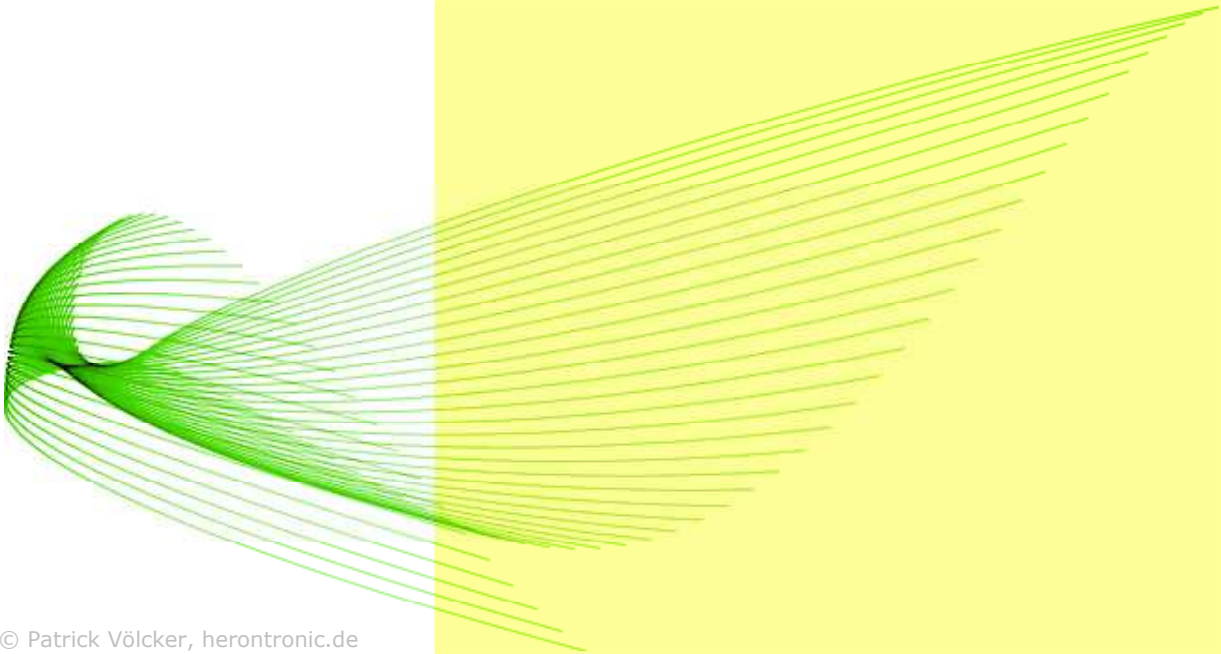
3D-Modelling mit DHTML

Dreidimensionalität im Web ist auch ganz ohne Flash oder andere Browser-Plugins möglich. Für einfache Darstellungen kann auch simples DHTML in geschickter Kombination mit wenigen Grafiken ausreichen.

Das folgende Tutorial soll zeigen, mit welchen Techniken und Tricks 3D-Rendering mit DHTML zu bewerkstelligen ist.



Problem:
Keine Grafikfunktionen

Um Dreidimensionalität darstellen zu können, müssen dem Programmierer prinzipiell grundlegende Grafikfunktionen zur Verfügung stehen.

Normalerweise funktioniert Grafik unter HTML nur mit Hilfe von fertigen Bildern, in denen die verschiedenen Ansichten bereits vorgerendert sind und die speicherintensiv geladen werden müssen. Bei flüssigen Animationsschritten werden dabei sämtliche möglichen Perspektiven vor- oder nachgeladen. Dadurch steigt die Datenmenge einer Webseite immens.

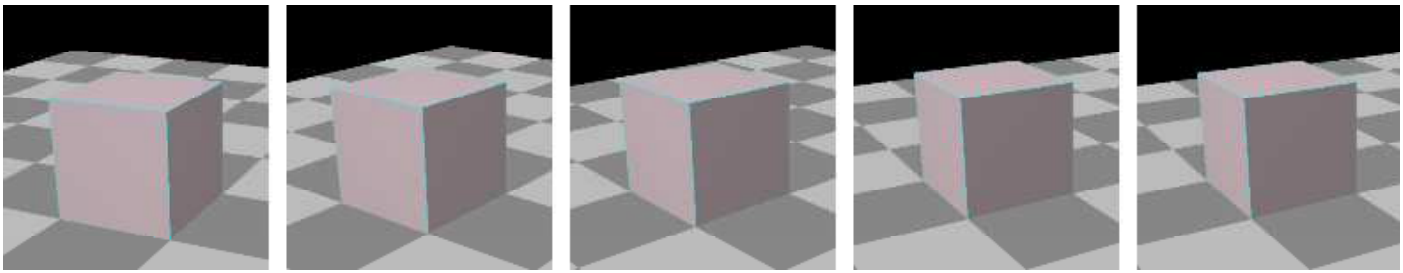


Abb.1: Vorgerenderte 3D-Ansicht eines Würfels

Ideal, weil weniger speicherintensiv und weniger ladezeitraubend, wäre eine eigene 3D-Grafik-Engine, die die verschiedenen Ansichten auf Anfrage in Echtzeit rendert. Dafür fehlt Javascript aber ein Befehlssatz mit einfachen Grafikfunktionen wie setPix() oder lineTo(), wie er von anderen Programmiersprachen her bekannt und für die Darstellung von 3D-Grafiken unbedingt erforderlich ist.

In diesem Tutorial erstellen wir uns nun für Javascript eine Grafikbibliothek, mit der wir einfache Funktionen wie Bildpunkte und Linien setzen können, um sie später im 3D-Raum zu nutzen.

Wichtig bei allen Beispielen ist dabei, dass sie auf allen Systemen laufen. Die Abwärtskompatibilität setze ich hier sehr tief bei Netscape Navigator 4.0 und Internet Explorer 4.0 an. Die Obergrenze ist mit Firefox, Safari und Opera natürlich nach oben offen und gottseidank durch die voranschreitende Anpassung der Webbrowser an die W3C-Standards kein Problem mehr.

Wir erstellen eine eigene Grafikbibliothek

Die wichtigste Grafikfunktion überhaupt ist das Setzen eines einzelnen Bildpunktes an eine beliebige Koordinate auf der Webseite. Ohne diesen Befehl geht in der Grafikprogrammierung gar nichts.

In HTML haben wir die Möglichkeit, einen Bildpunkt mit Hilfe eines Satzzeichens zu setzen. Hier bieten sich der normale Punkt `"."`, der Mittelpunkt `"·"` oder ganz einfach der Buchstabe `"O"` an, die wir in einen DIV-Container (oder Layer) setzen.

Im Folgenden nehmen wir dafür den Mittelpunkt, weil er einem Bildschirmpunkt in der Form am nächsten kommt.

Der erste Bildpunkt

```
<div id="punkt0" style="position:absolute; left:0px; top:0px; z-Index:0;">&middot;</div>
```

Wird die HTML-Seite aufgerufen, wird der Mittelpunkt (`&middot;`) an die linke obere Ecke der Website über die Parameter `left:0px; top:0px;` an die Koordinate (0|0) gesetzt. Der Wert stimmt nur ungefähr, weil der Mittelpunkt je nach Schriftgröße einen geringen Abstand zur „richtigen" Koordinate hat.

Um nun mehrere Punkte auf die Webseite zu setzen, brauchen wir also dementsprechend viele DIV-Container.

Abb.2: Die erste primitive Linie mit HTML

```
<div id="punkt0" style="position:absolute; left:0px; top:0px;z-Index:0;">&middot;</div>

<div id="punkt1" style="position:absolute; left:1px; top:0px;z-Index:0;">&middot;</div>

<div id="punkt2" style="position:absolute; left:2px; top:0px;z-Index:0;">&middot;</div>
```

Am Bildschirm wird nun durch die jeweilige Verschiebung der Punkte eine kleine waagerechte Linie mit der Länge 3 Pixel dargestellt. Die Darstellung ist sehr grob, aber für den Anfang reicht uns das.

Abb.3: Die verschobenen Punkte

Angenommen, wir wissen zu Beginn noch nicht, wo die einzelnen Punkte der Linie sitzen sollen, brauchen wir eine Javascript-Funktion, die uns die Punkte nach dem Ladevorgang nach Bedarf verschiebt.

Am besten geht das mit folgender Javascript-Funktion **setzePunkt**:

```
function setzePunkt(nr, x, y)
  {
  if(document.layers)
       {
       eval('document.layers["'+nr+'"].left='+x+';
             document.layers["'+nr+'"].top='+y);
       }
  else if(document.all)
       {
       eval('document.all.'+nr+'.style.left='+x+';
             document.all.'+nr+'.style.top='+y);
       }
  else if(document.getElementById)
       {
       eval("document.getElementById('"+nr+"').style.left="+x+";
             document.getElementById('"+nr+"').style.top="+y);
       }
  }
```

Auf die Funktionsweise möchte ich nicht weiter eingehen. Fortgeschrittene DHTML-Programmierer, für die dieses Tutorial gedacht ist, werden die Funktion auch so verstehen, Anfänger sollten sich einfach mit dem Wissen begnügen, dass die Funktion bei Aufruf einen Bildpunkte an die Koordinate (x|y) verschiebt.

Der Funktionsaufruf `setzePunkt("punkt0", 10, 10)` setzt also den Punkt mit der ID „punkt0" an die Position (10|10);

Mit den folgenden Befehlen kann nun ein angedachtes Dreieck wie in der Abbildung links dargestellt werden.

```
setzePunkt("punkt0",10,10);
setzePunkt("punkt1",10,50);
setzePunkt("punkt2",50,10);
```

Funktion linie()

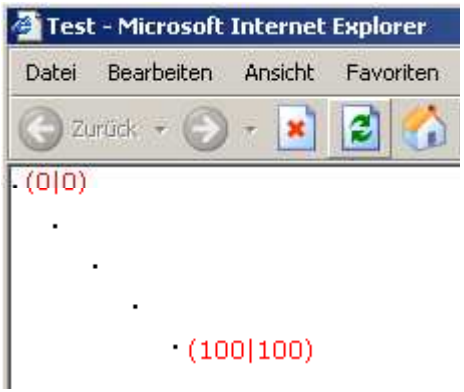


Abb.4: Die erste Linienfunktion

Um aus den Punkten nun eine Linie zu formen, brauchen wir nun eine Javascriptfunktion, die uns die Rechenarbeit abnimmt.

```
function linie(x0,y0,x1,y1)
  {
  for (p=0;p<5;p++)
       {
       var x = x0 + (x1-x0)/5 * p;
       var y = y0 + (y1-y0)/5 * p;
       setzePunkt("punkt"+p, x, y);
       }
  }
```

Der Befehl `linie(0,0,100,100)` verteilt fünf Punkte auf der Geraden zwischen den Punkten (0|0) und (100|100) gleichmäßig und deutet so eine Linie zwischen den beiden Endpunkten an.

Vor dem Ausführen der Funktion nicht vergessen, dass zwei zusätzliche DIV-Container in den Body mit den IDs „punkt3" und „punkt4" eingefügt werden müssen.

So ist zwar schon eine Linie erahnbar, von 3D-Modelling sind wir aber noch weit entfernt. Aber immerhin sind wir bis zu diesem Punkt noch ohne Fremdgrafik oder ein geladenes Bild ausgekommen und haben trotzdem schon eine grundlegende Grafikfunktion integriert. Diese lässt sich natürlich noch verbessern.

Setzen wir z. B. statt fünf DIV-Containern hunderte von ihnen für eine einzelne Linie ein, können wir immerhin schon eine durchgehende Linie zeichnen.
Das kostet nicht nur eine MengeTipparbeit für den Programmierer, sondern wir stellen auch fest, dass, wenn wir die linie-Funktion ein zweites Mal innerhalb der Webseite aufrufen, die erste Linie für die nachfolgende Linie verschwindet.

Das Problem liegt darin, dass wir pro Linie hunderte von Punkten einsetzen müssen. Mit jeder weiteren Linie müssen wir die Anzahl der Punkte um dieselbe Anzahl erweitern. Da bricht nicht nur der Programmierer beim Eintippen des Codes ein, sondern auch der Webbrowser, der im Animationsfall schon bei 80 Punkten zu ruckeln anfängt. Die Idee funktioniert also nur bei kleinen Objekten oder bei einer statischen Ausgabe ohne Animation.

Verbesserung der Darstellung

Bisher war das exakte Platzieren der Bildpunkte nicht möglich, weil der dargestellte Punkt einen ungenauen Offset in X- und Y-Richtung abhängig von der Schriftgröße hatte. Dieser ließe sich zwar über das exakte Definieren der Schriftproportionen korrigieren, aber sobald ein Benutzer eine bestimmte Schriftgröße innerhalb seines Webbrowsers festgelegt hat, stimmen die Berechnungen nicht mehr.

Eine Abhilfe schafft da eine kleinen, leeren Tabelle, die statt des Schriftzeichens eingefügt wird. `&middot;` ersetzen wir also nun mit:

```
<table cellspacing="0" cellpadding="1" border="0"
width="1"><tr><td bgcolor="#000000"></td></tr></table>
```

Der neue Punkt befindet sich nun an der richtigen Stelle.

Wer bei der Gestaltung der Webseite auf Benutzer des Netscape Navigators bis zur Version 4.x verzichten kann, liegt mit dieser Version richtig, auch wenn die Bildpunkte z. B. unter Firefox etwas breiter sind als sie z. B. im Internet Explorer dargestellt werden.

Um für alle Browser die ideale Bildpunktgröße von 1x1 Pixeln zu erreichen, müssen wir nun erstmals auf eine kleine Grafik zurückgreifen.

Gemeint ist das schon seit den ersten Tagen des Webdesigns für Tabellen gern genutzte transparente Gif zur pixelgenauen Gestaltung. Wir setzen den kleinen Helfer in der Größe 1x1 Pixel in die Tabelle und definieren den Zellenabstand cellpadding auf Null.

```
<table cellspacing="0" cellpadding="0" border="0"
width="1"><tr><td bgcolor="#000000"><img src="transparent.gif"
width="1" height="1" alt="" border=""></td></tr></table>
```

Nun können wir die Punkte endlich pixelgenau auf die Webseite einsetzen. Über Stylesheets ist es nun auch möglich, mit Javascript die Farbe der Bildpunkte zu bestimmen, wir müssen lediglich die Hintergrundfarbe der Tabellenzelle ändern.

Für Fortgeschrittene

Bis zu dieser Stelle wird für die meisten Webdesigner, die sich mit Javascript und DHTML beschäftigen, nicht wirklich etwas aufregend Neues dabei gewesen sein. Insgesamt ging es mir bisher auch nur um die prinzipiellen Ideen und Algorithmen, mit welchen sich auf einfache Weise Linien darstellen lassen.

Wenn man aber nun diese recht einfachen Ideen weiter denkt, stößt man tatsächlich auf Möglichkeiten, wie man „echte" Linien relativ einfach darstellen kann, ohne auf eine große Anzahl an Grafiken oder Containern zurückgreifen zu müssen.

Diese Möglichkeiten möchte ich nun auf den folgenden Seiten vorstellen:

Sonderweg:
echte Linien ab IE 6.0

Bekanntlich hat Microsoft immer seine eigene spezielle Art und Weise, um seine proprietären Standards durchzusetzen. Meist kam dabei viel Schrott heraus („JScript, Marquee, blink, etc.), manchmal waren aber auch interessante Ideen für Webdesigner dabei (wie z. B. Transformations, Alphablending), bei denen es meiner Meinung nach schade ist, dass die anderen Browserhersteller oder das W3-Konsortium nicht auf den Zug mit aufgesprungen sind, weil man sich so manche umständliche Programmierung hätte sparen können.

Mit einer Kombination aus zwei dieser proprietären Special Features ist es im Internet Explorer ab der Version 6.0 über ein paar Stylesheet-Tricks möglich, beliebige Linien darzustellen.

Der formale Aufbau des HTML-Dokuments ist zwar hinsichtlich bestehender Standards nahezu eine Katastrophe () und die Chance, dass diese Version auch in zukünftigen Browsergenerationen laufen wird, ist eher gering, nichtsdestotrotz möchte ich in diesem Tutorial diese Möglichkeit des 3D-Modellings über reines DHTML nicht unerwähnt lassen.

Der Quellcode

Folgende HTML-Seite macht nun nichts weiteres, als eine kleine, schwarze Linie auf die Webseite zu setzen, zumindest dann, wenn sie im Internet Explorer ab der Version 6.0 geöffnet wird. Das erschreckende daran: Es funktioniert!

```
<html>
<head></head>
<body>
<style>
          p\:* {behavior: url(#default#VML)
          }
</style>

<script>

  function line(x0,y0,x1,y1)
       {
       if(document.all&&window.print)
             {
             document.body.insertAdjacentHTML(
             "afterBegin",
             "<p:line id=linie0 style='position:absolute;
               left:0px;top:0px;' strokeweight='1pt'></p:line>");

             linie0.to=x0+","+y0;
             linie0.from=x1+","+y1;
             linie0.strokecolor="#000000";
             }
       }

line(0,0,50,50);

</script>
</body>
</html>
```

Auch hier kann über die ID „linie0“ nur eine einzelne Linie dargestellt werden. Fortgeschrittene wissen aber nach einem kurzen Analysieren des Quellcodes sicherlich, wie man die Funktion für die Anwendung mehrerer Linien anpassen kann.

Doch kommen wir nun endlich zu der Lösung, wie man in allen gängigen Webbrowsern beliebige Linien zeichnen kann.

Wir malen uns drei Linien

Zuerst einmal legen wir hierfür den Webeditor weg und starten unser Grafikprogramm.

(...wird fortgesetzt)