2. Graphische Datenverarbeitung - eine Disziplin der Informatik

Der Entwurf graphischer Benutzeroberflächen, die Visualisierung nichtgeometrischer Daten, die photorealistische Darstellung einer Szene, die Veränderung eines Bildes, die Extraktion von Information aus einem Bild, die interaktive Erstellung eines geometrischen Modells - dies alles sind Beispiele für Aufgaben, wie sie im 1. Kapitel dieses Skript aufgeführt wurden (Beispiele hierzu Abb. 2.1).

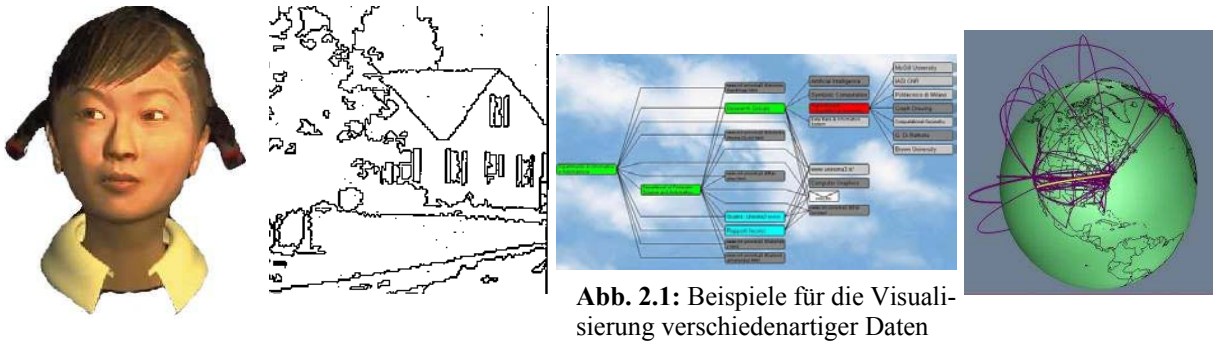


Abb. 2.1: Beispiele für die Visualisierung verschiedenartiger Daten

Die Bearbeitung einer breiten Palette von Aufgabenstellungen verlangt daher die Verwendung der unterschiedlichsten ***Datenstrukturen*** und ***Lösungstechniken***. Dieses Kapitel soll einen ersten Eindruck darüber vermitteln, mit welchen typischen Problemen und Besonderheiten beim Umgang mit graphischen Daten zu rechnen ist. Weiterhin soll gezeigt werden, dass Datenstrukturen wie Arrays, Tabellen, Listen, Bäume und Graphen (s. Abb. 2.2) in der Graphischen Datenverarbeitung eine breite Anwendung haben. Typische Probleme der Informatik und entsprechende Lösungstechniken findet man in der GDV wieder. Dennoch gibt es Besonderheiten, Schwierigkeiten und Herausforderungen, die typisch sind für dieses Gebiet, dem in den Geburtsjahren der Informatik von vielen Fachleuten keine Chance für eine allgemeine Verbreitung eingeräumt wurde...

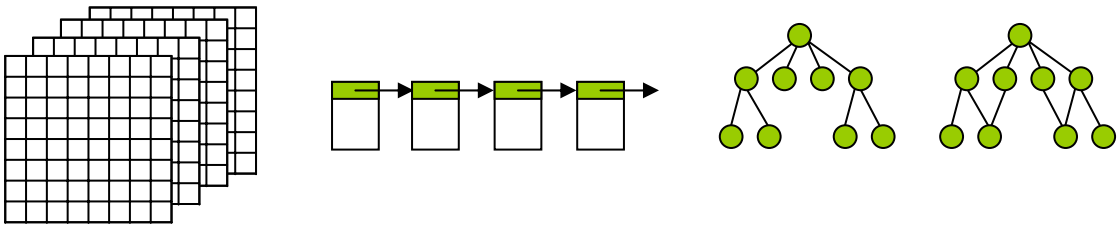


Abb. 2.2: Beispiele für in der GDV häufig verwendete Datenstrukturen

2.1. Graphische Daten und einige ihrer Besonderheiten

In der "Steinzeit" der Informatik waren Programmierer und Anwender sowohl bei der Programmerstellung und was die Laufzeiten von Programmen betraf daran gewöhnt sich in Geduld zu üben. Im Gegensatz hierzu war in bestimmten Anwendungen der Graphischen Datenverarbeitung schon frühzeitig eine Programmausführung in ***Echtzeit*** gefordert. Dies führte zur Entwicklung von Vektorgraphikdisplays (s. Kapitel 1). Beispiele derartiger Anwendungen sind die interaktive Konstruktion am Bildschirm (s. Sketchpad 1963 - Interaktion mit dem Lightpen) als auch Simulationen (z.B. Flugsimulatoren mit 3D - Hochleistungsdisplays). Die Forderung nach hohen Geschwindigkeiten machte die Graphik damals wie heute (Spieleindustrie, Virtual Reality (VR) ...) zum Motor technischer Innovationen. Beinahe nirgendwo gilt das Motto "Der Appetit kommt mit dem Essen" so sehr wie in der Graphischen Datenverarbeitung. In anderen Worten: Je schneller die Hardware und die Algorithmen, umso höher die Anforderungen an ***Geschwindigkeit***, ***Realitätstreue*** und das zu verarbeitende ***Datenvolumen*** (Bildauflösung, Echtfarben, Texturen, 3D).

Zu Zeiten der Verwendung von Vektordisplays musste sich eine Anwendung mit der Darstellung von Strichzeichnungen (aufgebaut aus einzelnen Linien: ***1D - Objekte*** = Kurven im Raum) begnügen. Die verwendeten "Farben" waren "***Schwarz***" und "***Weiß***".

Mit der Verwendung von Rasterdisplays wurde es möglich in Echtzeit Flächen (***2D - Objekte*** = Raumflächen) auszufüllen, da die Darstellungszeit von der Komplexität des Bildinhalts unabhängig wurde (s. Kapitel 5). Mit dem Preisverfall von Speicherelementen kam auch mehr und mehr ***Farbe*** ins Spiel (2^4 bit =16 Farben, 2^8 bit = 256 Farben, 2^{24} bit =16 Millionen Farben, 2^{32} bit wg. zusätzlicher Transparenz).

Schnellere Hardware macht heute die Echtzeitverarbeitung echter Volumenmodelle (***3D - Objekte*** = Körper) möglich (s. Kapitel 12)

Abb. 2.3: Treppeneffekte bei Linie, Kreis und Polygon

Wer mit graphischen Daten umgeht, kennt die unerwünschten sogenannten ***Treppeneffekte***. Diese treten z.B. bei schräg verlaufenden Linien oder Kanten auf. Sie machen sich besonders stark bemerkbar, wenn die Bildauflösung gering ist (Abb. 2.3). Dieser Effekt ist einer von vielen gleichartigen, die unter dem Begriff ***Aliasing*** bekannt sind. Die Ursache dafür ist, dass die (idealen) analogen Daten für den Rechner digitalisiert werden müssen. Dies bedeutet, dass eine ideale Strecke in eine endliche Anzahl diskreter Punkte (Pixel) umgesetzt wird (Abb. 2.3 links; s.a. Kapitel 3). Mit diesem Effekt ist bei jeder Umwandlung analoger in diskrete Daten zu rechnen. In der Graphik wird er nur besonders "schön" sichtbar. Im Kapitel 13 wird das Aliasing-Problem und Maßnahmen (***Antialiasing***) zu seiner Behebung näher besprochen.

In geographischen Informationssystemen (GIS) werden Daten in 2- oder 3-dimensionalen Koordinatensystemen abgelegt. Beispiel: Die Lage von Ortschaften einer bestimmten Region wird durch (x,y)-Koordinaten beschrieben. Eine Standardabfrage an das System könnte lauten: Welche Ortschaften befinden sich in der unmittelbaren Umgebung des Ortes Feldmoching? Wie müssen derartige Daten organisiert sein, damit ähnliche Abfragen effizient bearbeitet werden können? Es wäre kaum einsichtig, wenn eine der Koordinaten x oder y bevorzugt zur Sortierung der Daten verwendet werden würde. Graphische Datenverarbeitung, CAD, GIS, ... verfügen über Ansätze ***mehrdimensionale Daten*** effizient zu handhaben.

2.2. *Pixel, Vektoren und Hierarchien*

Die erste und zuweilen einzige Assoziation zum Fachgebiet der Graphischen Datenverarbeitung ist oft das auf dem Bildschirm dargestellte Bild. Ist dieses Bild ein sog. ***synthetisches Bild***, also kein Photo, das eingescannt und anschließend ausgegeben wird, muss es zunächst im Rechner erzeugt werden (s. Kapitel 4) . Häufig liegen die Bestandteile des Bildes (oder der 3D-Szene) dann als eine Menge graphische Objekte vor. Deren Eigenschaften oder ***Attribute*** (Form, Farbe, Größe etc.) sind in entsprechenden Datenstrukturen abgelegt (s. Kapitel 2.2.2). Das auf dem Bildschirm dargestellte Bild ist das Ergebnis eines Prozesses, der eine derartige abstrakte Beschreibung (=Repräsentation) in eine sichtbare Darstellung überführt.

2.2.1. Bilddaten und Rasterrepräsentation

Ein auf dem Bildschirm dargestelltes Bild besteht aus einzelnen Bildpunkten (***Pixel***). Da der Bildschirminhalt pro Sekunde 70-120 mal neu aufgebaut werden muss (***Bildregenerierung***, ***Bildwiederholung***), wird die Information über jedes Pixel (Farbe Helligkeit, Transparenz) in einem Puffer (***Bildspeicher*** = ***frame buffer*** s. Kapitel 5) abgelegt. Den frame buffer stellt man sich zweckmäßigerweise als 2-dimensionales Feld (array) von Pixeldaten vor. Die Abspeicherung von Bilddaten in 2- oder mehrdimensionalen Feldern ist allerdings nicht nur im frame buffer vorzufinden. Vor allem in 2 Anwendungsfällen werden Bilder rechnerintern matrixförmig repräsentiert:

1. Die Bilddaten selbst sollen verändert werden, also nicht die graphischen Objekte aus denen die Bilder erzeugt werden. Dies ist typischerweise bei der sog. Bildbearbeitung der Fall (s. Kapitel 6).

2. Aus den Bilddaten sollen Informationen gewonnen werden, die zur Gewinnung einer Bildbeschreibung dienen sollen. Dies ist ein Problem der Bildverarbeitung (s. Kapitel 1).

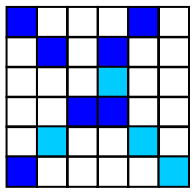


Abb. 2.4: Rasterdarstellung zweier sich kreuzender Linien

Da eine Bildmatrix aus Pixeln (wie ein Mosaik) kein Kontinuum darstellt, spricht man von einer ***Rasterrepräsentation*** der Bilddaten. Man spricht von ***Rastertechniken***, wenn es sich um den Umgang mit Rasterdaten (Veränderung, Speicherung, Ein- und Ausgabe) handelt. Zur Veranschaulichung soll Abb. 2.4 dienen. Dies könnte ein Ausschnitt einer eingescannnten Skizze sein. Die einzelnen dunkel gefärbten Pixel besitzen unterschiedliche Grauwerte. Eine sehr einfache Bildbearbeitungsaufgabe könnte darin bestehen den dunklen Pixeln einen einheitlichen Grauwert zuzuweisen. Diese Aufgabe kann z.B. mit einem "einfachen" ***Malprogramm*** bewältigt werden. Eine anspruchsvollere Aufgabe der ***Bildverarbeitung*** wäre es, zu erkennen, dass hier möglicherweise 2 sich kreuzende Linien vorliegen. (***Schriftzeichenerkennung***: Buchstabe "x").

Beachte, dass in einer Rasterrepräsentation tatsächlich immer nur einzelne isolierte Bildpunkte vorliegen. Erst durch einen - möglicherweise recht aufwendigen - Verarbeitungsschritt wurden im 2. Fall graphische Objekte (Kreuzung oder Buchstabe "x") erzeugt. Diese graphischen Objekte werden im Rechner völlig anders repräsentiert (s. Kapitel 2.2.2). Im Zusammenhang mit Rasterdaten spricht man vom sog. ***Bildraum***. Algorithmen der graphischen Datenverarbeitung, die Rücksicht auf die Rasterung eines Bildes nehmen, also Ergebnisse erzeugen, die von der Auflösung eines Bildes abhängen, bezeichnet man daher als ***bildraumorientierte Verfahren***. Soll zum Beispiel in Abb. 2.4 der Schnittpunkt der beiden dargestellten Linien dadurch ermittelt werden, dass in der Bildmatrix nach einer für einen Kreuzungsbereich typischen Anordnung von Pixeln gesucht wird, so ist das Ergebnis von der Abtastgenauigkeit/Bildauflösung abhängig.

2.2.2. Graphische Daten und graphische Datenstrukturen

Der Bildraum ist zur Repräsentation graphischer Objekte nicht besonders gut geeignet. Graphische Objekte besitzen Eigenschaften und Verhaltensweisen, die einzelnen Pixeln nicht zugeordnet werden können. Eine Strecke kann durch ihren Anfangs- und Endpunkt definiert werden. In der Graphik kommen noch Eigenschaften wie z.B. Dicke und Farbe hinzu. In einer Animation kann die Strecke ihre Lage, Orientierung, Länge, Farbe, Breite, ... verändern. Man kann sich gut vorstellen, dass nicht alle diese Veränderungen im Bildraum bequem durchzuführen sind.

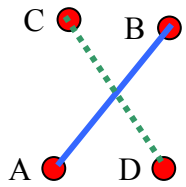


Abb. 2.5: Zwei sich kreuzende Strecken

Die Repräsentation der graphischen Objekte erfolgt im sog. ***Objektraum***. Das Beispiel der beiden sich kreuzenden Linien soll erneut aufgegriffen werden. Gegeben seien 2 Strecken AB und CD die sich schneiden (s. Abb. 2.5). Die Repräsentation der beiden Strecken erfolgt durch Angabe der Koordinaten der Punkte A,B,C und D (jeweils x,y(,z)) und eine Festlegung, dass jeweils 2 der Punkte durch eine Linie (evt. zusätzlich Farbe, Linienart) miteinander verbunden sind. Die Koordinaten der Endpunkte werden z.B. als Gleitkommagrößen mit doppelter Genauigkeit angegeben. Steht nun ein Algorithmus zur Verfügung, der den Schnittpunkt S zweier Geraden ermitteln kann, so können die Koordinaten von S entsprechend genau und völlig unabhängig von der Auflösung eines Darstellungsgerätes berechnet werden. Der Algorithmus nimmt die Berechnung im Objektraum vor. Beachte, dass diese Berechnung unabhängig davon durchgeführt werden kann, ob die beiden Strecken sich tatsächlich und nicht nur in ihren Verlängerungen schneiden. Übrigens ist der Begriff "Graphikobjekt" wesentlich älter als das Paradigma der Objektorientierung in der Informatik. Die Objektorientierung hat sich in der Graphik allerdings besonders frühzeitig durchgesetzt.

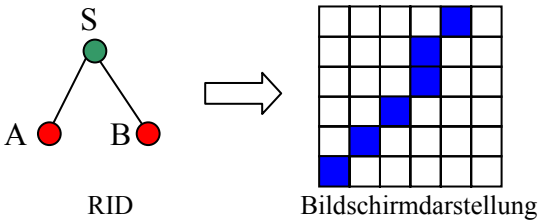


Abb. 2.6: Vektor-Rasterkonvertierung einer Strecke

Bildraum und Objektraum sind in der Graphik von elementarer Bedeutung. Der Übergang von jeweils einem der beiden Räume in den anderen gehört zu den grundlegenden Aufgabenstellungen der graphischen Datenverarbeitung. Abbildung 2.6 zeigt links die Repräsentation einer Strecke in einer einstufigen Hierarchie. A und B seien 2 durch ihre Koordinaten definierten Punkte. S ist die Strecke, die durch ihre 2 Endpunkte definiert wird. Diese Hierarchie zeigt einen besonders einfachen Fall einer sog. ***rechnerinternen Darstellung*** (***RID***= ***geometrisches Modell***) (s. Kapitel **8**). Soll diese Strecke auf einem Bildschirm ausgegeben werden, so muss die RID in eine Rasterdarstellung konvertiert werden. Der Vorgang wird als ***Vektor-Rasterkonvertierung*** (***scan conversion***) bezeichnet (s. Kapitel **3**). Die Bezeichnung rührt daher, dass die Repräsentation eines Graphikobjekts im Objektraum auch ***Vektordarstellung*** genannt wird. Frühe Graphikdisplays waren Vektorausgabegeräte (Vektordisplays). Zur Ausgabe der RID war eine Vektor-Rasterkonvertierung nicht erforderlich.

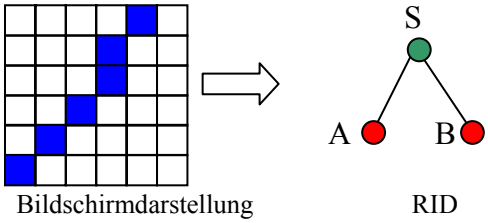


Abb. 2.7: Raster-Vektorkonvertierung

Der umgekehrte Vorgang, eine ***Raster-Vektorkonvertierung*** (***Vektorisierung***) ist ebenfalls gebräuchlich. Er ist dann erforderlich, wenn Bilddaten vorliegen und deren Bedeutung ermittelt werden soll. Die Ergebnisse können Graphikobjekte (Linien, Flächen, ... s. Abb.2.7) oder auch Schriftzeichen sein.

Den Vorgang Pixel zu "sinnvollen" Einheiten zu gruppieren nennt man ***Segmentation*** (s. Kapitel 1). Beispiele hierfür wären die Segmentation eines Tumors in CT-Datensätzen oder auch die Ermittlung von Symbolen bei der Schriftzeichenerkennung (***OCR*** = ***Optical Character Recognition***). Die Ergebnisse einer OCR können dann z.B. direkt in ein Textdokument exportiert werden.

Die Vektor-Rasterkonvertierung ist ein Vorgang, der standardmäßig bei jeder Graphikanwendung durchgeführt wird. Hierfür sind ausgeklügelte Algorithmen entwickelt worden, die i.a. durch Hardware unterstützt werden. Die Raster-Vektorkonvertierung ist eine wesentlich anspruchsvollere Aufgabe, da es prinzipiell schwieriger ist aus Bruchstücken ein Ganzes aufzubauen als ein bestehendes Objekt in kleine Einheiten zu "zerhacken". Die Raster-Vektorkonvertierung ist vor allem eine Aufgabe der sog. ***Szenenanalyse*** und ist wegen des hohen Schwierigkeitsgrades nach wie vor Gegenstand lebhafter Forschungsaktivitäten.

Eine gerade Strecke, wie im obigen Beispiel, stellt neben einem einzelnen Punkt die einfachste Form eines graphischen Objekts dar. Um die Definition eines Bildes oder einer 3D-Szene zu erleichtern, stellen Graphiksysteme weitere einfache Bausteine (man spricht von ***graphischen Primitiven***) zur Verfügung. Diese können systematisch in punktförmige, linienhafte und flächenhafte Primitive unterteilt werden. Hierzu ein Beispiel: Gegeben seien 6 Punkte in der Reihenfolge P_0 bis P_5. OpenGL (s. Kapitel 9), ein de facto Standard für die Graphik, kann hieraus verschiedene graphische Primitive bilden (Abb. 2.8).

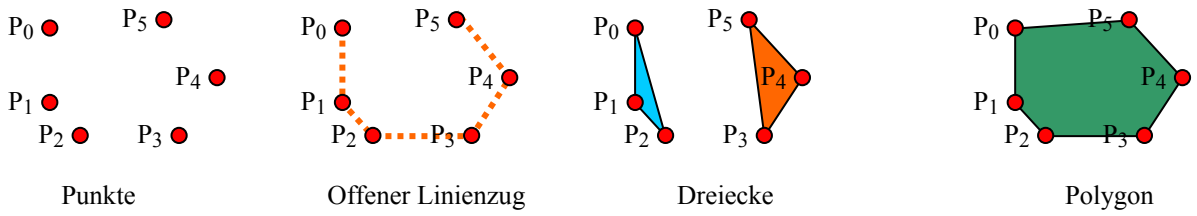
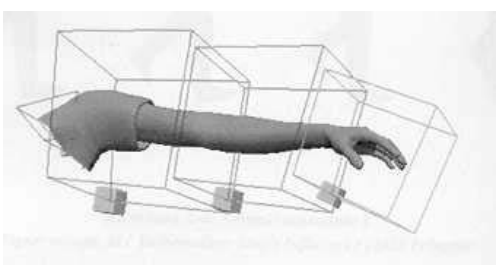


Abb. 2.8: Beispiele für graphische Primitive

Objektorientierte Klassenbibliotheken (vgl. Kapitel 9), die auf OpenGL aufsetzen, definieren weitere Bausteine (3D-Körper) durch ihre äußere Hülle. Beispiele hierfür sind Quader, Zylinder, Kegel, Kugel, ...

Die Welt der graphischen Datenverarbeitung besitzt i.a. eine hierarchische Struktur. So besteht z.B. ein Roboter aus Körper, Kopf, Beinen und Armen. Ein Arm besteht aus Ober-, Unterarm und Hand; eine Hand aus Handfläche, Daumen, Finger etc. (s. Abb. 2.9).



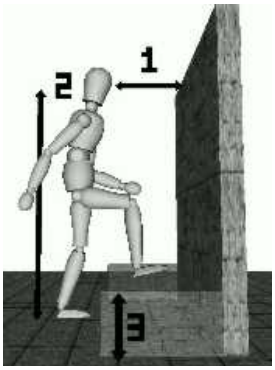


Abb. 2.9 Beispiele für hierarchische Objekte

Die rechnerinterne Repräsentation (RID) derartiger Objekte erfolgt in ***hierarchischen Datenstrukturen*** wie Bäumen, oder häufiger in gerichteten, zyklenfreien Graphen. In deren unterster Ebene stehen die geometrischen Daten für die graphischen Primitive (s. Kapitel **8**). Die Oberflächen komplexer Körper werden sehr häufig durch eine große Anzahl von Dreiecken approximiert (vgl Kap. 3.1 und 9 bis 11).

Graphische Datenstrukturen beschreiben also Bild- oder Szeneninhalte. Sie dienen dazu eine effiziente Ausgabe (***Rendering***) und Interaktionen mit den dargestellten Objekten zu ermöglichen. Daneben können auch weitere, diesen Zwecken nahestehende, Aufgaben unterstützt werden. Ein Beispiel hierfür ist die Definition sog. ***bounding boxes***. Dieses sind (z.B. quaderförmige) Umhüllende von Objekten, die z.B. dazu verwendet werden können gegenseitige Objektdurchdringungen schnell zu erkennen um sie zu verhindern (Kollisionserkennung).

Hierarchische Strukturen wie sie z.B. in VRML (Virtual Reality Modeling Language), OpenInventor oder Java3D verwendet werden, werden als ***Szenengraphen*** bezeichnet (s. Kapitel **8**). Allgemein sind graphische Datenstrukturen Bestandteil des jeweiligen Graphiksystems. Dies unterscheidet sie von den Geometriemodellen des folgenden Abschnitts.

2.2.3. Geometrische Modelle

Beim rechnergestützten Konstruieren (***CAD*** = ***Computer Aided Design***) entstehen häufig sehr komplexe Objekte (z.B. ein Motorblock, oder ein ganzes Fahrzeug). Die realitätsgetreue graphische Ausgabe der erzeugten Modelle ist nur eine Aufgabe unter anderen. So muss es z.B. möglich sein aus den eingegebenen geometrischen und weiteren Daten Eigenschaften wie Volumina, Oberflächen und Gewichte zu bestimmen. In vielen Anwendungen sind Berechnungen der statischen und dynamischen Belastbarkeit (Brücken, Tragflächen von Flugzeugen, ...) durchzuführen um ein Optimum bzgl. Sicherheit, Materialverbrauch, ... zu finden. Weiterhin müssen technische Zeichnungen erstellbar und die Erzeugung von Stücklisten möglich sein. Hierzu sind Modellierungsschemata erforderlich, die sich von denen eines Szenengraphen stark unterscheiden. Die hierfür heute prinzipiell verwendeten Repräsentationsschemata werden im Kapitel **12** vorgestellt.

2.2.4. Produktmodelle

CAD und CAM (Computer Aided Manufacturing) waren einst die Keimzellen der integrierten Fertigung in einem Betrieb. Heute kann die Integration wesentlich weiter reichen und außer der Qualitätskontrolle (CAQ), Lagerhaltung und Montage auch die Produktionsplanung und -steuerung umfassen. Man spricht dann von ***CIM*** oder ***Computer Integrated Manufacturing*** (s. Betriebsinformatik). Das CAD-Modell ist das Kernstück einer derartigen Integration. Den Modelldaten müssen allerdings weitere Informationen hinzugefügt werden. Beispiele hierfür: Herstellungsspezifische Informationen wie Werkzeuge, Werkzeugwege (für CAM), Kosten, Namen des Konstrukteurs, Zulieferer etc. In diesem Fall spricht man von Produktdaten. Die Handhabung dieser Daten ist nicht mehr Gegenstand der GDV.

2.3. Das Profil des "Graphikers"

Gemeint ist mit dem "Graphiker" nicht der künstlerisch Schaffende, sondern jemand, der sich sein Arbeitsgebiet im Umfeld der Graphischen Datenverarbeitung sucht. Über welche Grundkenntnisse und Interessen sollte er verfügen? Die Frage ist nicht eindeutig zu beantworten, da das Gebiet ein weites Spektrum umfasst, und es nur wenige Anwendungsgebiete der Informatik gibt in denen Techniken der graphischen Datenverarbeitung nicht sinnvoll eingesetzt werden können. Reduziert man die Fragestellung auf den Spezialisten, der sich auf dem Gebiet der "Kern-GDV" betätigen möchte, so verschärft sich das Anforderungsprofil. Ein kurzer Blick in die Literatur zeigt, dass die GDV aufs Engste mit der Entwicklung von Programmen und Algorithmen verknüpft ist. Die Programmiersprachen C, C++ und neuerdings Java, spielten von Anfang an eine herausragende Rolle. Ebenso hat sich die Graphik sehr schnell des objektorientierten Paradigmas bemächtigt, da dieses ihren Problemstellungen am besten gerecht zu werden scheint. Je nach Interessenlage (
-Entwicklung, VR-Thematik, Spieleprogrammierer, ...) sind zum Teil intime Kenntnisse der entsprechenden Hardware unabdingbar. Grundlegende Verfahren und Algorithmen können zuweilen nicht ohne solide Mathematikkenntnisse entwickelt werden. Dabei ist der souveräne Umgang mit homogenen Koordinaten (s. Kapitel **10**) eine Selbstverständlichkeit. Aber wer hat schon mal etwas von Quaternionen gehört? Zuweilen gehören auch diese zum "Geschäft". Dass der gekonnte Umgang mit (dynamischen) Datenstrukturen eine Selbstverständlichkeit ist, sollte aus dem Inhalt der vorausgegangenen Abschnitte deutlich geworden sein. Da in der Graphik alles Denkbare modelliert wird (materielle Objekte, Materialien, Farbe, Licht, Reflexionseigenschaften, ...) sollte ein "Graphiker" nicht zuletzt auch ein gewisses Interesse für elementare physikalische Gesetzmäßigkeiten mitbringen.