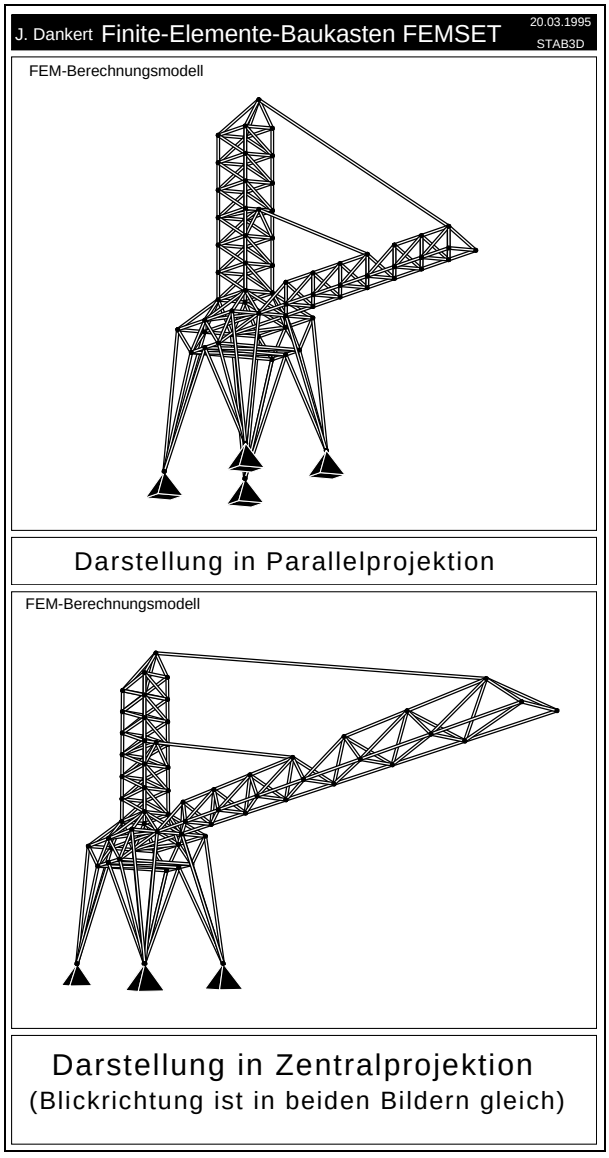
4 Projektionen

Das Problem, dreidimensionale Objekte auf eine zweidimensionale Zeichenfläche abzubilden, wird durch **Projektion** der dreidimensionalen Punkte auf eine Ebene nach fest zu definierenden Regeln gelöst:

- Bei der **Zentralprojektion** wird von einem **Projektionszentrum ("Eye point")** zu jedem Punkt des Körpers ein **Sehstrahl** gezogen. Der Schnittpunkt des Sehstrahls mit der Ebene, auf der die zweidimensionale Abbildung entstehen soll **(Projektionsebene)**, ist die Abbildung des 3D-Punktes in der Zeichenebene.
- Die **Parallelprojektion** kann als Sonderfall der Zentralprojektion angesehen werden, bei der das Projektionszentrum im Unendlichen liegt, so daß alle Sehstrahlen parallel verlaufen.

Während die Zentralprojektion (besonders mit nicht zu großen Entfernungen des Projektionszentrums vom Objekt, vgl. nebenstehendes Bild, "Eye point" etwa in realer Augenhöhe) den räumlichen Eindruck besonders gut vermittelt, hat die Parallelprojektion unter anderem die angenehme Eigenschaft, vertikale Linien in der Projektionsebene auch vertikal darzustellen.



4.1 Allgemeine Theorie der Zentralprojektion

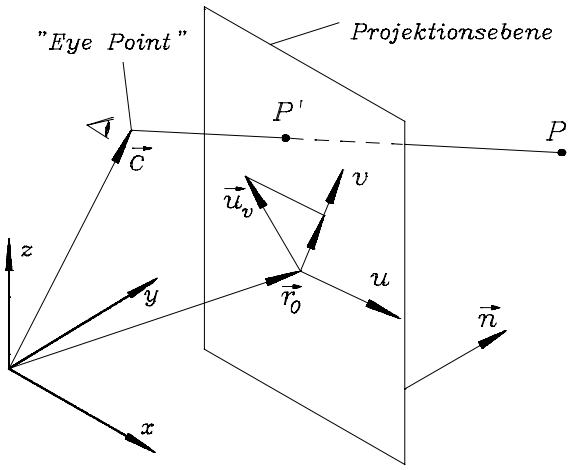
Eine **Zentralprojektion** wird definiert mit

- dem **"Eye point"** (Projektionszentrum), beschrieben durch einen Vektor $\vec{c}$,
- der **Projektionsebene**, die durch einen **Referenzpunkt** (Vektor $\vec{r_0}$) und einen **Normalenvektor** $\vec{n}$ beschrieben wird, und
- einem sogenannten **"Up vector"** $\vec{u_v}$ zur Definition des ebenen ***u*-*v***-Koordinatensystems der Bildebene.

Die Vektoren $\vec{c}$, $\vec{r}_0$, $\vec{n}$, $\vec{u}_v$ werden in einem dreidimensionalen raumfesten ***x-y-z***-Koordinatensystem ("World coordinates") beschrieben, für den Vektor $\vec{n}$ wird vereinbart, daß er zu der Seite der Projektionsebene zeigt, auf der der "Eye point" nicht liegt.

Das zweidimensionale Koordinatensystem der Bildebene definiert sich wie folgt:

- Der Ursprung fällt mit dem Referenzpunkt zusammen.
- Die Richtung der ***v***-Achse wird durch die Projektion des "Up vectors" auf die Projektionsebene festgelegt.



Allgemeine Definition der Zentralprojektion

- Die Richtung der ***u***-Achse wird so festgelegt, daß ***u***, ***v*** und $\vec{n}$ in dieser Reihenfolge ein Linkssystem definieren ("Eye point" befindet sich vor dem Bildschirm, $\vec{n}$ zeigt in den Bildschirm hinein und das ***u-v***-Koordinatensystem liegt in der Bildschirmebene wie die ebenen "User coordinates", wenn die Projektion des $\vec{u}_v$-Vektors auf dem Bildschirm vertikale Richtung hat).

Ein beliebiger Körperpunkt ***P***, der im dreidimensionalen ***x-y-z***-Koordinatensystem durch einen (in der Skizze nicht gezeichneten) Vektor $\vec{p}$ beschrieben wird und sowohl vor oder hinter und auch in der Projektionsebene liegen darf, wird auf die Projektionsebene abgebildet, indem der Schnittpunkt ***P'*** (nachfolgend durch den Vektor $\vec{p}'$ beschrieben) berechnet wird, den der **Sehstrahl**, der vom "Eye point" zum Punkt ***P*** gezogen wird, mit der Projektionsebene hat.

Da die Koordinaten des Punktes ***P'*** für den Zeichenvorgang im ebenen ***u-v***-Koordinatensystem benötigt werden, werden zunächst zwei Vektoren $\vec{u}$ und $\vec{v}$ in Richtung der ***u***- bzw. ***v***-Achse bestimmt, die dann zu den Einheitsvektoren $\vec{u}_e$ und $\vec{v}_e$ des Bildebenen-Koordinatensystems normiert werden:

Der Normalenvektor $\vec{n}$ wird durch Division durch seinen Betrag zum Normalen-Einheitsvektor

$$\vec{n}_e = \frac{\vec{n}}{|\vec{n}|} \quad . \tag{4.1}$$

Der "Up vector" $\vec{u}_v$ kann als Summe des Vektors $\vec{v}$ und eines Vektors $k\,\vec{n}_e$ (mit zunächst noch unbestimmten Faktor k) aufgeschrieben werden:

$$\vec{u}_v = \vec{v} + k\,\vec{n}_e \quad . \tag{4.2}$$

Multiplikation dieser Beziehung mit $\vec{n}_e$ führt wegen $\vec{n}_e \cdot \vec{v} = 0$ (Vektoren stehen senkrecht aufeinander) und mit $\vec{n}_e \cdot \vec{n}_e = 1$ auf

$$k = \vec{n}_e \cdot \vec{u}_v \quad , \tag{4.3}$$

was in die Beziehung (4.2) eingesetzt werden kann. Aus dem daraus berechneten Vektor

$$\vec{v} = \vec{u}_v - (\vec{n}_e \cdot \vec{v}_u) \cdot \vec{n}_e \tag{4.4}$$

entsteht schließlich der Einheitsvektor in Richtung der ***v***-Achse:

$$\vec{v}_e = \frac{\vec{v}}{|\vec{v}|} \quad . \tag{4.5}$$

Der Einheitsvektor in Richtung der ***u***-Achse kann nun einfach aus dem Vektorprodukt

$$\vec{u}_e = \vec{n}_e \times \vec{v}_e \tag{4.6}$$

berechnet werden.

Mit den (gesuchten) Koordinaten ***u*** und ***v*** des Punktes ***P′*** kann der Vektor $\vec{p}'$ zu diesem Punkt formal aufgeschrieben werden als

$$\vec{p}' = \vec{r}_0 + u\,\vec{u}_e + v\,\vec{v}_e \quad . \tag{4.7}$$

Da der Endpunkt des Vektors $\vec{c}$ ("Eye point") und die Punkte ***P′*** und ***P*** auf einer Geraden liegen (Sehstrahl), können sich die beiden Differenzvektoren $(\vec{p}' - \vec{c})$ und $(\vec{p} - \vec{c})$ nur um einen (zunächst ebenfalls noch unbekannten) Faktor unterscheiden:

$$(\vec{p}' - \vec{c})\,\lambda = (\vec{p} - \vec{c}) \quad . \tag{4.8}$$

Aus den Beziehungen (4.7) und (4.8) wird der Vektor $\vec{p}'$ eliminiert (die Koordinaten des Punktes ***P′*** im dreidimensionalen Koordinatensystem sind ohnehin nicht interessant), und es verbleibt mit

$$(\vec{r}_0 + u\,\vec{u}_e + v\,\vec{v}_e - \vec{c})\,\lambda = \vec{p} - \vec{c} \tag{4.9}$$

eine Vektorgleichung für die drei Unbekannten ***u***, ***v*** und λ. Die Vektoren werden durch ihre Komponenten dargestellt:

$$\vec{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad , \quad \vec{r}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad , \quad \vec{c} = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \quad , \quad \vec{u}_e = \begin{bmatrix} u_{ex} \\ u_{ey} \\ u_{ez} \end{bmatrix} \quad , \quad \vec{v}_e = \begin{bmatrix} v_{ex} \\ v_{ey} \\ v_{ez} \end{bmatrix} \quad , \tag{4.10}$$

für die Produkte der unbekannten Koordinaten ***u*** und ***v*** mit λ werden neue Unbekannte eingeführt:

$$\bar{u} = u\,\lambda \quad , \quad \bar{v} = v\,\lambda \quad . \tag{4.11}$$

Dann kann Gleichung (4.9) als lineares Gleichungssystem formuliert werden. Die Lösung von

$$\underbrace{\begin{bmatrix} u_{ex} & v_{ex} & x_0 - c_x \\ u_{ey} & v_{ey} & y_0 - c_y \\ u_{ez} & v_{ez} & z_0 - c_z \end{bmatrix}}_{\bar{P}} \underbrace{\begin{bmatrix} \bar{u} \\ \bar{v} \\ \lambda \end{bmatrix}}_{} = \underbrace{\begin{bmatrix} x - c_x \\ y - c_y \\ z - c_z \end{bmatrix}}_{\vec{b} \;=\; \vec{p} - \vec{c}} \quad , \tag{4.12}$$

liefert den Vektor der (ebenen) homogenen Koordinaten des Punktes ***P′*** im Koordinatensystem der Bildebene:

$$\vec{b} = \begin{bmatrix} \bar{u} \\ \bar{v} \\ \lambda \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \bar{u}/\lambda \\ \bar{v}/\lambda \end{bmatrix} \quad . \qquad \textbf{(4.13)}$$

Die Projektion versagt für $\lambda = \mathbf{0}$, was folgende Gründe haben kann:

- Der zu projizierende Punkt ***P*** ist identisch mit dem "Eye point" **($\vec{p} = \vec{c}$)**, oder
- der Sehstrahl und der Normalenvektor der Projektionsebene stehen senkrecht aufeinander.

In beiden Fällen ist natürlich auch keine sinnvolle Projektion möglich. Formal liefert die Projektion auch Bildkoordinaten ***u*** und ***v***, wenn der Körperpunkt ***P*** hinter dem "Eye point" liegt. Diese Fälle, die sich durch ein negatives λ äußern, sollten aussortiert werden.

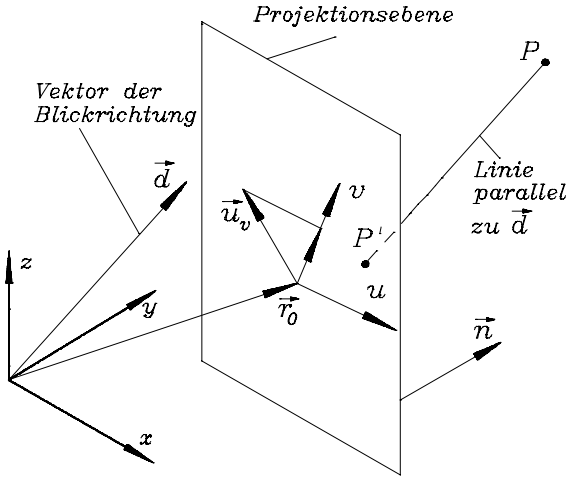
4.2 Allgemeine Theorie der Parallelprojektion

Eine **Parallelprojektion** wird definiert durch

- den "Vektor der Blickrichtung" $\vec{d}$,
- die **Projektionsebene**, die durch einen **Referenzpunkt** (Vektor $\vec{r}_0$) und einen **Normalenvektor** $\vec{n}$ beschrieben wird, und
- einen sogenannten **"Up vector"** $\vec{u}_v$ zur Definition des ebenen ***u-v***-Koordinatensystems der Bildebene.

Die Vektoren $\vec{d}$, $\vec{r}_0$, $\vec{n}$, $\vec{u}_v$ werden in einem dreidimensionalen raumfesten ***x-y-z***-Koordinatensystem beschrieben, für den Vektor $\vec{n}$ wird vereinbart, daß er einen spitzen Winkel mit dem "Vektor der Blickrichtung" bildet.

Die Definition des zweidimensionalen ***u-v***-Koordinatensystems der Bildebene erfolgt exakt nach der Vorschrift, die für die Zentralprojektion beschrieben wurde, so daß die Formeln (4.1) bis (4.7) unverändert gelten.



Allgemeine Definition der Parallelprojektion

Ein beliebiger Körperpunkt ***P***, der im dreidimensionalen ***x-y-z***-Koordinatensystem durch einen (in der Skizze nicht gezeichneten) Vektor $\vec{p}$ beschrieben wird und sowohl vor oder hinter und auch in der Projektionsebene liegen darf, wird auf die Projektionsebene abgebildet, indem der Schnittpunkt ***P′*** (nachfolgend durch den Vektor $\vec{p}'$ beschrieben)

berechnet wird, den **eine Parallele zum Vektor der Blickrichtung** durch ***P*** mit der Projektionsebene hat.

Der Vektor zum Punkt ***P*** kann also als Summe des Vektors zum Punkt ***P′*** und dem mit einem (zunächst unbekannten) Faktor multiplizierten "Vektor der Blickrichtung" $\vec{d}$ aufgeschrieben werden:

$$\vec{p} = \vec{p}' + \alpha \vec{d} \quad . \tag{4.14}$$

Da auch Formel (4.7) ihre Gültigkeit behält, kann aus (4.7) und (4.14) der Vektor $\vec{p}'$ (ähnlich zum Vorgehen bei der Zentralprojektion) eliminiert werden. Man erhält mit

$$\vec{r}_0 + u \vec{u}_e + v \vec{v}_e = \vec{p} - \alpha \vec{d} \tag{4.15}$$

eine Vektorgleichung für die drei Unbekannten ***u***, ***v*** und α. Die Vektoren werden durch ihre Komponenten dargestellt:

$$\vec{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} , \quad \vec{r}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} , \quad \vec{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} , \quad \vec{u}_e = \begin{bmatrix} u_{ex} \\ u_{ey} \\ u_{ez} \end{bmatrix} , \quad \vec{v}_e = \begin{bmatrix} v_{ex} \\ v_{ey} \\ v_{ez} \end{bmatrix} , \tag{4.16}$$

und Gleichung (4.15) kann als lineares Gleichungssystem formuliert werden. Die Lösung von

$$\begin{bmatrix} u_{ex} & v_{ex} & d_x \\ u_{ey} & v_{ey} & d_y \\ u_{ez} & v_{ez} & d_z \end{bmatrix} \begin{bmatrix} u \\ v \\ \alpha \end{bmatrix} = \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} , \tag{4.17}$$

$$\bar{P}_p \qquad \vec{b}_p = \vec{p} - \vec{r}_0$$

liefert unmittelbar die Koordinaten ***u*** und ***v*** des Punktes ***P′*** im Koordinatensystem der Bildebene (und den nicht interessierenden Parameter α). Man beachte, daß die Lösung von (4.17) im Gegensatz zur Lösung von (4.12) keine homogenen Koordinaten liefert, sondern direkt die kartesischen Koordinaten des ebenen ***u***-***v***-Koordinatensystems.

4.3 Empfehlungen zur Definition von Projektionen

Die volle Allgemeinheit, mit der die Zentralprojektion und die Parallelprojektion in den Abschnitten 4.1 und 4.2 beschrieben wurden, sollte nicht dazu verführen, beliebige Kombinationen der jeweils vier Vektoren, die eine Projektion definieren, zu verwenden. Folgende Empfehlungen können allgemein gegeben werden:

- Bei der **Zentralprojektion** wird der Fußpunkt des Lotes vom "Eye point" auf die Projektionsebene als "Hauptpunkt" bezeichnet. Es ist empfehlenswert, den Referenzpunkt der Projektionsebene (Vektor $\vec{r}_0$) mit dem Hauptpunkt zusammenfallen zu lassen. Da der Referenzpunkt der Ursprung des ebenen Koordinatensystems der Bildebene ist, steht die Projektionsebene dann senkrecht zum Sehstrahl auf den Nullpunkt des ***u***-***v***-Koordinatensystems.

- Bei der **Parallelprojektion** sollte die Projektionsebene senkrecht zum "Vektor der Blickrichtung" gelegt werden. Dann ist der Normalenvektor, der die Projektionsebene definiert, parallel zum "Vektor der Blickrichtung".
- Im allgemeinen ist ein "Up vector", der parallel zur ***z***-Achse des dreidimensionalen Koordinatensystems liegt, eine gute Wahl (vermittelt den Eindruck, als würde der Beobachter auf einer zur ***x***-***y***-Ebene parallelen Ebene aufrecht stehen). Ein solcher "Up vector" ist allerdings nicht möglich, wenn der "Eye point" der Zentralprojektion selbst auf der ***z***-Achse liegt, oder der "Vektor der Blickrichtung" der Parallelprojektion parallel zu ***z***-Achse ist.
- Die in technischen Zeichnungen üblichen Darstellungen (Vorderansicht, Seitenansicht, Draufsicht) sind als Sonderfälle der Parallelprojektion zu realisieren.

4.4 Vereinfachte Definition der Projektionen in der Klasse CPT

Die allgemeinen Definitionen von Zentralprojektion und Parallelprojektion mit jeweils vier Vektoren, wie sie in den Abschnitten 4.1 und 4.2 vorgestellt wurden, werden in der Klasse **CPT** verwaltet und können bei Bedarf für spezielle Untersuchungen genutzt werden. Wenn nicht wirklich gute Gründe gegen die nachfolgend beschriebenen vereinfachten Definitionen sprechen, die eine Projektion mit nur zwei Vektoren eindeutig beschreiben, dann sollten diese benutzt werden. Die **CPT**- und **CGI**-Funktionen, die in den folgenden Abschnitten vorgestellt werden, arbeiten mit diesen sinnvollen Einschränkungen.

Definition einer Zentralprojektion in der Klasse CPT:

- Es werden nur der **"Eye point"** (Projektionszentrum) und ein Punkt der Projektionsebene (**"Referenzpunkt"**) im raumfesten ***x***-***y***-***z***-Koordinatensystem ("World coordinates") definiert.
- Die Projektionsebene wird von den **CPT**-Funktionen automatisch so gelegt, daß sie senkrecht zur Verbindungslinie vom "Eye point" zum Referenzpunkt liegt (der Normalenvektor hat also die Richtung dieser Verbindungslinie, der Referenzpunkt ist gleichzeitig der sogenannte "Hauptpunkt" der Projektion).
- Als "Up vector" wird i. a. automatisch die ***z***-Achse gewählt (dies vermittelt den Eindruck, als würde der Beobachter auf einer zur ***x***-***y***-Ebene parallelen Ebene aufrecht stehen). Wenn der "Eye point" selbst auf der ***z***-Achse liegt, wird entweder die positive ***y***-Achse ("Eye point" liegt auf der positiven ***z***-Achse, "Draufsicht") oder die negative ***y***-Achse ("Eye point" liegt auf der negativen ***z***-Achse) zum "Up vector".

Diese Definition mit zwei Vektoren ist nicht nur eindeutig, sie erfüllt auch die im vorigen Abschnitt gegebenen Empfehlungen. Eine entsprechende Aussage gilt auch für die folgende "Zwei-Vektor-Definition" der Parallelprojektion.

Definition einer Parallelprojektion in der Klasse CPT:

- Es werden nur der **"Vektor der Blickrichtung"** und ein Punkt der Projektionsebene (**"Referenzpunkt"**) im raumfesten ***x*-*y*-*z***-Koordinatensystem ("World coordinates") definiert.
- Die Projektionsebene wird von den **CPT**-Funktionen automatisch so gelegt, daß sie senkrecht zum Vektor der Blickrichtung liegt (der Normalenvektor hat also die gleiche Richtung wie der Vektor der Blickrichtung).
- Als "Up vector" wird i. a. automatisch die ***z***-Achse gewählt (dies vermittelt den Eindruck, als würde der Beobachter auf einer zur ***x*-*y***-Ebene parallelen Ebene aufrecht stehen). Wenn der "Vektor der Blickrichtung" parallel zur ***z***-Achse ist, wird entweder die positive ***y***-Achse ("Vektor der Blickrichtung" hat negative ***z***-Komponente, "Draufsicht") oder die negative ***y***-Achse ("Vektor der Blickrichtung" hat positive ***z***-Komponente) zum "Up vector".

4.5 Die "pr...-Funktionen" in den Klassen CPT und CGI

Die sogenannten **"pr...-Funktionen"** (alle zugehörigen Funktionsnamen beginnen mit **pr**) lassen sich in zwei Gruppen unterteilen:

- Die **"vorbereitenden pr...-Funktionen"** gehören zur Klasse **CPT**. Sie definieren bzw. ändern die gültige Projektion (es kann immer nur eine Projektion gültig sein, entweder eine Zentral- oder eine Parallelprojektion).
- Die **"zeichnenden pr...-Funktionen"** gehören zur Klasse **CGI**. Ihnen werden die Punkte in "World coordinates" übergeben (bezogen auf ein festes ***x*-*y*-*z***-Koordinatensystem), die vor der Zeichenaktion mit der aktuellen Projektion umgerechnet und damit zu zweidimensionalen "User coordinates" werden.

Beim Erzeugen eines **CPT**-Objektes werden alle Projektions- und Transformations-Parameter vom Konstruktor **CPT::CPT** initialisiert, so daß eine 2D-Transformation, eine 3D-Transformation und eine Projektion eingestellt sind. Man kann jederzeit die Transformationen mit

```
CPT::tinit_pt ()
```

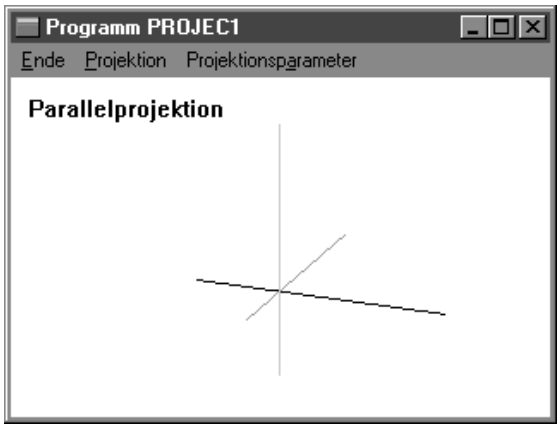
und die Projektion mit

```
CPT::prini_pt ()
```

auf die Anfangswerte, die vom Konstruktor vergeben wurden, zurücksetzen. Beim Aufruf der **CPT**-Methoden, die eine Transformation oder Projektion ändern, wird überprüft, ob dies realisierbar ist. Gegebenenfalls wird die Änderung abgelehnt, so daß stets gültige Transformationen und eine gültige Projektion aktiv sind.

- Weil in der Klasse **CPT** kein "Device context" verzeichnet ist, kann eine **CPT**-Instanz bereits beim Erzeugen eines Fensters (Botschaft WM_CREATE) erzeugt und z. B. in Dialog-Funktionen geändert werden.
- **CPT::prini_pt** definiert eine Zentral- und eine Parallel-Projektion, die (bei nicht "zu großen" Objekten) sehr ähnliche Ansichten erzeugen, weil als Referenzpunkt der Projektionsebene für beide Projektionen der Nullpunkt des 3D-Koordinatensystems eingestellt wird und sich der "Eye point"-Vektor (Zentralprojektion) vom "Blickrichtungs"-Vektor (Parallelprojektion) nur durch das Vorzeichen unterscheidet. Nach der Initialisierung ist zunächst die Parallelprojektion gültig ("vorsichtshalber", bei einer Zentralprojektion besteht immer die Gefahr, daß man mit dem "Eye point" im darzustellenden Objekt liegt).

Das Programm **projec1.c** zeigt das Zusammenspiel der wichtigsten **"pr...-Funktionen"** am ganz einfachen Beispiel (dargestellt werden nur drei gerade Linien, die entlang der Koordinatenachsen des 3D-"World coordinates"-Systems verlaufen, siehe nebenstehendes Bild). Die Projektionen werden selbständig in einer **CPT**-Instanz **m_cpt** verwaltet (gehört zur Klasse **CMainFrame**), erst beim Erzeugen der **CGI**-Instanz **gi** (für die Zeichenaktionen in **CMainFrame::OnPaint**) werden die aktuellen Projektionsparameter via Konstruktor in **gi** eingesetzt.



Programm projec1.cpp nach dem Start

Die wichtigsten Teile des Programms werden nachfolgend in Ausschnitten vorgestellt. In der Klasse **CMainFrame** ist eine **CPT**-Instanz angesiedelt:

```
class CMainFrame : public CFrameWnd
{
   private:
       CPT m_cpt ;
   // ...
} ;
```

Beim Erzeugen des Hauptfenster wird also auch das Objekt **m_cpt** erzeugt und mit dem Standard-Konstruktor **CPT::CPT** initialisiert. Dieser legt u. a. den Referenzpunkt in den Ursprung des "World coordinates"-System und stellt für die Parallelprojektion einen Blickrichtungsvektor (-2000;5000;-2000) ein. Damit wird das Bild unmittelbar nach dem Programmstart (siehe oben) gezeichnet.

Dieses **CPT**-Objekt kann über Dialoge geändert werden. In **projec1.h** werden die beiden (aus **CDialog** abgeleiteten) Dialog-Klassen **CParmCentralDlg** und **CParmParallelDlg** deklariert, über deren Konstruktoren werden die Ressourcen **IDD_DIALOG_Zentral** bzw. **IDD_DIALOG_Parallel** zugewiesen, die in **projec1.rc** definiert sind. Auf die Einzelheiten dazu wird hier nicht eingegangen (der Quellcode ist verfügbar).

Über die Auswahl der entsprechenden Menü-Angebote werden **CMainFrame::OnParmCentral** bzw. **CMainFrame::OnParmParallel** angesteuert, die die Dialoge auslösen. Beide Funktionen arbeiten gleichartig, deshalb wird hier nur die letztgenannte gelistet:

```
void CMainFrame::OnParmParallel ()
{
   CParmParallelDlg dlg (this) ;
   double  x , y , z ;
   m_cpt.prgtd_pt (&x , &y , &z) ;   // ... liefert Blickrichtungsvektor
   dlg.m_dirx = x ;
   dlg.m_diry = y ;
   dlg.m_dirz = z ;
   m_cpt.prgtr_pt (&x , &y , &z) ;   // ... liefert Referenzpunkt
   dlg.m_refx = x ;
   dlg.m_refy = y ;
   dlg.m_refz = z ;
   if (dlg.DoModal () == IDOK)   // Neue Parallelprojektion erzeugen:
   {
      m_cpt.prstp_pt (dlg.m_dirx , dlg.m_diry , dlg.m_dirz ,
                      dlg.m_refx , dlg.m_refy , dlg.m_refz) ;
      Invalidate () ;
   }
}
```

- In **OnParmParallel** werden nach dem Erzeugen der Dialog-Instanz **dlg** mit **CPT::prgtd_pt** die Koordinaten des Blickrichtungsvektors und mit **CPT::prgtr_pt** die Koordinaten des Referenzpunktes erfragt und in **dlg** eingesetzt, so daß sie beim Erzeugen der Dialog-Box (mit **DoModal**) über die **DoDataExchange**-Methode als Initialisierungswerte in der Box erscheinen.

- Nach dem Verschwinden der Dialog-Box finden die aktualisierten Werte wieder über den **DoDataExchange**-Algorithmus den Weg in das Objekt **dlg** zurück. Wenn die Dialog-Box mit **OK** geschlossen wurde, wird mit den neuen Werten eine neue Parallelprojektion erzeugt. Das geschieht mit **CPT::prstp_pt**, für diese Methode ist in **cgiw.h** der Prototyp

  ```
  int  prstp_pt  (double xwd , double ywd , double zwd ,
                  double xwr , double ywr , double zwr)
  ```

  verzeichnet, der zeigt, daß die 3 Koordinaten des Blickrichtungsvektors und die 3 Koordinaten des Referenzpunktes erwartet werden, mit denen versucht wird, die neue Parallelprojektion zu erzeugen. Der Return-Wert, der angibt, ob dies gelingt, wird in **CMainFrame::OnParmParallel** nicht ausgewertet. Im Falle eines Mißerfolgs (Return-Wert: 0) bleibt die alte Einstellung erhalten.

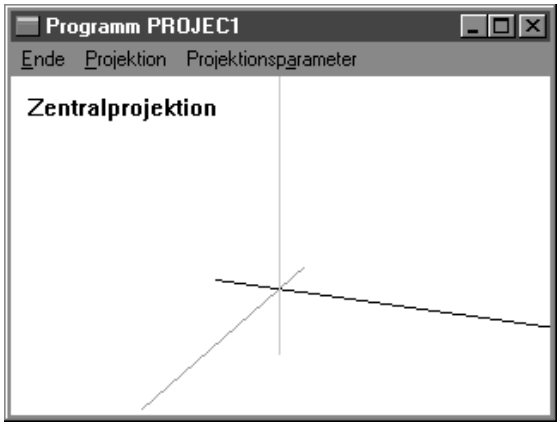
- In der (hier nicht gelisteten) Methode **CMainFrame::OnParmCentral** findet man die analog zu den beschriebenen **CPT**-Methoden arbeitenden Methoden **CPT::prgte_pt** zum Erfragen der aktuellen Koordinaten des "Eye points" und **CPT::prstc_pt** zum Erzeugen einer neuen Zentralprojektion, die an Stelle des von **CPT::prstp_pt** erwarteten Blickrichtungsvektors die "Eye point"-Koordinaten erwartet.

Die Menü-Kommandos zum Umstellen auf einen anderen Projektionstyp werden an **CMainFrame::OnCentral** bzw. **CMainFrame::OnParallel** geleitet, die letzgenannte wird gelistet:

```
void CMainFrame::OnParallel ()
{
   m_cpt.projn_pt (m_cpt.GI_PARALLEL) ;
   Invalidate () ;
}
```

- Die Festlegung eines Projektionstyps erfolgt mit **CPT::projn_pt**. Hier wird die **CPT**-Konstante **GI_PARALLEL** übergeben, das Pendant dazu heißt **GI_CENTRAL**.

Wenn sofort nach dem Programmstart auf die Zentralprojektion umgestellt wird, zeigt sich das nebenstehende Bild, das in diesem Fall deutlich anders als die Darstellung mit der Parallelprojektion ist. Dies liegt an dem relativ "großen Objekt" (vgl. **OnPaint** weiter unten), so daß der "Eye point" innerhalb des darzustellenden "World coordinates"-Bereichs liegt, so daß ein veränderter Ausschnitt ziemlich stark verzerrt dargestellt wird.



Programm projec1.cpp: Zentralprojektion

Die Botschaft WM_PAINT wird an **CMainFrame::OnPaint** geleitet, wo zunächst (wie üblich) der "Device context" und ein **CGI**-Objekt erzeugt werden. Letzteres geschieht mit einem Konstruktor, der bisher nicht verwendet wurde. Er ist in **cgiw.h** mit folgendem Prototyp vertreten:

```
CGI  (CWnd *wnd_p , CDC *dc_p , CPT *cpt_p , int vpclip = GI_CLIPPING) ;
```

Pointer auf das aktuelle Window (1. Parameter), Pointer auf den "Device context" (2. Parameter) und Einschalten des Viewport-Clippings als Standard-Vorgabe (4. Parameter) sind aus den vorigen Abschnitten bekannt. Neu ist der 3. Parameter: Hier wird ein Pointer auf ein **CPT**-Objekt erwartet, und das **CGI**-Objekt wird mit allen darin verzeichneten Transformationen und Projektionen konstruiert.

Es folgt das komplette Listing von **CMainFrame::OnPaint**:

```
void CMainFrame::OnPaint ()
{
   CPaintDC  dc (this) ;
   CGI       gi (this , &dc , &m_cpt) ;

   CPen  PenRed  (PS_SOLID , 1 , gi.mkrgb_gi (gi.GI_RED )) ;
   CPen  PenBlue (PS_SOLID , 1 , gi.mkrgb_gi (gi.GI_BLUE)) ;

   gi.stuci_gi (- 15000. , - 5000. , 15000. , 10000.) ;

   gi.prmov_gi (-5000. ,  0. ,  0.) ;               // Schwarze Linie entlang
   gi.prlin_gi (10000. ,  0. ,  0.) ;               // der x-Achse

   CPen* pPenOld = dc.SelectObject (&PenRed) ;
   gi.prmov_gi (0. , -5000. ,  0.) ;                // Rote Linie entlang
   gi.prlin_gi (0. , 10000. ,  0.) ;                // der y-Achse

   dc.SelectObject (&PenBlue) ;
   gi.prmov_gi (0. ,  0. , -5000.) ;                // Blaue Linie entlang
   gi.prlin_gi (0. ,  0. , 10000.) ;                // der z-Achse

   dc.TextOut (10 , 10 , (gi.prgtp_pt () == gi.GI_CENTRAL) ?
               "Zentralprojektion " : "Parallelprojektion" , 18);

   dc.SelectObject (pPenOld) ;
}
```

- Die drei geraden Linien werden (mit unterschiedlichen Farben) mit **CGI::prmov_gi** und **CGI::prlin_gi** gezeichnet. Da **CGI::prmov_gi** und **CGI::prlin_gi** die übergebenen 3D-Koordinaten ("World coordinates") in 2D-"User coordinates" umrechnen, muß ein geeigneter Bereich für die "User coordinates" eingestellt werden, was hier mit **CGI::stuci_gi** erledigt wird (in den folgenden Abschnitten werden Funktionen

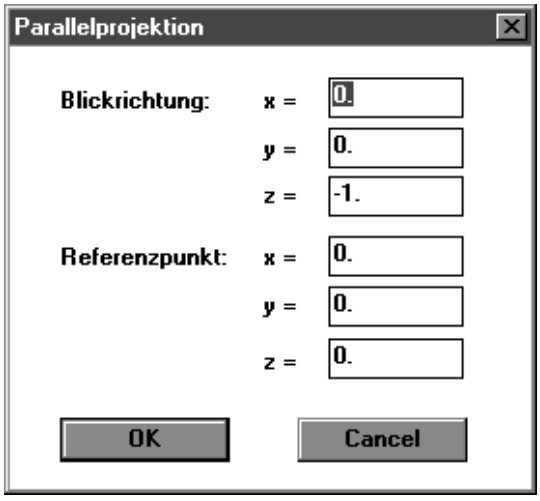
vorgestellt, mit denen man zu geeigneten Grenzwerten für den **CGI::stuci_gi**-Aufruf kommt).

- Der aktuelle Projektionstyp wird mit **CGI::prgtp_pt** erfragt und mit den Konstanten verglichen, die auch für das Setzen des Projektionstyps (mit **CPT::projn_pt**) verwendet werden.

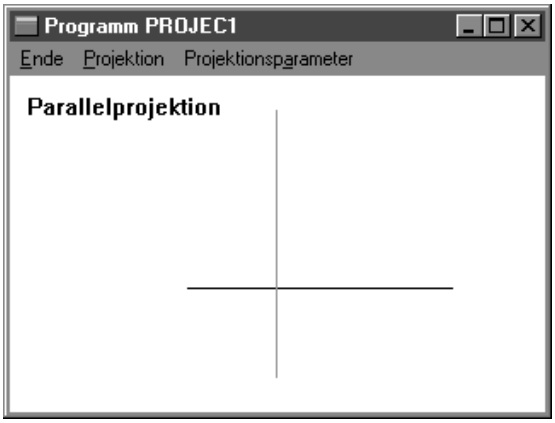
Die "zeichnenden pr...-Funktionen" (im Programm **projec1.cpp** sind dies **CGI::prmov_gi** und **CGI::prlin_gi**) arbeiten wie die "u...-Funktionen" mit ähnlichen Namen (z. B.: **CGI::umove_gi** bzw. **CGI::uline_gi**) mit folgenden Unterschieden:

- Den "zeichnenden pr...-Funktionen" werden 3D-Punkte übergeben ("World coordinates"), die von den Funktionen auf "User coordinates" (unter Benutzung der aktuellen Projektion) umgerechnet werden. Das kann fehlschlagen (z. B.: Punkt befindet sich "hinter dem Eye point"), dann wird die Aktion nicht ausgeführt, und ...
- die "pr...-Funktion" signalisiert den Mißerfolg mit dem Return-Wert **0** (bei Erfolg ist der Return-Wert **1**). Im Programm **projec1.cpp** werden die Return-Werte der "zeichnenden pr...-Funktionen" nicht ausgewertet.

Über die Dialoge sind beliebige Projektionen einstellbar. Die Abbildung unten links zeigt die Dialog-Box für die Parameter der Parallelprojektion. Die eingegebenen Werte (Nullpunkt als Referenzpunkt und Blick in Richtung der negativen z-Achse) entsprechen einer "Draufsicht". Das rechte Bild zeigt, daß man dann nur die x- und die y-Achse sieht.



"Parallelprojektions-Dialog"



"Draufsicht"

Während in der Regel (automatisch von den **CPT**-Funktionen) der "Up vector" zur Definition der v-Achse der "User coordinates" parallel zur z-Achse gewählt wird, so daß der Betrachter den Eindruck hat, er würde auf der xy-Ebene stehen, ist die "Draufsicht" zwangsläufig ein Sonderfall. Wie der Name suggeriert, schaut man "von oben" auf die xy-Ebene.

4.6 "t3...-Funktionen" der Klasse CPT und "pt...-Funktionen" der Klasse CGI

Die zur Klasse **CPT** gehörenden "t3...-Funktionen" (alle Funktionsnamen beginnen mit **t3**) sind die 3D-Versionen der im Abschnitt 3.4 vorgestellten "vorbereitenden t...-Funktionen" (es wird eine Transformation definiert oder geändert). Die zur Klasse **CGI** gehörenden "pt...-Funktionen" (alle zugehörigen Funktionsnamen beginnen mit **pt**) sind die 3D-Versionen der im Abschnitt 3.4 vorgestellten "zeichnenden t...-Funktionen" und der zugehörigen Hilfsfunktionen (vor der eigentlichen Zeichenaktion werden die Koordinaten einer Transformation unterworfen). Allerdings sind "zeichnende 3D-Funktionen" natürlich nur sinnvoll, wenn außerdem eine Projektion auf die Zeichenfläche ausgeführt wird (**pt** steht für "Projektion und Transformation").

Das Prinzip des Arbeitens mit den Funktionen, die eine Transformation setzen, ändern und auswerten, ist völlig analog zur Arbeitsweise, die im Abschnitt 3.4 vorgestellt wurde:

- Vor der ersten Verwendung einer "t3...- bzw. pt...-Funktion" müssen alle Projektions- und Transformationsparameter initialisiert werden. Dies wird im Konstruktor **CPT::CPT** erledigt, kann aber jederzeit noch einmal mit **CPT::tinit_pt** (Transformationen) und **CPT::prini_pt** (Projektionen) ausgeführt werden.
- Bei der Initialisierung wird die "Einheits-Transformation" eingestellt, so daß alle "pt...-Funktionen" zunächst nicht anders arbeiten als die entsprechenden "pr...-Funktionen", die im Abschnitt 4.5 vorgestellt wurden.
- Mit den "t3...-Funktionen" wird die 3D-Transformation modifiziert. Die jeweils gültige Transformation wird für die Punkte, die den "pt...-Funktionen" übergeben werden, vor der eigentlichen Zeichenaktion ausgeführt. Man beachte, daß (wie bei den 2D-"t...-Funktionen") nicht die Datenstruktur, die das Objekt beschreibt, geändert wird. Die Koordinaten werden **nur für die Darstellung modifiziert**.

Das Programm **projec2.cpp** ist eine Modifikation des Programms **projec1.cpp** aus dem Abschnitt 4.5, allerdings wurden alle "pr...-Funktionen" durch die entsprechenden "pt...-Funktionen" ersetzt. Zusätzlich wurde die Möglichkeit vorgesehen, die 3D-Transformation dadurch zu ändern, daß (inkrementell) Translationen in Richtung der 3 Koordinatenachsen und Rotationen um diese Achsen über die Tastatur eingebracht werden können, so daß das dargestellte Objekt beliebig verschoben und gedreht werden kann.

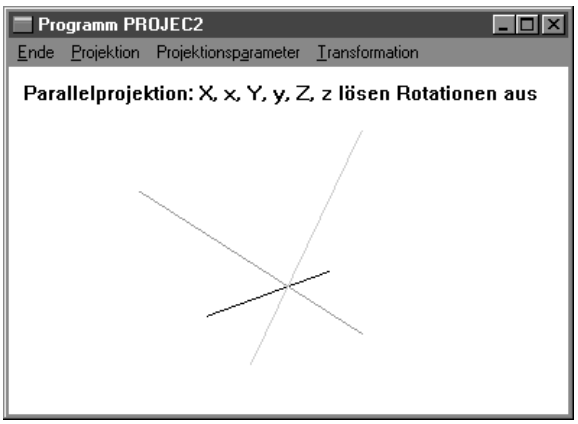
Da sich die Auswertung der Botschaft WM_PAINT in **CMainFrame::OnPaint** (neben einer etwas erweiterten Textausgabe) nur dadurch von der entsprechenden Funktion in **projec1.cpp** unterscheidet, daß **CGI::prmov_gi** durch **CGI::ptmov_gi** und **CGI::prlin_gi** durch **CGI::ptlin_gi** ersetzt wurden, wird auf ein Listing verzichtet (die "transformierenden und projizierenden" Funktionen erwarten auch die gleichen Parameter wie die "nur projizierenden" Funktionen).

Nach dem Programmstart sieht man das gleiche Bild wie nach dem Start von **projec1.cpp**. Durch Drücken der Tasten **x**, **y**, **z** bzw. **X**, **Y**, **Z** wird eine Rotationstransformation um die jeweilige Achse um **10°** bzw. **-10°** (zusätzlich zur aktuellen Transformation) eingestellt. Über das Menüangebot **Transformation** kann dies geändert werden: Wenn die Option **Translation** gewählt wird, bewirkt nachfolgendes Drücken einer der genannten Tasten, daß eine Translation um **1000** bzw. **-1000** in Richtung der jeweiligen Achse (zusätzlich) eingestellt wird.

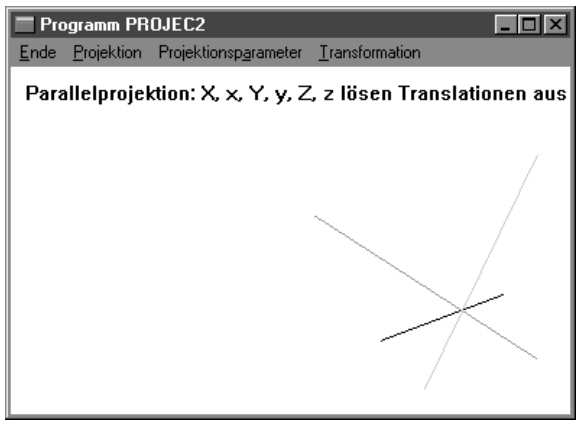
Die Modifikation der aktuellen Transformation wurde als Auswertung der Botschaft WM_CHAR, die an **CMainFrame::OnChar** geleitet wird, programmiert. In Abhängigkeit von einem Schalter **m_rot** (gehört zur Klasse **CMainFrame**, Voreinstellung: **1**), der über das genannte Menüangebot umgestellt werden kann, werden entweder eine zusätzliche Rotation mit **CPT::t3rot_gi** oder eine zusätzliche Translation mit **CPT::t3trn_gi** eingestellt:

```
void CMainFrame::OnChar (UINT nChar , UINT nRepCnt , UINT nFlags)
{
   switch (nChar)
   {
      case 'x' :  m_rot ? m_cpt.t3rot_pt (GI_PI / 18. , m_cpt.GI_AXISX) :
                          m_cpt.t3trn_pt (1000. , 0. , 0.) ;
                  break ;
      case 'X' :  m_rot ? m_cpt.t3rot_pt (- GI_PI / 18. , m_cpt.GI_AXISX) :
                          m_cpt.t3trn_pt (- 1000. , 0. , 0.) ;
                  break ;
      case 'y' :  m_rot ? m_cpt.t3rot_pt (GI_PI / 18. , m_cpt.GI_AXISY) :
                          m_cpt.t3trn_pt (0. , 1000. , 0.) ;
                  break ;
      case 'Y' :  m_rot ? m_cpt.t3rot_pt (- GI_PI / 18. , m_cpt.GI_AXISY) :
                          m_cpt.t3trn_pt (0. , - 1000. , 0.) ;
                  break ;
      case 'z' :  m_rot ? m_cpt.t3rot_pt (GI_PI / 18. , m_cpt.GI_AXISZ) :
                          m_cpt.t3trn_pt (0. , 0. , 1000.) ;
                  break ;
      case 'Z' :  m_rot ? m_cpt.t3rot_pt (- GI_PI / 18. , m_cpt.GI_AXISZ) :
                          m_cpt.t3trn_pt (0. , 0. , - 1000.) ;
                  break ;
      default  :  return ;
   }
   Invalidate () ;
}
```

Da alle Transformationen "addiert" werden, kann das Objekt in jede beliebige Lage im Raum gebracht werden. Wenn man "den Finger auf einer Taste läßt", ergibt sich der Eindruck einer einfachen Animation (empfehlenswert, wenn "Rotation" eingestellt ist, sonst ist das Objekt sehr schnell verschwunden). Natürlich läßt sich jede beliebige Ansicht des Objekts auch durch Änderung der Projektions-Parameter erreichen (Bewegung des Betrachters, nicht des Objekts), aber das ist eine andere Denkweise für den Progamm-Benutzer.



Darstellung nach einigen Rotationen ...



... und Translationen