche mit den z-Werten der anderen Oberflächen verglichen, und dadurch wird festgestellt, welche Oberfläche der viewing-Plane am nächsten ist und welche Pixel dargestellt werden.

Wenn eine Oberfläche näher als alle zuvor berechneten Oberflächen ist, so werden für diese Oberfläche die Farbe und die Tiefe gespeichert. Also je größer der z-Wert, umso näher ist die Oberfläche der viewing-plane, da wir mit unserer Kamera in negative z-Richtung schauen.

Depth-Buffer Algorithmen rechnen mit normalisierten Koordinaten, somit kann der Wert für die Tiefe sich nur zwischen 0.0 (view plane) und 1.0 befinden. Der depth-Buffer wird dazu verwendet, die Tiefeninformation für jede (x,y) Position einer Oberfläche zu speichern und der Frame-Buffer speichert die Farbinformation für jeden Pixel einer Oberfläche. Die Größe der Buffer hängt von der Bildschirmauflösung ab.

Vorgehensweise

Für jeden Bildpunkt merkt man sich zusätzlich zur Farbinformation in einem eigenen Speicher die Position des dargestellten Objektes. Da als Blickrichtung normalerweise die z-Richtung verwendet wird, entsprechen x- und y-Werte dieser Position denen der Abbildungsebene und es braucht nur der z-Wert gespeichert zu werden. Man braucht also zusätzlich zum Bildpuffer (frame buffer) einen weiteren Speicherbereich, der für jedes Pixel einen Koordinatenwert (z-Wert) aufnehmen kann, diesen Speicher nennt man z-Puffer oder Tiefenpuffer (z-buffer, depth buffer). Nun kann man alle Objekte in beliebiger Reihenfolge zeichnen. Die z-Werte des nächsten zu zeichnenden Objektes (meist ein Polygon) werden berechnet und mit den z-Werten der Pixel verglichen, in die das Objekt gezeichnet werden soll. Ist der neue z-Wert näher zum Betrachter (also normalerweise größer), dann wird das Objekt an dieser Stelle über den alten Bildwert darüber gezeichnet und der z-Wert im z-Puffer ebenfalls ersetzt. Andernfalls ist das neue Objekt verdeckt und wird an dieser Stelle nicht gezeichnet.

Vorgehensweise mittels Scan-Fill

- Man startet beim linken Pixel und geht entlang der Scanlinie nach rechts vor.
- Zuerst berechnet man den z-Wert der linken Polygonkante, dort wo sie die Scanlinie schneidet, daraufhin berechnet man den z-Wert für jeden Pixel entlang der Scanlinie.
- Man kann den depth-Buffer-Algorithmus so implementieren, dass er beim obersten Vertex des Polygons startet und dann die x-Koordinatenwerte abwärts der linken Polygonkante rekursiv berechnet.

- Der x-Wert für die Startposition einer jeden Scanlinie kann durch den vorherigen Startpunkt berechnet werden. x' = x-1/m (m ist die Steigung der Kante)

59) Wo stößt Z-Buffer an seine Grenzen?

Der Z-Buffer kann Transparenz nicht auswerten. Eine transparente Geometrie wie z.B. eine Glasscheibe verursacht Probleme, da dahinterliegende Objekte als „nichts" sichtbar angenommen werden, da sie vom z-Buffer aufgrund kleineren z-Wertes als hinter der transparenten Geometrie liegend angesehen werden.

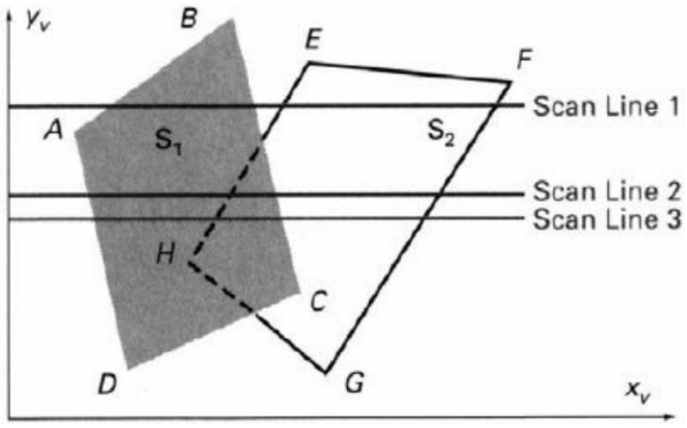
Korrekt wäre es, zu schauen wo es Transparenz gibt und welche Objekte die hinter dem transparenten Objekt liegen durch die Transparenz doch gesehen werden können. Problem kann man im Image Space lösen z.B. durch Raycasting.

Für interessierte OpenGL Coder gibt es unter LINK ein paar nette Workarounds.

60) Wie funktioniert die Scan-Line Methode? (S. 535)

Ist eine image-space Methode, die den scan-line Füll-Algorithmus für Polygone erweitert. Er braucht eine Tabelle mit Kanten (y-sortiert) und mit Polygonen (die auf die Kanten-Tabelle verweist) sowie eine active-edge Liste

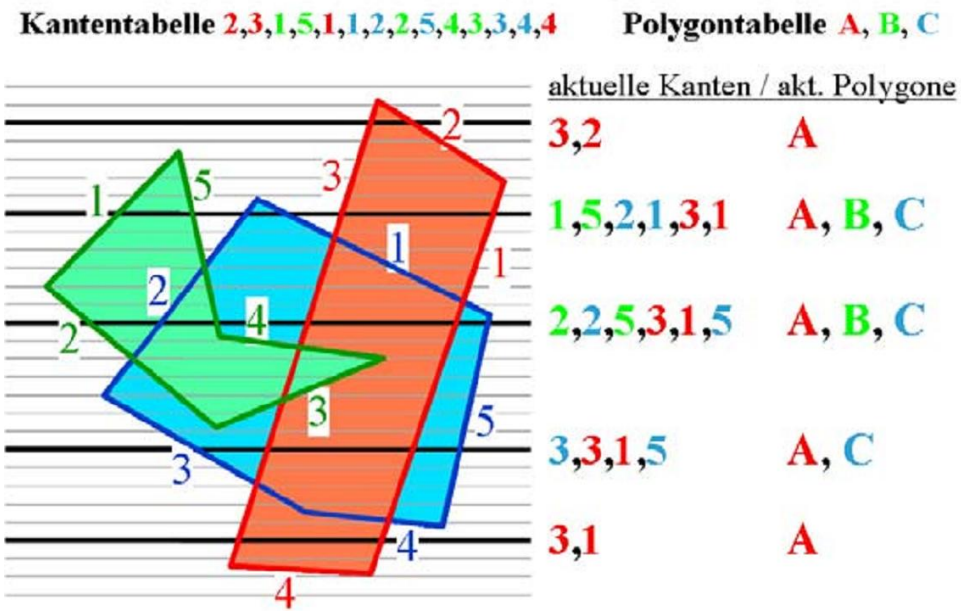
Berechnet die z-Werte entlang der Scanlinien für jeden Pixel, um festzustellen, welcher Pixel dargestellt wird, und zwar dann, wenn eine Scanlinie mehrere Flächen schneidet.



Entlang der Scanlinie werden Tiefeninformationen berechnet, um festzustellen, welche Oberfläche der View-Plane am nächsten ist. Hat man die sichtbare Fläche berechnet, dann wird für jedes Pixel die Oberflächenfarbe im Framebuffer gespeichert.

Man definiert eine Flag, die entweder „on" oder „off" sein kann, um festzustellen ob ein bestimmter Punkt der Scanlinie innerhalb oder außerhalb des Polygons liegt. Bei dieser Methode berechnet man die Pixel entlang der Scanlinie von links nach rechts, beim linken Schnittpunkt der Scanlinie mit der Kante eines konvexen Polygons, wird die Flag auf „on" gesetzt, beim rechten Schnittpunkt entlang der Linie auf „off". Bei konkaven Polygonen werden die Scanlinienschnittpunkte von links nach rechts sortiert und die Flag wird zwischen jedem Schnittpunktpaar auf „on" gesetzt. An den Stellen, wo die Flächen nicht überlappen, ist keine Tiefeninformation notwendig.

Überlappen sich zwei Flächen, wird zuerst einmal die Flag für den Schnittpunkt mit der ersten Oberfläche auf „on“ gesetzt, sobald die Scanlinie auch die zweite Fläche schneidet wird auch für diese Fläche die Flag auf „on“ gesetzt. Die Tiefeninformation wird nur für den Bereich berechnet, indem beide Flags auf „on“ sind.



Dieser Algorithmus funktioniert nur für Ebenen, die einander nicht durchschneiden oder sonstwie zyklisch überlappen. Tritt dieser Fall dennoch ein, so muss man die Polygone (im Buch mittels einer strichlierten Linie) teilen.

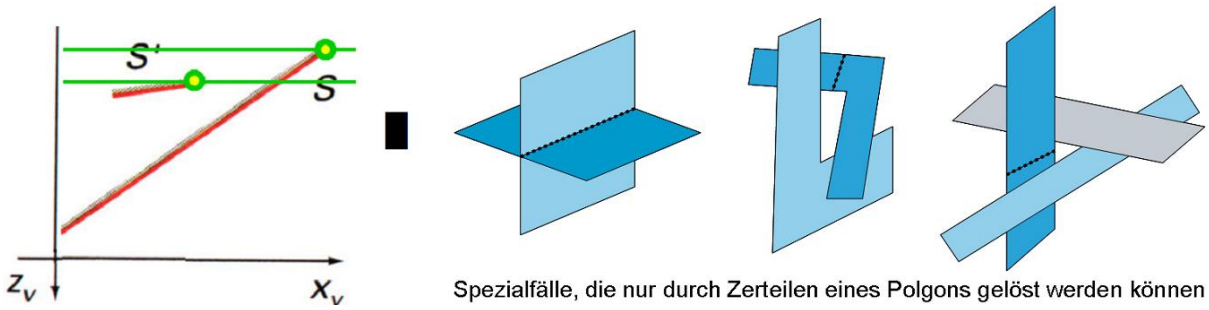
61) Wie funktioniert die Depth-Sorting Methode („Painter“-Methode)? (S. 537)

Ist eine Mischung aus einer image-space Methode (fine tuning) und einer object-space Methode (grobe Sortierung).

Die Oberflächen werden in absteigender Reihenfolge bezüglich der Tiefe sortiert. Die Oberflächen werden scan-konvertiert, wobei man mit der Oberfläche mit der größten Tiefe startet. Die Sortieroperationen werden sowohl im image- und im objekt-space ausgeführt, aber die Scankonvertierung der Polygonsoberflächen findet im image-space statt.

Sind die Objekte nebeneinander (aus Betrachtersicht), oder überdecken sich komplett, so muss kein „fine tuning“ mehr gemacht werden. Falls alle Tests fehlschlagen, so müssen die Objekte umgeordnet werden.

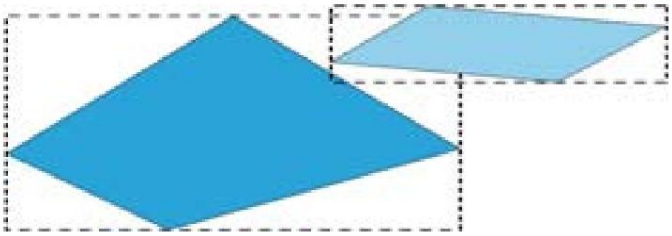
Der Hauptaufwand liegt hier beim Sortieren, das so erfolgen muss, dass kein Polygon ein anderes (partiell) verdeckt, welches in der Liste erst nachher kommt (also weiter „vorne“ liegt). Dazu wird zuerst schnell eine Grobsortierung durchgeführt und anschließend überprüft, ob alles stimmt, gegebenenfalls umsortiert.



Spezialfälle, die nur durch Zerteilen eines Polgons gelöst werden können

Vorgehensweise

1. Grobsortierung: ordne die Polygone nach ihrem kleinsten z-Wert (größte Tiefe)
2. Vergleiche jedes Polygon S mit jedem (!) anderen Polygon S': (* Annahme S liegt hinter S' *). (* jetzt beginnt eine Folge von Tests in aufsteigender Komplexität, bis Sortierung als korrekt erkannt ist *)
    a. der größte z-Wert von S ist kleiner als der kleinste z-Wert von S' → Sortierung korrekt
    b. die x- oder y-Intervalle der beiden Polygone schneiden sich nicht → Sortierung korrekt
    c. alle Eckpunkte von S liegen hinter Ebene von S' → Sortierung korrekt
    d. alle Eckpunkte von S' liegen vor der Ebene von S → Sortierung korrekt
    e. die Projektionen von S und S' auf die xy-Ebene schneiden sich nicht (siehe Bild unten) → Sortierung korrekt
    f. Sortierung wahrscheinlich falsch (siehe etwa das Beispiel, wo S' von S verdeckt wird) → Vertauschen und neu kontrollieren, sollte die Sortierung wieder falsch sein, dann liegt ein Spezialfall vor (siehe Abbildung oben) und muss durch Zerteilen eines Polygons gelöst werden.



Problem

Falls die Flächen sich zyklisch überlappen, müssen sie aufgetrennt werden (siehe zyklische Überlappung bei der scan-line Methode).

Verwendung des Buffers

Die Farbwerte für die am weitesten entfernte Fläche werden im refresh-Buffer eingetragen. Als nächstes kommt die nächste Oberfläche an die Reihe, und beim Speichern der Farbwerte für diese Fläche werden die alten Farbwerte im Framebuffer einfach überschrieben.

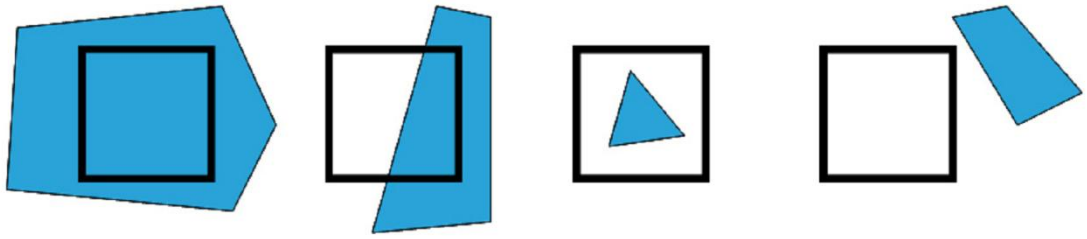
Diese Methode wird auch oft als „Painter's Algorithm" bezeichnet. Denn wenn ein Maler ein Bild mit z.B. Ölfarben malt, beginnt auch er zuerst mit dem Hintergrund und arbeitet sich dann von hinten nach vorne schrittweise vor.

62) Wie funktioniert die Area-Subdivision Methode? (S. 541)

Ist eine image-space Methode, die die viewing area immer weiter unterteilt bis die Sub-Fläche so klein ist, dass das Lösen des Sichtbarkeitsproblems leicht funktioniert. Die Vorgehensweise ist ähnlich, wie der Aufbau eines Quadtrees. Einfache Fälle werden in grober Auflösung gelöst und kompliziertere Fälle durch Unterteilung der Fläche in 4 Viertel vereinfacht. Rekursive Anwendung bis maximal zur Bildauflösung garantiert eine pixelgenaue Lösung des Sichtbarkeitsproblems.

Basis des Verfahrens ist eine Funktion, die schnell feststellt, in welcher Lage sich die Projektion eines Polygons bezüglich eines (quadratischen) Bildfensters befindet. Es gibt 4 mögliche Beziehungen, die Flächen miteinander haben können.

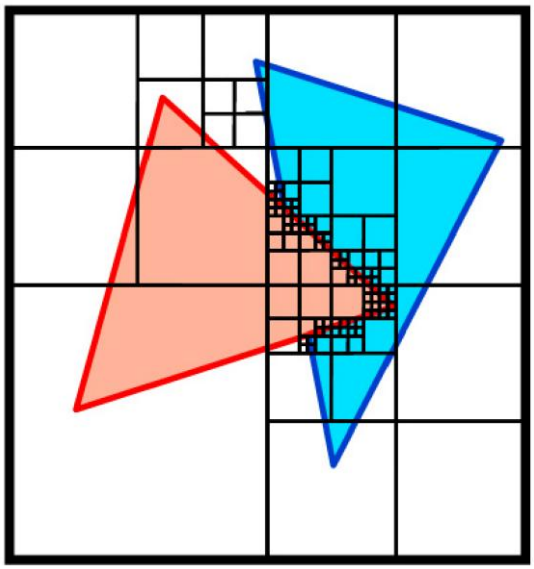
1. Surrounding Surface: Eine Fläche, die die andere komplett einschließt (Bild unten links)
2. Overlapping Surface: Eine Fläche, die eine andere überlappt. bzw. ein Teil einer Fläche ist innerhalb und ein Teil ist außerhalb einer Fläche (das zweite Bild unten).
3. Inside Surface: Eine Fläche, die komplett innerhalb einer anderen Fläche liegt (das dritte Bild unten).
4. Outside Surface: Ein Fläche die komplett außerhalb einer anderen Fläche liegt (Bild unten rechts).



Die Tests für die Oberflächensichtbarkeit können anhand dieser vier Klassifikationen durchgeführt werden. Man kann die Tests beenden, wenn eine der drei folgenden Konditionen eintritt:

1. Eine Fläche hat weder eine innere, überlappende oder einschließende Fläche.
2. Eine Fläche hat nur eine innere, überlappende bzw. einschließende Fläche
3. Eine Fläche hat eine umschließende Fläche, die alle anderen Flächen innerhalb ihres Randes verdunkelt.

Ist man auf Pixel-Ebene angelangt und keine der obigen 3 Konditionen um den Test zu beenden ist zutreffend (sprich alle 3 Tests sind fehlgeschlagen), so muss man die Flächen sortieren und die Intensität der nähersten Fläche auswählen. Es wird also das Fenster in 4 Viertel unterteilt, und diese werden rekursiv bearbeitet. Man beachte, dass Polygone, die bereits außerhalb lagen, auch außerhalb aller Teilfenster liegen, und dass Polygone, die das Fenster überdeckt haben, auch alle Teilfenster überdecken. Wenn ein Teilfenster nur noch die Größe eines Pixels hat, so wählt man dort das Polygon, das am weitesten vorne liegt. Wie man am Beispiel sieht, passiert das genau an den Kanten, an denen die Sichtbarkeit wechselt.



63) Wie funktioniert die Octree Methode? (S. 543)

Wenn die Szene statt in Polygonen als Octree repräsentiert ist, dann weiß die Datenstruktur bereits für jede Blickrichtung, was vorne und was hinten ist. Rekursiv kann man in jedem Würfel immer zuerst den entferntesten Teilwürfel rendern, dann die 3 nächstnäheren, dann die nächsten drei und schließlich den vordersten. Eine mögliche Reihenfolge für eine frontale Blickrichtung ist in dem folgenden Beispiel zu sehen:

Beruht auf dem rekursiven Abarbeiten von Informationen eines Objekts, die in einem Octree gespeichert sind. Die Reihenfolge der Abarbeitung der einzelnen Sub-Oktanten beruht auf der Blickrichtung. Gegebenenfalls muss der Octree in die richtige Position transformiert werden.

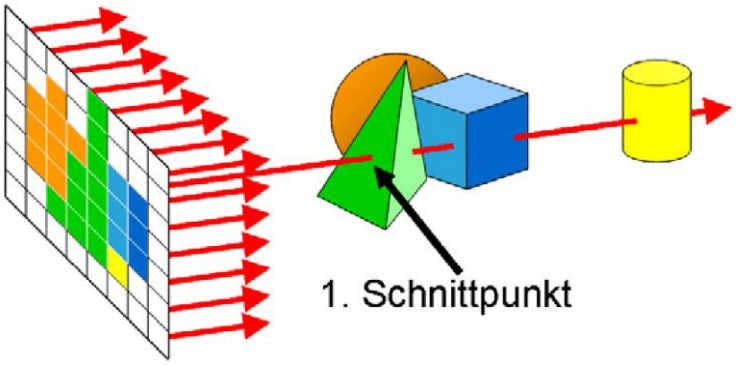
Ein Wert wird in den Frame-Buffer eingetragen, wenn noch kein Wert zuvor an dieser Position eingetragen worden ist ⇒ jedes Pixel (x, y) wird nur einmal eingetragen. Ist ein Oktant total ausgefüllt, so können alle dahinterliegenden Oktanten vernachlässigt werden, weil sie nicht sichtbar sind.

Alternativ kann man natürlich auch von vorne nach hinten zeichnen. Dann muss man sich alle Bereiche merken, in denen bereits etwas eingetragen ist, und zeichnet nur Dinge, die auch wirklich sichtbar bleiben. Der Vorteil gegenüber anderen Datenstrukturen bleibt das implizite Wissen darüber, was weiter vorne oder weiter hinten ist.

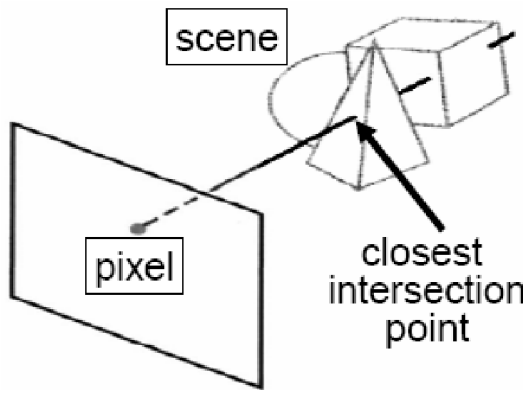
64) Wie funktioniert die Ray-casting Methode? (S. 487)

Ist eine spezielle Form der Ray-Tracing Methode (wobei Ray-Casting eine vereinfachte Version von Ray Tracing ist, mit dem noch viele weitere optische Effekte simuliert werden können) und ein Sichtbarkeitsverfahren, bei dem ein Strahl von der view plane pixelweise auf die davor liegenden

Objekte „abgeschossen“ wird. In umgekehrter Richtung gelangt auf dieser Linie das Licht aus der Szene auf die Bildebene, also zum Betrachter. Schneidet man diesen Blickstrahl mit allen Objekten/Polygonen der Szene, so erhält man eine Menge von Schnittpunkten, aus denen man den auswählt, der zum Betrachter am nächsten liegt. Die Farbe der Oberfläche an dieser Stelle bestimmt die Farbe des Pixels, durch das man den Strahl gelegt hat. Macht man das für alle Pixel, so erhält man für jeden Punkt die Farbe des dort vordersten Objektes, also des sichtbaren Objektes.



Der Tiefenwert für den z-Buffer wird nur für die Objektpunkte berechnet, in denen eine Objekt-Fläche vom „Pixelstrahl“ geschnitten wird. So gesehen ist diese Methode auch mit der z-Buffer-Methode „verwandt“.



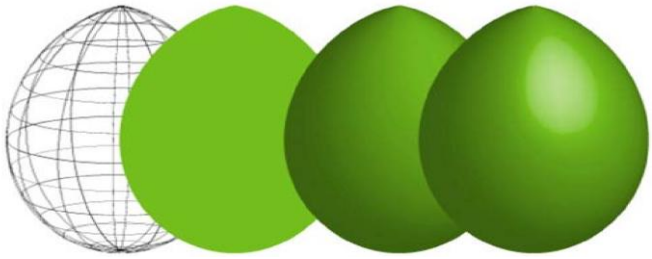
Mit Ray-Casting kann man außer Polygonen auch alle anderen Oberflächen (z.B. Freiformflächen, gebogenen Flächen, speziell Kugeln) leicht rendern, für die der Schnitt mit einer Geraden berechenbar ist.

Wie wir später sehen werden benötigt man normalerweise auch die Oberflächennormale an der Auftreffstelle, um eine brauchbare Schattierung zu erhalten. Andererseits ist das Verfahren aber sehr aufwändig, da man für jedes Pixel (das sind für einen Bildschirm etwa 1 Million) einen Schnitt mit jedem Objekt durchführen muss (und das können durchaus auch viele tausend sein). Effiziente Implementierung des Schnittpunkt-Tests und weitere Optimierungen sind daher notwendig.

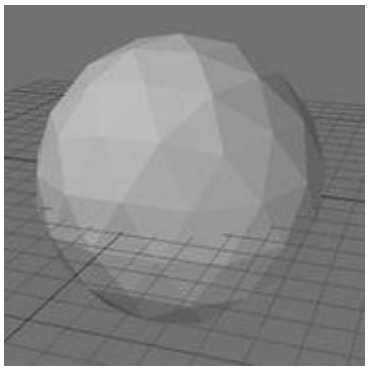
Kapitel 10 – Illumination Models and Surface-Rendering Methods (S. 556)

65) Was versteht man unter Schattierung? (S. ???) Quelle

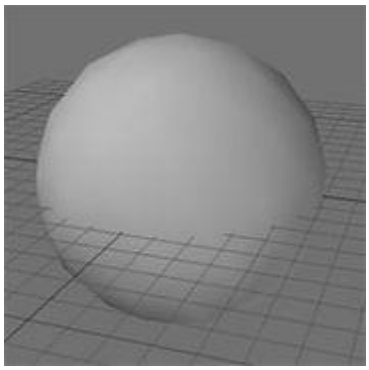
In Abhängigkeit des Blickwinkels oder Lichteinfallswinkels werden Oberflächen heller oder dunkler gefärbt.



Shading bezeichnet in der 3D-Computergrafik im allgemeinen Sinne die Simulation der Oberfläche eines Objekts. Dies wird unter anderem durch Beleuchtungsmodelle ermöglicht.



Kugel mit Flat Shading



Kugel mit Gouraud Shading

Im Spezialfall von Polygongeometrie bezeichnet Shading auch das Interpolationsverfahren, mit dem der Normalenvektor auf beliebigen Punkten der Oberfläche berechnet wird. Bei den 3 nun kommenden Verfahren haben jeweils alle Lichtquellen Einfluss auf das gesamte Polygon. Das Beleuchten eines Polygonteils, etwa mit einem Spotlight, ist nicht möglich!

Flat Shading

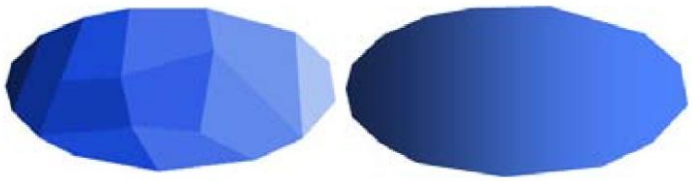
Beim Schattieren eines Polygons hat klarerweise jeder Punkt die gleichen Oberflächeneigenschaften, vor allem auch den gleichen Normalvektor. Beim einfachen Ausfüllen jedes Polygons mit einer Farbe werden die Grenzen zwischen den Polygonen deutlich störend erkennbar. Der sogenannte Mach-Band-Effekt, das ist ein kantenverstärkender Mechanismus des Auges, macht das Problem dabei noch ärger als es ist. Dieser Effekt lässt uns an Kanten die dunklere Seite dunkler wahrnehmen als sie ist, und die hellere Seite heller als sie ist. Die einfachste Lösung dieses Problems ist das Interpolieren der Schattierung zwischen den Polygonen. Dazu sind zwei Verfahren üblich: Gouraud-Schattierung und Phong-Schattierung.

Flat Shading führt keine Interpolation durch, sondern greift auf die Farbe des ersten Vertex des gerade zu zeichnenden Polygons zurück. Diese Farbe wird für alle Punkte des Polygons verwendet.

Gouraud Shading

Die Gouraud-Schattierung interpoliert die berechneten Helligkeitswerte über die Polygonflächen. Dazu werden an den Eckpunkten (Vertices) der Polygone Helligkeitswerte berechnet und von diesen aus wird durch lineare Interpolation jedes Polygon gefüllt.

Dennoch verbleiben Fehlerquellen. So wird die Silhouette natürlich nicht verändert, dadurch verbleiben störende Polygonkanten sichtbar (siehe Bild unten). Weiters kommt es im Bereich von Glanzpunkten zu zufälligen Interpolationsergebnissen, je nachdem ob es zufällig eine Normale gibt, die genau einen Glanzpunkt erzeugt oder nicht. Dies stört besonders bei bewegten Objekten.



Phong Shading

Als Alternative zur Gouraud-Schattierung erzeugt die Phong-Schattierung (nicht mit dem Phong-Beleuchtungsmodell verwechseln!!!) viel konsistentere Glanzeffekte. Ebenso wie bei der Gouraud-Schattierung werden in den Polygoneckpunkten (Vertices) die Normalen berechnet, aber nun werden diese Normalen entlang der Polygonkanten interpoliert, und anschließend entlang der Scanlines. Dann wird für jedes Pixel extra die Helligkeit nach einen Beleuchtungsmodell berechnet. Dies verursacht zwar einen höheren Aufwand, führt aber auch zu schöneren Ergebnissen.

66) Was verstehen Sie unter einem Illumination model (Beleuchtungsmodell)? (S. 557)

Wird verwendet, um die beleuchtete Position auf der Oberfläche eines Objektes zu bestimmen.

Eine Surface-Rendering-Methode verwendet die Farbberechnung des Illumination Models, um die Farbe für jeden Pixel in einer Szene zu berechnen. Man kann entweder das Illumination Model auf jede Position anwenden, oder man vollendet das Surface-Rendering, indem man die Oberflächenfarbe mittels eines kleinen Sets von Illumination-Model-Kalkulationen interpoliert. Scanlinienalgorithmen zum Beispiel verwenden hauptsächlich Interpolierungsschemata, Raytracing-Algorithmen hingegen berechnen das Illumination Model für jeden Pixel.

Die Oberflächenbeleuchtungseffekte beinhalten Reflexion, Transparenz, Muster und Schatten.

Sind die Parameter für die optischen Eigenschaften der Oberfläche, die relative Position der Fläche in einer Szene, die Farbe und Position der Lichtquelle, sowie die Charakteristika der Lichtquelle und die Orientierung und Position der Viewing Plane gegeben, berechnet das Illumination Model die Lichtintensität, die von einer bestimmten Oberflächenposition in eine bestimmte Richtung projiziert wird.

Genauere Systeme, wie z.B. der Radiosity Algorithmus, berechnen Lichtintensitäten, indem sie die Ausbreitung der Strahlen zwischen der Lichtquelle und den verschiedenen Oberflächen in einer Szene in Betracht ziehen.

67) Welche Arten von Lichtquellen kennen Sie? (S. 558)

Punktlichtquelle

Das einfachste Modell, das Licht erzeugen kann. Das Licht hat nur eine Farbe. Man verwendet die Position dieser Lichtquelle im Illumination Model um festzustellen, welche Objekte einer Szene von dieser Lichtquelle beleuchtet werden und um die Leuchtrichtung bezüglich einer bestimmten Objektoberfläche zu berechnen. Die Stärke der Lichtquelle nimmt mit der Entfernung ab.

Unendlich weit entfernte Lichtquelle

Wie z.B. die Sonne. Kann ebenfalls als Punktlichtquelle verstanden werden, nur mit dem Unterschied, dass solch eine Lichtquelle die Szene nur von einer bestimmten Richtung aus beleuchtet.

Richtungsorientierte Lichtquelle (spot light)

Haben bestimmte Position, Richtung, sowie einen Abstrahlwinkel und einen Abschwächungswinkel, die den Kegel des Lichts modellieren Reflektierende Lichtquellen: erzeugen selber kein Licht, sondern

reflektieren es nur. So ist es möglich, dass auch Objekte, die nicht direkt einer Lichtquelle ausgesetzt sind, sichtbar sind.

68) Welche grundsätzlichen Illumination Models kennen Sie? (S. 563 ff.)

Alle hier aufgezählten Modelle sind empirische Modelle. Faktoren, die zur Berechnung der Farbe in einem Punkt eingehen: Materialeigenschaften des Objekts, Augenpunkt des Betrachters, Normalenvektor des Objekts, Positionierung der Lichtquellen und die Art der Lichtquellen

Umgebungslicht / Hintergrundlicht (ambient light) Ia

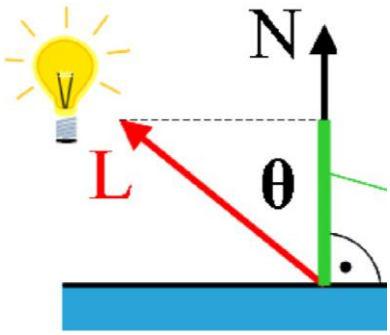
Da jedes Objekt einen Teil des auf ihn auftreffenden Lichtes auch wieder abstrahlt, ist es auch dort nicht ganz dunkel, wo keine Lichtquelle direkt hinleuchten kann. Dieses überall vorhandene Basislicht wird ambient Licht genannt, auch Hintergrundlicht. Einfache Beleuchtungsmodelle inkludieren dazu einen konstanten Wert Ia zu jeder Beleuchtungsberechnung.

- Keine Position, keine Richtung
- Grundhelligkeit eines Objekts
- Intensität: Ia
- Reflexions-Koeffizient: ka (0 <= ka <= 1)
- Extrem einfacher Fake für „Radiosity“

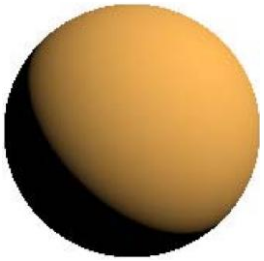
Diffuse Reflexion

Das Lambert’sches Gesetz besagt, dass je flacher Licht auf eine Oberfläche auffällt, desto dünkler erscheint diese Oberfläche. Erst durch diesen Effekt erhalten wir den Eindruck einer räumlichen Form.

Sei Il die Helligkeit der relevanten Lichtquelle, und sei kd der diffuse Reflexionskoeffizient der beleuchteten Oberfläche, der also angibt, wie viel Prozent des einfallenden Lichtes in alle Richtungen gleichmäßig wieder abgestrahlt wird. Natürlich gilt $0 \leq kd \leq 1$. Weiters sei θ der Winkel zwischen der Oberflächennormale und der Richtung zur Lichtquelle, also der Lichteinfallsrichtung. Dann gilt für die resultierende Intensität I an der Oberflächenstelle:



$$I = k_d \cdot I_l \cdot cos\theta = k_d \cdot I_l \cdot N \cdot L \quad [N \cdot L \; ist \; skalares \; Produkt \; (oben \; grün \; dargestellt)]$$

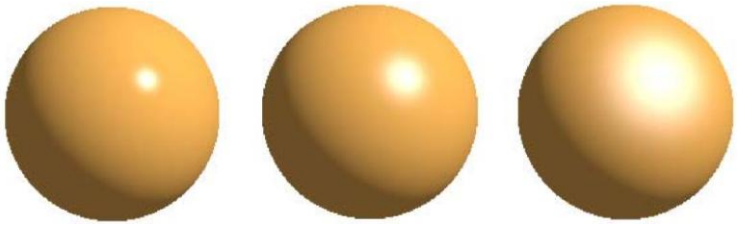


Wenn man nun auch noch das ambiente Licht hinzufügt, dann erhält man schon eine recht schöne Kugel (obere Kugel = nur diffuse Beleuchtung, untere Kugel = diffus + ambient).

- Je flacher der Lichteinfall auf eine Fläche ⇒ desto dünkler erscheint die Fläche (Lambert'sche Regel)
- Konstant für eine Fläche, aber abhängig davon wie die Fläche modelliert werden soll (Material-Eigenschaft)
- Reflexions-Koeffizient: kd

Spiegelnde Reflexion (specular reflection) bzw. Glanzpunkte (specular highlights) und das Phong Model

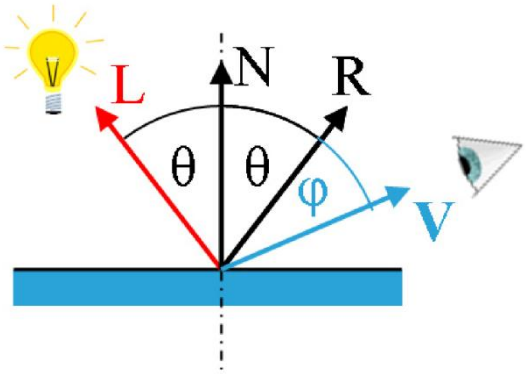
Fast jede Oberfläche ist auch etwas spiegelnd. Wenn man diesen Aspekt nicht mit modelliert, dann wirken alle Materialien gleich stumpf. Da die exakte spiegelnde Reflexion äußerst kompliziert zu berechnen ist, behilft man sich mit einer einfachen Funktion, die einen ähnlichen Verlauf hat wie das Highlight: cos^n. Mit dem freien Parameter n lässt sich dabei die „Poliertheit" der Oberfläche steuern: je größer n ist, desto kleiner wird der Glanzpunkt und desto glatter wirkt die Oberfläche (Bild unten linke Kugel), je kleiner das n ist, desto matter wirkt die Oberfläche (Bild unten rechte Kugel). Um diesen Effekt im richtigen Ausmaß zur Beleuchtung hinzufügen zu können, wird noch ein weiterer Faktor eingeführt, der spiegelnde Reflexionskoeffizient ks.



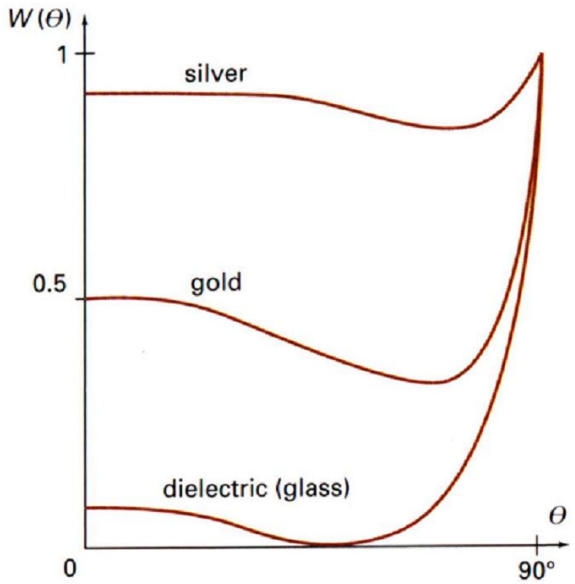
Der Glanz berechnet sich dann nach diesem sogenannten Phong-Beleuchtungsmodell so:

$$I_{l,spec} = k_s \cdot I_l \cdot cos^n\varphi = k_s \cdot I_l \cdot (R \cdot V)^n$$

Der Winkel φ ist dabei der Unterschied zwischen dem exakten Reflexionsstrahl und der Richtung zum Auge.



Etwas näher an der Wahrheit ist die Verwendung der Fresnel'schen Reflexionsgesetze, die beschreiben, dass der Spiegelungsgrad auch vom Lichteinfallswinkel abhängt, dass also der Koeffizient ks eigentlich eine Funktion W(θ) der Lichteinfallsrichtung ist. Für die meisten Materialien ist dieser Wert aber fast konstant. Daher wird auf diesen Aufwand verzichtet, wenn man nicht gerade ein Material darstellen will, bei dem der Effekt auffällt. Das Bild unten zeigt die Abhängigkeit dieser Funktion W(θ) vom Winkel zwischen Lichteinfall und Normale auf die Oberfläche für drei verschiedene Materialien.



- = Phänomen, dass in einem bestimmten Toleranzwinkel um die spiegelnden Reflexionswinkel das Licht (fast) total reflektiert wird (direkt in das Auge des Betrachters)
- wie groß dieser Winkel ist, ist wiederum eine Materialeigenschaft
- glänzende Materialien haben einen kleinen Bereich, matte einen größeren
- das Phong-Modell ist ein empirisches Modell, das versucht diese Eigenschaft zu wiederzugeben Reflexions-Koeffizient ns: groß für glänzende Materialien, kleine für matte

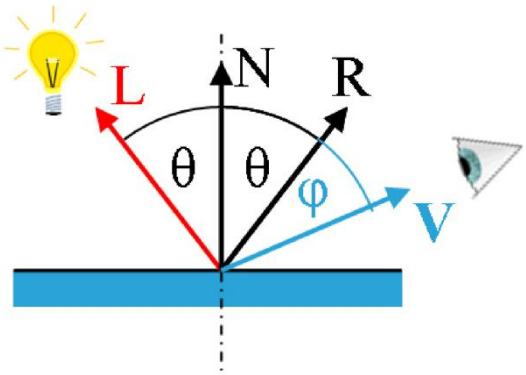
Weitere Aspekte

- Anisotrope Lichtquellen (Warn Model)
- Intensitätsabfall mit steigernder Entfernung
- Transparenz (Snell's Law)
- Schatten

69) Welche Faktoren spielen bei einer spiegelnden Reflexion (specular reflection) bzw. Glanzpunkte (specular highlights) eine Rolle? Wie schaut die specular reflcetion in einer Skizze in etwa aus?

Für einen Glanzpunkt sind folgende Faktoren wichtig (um eine Skizze zu zeichnen):

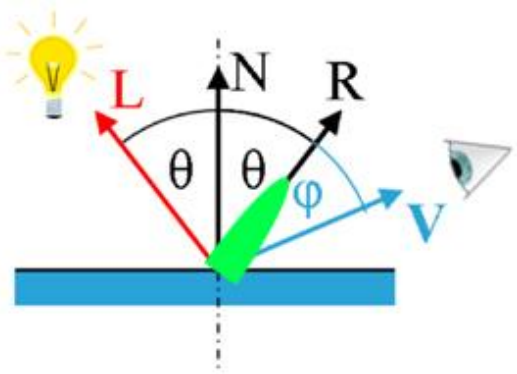
- Viewvektor V
- Normalvektor (der Oberfläche) N
- Lightvektor L
- Reflexionsvektor R



Θ ist der Winkel zwischen dem Lightvektor und dem Normalvektor. Da Einfallswinkel = Ausfallswinkel gilt, kann man somit auch leicht bestimmen wie der Reflexionsvektor aussieht.

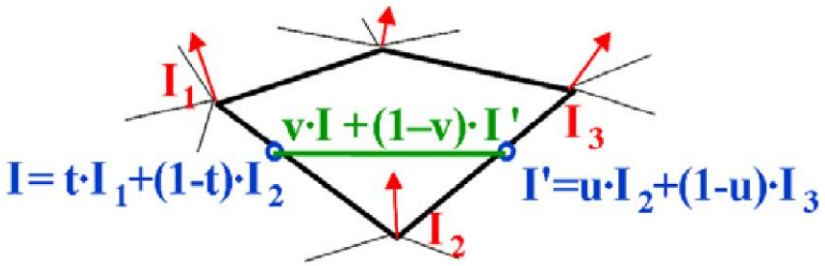
Der Winkel φ ist der Unterschied zwischen dem exakten Reflexionsstrahl und der Richtung zum Auge. Er ist ausschlaggeben dafür, wie „breit" der Glanzpunkt gezeichnet wird. Entspricht der Viewvektor genau dem Reflexionsvektor, erhält man den stärksten Punkt des Glanzlichtes. Umso weiter sich der Viewvektor vom Reflexionsvektor entfernt, umso größer wird der Winkel φ und somit auch der Glanzpunkt schwächer.

Somit ergibt sich folgende Schlussskizze, wobei der grüne Bereich angibt wo der Glanzpunkt gesehen wird, und wie „breit":



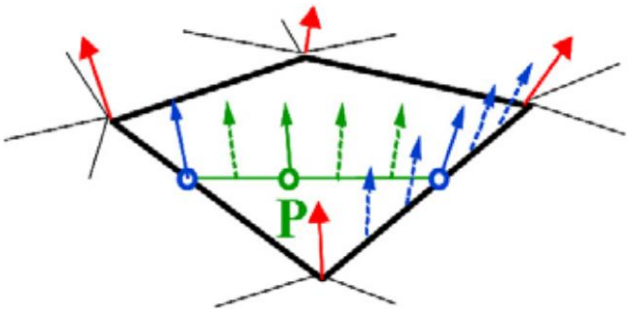
70) Wie funktioniert die Gouraud-Schattierung? (S. 592)

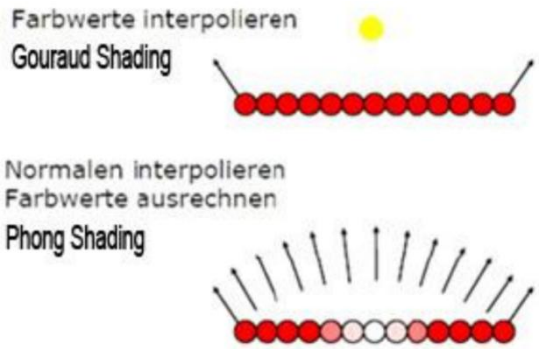
1. Berechnet die Normalvektoren in den Eckpunkten der Polygone
2. Berechnet die Lichtintensitäten in den Eckpunkten
3. Interpoliert die Licht-Intensitäten entlang der Polygon-Kanten
4. Interpoliert die Licht-Intensitäten für jeden Punkt auf der Scanline innerhalb des Polygons



71) Wie funktioniert die Phong-Schattierung? (S. 595)

1. Berechnet die Normalvektoren in den Eckpunkten der Polygone
2. Interpoliert die Normalvektoren entlang der Polygon-Kanten
3. Interpoliert für jeden Punkt auf der Scanline innerhalb des Polygons den Normalvektor
4. Berechnet aus dem interpolierten Normalvektor die Lichtintensität





72) Wozu verwendet man Gouraud- und Phong-Schattierung und was sind die Unterschiede und Gemeinsamkeiten(S. 523 ff.)

Interpolation von Farben bei Gourad, Normalvektoren bei Phong. Phong Illumination Modell sieht viel besser aus, da für jedes Pixel die Beleuchtung berechnet wird. Gourad aber schneller.

Was nötig damit Resultat Gouroudshading = Resultat Phongshading?

Das Modell müsste so detailiert modelliert werden, sprich so viele Polygone enthalten, dass sie nur noch als Pixel am Bildschirm angezeigt werden können und somit das gleiche Resultat bei Gouroud bieten würde als bei Phong.

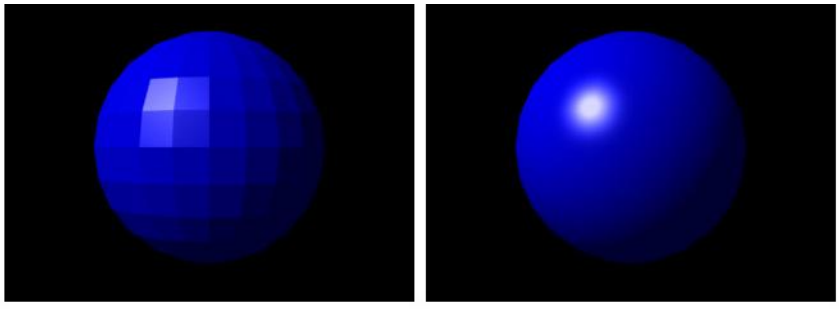
Weitere Informationen siehe Fragen zuvor.

73) Wie sieht das Beleuchtungsmodell aus, welches diffuse und spiegelnde (specular) Reflexionen mit mehreren Lichtquellen berücksichtigt? (S. 571)

Das ambiente Licht wird gleich berechnet wie bei nur einer Lichtquelle. Zusätzlich wird für jede Lichtquelle der diffuse und spiegelnde Anteil berechnet und für alle Lichtquellen aufsummiert.

74) Phong Shading vs. Phong Illumination Quelle1 Quelle2

Phong Shading ist ein Verfahren aus der 3D-Computergrafik, um Polygon-Flächen mit Farbschattierungen zu versehen. Beim Phong Shading werden an den Eckpunkten (Vertices) eines Polygons die Normalen berechnet und dann erweitert durch eine Berechnung von interpolierten Normalen entlang der Kanten für jede Projektion des Polygons auf einen Pixel. Farbstärken berechnen sich aus den interpolierten Normalen. Die Ergebnisse des Phong Shadings sind qualitativ besser als die des Gouraud Shading, allerdings sind die mathematischen Berechnungen aufwändiger. Durch die vermehrte Interpolation der Normalen erscheinen so facettierte Oberflächen eines dargestellten Objekts sehr weich.



FLAT SHADING PHONG SHADING

Das Phong-Beleuchtungsmodell ist ein Beleuchtungsmodell in der 3D-Computergrafik, das dazu verwendet wird, die Beleuchtung von Objekten zu berechnen. Das Phong-Modell ist zur Darstellung von glatten, plastikähnlichen Oberflächen geeignet. Dabei wird das Glanzlicht der Oberfläche durch den Term $\cos^n(\theta)$ beschrieben, wobei der Parameter *n* die „Rauhigkeit“ der Oberfläche bestimmt.

Es handelt sich um ein vollständig empirisches Modell, das auf keinerlei physikalischer Grundlage aufbaut. So erfüllt es z.B. den Energieerhaltungssatz nicht, der vorschreibt, dass eine Oberfläche nicht mehr Licht, als von der Lichtquelle zur Verfügung gestellt wird, reflektieren kann. Zudem ist es relativ langsam zu berechnen:

$$\begin{aligned} I_{\text{out}} &= I_{\text{a}}\, k_{\text{ambient}} + I_{\text{in}}\, k_{\text{diffus}}\, \cos\phi + I_{\text{in}}\, k_{\text{specular}}\, \cos^n\theta \\ &= I_{\text{a}}\, k_{\text{ambient}} + I_{\text{in}} \left(k_{\text{diffus}}\, \cos\phi + k_{\text{specular}}\, \cos^n\theta\right) \\ &= I_{\text{a}}\, k_{\text{ambient}} + I_{\text{in}} \left[k_{\text{diffus}} \left(\vec{L}\cdot\vec{N}\right) + k_{\text{specular}} \left(\vec{R}\cdot\vec{V}\right)^n\right] \end{aligned}$$

Das Phong Shading ist nicht zu verwechseln mit dem Phong-Beleuchtungsmodell. Es handelt sich bei letzterem um ein Beleuchtungsmodell und nicht um eine interpolative Schattierungstechnik. Es ist jedoch möglich, beide Techniken zu kombinieren.

75) Beleuchtungsmodelle (Illumination/ Lighting models) vs. Shading models Quelle

Beleuchtungsmodelle (illumination models, lighting models)

Beschreiben die Faktoren, die die Farbe eines Objekts an einem Punkt bestimmen.

Shading models

Beschreiben wann und wie ein Beleuchtungsmodell angewendet wird und erfüllen häufig (auch) den Zweck der visuellen Glättung von Polygonnetzen.

76) Was verstehen Sie unter Ray-Tracing? (S. 597 ff.)

Ist ein Verfahren in der Bildsynthese, bei dem die Ausbreitung von Lichtstrahlen simuliert wird.

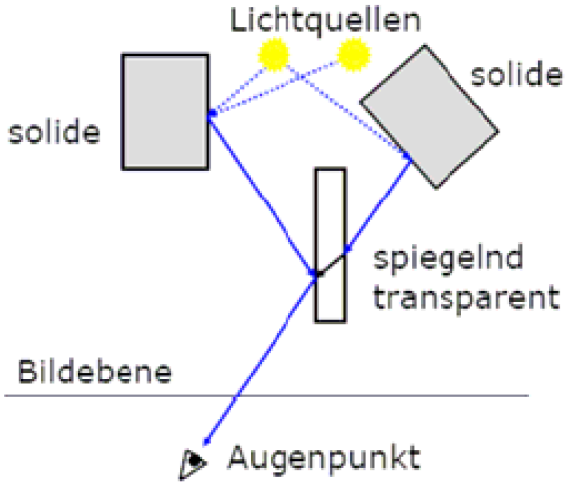
Das Ray-Tracing Prinzip

Die Basisidee ist es, das Licht, das auf einen Bildpunkt trifft, in umgekehrter Richtung zu verfolgen (also zu untersuchen, wo es herkommt) und daraus auf das Aussehen dieses Bildpunktes (Pixels) zu schließen.

Von einer Ebene vor der Kamera wird ein Strahl (ray) in die Szene „hineingeschossen“ und zurückverfolgt. Man spricht deshalb auch von backward ray tracing. Die Farbwerte der von den jeweiligen Strahlen getroffenen Objekte bestimmen die Farbe des entsprechenden Pixels. Backward ray tracing beschäftigt sich so mit der Frage, woher das Licht kommt.

Wenn der Strahl ein Objekt im Raum trifft, so wird er:

- refraktiert, also gebrochen (Glas, Luft)
- reflektiert (Spiegelungen und diffuse Abstrahlung)
- reflektiert und refraktiert, Anteile gemäß der Fresnel´schen Formeln



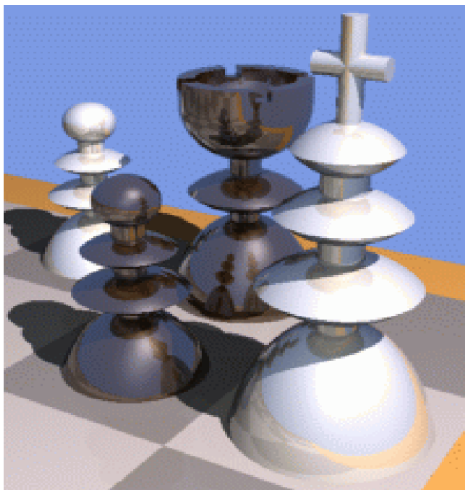
Korrekte Sichtbarkeit und Schattierung

Wie beim Ray-Casting legt man durch jeden Bildpunkt einen Blickstrahl („Primärstrahl“) und schneidet diesen mit allen Oberflächen der Szene. Aus den erhaltenen Schnittpunkten wählt man denjenigen aus, der am nächsten zum Bild liegt, und wählt die Schattierung dieses Objektpunktes (aus der Blickrichtung betrachtet) als Wert für den Bildpunkt. Tut man dies für alle Bildpunkte (also im einfachsten Fall für alle Pixel), so erhält man eine Abbildung der Szene in korrekter Sichtbarkeit. Dabei kann ein beliebiges Schattierungsmodell verwendet werden, also z.B. das Phong-Modell.

Schatten

Um die Schattierung eines Punktes zu berechnen braucht man neben der Oberflächennormale auch die Richtungen zu allen Lichtquellen. Eine Lichtquelle hat aber nur dann einen direkten Einfluss auf die Schattierung eines Punktes, wenn der Lichteinfall nicht durch andere Objekte verdeckt ist, wenn also keine Objekte zwischen dem Punkt und der Lichtquelle liegen. Um dies festzustellen, legt man einen Schattenfühler (das ist ein „Sekundärstrahl“) vom zu schattierenden Punkt zur Position der Lichtquelle, schneidet diese Gerade mit allen Objekten der Szene und verwirft die Lichtquelle, wenn man einen Schnittpunkt zwischen Objekt und Lichtquelle erhält. Auf diese Art erhalten alle

Objektteile, die im Schatten eines verdeckenden anderen Objektes bezüglich der Lichtquelle liegen, weniger Lichtquelleneinfluss als Objektteile, die von der Lichtquelle aus ungehindert sichtbar sind, und es entsteht daher automatisch ein Schattenwurf durch die dazwischen liegenden Objekte.



Spiegelbilder

Trifft der Blickstrahl auf ein Objekt auf, das ein Spiegel ist, so sieht man nicht dieses Objekt selbst, sondern jenes Objekt, das vom Auftreffpunkt aus in Spiegelungsrichtung sichtbar ist. Da das Reflexionsgesetz (Einfallswinkel ist gleich Ausfallswinkel) symmetrisch ist, kann man dieses gespiegelte Objekt dadurch finden, dass man den Blickstrahl an der Oberfläche spiegelt und in diese Spiegelungsrichtung einen Reflexionsstrahl (das ist auch ein „Sekundärstrahl“) verfolgt, daher wieder mit allen Objekten schneidet (also rekursiv arbeitet) und den zunächst liegenden Schnittpunkt auswählt. Die Schattierung dieses weiteren Auftreffpunktes (betrachtet aus der Eintreffrichtung des Reflexionsstrahles) ist dann das, was der ursprüngliche Blickstrahl sieht. Man beachte, dass das Reflexionsverhalten ganz lokal berechnet wird, man also ohne Mehraufwand gekrümmte Spiegel erzeugen kann.

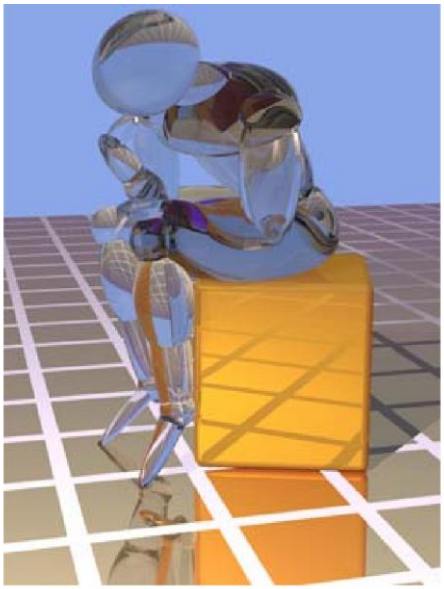
Durch die Rekursion spricht man auch von rekursivem Raytracing. Die Rekursion kann abgebrochen werden, wenn keine Objekte mehr getroffen werden, eine vorbestimmte Rekursionstiefe (eine Tiefe von 5-7 reicht oft aus) erreicht ist oder die Reflexion in der Rekursion keine nennenswerten Ergebnisse mehr beisteuert (engl. adaptive depth control).

Die Farbwerte der getroffenen Objekte werden dann nach ihrem Reflexionsgrad gewichtet, aufaddiert und mit der Lichtfarbe verrechnet. Analog verfährt Raytracing mit Strahlen, die nicht reflektiert, sondern in ein durchsichtiges Objekt gebrochen (Refraktion) werden.

Die Farbe der Oberfläche ergibt sich analog zum Lambert’schen und Phong Modell aus der Diffusions-, der Reflexions- und der Transparenzkomponente, wobei zusätzlich gelten muss, dass kd+kr+kt <= 1 sein muss.

Transparenz

Es bereitet nun auch keine Schwierigkeiten, transparente Objekte zu behandeln. Ist der erste Auftreffpunkt eines Blickstrahles auf einem durchsichtigen Objekt, so sieht man aus Blickrichtung das, was der durch das durchsichtige Objekt verlaufende Transparenzstrahl (auch wieder ein „Sekundärstrahl“) trifft. Dabei ist es ein Leichtes, vermittels des Brechungsgesetzes die Richtung des Transparenzstrahles so zu legen, dass das durchsichtige Material das Licht bricht. Der Transparenzstrahl wird ebenfalls mit allen Objekten geschnitten und der zunächst liegende Schnittpunkt ausgewählt. Die Schattierung dieses weiteren Auftreffpunktes (betrachtet aus der Eintreffrichtung des Transparenzstrahles) ist dann das, was der ursprüngliche Blickstrahl sieht.



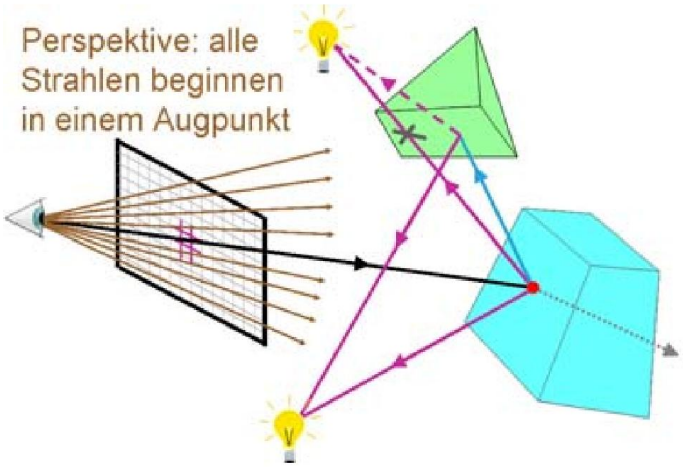
Rekursion

Jeder Strahl außer einem Schattenfühler, also jede Gerade, die einen verkehrt durchlaufenen Lichtstrahl repräsentiert, ist grundsätzlich gleichwertig. Daher trifft so ein Strahl auf eine Oberfläche, so ist es für die Aktion an dieser Stelle unerheblich, ob es sich um einen Primär- oder Sekundärstrahl

handelt. Auf diese Weise werden auch Mehrfachspiegelungen und Spiegelungen hinter transparentem Material usw. genauso einfach möglich.

Perspektive

Die Erzeugungsweise der primären Blickstrahlen bestimmt, wie die Abbildung der Szene auf die Bildebene erfolgt. Verwendet man parallele Strahlen, die normal auf die Bildebene stehen, so erhält man eine orthonormale Parallelprojektion der Objekte. Lässt man alle Blickstrahlen von einem fiktiven Augpunkt ausgehen, so wird das Bild in Perspektive erzeugt. Man beachte, dass die Perspektive dabei auf natürliche Weise ohne Mehraufwand entsteht (wenn man von Optimierungsverfahren absieht, die die Parallelität der Strahlen ausnützen).

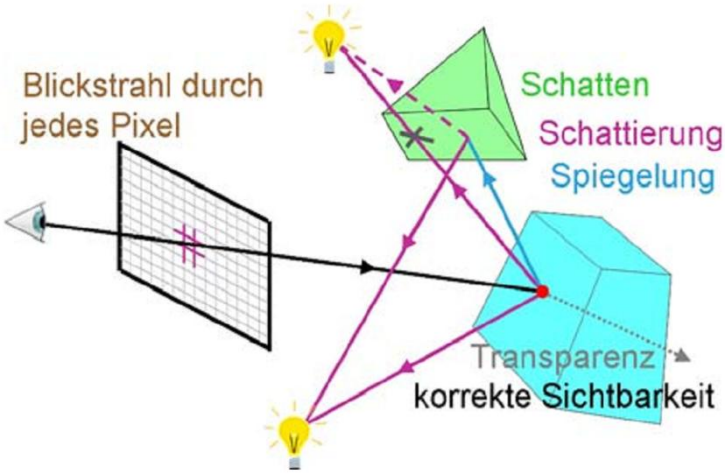


Raycasting vs. Raytracing Quelle

Im Gegensatz zu erweiterten Raytracing-Varianten ist beim Raycasting das Abtasten eines Strahls mit dem Aufeinandertreffen von Strahl und Objekt beendet, es findet also lediglich eine Verdeckungsberechnung statt. Die an diesem Schnittpunkt festgestellte Farbe bildet den Bildpunktfarbwert. Spiegelungen, Brechungen und Transmissionen des Objekts werden nicht beachtet. Diese Technik ermöglicht eine sehr schnelle Vorschau auf eine Szene.

77) Was sind Blickstrahlen beim Ray- Tracing? (S. 597 ff.)

Ray heißt Strahl und to trace heißt eine Spur verfolgen. Ray-Tracing ist also das „Verfolgen von Strahlspuren“, wobei Lichtstrahlen in verkehrter Richtung durchlaufen werden. Aufbauend auf dem Grundprinzip von Ray-Casting ist Ray-Tracing eine sehr mächtige Methode, mit der neben der korrekten Sichtbarkeit einige wichtige optische Effekte simuliert werden können: Schattierung, Schatten, Spiegelbilder, Lichtbrechung. Die Einfachheit des Verfahrens erlaubt es, auch sehr komplexe Objekte auf diese Art darzustellen, wie Freiformflächen, fraktale Oberflächen, mathematische Funktionen aller Art usw.



78) Was sind die Vorteile und Nachteile von Ray Tracing? (S. 597 ff.)

Vorteile

- Einfache Implementierung mit überschaubarer Komplexität
- Kein starres Schema wie bei der Render-Pipeline
- Leichtere Austauschbarkeit der Shader und dadurch erleichterte Implementierung neuer Shader
- Raytracing erzeugt, obwohl es in weiten Teilen nur von einer vereinfachten Realität ausgeht, dennoch überraschend realitätsnahe Licht- und Schatten- sowie Reflexionseffekte.
- Kann mit mathematischen Oberflächen- bzw. Volumenmodellen von zu modellierenden Objekten arbeiten. Die Oberflächen der Objekte müssen so nicht durch einzelne Flächen modelliert werden und liegen als auflösungsunabhängige Daten vor. Die Modellierung von Objekten mit CSG lässt sich leicht realisieren
- Ebenso ist die Berechnung von Schatten oder die Darstellung von Objekten, in denen sich das Licht bricht leicht realisierbar

Nachteile

- Es wird nur die direkte Beleuchtung simuliert und berücksichtigt daher nicht das indirekte Licht, dass von anderen Objekten reflektiert wird (wird nur in Monte-Carlo-Raytracing berücksichtigt)
- Es wird nicht berücksichtigt, dass bei der Refraktion Licht unterschiedlicher Wellenlängen stärker oder schwächer gebrochen wird (Dispersion) und das Licht so auffächert (Prismeneffekt). Dies liegt vornehmlich daran, dass Raytracer aus Gründen der Benutzerfreundlichkeit und Effizienz mit dem RGB-Farbmodell und nicht mit gesampelten Spektren arbeiten.
- Raytracing-Berechnungen gelten als sehr zeitintensiv. Die Schnittpunktberechnungen brauchen die meiste Zeit da bei jedem Strahl eine Schnittpunktberechnung durchgeführt werden muss.
- Seidenmatte Oberflächen kann nicht modelliert werden

79) Welche Konzepte der Reduzierung von Objekt-Strahl Schnittberechnungen kann man beim Ray-Tracing Algorithmus anwenden? (S. 604)

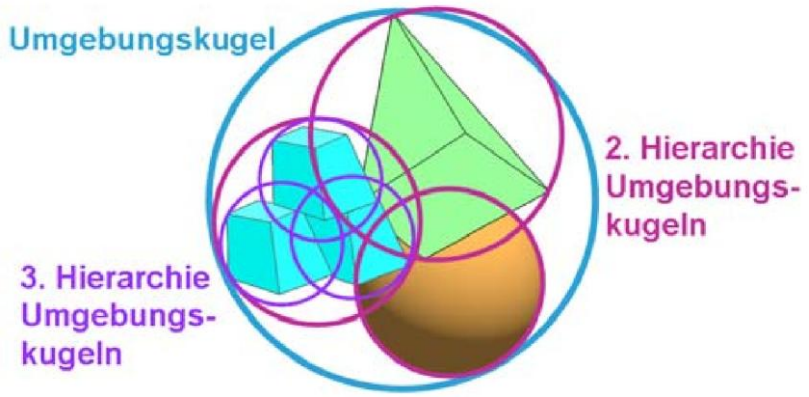
Ray-Tracing ist ein extrem aufwändiges Verfahren. Eine Szene mit nur 1000 Polygonen oder Objekten auf eine Fläche von 1000x1000 Pixel abzubilden erfordert ohne Optimierungen alleine für die Primärstahlen bereits 109 Schnittberechnungen. Daher ist es notwendig, das Verfahren signifikant zu

beschleunigen. Die wichtigste Methode dazu ist es, die Anzahl der notwendigen Schnittberechnungen durch Ausnutzung von Kohärenz zu reduzieren.

Schnitt-Berechnung machen bis zu 95% des Zeitaufwandes eines Ray Tracers aus, daher ist es wichtig, Verfahren zu benutzen, die die Anzahl der Schnitt-Berechnung minimieren.

Bounding Volumes bzw. Objektumgebungen

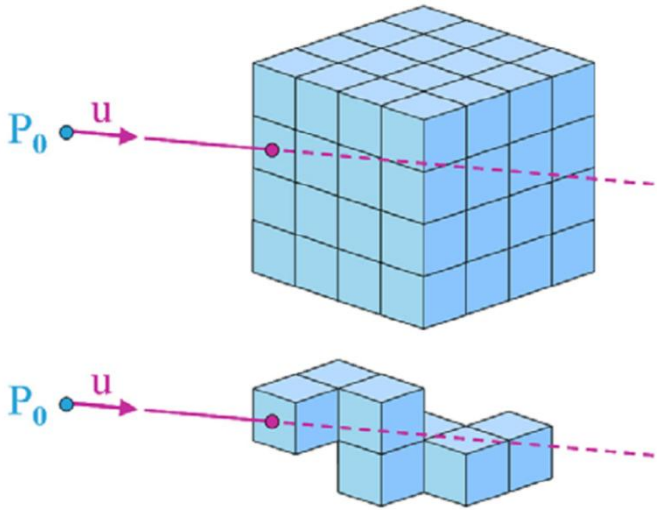
Bevor man alle Einzelteile eines komplexeren Objektes (oder eines Szeneteiles) mit einem Strahl schneidet, kann man überprüfen, ob der Strahl überhaupt in die Nähe dieses Objektes kommt. Dazu werden in der Datenstruktur zu solchen komplexeren Objekten Umgebungskugeln (bounding spheres) eingefügt, die das Objekt ganz umschließen. Strahlen, die diese Umgebungskugel nicht treffen, treffen auch das Objekt sicher nicht, und man erspart sich viele unnötige Schnittberechnungen. Das Konzept lässt sich hierarchisch anwenden, daher komplexe Teilobjekte erhalten alle ebenfalls Umgebungskugeln, deren Teile wieder usw., bis man an einfache Objekte gelangt. Auf diese Weise verringert man die tatsächlichen Schnittversuche von O (n) auf etwa O (log n). Statt Umgebungskugeln kann man beliebige andere Objektumgebungen verwenden, z.B. Umgebungsquader. Es gilt hier abzuwägen, ob der Mehraufwand durch deren kompliziertere Form den Gewinn durch deren engeres Anliegen rechtfertigt.



Space-Subdivision Methods

Alternativ kann man auch den ganzen Raum, in dem sich die Szene befindet, unabhängig von den Objekten in ein regelmäßiges Raster unterteilen. Dabei ist es egal, ob man die Teilwürfel in einem Array abspeichert oder einen Octree dazu verwendet. Man braucht dann nur jeweils die Objekte mit dem Strahl zu schneiden, die in den Teilräumen liegen, durch die der Strahl durchgeht. Man braucht also eine schnelle Berechnung, welches der nächste Teilwürfel auf dem Strahlpfad ist. Hat man in einem Teilwürfel einen Schnittpunkt gefunden, kann man aufhören! Dazu bieten sich Algorithmen ähnlich einem 3D-Bresenham-Verfahren an.

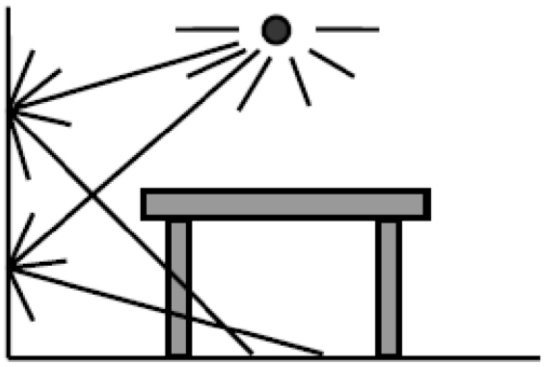
Für einzelne Teilwürfel kann man auch so vorgehen: Der Strahl $P = P0 + s \cdot u$ tritt an Pin in den Teilwürfel ein. Die Normalvektoren der Würfelflächen sind (1,0,0), (–1,0,0), (0,1,0), (0,–1,0), (0,0,1), (0,0,–1). Für die drei Flächen mit $u \cdot N > 0$ (die anderen kommen nicht in Frage!) bestimmt man den Schnittpunkt mit dem Strahl und wählt den vordersten Punkt (kleinstes s) aus. Diese Methode funktioniert auch, wenn die Würfel unterschiedlich groß sind (Octree).



Formen: Uniforme Subdivision (der Raum wird immer gleich aufgeteilt, z.B. in 8 Teile bei einem Octree) bzw. adaptive Subdivision (es wird nur dann weiter unterteilt, wenn sich in dem Teilraum Objekte befinden).

80) Erklären Sie Radiosity bzw. das Radiosity Modell (S. 615)

Ist ein globales Beleuchtungsmodell für die 3D-Computergrafik. Das Radiosity- Verfahren stammt ursprünglich aus der Wärmelehre und modelliert die Lichtausbreitung unter Beachtung des Energiegleichgewichtes in einem geschlossenen System. Das Verfahren beschreibt den physikalischen Vorgang der Ausbreitung von Licht in einer diffus reflektierenden Umgebung, also die Berechnung der Helligkeiten aller Flächen einer Szene unter Berücksichtigung der gegenseitigen Beeinflussung. Somit werden auch Flächen, die nicht direkt beleuchtet sind, eine gewisse Helligkeit erhalten. Jeder beleuchtete Gegenstand wirkt als sekundäre Lichtquelle. Für die Bildgenerierung wird zunächst die Lichtausbreitung im Raum berechnet, ohne dass die Kameraposition bekannt ist, wobei vereinfachend angenommen wird, dass der Beobachter die Ausbreitung des Lichtes nicht beeinflusst. Die Objekte können dann aus verschiedenen Richtungen dargestellt werden, ohne dass die Lichtausbreitung jedesmal neu berechnet werden muss.



Für jede Fläche wird eine Gleichung aufgestellt, die das emittierte Licht aus dem von den anderen Flächen empfangenen Licht und eventuell ihrer eigenen Leuchtkraft bestimmt. Insgesamt ergibt sich damit ein Gleichungssystem, dessen Lösung die Helligkeit jeder einzelnen Fläche angibt.

Mathematisch besteht das Verfahren in der Lösung eines Gleichungssystems. Hierfür kann ein iteratives Verfahren angewendet werden, das gegen die exakte Lösung konvergiert. Damit ist ein frei wählbarer Kompromiss zwischen Darstellungsqualität und Rechenzeit möglich.



Radiosity ist eine blickpunktunabhängige Methode zur Berechnung der Helligkeit der einzelnen (diffusen) Patches, nach der noch ein Renderingschritt notwendig ist. Meist verwendet man ein einfaches Polygon-Verfahren mit Gouraud-Schattierung. Um im fertigen Bild auch die mit Ray-Tracing möglichen Effekte erzielen zu können, kann man die so erzielten diffusen Schattierungswerte aber auch als Basiswerte für ein Ray-Tracing verwenden. Dadurch lassen sich dann auch Spiegelungen und Schatten usw. schön darstellen.

Das hier vorgestellte Grundprinzip von Radiosity lässt sich noch in vielfacher Hinsicht erweitern. Um die Anzahl der Patches zu verringern kann man diese hierarchisch strukturieren, so dass weiter entfernte Patches nicht einzeln behandelt werden müssen. Weiters gibt es diverse stochastische Ansätze, die einerseits die Formfaktoren, andererseits das Gleichungssystem mit Monte Carlo-Methoden zu lösen trachten. Bei der Path-Tracing-Methode werden Lichtstrahlen von den Lichtquellen aus verfolgt, also ähnlich wie bei Ray-Tracing, jedoch in Vorwärtsrichtung. An Auftreffpunkten wird die Wirkung des Lichtes gespeichert und später wird zwischen diesen Werten das Aussehen des Objektes interpoliert.

Radiosity vs. Raytracing Quelle

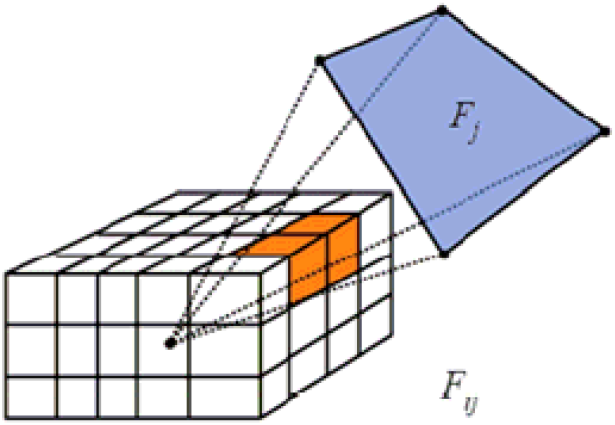
Historisch gesehen war Radiosity interessant, da es die Simulation indirekter diffuser Beleuchtung auf einfache Weise erlaubte, was mit Raytracing lange Zeit nicht möglich war. Andererseits war Raytracing gut für spiegelnde und transparente Objekte geeignet, wozu wiederum Radiosity nicht fähig war. Es wurden daher anfangs Vorschläge zur Kombination von Radiosity mit Raytracing gemacht, die jedoch aufwendig waren und sich letztendlich nicht im großen Maße durchsetzen konnten.

81) Wofür verwendet man Formfaktoren, wie berechnet man sie einfach und was sind ihre Eigenschaften? (S. 546)

Formfaktoren sind, unabhängig vom gewählten Algorithmus, der aufwendigste Schritt im Verfahren zur Berechnung der Radiosity (Berechnung eines Doppel-Integrals notwendig).

Ein Formfaktor gilt immer zwischen zwei Patches (= Flächen-Polygone mit konstanter Radiosity BK) und beschreibt die Menge der ausgetauschten Strahlung, liegt also zwischen 0 (keine Strahlung wird ausgetauscht) und 1 (alle Strahlung wird ausgetauscht).

Der Formfaktor ist rein geometrischer Natur und wird durch die Stellung der Patches zueinander bestimmt. Des Weiteren spielt die Sichtbarkeit der Patches eine Rolle. Die Sichtbarkeitsberechnung braucht bei weitem die meiste Zeit in der Berechnung.



Besondere Eigenschaften

- Konservierung der Energie: (Summe k=1..n) Fjk = 1
- Uniforme Lichtreflexion: AjFjk = AkFkj
- Kein Selbst-Einfall: Fkk = 0

Approximation der Berechnung durch Hemi-Cube Algorithmus

- Es wird ein repräsentativer Punkt der Empfängerfläche, der Mittelpunkt, ausgewählt.
- Die sichtbaren Teile der Senderfläche werden auf den Einheitswürfel (Approximation der Einheitshalbkugel nach Nusselt) um diesen Punkt projeziert.
- Die Flächen des Einheitswürfels sind in ein diskretes Gitter unterteilt.
- Jeder Gitterfläche wird ein Gewichtungsfaktor, der Delta-Formfaktor, zugeordnet, welcher von der Position der Gitterfläche abhängig ist. Ein Delta-Formfaktor ist damit der Formfaktor der Gitterfläche nach Nusselt. Die Summe der Delta-Formfaktoren ist 1.
- Für jede der fünf Halbwürfelflächen wird mit modifizierten Rasteralgorithmen (üblicher Weise Z-Buffer) ein Itembuffer berechnet. Dieser enthält für jede Gitterfläche die Identität der Objektfläche (Item, im Bild rotes Dreieck), die darauf projiziert wurde. Für jedes Item wird die Summe der Delta-Formfaktoren der überdeckten Gitterflächen berechnet (im Bild rote Gitterflächen). Diese Summe wird als Formfaktor zwischen Item und betrachteter Fläche aufgefasst.

82) Was sind die Vorteile und Nachteile des Radiosity Modells?

Vorteile

- Berechnung vom Standort und Blickwinkel des Betrachters unabhängig erfolgt ⇒ Berechnungen müssen so für eine Szene nur ein Mal gemacht werden. Danach kann die Szene in Echtzeit gerendert werden, was für Anwendungen wie virtuelle Architekturmodelle interessant ist.
- Diffuse Lichtreflexionen sind im Radiosity-Verfahren enthalten. Die Helligkeit und Farbe einer Fläche wird nicht allein aufgrund der direkten Beleuchtung einer Lichtquelle, sondern auch durch diffus reflektiertes Licht anderer Flächen bestimmt.

Nachteile

- Hauptnachteil des Radiosity-Verfahrens war, dass es bisher nur diffuse Reflexion kannte. Spiegelnde Flächen ließen sich mit dieser Methode nicht berechnen. Mittlerweile existieren zwar Verfahren, die das können, diese sind aber vorerst nur auf dem akademischen und noch nicht auf dem kommerziellen Level verfügbar.
- Für die möglichst realistische Darstellung einer Szene muss die Globale Beleuchtung simuliert werden, was aber nur in Spezialfällen mit Radiosity effizient möglich ist. Dies kann man zwar z.B. mit Monte-Carlo-Raytracing wettmachen, doch ist die Kombination mit Radiosity aus Effizienzgründen aufwändig, sodass hierzu meist nur entweder die eine oder die andere Technik verwendet wird.
- Simulation von Materialien mit beliebigen Beleuchtungsmodellen ist nur schwer möglich.

83) Erklären Sie Environment und Texture Mapping (S. 623 ff.)

Environment-Mapping

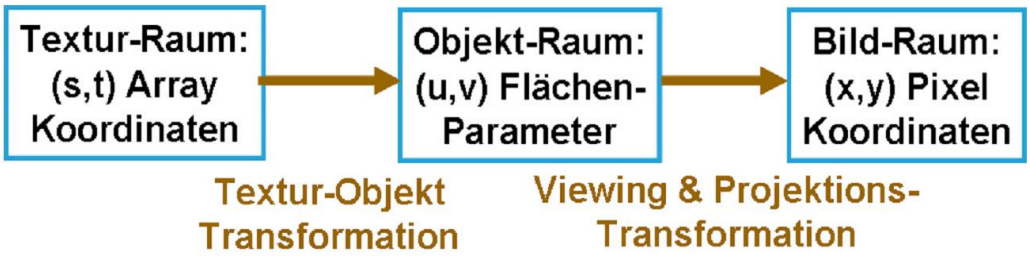
Ist eine Form des Reflexions-Mapping. Dabei werden Informationen des Universums (Kugel, Würfel, Zylinder), das ein Objekt umgibt, abgebildet und auf das Objekt selber aufgetragen ⇒ Textur enthält Projektion der Umgebung. Der Zugriff ist vom Augenpunkt abhängig.

Environment bezeichnet die Umgebung, die Umwelt eines Objektes. Je nach Oberflächeneigenschaften des Objektes wird sich die Umgebung verschieden auf das Erscheinen des Objektes auswirken. Für perfekt spiegelnde Oberflächen können wir mit Ray-Tracing das exakte Spielgelbild der Umgebung erzeugen. Für nicht perfekt diffuse Oberflächen können wir mit dem Phong-Modell Glanzeffekte in der Umgebung der Spiegelung von Lichtquellen annähern. Eine mehr oder weniger spiegelnde Oberfläche reflektiert aber ihre ganze Umgebung mehr oder weniger scharf, wobei die Genauigkeit sehr reduziert ist. Um nun effizient Objekte in einer komplexen Umgebung zu rendern ohne diese ganze Umgebung vollständig modellieren zu müssen und ohne Anspruch auf Exaktheit, verwendet man Environment Mapping. Dazu wird die Umgebung in einem Vorverarbeitungsschritt von einem zentralen Punkt aus (z.B. dem Mittelpunkt des Objektes oder der darzustellenden Szene) als Bild produziert. Ob das nun ein berechnetes oder ein fotografiertes Bild ist, spielt keine Rolle. Jedenfalls geht man davon aus, dass die Umgebung (z.B. Kugel oder Würfel) groß ist im Verhältnis zu den Objekten, die man darstellen will. Nun wird bei der Darstellung eines Objektes für jeden Oberflächenpunkt näherungsweise angenommen, dass er im Zentrum dieser

Umgebung liegt. Dadurch kann man allein aus der Richtung des Reflexionsstrahles schnell bestimmen, welcher Umgebungspunkt getroffen wird (z.B. mit Polarkoordinaten), und erspart sich aufwändiges Ray-Tracing.

Texture-Mapping

- Texturmuster wird auf die Oberfläche des Objekts aufgebracht
- Texturen erhöhen den Realismus von computergenerierten 3D-Szenerien deutlich, ohne dass die Anzahl der verwendeten Polygone erhöht werden muss.
- Bei Polygonen, die eine größere Ausdehnung in Sichtrichtung haben muss eine Perspektiven-Korrektur durchgeführt werden.
- Antialiasing wird verwendet um bessere Ergebnisse zu erzielen.



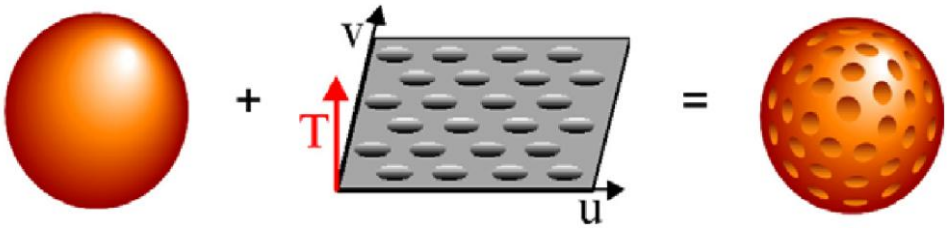
Viele Oberflächen sind nicht einfärbig, sondern haben ein Muster, z.B. Holzmaserung, Bilder an der Wand, Schrift auf Papier, Verschmutzung, Kleider, Marmor. Auch bei grober Modellierung lassen sich Details als Muster interpretieren, z.B. Fenster auf einer Hauswand, Wolken am Himmel, Gesichter, Knöpfe und Reißverschlüsse, Pflastersteine. Solche Muster bezeichnet man als Textur. Das Aufbringen von Texturen wird Texture Mapping genannt.

Nach der Erzeugung der Texturen erfolgt Texture Mapping in zwei Schritten. Zuerst muss definiert werden, welche Textur auf welche Oberfläche der Objekte wie orientiert, skaliert etc. aufzubringen ist. Dies ist eigentlich ein Teil der Modellierung, wo die Oberflächen ein Aussehen erhalten. Und im zweiten Schritt muss dann die Textur korrekt auf das Abbild der Objekte gerendert werden, also in das Bild transformiert werden.

84) Erklären Sie Bump Mapping, seine Funktionalität sowie Vorteile und Nachteile (S. 634)

Ist eine spezielle Form des Texture-Mappings, bei der durch die Modifikation von Normalvektoren eine Textur aufgebracht wird, die durch die Verwendung von Schattierungen einen 3D-Effekt erzielt. Viele Oberflächen haben eine geometrische Detailstruktur: Rinde, Münzen, Verputz, Leder, Stoff, Gemüse, Planeten, Schotterwege, Lasagne, Fliesen, Schokoladetafel, Regenwurm usw. Solche Objekte komplett zu modellieren ist sehr mühsam und erzeugt riesige Datenmengen. Mit Bump-Mapping kann man diesen Aufwand signifikant reduzieren. Bump Mapping ist somit nur ein (sehr wirksamer) Darstellungs-"Trick", der Oberflächenunebenheiten simuliert, die in der Geometrie des Modells gar nicht vorhanden sind.

Wenn man die graue Leiste oben ansieht, so hat man den Eindruck, dass sie sechs Ausbuchtungen und eine Einbuchtung hat. Wenn man sie aber angreift, ist sie ganz eben! Warum sehen wir die Unebenheiten? Weil die Schattierung alleine ausreicht um einen räumlichen Eindruck zu erwecken! Diesen Trick kann man verwenden, um mit wenig Aufwand den Eindruck von Oberflächenunebenheiten (Bumps) zu erwecken. Die Grundidee ist es, die Oberfläche unverändert zu lassen, aber den Normalvektor entsprechend der Unebenheiten zu verändern. Dadurch entspricht die Schattierung den Bumps, aber geometrisch braucht man nichts verändern.

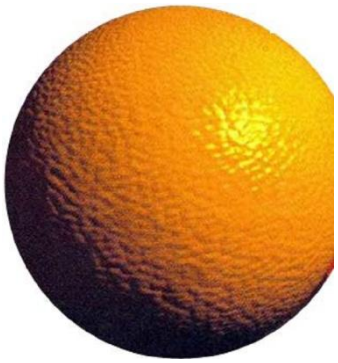


Vorteile

- Bumpmapping verbessert die Bildqualität und den Realismusgrad ohne sich groß auf die Performance niederzuschlagen.
- Reflexionen können mit dieser Technik ebenfalls dargestellt werden

Nachteile

- Manche Oberflächen schauen noch immer nicht real aus, weil die Schatten an den Rändern der Bumps nicht stimmen bzw. Schatten mit glatten Rändern wirft (obwohl sie nicht „glatt“ sein sollten).
- Der Rand (Silhouette) der Objekte (z.B. Kugel) bleibt unverändert (perfekt rund), also „eben“/ flach wie man auch gut unten im Rendering sieht.
- Schattierung wird nur geändert, nicht Geometrie.
- Bei flachen Winkeln wird die Struktur stark verzerrt.
- Bumps werfen keine Schatten aufeinander!
- Oberflächennormalen auf der lichtabgewandten Seite eines Objektes, die zum Licht gewandt sind, erhalten trotzdem Licht



Für jeden dieser Fehler gibt es mehr oder weniger aufwändige Hilfslösungen, korrekt ist es aber, die Oberfläche tatsächlich um die Bumphöhe zu verändern. Diese Methode nennt sich dann Displacement Mapping (siehe nächste Frage).

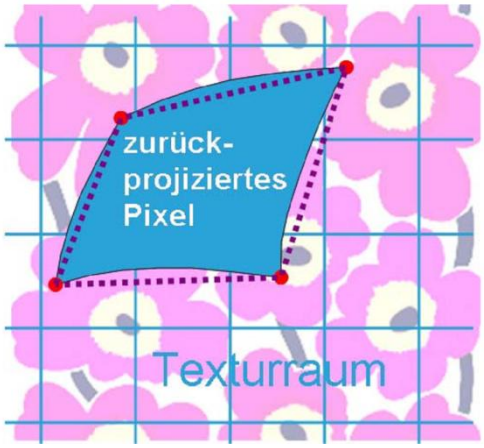
85) Was ist Displacement Mapping?

Anmerkung: Diese Frage fand ich in keiner Angabe bis jetzt, ist aber SEHR wichtig und kann mir sehr gut vorstellen das sie kommen wird!

Dabei werden die Oberflächenpunkte tatsächlich verschoben und es entsteht natürlich auch eine korrekte Silhouette. Dies ist aber natürlich viel aufwändiger zu implementieren und wird daher selten verwendet. Allerdings gibt es einen Trend dazu, diese Möglichkeit auf der Graphikkarte in Hardware zu unterstützen.

86) Mipmaping (Seite 633) Quelle

Texturen sind für Aliasing-Effekte besonders anfällig, insbesondere wenn die Muster vergrößert oder verkleinert werden. Korrekt müsste man den Textur-Durchschnittswert der Fläche berechnen, die von einer Rückprojektion des zu füllenden Pixels auf die Textur erzeugt wird. Näherungsweise reicht auch das Viereck, das durch gerade Verbindung der rückprojizierten Eckpunkte entsteht. Da dies sehr langsam wäre, verwendet man häufig eine von zwei Optimierungen: Mip-Mapping, bei dem die Textur in verschiedenen Größen vorberechnet und dann linear interpoliert wird (und die Summed-Area-Table-Methode, wo man aus einer Summentextur durch Differenzenbildung leicht die Durchschnittswerte rechteckiger Bereiche ermittelt).



Mip- Mapping ist eine Antialiasing-Technik für Texturen. Sie wird in modernen 3D-Grafikchips zur Verbesserung der Bildqualität, aber auch der Geschwindigkeit eingesetzt.

Probleme bei der Texturskalierung

Das Problem beim Texture Mapping von Objekten besteht darin, dass man diese von nah und fern betrachten können soll. Korrelativ zum Erscheinungsbild des Objekts muss also auch die Textur ihre Form und Größe verändern:

- *Maxifikation* - die Textur muss stark vergrößert werden, um das Objekt zu bedecken, wenn es nah am Betrachter liegt.
- *Minifikation* - die Textur muss stark verkleinert werden, wenn das Objekt sehr weit vom Betrachter entfernt ist (im Extremfall ist das Objekt dann nur noch genau 1 Pixel groß)

Werden keine Gegenmaßnahmen ergriffen, treten bei diesen Skalierungen Aliasing-Effekte auf, die als Artefakte sehr störend sind, besonders bei Mustern mit schrägen oder gerundeten Linien, als Verzerrungen oder Flimmern (Moiré-Effekte).

MIP-Map

MIP ist eine Abkürzung für **M**ultum **I**n **P**arvo, was soviel bedeutet wie „viel auf kleinem Platz“. Eine MIP-Map (auch Bildpyramide) ist eine Folge von Rasterbildern desselben Motivs, jedoch mit abnehmender Auflösung. Die Kantenlänge jedes Bildes ist genau halb so groß wie die des Vorgängerbildes. Das kleinste Bild hat je nach Implementierung eine Größe von 1x1 oder 2x2 Pixel. Daraus folgt, dass alle Bilder quadratisch sein müssen und als Kantenlänge eine 2er-Potenz haben. Man spricht bei diesen Stufen auch von Level of Detail (LOD).

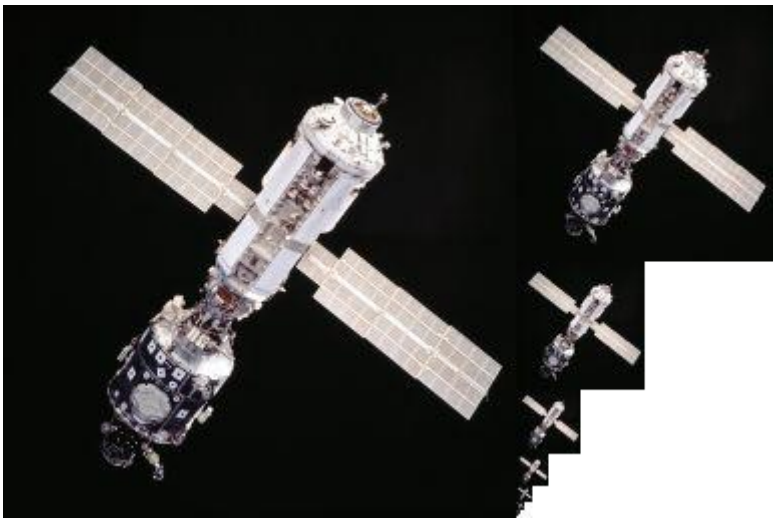
MIP-Maps haben einen um 1/3 höheren Speicherbedarf als das größte Bild alleine: $1 + 1 / 4 + (1 / 4)^2 + (1 / 4)^3 + (1 / 4)^4 ... = 1 + 1 / 4 + 1 / 16 + 1 / 64 + 1 / 256 + ... = 1 + 0.25 + 0.0625 + 0.015625 + 0.00390625... = 1.33203125 = 1 + 1 / 3$

MIP-Maps können einfach berechnet werden, indem für jedes Pixel des verkleinerten Bildes der Mittelwert der vier korrespondierenden Pixel des Ausgangsbildes berechnet wird. Dies ist rekursiv für alle weiteren Stufen durchzuführen. Dieses Verfahren wird häufig implizit von der Grafikhardware beim Texturladen angewandt, um Speicherbandbreite zu sparen. Eine aufwändigere aber zu besserer Bildqualität führende Methode zur MIP-Map-Berechnung bietet die Fouriersynthese.

MIP-Mapping als Texturfilter

Bei der Minifikation von Texturen beim Texture Mapping wird dasjenige LOD ausgewählt, bei welchem ein Texel groß genug ist, um ein Pixel vollständig zu überdecken.

Auf diese Weise ist sichergestellt, dass jedes Texel der Ursprungstextur Einfluss auf die gestauchte Textur hat und somit keine abtastbedingten Alias-Effekte auftreten. Leider führen gerade die Sprünge zwischen den LODs zu Unstetigkeiten, die als Linien zwischen den MIP-Bändern deutlich sichtbar sind. Jedes dieser Bänder hat einen Schärfegrad, der sich stark von den Nachbarbändern unterscheidet. Außerdem kommt es bei jeder Stauchung, die keiner 2er Potenz entspricht, und auch bei anisotropen Verzerrungen immer zu einem übermäßigen Schärfeverlust.



Beispiel für eine MIP-Map. Die Kantenlänge des größten Bildes beträgt 2^8 = 256 Pixel.

87) Was ist der Machbandeffekt (S. 594) Quelle

Bei einer Abfolge von Flächen unterschiedlicher Graufärbung, die in sich keine Farbgraduierung haben, beobachten wir entlang der Grenzen machsche Streifen (nach Ernst Mach 1865). Hierbei handelt es sich um helle und dunkle Streifen, die den Kontrast zwischen den Flächen verstärken.

Ein basaler Mechanismus für die Verarbeitung visueller Eindrücke beim Menschen ist das Erkennen von Linien und Kanten. Strichzeichnungen erkennen wir oft schneller als Fotos, die nur Farbschattierungen enthalten. Die Marskanäle sind ein Beispiel für eine Sinnestäuschung, da auch Linien zu sehen sind, wo eigentlich gar keine sind.

Bereits die Verarbeitung im Auge betont Kontraste. Obwohl in der Abbildung rechts die oberen Flächen nur diskrete Grautöne besitzen, sehen wir an den Übergängen einen Helligkeitsverlauf, der den Kontrast überhöht: Die dunkle Kante erscheint dunkler, der helle Bereich heller. Der Graukeil unten im Bild erscheint inhomogen, obwohl sich der Grauwert von links nach rechts linear ändert.

Die Ursache für dieses Wahrnehmungsphänomen liegt in der Verschaltung der Rezeptoren in der Netzhaut. Ca. 100 Mio. Rezeptoren steuern ca. 1 Mio. rezeptive Felder, die die Signale zur Verarbeitung bereitstellen. Durch die Verschaltung verstärken die Rezeptoren nicht nur Signale, sondern können sie abhängig vom Entstehungsort auch abschwächen (laterale Hemmung).

Dadurch wird das Signal an den Grenzen zweier unterschiedlich heller Bereiche stärker oder weniger stark gehemmt und es ergeben sich die machschen Streifen.

Der Effekt tritt unter anderem in der Computergrafik bei berechneten, beleuchteten Flächen auf (z. B. beim Flat Shading).

88) Praxisfrage: Wie Igel modellieren und darstellen, damit er realistisch aussieht?

Gibt verschiedene Möglichkeiten mit ihren Vor und Nachteilen:

- Viele Polygone.
- Wenige Polygone und dafür Bump- Mapping.
- Noch weniger Polygone und dafür Displacement Mapping.
- Andere Techniken: NURBS (Flächen/ Objekte) oder Subdivision Surfaces
- „Grundformen"/ Primitiven: Ein Kreis für den Körper, eine Pyramide für die Nase, ...

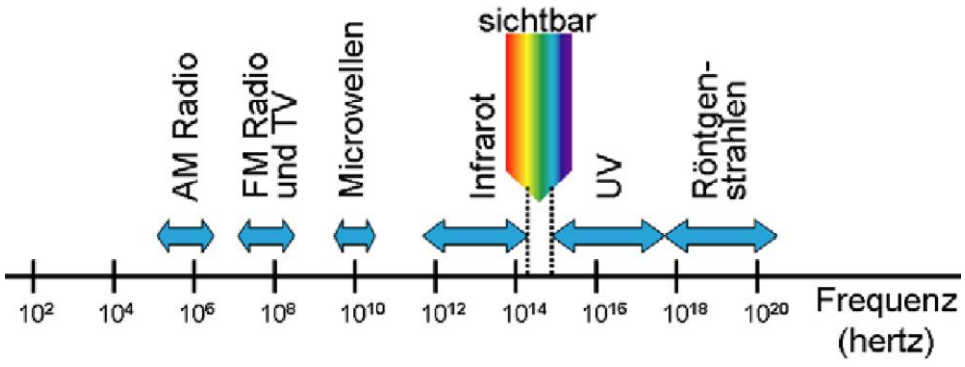
Darstellen

Berechnen mit Ray-Tracing, vielleicht auch mit Radiosity oder Global Illumination. Den auf der Tischplatte erscheinenden hellen Fleck (Kaustiken, durch das fokussierte Licht) geht mit Raytracing NICHT, da zwar bei spiegelnden Reflexionen, nicht jedoch bei diffusen Oberflächen Sekundärstrahlen ausgesendet werden. Kaustiken erhält man nur mit Path Tracing, Light Ray Tracing, Global Illumination,…!

Kapitel 12 – Color Models and Color Applications (S. 712)

89) Was ist Licht bzw. Farbe? (S. 713 ff.)

Licht ist nichts anderes als ein schmales Spektrum elektromagnetischer Wellen. Jeder Frequenzwert des elektromagnetischen Spektrums entspricht einer Spektralfarbe. Bei den niedrigen Frequenzen befinden sich die Rottöne, bei den hohen Frequenzen die Blautöne.



Der sichtbare Bereich liegt zwischen 780nm und 380nm. Weitere Parameter, die die Farbe beschreiben, sind Helligkeit abhängig von der Leuchtkraft der Quelle (wie viel Licht das Objekt

reflektiert) und die Sättigung (wie intensiv die Farbe ist – Abmischung selbiger zwischen weiß und schwarz).

90) Was sind Primärfarben bzw. Komplementärfarben? (S. 716)

Komplementärfarben

Mischung ergibt weißes Licht (z.B. rot – cyan, grün – magenta, blau – gelb)

Primärfarben

- Meistens 3 Farben, z.B. Rot-Grün-Blau
- Alle möglichen Farben innerhalb des Farbmodels (= „color gamut“) können damit durch Mischen hergestellt werden
- Dennoch sind nicht alle sichtbaren Farben möglich (manche liegen außerhalb des „color gamut“)

91) Was versteht man unter subtraktiver Farbmischung? (S. ???) Quelle

Die Subtraktive Farbsynthese (auch Subtraktive Farbmischung) ist ein optisches Modell, das die Entstehung von Körperfarben beschreibt. Reflektiert ein Körper Licht, wird dabei nicht nur die Lichtmenge, sondern auch Farbe des Lichtes verändert. Dabei werden immer Anteile der Lichtfarbe reduziert (subtraktiv = reduzierend). Werden dagegen Lichtfarben (hinzu)gemischt, spricht man von additiver Farbsynthese.

Drei Farbfilter in den Primärfarben Gelb (Y), Magenta (M) und Cyan (C), teilweise übereinander liegend. Wird diese Fläche mit einem neutralweißen Licht beleuchtet, werden durch Absorption komplementärfarbige Spektralbereiche herausgefiltert. So entstehen als Sekundärfarben die Farben Rot (R), Grün (G) und Blau (B). Wo alle drei Filterschichten übereinander liegen, entsteht Schwarz (K), weil kein Licht die drei Filterschichten mehr passieren kann.

Quantitativer physikalischer Aspekt

Quantitativ wird die spektrale Leistungsverteilung von Licht beim Durchdringen eines Filters in den relevanten Wellenlängenbereichen um einen Faktor (zwischen 0 und 1) verkleinert. Sie wird also nicht subtraktiv, sondern multiplikativ verändert. Der Begriff multiplikative Farbmischung hat sich aber in der Farbenlehre nicht durchgesetzt.

92) Was versteht man unter „color gamut“ und welcher Zusammenhang zum CIE Diagramm besteht? (S. 719) Quelle

Color gamut

Viele Geräte wie Scanner, Bildschirme und Drucker haben einen unterschiedlichen Farbenumfang. Der Farbenumfang ("Colorgamut") umfasst alle Farben, die ein Gerät darstellen kann.

Weder Drucker, Bildschirme und Flachbildschirme können alle für den Menschen wahrnehmbare Farben abbilden. Insgesamt können Bildschirme auch deutlich mehr Farben darstellen als Drucker wiedergeben. Daraus resultieren Bildschirmfarben, die beim Drucken nur angenähert werden können.

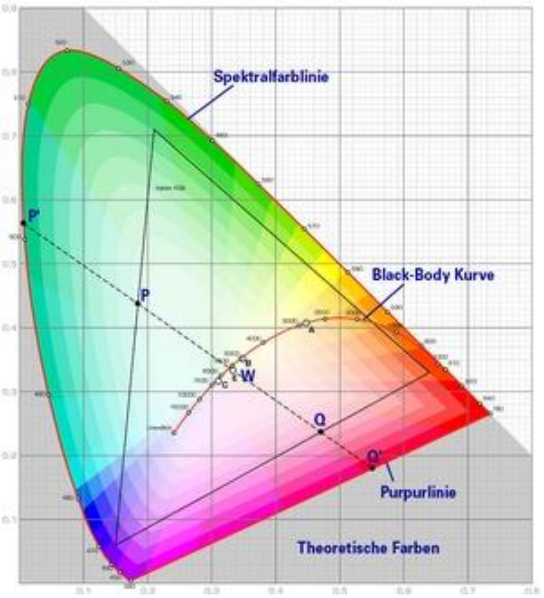
Professionelle Grafiksoftware kann nicht druckbare Farben in einem Bild markieren. Ein entsprechend ausgebildeter Grafiker hat dann die Möglichkeit, die Druckausgabe an dieser Stelle manuell zu optimieren.

Umgekehrt können im CMYK - Farbmodell Farben erzeugt werden, die im RGB Farbmodell nicht vorhanden sind, wie z.B. reines Gelb. Da die Erzeugung und Bearbeitung fotorealistischer Bilder aber in der Regel im RGB Farbraum erfolgt, treten hier keine nachteiligen Effekte auf. Ein Grafiker kann diese Eigenschaften aber ausnutzen.

- Wenn man (2 oder) 3 Farben auswählt, die die Primärfarben eines Farbmodells darstellen sollen und im Diagramm (eine Linie bzw.) ein Dreieck zwischen ihnen aufspannt, so wird dies als „color gamut“ bezeichnet.
- Alle Farben die (auf der Linie bzw.) innerhalb des Dreiecks liegen, können mit den Primärfarben dargestellt werden.
- Man sieht auf den ersten Blick, dass die Fläche immer ein konvexes Dreieck ist und daher nie alle Farben umfassen kann.

CIE

Darstellung von Farben beruhend auf dem menschlichen Wahrnehmungsapparat. Additive Farbmischung. Grundlage ist ein sogenannter Normalbeobachter mit einem Sichtfeld von 2° (CIE1931) bzw. 10°(CIE1964). Normfarbwerte X (virtuelles rot), Y (virtuelles grün), Z (virtuelles blau)



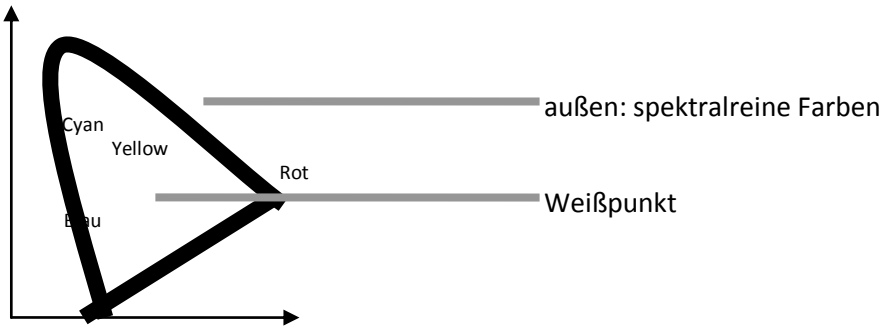
- **Darstellung:** Querschnitt durch den Raum; alle Farben befinden sich innerhalb der Fläche.
- **Wellenlänge:** gebogene, äußere Linie.
- **Purpurgerade:** gerade Verbindungslinie zwischen Rot und Blau.
- **Spektralfarben:** höchster Sättigungsgrad; am äußeren Kurvenrand, reines Weiß in Zentrumsnähe (C)
- **Mittelpunktvalenz:** x = 0,33, y = 0,33 (C)
- **Dominate Wellenlänge:** zwischen Farbe und Weißpunkt Gerade ziehen und bis Spektralkurve verlängern.
- **Komplementärfarbe:** liegt gespiegelt zum Weißpunkt auf selben Gerade.

Man ermittelt mit Farbvergleichsexperimenten welche Kombination von den 3 Grundfarben welche Farbe erzeugt. Da sich manche Farben aus keiner solchen Kombination erzeugen lassen, muss man manchmal auch negative Anteile verwenden, was dadurch geschieht, dass man zur zu erzeugenden Farbe etwas Licht dazu mischt. Dadurch ergeben sich virtuelle Grundfarben X, Y und Z, die es in Wirklichkeit nicht gibt. Normiert man die so erzeugten Farben auf die Helligkeit 1 und projiziert das Ergebnis auf die XY-Ebene, so erhält man das CIE-Diagramm, das 1931 von der **C**ommission **I**nternationale d'**E**clairage normiert wurde. Die Farben werden durch die Koordinaten (x,y) beschrieben, aus x+y+z=1 folgt z. Zusätzlich kann man noch die Helligkeit angeben, diese wird mit Y bezeichnet. Somit ist eine vollständige Farbdefinition durch (x,y,Y) gegeben.

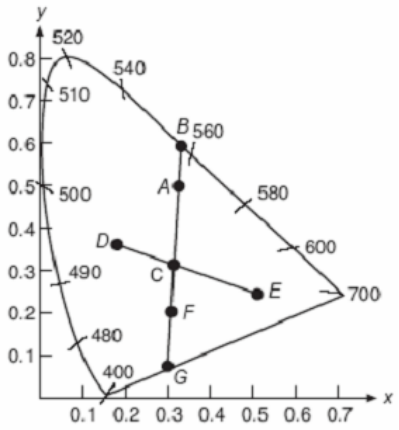
Die mit einem RGB-Monitor darstellbaren Farben, also die Linearkombinationen der Farben Rot, Grün und Blau, die der Monitor erzeugen kann, liegen alle innerhalb des von diesen 3 Punkten aufgespannten Dreieckes (siehe Skizze). Da es keine drei Farben gibt, die das ganze Diagramm enthalten, kann kein Monitor alle Farben darstellen.

Wenn A die betrachtete Farbe ist, so wird sie durch die Wellenlänge B dominiert. Wird B mit C gemischt, so erhält man A. Die in F dominierende Wellenlänge G ist definiert als das Komplement der dominierenden Wellenlänge B in A. Werden Komplementärfarben gemischt (E mit D, F mit A), so

erhält man weiß. Deshalb kann CIE benutzt werden um Farben zu identifizieren und analysieren, oder um die Leistung eines CRT zu messen (wie viel des CIE Farbraums kann er darstellen)



Auf der äußeren Kurve liegen die reinen Spektralfarben, ca. in der Mitte liegt Weiß(C). Wenn eine Farbe A im Raum zwischen Weiß und der Spektralkurve liegt so bekommt man die dominante Wellenlänge, indem man die Gerade, die durch C und A geht bis zur Kurve verlängert. Der Schnittpunkt zeigt dann die dominante Farbe an. Würde der Schnittpunkt auf der Geraden liegen die den Raum abschließt, so wird die dominante Wellenlänge als das Komplement von A genommen.



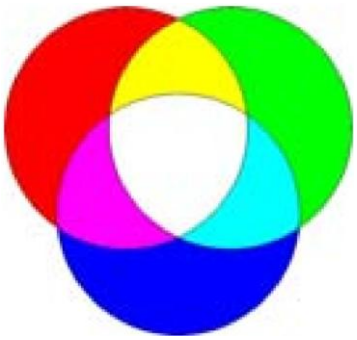
Verwendung von CIE

- Der CIE Farbraum wird verwendet um die anderen Farbräume (CMYK, RGB, usw.) zu kalibrieren, da alle in diesem Modell enthalten sind.
- zur Identifikation und Analyse von Farben.
- zur Messung für CRT Leisten („Performancemessung“), sprich wie viel vom CIE Farbraum. wird durch CRT Abgedeckt.

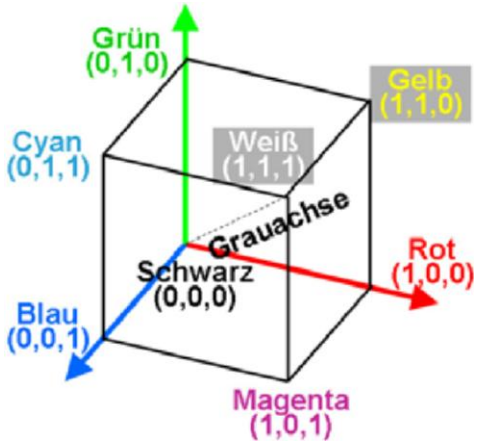
93) Was wissen Sie zum RGB-Farbsystem? (S. 720)

Neben Farbräumen (eigentlich Farbraumbeschreibungen) wie dem CIE-Modell, die alle Farben zu beschreiben imstande sind, gibt es Farbräume zur Beschreibung der Farben eines Gerätes. Für Bildschirm wird fast immer das RGB-Modell verwendet. Dabei wird ein Pixel aus drei kleinen Farbpunkten zusammengesetzt, deren Lichtsumme (additive Farbmischung!) einen Farbeindruck erzeugt.

Je nach verwendeter Technologie und konkreten Materialien hat jeder Monitor geringfügig unterschiedliche Grundfarben, aus denen unterschiedliche Teilmengen aller Farben erzeugt werden können. Den Raum der erzeugbaren Farben nennt man Gamut.



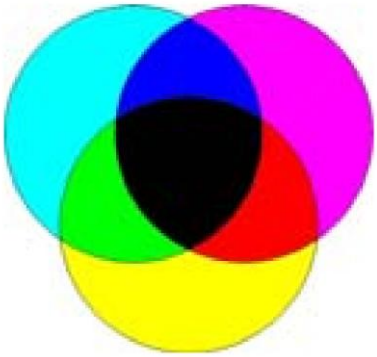
- Primärfarben: Rot, Grün, Blau
- Additives Farbmodell (z.B. für Monitore)
- Basiert auf der menschlichen Farbwahrnehmung
- Farbe wird dargestellt, indem man das Beleuchtungsmodell auf die 3 Primärfarben anwendet
- Darstellung durch einen Würfel



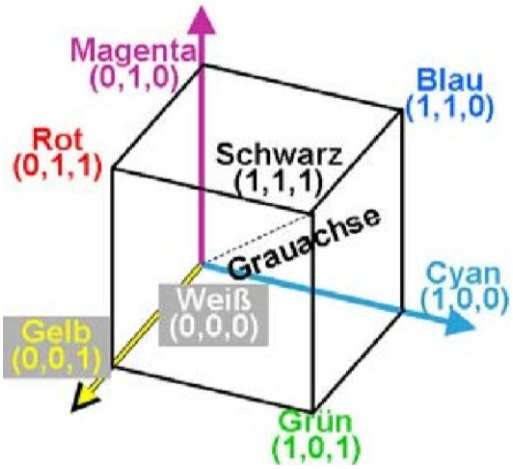
94) Was wissen Sie zum CMY-Farbsystem? (S. 723)

Das Mischen von farbiger Tinte auf einem Blatt Papier unterliegt ganz anderen Regeln als die additive Farbmischung von Licht. Je mehr Tinte man verwendet, desto dunkler wird das Ergebnis, weil man ja eigentlich einen Filter vor das passiv reflektierende Papier aufbringt, daher spricht man hier von subtraktiver Farbmischung. Das CMY-Modell dazu ist das Komplement des RGB-Raumes. Für einfache Anwendungen gilt daher: [C,M,Y] = [1,1,1] – [R,G,B]

Vielfach kommt einem auch das CMYK-Modell unter. K steht dabei für Key, das entspricht der Farbe Schwarz. Beim Druck werden hierbei alle Grauanteile mit schwarzer Farbe extra gedruckt statt sie als Mischung gleicher Anteile von Cyan, Magenta und Yellow teurer und schlechter zu erzeugen.



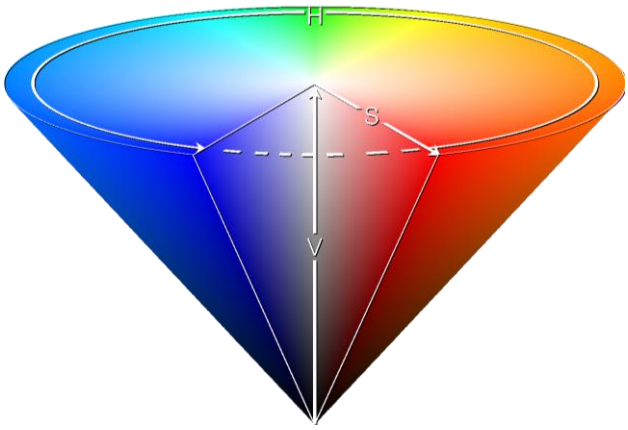
- Primärfarben: Cyan-Magenta-Gelb (yellow)
- Subtraktives Farbmodell (z.B. für Kopiergeräte)
- Gelb kann dargestellt werden in dem man Grün und Blau mischt
- Komplementär-Modell zu RGB: (C,M,Y) = (1,1,1) – (R,G,B)
- Bei Druckern wird oft zusätzlich noch schwarz als 4 Farbe verwendet, da die Mischung aus C+M+Y oft nur ein starkes Dunkelgrau ergibt



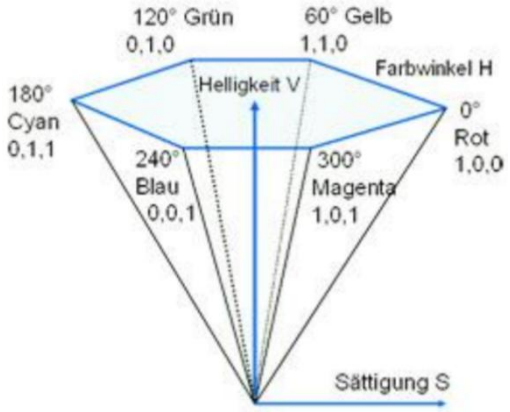
95) Was wissen Sie zum HSV-Farbsystem? (S. 724)

Neben den für Geräte sinnvollen Farbräumen gibt es noch Beschreibungen der Farben in einer Weise, die dem menschlichen Benutzer entgegen kommt. Wir können nur sehr schwer und mit viel Übung eine Zielfarbe aus den Komponenten R, G, B oder C, M, Y beschreiben. Unsere üblichen Beschreibungen von Farben setzen sich aus Qualitäten wie einem Farbwort, einer Helligkeit und einer Farbreinheit zusammen. Daher werden für das User-Interface zur Farbdefinition solche Farbsysteme verwendet, die in diesen Dimensionen funktionieren. Dazu gehören HLS, HSV, Munsell, RAL, NCS, Coloroid und einige andere.

HSV steht für Hue, Saturation und Value. Hue heißt Buntton oder Farbton, bezeichnet die Farbe entlang eines Farbkreises, der von Rot über Orange, Gelb, Grün, Cyan, Blau, Violet, Magenta wieder ins Rot geht. Wenn man den RGB-Würfel genau in Richtung seiner Grauachse anschaut, so sieht man diesen Farbkreis als Grenze des entstehenden Sechseckes. Saturation heißt Sättigung und gibt an wie rein eine Farbe ist, wie stark sie sich also von Grau unterscheidet. Value heißt Wert und gibt so etwas wie die Helligkeit der Farbe an.



Je dunkler nun eine Farbe ist, desto weniger Abstufungen der Sättigung gibt es. Dadurch lassen sich alle Farben in einer Pyramide darstellen, deren Spitze schwarz ist und deren Grundfläche das Farb-Sechseck ist. Die Farbe wird in Grad entlang der Basiskante angegeben (Rot=0°, Grün=120°, Blau=240°), die Sättigung in Prozent des Abstandes von der Pyramidenachse und die Helligkeit als Prozent des Abstandes der Grundfläche von der Spitze. Ein mittelhelles gesättigtes Gelb hat damit den HSV-Wert (60, 1, 0.5).



- Intuitivere Spezifikation von Farbe
- Stellt man den Würfel des RGB-Modells auf die Diagonale zwischen schwarz (unten) und weiß (oben), so sieht man von oben ein 6-Eck mit den G-Y-R-M-B-C
- Farbkomponenten: hue (Farbton), saturation (Sättigung) und value (Helligkeit)
- Blickt man von der Seite auf den Querschnitt der Farbpyramide, so sieht man unten an der Spitze die Farbe Schwarz, oben in der Mitte die Farbe Weiß, und ganz rechts oben den Ausgangspunkt mit S=1 und V=1.

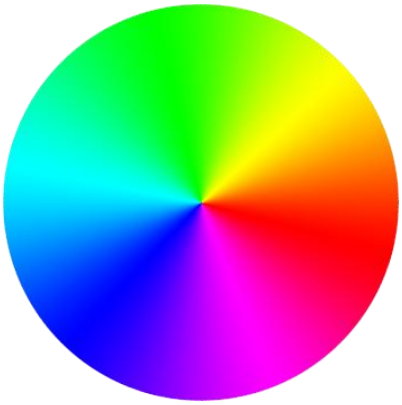
Farbdefinition

1. Spektralfarbe auswählen, S=1, V=1
2. schwarze Pigmente hinzufügen ⇒ Helligkeit verringern (weiter runter im Kegel)
3. weiße Pigmente hinzufügen ⇒ Sättigung verringern (weiter in die Mitte im Kegel)

Die RGB–Werte und die CYM-Werte sind jeweils 60° von einander entfernt, will man nun die Komplementärfarbe zu einer Farbe wissen, so nimmt die 180° entfernte Farbe.

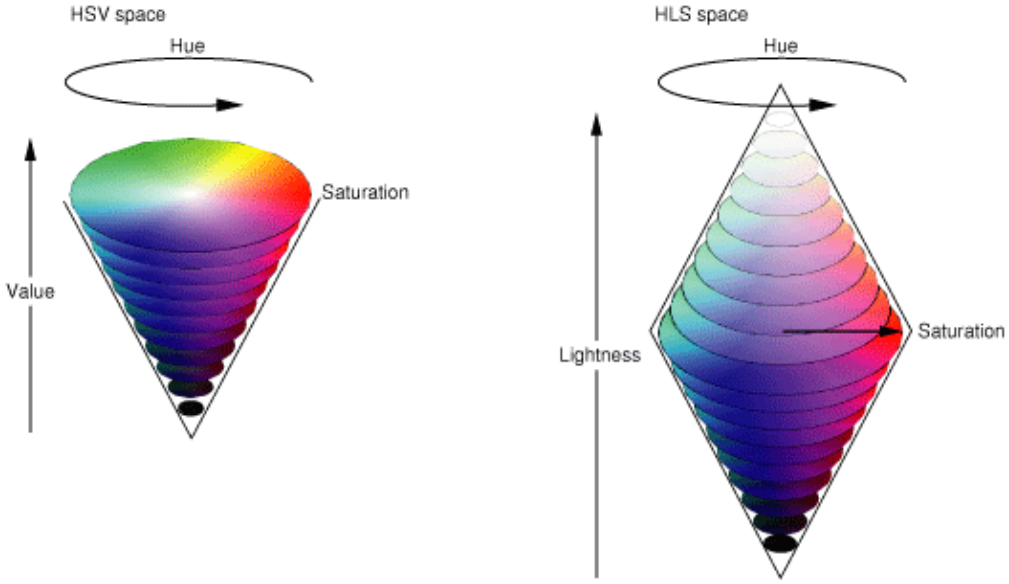
96) Was wissen Sie zum HLS-Farbsystem? (S. 728)

Ganz ähnlich wie das HSV-Farbsystem funktioniert das HLS-System (auch HSL), bei dem H=Hue, L=Lightness oder Luminance, S=Saturation heißen. Die Form des Modells ist jedoch diesmal ein Doppelkegel, der oben an der Spitze weiß ist und unten schwarz. Der Hintergrund ist die Annahme, dass Weiß viel heller ist als jede reine Farbe.



- Farbkomponenten: hue, lightness (Helligkeit) und saturation
- Abbildung des Farbraumes mittels eines Doppel-Kegels (siehe unten rechtes Bild)
- Der H-Wert entspricht auch hier einer Spektralfarbe. H=0° entspricht in diesem Modell blau.
- Die Primärfarben liegen bei L=0,5. Erhöht bzw. vermindert sich der L-Wert so hat man entweder mehr weiß oder schwarz in der Farbe.

Vergleich HSV und HLS: Quelle



Kapitel 15 – Graphics File Formats (S. 768)

97) Welche Anforderungen bringen graphische Daten an die verwendeten Datenstrukturen? (S. ???) Quelle

- Namensgebung/ Namensänderung
- Kopieren und einsetzen von Objekten
- Transformieren von Objekten
- Ändern der Eigenschafen

98) Was sind objektorientierte Graphikdateiformate und welche kennen sie? (S. ???) Quelle

Bei objektorientierten Dateien werden die Bildinformationen auf Objekte abgebildet und diese Objekte werden mit den zugehörigen Informationen, so genannten Attributen gespeichert.

Die Vorteile bei objektorientierten Graphikdateien sind die Möglichkeit des numerischen Arbeitens, insbesondere in projektspezifischen Programmen wie CAD (Computer Aided Design) oder GIS (Geographical Information System), die Skalierbarkeit, die Gesamtheit der Objekte und die Möglichkeit der Trennung von Objekt und den zugehörigen Attributen. Nachteile bestehen jedoch in der nur bedingten Editierbarkeit und die Darstellung von natürlich wirkenden Effekten ist sehr schwierig.

Wird z.B. in Macromedia Freehand, vielen Konstruktions und CAD Programmen verwendet.

EPS - Encapsulated Postscript

Dieses Graphikdateiformat gehört zur Gruppe der objektorientierten Graphikformate, jedoch lassen sich auch pixelorientierte Elemente innerhalb der Datei speichern. Vorteil dieser Art der Speicherung von Graphiken ist die verlustfreie Skalierbarkeit der Graphik. Nachteilig ist, dass nur wenige Programme EPS-Graphiken anzeigen können.

99) Was sind vektororientierte Graphikdateiformate und welche kennen Sie? (S. ???)

Eine Vektorgrafik ist eine Computergrafik, die aus grafischen Primitiven wie Linien, Kreisen, Polygonen oder allgemeinen Kurven (Splines) zusammengesetzt ist. Meist sind mit Vektorgrafiken Darstellungen gemeint, deren Primitiven sich zweidimensional in der Ebene beschreiben lassen. Eine Bildbeschreibung, die sich auf dreidimensionale Primitiven stützt, wird eher 3D-Modell oder Szene genannt.

Um beispielsweise das Bild eines Kreises zu speichern, benötigt eine Vektorgrafik mindestens zwei Werte: die Lage des Kreismittelpunkts und den Kreisdurchmesser. Neben der Form und Position der Primitiven werden eventuell auch die Farbe, Strichstärke, diverse Füllmuster und weitere, das Aussehen bestimmende Daten, angegeben.

Die Stärke von Vektorgrafiken liegt bei Darstellungen, die als Zusammenstellung von grafischen Primitiven befriedigend beschrieben werden können, zum Beispiel Diagramme oder Firmenlogos. Sie sind nicht geeignet für gescannte Bilder und Digitalfotos, die naturgemäß als Rastergrafik erfasst werden und nicht verlustfrei umgewandelt werden können. Ebenfalls an die Grenzen stoßen Vektorformate bei komplexen gerenderten Bildern, die ebenfalls direkt als Rastergrafik berechnet werden. Allerdings spezialisieren sich mehr und mehr Firmen auf die Vektorisierung von Rastergrafiken. Dies ist vor allem von Interesse für großflächige Bildwerbung, Fahrzeugbeschriftung oder wenn die Vektorisierung als grafischer Effekt genutzt wird.

Grafikanwendungen

Zur Erstellung von Illustrationen, insbesondere für die Erstellung von Logos, können vektorbasierte Zeichenprogramme verwendet werden. Für technische Zeichnungen finden CAD-Programme Verwendung; hier wird meist das Drawing Interchange Format (DXF) zur Speicherung verwendet. Die von 3D-Modellierungswerkzeugen erzeugten 3D-Szenen können auch als Vektorgrafiken betrachtet werden.

Seitenbeschreibungssprachen

Vektorgrafiken erlauben es, Dokumente unabhängig von der Auflösung des Ausgabegeräts zu beschreiben. Mit Hilfe einer vektorgrafikfähigen Seitenbeschreibungssprache wie PostScript oder dem daraus hervorgegangenen Portable Document Format (PDF) können Dokumente, im Gegensatz zu Rastergrafiken, mit der jeweils höchstmöglichen Auflösung auf Bildschirmen verlustfrei dargestellt oder gedruckt werden.

Computerschriften

Auf gängigen Computersystemen finden heute überwiegend so genannte Outline-Schriften Verwendung, die die Umrisse jedes Zeichens als Vektorgrafik beschreiben. Wichtige Formate sind TrueType, PostScript und OpenType.

Internet

Im World Wide Web liegen Vektorgrafiken meist im offenen Format SVG oder als proprietäre SWF-Dateien (Adobe Flash) vor.

Geoinformationssysteme

Bei Geoinformationssystemen (GIS) kann die Geometrie von Flurstücken und Landkarten in Form von Vektordaten gespeichert werden. Solche Vektorgrafiken lassen sich vergleichsweise einfach mit Sachdaten verknüpfen. Ein typisches GIS-Vektorformat ist das Shapefile.

100) Was wissen sie zu geographischen Informationssystemen? (S. ???) Quelle

Ein Geoinformationssystem (Kurzform GIS) oder Geographisches Informationssystem ist ein „rechnergestütztes *Informationssystem*, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können *raumbezogene Daten* digital erfasst und *redigiert*, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und grafisch präsentiert werden

Es vereint eine Datenbank und die zur Bearbeitung und Darstellung dieser Daten nützlichen Methoden (Kurzdefinition nach *Fédération Internationale des Géomètres*).

Datenmodell

Datenmodelle beschreiben, welche Daten in einem Informationssystem gespeichert werden können und wie diese Daten strukturiert sind. Es handelt sich dabei also um Informationen über reale Objekte (Personen, Flurstücke, Flüsse). Diese Objekte werden durch ausgewählte Attribute beschrieben. Beispielsweise kann man allen Flurstücken die Attribute Gemarkungsnummer, Flurstücksnummer und Nutzungsart zuordnen. Bei den genannten Eigenschaften handelt es sich um solche, die ein Objekt des Typs Flurstück eindeutig bezeichnen (Gemarkung, Flurstücksnummer) und seiner Beschaffenheit nach beschreiben. Man spricht auch von „beschreibenden Daten“, „thematischen Daten“, „Sachdaten“ oder „Attributdaten“.

Die „klassischen“ Informationssysteme beschränken sich auf die reine Verwaltung und Verarbeitung von Sachdaten. In GIS werden den Sachdaten noch die sogenannten Geometriedaten gegenübergestellt. Sie beschreiben die geographische Lage, Form, Orientierung und Größe von Objekten (siehe auch raumbezogene Objekte). Man unterscheidet Vektordaten und Rasterdaten. Vektordaten repräsentieren die Objektgeometrie anhand grafischer Primitiva (zum Beispiel Punkte, Linien, Kreisbögen). Rasterdaten beschreiben die Objektgeometrie in Form von digitalen Bildern (Kartenbildern oder Luft- bzw. Satellitenaufnahmen).

Ausgedrückt mittels Vektordaten gibt man die Geometrie eines Flurstücks also in Form der Grenzpunktkoordinaten und der Geometrie der Grenzlinien (Strecke, Kreisbogen) an. Der Auszug eines digitalen Luftbildes (meist in Form eines Orthofotos) repräsentiert die Flurstücksgeometrie in Form von Rasterdaten.

Neben den Informationen der einzelnen Objekte speichern Informationssysteme noch Beziehungen zwischen diesen Objekten. Es kann sich um sachlogische Beziehungen oder raumbezogene Beziehungen handeln oder es können beide Beziehungskategorien abbildbar sein, so wie bei einem GIS. Eine sachlogische Beziehung kann man z. B. zwischen Flurstücken und Personen herstellen: Eine „Person“ (Objekt) ist „Eigentümer“ (sachlogische Beziehung) von dem „Flurstück“ (Objekt). Die sachlogischen Beziehungen lassen sich in einem Informationssystem auswerten; Beispiel: Abfrage aller Flurstücke einer bestimmten Person.

Raumbezogene (=topologische) Beziehungen gehen zum Beispiel Flurstücke untereinander ein: ein Flurstück (präziser: die Flurstücksfläche) „ist Nachbar“ (topologische Beziehung) eines anderen Flurstücks. Auch topologische Beziehungen lassen sich in einem GIS auswerten. Beispiele: Die Abfrage aller Nachbargrundstücke zu einem Flurstück.

GIS beherrschen die integrierte Verwaltung der Sach- und Geometriedaten sowie sachlogischer und topologischer Beziehungen. Dadurch können sich Abfragen oder Auswertungen auch auf beide Informationsarten beziehen. Beispiel: Abfrage der Eigentümerdaten (sachdatenbezogener Aspekt) zu allen Flurstücken, die zu einem ausgewählten Flurstück benachbart (topologischer Aspekt) sind und eine Fläche haben, die größer als 1000 m² (geometriebezogener Aspekt) ist.

Für die Speicherung der Sach- und Geometriedaten (vorrangig der Vektordaten) nutzten zu Beginn der GIS-Ära nur wenige GIS-Basissysteme marktgängige Datenbanksysteme (z. B. DBase oder Oracle). Eine Vielzahl von Systemen basierten auf proprietären Datenbankmanagementsystemen. Heute hat sich die Nutzung von marktgängigen relationalen bzw. objektrelationalen Datenbanksystemen für die Geodatenverwaltung durchgesetzt.

Datenstrukturmodell

Ein Datenstrukturmodell gibt an, auf welche Weise Objekte und ihre gegenseitigen Beziehungen in einem Informationssystem, hier speziell einem GIS, abgebildet werden können. Für die Speicherung der Objekteigenschaften und -beziehungen hat sich z. B. das Relationenmodell durchgesetzt. Alle Attribute gleichartiger Objekte werden in Tabellen verwaltet; gleiches gilt für die Beziehungen zwischen den Objekten.

Vektorbasierte Datenstrukturmodelle ermöglichen es, die Objektgeometrie mit Hilfe von geometrischen Primitiva (z. B. Punkte, Kreisbögen, Linien) zu beschreiben; die Grundelemente lassen sich durch geordnete oder ungeordnete Gruppierung zu höherwertigen Geometrien zusammenfassen (z. B. Linienzügen oder Flächen). Vektordaten lassen sich relativ einfach mit Sachdaten verknüpfen.

Das rasterbasierte Datenstrukturmodell kennt nur ein einziges Datenstrukturelement, nämlich das Rasterelement, je nach Rasterart auch Pixel oder „Bildpunkt“ genannt. Den Rasterelementen können zwei Eigenschaften zugeordnet werden: die geometrische und die radiometrische Auflösung. Die geometrische Auflösung gibt an, welche Länge und Breite ein Rasterelement in der Natur besitzt; die radiometrische Auflösung bezeichnet die unterscheidbaren Grauwerte je Rasterelement.

Topologie

Die Topologie bezeichnet die räumliche Beziehung von Geoobjekten zueinander (Nachbarschaftsbeziehungen). Im Gegensatz zur Geometrie, die die absolute Form und Lage im Raum betrifft, sind topologische Beziehungen zwischen Geoobjekten unabhängig von Maßen wie der Distanz. Die wichtigsten topologischen Beziehungen zwischen zwei Geoobjekten A und B nach Egenhofer sind:

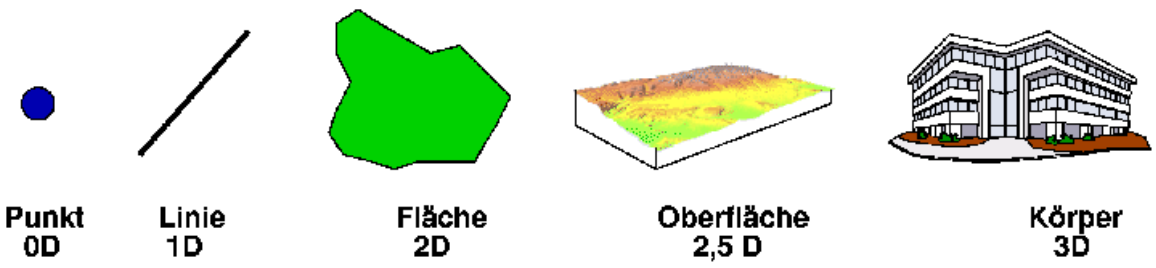
- A ist disjunkt zu B
- A liegt innerhalb B
- B liegt innerhalb A

- A überdeckt B
- B überdeckt A
- A trifft B
- A gleicht B

Dimension

Die Dimension gibt an, wie viele Koordinatenwerte einem Objekt im GIS zugeordnet sind:

- zweidimensional *(2D):* Jeder Punkt hat eine x- und eine y-Koordinate. Linienverbindungen oder Flächen, die auf die Punkte aufbauen, liegen also in einer Ebene (xy-Ebene) vor. Dies entspricht der normalen Kartendarstellung und der Datenhaltung im Kataster.
- zwei-plus-eins-dimensional *(2+1D):* Jedes Objekt hat zusätzlich eine attributive Information über die Höhe (z. B. eine Gebäudehöhe am Gebäude). Diese Form findet sich in einigen Katasterdaten wieder.
- zweieinhalbdimensional *(2,5D):* Jeder Punkt der Grundrissdarstellung hat zusätzlich zur x- und y-Koordinate eine Höhe. Damit ist die Höhe jedoch nur eine Funktion der Lage, d. h. es gibt immer nur genau einen Höhenwert zu einer Lagekoordinate (x,y). In dieser Form liegen die meisten digitalen Geländemodelle vor. Senkrechte Wände und Überhänge sind auf diese Weise nicht modellierbar.
- dreidimensional *(3D):* Alle Punkte haben x-, y- und z-Koordinate (bzw. Höhe). Linienverbindungen sind räumliche Linien, die nicht in einer Ebene liegen. Wenn Kreisbögen als Verbindungen vorkommen, werden diese streng genommen Ellipsenabschnitte, die in einer geneigten Ebene liegen; oder sie müssen durch Linienzüge mit entsprechend kurzen Segmenten angenähert werden. Flächenobjekte sind nur dann ebene Flächen, wenn sie durch genau 3 Punkte begrenzt werden, ansonsten sind es gekrümmte Raumflächen.
- vierdimensional *(4D):* Zusätzlich zu den 3 Koordinaten im Raum wird eine vierte Information mitgeführt, die sich aus dem zeitlichen Ablauf ergibt. Das wird z. B. durch Verwendung eines Zeitstempels für jedes Objekt ermöglicht. Damit kann abgefragt werden, zu welchem Zeitpunkt ein Objekt existiert hat oder nicht. Aus diesen Daten können dann Darstellungen der Vergangenheit kreiert werden (z. B.: Wie sah das Ortsbild am 15. Februar 2002 aus, bevor der Neubau errichtet wurde); auch zeitabhängige Animationen können erzeugt werden (z. B. der Fortschritt des Kohleabbaus in einem Bergwerk).



101) Was ist IGES? (S. ???) Quelle

Die Initial Graphics Exchange Specification (IGES) definiert ein neutrales, herstellerunabhängiges Datenformat, welches dem digitalen Austausch von Informationen zwischen Computer Aided Design-

Programmen (CAD) dient. Die Anwendung reicht von traditionellen, zweidimensionalen Zeichnungen bis hin zu dreidimensionalen Modellen für Simulationen oder Fertigung.

Die Daten werden in einzelnen Einheiten, den sogenannten Entities gespeichert. Diese können in einer beliebigen Hierarchie angeordnet werden. Die Primitive einer Zeichnung oder eines Modells (z. B. Linien, Kurven, Flächen) werden durch verschiedene Entity-Typen repräsentiert. Seit der IGES-Spezifikation Version 5 sind auch Körper vorgesehen. D. h. ein geschlossener Flächenverband hat zusätzlich noch die Information „innen“ und „außen“. Diese Funktionalität von IGES wurde aber von den wenigsten CAD-Systemherstellern implementiert. Um diese Information auszutauschen sollte das neuere Datenaustauschformat Standard for the Exchange of Product Model Data (STEP) verwendet werden.

Laut Spezifikation können die Informationen textbasiert (ASCII) sowie binär gespeichert werden. Das letztere Format wird jedoch offiziell nicht mehr unterstützt. Dies führt dazu, dass der benötigte Speicher für eine IGES-Datei leicht mehrere Megabyte betragen kann. Die allgemein anerkannte Dateiendung für IGES-Dateien ist .igs.

Der Verband der Automobilindustrie hat mit dem Ziel der einfacheren Implementierung verschiedene Untermengen des IGES-Standards definiert und zusammen mit weiteren Anforderungen unter der Bezeichnung **VDA-IS** veröffentlicht:

Bemaßung

B2: + Ebenen, Zeichnungen mit Ansichten, ...

B1: Texte, Pfeile, diverse Bemaßungen, ...

Rational B-Spline

AF2: + Freiformkurven und Flächen (Rationale Form),...

AF1: Freiformkurven und Flächen (Polynominale Form), ...

Geometrie

G3: + Einfache Flächen, parametrische Spline-Flächen, ...

G2: + Sonstige Daten

G1: Kurven, Kreise, Linien, Punkte, ...

Stand der Dinge

Inzwischen sind nur noch IGES-Schnittstellen mit dem kompletten Umfang implementiert. VDA-IS hat somit seine ursprüngliche Bedeutung beim Datenaustausch verloren.

102) Was versteht man unter RLE-Codierung? (S. 772) Quelle

Die Lauflängenkodierung (engl. Run-length encoding, kurz RLE) ist ein sehr einfacher verlustfreier Kompressionsalgorithmus für digitale Daten. Sie ist besonders gut geeignet, Wiederholungen oder Sequenzen von gleichen Werten verkürzt darzustellen. Liegt eine Wiederholung vor, wird die Anzahl der Wiederholungen sowie der wiederholte Wert gespeichert.

Um den Beginn einer Wiederholung zu kennzeichnen, werden sogenannte *Marker-Bytes* eingesetzt. Das sind Bytes, die nicht im Datenstrom vorkommen. Der *Offset* ist die Mindestwiederholrate, ab der kodiert wird. Bei einem Offset von 4 wird ab einer Wiederholung von 4 lauflängenkodiert. Dabei ergibt sich der Wert folgendermaßen: *AnzahlderWiederholungen* − *Offset* = *Wert*. FF FF FF FF FF wird also zu AA 1 FF.

Beispiel

Als Beispiel sei ein Schwarz-Weiß-Bildschirm angenommen, auf dem sich schwarzer Text auf weißem Hintergrund befindet. Hier sind lange Folgen von weißen Punkten bzw. Pixeln nur selten durch einzelne schwarze Punkte unterbrochen. Bezeichnet man weiße Punkte mit "W" und schwarze mit "S", so ergibt sich z.B. in einer einzelnen Bildzeile die folgende Information:

WWWWWWWWWWSWWWWWWWWWWSSS

Nach Anwendung der Lauflängenkodierung erhält man: **10**W**1**S**10**W**3**S

Praktische Anwendung

Die unterschiedlichen Kompressionsprogramme speichern die komprimierte Datenfolge im Gegensatz zum obigen Beispiel in binärer Form ab, die teilweise sehr unterschiedlich sein kann. Einige Anwendungen speichern auch zusätzlich Folgen von mehreren Datenwerten, machen also aus `-ABC-ABC-ABC-ABC-` die gekürzte Darstellung `4[-ABC]-`.

Geeignet ist die Lauflängenkodierung also da, wo es lange Folgen des gleichen Wertes gibt. Dies ist sehr häufig bei älteren Icons oder Clip-Art-Bildern der Fall, die meist nur mit wenigen Farben gezeichnet sind.

Bekannte Dateiformate, die die Lauflängenkodierung anwenden, sind daher viele ältere Grafikformate wie beispielsweise Windows Bitmap, GEM Image, Targa oder PCX.

Unter Microsoft Windows wird die Dateiendung **.RLE** üblicherweise für RLE-komprimierte BMP-Bilder verwendet, die Dateiendung .BMP meist für unkomprimierte Bilder.

Wichtiges von CG1LU

Unter http://stud4.tuwien.ac.at/~e0402913/tu.html gibt es 2 Ausarbeitungen von mir, welche ich für die LVA CG1LU schrieb. In dieser Ausarbeitung gibt es viele Details z.B. zu Matrizen, Kreuzprodukt, Z-Puffer, Clipping, Shading, Illumination,… für Studenten die mehr Hintergrundwissen haben wollen, diese Themen interessiert bzw. die Antworten in dieser Ausarbeitung teilweiße bei einigen Fragen etwas „zu kurz“ finden.

Bei den nächsten Fragen findet man einige sehr wichtige Hintergrundinformationen zusammengefasst die sicher nicht schaden zu wissen (aber so nie gefragt wurden bei CG1Vo, nur CG1LU)!

103) Welche Sichtbarkeitsverfahren arbeiten im Objektraum und welche im Bildraum?

Objektraum-Verfahren

Backface Detection, Depth Sorting, Octree-Methode

Bildraum-Verfahren

Z-Puffer, Scanline-Methode, Area Subdivision, Ray Casting

104) Warum links und rechts Multiplikationen? Was ist der Unterschied?

Matrizenmultiplikation sind nicht kommutativ und somit gilt nicht AxB = BxA. Somit kommt man auch bei beiden Berechnungen auf unterschiedliche Ergebnisse!

- Die links Multiplikation wird mit NeueMatrix * BasisMatrix berechnet
- Die rechts Multiplikation wird mit BasisMatrix * NeueMatrix berechnet

Der Unterschied liegt darin, dass bei Linksmultiplikation die Rotation sich auf das Weltkoordinatensystem auswirkt und bei Rechtsmultiplikation. auf die Objektkoordinaten.

Infos

Über die Ausarbeitung

Die Fragen stammen aus dem Informatik Forum, von Kollegen die die Prüfung machten und MTB. Falls jemand Angaben / Fragen hat die hier nicht zu finden sind, wäre es sehr nett sie mir zukommen zu lassen damit ich es hier ergänze/ hinzufüge!

Ich habe die Ausarbeitung so gut es geht gemacht, aber trotzdem können sich Fehler einschleichen! Falls man welche findet, bitte per E- Mail oder PM an mich weiter leiten damit ich sie ausbessere! Bei rot geschriebenen (meistens auch durch ??? markiert) Sätzen bin ich mir nicht ganz sicher ob sie so stimmen und deswegen würde ich mich sehr über Feedback (ob es so stimmt oder nicht) freuen ☺.

Zusätzliche Informationen

Version:	0.6.9
LVA Webseite:	http://www.cg.tuwien.ac.at/courses/CG/VO.html
Neuste Version:	http://stud4.tuwien.ac.at/~e0402913/uni.html
Ausarbeitung:	Martin Tintel (mtintel)
Ausarbeitungen auf die aufgebaut wird:	Maciej Makowski Andreas Bauer Textblätter von Werner Purgathofer
Weiterführendes:	Vorlesungsfolien: http://www.cg.tuwien.ac.at/courses/CG/slides.html Textblätter: http://www.cg.tuwien.ac.at/courses/CG/textblaetter.html CG1LU Ausarbeitungen: http://stud4.tuwien.ac.at/~e0402913/tu.html