Liebe APL Freunde,

es freut mich, Ihnen die Doppelausgabe des AP-Journals 2008 zukommen zu lassen.

In diesem Heft sind die Vorträge aus unseren Tagungen abgedruckt, soweit diese von den Referenten geliefert wurden. Im vergangenen Jahr sind wir im Juni bei der IBM Niederlassung Hannover und im November dei DPC in Augsburg zu Gast gewesen. Den Gastgebern möchte ich an dieser Stelle noch einmal danken für das professionelle Ausrichten der Tagungen.

Den Dank an die Referenten, welche sich die Mühe gemacht haben, interessante Beiträge vorzubereiten und auch in druckfähiger Version zu liefern verbinde ich mit der Einladung an die MItglieder, weiterhin aktiv zu bleiben und den Organisatoren der APL Veranstaltungen Vortragsangebote zu liefern.

Im Mai diesen Jahres sind wir bei der Allianz in Stuttgart gewesen. Unsere nächste APL Tagung in diesem Jahr wird am 29. und 30. Oktober bei dem RSGV in Düsseldorf stattfinden. Ich möchte zu dieser Tagung herzlich einladen und bitte Sie, den Termin vorzumerken.

Eine interesante Lektüre wünscht Ihnen

Ihr

Reiner Nussbaum

INHALT

Reiner Nussbaum
Dynamisches Erstellen von HTML-Seiten
Dieser Beitrag beschreibt Lösungswege, wie HTML-Seiten von einer Anwendung auf sehr einfache und flexibe Weise dynamisch erstellt werden können. Es werden am Beispiel von GPS-Info stufenweise Möglichkeiten aufgezeigt, wie eine Anwendung durch die Parameterisierung von aussen immer weiter steuerbar wird.

Herbert Voß
3D-Abbildungen im Textmodus erstellen
Das APL-Journal hat bereits in verschiedenen Ausgaben das Satzprogramm TEX behandelt. In diesem Artikel wird als Ergänzung zum letzten Heft die Möglichkeit gezeigt, mit normalen Textanweisungen qualitativ hochwertige dreidimensionale Abbildungen zu erreichen.

Jürgen Fenn
Literaturverwaltung mit Firefox, Zotero und BIBTEX
Mit der neuen Literaturverwaltung Zotero, die als Plugin in Firefox 2 eingebunden wird, können auch BIBTEX-Daten verarbeitet werden.

Albert Badior
Galileo Galilei und die Wurfparabel
Von 1945 bis 1955 wurden in den USA ballistische Berechnungen vorwiegend auf der ENIAC (Electronical Numerical Integrator and Computer) durchgeführt. Gegenüber den elektromechanischen Tischrechnern wurde die Zeit für die Bahnberechung um den Faktor 200 verkürzt. Ab 1965 brachte die Firma IBM das Computersystem 7090 auf den Markt; auch auf diesen Rechnern wurden ballistische Aufgaben, nun wesentlich schneller als auf der ENIAC gelöst. Auf der 7090 lief auch das erste APL-System unter dem Namen IVSYS (Iverson System), das viel später zum APL2 wurde. Diese Möglichkeiten standen Galileo Galilei nicht zur Verfügung, als er Anfang des 17. Jahrhunderts das Fallgesetz und die Flugparabel entdeckte.

Dynamisches Erstellen von HTML-Seiten **3**

3D-Abbildungen im Textmodus erstellen **14**

APL-Programm zur Lösung von SUDOKU Rätseln **36**

Neues in Workstation APL2 V2.0 **40**

Literaturverwaltung mit Firefox, Zotero und BIBTEX **42**

Galileo Galilei und die Wurfparabel **46**

Impressum **56**

Bildnachweis:
Martin Barghoorn (Umschlagseiten, S.3), manwalk_pixelio.de. (S. 35), Gabi-Schoenemann_pixelio.de (S. 36), manfred-walker_pixelio.de (S. 42), Stephanie-Hofschlaeger_pixelio.de (S. 46), Ingrid-Ruthe_pixelio.de (S. 55),

Reiner Nussbaum

Dynamisches Erstellen von HTML-Seiten

(Einbinden von Anwendungen am Beispiel von GPS-Info)

Dieser Beitrag beschreibt Lösungswege, wie HTML-Seiten von einer Anwendung auf sehr einfache und flexibe Weise dynamisch erstellt werden können. Es werden stufenweise Möglichkeiten aufgezeigt, wie eine Anwendung durch die Parameterisierung von aussen immer weiter steuerbar wird.

Auf der Basis von gift-ALU (Patent DE 102.60.999) wird GPS-Info für das Erstellen der dynamischen HTML-Seiten herangezogen.

Ausgehend von der Trennung von Programm und Daten (HTML-Fragmente als Variable) über das Auslagern der HTML-Fragmente in externe, konfiguratierbare Dateien bis zum Einsatz von Platzhaltern für Anwendungsinformationen wird der Grad der Flexibilität erhöht.

So ist es am Ende möglich, mit HTML-Fragmenten verschiedene Anwendungen in die dynamisch erstellten HTML-Seiten einzubinden, ohne die eigentliche Anwendung zu modifizieren.

1. Arbeitsweise: HTML-Seiten Schritt 1

Die GPS-Info-Anwendung dient dazu, bei gestohlenen Endgeräten den rechtmässigen Besitzer zu ermitteln. Dabei werden externe Datenträger ausgewertet, welche Informationen enthalten und auch dann zugänglich sind, wenn das entsprechende Gerät passwortgeschützt ist oder keinen Strom mehr hat. Die interne Arbeitsweise wurde in dem Artikel *gift_ALU GPS-Info (APL-Journal 2007)* bereits beschrieben.

Anhand der point of interest- oder Konfigurationsdateien, welche in ein fallbezogenes Verzeichnis kopiert werden, erfolgt die Auswertung der angelieferten Daten, sowie die Ausgabe der Ergebnisse in einem dazugehörenden Ausgabeverzeichnis.

Als Ergebnisse werden dort abgeliefert:

- Die Originaldatei selbst.
- Eine Meldungsdatei, wenn aus der angegebenen Datei keine Informationen extrahiert werden konnten.
- Eine Textdatei, in der alle gefundenen GPS-Informationen enthalten sind. Aus praktischen Überlegungen werden die GPS-Informationen in verschiedenen Darstellungsformen, als Grad und Bruchteile davon, sowie in der Schreibweise Grad, Minuten, Sekunden und Bruchteilen von Sekunden angegeben. Diese Datei kann für weitere Ermittlungsarbeiten oder für Archivzwecke (elektronische Akte) verwendet werden.
- Eine HTML-Seite, welche die gefundenen GPS-Informationen enthält. Über

geeignete Verweise ist es anschließend unmittelbar möglich, die GPS-Informationen auf einer Landkarte darstellen zu lassen. Als Landkarten kommen öffentliche Informationen oder auch spezielle kommunale Landkarten mit einem weitergehenden Detaillierungsgrad in Frage. Die HTML-Lösung hat den Vorteil, dass eine unmittelbare Zuordnung von GPS-Geodaten zu Orten, Straßen und Hausnummer möglich ist, so dass ein sehr einfacher Abgleich mit als gestohlen gemeldeten Geräten möglich ist. Dies vereinfacht die Zuordnung zu den rechtmäßigen Eigentümern entscheidend.

An dieser Stelle werden Methoden aufgezeigt, mit denen eine Anwendung flexibel auf Anforderungen unterschiedlicher Darstellungen realisierbar ist.

Natürlich können Programme HTML-Seiten erstellen. Aus Gründen der Wartbarkeit und der besseren Strukturierung von Anwendungen hat sich die Trennung von Programm und Daten durchgesetzt. In bezug auf die hier aktuelle Fragestellung bedeutetet dies, dass APL-Variablen ein Rohgerüst für die zu erstellenden HTML-Seiten enthalten, welches von der GPS-Info Anwendung bei Vorhandensein von Auswertungsdaten zu einer konkreten (fallbezogenen) HTML-Seite umgewandelt wird.

Als Beispiel wird eine einfache HTML-Seite herangezogen, welche auf der linken Seite die Benutzerführung und auf der rechten Seite die Visualisierung der Auswertungsinhalte hat. Die Trennung von Programm und Daten bedeutet in diesem Fall, dass wir für jede Komponente eine APL-Variable definieren müssen (Die dargestellten Informationen der APL-Variablen sehen im APL Zeichensatz etwas gewöhnungsbedürftig aus, funktionieren als HTML-Ausgabe jedoch problemlos.):

Die Hauptseite ({prefix_}index.html)

```
      RAW_INDEX
<html><head><meta http-equiv="Content-Type" content="text/html; charset=iso-885
      9-1">
 <meta http-equiv="Content-Language" content="de">
 <title>Dr.Nussbaum gift GPS Informationen</title></head>
 <frameset cols="20%,80%" border="0">
 <frame src="LINKS.htm" name="linkesfenster">
 <frame src="RECHTS.htm" name="anzeigefenster"></frameset>
 </html>
```

Hier wird die darzustellende Seite in zwei Bereiche (links und rechts) unterteilt.

Der linke, dynamische Teil

```
      RAW_LINKS
<html>
 <body text="#000000" bgcolor="#FFFFFF" link="#0000EE" vlink="#551A8B" alink="#
      0000EE">
 <font color="#3333FF"><font size=+4>Inhalt</font></font>
 <p><b><font color="#3333FF"><font size=+0><a href="RECHTS.htm" target="anzeige
      fenster">Start</a></font></font></b>

 <p><b><font color="#3333FF"><font size=+0>GPS Informationen</font></font></b>

 LINK
 <br> 
 </body>
 </html>
```

In diesem Bereich ist die Stelle LINK dazu vorgesehen, um die Bewegungsdaten einzufügen, welche von der Anwendung gefunden wurden.

Der rechte Teil enthält als Anfangsinformation die Benutzerführung.

zu verlagern. Dies führt dazu, dass die HTML-Fragmente nicht unmittelbar aus APL-Variablen, sondern aus Dateien, welche an einer bestimmten Stelle erwartet werden, eingelesen werden. Dann können die APL-Variablen als Defaultwerte eingesetzt werden, soweit die entsprechenden exter-

```
        RAW_RECHTS
<html>
  <body text="#FFFFFF" bgcolor="#0000FF" link="#0000FF" vlink="#0000FF" alink="#
      FF0000">

  <div align=right><font size=+0>Dr.Nussbaum <b><i>gift</i></b></font></div>

  <br> 
  <br> 
  <p><font size=+1>Das GPS Auskunft System</font>
  <p><font size=+1>Auf der linken Seite sind die Informationen angegeben,
  welche aus den entsprechenden Dateien extrahiert wurden.
  Durch Anklicken der entsprechenden Informationen wird eine Landkarte
  eingeblendet, welche die Position auf einer Landkarte darstellt.
  <p>
  Ein Klick auf Start führt zu dieser Seite zurück.</font>
  </body>
  </html>
```

Die Anwendung ist damit in bezug auf die Ansicht nach aussen leicht wartbar, weil es nur drei Variablen gibt, welche das Aussehen bestimmen. Dabei ist es nicht erforderlich, in die Anwendungslogik einzugreifen, so dass hier keine Fehlerquellen für die Anwendung sind.

Natürlich kann das Ergebnis der Anwendung unbrauchbar werden, wenn in den HTML-Fragmenten grob fehlerhafte Informationen stehen.

2. Schritt 2: HTML-Fragmente als externe Dateien

Damit die Anwendung nach aussen flexibler gestaltbar ist, kann man sich überlegen, die APL-Variablen nach aussen

nen Dateien nicht vorhanden sind. So ist die Anwendung stets lauffähig.

Konsequenz daraus ist, dass aus ein und derselben Anwendung völlig unterschiedliche Darstellungsformen realisierbar sind. Wer gelegentlich die APL-Germany.de Website aufsucht, hat vielleicht schon gesehen, dass die Anfahrtsinformationen zu den Tagungen als HTML-Seite verfügbar sind. Ziel ist, eine Orientierungshilfe für den Tagungsort, das empfohlene Hotel sowie weitere nützliche Informationen bereitzustellen.

Für diese Aufgabe war es neben dem Bereitstellen des APL-Germany-Logos nur erforderlich, den linken und den rechten Teil der HTML-Hauptseite als Textdatei neu zu gestalten.

Dynamische HTML-Seiten mit GPS Info

Der linke Teil ist für die APL-Germany-Anfahrtinformationen:

```
<html>
```

```
<body bgcolor="#FFFCEF">
<a name="top"><td><img src="logosmal.gif"></td></a>
<tr>
<td style="background-color:#DDFFF4;font-size:10pt;line-height:14pt;" valign="top">
<ul>
<li><a href="index.htm" style="text-decoration:none;cursor:hand;">Homepage</a></li>

</ul>
</td>
<p><b><font color="#3333FF"><font size=+0><a href="RECHTS.htm"
target="anzeigefenster">Start</a></font></font></b>

<p><b><font color="#3333FF"><font size=+0>GPS Informationen</font></font></b>
LINK
</tr>
<p style="font-size:9pt;">
Webmaster: <a href="mailto:apl-de@mbaas.de">apl-de@mbaas.de</a>
</p>
</body>
</html>
```

In dieser Datei werden im wesentlichen das APL-Germany Logo sowie eine Kontaktadresse definiert. Der rechte Teil wurde geändert für die problembezogene Benutzerinformation.

```
<html>
```

```
<body text="#FFFFFF" bgcolor="#0000FF" link="#0000FF" vlink="#0000FF"
alink="#FF0000">

<div align=right><font size=+0>Dr.Nussbaum <b><i>gift</i></b></font></div>

<br> 
<br> 
<p><font size=+1>Orientierungsinformationen für unsere Tagungsorte</font>
<p><font size=+1>Auf der linken Seite sind die Informationen zu unseren
Tagungsorten angegeben.
Durch Anklicken der entsprechenden Informationen wird eine Landkarte
eingeblendet, welche die Position auf einer Landkarte darstellt.
Zur Darstellung benutzen wir die Informationen von Mapquest.
<p>
Ein Klick auf Start führt zu dieser Seite zurück.
<br><br>
Ein Klick auf Homepage führt zur APL-Germany Homepage zurück.
</font>
</body>
</html>
```

Das Ergebnis für unsere Berlin Tagung im November 2007 ist dann:

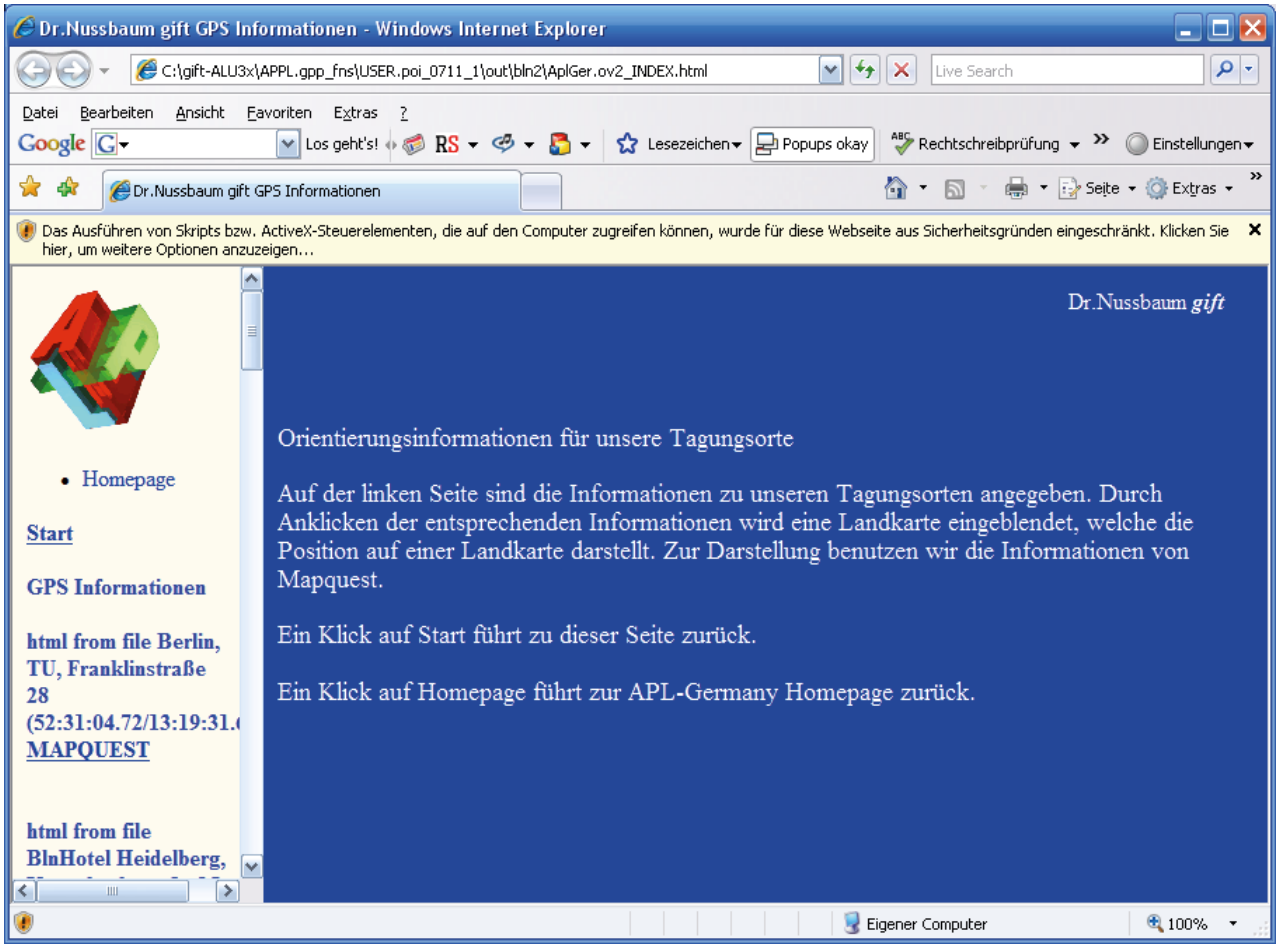


Abb. 1: Darstellung auf der HTML-Hauptseite

3. Herausgabe von Anwendungsdetails (Platzhalter)

Eine weitere Möglichkeit, die Flexibilität nach aussen zu erhöhen besteht darin, einen Teil der Anwendungslogik konfigurierbar zu machen.
Wenn der bis hier betrachtete dynamische Teil LINK (Anwendungsdaten aus der Auswertung von GPS- Informationen) weiter detailliert wird, so ist es möglich, diesen Bereich von aussen steuerbar zu machen. Letztendlich kommt von der Anwendung zu einer GPS-Position die Information „NORD", „West" und „TextINFO".

Wenn für diese Detailinformationen eine Detail-Datei wie folgt bereitgestellt wird, dann ändert sich an dem Layout des Ergebnisses zunächst überhaupt nichts:

```
  <p><b><font color="#3333FF"><font size=+0>html from file INFO
<a href="http://www.mapquest.com/maps/map.adp?latlongtype=decimal&latitude=INFN
&longitude=INFE&zoom=9" target="anzeigefenster">MAPQUEST</a>
</font></font></b>
<br> 
```

Dynamische HTML-Seiten mit GPS Info

Das Ergebnis ist die gleiche HTML-Seite für Berlin: Die fundamentale Erweiterung besteht darin, dass es nun auch von aussen möglich ist, andere Visualisierungen für die gefundenen GPS-Informationen verfügbar zu machen.

Dazu muss in der Anwendung selbst nichts geändert werden.

Nehmen wir als eine alternative Detail-Datei nachstehendes Beispiel:

```
<p><b><font color="#3333FF"><font size=+0>html from file INFO

<a href="http://maps.google.com/maps?ie=UTF8&om=1&ll=INFN,INFE&t=k&z=17"
target="anzeigefenster">SAT_VIEW</a></font></font></b>
<br> 
```

Dies hat als Ergebnis, dass nun nicht mehr die Landkarten Ansicht, sondern die Satellitenansicht dargestellt wird.

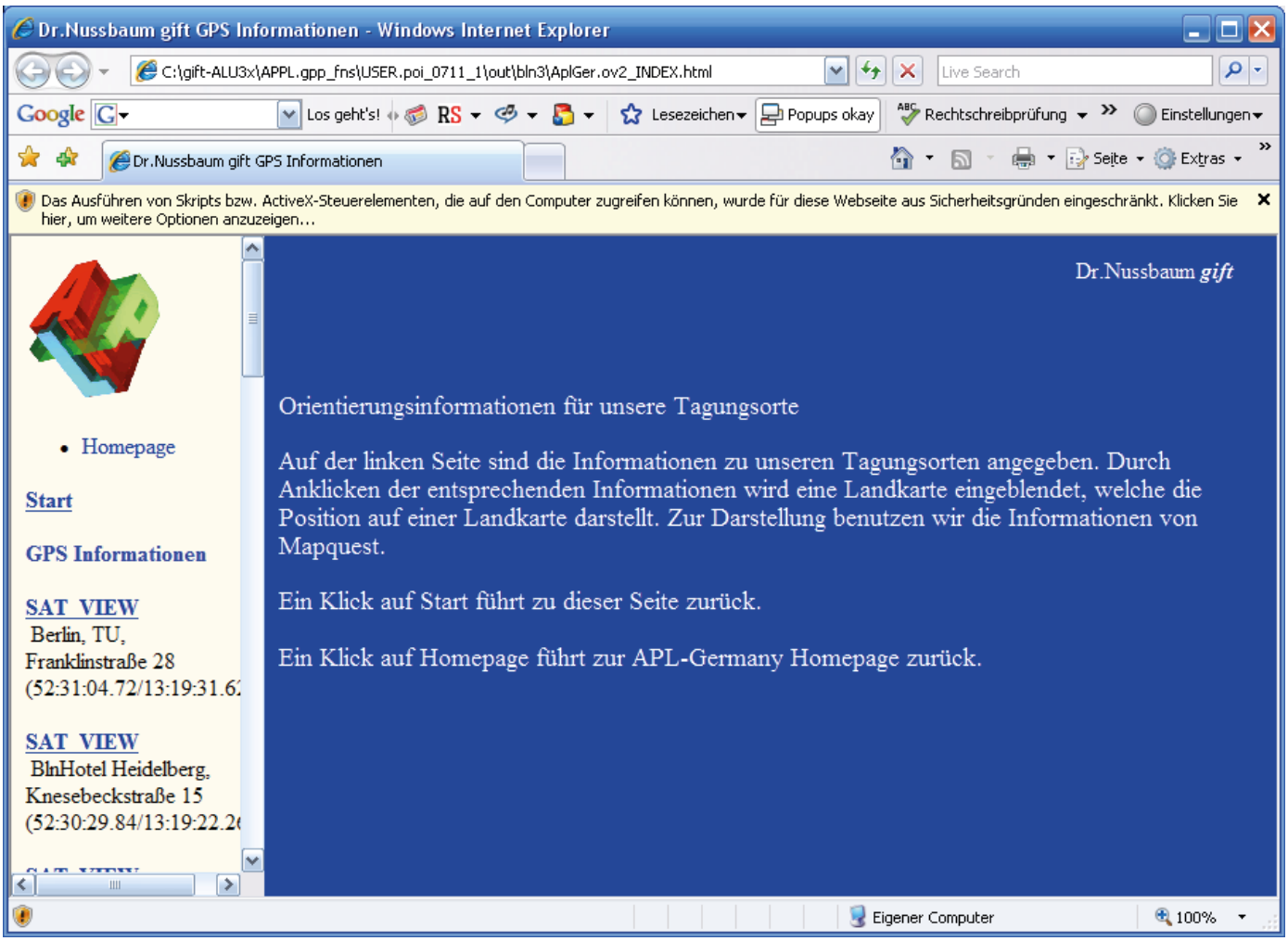


Abb. 2: Satellitenansicht

Ein Anklicken des SAT-VIEW-Links auf TU Berlin ergibt nun:

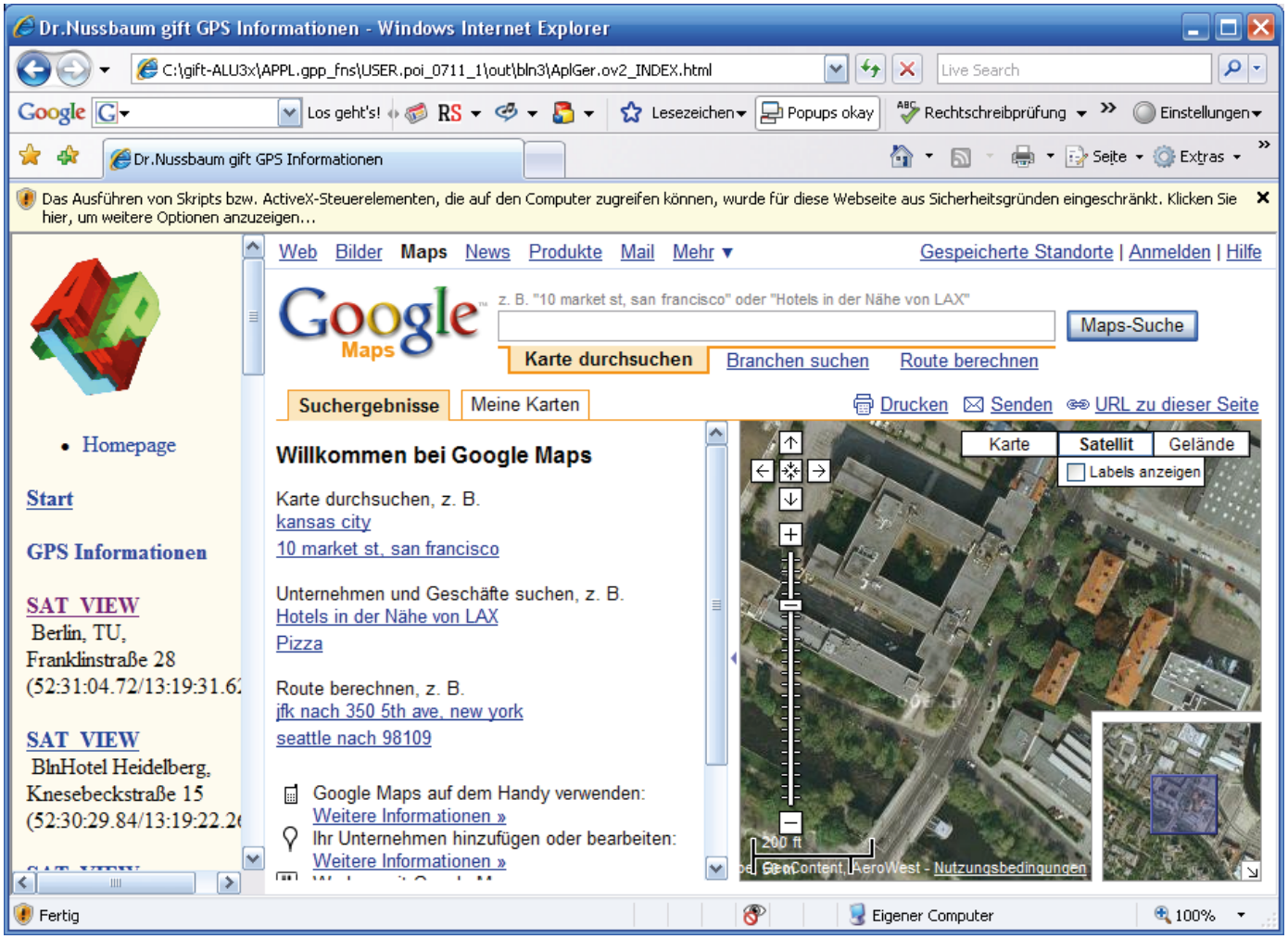


Abb. 3: SAT VIEW

Bei der ganzen Flexibilität, welche nach aussen getragen wird, darf man nicht vergessen, dass bei dem Erstellen dieser Detail-HTML-Fragmente auch eine gewisse Qualitätssicherung geboten ist. Schließlich kann ein Vertauschen von Nord- und Ost-Informationen zu einem Interpretationsbedürfnis führen.

4. Alles ist nicht genug – Einbinden mehrerer Anwendungen

Wenn es möglich ist, mehrere Visualisierungsanwendungen für die Darstellung von (GPS-)Informationen von aussen konfigurierbar zu machen, dann kann man auf die Idee kommen, diese Anwendungen auch „auf einen Schlag" verfügbar zu machen.

Dazu war es erforderlich, die Anwendung leicht zu erweitern. Gebraucht wurde eine Funktion, welche es ermöglicht, einen bestimmten Suchbegriff in einem String (hier HTML-Detail-Fragment) beliebig oft zu ersetzen. Dabei kann der Suchbegriff länger oder kürzer als der zu ersetzende Begriff sein.

Wie eine Lösung dazu aussehen kann, zeigt die nachfolgende Abbildung (nächste Seite oben).

Wenn diese Funktion vorhanden ist, dann kann bei der Ausgabe der gefundenen GPS-Informationen als Detailinformation die APL Anwendung wie folgt erweitert werden (siehe nachfolgende Abbildung, nächste Seite, zweite von oben).

Die Konsequenz davon ist, dass es nun möglich ist, in einem Durchgang der

Dynamische HTML-Seiten mit GPS Info

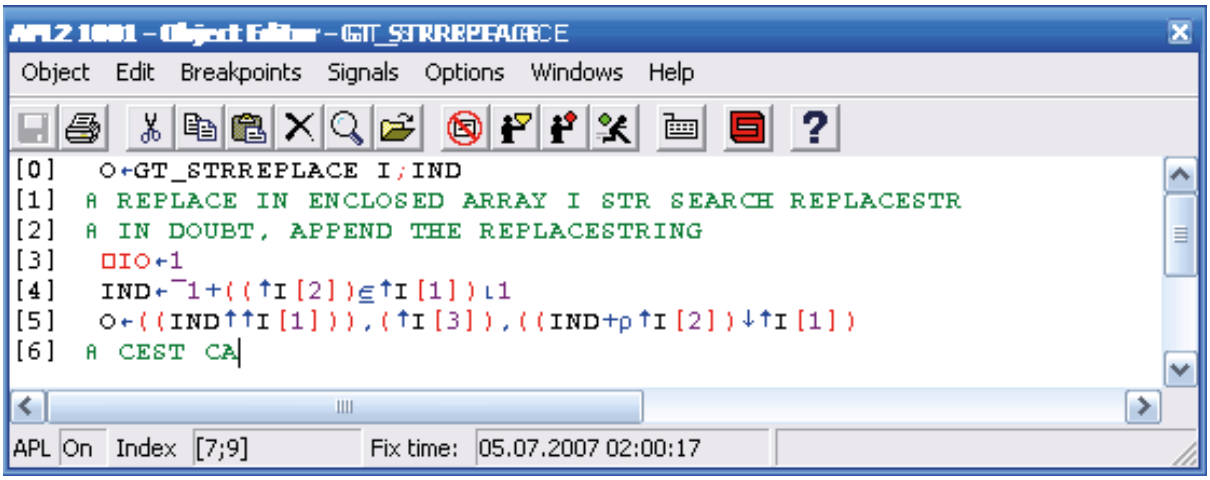


Abb. 4: Ersetzungsfunktion für einen Suchbegriff in einem HTML-Detail-Fragment

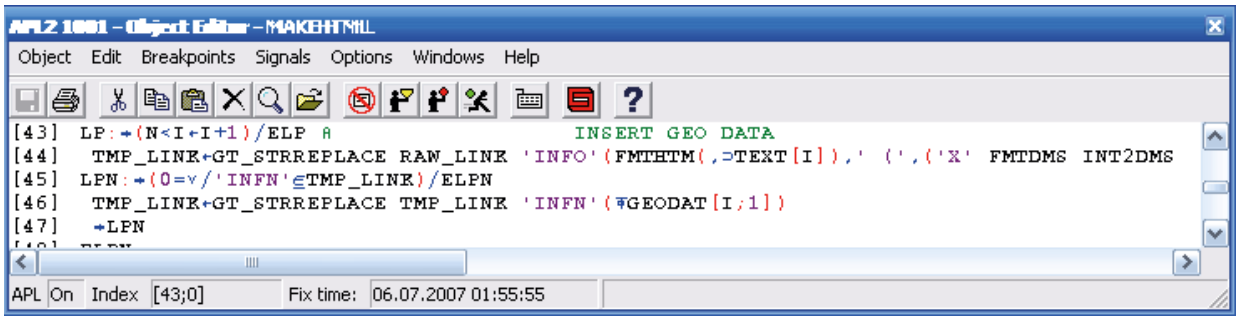


Abb. 5: Erweiterung der APL Anwendung

Auswertung vom GPS-Informationen als Ergebnis den Zugang zu verschiedenen Anwendungen für die Visualisierung der Ergebnisse zu Ermöglichen. Bei der vorgenannten Beispielen ist es etwa unmittelbar möglich, von der Stadtplan- zu der Satellitenansicht und umgekehrt hin- und herzuschalten.
Nehmen wir an, dass die Detail-Datei nun wie folgt aufgebaut ist:

```
<p><b><font color="#3333FF"><font size=+0>html from file INFO
<a href="http://www.mapquest.com/maps/map.adp?latlongtype=decimal&latitude=INFN
&longitude=INFE&zoom=9"
   target="anzeigefenster">MAPQUEST</a></font></font></b>
<b><font color="#3333FF"><font size=+0>
<a href="http://maps.google.com/maps?ie=UTF8&om=1&ll=INFN,INFE&t=k&z=17"
   target="anzeigefenster">SAT_VIEW</a></font></font></b>
<br> 
```

Dann werden in der erstellten HTML-Ansicht die Optionen Stadtplan und Satellitenansicht angeboten und beide Links funktionieren.

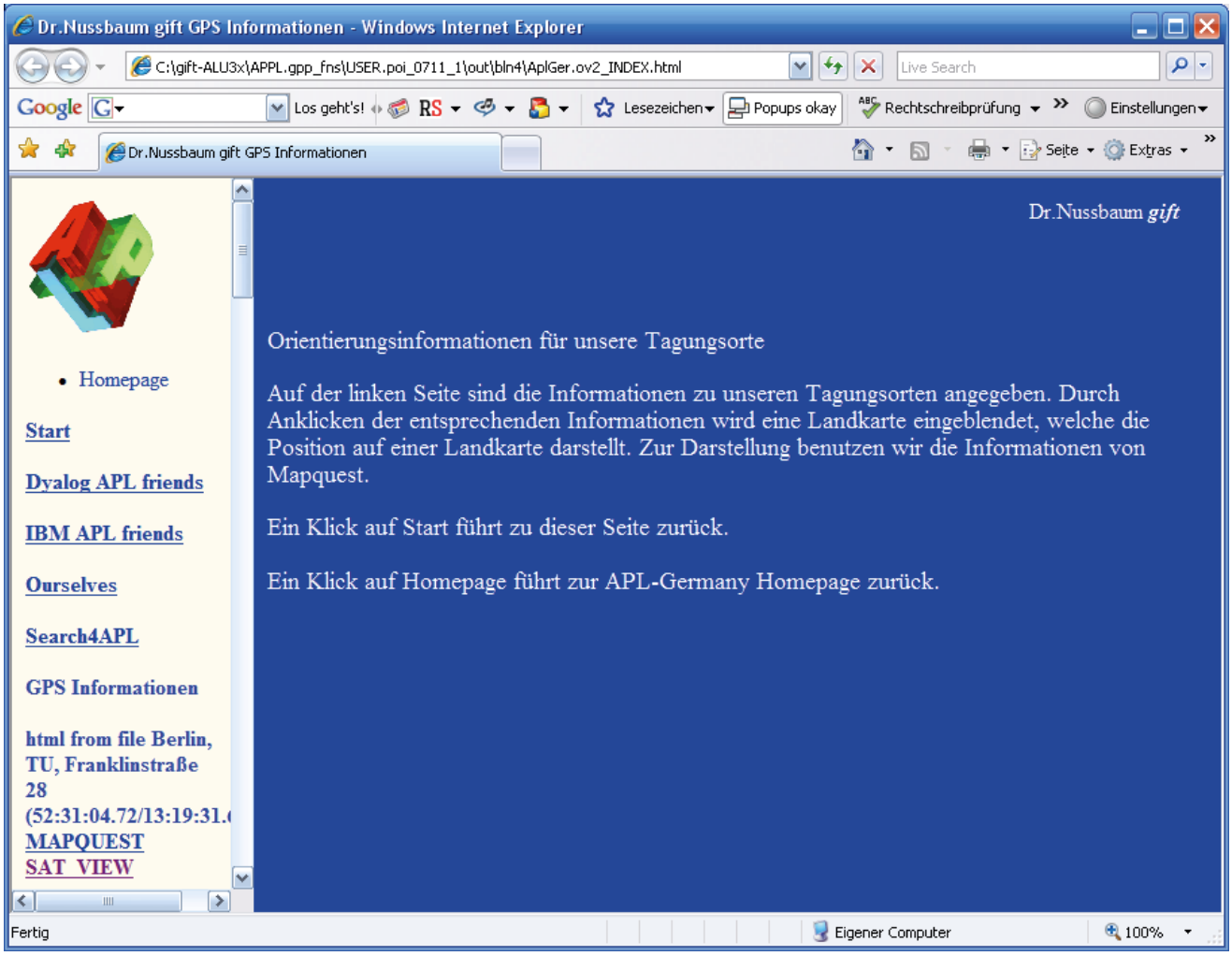


Abb. 6: HTML-Ansicht mit den Optionen Stadtplan und Satellitenansicht

5. Ausblick

Neben den öffentlich zugänglichen Visualisierungs Anwendungen für GPS-Informationen existieren auch noch weitere Anwendungen, mit denen aus den gefundenen Informationen weitere Erkenntnisse gewonnen werden können.

So haben die meisten Kommunen ihren geographischen Zuständigkeitsbereich digitalisiert. Je nach Problemstellung kann dann bei Kenntnis der GPS Information eine Ansicht auf Telefonzelle in der Nähe, jeden Kabel- oder Kanalschacht, Grundrisse besonderer Gebäude usw. dargestellt werden. Dies kann hilfreicher sein, als eine Besichtigung „vor Ort".

Hier ein öffentlich zugängliches Beispiel für ein kommunales geo Informationssystem: Der Musterort ist Hausen in der Nähe von Wiesbaden.

Die Grenzen einer stadtplan-orientierten Auswertung zeigt beispielsweise die nachfolgende Abbildung Abbildung 7 (nächste Seite, oben).

Auch den Satelliten sind Grenzen gesetzt. Insbesondere können sie nicht „unter die Dächer" oder „unter die Erde" sehen (siehe nachfolgende Abbildung 8, nächste Seite unten).

Dynamische HTML-Seiten mit GPS Info

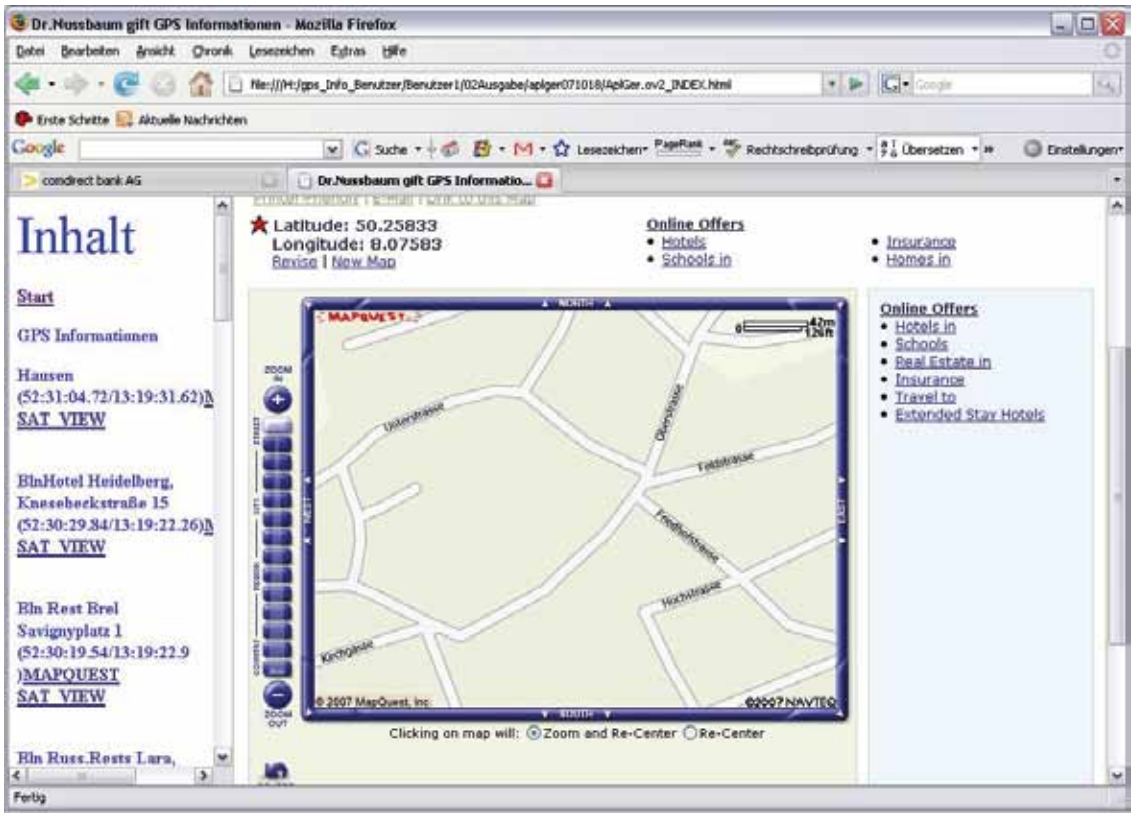


Abb. 7: Grenzen einer stadtplan-orientierten Auswertung

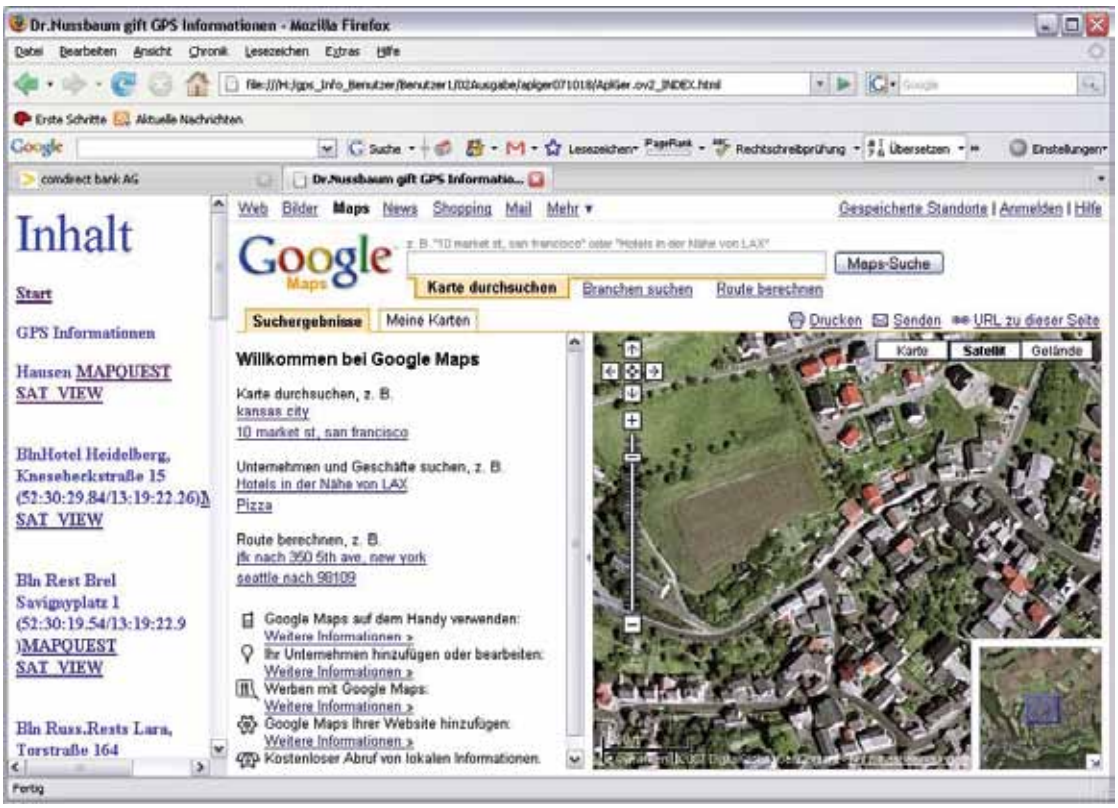


Abb. 8: Begrenzte Möglichkeiten der Satellitenansicht

Nachfolgende Abbildung 9 vermittelt ein Beispiel für die Darstellungsmöglichkeiten, die man mittels vor Ort digitalisierter Informationen realisieren kann:

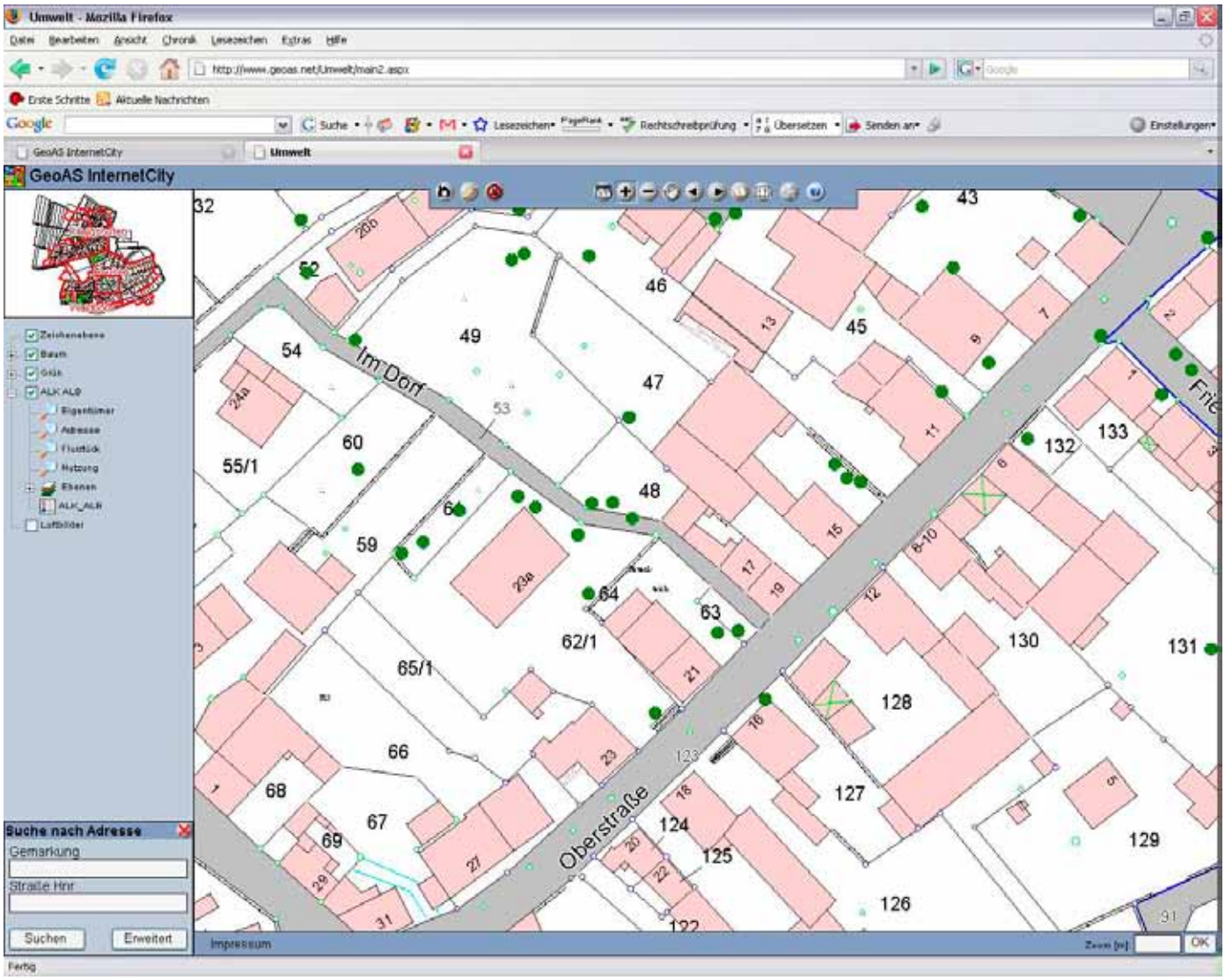
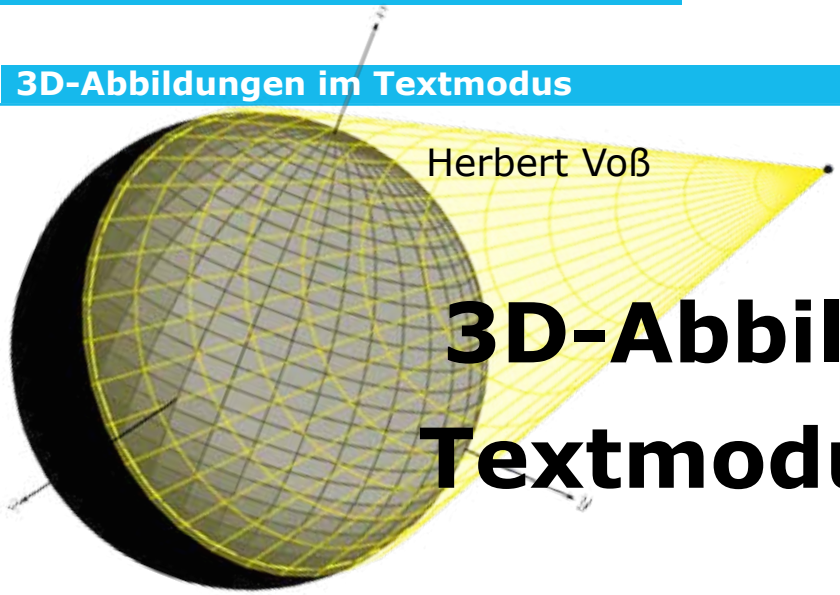


Abb. 9: Zielgruppenspezifische Möglichkeiten vor Ort digitalisierter Informationen

Das interessante an solchen Anwendungen ist, dass mittels der geographischen Informationen je nach Zielgruppe unterschiedliche Ansichten darstellbar sind. So wie in dem obigen Beispiel jeder Baum angegebenen ist (Ansicht Umweltamt), so kann auch jeder Kanalschacht oder jede Telefonzelle dargestellt werden.

Adresse des Autors:
Reiner Nussbaum, Dr. Nußbaum gift mbH, Buchenerstraße 78, D-68259 Mannheim, e-Mail: info@nussbaum-gift.de, Internet: www.nussbaum-gift.de/, Tel. 0621/7152190



Herbert Voß

3D-Abbildungen im Textmodus erstellen

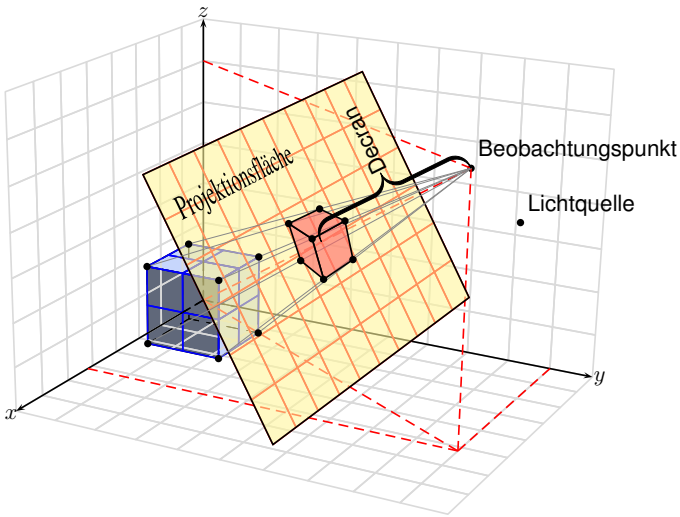
Das APL-Journal hat bereits in verschiedenen Ausgaben das Satzprogramm TEX behandelt. In diesem Artikel wird als Ergänzung zum letzten Heft die Möglichkeit gezeigt, mit normalen Textanweisungen qualitativ hochwertige dreidimensionale Abbildungen zu erreichen.

1. Einführung

Im letzten Heft hat Hendrick Rama ausführlich die Möglichkeiten von GUIGL und OpenGL zur Erstellung von dreidimensionalen Abbildungen beschrieben. Voraussetzung ist dabei eine hinreichende Kenntnis der entsprechenden Programmmodule. Immer dann, wenn man lediglich einzelne Grafiken für Veröffentlichungen benötigt, kann es hilfreich sein, wenn man diese auf Textebene beschreiben und erzeugen kann, um so bei der Erstellung eines Manuskripts Parameter schnell ändern zu können. Das Satzprogramm TEX erlaubt durch die Verwendung der grafischen Seitenbesschreibungssprache Post-Script die Formulierung von dreidimensionalen Abbildungen auf Textebene. Benötigt wird dazu lediglich das Paket `pst-solides3d`, welches die Algorithmen bereitstellt.

Die Verwendung von TEX bzw. dessen aktuellen Nachfolgers pdfTEX sollen hier nicht weiter beschrieben werden, es wird auf die entsprechenden Beiträge im APL-Journal [2, 3, 4] oder [6] verwiesen. Erscheinen in den Makrodefinitionen farbig hinterlegte Angaben, so sind diese grundsätzlich optional.

Die Festlegung des Betrachtungspunktes (Fluchtpunktes) erfolgt durch die Option `viewpoint=x y z` und ist bei der perspektivischen Betrachtung von dreidimensionalen Objekten immer anzugeben. Die folgende Abbildung veranschaulicht noch einmal die Betrachtungsweise eines Körpers.



Ausgehend vom Beobachtungspunkt `V` (`viewpoint`) wird das Objekt gesehen, hier der (blaue) Würfel. Senkrecht zur Verbindungslinie $\overline{OV}$ (Koordinatenursprung–Beobachtungspunkt) befindet sich der virtuelle zweidimensionale Bildschirm (Projektionsfläche), der vom Beobachtungspunkt `V` den standardmäßigen Abstand `Decran=50` hat.

Dieser Wert kann sowohl positiv als auch negativ sein. Unabhängig vom `viewpoint` kann eine Lichtquelle über Zuweisung von xyz-Koordinaten an `lightsrc` angeordnet werden, sodass eine entsprechende Schattenbildung möglich ist. Zu beachten ist allerdings, dass standardmäßig keine Lichtquelle vorgesehen ist.

Ein kleinerer Wert für `Decran` führt jetzt bei gleicher z-Koordinate ebenfalls zu einer kleineren Abbildung und ein größerer Wert dann zu einer entsprechend größeren Abbildung. Die Konstruktion folgt den einfachen Gesetzen des Strahlensatzes, unabhängig von den jeweiligen Werten von `viewpoint` und `Decran`, wobei allerdings `viewpoint= 0 0 0` und `Decran=0` keinen Sinn machen. Man beachte, dass das Zeichnen der Achse mit `\psaxes` nicht von `Decran` abhängt; das Makro aus dem Paket `pst-plot` wertet diesen Parameter nicht aus.

Mit dem Beobachtungspunkt kann man um die Originalabbildung in alle drei Richtungen beliebig „herumwandern". Dabei kann es nur dann zu Problemen in der Darstellung kommen, wenn eine Fläche nur mit einer Vorderseite definiert ist. Dies ist in der folgenden Darstellung bei dem Objekt `grille` zu sehen, welches bei einer Betrachtung von hinten nicht mehr sichtbar ist, wenn man nicht mit `action=draw` für eine „Durchsichtigkeit" des Körpers sorgt. In der Abbildungsmitte sind die aktuellen Koordinaten des `viewpoint` angegeben.

Ausgehend von `viewpoint=0 0 20`, wird die x-Koordinate um jeweils 10 Längeneinheiten vergrößert. Aufgrund der zentrischen Betrachtungsweise sieht man auch im ersten Beispiel nicht direkt auf die einzelnen Körper, obwohl man direkt von oben auf die Bildmitte sieht. Nur dieser Strahl steht in der linken Abbildung senkrecht auf der Bildebene. Durch die Vergrößerung der x-Koordinate bei gleich bleibender z-Koordinate bleibt die Anordnung parallel zur y-Achse. Durch den jeweils größer werdenden Abstand zwischen Betrachtungspunkt und Koordinatenursprung ($d=\sqrt{x_V^2+y_V^2+z_V^2}$) bei gleich bleibendem Wert für `Decran` ergibt sich folgerichtig ein kleineres Bild des Originals.

In den nächsten drei Beispielen wird jetzt ausgehend von der oberen rechten Abbildung die y-Koordinate in drei Schritten verändert, sodass in der rechten Abbildung bei einem Betrachtungspunkt

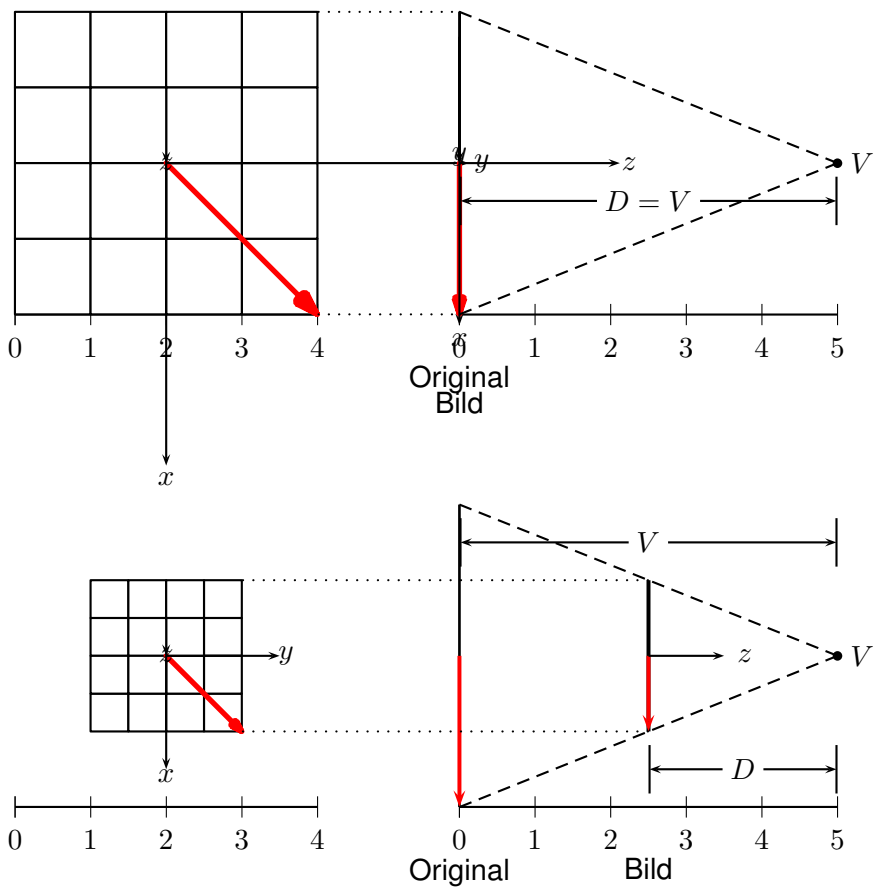


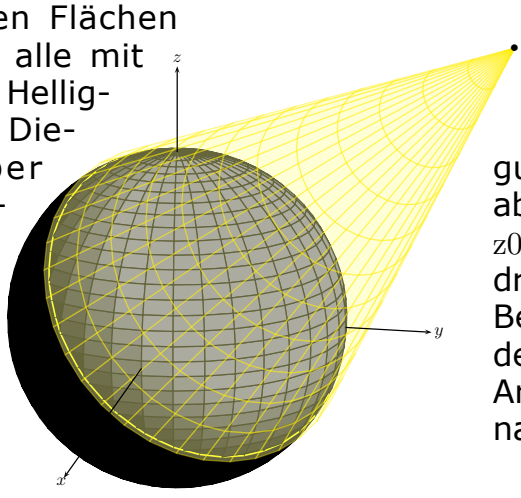
Abbildung 1: Bedeutung von Decran und viewpoint

von `viewpoint=20 20 20` eine Betrachtung unter 45 Grad erfolgt.

Die letzten drei Beispiele zeigen die Auswirkung von negativen Werten für die einzelnen Koordinaten. Ausgehend von der rechten Abbildung des obigen Beispiels wird nur das Vorzeichen der x-Koordinate verändert, die Abbildung wird hier bei gleichem Abstand von der anderen x-Seite betrachtet. Die nächsten beiden Beispiele ändern dann jeweils das Vorzeichen der y- und der z-Koordinate.

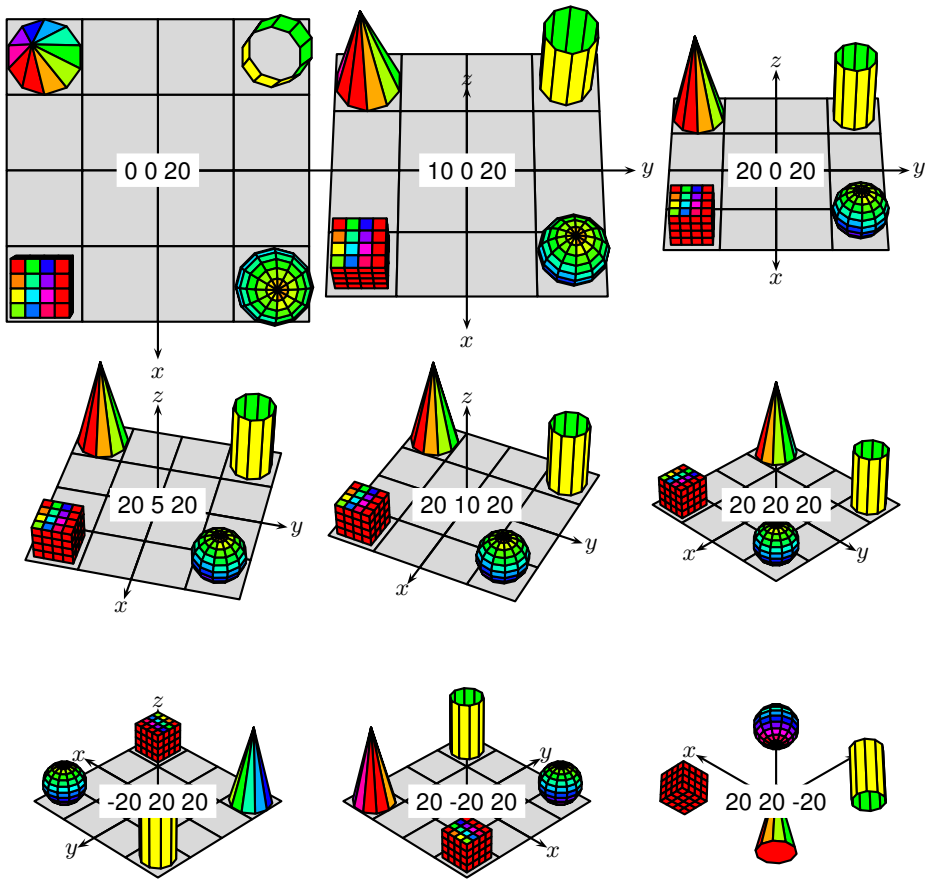
Lichtquellen können auch lokal für einzelne Objekte verändert oder auch ausgeschaltet werden. Letztgenanntes erfolgt durch Zuweisung eines leeren Wertes an `lightsrc`, wovon das folgende Beispiel gebrauch macht. Dadurch kann der transparente Lichtkegel mit hellerer Farbe gezeichnet werden. Praktisch wäre die Außenseite des hier sichtbaren Kegels „außerhalb" des eigentlichen Lichtkegels.

Werden die Koordinaten der Lichtquelle auf `viewpoint` gesetzt, so entsteht zwar keine Schattenwirkung, die sichtbaren Flächen werden aber alle mit der gleichen Helligkeit erstellt. Diese kann über den Parameter `lightintensity` beeinflusst werden, der mit dem Wert 2 vorbesetzt ist.



\psSolid

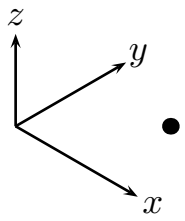
Die Zahl der zur Verfügung stehenden Makros ist relativ gering, da die meisten Objekte durch Zuweisung eines Parameters auf PostScript-Ebene erstellt werden. `\psSolid` ist das eigentliche Hauptmakro des Pakets und kann über seine vielen optionalen Argumente alle dreidimensionalen Objekte abbilden. Das optionale Zahlentripel (x0, y0, z0) kennzeichnet innerhalb des aktuellen dreidimensionalen Koordinatensystems den Bezugspunkt des Objekts, beispielsweise den Mittelpunkt einer Kugel. Fehlt dieses Argument, so ist automatisch der Koordinatenursprung auch der Bezugspunkt.

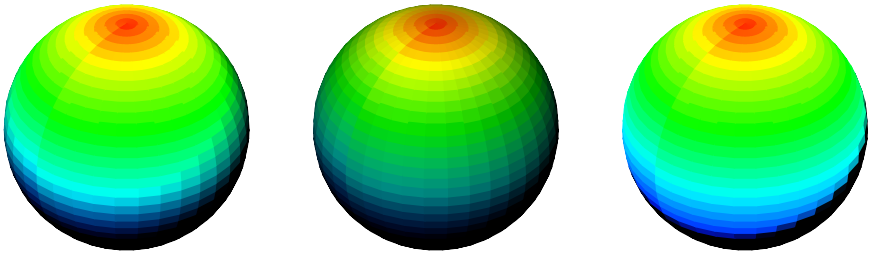


\psSolid[Optionen] (x_0, y_0, z_0)

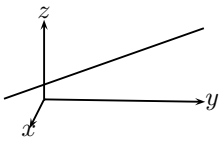
Die folgende Zusammenstellung zeigt die jeweiligen Basisobjekte, deren Liste beliebig durch eigene Kreationen erweitert werden kann.

Abbildung 2: Demonstration des Parameters lightintensity für die Werte 2 (Standard), 1 und 10.

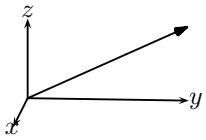




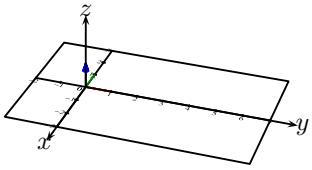
\psSolid[object=point]



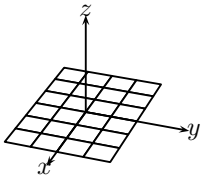
\psSolid[*object=line*]



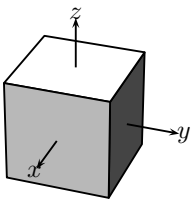
\psSolid[*object=vector*]



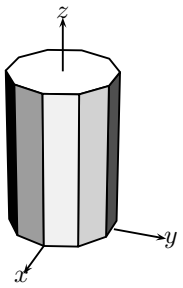
\psSolid[*object=plan,base=-2 7 -3 3*]



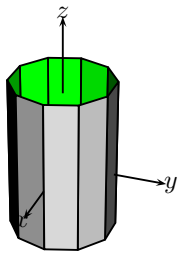
\psSolid[*object=grille,base=-3 3 -2 2*]



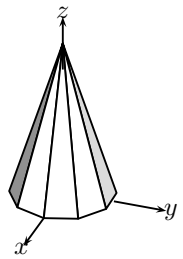
\psSolid[*object=cube*]



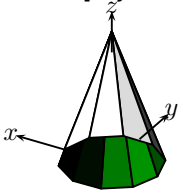
\psSolid[*object=cylindre*]



\psSolid[*object=cylindrecreu*

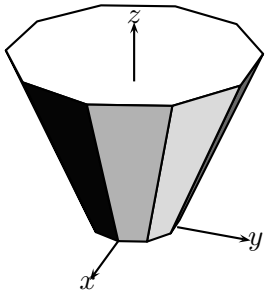


\psSolid[*object=cone*]

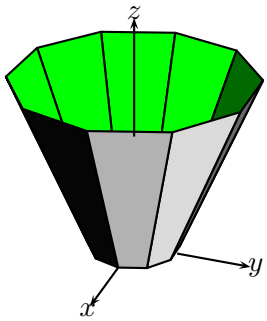


\psSolid[*object=conecreux*]

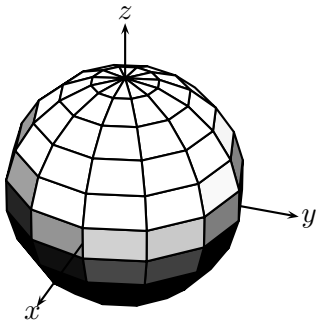
3D-Abbildungen im Textmodus



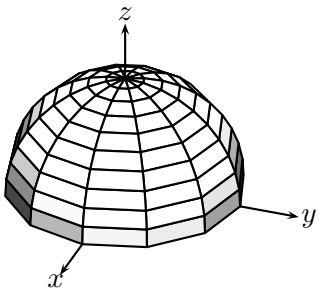
\psSolid[*object=tronccone*]



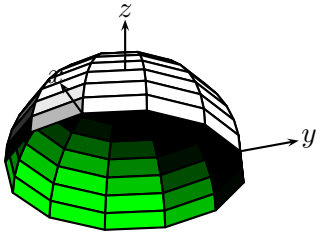
\psSolid[*object=troncconecreux*]



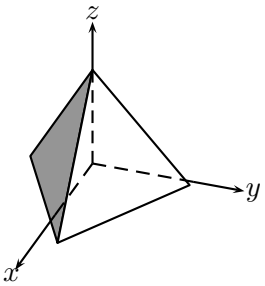
\psSolid[*object=sphere*]



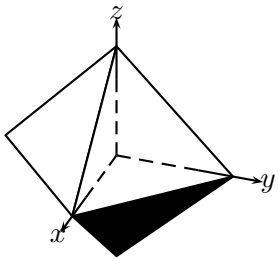
\psSolid[*object=calottesphere*]



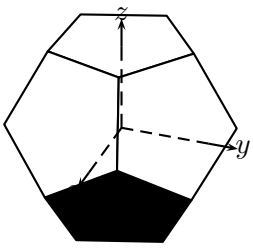
\psSolid[*object=calottespherecreuse*]



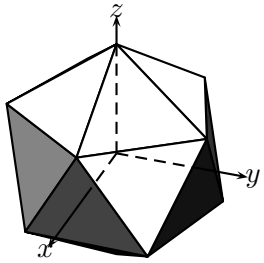
\psSolid[*object=tetrahedron*]



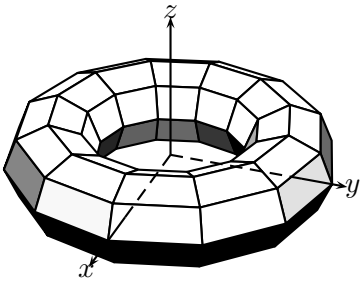
\psSolid[*object=octahedron*]



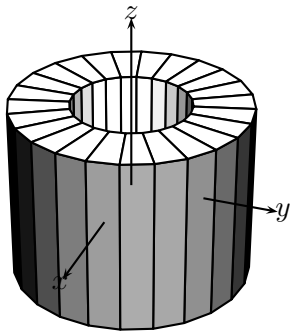
\psSolid[*object=dodecahedron*]



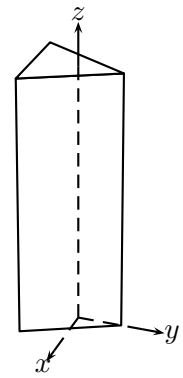
\psSolid[*object=icosahedron*]



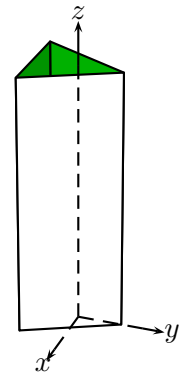
\psSolid[*object=tore*]



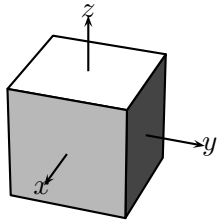
\psSolid[*object=anneau*]



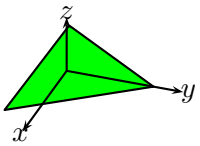
\psSolid[*object=prisme*]



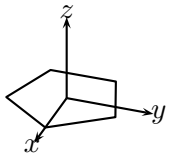
\psSolid[*object=prismecreux*]



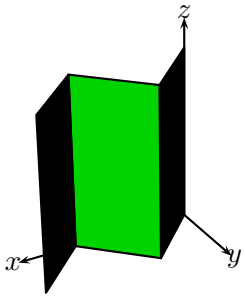
\psSolid[object=parallelepiped]



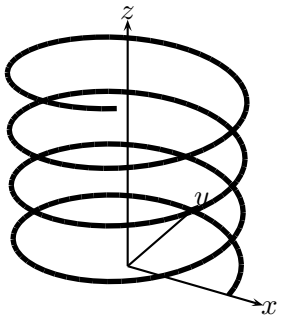
\psSolid[*object=face*]



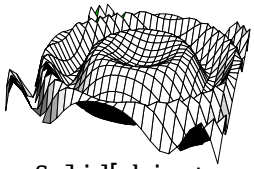
\psSolid[object=polygoneregulier,
ngrid=...]



\psSolid[*object=ruban*]

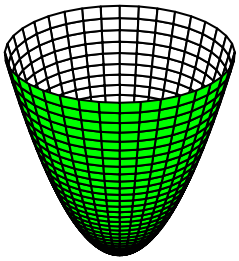


\psSolid[object=courbe,function=...,
range=...]

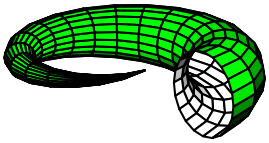


\psSolid[object=surface,function=...,
ngrid=dx dy]

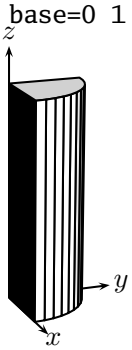
3D-Abbildungen im Textmodus



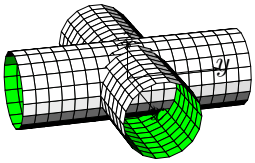
```
\psSolid[object=surface*,function=...,
     ngrid=...]
```



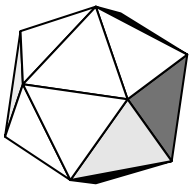
```
\psSolid[object=surfaceparametree,
     function=...,
     base=0 1 0 2 pi mul, ngrid=20
```



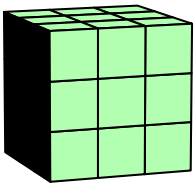

```
\psSolid[object=pie]
```



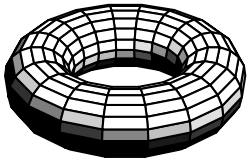
```
\psSolid[object=fusion]
```



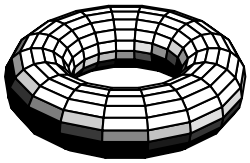
```
\psSolid[object=geode]
```



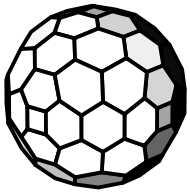
```
\psSolid[object=load]
```



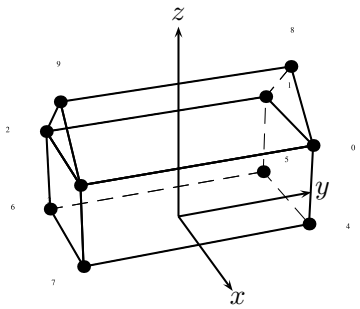
```
\psSolid[object=offfile,file=Datei]
```



```
\psSolid[object=objfile,file=Datei]
```



```
\psSolid[object=datfile,file=Datei]
```




```
\psSolid[object=new,sommets=...,
     faces={[...]},...
```

\axesIIID

Die Koordinatenachsen werden ausgehend vom Koordinatenursprung als gestrichelte Linien bis zum Punkt $P_0(x_0, y_0, z_0)$ und davon ausgehend als durchgehende Linien bis zum Endpunkt $P_1(x_1, y_1, z_1)$ gezeichnet.

`\axesIIID[`Optionen`]` $((x_0, y_0, z_0))(x_1, y_1, z_1)$

Die Angabe zweier Zahlentripel hat Auswirkungen auf die Linienform. Vom Ursprung bis zum ersten angegebenen Punkt werden gestrichelte und vom ersten zum zweiten angegebenen Punkt durchgehende Linien erstellt. Mit `showOrigin=false` können die

Name	Werte	Vorgabe	Bedeutung
showOrigin	Boolean	true	Zeichnet gestrichelte Linien vom Koordinatenursprung bis zum ersten Zahlentripel des Makros.
mathLabel	Boolean	true	Setzt die Label im mathematischen Modus, was bei der Zuweisung von Befehlen an axisemph zu beachten ist.
Axisnames	{x,y,z}	{x,y,z}	Achsenbezeichungen, müssen in geschweifte Klammern gesetzt werden, beispielsweise axisnames={u,v,w}
Axisemph	Makro	{}	Kann für die Schriftgröße und/oder den Schriftstil verwendet werden. In Abhängigkeit vom gewählten Text- oder Mathematikmodus sind unterschiedliche Makros zulässig.
Labelsep	Wert Einheit	10pt	Abstand der Achsennamen von der Achsenspitze relativ zur Achsenrichtung.

Tabelle 1: Zusammenstellung der relevanten Parameter für das `\axesIIID`*-Makro*

gestrichelten Linien unterdrückt werden. Die Koordinatenachsen werden grundsätzlich über eine bereits bestehende Grafik gezeichnet; der „hidden line"-Algorithmus kommt hier nicht zur Anwendung!

\gridIIID

Dieses Makro erzeugt ein Drahtmodell einer Fläche oder eines Quaders zur Maßeinteilung der drei Achsen, wobei standardmäßig der z-Achse ein Bereich von $[-4 \ldots 4]$ zugeordnet wird. Über die optionalen Argumente für `zMin` und `zMax` kann dieser vorgegebene Bereich geändert werden. Die Label werden nicht an den eigentlichen Koordinatenachsen, sondern außerhalb des Gitters angebracht. Die beiden notwendigen Parameter für `\gridIIID` kennzeichnen jeweils das x- und das y-Intervall.

`\gridIIID`[Optionen] $(x_{min}, x_{max})(y_{min}, y_{max})$ [dx,dy,dz]

Die optionalen Angaben von [dx,dy,dz] ermöglichen eine individuelle Angabe für die Länge der einzelnen Achsen. Innerhalb des Gitters werden diese gestrichelt dargestellt und außerhalb mit einer durchgezogenen Linie, deren Länge durch [dx,dy,dz] gegeben ist, standardmäßig [2,2,3]. Die weiteren möglichen Optionen sind in der folgenden Tabelle 2 zusammengestellt.

Name	Werte	Vorgabe	Bedeutung
spotX	r|u|l|d|ul|cl|bl| dl|ub|cb|bb|db| uc|cc|bc|dc|ur| cr|br|dr	dr	Position für x
spotY	*dito*	dl	Position für y
spotZ	*dito*	l	Position für z
Zmin	*Wert*	-4	Koordinatengitter
Zmax	*Wert*	4	Koordinatengitter
stepX	*Wert*	1	x-Schrittweite der Ticks
stepY	*Wert*	1	y-Schrittweite der Ticks
stepZ	*Wert*	1	z-Schrittweite der Ticks
ticklength	*wert*	0.2	Strichlänge der Ticks

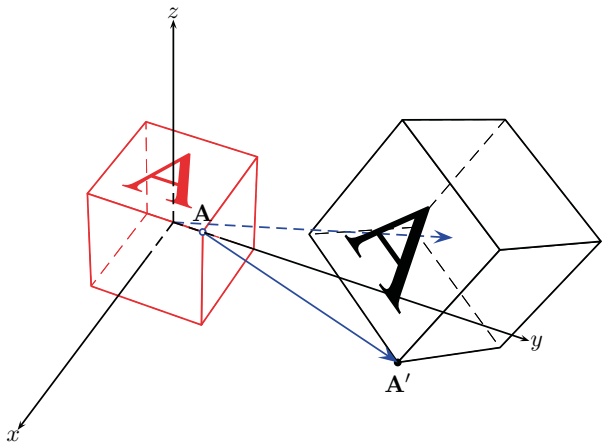
Tabelle 2: Zusammenfassung sinnvoller Parameter für `\gridIIID`

Punkttransformation mit \psTransformPoint

\psTransformPoint[Optionen] (x, y, z) (dx, dy, dz){Name}

Der Punkt P (x, y, z) kann über das optionale Argument durch Zuweisung von Werten an die Parameter RotX, RotY und RotZ rotiert und anschließend durch (dx, dy, dz) verschoben werden. Die Reihenfolge der Rotation ist dabei per Definition x–y–z. Das Ergebnis der Transformation wird in dem Knoten Name gespeichert.
Im folgenden Beispiel entsteht der gleich große zweite Würfel mit einem Mittelpunkt in M (7.5, 11.25, 10) durch Rotation um alle drei Achsen. Die Koordinaten des analogen Punktes A′ sind daher nicht ohne weiteres bekannt. Mithilfe von \psTransformPoint lässt sich nun der Originalpunkt Aerst mit den gleichen Winkeln rotieren und dann um den Mittelpunktsvektor des zweiten Würfels verschieben. Dadurch erhält der Knoten A die Koordinaten des Eckpunktes.

```
\begin{pspicture}(-3,-3.5)(7,4)
\psset{unit=0.75,viewpoint=40 20 40,Decran=40}
\psSolid[object=cube,a=4,action=draw*,linecolor=
  red]%  kleiner Wuerfel
\psPoint(2,2,2){A}
\psSolid[object=cube,a=4,action=draw*,RotX=-30,
  RotY=60,RotZ=-60](7.5,11.25,10)% grosser
\psTransformPoint[RotX=-30,RotY=60,RotZ=-60](2 2
  2)(7.5,11.25,10){A'}
\psdot(A')\psline[linecolor=blue,arrowsize=0.3]{o
  ->}(A)(A')
\uput[-90](A'){$\mathbf{A'}$}\uput[u](A){$\mathbf{
  A}$}
\psLineIIID[linecolor=blue,linestyle=dashed,
  arrowsize=0.3]{->}(0,0,0)(7.5,11.25,10)
\psset{solidmemory,action=none}
\psSolid[object=cube,a=4,name=A1](0,0,0)
\psSolid[object=plan,definition=solidface,args=A1
  0,name=P0]
\psset{fontsize=100}
\psProjection[object=texte,linecolor=red,text=A,
  plan=P0]
\psSolid[object=cube,a=4,RotX=-30,RotY=60,RotZ
  =-60,name=A2](7.5,11.25,10)
\psSolid[object=plan,definition=solidface,args=A2
  0,name=P'0]
\psProjection[object=texte,text=A,plan=P'0]
\axesIIID(2,2,2)(10,12,8)
\end{pspicture}
```



Name	Werte	Vorgabe
object	Objektname	
definition	Objektname	
args	Wert	Wert
pos	ul|cl|bl|dl|	bl
	ur|cr|br|dr	
	uc|cc|bc|dc|	
	ur|cr|br|dr	
text	String	
name	String	
plan	Object	-
path	PS code	-
Planmarks	Boolean	false
showBase	Boolean	
base	x_0 x_1 y_0 y_1	-3 3 -2 2
range	x_0 x_1 y_0 y_1	-3 3 -2 2
fontsize	Wert	
resolution	Wert	
function	Funktion	-
transform	PS code	-

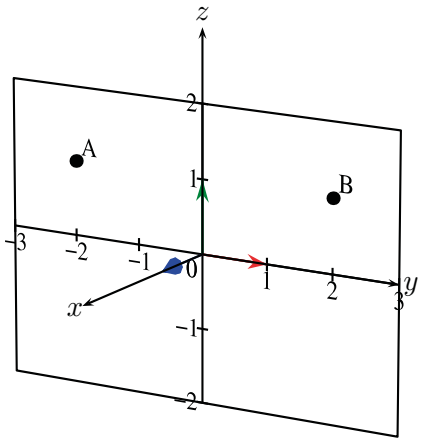
Tabelle 3: Zusammenfassung aller Parameter für pst-solides3d

Projektionen mit \psProjection

Das Makro erlaubt Projektionen von dreidimensionalen Objekten auf eine beliebige zweidimensionalen Ebene, die jeweils zuvor durch das Makro \psSolid festgelegt werden muss.

`\psProjection[`Optionen`]` (x, y)

Die Ebene wird durch ihre Gleichung $r \cdot x + s \cdot y + t \cdot z + b = 0$ festgelegt. Im folgenden Beispiel zeigt der Normalenvektor in die x-Richtung (1 0 0) und die Ebene geht durch den Ursprung, da die vierte Variable gleich Null ist. Dies entspricht der Gleichung $x = 0$. Zusätzlich wird die Ebene um 90 Grad gedreht. Die Ebene bekommt den willkürlich festgelegten Objektnamen Ebene, auf den dann mit `\psProjection` bezug genommen werden kann.




```
\begin{pspicture}(-3,-2.75)(4,3)
\psset{viewpoint=50 30 15,Decran=60,
  solidmemory}
\psSolid[object=plan,definition=equation,
  args={[1 0 0 0] 90},name=Ebene,planmarks,
    showBase]
\psset{plan=Ebene}
\psProjection[object=point,args=-2 1,text=A,pos=ur
  ]
\psProjection[object=point,text=B,pos=ur](2,1)
\axesIIID(3,3,3)
\end{pspicture}
```

Bei der Projektion von Vektoren stehen verschiedene Werte für den Parameter `definition` zur Verfügung:

`definition=vecteur`: Der Vektor $\overrightarrow{AB}$ benötigt zwei Argumente `args=A B`.

`definition=orthovecteur`: Der orthogonale Vektor zu $\vec{u}$ gleicher Länge benötigt ein Argument `args=u`.

`definition=normalize`: Der Normalenvektor $\|\vec{u}\|^{-1}\vec{u}$, falls $\vec{u} \neq \vec{0}$ und $\vec{0}$ benötigt ein Argument `args=u`.

`definition=addv`: Die Vektoraddition $\vec{u}+\vec{v}$ benötigt zwei Argumente `args=u v`.

`definition=subv`: Die Vektorsubtraktion $\vec{u}-\vec{v}$ benötigt zwei Argumente `args=u v`

`definition=mulv`: Die skalare Vektormultiplikation $c \cdot \vec{u}$ benötigt zwei Argumente `args=u c`.

Bei der Projektion von Geraden stehen verschiedene Werte für den Parameter `definition` zur Verfügung, die sich auf die üblichen Definitionen einer Gerade beziehen.

`definition=horizontale`: Eine horizontale Linie mit $y = b$ durch `args=b`.

`definition=verticale`: Eine vertikale Linie $x = a$ durch `args=a`.

`definition=paral`: Eine zu $d = f(x)$ parallele Linie, die durch den Punkt A geht, durch `args=d A`. Die Linie d muss als Objekt vorher definiert worden sein.

`definition=perp`: Eine zu $d = f(x)$ rechtwinklige Linie, die durch den Punkt A geht, durch `args=d A`. Die Linie d muss als Objekt vorher definiert worden sein.

`definition=mediatrice`: Eine senkrechte Gerade durch die Mitte eines Intervalls [AB] durch `args=A B`.

`definition=bissectrice`: Eine Winkelhalbierende zu $\widehat{ABC}$ durch `args=A B C`.

`definition=axesymdroite`: Eine symmetrische Linie zu d bezogen auf die Symmetrieachse D durch `args=d D`.

`definition=rotatedroite`: Eine um den Punkt I um den Winkel r (in Grad) rotierte Linie d durch `args=d I r`.

3D-Abbildungen im Textmodus

`definition=translatedroite`: Eine um den Vektor $\vec{u}$ verschobene Linie `d` durch `args=d u`.

Der einzige Unterschied zwischen Geraden und Linien ist, dass Linien grundsätzlich durch beliebig viele Punkte beschrieben werden. Dem Objekttyp `line` sind daher über das Argument die Koordinaten der Punkte oder die Punktnamen mitzuteilen: `args=A0 A1...An`. Gleiches gilt für das Objekt `polygone`, welches im Gegensatz zum Linienzug grundsätzlich am Ende geschlossen wird. Dazu wird der letzte Punkt durch eine direkte Linie mit dem ersten Punkt verbunden. Für das Polynom existieren die folgenden Definitionen, die teilweise auch auf das Objekt `line` angewendet werden können.

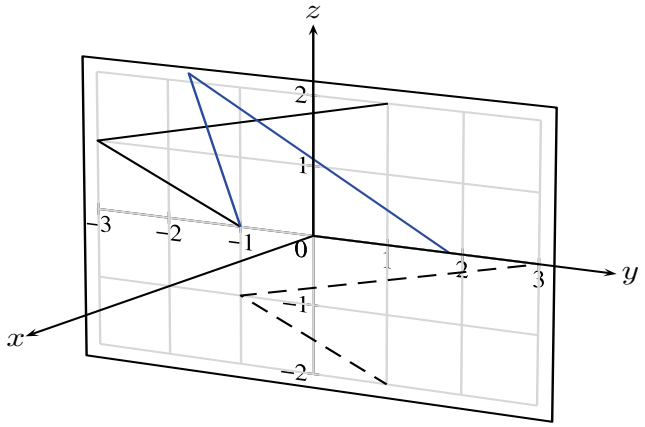
`definition=translatepol`: Ein um den Vektor $\vec{u}$ verschobenes Polygon `pol` durch `args=pol u`.

`definition=rotatepol`: Ein um den Punkt `I` um den Winkel α (in Grad) rotiertes Polygon durch `args=pol I` α .

`definition=hompol`: Ein mit dem Streckzentrum `I` um den Wert α gestrecktes Polygon durch `args=pol I` α .

`definition=sympol`: Ein zum Punkt `I` symmetrisches Polygon durch `args=pol I`.

`definition=axesympol`: Ein symmetrisches Polygon bezogen auf die Symmetrieachse D durch `args=pol D`.




```
\begin{pspicture}(-3,-3)(4,3.5)%
\psset{lightsrc=50 20 20,Decran=60,
  viewpoint=50 30 15,solidmemory}
\psSolid[object=plan,definition=equation,
  name=Ebene,args={[1 0 0 0] 90},
  base=-3.2 3.2 -2.2 2.2,planmarks]
\psset{plan=Ebene}
\psSolid[object=plan,args=Ebene,plangrid,
  linecolor=gray!40,action=none]
\psProjection[object=line,
  args=-1 0 -3 1 1 2,name=P]
\psProjection[object=line,args=P -1 0 -45,
  definition=rotatepol,linecolor=blue]
\psProjection[object=line,linestyle=dashed,
  definition={2 -2 addv} papply,args=P]
\axesIIID(6,4,3)
\end{pspicture}
```

Bei der Projektion von Kreisen oder Kreisbögen stehen zwei verschiedene Werte für den Parameter `definition` zur Verfügung.

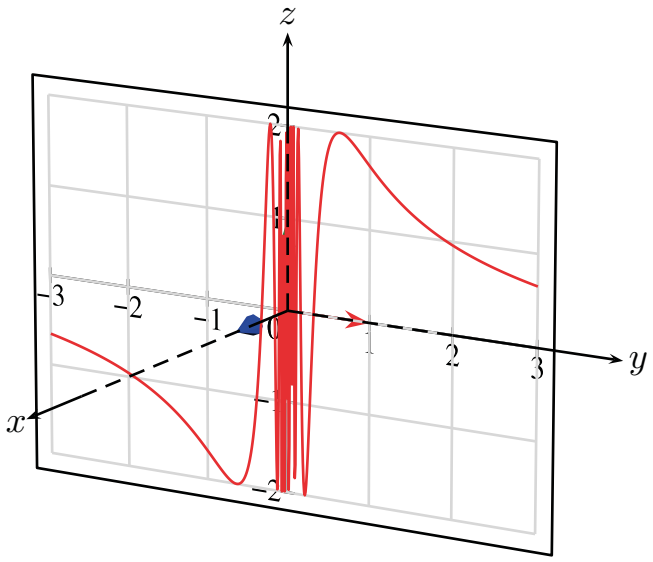
`definition=ABcercle`: Ein Kreis durch die drei Punkte `A`, `B` und `C` durch `args=A B C`.

`definition=diamcercle`: Ein Kreis mit dem Durchmesser $\overline{AB}$ durch `args=A B`.

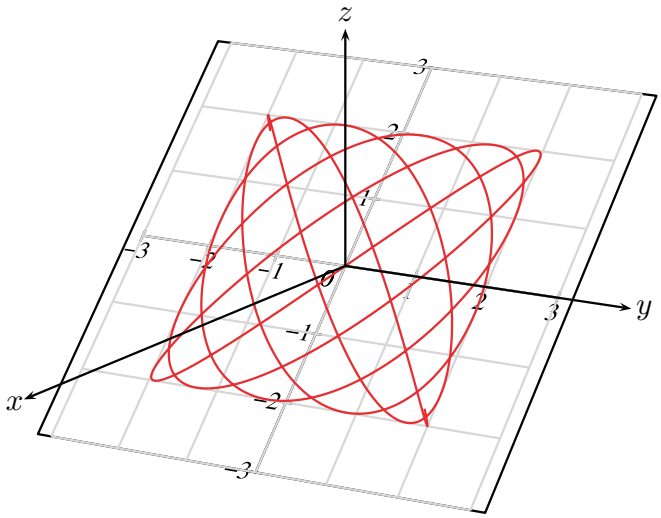
Rechte Winkel lassen sich einfach mit dem Objekttyp `rightangle` markieren, welcher drei Punkte als Parameter erwartet.

Die Projektion von Funktionen erfolgt über die Definition der Funktion mit dem Makro `\defFunction`. Über das optionale Argument kann festgelegt werden, ob die Beschreibung der Funktion in algebraischer oder PostScript-Notation (Standard) erfolgt. Für die Variable `t` kann auch jeder beliebige andere Buchstabe eingesetzt werden, der allerdings dem entsprechen muss, der durch den ersten Parameter deklariert worden ist.

```
\defFunction[Optionen] {Funktionsname}
      (x){f(x)}{}{}
\defFunction[Optionen] {Funktionsname}
      (t){x(t)}{y(t)}{}
\defFunction[Optionen] {Funktionsname}
      (t){x(t)}{y(t)}{z(t)}
```

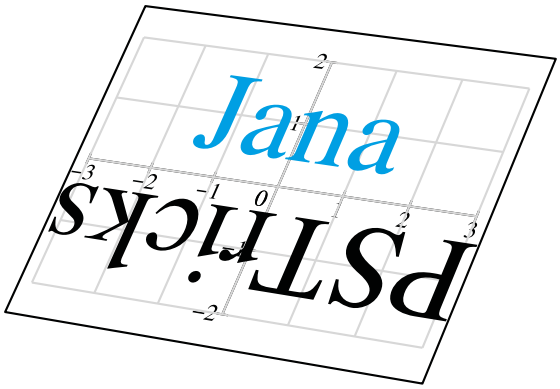



```
\begin{pspicture}[lightsrc=50 20 20,viewpoint=50
  30 15,Decran=60,solidmemory](-3,-3)(4,3.5)
\defFunction[algebraic]{1_sin}(x){2*sin(1/x)}{}{}
\psSolid[object=plan,definition=equation,args={[1
  0 0 0] 90},base=-3.2 3.2 -2.2 2.2,
   planmarks,showBase,name=Ebene]
\psset{plan=Ebene}
\psSolid[object=plan,args=Ebene,linecolor=gray!40,
  plangrid,action=none]
\psProjection[object=courbe,linecolor=red,range=-3
   3,resolution=720,function=1_sin]
\axesIIID(4,2,2)(5,4,3)
\end{pspicture}
```



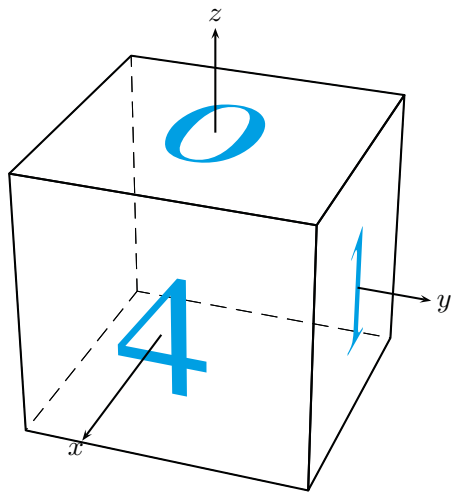

```
\begin{pspicture}[lightsrc=50 20 20,viewpoint=50
  30 15,Decran=60,solidmemory](-4.5,-3)(3,3.5)
\defFunction[algebraic]{F}(t){2*sin(0.57735*t)}{2*
  sin(0.707*t)}{}
\psSolid[object=plan,definition=equation,args={[1
  0 1 0] 90},base=-3.2 3.2 -3 3,
   planmarks,name=Ebene]
\psset{plan=Ebene}
\psSolid[object=plan,args=Ebene,linecolor=gray!40,
  plangrid,action=none]
\psProjection[object=courbeR2,range=-25.12 25.12,
  resolution=720,normal=1 1 2,
    linecolor=red,function=F]
\axesIIID(7,4,3)
\end{pspicture}
```

In einigen Beispielen wurde bereits die Position von Textelementen (Buchstaben) mit dem Makro `\psProjection` gezeigt. Diese können leicht auf längeren Text übertragen werden.




```
\begin{pspicture}(-4,-3)(4,3)%
\psset{lightsrc=50 20 20,Decran=60,viewpoint=50 30
   15,solidmemory}
\psSolid[object=plan,definition=equation,args={[1
  0 1 0] 90},planmarks,
  base=-3.2 3.2 -2.5 2.5,name=Ebene]
\psset{plan=Ebene}
\psSolid[object=plan,args=Ebene,plangrid,linecolor
  =gray!40,action=none]
\psProjection[object=texte,fontsize=50,linecolor=
  cyan,text=Jana](0,1)
\psProjection[object=texte,fontsize=50,text=
  PSTricks,phi=180](0,-1)
\end{pspicture}
```

3D-Abbildungen im Textmodus




```
\begin{pspicture}(-4,-3)(3,4)
\psset{unit=0.5,fontsize=150,Decran=50,
  viewpoint=50 20 30 rtp2xyz,solidmemory}
\psSolid[object=cube,a=8,action=draw,name=A]
\psSolid[object=plan,action=none,
  definition=solidface,args=A 0,name=P0]
\psProjection[object=texte,text=0,
  plan=P0,linecolor=cyan]
\psSolid[object=plan,action=none,
  definition=solidface,args=A 1,name=P1]
\psProjection[object=texte,text=1,
  plan=P1,phi=180,linecolor=cyan]
\psSolid[object=plan,action=none,
  definition=solidface,args=A 4,name=P4]
\psProjection[object=texte,text=4,
  plan=P4,phi=-90,linecolor=cyan]
\axesIIID[showOrigin=false](4,4,4)(9,6,7)
\end{pspicture}
```

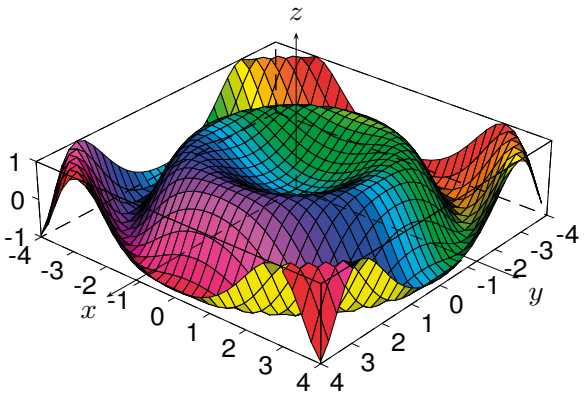
Darstellung mathematischer Funktionen mit \psSurface

Einige Beispiele haben bereits gezeigt, wie man mit dem Makro `\psSolid` zwei- oder dreidimensionale Funktionen berechnen und darstellen kann. Das Makro `\psSurface` bringt keine erweiterten Funktionen, erleichtert jedoch die Anwendung für den Sonderfall der Funktionsdarstellung. `\psSurface`[*Optionen*] $(x_0, x_1)(y_0, y_1)$ {*Funktion* $z = f(x, y)$}

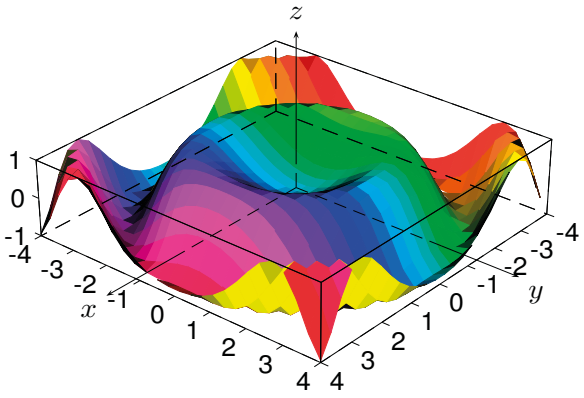
Die Funktion muss standardmäßig in der PostScript-Notation oder alternativ mit der Option `algebraic` in algebraischer Notation eingegeben werden. Sinnvolle Optionen für das Makro sind in der Tabelle 4 zusammengestellt. Optionen für das Koordinatengitter findet man in Tabelle 2.

Name	*Werte*	*Vorgabe*	*Bedeutung*
`algebraic`	*Boolean*	`false`	Gleichungsnotation
`ngrid`	*n0 n1*	-	d_x und d_y, wenn <0 N_x und N_y, d wenn > 0
`Grid`	*Boolean*	`true`	Gitterlinien
`Axesboxed`	*Boolean*	`false`	Koordinatenachsen

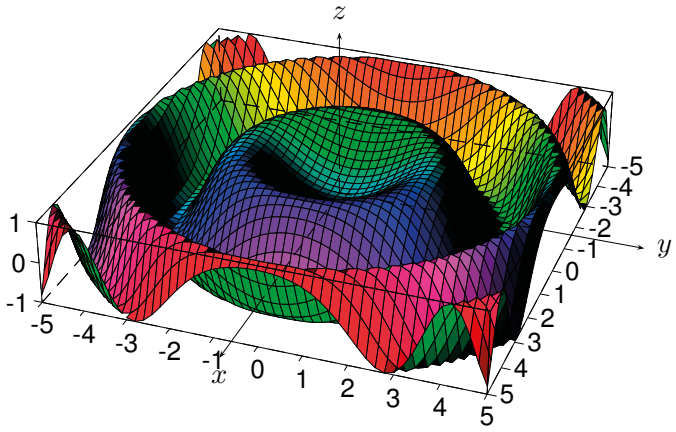
Tabelle 4: Zusammenfassung sinnvoller Parameter für \psSurface



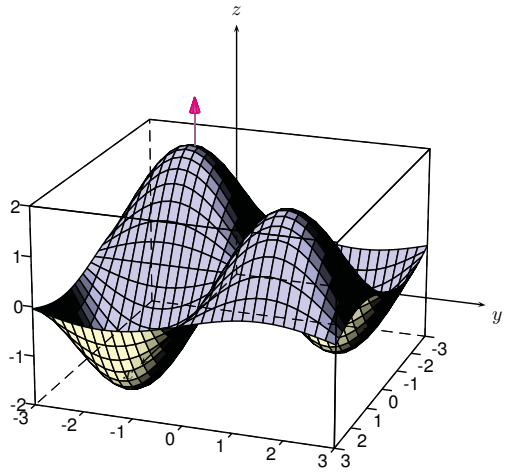

```
\psset{viewpoint=50 40 30 rtp2xyz, Decran=30,
  lightsrc=viewpoint}
\begin{pspicture}(-4,-3)(4,2.75)
\psSurface[ngrid=.25 .25,incolor=yellow,
  linewidth=0.01,axesboxed,algebraic,
  hue=0 1,Zmin=-1,Zmax=1](-4,-4)(4,4)%
    {sin((x^2+y^2)/3)}
\end{pspicture}
```



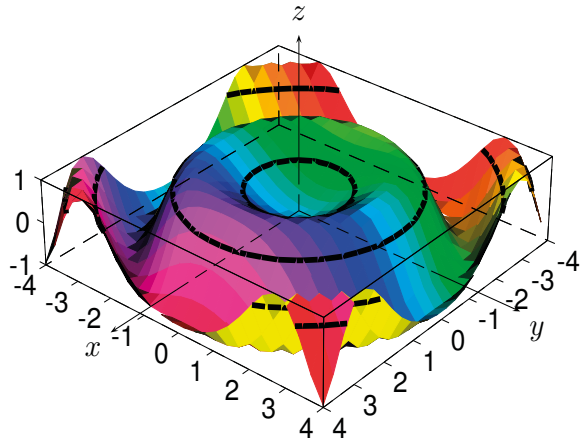

```
\psset{lightsrc=30 30 25,
  viewpoint=50 40 30 rtp2xyz,Decran=30}
\begin{pspicture}(-4,-3)(4,2.75)
\psSurface[fillcolor=red!50,ngrid=25 25,
  incolor=yellow,linewidth=0.01,grid,
  axesboxed,hue=0 1,Zmin=-1,Zmax=1]%
  (-4,-4)(4,4){ y dup mul x dup mul
    add 3 div RadtoDeg sin }
\end{pspicture}
```



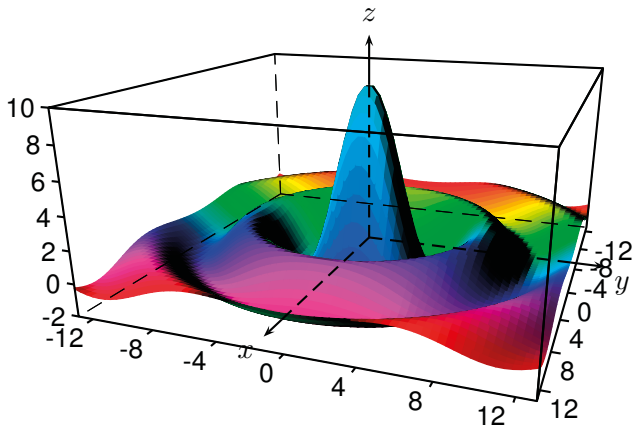
```
\psset{lightsrc=30 -10 10,viewpoint=50 20 30 rtp2
  xyz,Decran=30}
\begin{pspicture}(-4,-3)(4,2.75)
\psSurface[ngrid=.2 .2,algebraic,axesboxed,Zmin
  =-1,Zmax=1,linewidth=0.01,
  spotX=r,spotY=d,spotZ=l,hue=0 1](-5,-5)(5,5){
    sin((x^2+y^2)/3) }
\end{pspicture}
```



```
\def\NormalSin#1#2#3{\psset{linecolor=#3}\pstVerb{
 /xP #1 def /yP #2 def /zP 2 #1 Sin mul #2 Sin mul
    def
 /normaleX 2 #1 Cos mul #2 Sin mul neg def /
   normaleY 2 #2 Cos mul #1 Sin mul neg def
 /Norme normaleX dup mul normaleY dup mul add 1
   add sqrt def
 /Nx normaleX Norme div def /Ny normaleY Norme div
    def /Nz 1 Norme div def}%
\psSolid[object=vecteur,args=Nx Ny Nz](xP,yP,zP)}
\psset{viewpoint=50 20 20 rtp2xyz,Decran=50}
\begin{pspicture}(-5,-4)(5,5.5)
\psSurface[algebraic,ngrid=0.2 0.2,incolor=yellow
  !20,axesboxed,Zmin=-2,Zmax=2,
  fillcolor=blue!20](-3,-3)(3,3){2*sin(x)*sin(y)}%
\NormalSin{1.57}{0.75}{red}\NormalSin{1.57}{1.57}{
  blue}%
\NormalSin{2}{1.57}{green}\NormalSin
  {-1.57}{-1.57}{magenta}
\end{pspicture}
```



```
\psset{lightsrc=30 30 25,
  viewpoint=50 40 30 rtp2xyz,Decran=30}
\begin{pspicture}(-4.5,-3)(4,2.75)
\psSurface[fillcolor=red!50,ngrid=25 25,
  intersectionplan={[0 0 1 -0.5]},
  intersectioncolor=(noir),grid,
  intersectionlinewidth=2,Zmin=-1,
  Zmax=1,intersectiontype=0,
  incolor=yellow,linewidth=0.01,
  axesboxed,hue=0 1,](-4,-4)(4,4)%
  { y dup mul x dup mul add 3 div
    RadtoDeg sin }
\end{pspicture}
```



```
\psset{lightsrc=30 -10 10,
  viewpoint=50 20 20 rtp2xyz,
  Decran=10}
\begin{pspicture}(-5,-3)(4,2.75)
\psSurface[ngrid=0.4 0.4,algebraic,
  axesboxed,Zmin=-2,Zmax=10,
  grid=false,ticklength=0.5,
  stepX=4,stepY=4,stepZ=2,hue=0 1]%
   (-13,-13)(13,13)%
    {10*sin(sqrt((x^2+y^2)))%
      /(sqrt(x^2+y^2)) }
\end{pspicture}
```

2. Parameter

Die Zahl der optionalen Parameter ist sehr groß und man kann dabei leicht den Überblick verlieren. Eine Zusammenstellung aller Parameter, die für das Makro `\psSolid` relevant sind, findet man in [5].

Aktionen

Der Parameter `action` bestimmt, wie mit dem Objekt intern verfahren werden soll. `action` kann einen der folgenden Werte annehmen:

`none` Das Objekt verbleibt lediglich im Speicher, es erfolgt keine Ausgabe.

`draw` Das Objekt wird in Form eines Drahtmodells ausgegeben, wobei die theoretisch nichtsichtbaren Linien gestrichelt gezeichnet werden.

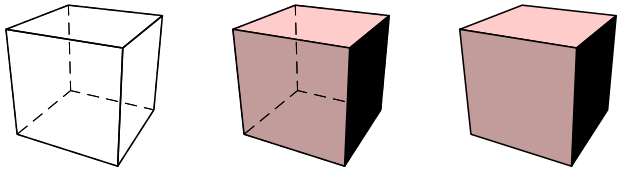
`draw*` Entspricht faktisch der Option ohne Stern, nur dass die Flächen mit der angegeben Farbe gefüllt werden.

`draw**` Verdeckte Flächen werden jetzt berücksichtigt, ebenso wie die Daten einer vorhandenen Lichtquelle, sodass Schattenwirkungen erfolgen.

`writesolid` Erstellt die folgenden Dateien:
`<Name>sommets.dat`: Liste der Eckpunkte
`<Name>faces.dat`: Liste der Polygonflächen
`<Name>couleurs.dat`: Liste der Flächenfarben
`<Name>io.dat`: Grenzen der Flächen

`writeobj` Externe Daten im `.obj`-Dateiformat schreiben[1]

`writeoff` Externe Daten im `.off`-Dateiformat schreiben[2]



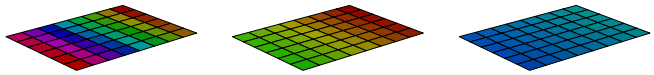
Gitter

In der Regel wird man die Zahl der zu bestimmenden Polygonflächen durch die Option `ngrid` vorgeben wollen. Für die Objekte `cube`, `tore`, `sphere`, `cylindre`, `cylindrecreux`, `cone`, `tronccone`, `troncconecreux`, `conecreux`, `calotesphere` und `calotespherecreuse` kann man durch Vorgabe eines Modus diese Eingabe vereinfachen. Zulässig sind nur Werte von 0. . . 4, wobei die Vorgabe gleich 2 ist.

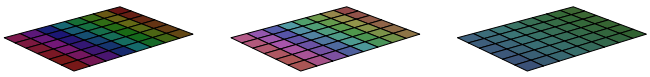
Mit größer werdender Zahl wird das Gitternetz feiner.

Farben

Die Farbverläufe der einzelnen Polygonzüge kann durch die Optionen `hue`, `inhue` und `inouthue` gesteuert werden. Als Parameter werden 2, 4 oder 7 Werte erwartet, wobei im einfachsten Fall zwei Angaben zum H-Wert des HSB-Farbmodells gemacht werden: `hue`=h_0 h_1, mit $0 \leq h_0 < h_1 \leq 1$.



Bei vier Angaben `hue`=$\mathrm{h_0}$ $\mathrm{h_1}$ s b werden die ersten beiden wie oben beschrieben angewendet, während s den festen Wert für die Sättigung und b den für die Schärfe kennzeichnet.

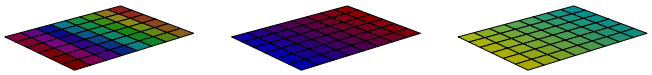


Bei der Angabe von sechs Parametern ist zusätzlich das Farbmodell (hsb) anzugeben, da die Werte jeweils Minimum und Maximum für HSB oder RGB darstellen können, wobei RGB Standard ist: `hue`=$\mathrm{h_0}$ $\mathrm{s_0}$

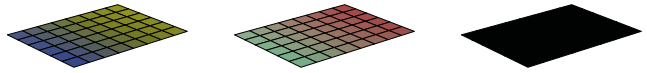
1 Siehe dazu auch `http://local.wasp.uwa.edu.au/~pbourke/dataformats/obj/`.

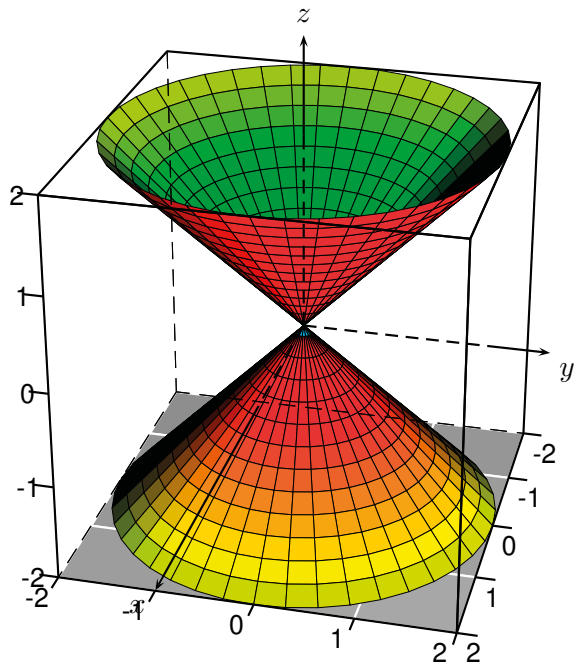
2 Siehe dazu auch `http://local.wasp.uwa.edu.au/~pbourke/dataformats/oogl/#OFF`.

b_0 h_1 s_1 b_1 (hsb) oder alternativ `hue=`r_0 g_0 b_0 r_1 g_1 b_1. Bei acht Parametern wird dagegen automatisch CMYK angenommen: `hue=`c_0 m_0 y_0 k_0 c_1 m_1 y_1 k_1.



Die letzte mögliche Variante ist die explizite Vorgabe zweier Farbnamen, die in runde Klammern zu setzen sind und selbstverständlich vorher durch das System oder den Benutzer definiert sein müssen: `hue=(Farbe1) (Farbe2)`. Es können sowohl die Standardfarben als auch sämtliche durch die PSTricks-Paketoption `dvipsnames` oder `svgnames` angeforderten Farbnamen benutzt werden. Zusätzlich können den internen Farboptionen `color1`, `color2`, `color3` und `color4` eigene Farben über Zahlenwerte zugewiesen werden. Dabei ist eine `xcolor`-Notation zulässig. Mit der Option `deactivatecolor` kann das Füllen der Polygonzüge unterbunden werden.

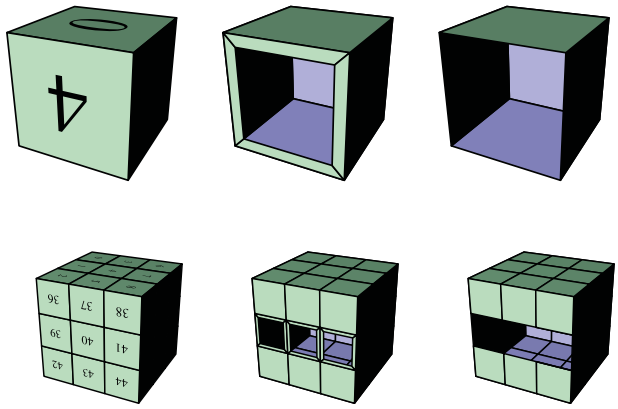





```
\begin{pspicture}(-4,-4.2)(4,4)
\psset[pst-solides3d]{viewpoint=20 5 10,Decran=30,
  lightsrc=20 10 5}
\psSolid[object=grille,base=-2 2 -2 2,linecolor=
  white](0,0,-2)
\defFunction{cone}(u,v){u v Cos mul}{u v Sin mul}{
  u}
\psSolid[object=surfaceparametree,base=-2 2 0 2 pi
   mul, inhue=0.8 0.2,hue=0.8 0.2,
  function=cone,linewidth=0.01,ngrid=25 40]
\gridIIID[Zmin=-2,Zmax=2](-2,2)(-2,2)[3,0.5,1]
\end{pspicture}
```

Löschen von Teilflächen

Mit der Option `affinage` kann bei Objekten mit mehr als einer Fläche ein Blick in das Innere erreicht werden, wobei `affinagecoeff` den relativen Grad der Öffnung angibt. Notwendig ist allerdings, dass das Innere durch `incolor` mit einer anderen Farbe versehen wird.

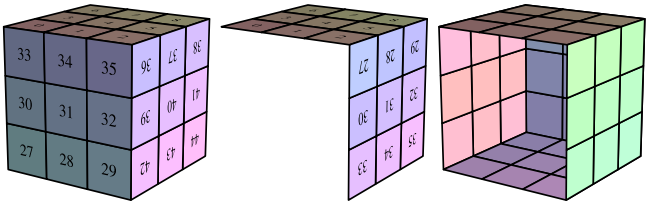


Einzelne Teilflächen können ebenfalls über die Option `rm` ausgeblendet werden, wobei dann in der Regel eine weiße Fläche die Folge ist, da eventuell dahinterliegende Flächen wegen der formalen „Unsichtbarkeit“ nicht gezeichnet wurden. Im Gegensatz zu `affinage` wird immer die komplette Teilfläche entfernt. Zu beachten ist allerdings, dass sich die Zählung der Teiloberflächen nur auf die sichtbaren Flächen bezieht und sich diese daher ändern kann, wenn bestimmte, sonst sichtbare Teilflächen, ausgeblendet werden. Mit der Option `hollow` kann bei Objekten mit mehr

3D-Abbildungen im Textmodus

als einer Fläche ein Blick in das Innere erreicht werden, indem der Option `action` nicht standardmäßig `draw**` zugeordnet wird, sondern `draw*` (siehe auch Tabelle 4). Notwendig ist allerdings, dass das Innere durch `incolor` oder `inouthue` mit einer anderen Farbe versehen wird.

Die Nummerierung der Teilflächen sollte dann abgeschaltet werden, da mit `hollow` auch die sonst nichtsichtbaren Flächen nummeriert werden, was sehr unübersichtlich werden kann.

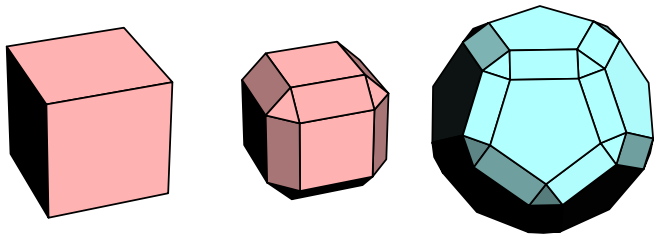


Die Option `rm` erwartet eine Liste von Flächennummern, die bei längeren Sequenzen auch durch einen entsprechenden PostScript-code ersetzt werden kann. So hätte man obige Liste auch durch `rm=27 1 35 {} for` oder durch `rm=9 {27} repeat` ersetzen können. Die Reihenfolge der Elemente spielt keine Rolle, da sie intern absteigend sortiert werden, bevor die entsprechenden Teilflächen gelöscht werden. Somit sind die folgende Angaben äquivalent:

```
rm=27 28 29 30 31 32 33 34 35
rm=35 -1 27 {} for
rm=35 34 33 32 31 30 29 28 27
rm=27 1 35 {} for
rm=27 27 27 27 27 27 27 27 27
rm= 9 {27} repeat
```

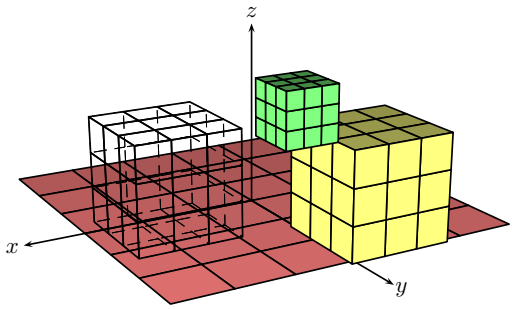
Kanten abflachen

Alle Kanten eines Körpers können über die Option `chanfrein` symmetrisch zum Körperzentrum „abgeschnitten“ werden, sodass aus einer Kante zwei folgen, und damit eine zusätzliche Fläche entsteht. Der abgeschnittene Anteil wird relativ durch `chanfreincoeff` festgelegt ($0 \ldots 1$).



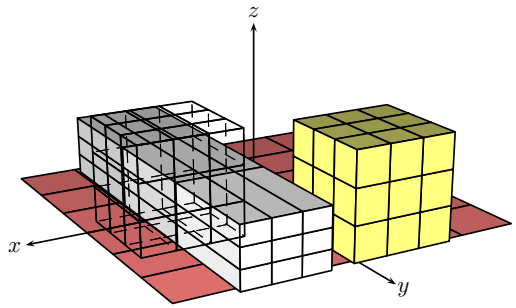
Transformationen

Im folgenden Beispiel ist der „gelbe“ (rechte) Würfel das Original mit dem Mittelpunkt $P(-2,0,0)$. Der kleine Würfel enstand aus dem großen durch Verschiebung des Mittelpunkts nach $P(-1,0,3)$ und skalare Multiplikation der Vektoren der Eckpunkte mit dem Faktor 0.5. Das Drahtmodell des Würfels hat seinen ursprünglichen Mittelpunkt in $P(-2,0,0)$ und wurde danach um den Vektor $V(3,-2,0)$ verschoben. Bei der Definition der Transformation ist davon auszugehen, dass auf dem Stack der aktuelleWert des dreidimensionalen Punktes liegt, also x, $y$und z. Folgerichtig müssen *nach* den PostScript-Operationen ebenfalls wieder drei Werte in derselben Reihenfolge auf dem Stack liegen.

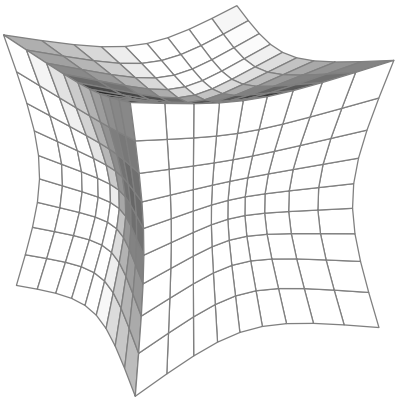



```
\psset{viewpoint=20 60 20 rtp2xyz,lightsrc=10 15
  7,Decran=20}
\begin{pspicture}(-5,-2.5)(6,3.5)
\psSolid[object=grille,base=-3 3 -3 3,fillcolor=
  red!50]
\axesIIID(4,4,3)
\psSolid[object=cube,fillcolor=yellow!50,a=2,ngrid
  =3](-2,0,0)
\psSolid[object=cube,fillcolor=green!50,a=2,
  transform={ 0.5 mulv3d },ngrid=3](-1,0,3)
\psSolid[object=cube,transform={ 3 -2 0 addv3d },
  action=draw,a=2,ngrid=3](-2,0,0)
\end{pspicture}
```

Für die Vektoroperationen stehen sämtliche PostScript- Funktionen zur Verfügung, wobei hier nur einige der internen Funtktionen genannt werden sollen: `addv3d`, `mulv3d`, `scale0point3d` und `rotate0point3d`. Weitere sind in der Dokumentation zur Prologdatei `solides3d.pro` erklärt.

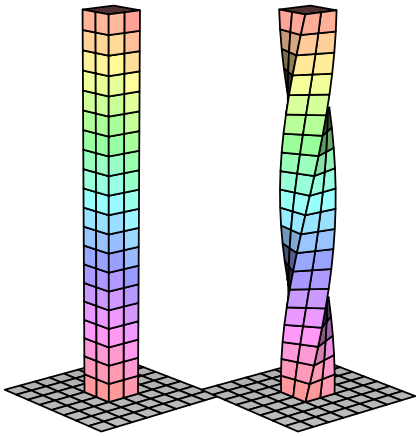


Mithilfe des Makros `\pstVerb` kann auch eine eigene PostScript-Funktion definiert werden, auf die dann bei der Transformation bezug genommen werden kann. Auf diese Weise sind dann Verformungen, beispielsweise eine Torsion, eines Körpers zu erreichen.



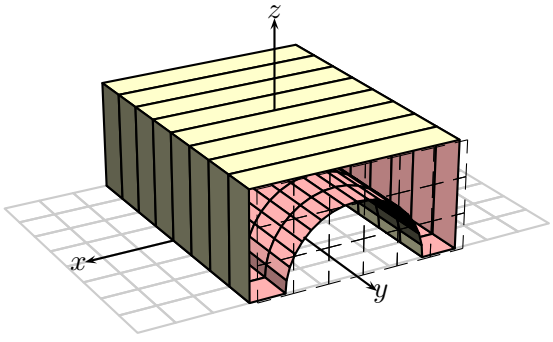
```
\psset{viewpoint=20 60 20 rtp2xyz,
  lightsrc=10 15 7,Decran=20}
\pstVerb{
  /verform {
    4 dict begin   % lokal halten
    /M defpoint3d  % als Vektor speichern
    /a .5 def /b 1 a 3 sqrt mul sub def
    /k M norme3d a mul b add def
    M k mulv3d
  end
  } def }
\begin{pspicture}(-3,-4)(3,3)
\psset{linewidth=.02,linecolor=gray}
\psSolid[object=cube,a=3,ngrid=9,
    transform=verform]%
\end{pspicture}
```

Im folgenden Beispiel hat man zuerst ein Prisma der Höhe 10 und der Seitenlänge 1. Die Seiten sind in Gittern zu je 20×2 eingeteilt. Die PostScript-Funktion `torsion` speichert zuerst den aktuellen Punkt in M, um danach die Koordinaten wieder aufzurufen und nur die z-Koordinate zu speichern. Dann wird der aktuelle Punkt wieder auf den Stack gelegt, danach kommen die Winkelwerte 0 0 und $z\cdot18$, die dann für die Rotation um die z-Achse benutzt werden. Da z maximal den Wert 10 annimmt, beträgt die Torsion des Prismas daher insgesamt 180 Grad.



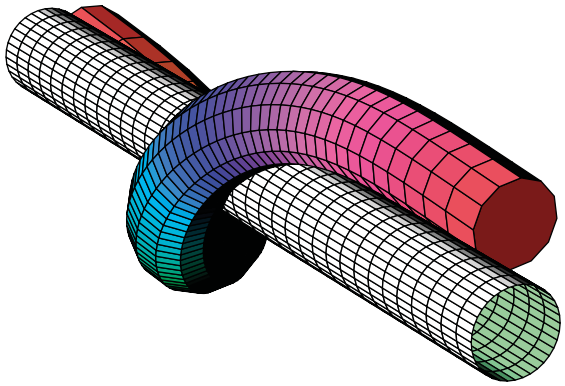
Definition neuer Objekte

Im folgenden Beispiel wird eine spezielle Grundfläche durch Kombination eines Linienzuges und einer Funktion erzeugt. Entscheiden ist hierbei, dass ein geschlossener Linienzug für die einhüllende Kurve der Grundfläche entsteht. Die Ansicht der Grundseite erreicht man durch einfache Drehung um die x- Achse mit `RotX`=90.



3D-Abbildungen im Textmodus

```
\psset{lightsrc=10 20 30,viewpoint=50 60 25 rtp2
  xyz,Decran=25}
\begin{pspicture}(-4,-2)(3,3)
\defFunction{G}(t){ t cos 2 mul }{ t sin 2 mul }{}
\psSolid[object=grille,base=-6 6 -4 4,action=draw,
  linecolor=black!20]
\psSolid[object=prisme,h=8,fillcolor=yellow!20,
  RotX=90,hollow,% zeige Innenteil
  ngrid=8,incolor=red!30,resolution=19,% fuer
    courbeR2+
  base=2 0 3 0 3 3 -3 3 -3 0 -2 0 % x y ...
       180 0 {G} CourbeR2+](0,4,0)
\axesIIID[showOrigin=false](3,1.5,3)(5.5,5,5.5)
\psSolid[object=grille,base=-3 3 0 3,RotX=90,
  action=draw](0,4,0)
\end{pspicture}
```

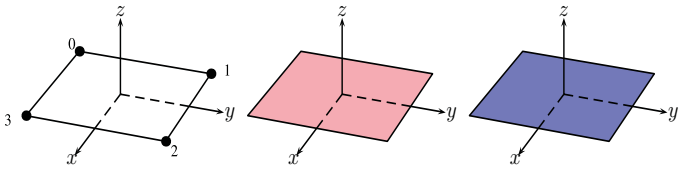


```
\begin{pspicture}(-7,-5)(7,4)
\psset{lightsrc=80 30 30,viewpoint=100 60 20 rtp2
  xyz,Decran=100}
\psset{solidmemory}
\defFunction[algebraic]{Func}(t){2+2*cos(t)}{2*tan
  (t/2)}{2*sin(t)}
\defFunction[algebraic]{Func'}(t){-2*sin(t)}{2*(1+
  tan(1/2*t)^2)}{2*cos(t)}
\psSolid[object=courbe,range=-2.8 2.6,ngrid=72 12,
  function=Func,name=H1,
  hue=0 1 0.7 1,action=none,r=1]
\psSolid[object=cylindrecreux,h=20,r=1,RotX=90,
  incolor=green!30,action=none,name=C1,
  ngrid=36 36](2,10,0)
\psSolid[object=fusion,base=H1 C1]
\composeSolid
\end{pspicture}
```

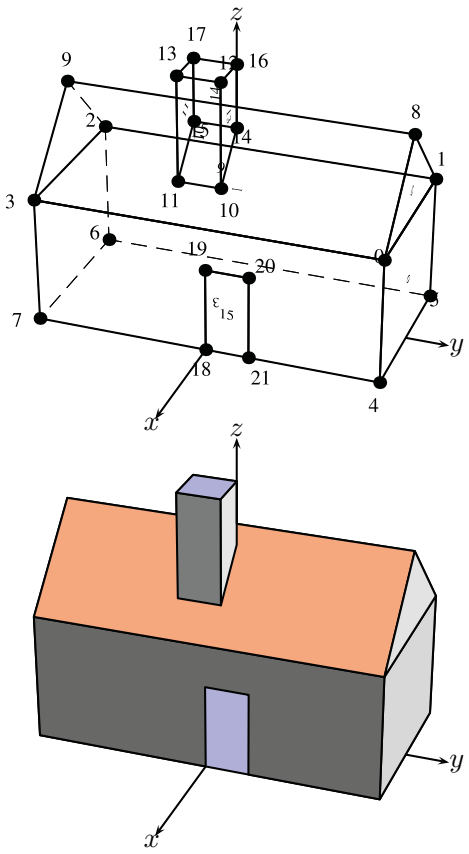
Neben den standardmäßigen Objekten, die weiter oben zusammengestellt sind, kann man sich neue Objekte erzeugen durch Festlegung der Eckpunkte und der daraus resultierenden Teilflächen oder alternativ bei Prismen durch Festlegung der Grundfläche. Der Option `object` wird dabei der Wert `new` zugewiesen. Die Eckpunkte sind über das optionale Argument `sommets` und die Teilflächen über `faces` festzulegen. Die Grundfläche der Prismen kann auch durch zweidimensionale Funktionen erfolgen, wozu der PostScript-Funktion `CourbeR2+` die mit `\defFunction` definierte Funktion zu übergeben ist.

Drahtmodelle werden über die Definition der Eckpunkte und Flächen vorgenommen. Hierbei muss PostScript mitgeteilt werden, wie die Fläche im Raum bezüglich vorne/hinten liegt. Dies ergibt sich aus dem Drehsinn beziehungsweise aus dem Richtungsvektor der Fläche. Mit der „rechten-Hand-Regel" lässt sich dies einfach festlegen; der Daumen stellt den Richtungsvektor der vorne/außen liegenden Fläche dar und die Finger geben dann die Drehrichtung vor.

Dies lässt sich leicht an einem Viereck zeigen.



Die linke Abbildung in dem obigen Beispiel zeigt die interne Nummerierung der Punkte. Die mittlere Abbildung definiert die Fläche durch `faces={[0 3 2 1]}`, sodass der Daumen der Rechten Hand dem Betrachter entgegenkommt. Die z-Achse ist somit gleich dem Richtungsvektor und die sichtbare Fläche entspricht „außen", sie nach Vorgabe mit `fillcolor=`*red!30* (30% Rot) gefüllt wird. Im rechten Beispiel wurde der Drehsinn des Rechtecks geändert; [0 1 2 3] ist jetzt entgegengesetzt orientiert, der Richtungsvektor ist daher entgegengesetzt zur z-Achse. Die nun sichtbare Fläche entspricht daher „innen" und wird mit `incolor=`*blue!20* (20% Blau) gefüllt, allerdings nur unter der Voraussetzung, dass innen liegende Flächen durch die Option `hollow` sichtbar gemacht werden.

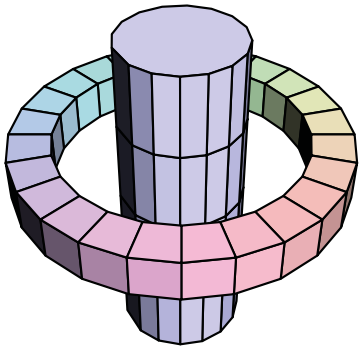



```
\begin{pspicture}(-3,-1.5)(3,4)
\psSolid[object=new,action=draw**,sommets=
 2 4 3  -2 4 3  -2 -4 3  2 -4 3  2 4 0  -2 4 0  -2
    -4 0  2 -4 0  0 4 5  0 -4 5
 1 0 3.5  1 -1 3.5  1 0 6  1 -1 6  0 0 4.5  0 -1
    4.5  0 0 6  0 -1 6
 2 0 0  2 0 2  2 1 2 2 1 0,
 faces={[0 4 5 1][0 1 2 3][7 6 5 4][0 3 7 18 19 20
    21 4][3 9 2][1 8 0][8 9 3 0][9 8 1 2]
  [6 7 3 2][2 1 5 6][10 12 13 11][11 15 17 13][12
    13 17 16][12 10 14 16]
  [14 16 17 15][18 21 20 19]},fillcolor=black!10,
    incolor=blue!30,
  fcol=24 (Goldenrod) 6 (Apricot),rm=15,hollow]
\axesIIID[showOrigin=false](2,4,5)(5,5,7)
\end{pspicture}
```

3. Komplexe Objekte

In allen bisherigen Beispielen wurde alle Objekte der Reihe nach gesetzt und verdeckten dadurch eventuell andere Objekte. Dabei spielte das Prinzip der verdeckten Linien beziehungsweise Flächen keine Rolle. `pstsolides3d` kann dies bei entsprechender Vorgabe jedoch berücksichtigen. Mit der Option `solidmemory` wird erreicht, dass alle Objekte gesammelt werden, bis sie am Ende mit dem Makro `\coposeSolid` ausgegeben werden. Dazu wird es im Allgemeinen notwendig sein, allen „gesammelten" Objekten `action=none` und mit `name=. . .` einen Objektnamen zuzuweisen.

Betrachten wir das folgende Beispiel, bei dem ein Zylinder in einem Ring steckt. Sowohl Ring als auch Zylinder verdecken gegenseitig einige Teilflächen. Um dies sichtbar zu machen,müssen erst alle Teilflächen *beider* Objekte gespeichert werden, bevor am Ende eine Kombination der beiden ausgegeben wird.



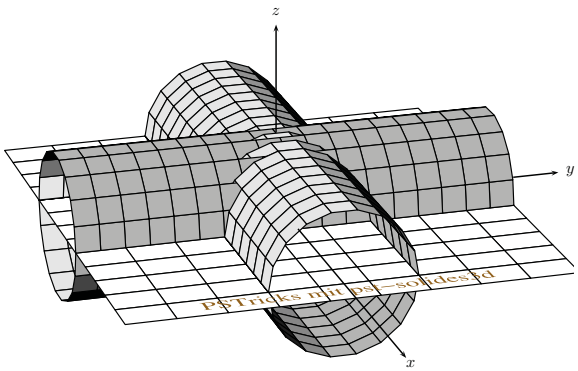
```
\begin{pspicture}[solidmemory,Decran=10](-4,-3)
  (4,2)
\psSolid[object=anneau,h=6,hue=0 1 0.2 0.9,
  R=4,r=3,h=1,action=none,name=Ring](0,0,-1)
\psSolid[object=cylindre,h=8,fillcolor=blue!20,
  r=1.5,ngrid=4 16,action=none,name=Zylinder
    ](0,0,-6)
\psSolid[object=fusion,action=draw**,
  base=Ring Zylinder]
\end{pspicture}% gibt Objekte aus
```

Hierbei spielt die Reihenfolge der Objekte keine Rolle, da sie ohnehin intern mit ihren Teilflächen derart sortiert werden, dass verdeckte Flächen beziehungsweise Linien nicht gezeichnet werden. Die Objekte werden in der üblichen Weise erstellt, jedoch mit der Option `action=none`, sodass keine Ausgabe erfolgt, der erstellte PostScript-Code jedoch weiterhin intern zur Verfügung steht. Dies kann über die willkürlich gewählten Namen *Ring* und *Zylinder* erfolgen. Das letzte erstellte Object `fusion` bekommt die Namen über die Option `base` zugewiesen, wodurch intern die beschriebene Sortierung aller Teilflächen mit anschließender

3D-Abbildungen im Textmodus

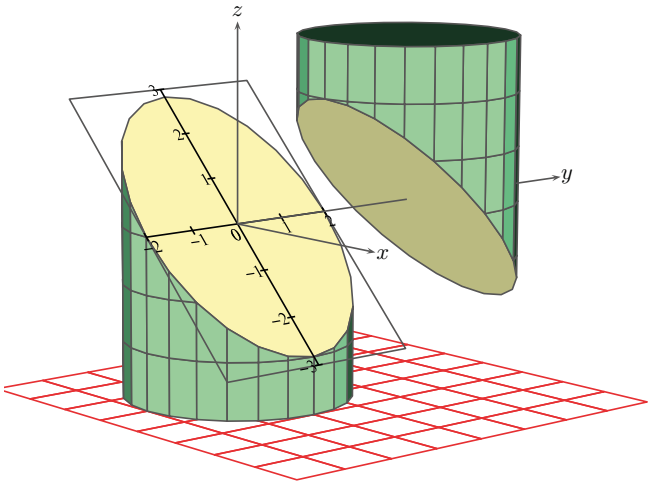
Ausgabe erfolgen kann, denn jetzt wird der Option `action` die Aktion `draw**` zugewiesen.

Die Ebene in dem Bild wurde als Trennfläche für die beiden Zylinder benutzt. Über die Option `plansepare` kann eine Trennebene angegeben werden, die dann den Gesamtkörper in zwei Teilkörper terlegt, wobei an einen gegebenen Namen 0 und 1 angehängt wird. Im folgenden Beispiel werden also *zylinder1* und *zylinder2* zerlegt in *zylinder10* und *zylinder11*, beziehungsweise *zylinder20* und *zylinder21*. Auf diese kann dann in der üblichen Weise zurückgegriffen werden.




```
\begin{pspicture}[solidmemory](-8,-5)(7,5)
\psset{lightsrc=50 -50 50,viewpoint=100 -30 40,
  Decran=100}
\psSolid[object=plantype,definition=equation,args
  ={[0 0 1 0] 90},
   base=-6 6 -7 6,ngrid=9 13,name=monplan]
\defFunction[algebraic]{Cyl1}(t){2*cos(t)}{-6}{2*
  sin(t)}
\defFunction[algebraic]{Cyl2}(t){-6}{2*cos(t)}{2*
  sin(t)}
\psSolid[object=cylindre,range=0 6.28,h=12,
  function=Cyl1,axe=0 1 0,
   incolor=black!30,fillcolor=black!10,ngrid=18
     18,
   plansepare=monplan,name=zylinder1,action=none]
\psSolid[object=cylindre,range=6.28 0,h=12,
  function=Cyl2,axe=1 0 0,
   incolor=black!10,fillcolor=black!30,
   plansepare=monplan,ngrid=18 18,name=zylinder2,
     action=none]
\psSolid[object=fusion,base=zylinder11 zylinder21]
  % untere Haelften
\psSolid[object=plan,args=monplan]          % Ebene
\psSolid[object=fusion,base=zylinder10 zylinder20]
  % obere Haelften
\psProjection[object=texte,plan=monplan,linecolor
  ={[cmyk]{0.4,0.6,1,0.2}},
   text=PSTricks mit pst-solides3d,fontsize=20,pos
     =cc](0,-6.5)
\axesIIID[linewidth=1pt,labelsep=10pt,showOrigin=
  false](2,6.7,2)(11,8,5)
\end{pspicture}
```

Im folgenden Beispiel wird der senkrechte Zylinder vom Radius 2 LE und der Höhe 6 LE (Angabe jeweils in Längeneinheiten) durch die Ebene $\frac{\sqrt{2}}{2}\vec{x} + \frac{\sqrt{2}}{2}\vec{z} = 0$ (`plansepare={[0.707 0 0.707 0]}`[3] in zwei Hälften *Zylinder0* und *Zylinder1* geteilt, wobei der Name *Zylinder* willkürlich gewählt wurde.




```
\begin{pspicture}[solidmemory](-4,-5)(7,4)
\psset{viewpoint=50 -40 10 rtp2xyz,Decran=50,
  linecolor=darkgray,lightsrc=viewpoint}
\psSolid[object=grille,action=draw,base=-3 5 -3 5,
  linecolor=red](0,0,-3)
\psSolid[object=cylindre,r=2,h=6,ngrid=6 24,
  plansepare={[0.707 0 0.707 0]},
   name=Zylinder,action=none](0,0,-3)
\psSolid[object=load,load=Zylinder1,fillcolor={[
  rgb]{0.7 1 0.7}},
   fcol=0 (1 1 0.7 setrgbcolor)]
\psSolid[object=load,load=Zylinder0,RotZ=90,
  fillcolor={[rgb]{0.7 1 0.7}},
   fcol=0 (1 1 0.7 setrgbcolor)](0,4,0)
\psSolid[object=plan,action=draw,definition=
  equation,
   args={[0.707 0 0.707 0] 90},base=-2 2 -3 3,
     planmarks]
\psSolid[object=line,args=0 0 0 0 4 0]% erste
  Haelfte y-Achse
\axesIIID[showOrigin=false](0,6.8,0)(3.5,8,3.5)
\end{pspicture}
```

Die y-Achse liegt aufgrund der Definition in der Ebene und der Winkel zwischen der Ebene und den anderen beiden Achsen beträgt jeweils 45 Grad. Der Gesamtzylin-

3 Die Angabe muss zwingend in geschweifte Klammern gesetzt werden, um die umgebenden Klammern nicht zu beeinflussen.

der liegt mit seinem Mittelpunkt im Koordinatenursprung und wird wird durch die Angabe `action=none` nicht gezeichnet. Die beiden Hälften werden durch `object=load` und der entsprechenden Zuordnung mit `load=...` geladen. Die obere Hälfte, intern als *Zylinder0* bezeichnet wird zusätzlich um 90Grad um die eigene z-Achse gedreht und in y-Richtung um 4 LE versetzt angeordnet. Die untere Hälfte, intern als *Zylinder1* bezeichnet, verbleibt an der ursprünglichen Stelle.

Literatur

[1] *Hendrik Rama;* APL2, GUIGL und OpenGL – alles in 3D und Farbe, APL-Journal 2/2007, S. 3– 23.

[2] *Herbert Voß;* LyX – OpenSource Document Processor, APL-Journal 1/2001, S. 29–43.

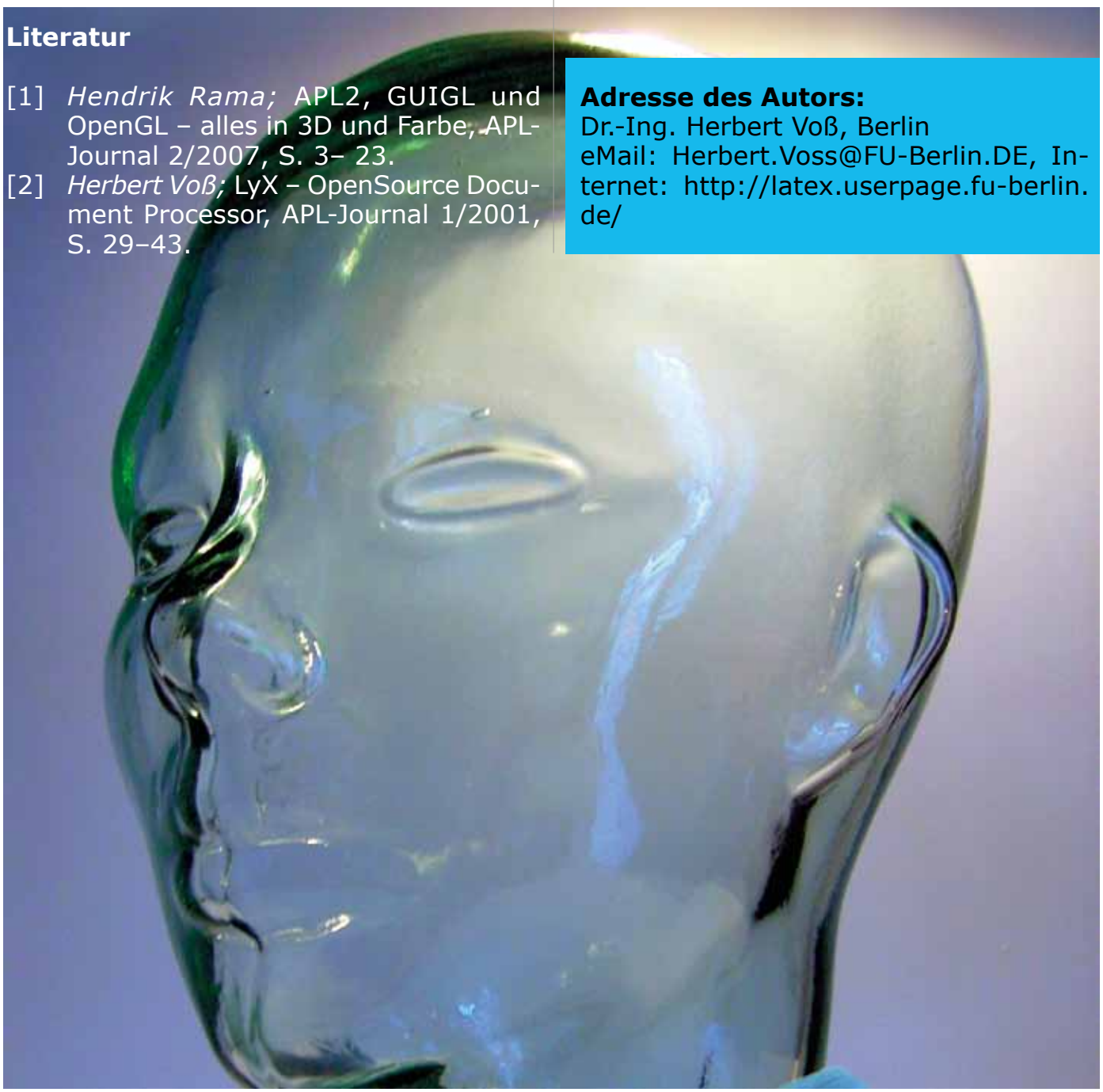
[3] *Herbert Voß;* Die mathematischen Funktionen von PostScript, APL-Journal 2/2001, S. 13–15.

[4] *Herbert Voß;* LaTeX – ein flotter Oldie, APLJournal 2/2004, S. 12–16.

[5] *Herbert Voß;* PSTricks – Grafik für TeX und LaTeX, 5.Aufl., Heidelberg/Berlin 2008.

[6] *Frank Mittelbach et al.;* Der LaTeX-Begleiter, 2. Aufl., München 2005.

Adresse des Autors:
Dr.-Ing. Herbert Voß, Berlin
eMail: Herbert.Voss@FU-Berlin.DE, Internet: http://latex.userpage.fu-berlin.de/



Hans-Georg Leis

APL-Programm zur Lösung von SUDOKU-Rätseln

Das Programm, das hier vorgestellt wird, wurde mittels APL-Befehlen generiert und läuft unter APL2 version 2.0 service level 14 von IBM.

1. SUDOKU Anwendungsbeispiel

1.1 Allgemeiner Programm-Aufruf

```
KW SUDPRN TMATR
⍝ KW = 0/1 ↔ ohne / mit Ausgabe von
       Zwischenergebnissen
⍝ TMATR = Name der 9x9 SUDOKU Eingabe-Matrix
```

1.2 SUDOKU-Rätsel aus http://www.dailysudoku.com vom 11.5.2009 „very hard"

➤ Dateneingabe

```
MD090511←9 9⍴'o'
```

Mit Hilfe des Editors ergibt sich daraus die Vorgabematrix:

```
o2o7ooooo6
1o7oo8oo9
oo5ooo413
6ooooooo7
394ooo8oo
7oo4oo6o1
4oooo7o8o
oo1o5oo2o
```

o steht dabei für die ursprünglichen Leerstellen.

➤ Ergebnisse

Der Programmaufruf

```
0 SUDPRN 'MD090511'
```

ergibt als Startbedingung:

```
MD090511/START/ IC=0   LC=0
```

	4			3		2		
	2		7					6
1		7			8			9
		5				4	1	3
6								7
3	9	4				8		
7			4			6		1
4					7		8	
		1		5			2	

die Lösung:

```
MD090511/ENDE/ 484 msec / IC=8   LC=1
```

5	4	6	1	3	9	2	7	8
8	2	9	7	4	5	1	3	6
1	3	7	2	6	8	5	4	9
2	7	5	8	9	6	4	1	3
6	1	8	3	2	4	9	5	7
3	9	4	5	7	1	8	6	2
7	5	3	4	8	2	6	9	1
4	6	2	9	1	7	3	8	5
9	8	1	6	5	3	7	2	4

Die Lösung ist dadurch gekennzeichnet, dass jede Zeile und jede Spalte, sowie jede 3x3 Teilmatrix der Ergebnismatrix genau die Ziffern von 1 bis 9 enthält.
Ein typisches Iterations-Zwischenergebnis ergibt sich als Teil der Ausgabe des Programmaufrufs.

```
1 SUDPRN 'MD090511'

MD090511/ LAUF: ZWISCHENERGEBNIS:/ IC=6   LC=1
```

5	4	6	1 9	3	1 9	2	7	8
8	2	9	7	4	5	1	3 5	6
1	3	7	2 6	2 6	8	3 5	4	9
2	7	5	8	9	6	4	1	3
6	1	8	3	2	4	5 9	5 9	7
3	9	4	5	7	1	8	6	2
7	5	2 3	4	8	2 9	6	3 9	1
4	6	2 3	2 9	1	7	3 9	8	5
9	8	1	3 6	5	3 6	7	2	4

2. Roter Faden des Programmablaufs

Auf die Beschreibung sämtlicher verwendeter Unterprogramme wurde verzichtet. Für weitergehende Informationen und Diskussionen steht der Autor jederzeit gerne zur Verfügung.

2.1. Programmstruktur – generiert mit Hilfe von FNSTREE.ATF:

Die Programme, die im Folgenden einzeln beschrieben werden, sind durch die Farbe Blau hervorgehoben.

```
SUDPRN
 |  START   (ROW:6)
 |  |  MATFORM   (ROW:2)
 |  |  |  SUDTABN   (ROW:4)
 |  |  |  |  TABSTRUKT   (ROW:11 15)
 |  |  |  IMATR   (ROW:9)
 |  |  |  |  KOMB   (ROW:3)
 |  |  DISPLAY   (ROW:3 4 8)
 |  |  |  DISPLAY   (ROW:28 32)   * SEE ROW: 37
 |  LAUF   (ROW:17)
 |  |  EEEN   (ROW:3)
 |  |  |  EVRED1A   (ROW:11)
 |  |  |  |  REDKRIT   (ROW:1)
 |  |  |  |  |  REDKRIT   (ROW:2)   * SEE ROW: 42
 |  |  |  |  REDUZ   (ROW:9)
 |  |  |  SUDSUM   (ROW:12 18 25 31 37)
 |  |  |  EVRED1E   (ROW:17)
 |  |  |  |  REDKRIT   (ROW:1)   * SEE ROW: 42
 |  |  |  |  SETZE   (ROW:5)
 |  |  |  EVRED12   (ROW:24)
 |  |  |  |  REDKRIT   (ROW:1)   * SEE ROW: 42
 |  |  |  |  EVRED12K   (ROW:7)
 |  |  |  |  |  REDUZ   (ROW:15)
 |  |  |  EVRED2A   (ROW:30)
 |  |  |  |  REDKRIT   (ROW:1)   * SEE ROW: 42
 |  |  |  |  REDUZ   (ROW:20)
 |  |  |  EVRED2E   (ROW:36)
 |  |  |  |  REDKRIT   (ROW:1)   * SEE ROW: 42
 |  |  |  |  EVRED2EK   (ROW:9)
 |  |  SUDSUM   (ROW:4)
 |  |  SUDTABN   (ROW:11 17 23 27)   * SEE ROW: 33
 |  |  LAUF   (ROW:17 23)   * SEE ROW: 39
 |  DISPLAY   (ROW:28 31)   * SEE ROW: 37
 |  PROBE   (ROW:41)
 |  |  PROBE   (ROW:12)   * SEE ROW: 63
 |  |  SUDTABN   (ROW:12)   * SEE ROW: 33
 |  SUDPRN   (ROW:45)   * SEE ROW: 30
 |  SUDTABN   (ROW:45)   * SEE ROW: 33
```

2.2. Hauptprogramm SUDPRN mit Erklärungen

Im Folgenden sind (Unter-)Programm-Namen blau gedruckt.

```
ERG←KW SUDPRN TMATR;DSUG;EE;EE0;IC;ICM;ICTS;
INDM;INDZS;IZTS;LC;LCM;RC;SUG;SUG0;TS0;U
⍝
⍝  KW = 0/1 ↔ ohne / mit Ausgabe von
      Zwischenergebnissen
⍝ TMATR = Name der 9x9 SUDOKU Eingabe-Matrix
⍝
 TS0←+/(⌽×\⌽60 60 1000 1)×3↑⎕TS ⍝ Start-Tageszeit in msec
 (EE INDM INDZS SUG)←KW START TMATR
```

➤ Ergebnisse von START:

EE ist ein Vektor der Länge 81, der den einzelnen Elementen der SUDOKU-Matrix zugeordnet ist. Das einzelne EE-Element enthält dabei als geschachtelte Größe den Vektor der möglichen Ziffern dieses Matrix-Elements. Dementsprechend wird den einzelnen leeren Elementen der Vorgabematrix beim Programmstart jeweils der Wert ⊂⍳9 (d.h. 1 2 3 … 9) zugeordnet. Die Matrix INDM enthält die Indizes von EE für die Zuordnung der einzelnen SUDOKU-Matrix-Elemente.

INDM

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Lösung von SUDOKU Rätseln

Die Matrix INDZS enthält zeilenweise nach dem Zeilenindex jeweils die Indizes der relevanten Bereiche der SUDOKU-Matrix:

- Die ersten 9 Zeilen enthalten die EE-Indizes der einzelnen Zeilen,
- die anschließenden 9 Zeilen die Indizes der einzelnen Spalten,
- die letzten 9 Zeilen die Indizes der einzelnen 3x3 Teilmatrizen.

INDZS

1	1	2	3	4	5	6	7	8	9
2	10	11	12	13	14	15	16	17	18
3	19	20	21	22	23	24	25	26	27
4	28	29	30	31	32	33	34	35	36
5	37	38	39	40	41	42	43	44	45
6	46	47	48	49	50	51	52	53	54
7	55	56	57	58	59	60	61	62	63
8	64	65	66	67	68	69	70	71	72
9	73	74	75	76	77	78	79	80	81
10	1	10	19	28	37	46	55	64	73
11	2	11	20	29	38	47	56	65	74
12	3	12	21	30	39	48	57	66	75
13	4	13	22	31	40	49	58	67	76
14	5	14	23	32	41	50	59	68	77
15	6	15	24	33	42	51	60	69	78
16	7	16	25	34	43	52	61	70	79
17	8	17	26	35	44	53	62	71	80
18	9	18	27	36	45	54	63	72	81
19	1	2	3	10	11	12	19	20	21
20	4	5	6	13	14	15	22	23	24
21	7	8	9	16	17	18	25	26	27
22	28	29	30	37	38	39	46	47	48
23	31	32	33	40	41	42	49	50	51
24	34	35	36	43	44	45	52	53	54
25	55	56	57	64	65	66	73	74	75
26	58	59	60	67	68	69	76	77	78
27	61	62	63	70	71	72	79	80	81

Die Variable SUG enthält die Anzahl der einzelnen Ziffern aller Elemente. Das Kriterium für ein erfolgreiches Ende der Iteration ist der Zielwert SUG=81.

```
⍝
 LC←0   ⍝ s.u. RC=2 ↔ Zählparameter der Versuchsschleifen
 LCM←0  ⍝ → Anzahl der Versuchsschleifen
 IC←0   ⍝ Zählparameter der Programm-Schleifen
 ICM←99 ⍝ Maximalanzahl der Programm-Schleifen
ΔA:IC←IC+1
 (RC SUG DSUG EE)←LAUF(IC LC LCM KW INDZS ICM SUG TS0 EE)
```

Das Unterprogramm Lauf gibt das Ergebnis einer Iterationsschleife wieder:

Die Variable DSUG gibt die Abnahme der gesamten Ziffernanzahl SUG wieder.
Der resultierende Return-Code RC ergibt folgende Möglichkeiten:

RC = 1 ⇔ Iteration erfolgreich beendet ⇔ EE enthält nur noch 1-ziffrige Elemente

RC = 0 ⇔ Iteration ergab mindestens für ein EE-Element eine Reduzierung der Ziffern; weiterer Iterationsschritt schließt sich an

RC = 3 ⇔ Iteration abgebrochen nach 99 Iterationsschritten; bisher nicht aufgetretener Fall

RC = 2 ⇔ Iterationsschritt lässt alle EE-Elemente unverändert; trial-and-error method: sequentieller Test der einzelnen 2-ziffrigen EE-Elemente durch Ausprobieren deren einzelner Ziffern.

```
U←'SUDPRN: IC=',(⍕IC),' RC=',(⍕RC),' LC=',(⍕LC),' SUG=',(⍕SUG),
'DSUG=',(⍕DSUG)
 ⍞((KW>0)∨(RC=3))/' DISPLAY U'  ⍝ Ausgabe Zwischenergebnis
 →((RC=1)∨(RC=3))/ΔE
 ⍞(RC=0)/'LCM←LCM+LC ◊ LC←0 ◊ →ΔA'
⍝ RC=2: trial-and-error method
 ⍞(LC=0)/'EE0←EE ◊ SUG0←SUG' ⍝ Referenz-Vergleichswerte
 LC←LC+1              ⍝ trial-and-error Nr. LC
 EE←EE0◊SUG←SUG0 ⍝ Aktualisierung der Referenz-Vergleichswerte
 EE←PROBE(LC EE) ⍝ → EE Test-Variante Nr. LC
⍝ Ausgabe des Zwischenergebnisses:
 U←' IC=',(⍕IC)' LC=',(⍕LC),' SUG=',(⍕SUG)
 ⍞(KW>0)/'(''SUDPRN: PROBE '',U)SUDTABN IC LC'
→ΔA
ΔE:
```

2.3. Das Unterprogramm Lauf beinhaltet die einzelne Programmschleife zur iterativen Verbesserung des Ergebnis-Vektors EE

```
ERG←LAUF PAR;DSUG;EE;IC;ICM;INDZS;KW;LC;LCM;RC;SUG;SUGV;TS0;U
(IC LC LCM KW INDZS ICM SUGV TS0 EE)←PAR
⍝ SUGV ↔ bisherige Gesamtanzahl der Ziffern in EE
 EEEN←⊂[2]INDZS  ⍝ Einzelzeilen aus INDZS
```

Durch EEEN werden die einzelnen Zeilen aus INDZS, d.h. alle Zeilen, Spalten und 3x3 Teilmatrizen der SUDOKU-Matrix analysiert und gegebenenfalls reduziert.

```
 (RC SUG)←2 SUDSUM EE
⍝ SUG ↔ aktualisierte Gesamtanzahl der Ziffern
in EE
 DSUG←SUGV-SUG        ⍝ Änderung der Ziffernanzahl
→(RC≠1)/Δ2
 U←+/(⌽×\⌽60 60 1000 1)×3↓ TS  ⍝ Ende-Tageszeit in msec
 U←U-TS0                       ⍝ Rechenzeit
in msec
 ('ENDE/',(⍕U)'msec')SUDTABN IC LCM
→ΔE
Δ2:→(IC≠ICM)/Δ4
 RC←3
 U←' RC=',(⍕RC),' IC=',(⍕IC),' ICM=',(⍕ICM)
 ⍞(KW>0)/'(" LAUF: ABBRUCH/",U)SUDTABN IC LC '
→ΔE
```

```
Δ4:→(DSUG≠0)/Δ6
 RC←2
 U←' RC=',(⍕RC),' LC=',(⍕LC),' SUG=',(⍕SUG),' DSUG=',(⍕DSUG)
 ⍞(KW>0)/'(é' LAUF: ABBRUCH/'',U)SUDTABN IC LC'
→ΔE
Δ6:→(KW=0)/ΔE
 ' LAUF: ZWISCHENERGEBNIS:' SUDTABN IC LC
ΔE:ERG←(RC SUG DSUG EE)
```

2.4. Unterprogramm EEEN / fünf Datenreduzierungskriterien

```
EEEN II;BL;EV;IV;IZL;KK;LL;MM;W;RC;U
⍝ global aus LAUF: IC LC KW SUK0
 IZL←II[1]      ⍝ NR. DER BETRACHTETEN ZEILE IN INDZS-MATRIX
 II←1↓II        ⍝ ENTSPR. EE-INDIZES DER EINZELZEILE VON INDZS
 EV←,¨EE[II]    ⍝ VEKTOR, ρEV=9
```

EV-Element = EE-Elemente einer Zeile, Spalte bzw. 3x3 Teilmatrix der SUDOKU-Matrix

```
⍝
 EV←EVRED1A EV
```

Besteht ein EV-Element nur aus einer 1-ziffrigen Zahl, so wird diese Ziffer in den übrigen 8 EV-Elementen gelöscht.

```
(RC U)←1 SUDSUM EV ⍝ U ↔ DUMMY
→RC/ΔE
```

RC=1: EV besteht nur aus den 1-ziffrigen Zahlen von 1 bis 9, d.h. EV ist optimiert.

```
⍝
⍝ EV←EVRED1E EV
```

Tritt eine Ziffer nur in einem EV-Element auf, so wird das betreffende EV-Element auf diese Ziffer reduziert.

```
 (RC U)←1 SUDSUM EV
→RC/ΔE
⍝
 II EVRED12 EV
```

Kommt *eine* Ziffer nur in zwei 2-ziffrigen EV-Elementen vor, so wird diese Ziffer in den übrigen EV-Elementen gelöscht.

```
 (RC U)←1 SUDSUM EV
→RC/ΔE
⍝
 EV←EVRED2A EV
```

Treten bei den 9 EV-Elementen zwei identische 2-ziffrige Zahlen auf, so werden diese beiden Ziffern in den übrigen EV-Elementen gelöscht.

```
 (RC U)←1 SUDSUM EV
→RC/ΔE
⍝
 EV←EVRED2E EV
```

Treten 2 bestimmte Ziffern nur in 2 EV-Elementen auf, so können die beiden betroffenen EV-Elemente auf diese beiden Ziffern reduziert werden.

```
 (RC U)←1 SUDSUM EV
→RC/ΔE
⍝
ΔE:EE[II]←EV
```

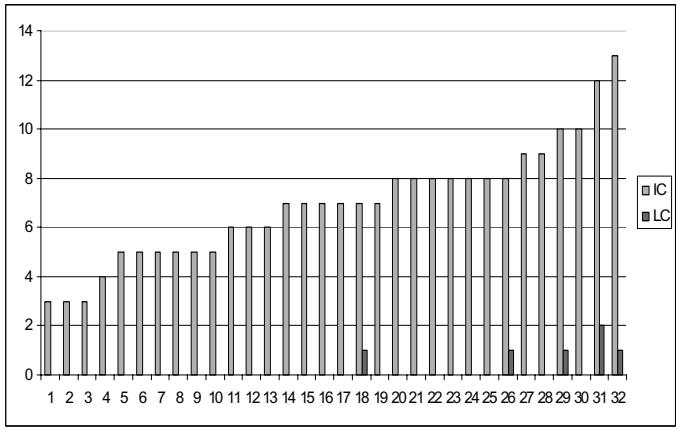
2.5. Unterprogramm SUDSUM ergibt:

- SUM ⇔ Anzahl der Einzelziffern in EV
- RC=1 ⇔ EV enthält nur 1-ziffrige Elemente

```
ERG←KV SUDSUM EV;RC;SUM
 SUM←ρ∈EV                ⍝ Anzahl der Ziffern in EV
⍝ KV=1 ↔ EV=Vektor der SUDOKU-Matrixelemente
⍝              einer Zeile, Spalte bzw. 3x3 Teilmatrix
 ⍞(KV=1)/'RC←SUM=9'  ⍝ RC=1 ↔ Teil-Iteration beendet
⍝ KV=2 ↔ EV=Vektor aller SUDOKU-Matrixelemente
  ⍞(KV=2)/'RC←SUM=81'  ⍝ RC=1 ↔ Gesamt-Iteration
       beendet
ERG←RC SUM
```

2.6. Auswertungsstatistik

Das Programm wurde bisher an 32 SUDOKU Rätseln erfolgreich angewendet.



Das Diagramm zeigt die Statistik von:

IC = Anzahl der notwendigen Iterationsschleifen
LC = Anzahl der zusätzlich notwendigen trial and error Schritte

Kontakt zum Autor:
Hans-Georg Leis,
eMail: hgleis@t-online.de

Neues in Workstation APL2 V2.0 Service Level 12

Seit der 1. Maiwoche 2008 ist der Service Level 12 von Workstation APL2 Version 2 für alle APL2-Anwender und für alle APL2-Plattformen verfügbar. Neben den üblichen Fixes zu bekannt gewordenen Problemen bietet dieser Service Level wieder einige besonders wichtige Erweiterungen, die von deutschen APL2-Anwendern schon lange gewünscht wurden. Hier die Übersicht (Stand: Mai 2008).

Für alle Plattformen:

◇ Neue externe Operatoren PEACHP und PEACHT:
Die neuen externen Operatoren `PEACHP` und `PEACHT` erlauben die automatische Verteilung von umfangreichen Rechenoperationen auf mehrere Prozessoren bzw. Threads auf der lokalen Maschine oder ggf. auf entfernten Maschinen. Dadurch kann die für die APL2-Anwendung verfügbare Prozessorleistung umso effektiver gesteigert werden, je rechenintensiver die Algorithmen bzw. je größer die zu verarbeitenden Datenmengen sind. Die „parallele Architektur" der APL2-Sprache ist hier von besonderem Nutzen, sodass die Leistung bei optimaler Programmstruktur und großer Datenmenge proportional der Anzahl der benutzten Prozessoren gesteigert werden kann.

◇ Aufruf von „DB2 Stored Procedures":
Über AP 127 und AP 227 ist jetzt der Aufruf von "DB2 Stored Procedures" (= vordefinierte, gelinkte, separat autorisierte und gespeicherte SQL-Stapelprozeduren) möglich einschließlich Parameter-Übergabe und Rücklieferung der Ergebnisse (analog dem Aufruf von Unterprogrammen in anderen Programmiersprachen). Der mitgelieferte Arbeitsbereich SQL und die zugehörige Dokumentation wurden ergänzt, um derartige Aufrufe zu ermöglichen. Diese erfolgen mit Hilfe der Standardfunktion `SQL` sowie der neu entwickelten Funktion `PROCEDURE`.

◇ Neue externe Funktion `DATATYPE`:
Die neue externe Funktion `DATATYPE` liefert APL2-interne Informationen über Datenobjekte im Arbeitsbereich. Sie ist kompatibel mit der gleichnamigen Funktion auf dem Großrechner.

◇ Bei den Assoziierten Prozessoren:

- Bei *Prozessor 10* wurde die Geschwindigkeit der mitgelieferten Dateiverarbeitungsfunktionen stark verbessert.
- *Prozessor 11* unterstützt jetzt durch „Null Character" getrennte Strings aus 2-Byte-Zeichen, wenn sie in den Datenmustern für die mitgelieferten Funktionen `ATR` und `RTA` (sowie auch für den Prozessor 15 und für Argumente bei Nicht-APL-Funktionen) spezifiziert werden.
- *Prozessor 12* unterstützt jetzt 5 neue Datenmuster, wenn eine Datei als einfacher Vektor angezeigt wird. Die Geschwindigkeit der Verarbeitung von durch Zeilenendezeichen begrenzten Dateien wurde erhöht.
- Der *neue Prozessor 15* ermöglicht den direkten Zugriff auf Speicherpositionen außerhalb des APL-Arbeitsbereiches unter Benutzung der internen APL-Variablennamen im Arbeitsbereich. Der

Speicherplatz wird durch P15 mittels eines vom Benutzer definierten Musters als APL2-Datenstruktur formatiert.

Für die Windows-Plattform:

◇ Sitzungsmanager und Objekteditor:
- In beiden zeigt der Cursor eine Sanduhr, solange APL-Funktionen laufen.
- In `APL Assist`, der in SL11 neu implementierten Anzeigeleiste für alle besonderen APL-Zeichen, können diese jetzt nach Belieben angeordnet werden.
- Einfache Tastenkombinationen ermöglichen das schnelle Öffnen der Liste der zuletzt bearbeiteten Objekte und des Objekt-Bearbeitungsdialogs.
- Im numerischen Tastenfeld, falls vorhanden, können jetzt die APL-Multiplikation und -Division mit den dortigen Tasten „*" und „/" verknüpft werden.
- Im Objekteditor können jetzt die Ausführungseigenschaften angezeigt und geändert werden.
- Die Tastenkombinationen „Strg + Pfeil rauf" und „Strg + Pfeil runter" bewegen jetzt im Objekteditor die Anzeigeraute (caret) von einem Label zum nächsten bzw. vorhergehenden.

◇ Der AP 145 wird erweitert:
- Microsoft ActiveX Controls können in AP 145-Dialogen verwendet werden.
- Neue Dialog-Stile `NoIcon` und `NoIgnore`
- Eingebaute `picture`-Unterstützung für Menü-Positionen.
- Die Funktionen `LOADAPI` und `CALLAPI` unterstützen die direkte Adressierung von Windows APIs

◇ Im Kapitel `COM` des **`User's Guide`** wurde ein neuer Abschnitt eingefügt, der demonstriert, wie man Visual Basic Macros zwischen APL2 und der Microsoft Office-Umgebung hin und her transferieren kann.

Weitere Informationen zu APL2 Version 2 Service Level 12 erhalten Sie im `APL2 User's Guide` und der Online-Hilfe nach der Installation des Service Levels oder auf der IBM-APL2-Website: http://www-306.ibm.com/software/awdtools/apl/support.html
Darüber hinaus Informationen zu Workstation APL2 Version 2 bei Dittrich & Partner Consulting GmbH, Kieler Str. 17, 42697 Solingen, Tel. 0212-26066-0, Fax: 0212-26066-66, e-Mail: info@dpc.de

Jürgen Fenn

Literaturverwaltung mit Firefox, Zotero und BIBTEX

Der Beitrag stellt eine neuentwickelte Literaturverwaltung namens Zotero vor, die als Plugin in Firefox 2 eingebunden wird. Damit können auch BIBTEX-Daten verarbeitet werden.

Das Web 2.0 entlässt seine Kinder

Der Hype um das „Web 2.0" in den einschlägigen Foren, Blogs und Zeitschriften ist gerade etwas am Abklingen, da tauchen die ersten ernsthafteren Anwendungen auf, die – abseits von den mittlerweile allgegenwärtigen RSS-Feeds, Widgets und Podcasts – auch dem technisch weniger versierten Nutzer einen guten Eindruck davon vermitteln, was für ein Potential tatsächlich im „semantischen Web" [9] stecken könnte und insbesondere wie sich die Technik des wissenschaftlichen Arbeitens durch ihren Einsatz schon sehr bald nachhaltig ändern wird.

Vor allem die Arbeit mit Online-Bibliographien und BibTeX [3] kann mithilfe des Add-On **Zotero** [12] zu dem Browser **Firefox 2** ganz wesentlich erleichtert werden. Aber auch die Arbeit mit anderen online verfügbaren Quellen in wissenschaftlichen Arbeiten vereinfacht sich damit teilweise erheblich.

Die Funktionsweise

Im Gegensatz zu einem grafischen Frontend wie **Jabref** [5], das BibTeX-Dateien unter einer grafischen Oberfläche verwaltet, kann man mit Zotero ganze Projekte organisieren. Hiermit können die zu zitierenden Quellen selbst ebenso wie die ihnen zugehörigen Metadaten systematisch archiviert, katalogisiert, kommentiert und untereinander verknüpft werden.

Die *Wikipedia* [10] zitiert zwei Meldungen, wonach die nächsten Versionen der Browser **Firefox** und **Internet Exporer** sogenannte *Mikroformate* nativ unterstützen werden. Web-Seiten können mit dieser Technik, vom Benutzer unbemerkt, mit Informationen über die Bedeutung des Texts, der Bilder oder anderer Objekte, die in ihnen enthalten sind oder die über sie abgerufen werden können, versehen werden. Diese zusätzlichen (Meta-)Informationen sind im HTML/XHTML-Quelltext der Seite enthalten. Hierfür können etwa die HTML-Tags **div** und **span** sowie das class-Attribut verwendet werden (vgl. hierzu bereits [4, Abschn. 7.4.1.4]). Diese Informationen können deshalb vom Browser, ebenso wie der übrige Inhalt einer Web-Seite, geparst und automatisch weiterverarbeitet werden.

Für unser Thema sind dabei natürlich vor allem diejenigen Anwendungen von Interesse, die bibliographische Daten bereitstellen. Insbesondere das COinS-Format

(„**ContextObjects in Spans**") hat sich mittlerweile etabliert, es wird beispielsweise auch in der *Wikipedia* verwendet. [11] Mithilfe von Mikroformaten stehen beim Aufruf einer Web-Seite im Browser sofort weitere Informationen zur Verfügung, die dann auch in eine Literaturverwaltung übernommen werden können. Ein übergreifender Standard namens „*cite*" zur Auszeichnung von Zitaten befindet sich derzeit in der Diskussion. [6]

Hier setzt **Zotero** als Erweiterung von **Firefox 2** an, denn es wird als Plug-In betrieben und arbeitet deshalb nahtlos mit dem Browser zusammen, kann also direkt auf die Inhalte zugreifen, die im Browser beim Surfen im Netz ankommen. `Zotero` ist eine Literaturverwaltung im Browser, die alle möglichen online verfügbare Inhalte sammeln und später weiterverarbeiten kann. Sie ist auf jeder Plattform verfügbar, auf der **Firefox** läuft, also praktisch auf jedem heute gebräuchlichen Computer – ebenso wie TEX & *Friends*.

Eine Anwendung wie diese ist natürlich auf die Unterstützung der Anbieter im WWW angewiesen. Die Zahl der Seiten, die Mikroformate verbreiten, hatsich in der letzten Zeit schon beträchtlich vergrößert. Einen Überblick einschlägiger Seiten findet man unter http://www.`zotero`.org/translators/. Hier finden sich nicht nur „große Namen" wie *Amazon*, arXiv.org, die Zeitschrift *Nature*, *Google Books* und *Google Scholar* sowie *CiteSeer* und die *Wikipedia* [11], sondern auch viele amerikanische Bibliothekskataloge, vor allem natürlich die *Library of Congress*. Der *Gemeinsame Bibliotheksverbund GBV* unter http://www.gbv.de/vgm/ ist der erste deutsche Verbundkatalog, der mit **Zotero** zusammenarbeitet.

Mit Zotero arbeiten

Die Installation von **Zotero** ist denkbar einfach, sie erfolgt wie bei anderenPlugins auch automatisch beim Herunterladen bzw. beim Öffnen der xpi-Datei mit **Firefox** (Download-Link auf der Startseite der Projekt-Homepage http://www.`zotero`.org/). Das Plug-In steht nach einem Neustart des Browsers zur Verfügung, was an dem **Zotero**-Logo unten rechts in der Statuszeile des Browsers erkennbar ist. Wie bei anderen Plugins auch, ist ein automatisches Update möglich, wenn eine neue Version bereitsteht.

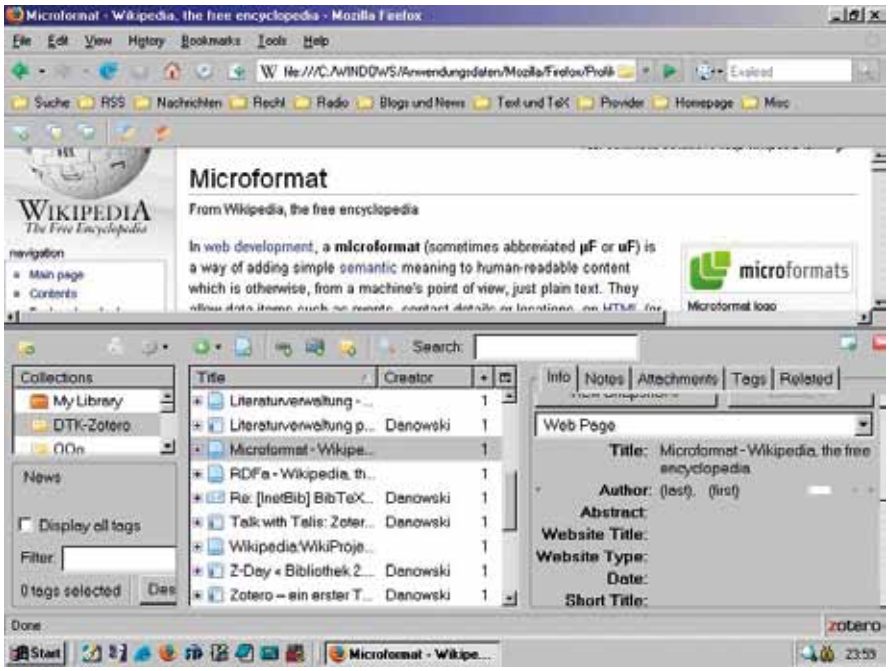


Abbildung 1: **Zotero**-*Oberfläche bei der Darstellung des gespeicherten shapshot einer HTML-Seite*

Klickt man auf das Logo, öffnet sich die grafische Benutzeroberfläche im unteren Teil des Browserfensters. Sie ist in drei Spalten geteilt, welche – von links nach rechts – sämtliche Quellen, organisiert in Ordnern (und Unterordnern); die einzelnen Quellen in dem jeweils markierten Ordner der Sammlung; sowie die bibliographischen Angaben eines im mittleren Fenster markierten Objekts zeigen (vgl. Abb. 1).

Per Kontextmenü können Titel ordnerweise oder einzeln in eine Datei zur weiteren Verarbeitung exportiert werden. Unterstützt werden neben BibTeX derzeit die Formate MODS, Endnote, RIS, Unqualified Dublin Core XML und die *Wikipedia*-Zitiervorlage.

Zotero speichert ganze Web-Seiten als snapshot komplett ab, so dass sie später lokal wieder angezeigt werden können. Jede Quelle kann – anscheinend ohne Begrenzung – mit beliebigen Notizen, Such-Tags oder Anhängen versehen werden. Es steht eine sehr große Zahl an Quellentypen zur Verfügung, von Büchern und Aufsätzen über Blog Postings bis hin zu Gerichtsentscheidungen, die bekanntlich zunehmend online im Volltext verbreitet und zitiert werden. Natürlich kann diese Datenbank durchsucht werden. Die gesamte collection kann in dem eigenen Format „**Zotero** RDF" mit allen Anhängen und Notizen gespeichert werden, um ein Backup zu erstellen.

Erfahrungen und Probleme

Obwohl sich das Programm derzeit noch im Beta-Stadium befindet, ist es schon ziemlich leistungsfähig. **Zotero v1.0.0b4.r5** vom August 2007 lief im Test stabil.

Was für die Arbeit mit Online-Bibliographien allgemein gilt, [1, S. 91] ist auch hier zu beachten: Blind kann man dem Output nicht vertrauen. Fast alle Zitate des vorliegenden Artikels wurden mit **Zotero** erzeugt und im Emacs weiterbearbeitet. Alle BibTeX-Datensätze, die aus **Zotero** exportiert wurden, waren formal valide, aber inhaltlich unvollständig, so dass sie händisch ergänzt werden mussten. Zitierte URLs mussten z.B. von dem **url**-Feld in das **note**-Feld des Eintrags verlegt werden, weil der Bibliographiestil der DTK das Feld **url** nicht unterstützt. Bei den Artikeln aus der *Wikipedia* ist es derzeit sogar einfacher, deren eigene Zitierfunktion [3, S. 44f] zu verwenden.

Sucht man beispielsweise nach dem „BibTeX-Begleiter" [7] im GBV, so wird in der Adresszeile des Browsers rechts ein Dokumentensymbol angezeigt. Klickt man hierauf (wie beim Abonnieren eines RSS-Feeds), speichert **Zotero** die bibliographischen Angaben zu dem Buch. Exportiert man sie ins BibTeX-Format, erhält man einen Eintrag, der nach einer leichten Nachbearbeitung im Emacs wegen des Zeilenumbruchs wie in Abbildung 2 aussieht. Hier fehlen nicht nur einige Autorennamen (nämlich David Carlisle, Chris Rowley und Joachim Schrod), sondern auch die Angabe der Auflage, die im Katalog durchaus vorhanden ist. Auch der Schriftzug „LaTeX" wurde nicht korrekt erkannt und umgesetzt.

```
@book{mittelbach_der_2005,
   address =   {M{\"u}nchen [u.a.]},
   series =    {Scientific tools ; 7166},
   title =     {Der LATEX-Begleiter},
   isbn =      {3-8273-7166-X},
   publisher = {Pearson Studium},
   author =    {Frank Mittelbach and Michel
                Goossens and Johannes Braams
                and Christine Detig},
   year =      2005,
   keywords =  {LATEX <Programm> ,LATEX 2E }
}
```

Abbildung 2: Eintrag des LaTeX-Begleiters[7] aus dem GBV, wie er von `Zotero` *ins BibTeX -Formst exportiert wurde*

Der Import bestehender BibTeX-Dateien in **Zotero** funktioniert meist reibungslos. Kommt es hier zu Problemen, was auch bei zweifelsfrei validen bib-Dateien passieren kann – im Test war eine sehr große Datei betroffen –, ist es allerdings kaum möglich, den Fehler einzugrenzen, weil keine Fehlermeldungen erfolgen. Probleme ergeben sich vor allem bei der Weiterverarbeitung, wenn die Einträge LaTeX-Makros enthalten. Beim Import werden die geschweiften Klammern entfernt, der Backslash vor dem Makronamen wird aber übernommen. Exportiert man einen solchen Eintrag wiederum aus **Zotero** nach BibTeX, so wird ein zweiter

Backslash hinzugefügt, den man händisch entfernen muss. Auch der Export des maskierten scharfen S funktioniert derzeit nicht korrekt. Umlaute werden dagegen richtig behandelt. Anwender berichten auch darüber, dass machmal Autorennamen nicht vollständig exportiert würden, vielleicht ähnlich wie zuvor in Abb. 2 erkennbar.

Zusammenfassung und Ausblick

Zotero ist schon heute sehr hilfreich bei der Generierung von BibTeX-Einträgen für die eigene Bibliographie und sollte deshalb unbedingt einbezogen werden, wenn ein größeres Projekt mit umfangreicher Quellenarbeit bevorsteht. Die Entwicklung von **Zotero** verläuft sehr zügig, wie man in dem projekteigenen Blog unter http://www.zotero.org/blog/ und in dem Blog von Patrick Danowski [2] verfolgen kann. Deshalb ist anzunehmen, dass die derzeit noch verbliebenen Kinderkrankheiten bald behoben sein werden und **Zotero** ein wichtiges und hilfreiches Tool für die Arbeit mit BibTeX wird, vor allem weil die Unterstützung für Mikroformate in den Bibliothekskatalogen auch hierzulande bald noch weiter zunehmen wird. Updates können über einen Schalter in **Firefox** unter Tools|Add-ons automatisiert werden.

Verglichen mit den Lösungen, die in [3] vorgestellt worden waren, zeigt das Programm noch einige Schwächen, ist aber ganz sicherlich auf einem guten Weg.

Wer sich für die weitere Entwicklung bei Werkzeugen zur Literaturverwaltung interessiert, sollte übrigens die Blogs und die Mailinglisten für Bibliothekare, etwa das gerade genannte Blog von *Danowski* [2], verfolgen. Bekannt ist insbesondere die von *Michael Scharwächter* betriebene *INETBIB*-Mailingliste [8], die auch über ein durchsuchbares Archiv verfügt. Man kann die Liste auch über den Newsserver news://news.gmane.org lesen, wo sie unter gmane.culture.libraries.inetbib geführt wird.

Literatur

[1] Nelson H. F. Beebe: *A Bibliographer's Toolbox; TUGboat*; 25(1), S. 89–104; 2004; Proceedings of the Pratical TeX 2004 Conference, http://www.tug.org/TUGboat/Articles/tb25-1/beebe-bib.pdf.

[2] Patrick Danowski: *Bibliothek 2.0 und mehr*; http://www.bibliothek2null.de/?p=187, früher unter: http://bibliothek2. wordpress.com/.

[3] Jürgen Fenn: *Online-Bibliographien nutzen mit BibTeX*; *Die TeXnische Komödie*; 4/06, S. 40–46; 2006.

[4] Michel Goossens, Sebastian Rahtz et al.: *The LaTeX Web Companion. Integrating TeX, HTML, and XML; Tools and Techniques for Computer Typesetting*; Addison-Wesley; Boston et.al.; 1999; eighth printing, January 2006.

[5] JabRef Projekt-Homepage: http://jabref.sourceforge.net/.

[6] Microformats Wiki: *citation – Microformats*; 2007; http://microformats.org/wiki/cite, Zugriff am 13. August 2007.

[7] Frank Mittelbach, Michel Goossens et al.: *Der LaTeX-Begleiter; Scientific tools* ; 7166; Pearson Studium; München [u.a.]; 2. Aufl.; 2005.

[8] Michael Scharwächter: *INETBIB-Mailingliste. Internet in Bibliotheken*; http://inetbib.de/.

[9] W3C: *Semantic Web Activity*; http://www.w3.org/2001/sw/.

[10] Wikipedia: *Microformat – Wikipedia, the free encyclopedia*; 2007; http://en.wikipedia.org/wiki/Microformat, Zugriff am 13. August 2007.

[11] Wikipedia: *Wikipedia:WikiProject Microformats/COinS – Wikipedia, the free encyclopedia*; 2007; http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Microformats/COinS, Zugriff am 13. August 2007.

[12] Zotero Projekt-Homepage: *Zotero -The Next-Generation Research Tool*; 2007; http://www.zotero.org/.

Dr. Jürgen Fenn, Friedensallee 174 / 20, D-63263 Neu-Isenburg, e-Mail: juergen.fenn@gmx.de, Internet: http://www.juergenfenn.de/tex.html

Albert Badior

Galileo Galilei und die Wurfparabel

Galileo Galilei wird am 15. Februar 1564 in Ammanti geboren. Mit 25 Jahren wird er zuerst Dozent für Mathematik an der Universität von Pisa und danach an der Universität von Padua. Im Jahr 1604 entdeckt er das Fallgesetz und 1608 die Flugparabel, ohne seine Erkenntnisse zu veröffentlichen. Angeregt durch das Erscheinen eines „neuen Sterns" am Himmel – einer Supernova – wendet sich Galilei der Erforschung des Sternenhimmels zu, indem er das kurz zuvor in Holland entwickelte Teleskop einsetzte. Er entdeckte dabei u.a. die Sonnenflecken und die Saturnringe und veröffentlichte diese Erkenntnisse, auf deren Grundlage er sich für die Richtigkeit des von Kopernikus (1473 –1543) postulierten heliozentrischen Weltbildes ausspricht. Im Jahr 1632 erscheint sein Buch „Dialog über die beiden hauptsächlichen Weltsysteme, das ptolemäische und das kopernikanische".

Die Ansichten, die Galileo Galilei darin äußert, stehen im Gegensatz zur Lehrmeinung der katholischen Kirche; Galilei wird von der Inquisitionsbehörde angeklagt und widerruft seine Aussagen 1633. Er wird zu dauerndem Hausarrest in seinem Landhaus verurteilt; dort schreibt er sein für die Entwicklung der Physik wichtigstes Werk, die „ Unterredungen und mathematischen Demonstrationen über zwei neue Wissenszweige, die Mechanik und die Fallgesetze betreffend". Es wurde 1638 in Holland veröffentlicht, also 30 Jahre nachdem er das Fallgesetz und die Parabellösung gefunden hatte.

Die Fallgesetze und die parabelförmige Wurfbahn werden in den Kapiteln mit den Überschriften „Der dritte Tag" und „Der vierte Tag" behandelt – nur sie werden hier im weiteren betrachtet.

Zuvor soll jedoch kurz das „naturphilosophische" Umfeld geschildert werden, in dem Galilei sich befand.

Im 12. und 13. Jh. wurden die Schriften des Aristoteles (384 – 328 v. Chr.) aus dem arabischen und dem griechischen ins Latein übersetzt und damit einem Teil der gebildeten Schichten in Europa zugänglich. Einen großen Anteil an den Übersetzungen hatten arabische, jüdische und christliche Gelehrte, die im islamischen Kalifat von Cordoba lebten. Eine andere Quelle waren zurückkehrende Kreuzritter, die viele antike Manuskripte u.a. aus Byzanz mitbrachten.

Im 13. Jh. vereinigte Thomas von Aquin diese Theorien mit der christlichen Glaubenslehre zu einem einheitlichen Weltbild, worin die Wissenschaft jedoch als die „Magd der Theologie" aufgefasst wurde.

Die Aussagen des Aristoteles zur Bewegungslehre können wie folgt zusammengefasst werden:

- es gibt eine *natürliche Bewegung* , mit der alle Körper ihrem *natürlichen Ruheort* zustreben. Danach ist z.B. Erde nur schwer und Feuer nur leicht; schwer ist Wasser in Luft und Luft in Feuer; leicht ist dagegen Wasser in Erde und Luft in Wasser. Ein Körper bewegt sich mit zunehmender Geschwindigkeit seinem *natürlichen Ort* zu. Deshalb fallen Steine zu Boden und Rauch steigt in den Himmel.

- Es gibt eine *erzwungene Bewegung*, die durch eine wirkende Ursache und durch unmittelbaren Kontakt auf den Körper übertragen wird. Beim Steinwurf z.B. geht die Kraft des Menschen auf die umgebende Luft über, wenn der Stein die Hand verlassen hat – die Luft ist somit gleichzeitig Antriebsmedium als auch Verzögerungsmedium durch den Luftwiderstand. Die *erzwungene Bewegung* verlangsamt sich, bis sie in die *natürliche Bewegung* übergeht. Als Folge dieses Gedankens musste Aristoteles das Vakuum ablehnen.

Für Aristoteles war die Bewegung ein Prozess und nicht ein Zustand, wie wir es seit Isaac Newton wissen.

Bereits in der Antike und auch im frühen Mittelalter wurden die Gedanken Aristoteles zu den Bewegungsgesetzen kritisiert. Philoponos (ca. 490 –570 n. Chr.) geht davon aus, dass es eines *Bewegers* bedürfe, der dem Körper ein *inneres Bewegungsvermögen* erteilt. Die Bewegung dauert nach der Trennung vom Beweger und ohne Mitwirkung des umgebenden Mediums an. Das innere Bewegungsvermögen wird langsam verbraucht, da der Körper Widerstände zu überwinden hat und der Körper kommt zum Stillstand. Diese Gedanken wurden im 14. Jh. vom Rektor der Universität Paris – Jean Buridan – weiterentwickelt und zur *Impetustheorie* ausgeformt. Diese Theorie dehnte er auch auf die Bewegungen der Himmelskörper aus. Die Impetustheorie schloss das Vorhandensein eines Vakuums nicht aus.

Das Ziel dieser Theorie, ein neues ganzheitliches Weltbild zu formulieren, das das des Aristoteles und das des Thomas von Aquin ablösen sollte, wurde nicht erreicht. Der Ansatz von Buridan blieb ein metaphysischer Erklärungsversuch, der im Geist der Scholastik gefangen war, der aber einen Weg für die Weiterentwicklung der Dynamik aufgezeigt hat.

Die Naturphilosophie, als Vorläufer der Physik, war in weiten Teilen dadurch gekennzeichnet, das die Richtigkeit von theoretische Ansätzen nicht durch Versuche verifiziert wurde; man bestätigte oder verwarf neue Ideen auf der Basis von Diskussionen und/oder der Lektüre anderer Veröffentlichungen.

Galilei war davon überzeugt, dass nur mit Hilfe der Mathematik tiefere Einsichten in die Zusammenhänge der Naturgesetze zu finden seien. Dieser Gedanke und sein Postulat, dass die Theorie durch Experimente verifiziert werden müsse, führten den Bruch mit den bisherigen Lehrmeinungen herbei.

Was die mathematischen Grundlagen zur Zeit Galileis betrifft, so waren sie zu Beginn des 16. Jh. nicht weit über das Niveau des Archimedes hinaus gekommen. Von den bis 1545 erschienenen Büchern beschäftigten sich nur etwa 10 % mit den Gebieten der Mechanik, Arithmetik, Geometrie und Astronomie; mit theologischen Themen jedoch 26 % [6; S.175].

Langsam setzten sich die Dezimalschreibweise von Zahlen und die mathematischen Symbole +, -, ×, ÷, =, √ durch und die Trigonometrie wurde ausgebaut. Im Vordergrund stand noch die klassische Geometrie,

mit deren Hilfe die meisten Beweise geführt wurden.

Etwas, was für uns heute selbstverständlich ist – Größen mit verschiedenen Dimensionen formelmäßig zueinander in Beziehung zu setzen, wie z.B. Strecken zu Zeiten, um Geschwindigkeiten auszudrücken – war unbekannt.

Galilei und seinen Zeitgenossen war es nur möglich, Größen miteinander zu verbinden, wenn diese zueinander „homogen" waren, also z.B. nur Strecken oder nur Dreiecke oder nur Rechtecke.

Die analytische Geometrie, die Galilei die Arbeit sehr erleichtert hätte, entstand erst auf der Grundlage der Gedanken von Fermat und Descartes – also in einer Zeit, in der Galilei seine Arbeiten schon abgeschlossen hatte. Im folgenden Abschnitt werden die Gedanken von Galilei zum Thema *Bewegung* dargestellt.

Zuerst behandelt er die gleichförmige Bewegung, die wie folgt definiert: *„Ich nenne diejenige Bewegung gleichförmig, bei welcher die in irgend welchen gleichen Zeiten vom Körper zurückgelegten Stecken untereinander gleich sind"* [2; S.141]. Danach folgen seine „Axiome" und „Theoreme", in denen er den Begriff der Geschwindigkeit nur in qualitativer Bedeutung verwendet. Die Strecken und Zeiten werden nur geometrisch dargestellt und zueinander in Beziehung gesetzt.

Zuerst äußerte Galileo Galilei diese Gedanken 1593 in dem für seine Studenten verfassten Text „Le Mecaniche". Er überlegte sich, dass eine ideale Kugel auf einer idealen Ebene indifferent gegenüber dem Zustand der Ruhe oder der Bewegung sein müsse; indifferent in dem Sinn, dass sie sich ständig mit gleicher Geschwindigkeit bewege, wenn sie einmal angestoßen sei und dass sie danach keine Kraft mehr benötige.

Der freie Fall eines Körpers war bereits vor Galilei gedanklich untersucht worden und man vertrat die folgende Ansicht: Verlässt ein Körper seine Ruhelage, folgt eine kurze Übergangsphase der Beschleunigung. Am Ende dieser Phase hat der Körper seine Endgeschwindigkeit erreicht, mit der er dann weiter fällt.

Galilei stellte diese Aussage in Frage und untersuchte gedanklich zwei Ansätze, in denen er die Vakuumprämisse aufrecht erhielt und die Erdanziehung berücksichtigte. Der erste Ansatz ging davon aus, dass sich die Strecken (s) proportional zur Geschwindigkeit (v) verhalten ($s \sim k*v$; mit einem Faktor k = const.). Diesen Ansatz hat er aber verworfen, weil er auf Grund von Experimenten zu einer anderen Einsicht kam. Es deutet nichts darauf hin, dass er aus der falschen Prämisse dieses Ansatzes gedanklich zum richtigen Ergebnis kam, wie es teilweise in der Literatur behauptet wird [1; S.29] und [6; S.201,204]. Galilei betont dagegen an verschiedenen Stellen, dass er zu seinem zweiten Ansatz ($s \sim t^2$) durch Versuche kam [2; S. 149, 158, 158]. Im freien Fall verhalten sich die Strecken wie die Quadrate der Zeiten und für die Geschwindigkeiten folgt dann daraus: $v \sim t$. Wie unklar der Begriff „Geschwindigkeit" damals war, zeigen auch die anderen von Galilei dafür verwendeten Begriffe wie „Moment" , „Impuls" oder „Geschwindigkeitsmoment".

Betrachten wir nun den Versuchsaufbau von Galilei; er verwendete ein Holzbrett von 12 Ellen Länge mit einer Rinne, in der er eine polierte Messingkugel herabrollen ließ, nachdem er das Brett am Ende mal um eine Elle und dann um zwei Ellen anhob [2; S. 162]. Galilei hatte den Einfluss des Rollwiderstandes erkannt und kleidete die Rinne zu dessen Verringerung mit feinstem Pergament aus.

Die Zeitmessung für die zurückgelegten Strecken erfolgte zuerst durch die Zählung

der Pulsschläge und durch das Schlagen von Taktfolgen – Galilei war ein guter Musiker und kam wohl deshalb auf diese Idee. Er stellte aber fest, dass die damit erzielbare Genauigkeit seinen Anforderungen nicht genügte. Nun konstruierte Galilei eine Wasseruhr; aus einem Gefäß floss Wasser in einem sehr feinen Strahl in einen Napf – die Durchflussmengen wurden nach jedem Rollversuch mit einer Apotheker- oder Münzwaage gewogen und zu den durchlaufenen Strecken in Beziehung gesetzt.

Die Ergebnisse führten Galilei auf den Ansatz ($s \sim t^2$). Geht man von unserem heutigen Wissenstand aus ($s \sim 0{,}5 \times g \times t^2$); mit g als Fallbeschleunigung und einem Wert von ca. 9,81 m/s^2, hat er, bedingt durch den Rollwiderstand diese Zahlenwerte nicht erreicht. Den Charakter des Fallgesetzes mit seiner stetigen Zunahme der Geschwindigkeit hat er aber klar erkannt. Die Aussagen in einigen Stellen der Literatur, dass Galilei am „Schiefen Turm" von Pisa aus Fallversuchen das Gesetz ermittelt habe, sind wohl Legenden. Ein im freien Fall befindlicher Körper konnte damals nicht gemessen werden, dazu war man erst viel später in der Lage.

Aus den Versuchsergebnissen leitet Galilei dann geometrisch ($s \sim t^2$) her, indem er nur mit „qualitativ" gleichartigen („homogenen") Größen arbeitet, wie es ja dem Stand der damaligen Mathematik (s.o.) entsprach [2; S. 160] und wie es von ihm in Bild 1 dargestellt ist. Der Abbildung Galileis hat der Verfasser am rechten Rand die Zeitintervalle t_0 bis t_3 angefügt.

Die Aussage ($s \sim t^2$) ergibt sich, indem man die Dreiecksflächen von ABC, AFI und APO zueinander in Beziehung setzt; man erhält dann die Verhältnisse von 1 : 4 : 9.

Vergleicht man die zu den einzelnen, gleichlangen Zeitintervallen gehörigen Dreiecksflächen miteinander, ergibt sich die Folge der ungeraden Zahlen 1, 3, 5,...,. Die in gleichen Zeitintervallen durchlaufenen Strecken verhalten sich somit wie diese Folge.

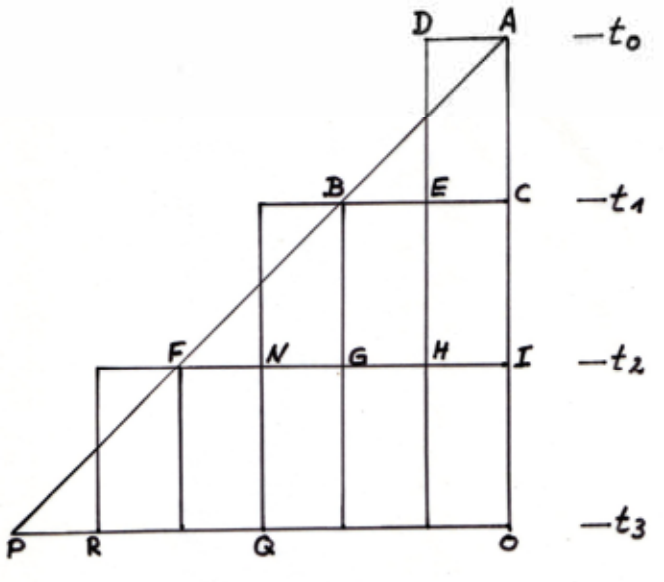


Bild 1: Galileis geometrische Herleitung des Ansatzes $s \sim t^2$ (Zeitintervalle ergänzt)

Die waagerechten Strecken wie z.B. BC oder FI oder PO können als Geschwindigkeiten in den Zeitpunkten t_1, t_2 und t_3 gedeutet werden.

Nach der Formulierung der Gesetze von der gleichförmigen Bewegung und der gleichförmig beschleunigten Bewegung interessierte sich Galilei dafür, wie der Bahnverlauf eines geworfenen Körpers im Vakuum aussieht, wenn beide Bewegungen zusammen auftreten. Er vermutete, dass es eine Parabel sein müsse. („M*an hat beobachtet, dass Wurfgeschosse eine gewisse Curve beschreiben; dass letztere aber eine Parabel sei, hat niemand gelehrt*." [2; S.140]. Seine Annahme gründete Galilei auf die Schriften des Appolonius und dessen Abhandlung über die Kegelschnitte. Mit seinen, aus den Versuchen gewonnenen Erkenntnissen, konnte er nun geometrisch eine Parabel zeichnen. Galilei beschränkt sich dabei immer auf die Darstellung einer Halbparabel, wie sie in Bild 2 zu sehen ist.

Galilei und die Wurfparabel

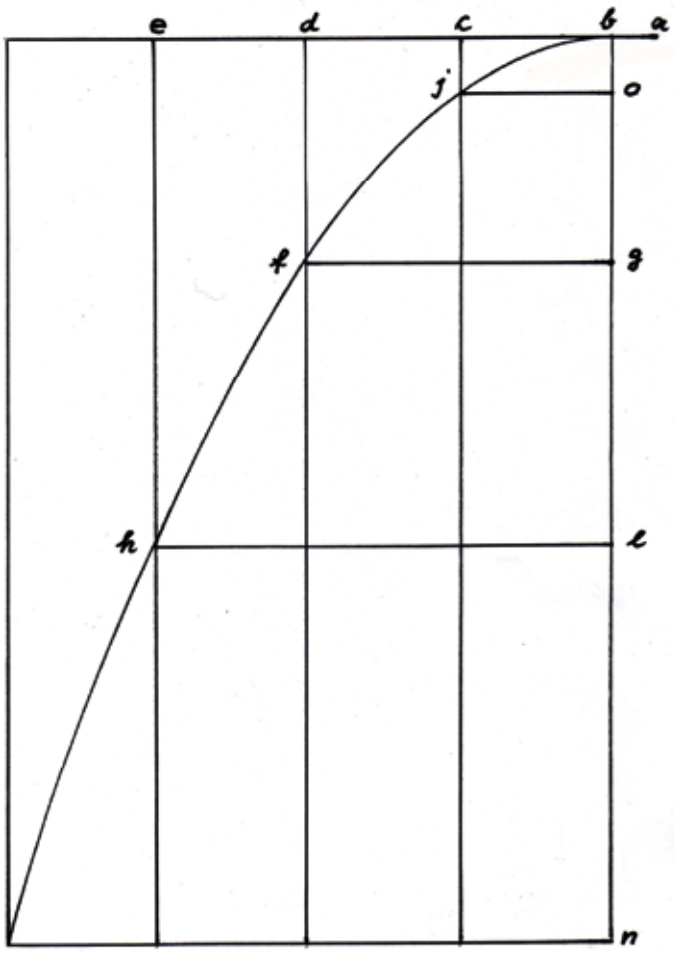


Bild-2: Galileo Galileis Darstellung einer Halbparabel

Galilei betrachtet die Entwicklung der Parabel vom Punkt b aus. Die horizontale Entfernung von b bis zum Auftreffpunkt bezeichnet er als „Amplitude". Im Punkt b muss der Wurfkörper eine Horizontalgeschwindigkeit besitzen, um überhaupt eine Bahnkurve beschreiben zu können. Da Galilei, wie oben beschrieben, nur mit „homogenen" Größen rechnen konnte, nahm er an, dass der Körper aus einer gewissen Höhe über b herabfalle und beim Eintreffen in b horizontal abgelenkt würde. Mit diesem Trick erzeugte Galilei gedanklich die horizontale Anfangsgeschwindigkeit im Punkt b. Dieser Sachverhalt ist in Bild 3 dargestellt.

Die Strecken haben die folgenden Bedeutungen:

AC : Gipfelhöhe (Yg) der Bahn
CF = AD : Gipfelweite (Xg), bei Galilei = Amplitude
EA : Fallhöhe des Körpers (Ys) zur Erreichung der horizontalen Anfangsgeschwindigkeit. Galilei bezeichnet sie als „Sublimität".

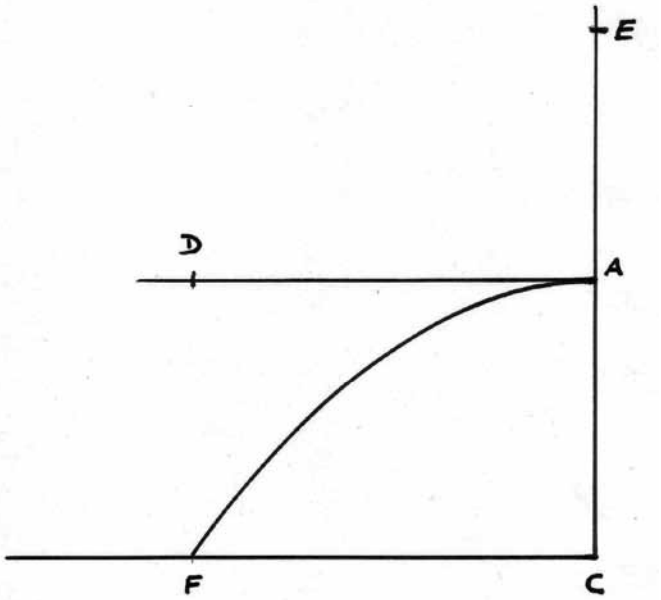


Bild-3: Horizontale Anfangsgeschwindigkeit im Punkt b

Galilei ermittelt Ys auf geometrischem Weg, indem er Strecken zueinander in Beziehung setzt, wofür er einige Seiten benötigt.

In unserer heutigen Schreibweise lautet die Formel:

$$Ys = \left(\frac{Xg}{2}\right)^2 \times 1 \div Yg$$

Danach bestimmt Galilei geometrisch die Geschwindigkeiten in den einzelnen Punkten der Parabel, insbesondere die Auftreffgeschwindigkeit auf dem Boden (Punkt F in Bild 3). Die Bestimmung dieser Geschwindigkeit ist deshalb von Interesse, weil damit die Anfangsgeschwindigkeit (V_0)

eines Wurfkörpers ermittelt wird, wenn die Parabel von F nach A durchlaufen wird. In unserer heutigen Notation ist das :

$$V_0 = \sqrt{2 \times g \times (Ys + Yg)}$$

Auf Grund der Symmetrieeigenschaft der Parabel gilt:

Gesamtwurfweite (Xe) =2 × Xg und Endgeschwindigkeit (Ve) = V_0, wenn Abwurfhöhe = Auftreffhöhe. Den heutigen Begriff der Anfangsgeschwindigkeit bezeichnet Galilei mit „Impuls".

Galilei stellt noch weitere Betrachtungen zur Parabel an, die aber für das hier behandelte Thema keine weiteren Erkenntnisse bringen.

Aufbauend auf seinen Ergebnissen entwickelt er eine Anzahl von Tabellen, die man heute als „Schusstafeln" für den luftleeren Raum bezeichnen könnte. So stellt er Tafeln auf für:

a) Die Werte von Yg bei verschiedenen Abgangswinkeln (δ_0) und konstanter V_0, die bei Galilei als „gleiche Impulse" bezeichnet werden.

$$Yg = \frac{V_0^{\,2}}{2 \times g} \times \sin^2 \delta_0$$

b) Die „Amplituden" der Halbparabeln (Xg) bei verschiedenen δ_0 und konstanter V_0.

$$Xg = \frac{V_0^{\,2}}{2 \times g} \times \sin 2 \times \delta_0$$

c) Die Höhen (Yg) und Sublimitäten (Ys) von Halbparabeln gleicher „Amplitude" (= 10.000 Einheiten) bei verschiedenen δ_0.

$$Yg = 0{,}5 \times Xg \times \tan \delta_0$$

$$Ys = 0{,}5 \times Xg \times \cot an \, \delta_0$$

Die obigen Formeln sind in der heute üblichen Notation geschrieben.

Es war nicht Galileis Absicht, diese Tabellen als Schusstafeln für Feuerwaffen im lufterfüllten Raum gewertet zu wissen.

Galilei sagt in Hinblick auf die zu seiner Zeit nicht messbare Geschwindigkeit von Geschossen für Feuerwaffen: *„Mich veranlasst dazu die ungeheuere, sozusagen übernatürliche Wucht solcher Geschosse, dass ich selbst ohne Übertreibung sagen möchte, dass die Geschwindigkeit , mit der eine Flinten- oder Kanonenkugel den Lauf verlässt , übernatürlich genannt werden könnte*" [2; S.227]. Er geht auch auf den Luftwiderstand und seine Wirkung auf die Form der Flugbahn ein: *„Der übermäßige Impuls solcher Kraftgeschosse kann die Wurflinie ändern; der Anfang der Parabel wird weniger geneigt sein, das Ende stärker gekrümmt.*" [2; S. 228]. Damit hat er, wie wir heute wissen Recht.

Auch auf den Einfluss der Querschnittsbelastung, des Verhältnisses von Geschossmasse zu seiner Querschnittsfläche, geht er ein. Er sagt völlig korrekt, dass eine große Querschnittsbelastung günstig für die Überwindung des Luftwiderstands sei.

Die Anwendbarkeit der Parabel auf Flugbahnen im lufterfüllten Raum lässt Galilei nur für Bogenpfeile, Armbrustbolzen und Mörsergeschosse gelten [2; S. 225,228]. Bei diesen Projektilen handelt es sich um Geschosse mit kleiner V_0 (< 60 m/s) und großer Querschnittsbelastung; diese weisen in der Praxis tatsächlich nahezu parabelförmige Flugbahnen auf. Das gleiche gilt für Mörsergeschosse, auch sie haben kleine Anfangsgeschwindigkeiten bei großen Geschossmassen. Mörser sind kurzrohrige Artilleriewaffen für Steilfeuer. Nur für diesen Waffentyp konnte das Militär Nutzen aus Galileis Tabellen ziehen; Pfeil und Bogen sowie Armbrüste waren zu Galileis Zeit längst von Feuerwaffen verdrängt worden.

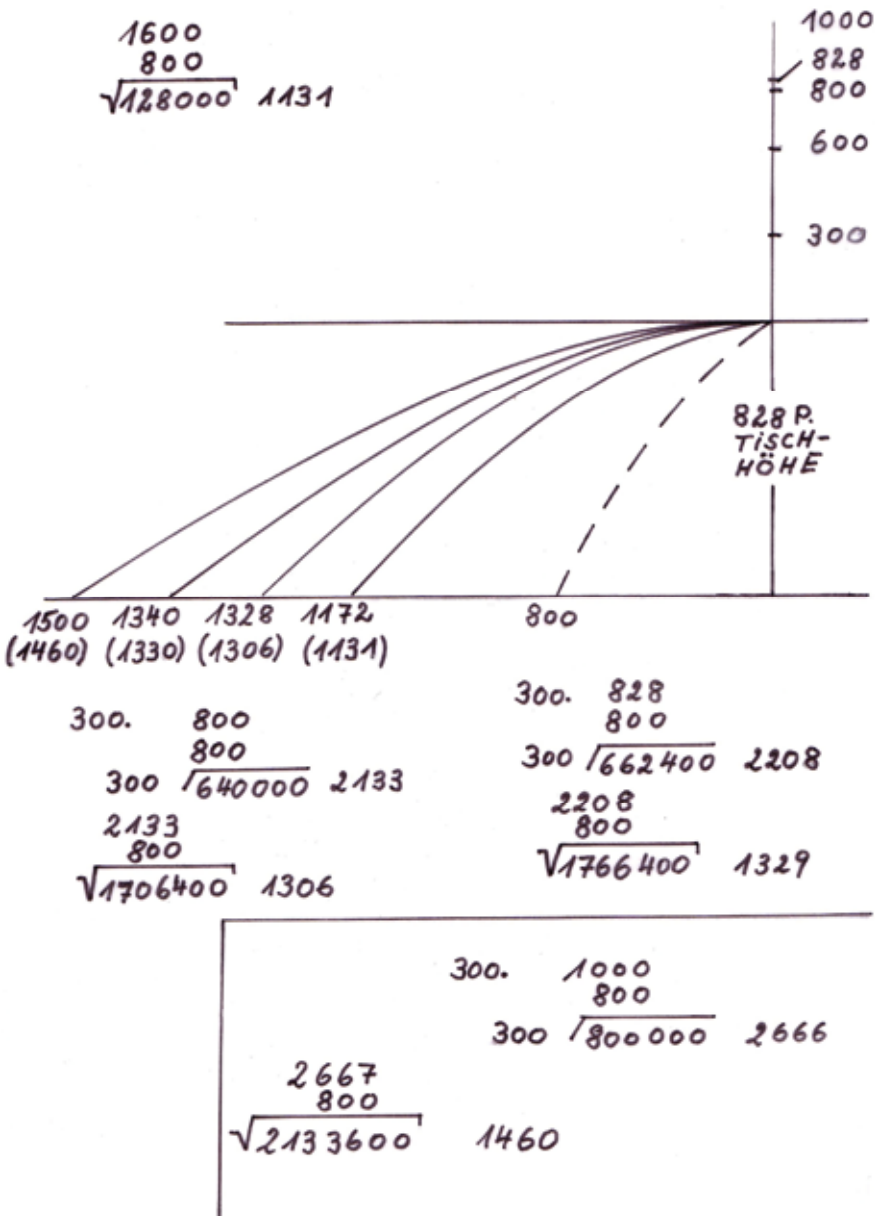
Galilei und die Wurfparabel

In der Literatur ist wiederholt bestritten worden, dass Galilei zur Verifizierung seiner Parabeltheorie Versuche angestellt habe. So hat u.a. der französische Wissenschaftshistoriker Alexandre Koyré (1892 – 1964) geschrieben, *„diese Experimente seien Anfang des 17. Jh. mit den verfügbaren technischen Möglichkeiten nicht zu realisieren gewesen; sie müssten daher Gedankenexperimente gewesen sein, die Galilei, um seine Gegner und Kritiker zu überzeugen, dann so aufgezeichnet habe, als seien sie in seinem Laboratorium realisiert worden.*" [1; S. 45].

Gegen Ende der 60-er Jahre des 20. Jh. fand man Galileis Notizen, die sich mit den Fragen der zusammengesetzten Bewegung befassten. Sie befanden sich im Band 72 in der Nationalen Zentralbibliothek in Florenz und waren nicht in den gesammelten Werken Galileis veröffentlicht worden. Diese Quelle nutzten S. Drake und J. MacLachlan, um zu untersuchen, ob Galilei tatsächlich Versuche angestellt hat. Dazu baute MacLachlan ein geneigtes Brett mit einer glatten Rinne, in der eine Kugel lief; er empfand also den von Galilei beschriebenen Aufbau nach. Am unteren Ende der Rinne war ein Deflektor angebracht, so dass die Kugel in horizontaler Richtung die Rinne verließ. Der Neigungswinkel des Brettes betrug 30 Grad; Das Aussehen des Versuchsaufbaus glich einer kleinen Ski-Sprungschanze. Danach wurde die Versuchsanordnung so auf einem Tisch befestigt, dass der Abstand zwischen Deflektor und Boden 828 Punti betrug (1 punt ~ 0,9389 mm), so wie es in Galileis Zeichnung steht.

In der Zeichnung Galileis ist die Rinne mit dem Deflektor nicht zu sehen, es sind nur die Ergebnisse der Würfe mit ihren Istwerten, den Sollwerten und den Differenzen zu den Sollwerten angegeben. In Bild 4 sind die Istwerte an der Abszisse eingetragen, die dazu gehörigen Sollwerte stehen in runden Klammern; auf die Angabe der Differenzen hat des Verfasser verzichtet.

Bild 4: Zeichnung Galileis zur zusammengesetzten Bewegung (Werte nachgetragen)



Die Wurfbahnen beginnen alle im Koordinatenursprung; auf der Ordinate sind die Höhen („Sublimitäten") eingetragen, von denen aus jeweils die Kugel losgelassen wurde.

Die gestrichelte Flugbahn diente Galilei als Referenzbahn für die weiteren Wurfversuche; sie erfolgte mit einer Sublimität von 300 Punti und erreicht eine Weite von 800 Punti .

Die mit dem Versuchsaufbau von MacLachlan in Toronto 1972 erzielten Weiten wichen nur um – 0,7 % bis 2,8 % von den Istwerten Galileis ab. [1; S. 109].

Damit kann es als gesichert gelten, dass Galilei diese Versuche wirklich durchgeführt hat.

Zusätzlich sind in der Zeichnung Galileis auch vier Berechnungen enthalten, auf die Drake und MacLachlan nicht eingehen. Jede Berechnung besteht aus zwei Schritten und dient zur Ermittlung der Sollwurfweite. Am Beispiel für die Bahn mit der größten Weite (1460 Punti) soll der Rechengang interpretiert werden; er ist in der Zeichnung unten rechts eingerahmt. Galilei verwendet das uns bekannte Symbol für die Quadratwurzel, nicht jedoch Symbole für die Multiplikation und Division.

Die beiden Zahlen mit dem Wert 300 stellen die Sublimität der Referenzbahn (Ysr) dar; die Zahlen 800 sind die Wurfweite der Referenzbahn (Xr). Die Zahl 1000 ist die Sublimität für die zu berechnende Bahn (Ysa).

Der erste Rechenschritt liest sich dann wie folgt:

1000×800 = 800000 ÷ 300 = 2666

und der zweite Schritt:

$$\sqrt{2667 \times 800} = 1460$$

Im zweiten Schritt hat Galilei aus dem Ergebnis des ersten Schritts = 2666 die Zahl 2667 gemacht.

Unter Nutzung der oben eingeführten Symbole folgt:

$$\mathrm{Ysa} \times \mathrm{Xr} \div \mathrm{Ysr} = \mathrm{A}$$

$X = \sqrt{A \times Xr}$; worin X die Sollwurfweite ist.

Die beiden Schritte lassen sich vereinfachen und zusammenfassen zu:

$$X = \frac{\sqrt{Ysa}}{\sqrt{Ysr}} \times Xr$$

Hier wird sichtbar, dass Galilei nur „homogene" Größen (s.o.) - hier Strecken – zueinander in Beziehung gesetzt hat, um die Wurfweite zu berechnen.

Mit unserem heutigen Wissen lautet die Formel für die Fallgeschwindigkeit (ν) im luftleeren Raum :

$\nu = \sqrt{2 \times g \times s}$; worin s die Fallhöhe in m und g die Fallbeschleunigung mit ~ 9,81 m/s² sind. Für Galilei galt $\nu = \sqrt{s}$. Beide Formeln unterscheiden sich nur durch den konstanten Faktor $2 \times g$.

Galilei hat selbst gesagt, dass er die Ursache der Beschleunigung nicht untersuchen wolle; für die Fallbeschleunigung hätte er jedoch einen Wert von ca. 5 m/s² erhalten, wie aus seinen Aufzeichnungen hervorgeht, die später ausgewertet wurden. *„Diese Abweichung kann vielleicht damit erklärt werden, dass Galilei den Zahlenwert der Beschleunigung, den wir heute als grundlegende Größe ansehen, keine Aufmerksamkeit geschenkt und dieses Problem als nebensächlich angesehen hat."* [6; S. 205].

Die Warnung Galileis, die Parabel nicht als allgemeingültige Form der Flugbahn

für Projektile mit großen Anfangsgeschwindigkeiten anzusehen, wurde nicht befolgt. Im Jahr 1683 erschien ein Buch von Francois Blondel mit dem Titel „L' Art de Jetter les Bombes", in dem der Autor viele der bisher bekannten Verfahren zur Flugbahnberechnung beschreibt und verwirft. Nur die Parabellösung von Galilei wird von ihm anerkannt und als einzig richtige dargestellt. Man kann vermuten, dass Blondel als ehemaliger Mathematiklehrer des Sonnenkönigs Ludwig XIV von dem Lösungsansatz Galileis begeistert war und sie deshalb propagierte – als späterer Marschall von Frankreich hätte er es aus der Praxis besser wissen können. (Anmerkung des Verfassers: In jener Zeit hatten Kanonenkugeln eine Anfangsgeschwindigkeit > 500 m/s; ihre Flugbahn wich merklich von der Parabel ab, wenn der Abgangswinkel > 5 Grad betrug).

Es stellt sich trotzdem die Frage, ob es Fälle gibt, in denen die Bahn im lufterfüllten Raum eine Parabel ist oder ihr sehr nahe kommt. Das trifft dann zu, wenn:

- eine kleine Anfangsgeschwindigkeit und eine große Querschnittsbelastung vorliegen, also z. B. bei den oben genannten Mörsergeschossen.
- eine große Anfangsgeschwindigkeit und ein sehr kleiner Abgangswinkel vorliegen. Das Geschoss fliegt in kurzer Zeit nur einen Bruchteil seiner Maximalweite; die Fallbeschleunigung kann sich daher kaum auswirken. Solche Bahnen werden als *Flachbahnen* bezeichnet.
- ein Teil der Bahn eine Höhe von mehr als 11 km erreicht. In diesem Bahnabschnitt ist die Flugbahn eine Parabel; solche Bahnen werden als *Fernbahnen* bezeichnet.
- Steine von Brücken auf fahrende Autos geworfen werden. In den Gerichtsgutachten wird dann von Galileis Parabelansatz mit der Sublimität ausgegangen, um dem Wurf nachzuvollziehen. In diesem Fall wird von einem *fahrzeugfesten Bezugssystem* ausgegangen.

Der Grundgedanke der Parabel veranlasste spätere Wissenschaftler eine der beiden anderen Möglichkeiten der Kegelschnitte, die Hyperbel daraufhin zu untersuchen, ob sie zur realistischen Darstellung von Flugbahnen geeignet sei. So wurde der Hyperbelansatz von I. Newton (1710) betrachtet, die Integration mit einem von ihm vorgegebenen Luftwiderstandsgesetz gelang ihm aber nicht. Als einer der Letzten war es E. Ökinghaus am Ende des 19. Jh., der erfolglos von der Hyperbel als der „wahren Flugbahn" ausging.[15;].

Es war in der Tat verlockend, einen Ansatz zu finden, mit dem die Flugbahn als geschlossener Ausdruck unter Einbeziehung eines allgemeingültigen Luftwiderstandsgesetzes berechnet werden konnte. Dieser Weg war nur im Fall der o.g. Flachbahnen gangbar, bei denen von einer konstanten Luftdichte und Temperatur und einem konstanten Wert für die Fallbeschleunigung ausgegangen werden kann. Einige Verfahren zur Berechnung von Flachbahnen basieren auf modifizierten Parabelansätzen.

Für Abgangswinkel > 5 Grad musste man und (muss es noch immer) die Flugbahn schrittweise berechnen. Dieser Vorgang dauerte auf mechanischen Tischrechenmaschinen in der ersten Hälfte des 20. Jh. bis zu 44 Stunden für eine Bahn, wenn eine hohe Genauigkeit verlangt wurde.

Das Ende dieser mühsamen Rechenarbeit und damit auch das Ende für alle analytischen oder graphischen Näherungsverfahren kam mit dem Einsatz des Computers. Damit wurden auch endgültig die modifizierten Parabelansätze obsolet.

Im Jahr 1945 war die ENIAC (Electronical Numerical Integrator and Computer) in den USA einsatzbereit. Auf dieser Maschine

wurden bis 1955 vorwiegend ballistische Berechnungen durchgeführt. Gegenüber den elektromechanischen Tischrechnern wurde die Zeit für die Bahnberechung um den Faktor 200 verkürzt.

Ab 1965 brachte die Firma IBM das Computersystem 7090 auf den Markt; auch auf diesen Rechnern wurden ballistische Aufgaben, nun wesentlich schneller als auf der ENIAC, gelöst.

Auf der 7090 lief auch das erste APL-System unter dem Namen IVSYS (Iverson System), das viel später zum APL2 wurde.

Literatur

1 Galilei in: Spektrum der Wissenschaft; Biographien 1/1998

2 Unterredungen und mathematische Demonstrationen über zwei neue Wissensbereiche, die Mechanik und die Fallgesetze betreffend. Von Galileo Galilei. Hrsg.: A. von Oettingen; Leipzig 1890 – 1904

3 L' Art De Jetter Les Bombes; F. Blondel ; Paris 1683

4 Ballistik; R. E. Kutterer ; Braunschweig ; 1959

5 Geschichte der Physik; W. Schreier u.a. ; Berlin

6 Kulturgeschichte der Physik; K. Simonyi; Frankfurt/M ; 2001

7 Geschichte der mechanischen Prinzipien; I. Szabo ; Basel 1996

8 Steinwurf von Brücken auf Fahrzeuge: Fragen an den Ballistiker. J. G. Hauck; unveröffentlichtes Manuskript

9 Galileo at Work ; St. Drake ; New York 1978

10 Galileo's Discovery of the Parabolic Trajectory; St. Drake u. J. MacLachlan. In: Scientific American 232; March 1975

11 Dresdener Beiträge zur Geschichte der Technikwissenschaften; Heft 7; Technische Universität Dresden 1983

12 Geschichte der Mathematik; H. Kaiser u. W. Nöbauer; München 2006

13 Modern Exterior Ballistics; R. L. McCoy; Atglen; USA; 1999

14 Äußere Ballistik; H. Molitz u. R. Strobel; Berlin 1963

15 Die Hyperbel als ballistische Kurve; E. Ökinghaus; in: Archiv für Artillerie und Ingenieur Offiziere (Nr. 10) 1893

16 Ein einfaches Verfahren zur angenäherten Darstellung von Flugbahnen; R. Schmidt; in: Wehrtechnische Monatshefte; April 1938

Kontakt zum Autor:
Albert Badior, Gutenbergstraße 6,
D-76448 Durmersheim

Impressum

APL-Journal
27. Jg. 2008, ISSN 1438-4531
Herausgeber: Rainer Nußbaum, APL-Germany e.V., Mannheim, http://www.apl-germany.de
Redaktion: Dipl.-Volksw. Martin Barghoorn (verantw.), TU Berlin, Franklinstr. 28, D-10587 Berlin, Tel. (030) 314 24392, Fax (030) 314 25901
Verlag: RHOMBOS-VERLAG, Berlin, Kurfürstenstr. 17, D-10785 Berlin, Tel. (030) 261 9461, Fax (030) 261 6300, eMail: verlag@rhombos.de, Internet: www.rhombos.de
Erscheinungsweise: halbjährlich
Erscheinungsort: Berlin
Gestaltung: RHOMBOS-VERLAG
Druck: dbusiness.de GmbH, Berlin
Copyright: APL Germany e.V. (für alle Beiträge, die als Erstveröffentlichung erscheinen)

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutzgesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen. Eine Haftung für die Richtigkeit der veröffentlichten Informationen kann trotz sorgfältiger Prüfung von Herausgeber und Verlag nicht übernommen werden.
Mit Namen gekennzeichnete Artikel geben nicht unbedingt die Meinung des Herausgebers oder der Redaktion wieder. Für unverlangte Einsendungen wird keine Haftung übernommen. Nachdruck ist nur mit Zustimmung des Herausgebers sowie mit Quellenangabe und Einsendung eines Beleges gestattet. Überarbeitungen eingesandter Manuskripte liegen im Ermessen der Redaktion.

Allgemeine Informationen

(Stand 2008)

APL-Germany e.V. ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache APL, sowie die Verbreitung des Verständnisses der Mensch-Maschine Kommunikation zu fördern. Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen, sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet APL-Germany den adäquaten organisatorischen Rahmen.

Auf Antrag, über den der Vorstand entscheidet, kann jede natürliche oder juristische Person Mitglied werden. Organe des Vereins sind die mindestens einmal jährlich stattfindende Mitgliederversammlung sowie der jeweils auf zwei Jahre gewählte Vorstand.

1. Vorstandsvorsitzender
Dr. Reiner Nussbaum
Dr. Nussbaum gift mbH, Buchenerstr. 78, 69259 Mannheim, Tel. (0621) 7152190.

2. Vorstandsvorsitzender:
Martin Barghoorn
Sekr. FR 6-9, Technische Universität Berlin, Franklinstr. 28, 10587 Berlin, Tel. (030) 314 24392

Schriftführer
Michael Baas
Dynamic Logistics Systems GmbH, Wilhelm-Schöffer-Str. 29, 63571 Gelnhausen, Tel. (06051) 13068, Fax (06051) 1642, eMail: m.baas@dls-planung.de

Schatzmeister
Uwe Janza
Janza Software Consulting, Bauerngasse 18, 90443 Nürnberg, Tel. 0911-327075, Fax 0911-327076

Beitragssätze:

Ordentliche Mitglieder:
Natürliche Personen 32,- EUR*
Studenten / Schüler 11,- EUR*

Außerordentliche Mitglieder:
Jurist./natürl. Pers. 500,- EUR*

* Jahresbeitrag

Bankverbindung:
BVB Volksbank eG Bad Vilbel
BLZ 518 613 25
Konto-Nr. 523 2694

Hinweis:
Wir bitten alle Mitglieder, uns Adressenänderungen und neue Bankverbindungen immer sofort mitzuteilen. Geben Sie bei Überweisungen den Namen und/oder die Mitgliedsnummer an.

Einzugsermächtigung

Mitglieds-Nr.: _____

Ich erkläre mich hiermit widerruflich damit einverstanden, daß APL Germany e.V. den jeweils gültigen Jahres-Mitgliedsbeitrag von meinem unten angegebenen Konto abbucht. Mit der Datenübermittlung an das oben genannte Kreditinstitut bin ich einverstanden. Einen eventuell bestehenden Dauerauftrag habe ich bei meiner Bank gelöscht.

Bankbezeichnung: _____

BLZ: _____ Konto-Nr.: _____

Datum: _____ Unterschrift: _____

APL 2010 LPA – Berlin
Array **P**rocessing **L**anguages
Learn **P**arallel **A**pplications

Call for Papers

Chairman's Address

Dear friends of APL and array processing languages,

Following the successful APL Berlin 2000 conference we would like to inform you that the global APL2010 conference is taking place again in Berlin from

September 13 to 16, 2010, in Berlin
at the Technical University (TU)

The members of APL Germany and GSE Working Group APL will jointly organise and host this event supported by the GUIDE SHARE Europe (GSE) organisation.

The vendors of APL and Array Processing Systems as well as array-oriented application software and their user groups will be invited to join the conference as sponsors, exhibitors, presenters and participants. Dyalog '10 is already scheduled for taking place integrated into APL2010.

The program committee headed by Dieter Kilsch and Axel Holzmueller is to take care about the APL2010 Conference program and its publications. An APL2010 Conference website is being created.

All APL and array oriented user groups are most heartily invited to join the APL2010 Conference as well as to support it by contributions to the program and to cooperate with the hosting organisations.

We would appreciate very much your taking into account to join APL2010 as well as to provide some contribution. So we would be delighted if you would continue to read this Call for Papers as a very hearty invitation.

We are looking forward to welcoming you in Berlin at APL2010 and honouring you as a speaker.

For the hosting organisations

Gerald Dittrich
Chairman of the APL2010 Organisation Committee

APL2010 Topics

The APL2010 Berlin Conference is seeking contributions which highlight the value and importance of array processing languages in finding solutions to (application) problems and in creating new strategies for programming. How do array processing languages facilitate rapid development, simulations, prototyping, and toolbox creation?

Computers are getting faster and faster, but a real speed-up can only be achieved by exploiting parallel architectures, spreading the workload on several processors in one computer or on several (distributed) computers, maybe worldwide. From their nature array processing languages are supporting the implementation of parallelism. What are their benefits in existing and future parallel HPC environments and what are today's and tomorrow's applications? Where are the limits of parallelism? To which extent (new and existing) array processing applications can automatically be parallelized?

Programming is a constantly evolving discipline. New and challenging applications keep requiring more realistic simulations, stochastic methods, massive data manipulation, and distributed high performance computing. What is the role array processing languages are playing in these areas? How does parallelism help?

Many programming languages and symbolic interpreters, including the different APL implementations, J, K, MatLab, Maple, S, R are becoming stronger array processing languages (in the following referred to as APLs). What features do these languages have in common? How do they differ? Are they converging in terms of primitives, parallelism, functionality, data and control flow, interfaces to hardware and software, and in communication inside one computer and to the outer world? Do these languages miss important data structures or primitives?

The success of a language depends not only on its quality, but also on its development environment - editors, profilers, and debuggers - and on its toolboxes and libraries. These permit people to attack a problem quickly and successfully.

Topics of interest include, but are not restricted to:

- State of the art of array processing languages:
  - Actual situation
  - Forthcoming features
  - Future expectations
- Array processing languages in computer sciences:
  - Parallel Programming
  - (Automatic) parallelization and concurrency
  - Object-oriented programming
  - Meta level programming
  - Interfaces and communication to external environments
  - Artificial intelligence
  - APLs and the World Wide Web

Sponsored by:

Gold Partner

Gold Partner

Silver Partner

- Challenging (production type) applications solved with array processing languages:
  - Finance and insurance
  - Marketing and ordering
  - R & D HPC calculations
  - Simulation of scientific, technical, economical and social phenomena
  - Control of processes and plants
  - Knowledge acquisition
- Toolboxes for all varieties of applications:
  - Data representation, graphics
  - APLs and (Relational) Data Base Systems
  - Graphical User Interfaces
  - Neural networks, fuzzy logic
  - Automatic parallelization
- Array processing languages in education:
  - Teaching APLs
  - Teaching with APLs
  - APLs in Schools
  - Student Projects
- Theory and practice in array processing language design and implementation:
  - Optimization of APL programs
  - Language design in APLs
  - APLs and irregular data structures
  - Shortcomings of today's APLs

Types of Contributions

Paper:
45 min presentation (+ 15 min discussion)

Short communication:
20 min presentation (+ 10 min discussion)

Tutorial:
90 min interactive session, several sessions may be booked. The main intention should be to get new people interested in APLs.

Workshop:
2 h computer based introduction into an APL language, a tool box or a new technique, several connected or separate sessions may be booked. The main intention should be to get new people interested in APLs.

Poster:
Presentation of achievements, ideas and topics to be discussed.

Birds-of-a-feather:
e.g. national user groups, special purpose groups.

Papers and short communications will be reviewed.

APL2010 Proceedings

Deadlines

March 15, 2010	Contribution commitment indicating the category of contribution	Author
April 15, 2010	1 page abstract	Author
May 17, 2010	Acceptance notification	Program Committee
June 15, 2010	Preliminary agenda	Program Committee
July 31, 2010	Final version	Author

There are no deadlines for Birds-of-a-feather sessions.

Intent of Submission

We would appreciate very much your contribution to the conference. Your intention to do so should be emailed as soon as you can to contribution@apl2010.de. Technical details for submission layouts will be sent out end of November 2009.

Dieter Kilsch
kilsch@fh-bingen.de

Axel Holzmueller
axel.holzmueller@dpc.de

Program Chair