4. 3. Grafik-Programmierung

- es gibt wenig Standards
- auch in C sind die zur Verfügung stehenden Grafik-Befehle abhängig vom Betriebssystem und von der verwendeten C-Erweiterung
- grundsätzliche Unterscheidung: 2D-Grafik, 3D-Grafik

weitere Unterscheidung nach dem Zweck:

- Diagramme, Funktionsdarstellungen, Visualisierung von Messergebnissen
- grafische Benutzungsoberflächen (Fenster, Buttons, Slider...)
- Bildmanipulation
- Konstruktion (CAD)
- Architektur (virtuelle Gebäude, Landschaften...)
- Computerkunst
- Animation
- fotorealistische Darstellung
- Virtual Reality

hier 2 Beispiele angedeutet (überblicksartig; zur praktischen Anwendung Handbücher und Internet-Seiten konsultieren):

- Borland Graphics Library für C/C++
- VRML

4. 3. 1. Die Borland-Grafikbibliothek

- 2D-Grafik
- steht nur unter Borland C++ zur Verfügung
- nicht mehr letzter Stand der Technik (heute wird mehr mit OpenGL und anderen, stärker event-orientierten Libraries gearbeitet)
- aber leicht zu verstehen und schnell zu implementieren
- ähnliche Funktionen (mit ggf. anderer Notation) auch in anderen prozeduralen Sprachen, z.B. PostScript, wiederzufinden
- Nachteil: keine Interaktion mit der Maus vorgesehen
- für Maus-Eingabe müssen andere Libraries herangezogen werden

Beispielprogramm:

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

int main()
{
int gdriver=DETECT, gmode, errorc, l, r, o, u;
char txt[80];
/* Initialisieren der Grafik: */
initgraph(&gdriver, &gmode, "\\TC\\BGI\\");
/* Fehlermeldung bei fehlgeschlagener
   Initialisierung: */
errorc = graphresult();
if (errorc != grOk)
   {
   printf("Grafikfehler: %s\n",
      grapherrormsg(errorc));
   printf("Druecken Sie eine beliebige"
          " Taste!");
   getch(); exit(1);
   }
```

```
/* Positionieren des Grafik-Cursors: */
moveto(20, 30);

/* Zeichnen einer diagonalen Linie
   von der aktuellen Position aus: */
lineto(40, 50);

/* Setzen der Farbe für folgende
Zeichenbefehle: */
setcolor(14);       /* 14 = gelb */

/* Zeichnen einer Linie von (40; 50) bis
   (80; 90): */
line(40, 50, 80, 90);

/* Zeichnen eines einzelnen Pixels
   bei Position (100, 50) mit Farbe 1: */
putpixel(100, 50, 1);

/* Ausgabe eines Texts
   an der aktuellen Grafik-Cursor-Position: */
outtext("Marke A");

/* Textausgabe an Position (200, 70): */
outtextxy(200, 70, "Marke B");

/* Ermitteln d. aktuellen Grafikcursor-Pos.: */
sprintf(txt, "(%d, %d)", getx(), gety());
outtextxy(200, 100, txt);

/* Zeichnen eines Kreises mit
   Mittelpkt. (300, 100) und Radius 50: */
circle(300, 100, 50);

/* Zeichen eines Rechtecks mit gegebener
   ob. linker und unt. rechter Ecke in der
   Mitte des Bildschirms: */
l = getmaxx()/2 - 50;
o = getmaxy()/2 - 40;
r = getmaxx()/2 + 50;
u = getmaxy()/2 + 40;
rectangle(l, o, r, u);
```

```
/* Warten auf Tastendruck,
   um Zeichnung betrachten zu lassen: */
getch();

/* Beendigen der Grafikanzeige: */
closegraph();

return 0;
}
```

Es gibt weitere Befehle für Kreisbögen, Ellipsenbögen, Füllen von Recktecken und Polygonen, Wechseln zwischen Text- und Grafikmodus...
siehe Handbuch.

4. 3. 2. VRML

"Virtual Reality Modelling Language"

Beschreibungssprache für VR-Szenen

- eigene Beschreibungssprache
- nicht unmittelbar mit C kompatibel
- VRML-Dateien sind Textdateien $\Rightarrow$ man kann mit C-Programmen VRML-Dateien schreiben lassen
- zur Ausführung der VRML-Dateien sind Internet-Browser geeignet (Microsoft Explorer, Netscape) – aber mit Plugin (Zusatzsoftware, kostenlos)

- Spezifikation von 3D-Objekten und Szenen

Seite mit Links zu verschiedenen Download-Sites:
http://www.web3d.org/vrml/vrml.htm

VRML-Umfang:

3D-Grafik
+ Dynamik (Animation)
+ Interaktion
+ Klang
+ Internetfähigkeit

VRML-Dateien

Endung .wrl ("world")
ASCII-Datei (genauer: ab V. 2.0 UTF-8 Zeichensatz gem. ISO 10646-1:1993)

Aufbau einer VRML-Datei:

- Header (Version und Zeichensatzangabe, obligatorisch)
- Zeilenkommentare
- VRML-*Knoten*
- innerhalb der Knoten-Spezifikationen: *Felder* (= festgelegte Attributierungen von Knoten, denen Werte zugewiesen werden)
- PROTO-Statements
- ROUTE-Statements

Zuordnung von Feldangaben zu Knoten mit geschweiften Klammern { ... }

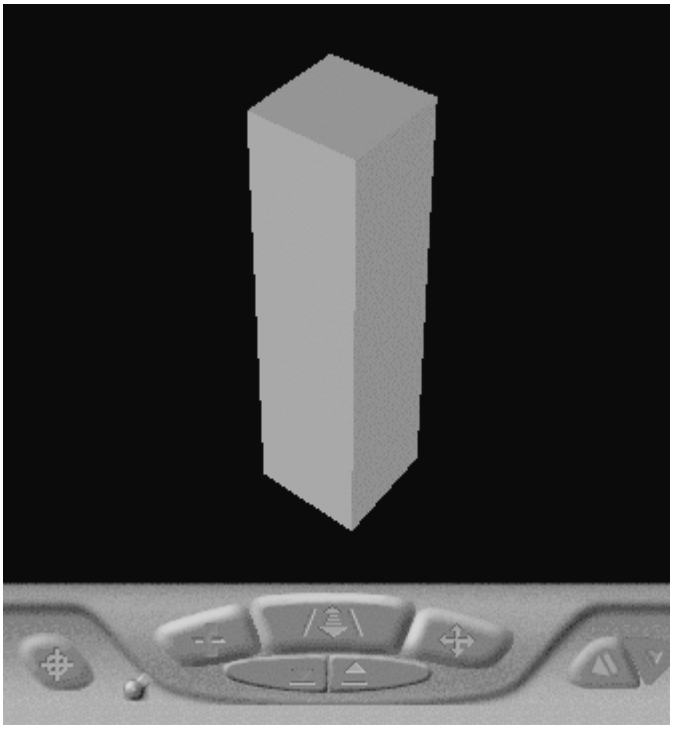
Gruppierung von Knoten durch Gruppen-Knoten, "Kinder" in eckigen Klammern [...]

Beispiel einer VRML 2.0 - Datei:

```
#VRML V2.0 utf8
Shape
   {
   geometry Box
      {
      size 1 4 1
      }
   appearance Appearance
      {
      material Material
         {
         diffuseColor 0.5 0.5 0.5
         }
      }
   }
```

enthält Seitenlängen (**`size`**) und Farbspezifikation (mittleres Grau; 3 Zahlenangaben: Rot-, Grün- und Blaukomponente).

Ergebnis:



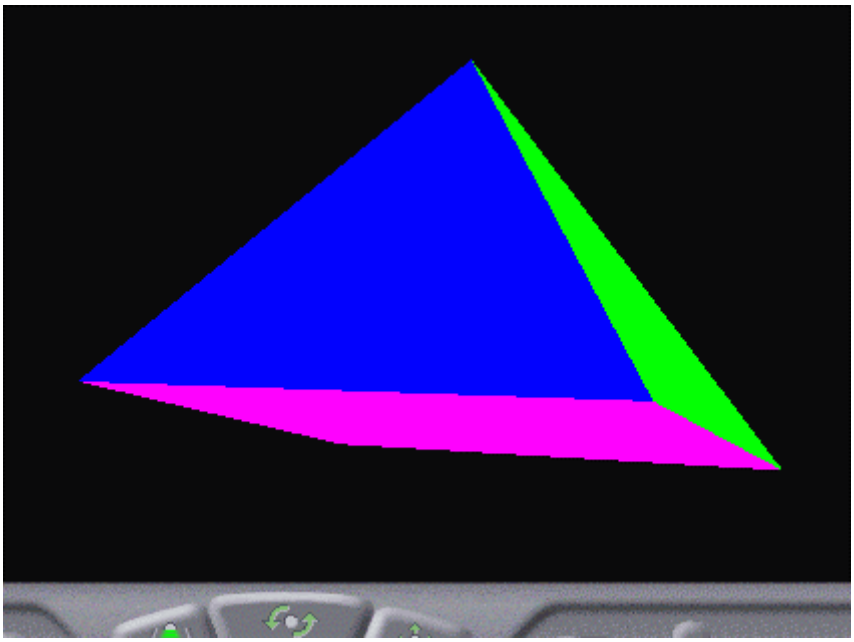
(Vordergrund = Schaltfläche des VRML-Browsers, zur virtuellen Navigation)

weiteres Beispiel:

Pyramide als Körper mit durch Punktindices definierten Facetten

```
#VRML V2.0 utf8
Shape
   {
   geometry IndexedFaceSet
      {
      colorPerVertex FALSE
      coord Coordinate
         {
         point [ -2 0 -2 # Ecke 0
                  2 0 -2 # Ecke 1
                 -2 0 2  # Ecke 2
                  2 0 2  # Ecke 3
                  0 3 0  # Ecke 4 = Spitze
               ]
         }
      coordIndex [ 4 1 0 -1,
                   4 3 1 -1,
                   4 2 3 -1,
                   4 0 2 -1,
                   2 0 1 3   # Boden
                 ]
      color Color
         {
         color [ 1 0 0,
                 0 1 0,
                 0 0 1,
                 1 1 0,
                 1 0 1 ]
         }
      }
   }
```

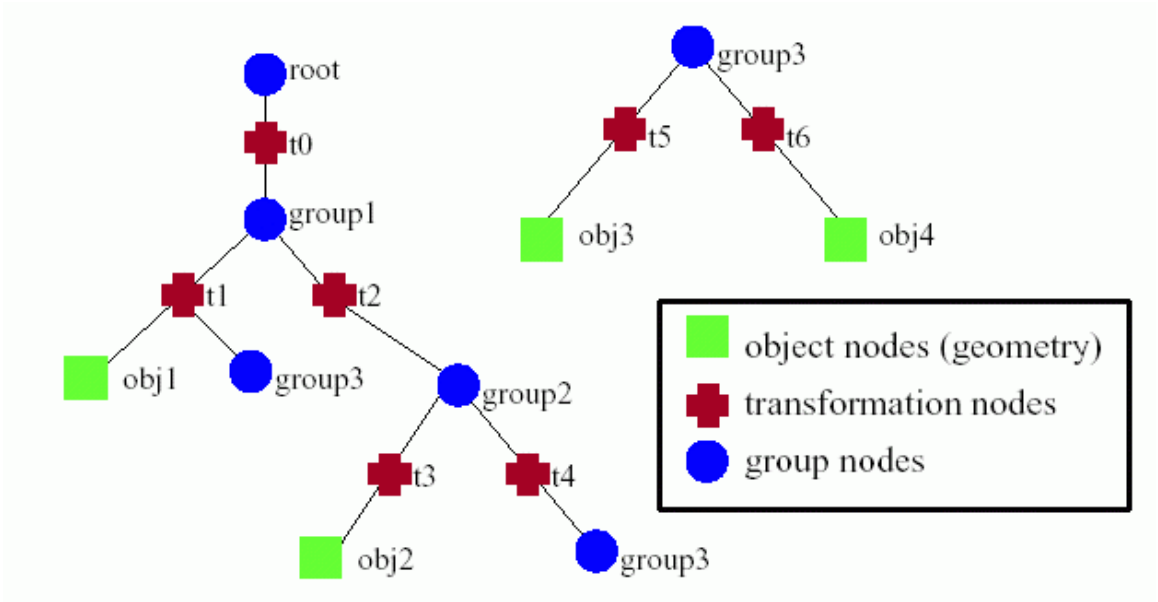
Ergebnis:



Grundlegende Datenstruktur in VRML zur Beschreibung komplexerer Objekte und Szenen:

Szenengraph

Es gibt Objektknoten, Transformationsknoten (stehen für Drehungen, Verschiebungen etc.) und Gruppenknoten



jeder Knoten gilt nur für die im Graphen darunterliegenden Elemente
z.B. wird im abgebildeten Beispiel t1 auf obj1 angewandt, aber nicht auf obj2.

selbe Datenstruktur (weiter ausgebaut) auch in Java 3D (Java-Erweiterung für 3D-Grafik).

Beispiel für Verwendung eines **`Transform`**-Knotens in VRML:

```
#VRML V2.0 utf8
Transform
   {
   translation 0 0 -5
   children
      [
      Shape
         {
         geometry Sphere { radius 0.2 }
         appearance Appearance
            {
            material Material
               { diffuseColor 1 0 0 }
            }
         }
      ]
   }
```

beschreibt rote Kugel, die um 5 Längeneinheiten in negativer *z*-Richtung verschoben ist.

Szenengraph: