sprachen. Außerdem bietet Delphi auch im Bereich der Grafikausgabe eine sehr effektive und relativ einfach zu handhabende Plattform. Ich habe für die Entwicklung die Entwicklungsumgebung „Borland Delphi 5“ der Firma Inprise verwendet.

3. Umgang mit dem dreidimensionalen Raum

Für eine realistische Simulation ist es natürlich erforderlich, den Ablauf im dreidimensionalen Raum zu betrachten. Den nun folgenden Modellbeschreibungen liegt ein Koordinatensystem dieser Form zugrunde:

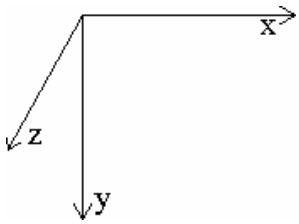
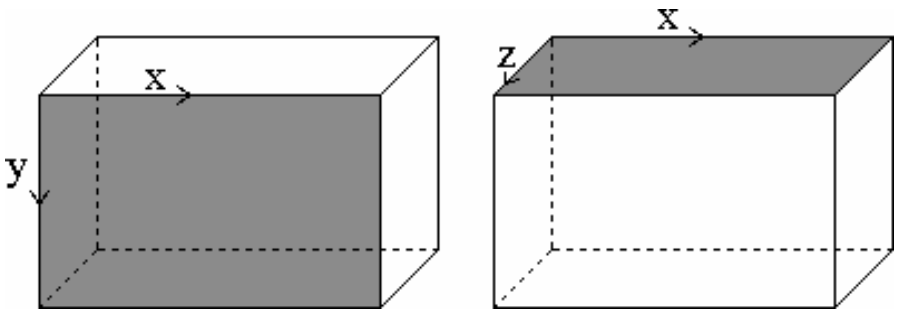


Abb. D.3. 1

Diese Form des Koordinatensystems bietet sich besonders an, da bei der „Vorderansicht“ die x-y-Koordinaten und bei der „Draufsicht“ die x-z-Koordinaten direkt zum Zeichnen verwendet werden können. Beim Zeichnen auf die Zeichenfläche eines Programms in Delphi hat das zweidimensionale Koordinatensystem den Ursprung nämlich ebenfalls in der linken oberen Ecke (vgl. Abb. D.3. 2).



Vorderansicht **(Abb. D.3. 2)** **Draufsicht**

a) Felder und Wände

Für einen geeigneten Versuchsaufbau werden verschiedene „Bauteile“ benötigt. Drei davon sind die elektrischen und magnetischen Felder sowie Wände. Wände können z.B. zum Aufbau einer Blende o.ä. dienen.

Der geometrische Aufbau dieser drei Elemente ist identisch: Es handelt sich hierbei um Körper in der Form eines Parallelflachs. Geometrisch beschrieben

werden diese durch einen Antragspunkt P und drei Richtungsvektoren – den Seiten $\vec{a}$, $\vec{b}$ und $\vec{c}$ (vgl. Abb. D.3. 3).

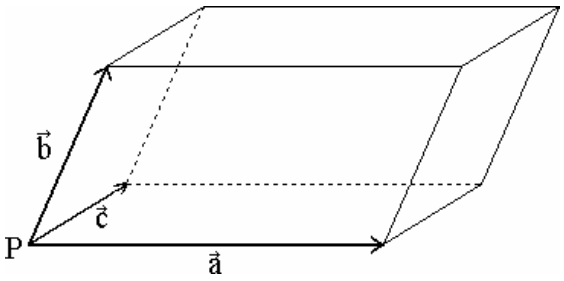


Abb. D.3. 3

Die Punkte des Körpers werden also beschrieben durch

$$\vec{x} = \begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix} + k \cdot \begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} + l \cdot \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} + m \cdot \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \qquad \text{mit } 0 \le k,l,m \le 1.$$

Zum Speichern des Antragspunktes und der Richtungsvektoren sind zunächst zwei Klassen notwendig: TPunkt für einen Punkt und TRichtungsvektor für einen Richtungsvektor im Raum:

```
01    type
02    TPunkt = class
03    private
04    (...)
05    public
06        (...)
07        property x: Extended;       // x-Koordinate
08        property y: Extended;       // y-Koordinate
09        property z: Extended;       // z-Koordinate
10    end;
11
12  TRichtungsvektor = class
13    private
14    (...)
15    public
16        (...)
17        property x: Extended;       // x-Richtung
18        property y: Extended;       // y-Richtung
19        property z: Extended;       // z-Richtung
20        property Laenge: Extended;
21    end;
```

Unit: URaumpos.pas

Die Eigenschaften x, y und z sind die jeweiligen Koordinaten des Vektors. Das Abrufen der Eigenschaft „Laenge“ eines Objekts der Klasse TRichtungsvektor liefert die Länge des Vektors nach der Formel $\left\| \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right\| = \sqrt{x^2 + y^2 + z^2}$.

Es wird also für jede Wand ein Objekt der Klasse TWand angelegt, in dem die Eigenschaften gespeichert werden:

```
01  type
02     TWand = class
03     private
04     (...)
05     public
06        (...)
07        property Position: TPunkt;
08        property Seitea: TRichtungsvektor;
09        property Seiteb: TRichtungsvektor;
10        property Seitec: TRichtungsvektor;
11     end;
```

Unit: UWand.pas

Für jedes Feld wird ein Objekt der Klasse TFeld angelegt, das neben den Eigenschaften Position, Seitea, Seiteb und Seitec außerdem vier weitere Eigenschaften trägt: Feldtyp, E bzw. B und Feldlinien.

```
01  type
02     TFeld = class
03     private
04     (...)
05     public
06        (...)
07        property Feldtyp: Integer; // ftEFeld oder ftMFeld
08        property B: Extended; // magn. Flussdichte in T
09        property E: Extended; // elektr. Feldstärke in N/C
10        property Feldlinien: TRichtungsvektor;
                                  // Richtung der Feldlinien
11     end;
12  const
13     ftEFeld = 1;  // Konstante für ein elektrisches Feld
14     ftMFeld = 2; // Konstante für ein Magnetfeld
```

Unit: UFeld.pas

Der Wert der Eigenschaft *Feldtyp* gibt Auskunft darüber, ob es sich um ein elektrisches oder ein Magnetfeld handelt. Die Eigenschaften *B* und *E* speichern den Wert der magnetischen Flussdichte in Tesla bzw. die elektrische Feldstärke in N/C. Die Eigenschaft Feldlinien gibt die Richtung der Feldlinien an.

b) Teilchenquelle

Die Teilchenquelle ist das einfachste aller Elemente, die für den Versuchsaufbau benötigt werden. Sie gibt an, wo und in welche Richtung die Teilchen (bzw. der sog. „Kanalstrahl“) emittiert werden. Dazu benötigt sie lediglich zwei Eigenschaften: eine Position und eine Richtung. Auch dafür wird eine eigene Klasse erstellt:

```
01  type
02      TTeilchenquelle = class
03      private
04      (...)
05      public
06          (...)
07          property Position: TPunkt;
08          property Richtung: TRichtungsvektor;
09      end;
```

Unit: UTeilchenquelle.pas

Wird also später ein Teilchen simuliert, so erhält es zu Beginn der Simulation die Position der Teilchenquelle und die Startgeschwindigkeit wird in die angegebene Richtung ausgerichtet.

c) Photoschirm

Der Photoschirm ist vor allem für den Massenspektrographen das wichtigste Instrument zur Auswertung des Versuchs. Teilchen, die auf den Photoschirm treffen, werden in einem Extrafenster der Anwendung als Punkte dargestellt. Dabei können die genauen Koordinaten ermittelt werden, in denen das Teilchen den Schirm trifft.

Der Schirm an sich ist eine Ebene im Raum. Diese wird wieder durch einen Antragspunkt P und zwei Richtungsvektoren – den Seiten $\vec{a}$ und $\vec{b}$ – beschrieben:

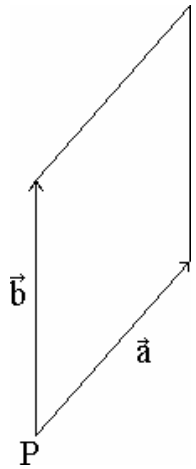


Abb. D.3. 4

Als Objektklasse in Delphi sieht das dann wie folgt aus:

```
01  type
02      TPhotoschirm = class
03      private
04      (...)
05      public
06          (...)
07          property Position: TPunkt;
08          property Seitea: TRichtungsvektor;
09          property Seiteb: TRichtungsvektor;
10      end;
```
Unit: UPhotoschirm.pas

d) Teilchen

Jetzt fehlt nur noch ein Element – das Teilchen an sich: Es wird vereinfachend als eine punktförmige Ladung mit einer bestimmten Masse und Geschwindigkeit angesehen. Auch dafür gibt es eine eigene Klasse: die Klasse TTeilchen. Dies sind die wichtigsten Eigenschaften:

```
01  type
02      TTeilchen = class
03      private
04      (...)
05      public
06          (...)
07          property Position: TPunkt;
08          property v: TRichtungsvektor; // Geschwindigkeit
09          property vgesamt: Extended; //Betr. der Geschw.
10          property Ladung: Extended; // Ladung in C
11          property LadungInE: Extended; // Ladung in e
12          property Ruhemasse: Extended; // Ruhemasse
                                                  in kg
13          property RuhemasseInU: Extended; // Ruhem. in
                                                  u
14          property RelativeMasse: Extended; // rel. Masse
                                                  in kg
15      end;
```
Unit: UTeilchen.pas

Die hier aufgeführten Eigenschaften bedürfen natürlich einer Erklärung:

Eigenschaft	Beschreibung
Position	Beschreibt die aktuelle Position des Teilchens im Raum.
v	Vektor der Geschwindigkeit: $\vec{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$
vgesamt	Geschwindigkeitsbetrag $\left\|\vec{v}\right\| = \sqrt{v_x^2 + v_y^2 + v_z^2}$

Ladung	Ladung des Teilchens in Coulomb
LadungInE	Ladung des Teilchens als Vielfaches der Elementarladung e
Ruhemasse	Ruhemasse des Teilchens in kg
RuhemasseInU	Ruhemasse des Teilchens als Vielfaches der atomaren Masseneinheit u
RelativeMasse	Liefert die relative Masse des Teilchens in kg nach der Formel: $m = m_0 \cdot \frac{1}{\sqrt{1-\left(\frac{v}{c}\right)^2}}$. (Ist die relativistische Berechnung im Programm abgeschaltet, so liefert diese Eigenschaft den Wert der Ruhemasse in kg!)

4. Ablauf der Simulation

Ist nun ein Versuchsaufbau erstellt, so kann die Simulation durchgeführt werden. Diese erfolgt für jedes Teilchen einzeln:

a) Die Beschleunigung des Teilchens in der Teilchenquelle

In den Einstellungen des Simulationsprogramms wurden zwei Spannungswerte angegeben: $U_{\min}$ und $U_{\max}$ sowie die Anzahl der „Zwischenspannungen“, d.h. das Teilchen wird jeweils mit so vielen Spannungen zwischen $U_{\min}$ und $U_{\max}$ in der Teilchenquelle beschleunigt und erhält somit seine Anfangsgeschwindigkeit. Sind also beispielsweise $U_{\min} = 0V$ und $U_{\max} = 10V$ mit 10 Zwischenspannungen angegeben, so wird für jedes ausgewählte Isotop die Simulation mit einer Beschleunigungsspannung von jeweils 1, 2, 3, 4, 5, 6, 7, 8, 9 und 10V durchgeführt. Die Berechnung der Anfangsgeschwindigkeit erfolgt entweder relativistisch oder nicht relativistisch – je nach Einstellung[3]:

nicht relativistisch: $Q \cdot U = \frac{1}{2} m_0 v^2 \Rightarrow v = \sqrt{\frac{2 \cdot Q \cdot U}{m_0}}$

relativistisch: $E_{ges} - E_0 = E_{kin}$

$mc^2 - m_0 c^2 = Q \cdot U$

[3] Im gesamten Verlauf der Simulation wird je nach gewählter Einstellung zur Berechnung der Beschleunigungen usw. ausschließlich die Ruhemasse bzw. die relativistische Masse des Teilchens verwendet.