1. Inhaltsverzeichnis

2. Einleitung

Meine Zielsetzung ist es, den Strahlenverlauf des Lichtes durch z.B. Platten, Linsen oder Linsensysteme mit Java-Applets möglichst exakt zu berechnen. Der Vorteil von Java-Applets liegt v.a. darin, dass man sie in einem Browser öffnen kann und nicht erst ein Programm downloaden muss, weil sie direkt in den html-code einer website implementiert sind, was heutzutage in den Zeiten des Internets von großer Bedeutung ist. Als Software zur Erstellung der Applets habe ich Dynageo Euklid von Roland Mechling (www.dynageo.de) und geometria von Dr. Timo Ehmke (www.geometria.de) verwendet. Herr Ehmke hat eine Anleitung seiner Arbeit geschrieben, die Sie sich von www.geometria.de downloaden können. Zum Betrachten der Applets selbst muss man die html-Dateien auf der CD mit einem javafähigen Browser (am besten Microsoft Internet Explorer 5.5 mit möglichst aktuellem Java-Plugin) öffnen, den Code des Applets kann man lesen oder verändern, indem man die script-, style- und js-Dateien mit dem Notepad öffnet. Sollte es techn. Probleme geben hilft manchmal ein Klick auf „update" weiter. Sie können mich aber auch jederzeit unter Tel.: 09721/82727 oder Mail: marcel-sl@gmx.de erreichen.

So viel zum Technischen, jetzt zur Physik: Meine erste Zielsetzung war es den Stahlenverlauf durch Linsen bzw. Linsensysteme, wie z.B. Fernrohr, für den Benutzer möglichst einfach, aber ebenso möglichst exakt zu berechnen. Als ich erkannte, dass bei der Berechnung des Strahlenverlaufs durch dünne Linsen eigentlich nur Näherungen verwendet werden (wenn die Linse dicker wird, wird das Ergebnis ungenau/falsch) habe ich nach einer allgemeinen Möglichkeit, die auch für dicke Linsen exakt ist, gesucht und bin auf die Gaußschen Hauptebenen gestoßen. Weil es damit jedoch sehr kompliziert ist nicht nur bikonvexe Linsen zu berechnen und das Einbringen einer 2. Linse nur sehr schwer zu überblicken ist, habe ich mich entschlossen das Snelliusche Brechungsgesetz zu verwenden. Auf dem Stand in Knetzgau werde ich die Ergebnisse der Applets dann mit realen Experimenten beweisen.
Die einzelnen Applets werde ich unter „Vorgehensweise" genauer beschreiben:

3. Vorgehensweise

Dieses Kapitel unterteile ich in mehrere Abschnitte, wobei der Inhalt immer komplexer wird und tiefer in die geometrische Optik hineingeht. In einem Abschnitt wird jeweils ein Applet behandelt.

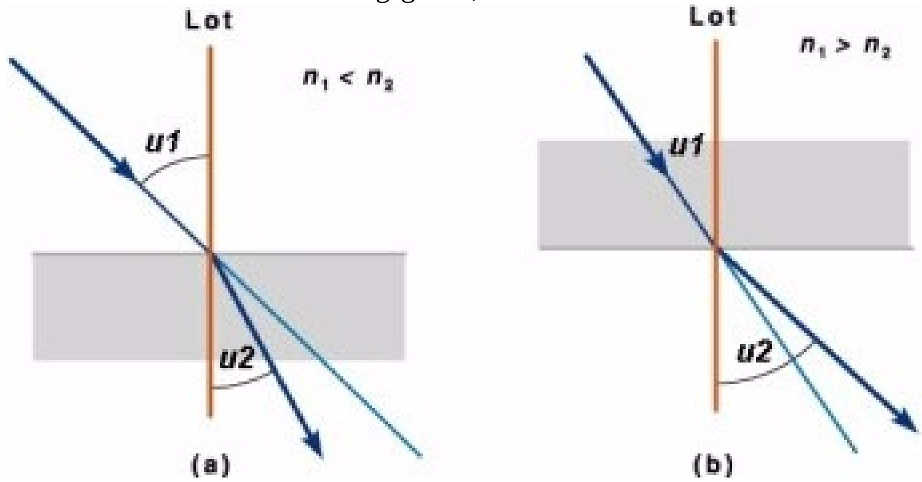
3.1. Die camera obscura in 3D

In diesem Abschnitt möchte ich kurz zeigen wie die camera obscura (auch: Lochkamera) funktioniert. Im Gegensatz zu allen anderen Bestandteilen dieser Arbeit handelt es sich hier aber nicht um ein Java-Applet, sondern um eine DreiDGeo-Datei, weil man mit DreiDGeo wesentlich einfacher dreidimensionale Figuren erzeugen kann. Die _camera-obsura.ddd kann mit den meisten 3D-Programmen betrachtet werden. Auf die Funktionalität der Lochkamera gehe ich hier nicht ein, weil man das in jedem Physikbuch der 8. Klasse nachlesen kann; nur hat man dann nicht eine solche interaktiv drehbare Figur..

3.2. Brechung des Lichtes und Totalreflexion

In diesem Abschnitt möchte ich zeigen wie das Licht beim Übergang von einem Medium in ein anderes gebrochen wird und wie es zur Totalreflexion kommt. Dabei habe ich v.a. „Bergmann/Schäfer: Optik“ (siehe Literaturverweis) Seite 28ff verwendet.

Der Effekt der Brechung des Lichtes ist uns allen bekannt: Wenn man in ein mit Wasser gefülltes Glas einen Löffel legt und dann diesen Löffel von der Seite betrachtet scheint der Löffel beim Übergang von Luft zu Glas einen „Knick“ zu haben. Dieser Knick lässt sich mit dem Brechungsgesetz, das 1620 von W. Snellius entdeckt wurde, berechnen:



(Abb. 1)

$$\sin u1 \cdot n1 = \sin u2 \cdot n2 \qquad (F1)$$

wobei u1 der Einfallswinkel und u2 der Austrittswinkel ist (siehe Bild und _ap2.html auf CD), n1 und n2 sind die Brechzahlen der beiden Medien (z.B. für Luft 1,0003; für Wasser 1,3330; siehe _ap2-help.html). Diese Brechzahl ergibt sich aus der Fortpflanzungsgeschwindigkeit des Lichtes im jeweiligen Medium. Das Brechungsgesetz steht in der tore01a1.script unter e[21], wobei der Einfallswinkel und die beiden Brechzahlen wegen der Schieberegler bereits bekannt sind, was bedeutet, dass nur noch u2 zu berechnen ist:

$$u2 = \arcsin\left(\frac{\sin u1 \cdot n1}{n2}\right) \qquad (F1b)$$

In dem Applet in der _ap2.html ist vorausgesetzt, dass das Licht von oben kommt. Die rote waagerechte Gerade ist die Grenze zwischen den beiden Medien. Den Einfallsstrahl kann man mit den Punkten T und Tob verändern, die Brechzahlen der Medien lassen sich über die Schieberegler unten regeln.

Es lässt sich mit diesem Applet auch leicht nachvollziehen, warum im Sommer die Luft über der Straße „flimmert“; das liegt daran, dass die Brechzahl der Luft von der Temperatur der Luft abhängt. Weil sich die Luft über einer schwarzen Straße sehr leicht erhitzt ist sie wesentlich wärmer wie die darüberliegende Luft, d.h. das Licht von der Sonne wird in Richtung des Beobachter gebrochen. Kommt der Beobachter zu nahe an diesen Punkt, so verschwindet das Flimmern auf Grund der Totalreflexion (siehe unten). Analog dazu lässt sich eine Fatamorgana erklären.

Der Lichtstrahl wird aber nicht nur gebrochen, sondern auch reflektiert, was an dem hellgrauen Lichtstrahl im Applet gezeigt wird. Es gibt hier allerdings einen Sonderfall: die Totalrefelxion. Dieser Sonderfall tritt ein wenn die Brechzahl des Mediums aus dem der Lichtstrahl kommt größer wie die des anderen Mediums ist (in dem Applet also n1 > n2) und der Einfallswinkel u1 „zu groß" wird. Zu groß heißt hier, dass u1 größer wie der sog. Grenzwinkel zweier Medien, der in dem Applet unter „Grenzwinkel = .." angezeigt wird, ist. Der Grenzwinkel Gre lässt sich wie folgt berechnen:

$$Gre = \arcsin \frac{n2}{n1} \qquad (F2)$$

Die Totalreflexion spielt insbesondere bei Prismen und in der Faseroptik eine große Rolle.
Sollte das Applet aus irgendeinem Grund etwas falsches zeigen, kann ein Klick auf „update" helfen, weil dann die Figur neu gezeichnet und überprüft wird.

An diesem Beispiel möchte ich kurz den Aufbau und die Funtionsweise eines solchen Applets zeigen und etwas erläutern, weil es - im Vergleich zu den noch folgenden - relativ leicht zu verstehen und zu überschauen ist:

Programmlisting:

Inhalt der tore01a1.script:
die einzelnen Konstruktionsschritte werden mit e[..] durchnummeriert.

```
e[1] = O;   point;      fixed;      0.0, 0.0;   0;192,192,192;192,192,192;default;
```

„`= O`" ist der Name des in dieser Zeile erzeugten Objekts
„`point; fixed`" bedeutet fester, nicht ziehbarer Punkt - die anderen weiter unten aufgeführten Objektbezeichner kann man in der konstruktionsreferenz.pdf, die es unter www.geometria.de zum Download gibt, nachlesen.
„`0.0, 0.0;`" sind die Koordinaten des Punktes
„`0;192,192,192;192,192,192;default;`" legt die Farbe des Punktes fest.

```
e[2] = _xe; point;      fixed;      1.0, 0.0;   "hidden"
e[3] = _ye; point;      fixed;      0.0, 1.0;   "hidden"
e[4] = _KO; point;      coordSystem;      O, O, _xe, O, _ye, 1000, 1000, 1000, 1000;
      0;192,192,192;192,192,192;default;
e[5] = xa;  line; straightLine;     O, _xe;0;0;192,192,192;0;
e[6] = ya;  line; straightLine;     O, _ye;0;0;192,192,192;0;
e[7] = P1;  point;      dragable;   4.28625, 5.05354; "hideLabel"
e[8] = g1;  line; parallel;   P1, xa;     0;red;red;red
```

// das ist die Grenze zwischen den beiden Materialien

```
e[9] = T;   point;      dragable;   10.37167, 5.05354, g1;
e[10] = g2; line; perpendicular;    T, g1;      0;gray;gray;gray
e[11] = Tob;      point;      dragable;   6.93208, 7.72583;
e[12] = h1; line; ray;  T, Tob;     0;yellow;yellow;yellow
```

// das ist der obere Lichtstrahl („Strahl von T durch Tob")

```
e[13] = _P1_k1;   point;      functionDepend;   "coordinateX(T) + 5",
"coordinateY(T)"; "hidden"
e[14] = k1; circle;     radius;     T, _P1_k1;  "hidden"
e[15] = noob;     point;      intersection;     g2, k1, 1;  "hidden"
e[16] = noun;     point;      intersection;     g2, k1, 2;  "hidden"
e[17] = w(noob, T, Tob);      sector;     angle;      noob, T, Tob, 5, 5;     "hidden"
e[18] = u1; measure;    calculate;  "angle(noob,T,Tob)", 468, 27, "u1 = ", "";
      "hideLabel"
```

// hier wird der Winkel u1, also w(noob,T,Tob) berechnet

```
e[19] = n1; measure;    controller; 1.000, 0.0002, 1.0000, 2.5000, 250, "n1 = ", "
(oben)";    0;128,128,128;128,128,128;default;
```
// das ist der Schieberegler für n1
```
e[20] = n2; measure;    controller; 1.500, 0.0002, 1.0000, 2.5000, 250, "n2 = ", "
(unten)";   0;128,128,128;128,128,128;default;
```
// das der Schieberegler für n2
```
e[21] = u2me;     measure;    calculate;
      "ASIN(SIN(calculate(u1)*0.01745329252)*calculate(n1)/calculate(n2))", 515, 68,
"u2 = ", "";       "hideLabel"
```
// hier wird nach *(F1b)* der Winkel u2 berechnet
```
e[22] = u2; measure;    calculate;  "calculate(u2me)";
e[23] = z12b;     point;      free; 1.0, 400.0; "hidden"
e[24] = z12c;     point;      free; 1.0, 400.0; "hidden"
e[25] = z12d;     point;      free; 1.0, 400.0; "hidden"
e[26] = z18;      point;      free; 1.0, 400.0; "hidden"
e[27] = z19;      point;      free; 1.0, 400.0; "hidden"
e[28] = z20;      point;      free; 1.0, 400.0; "hidden"
e[29] = z21;      point;      free; 1.0, 400.0; "hidden"
e[30] = z23a;     point;      free; 1.0, 400.0; "hidden"
e[31] = z23b;     point;      free; 1.0, 400.0; "hidden"
e[32] = z24;      point;      free; 1.0, 400.0; "hidden"
```
// diese Punkte sind für das Applet überflüssig und durch „hidden“ unsichtbar. Sie sind eine Notlösung aus einem progarmmiertechn. Problem.
```
e[33] = _t_g3;    measure;    calculate;  "-1*calculate(u2)";     "hidden"
```
// für die Leerzeichen müssten eigtl. ‘_’ stehen - ist aber in word mit dieser Schrift nicht möglich.
```
e[34] = _e_g3;    measure;    calculate;  "1";  "hidden"
e[35] = _P1_g3;   point;      rotation;   noun, T, _t_g3, _e_g3;  "hidden"
e[36] = g3; line; straightLine;     T, _P1_g3;  "hidden"
```
// hier wird der Winkel u2 konstruiert
```
e[37] = Tun;      point;      intersection;     g3, k1, 2;  "hidden"
e[38] = Tunxx;    point;      intersection;     g3, k1, 1;  "hidden"
e[39] = h2; line; ray;  T, Tun;     0;yellow;yellow;yellow
```
// das ist der untere, gebrochene Lichtstrahl
```
e[40] = _t_g4;    measure;    calculate;  "-0.01745329252*(360-calculate(u1))";
      "hidden"
e[41] = _e_g4;    measure;    calculate;  "1";  "hidden"
e[42] = _P1_g4;   point;      rotation;   noob, T, _t_g4, _e_g4;  "hidden"
```
// hier wird der Winkel für die Reflexion konstruiert
```
e[43] = g4; line; straightLine;     T, _P1_g4;  "hidden"
e[44] = P2; point;      dragable;   13.51490, 7.49560, g4;  "hidden"
e[45] = h3; line; ray;  T, P2;      0;0;192,192,192;0;
```
// das ist der reflektierte, graue Strahl
```
e[46] = gre1;     measure;    JSfunction; "asin_grenz", "u1", "u2", "n1", "n2";
e[47] = gre2;     measure;    calculate;  "calculate(gre1)*57.29577951";
e[48] = gre3;     measure;    JSfunction; "fall_grenz", "gre1", "gre2", "u1", "n1",
"n2";
```
// Versuch der Fallunterscheidung über JavaScript (siehe in der js-Datei), wird aber weiter unten nicht mehr verwendet, weil es über diesen Quelltext selbst besser funktioniert:
```
e[49] = n1zun2;   measure;    calculate;  "if (calculate(n1) > calculate(n2)) then (1)
else (0)"
```
// Fallunterscheidung: „Wenn n1 größer als n2 ist, dann 1, sonst 0“
```
e[50] = grf1;     measure;    calculate;  "if (calculate(u1) > 90) then (1) else (0)"
```
// Fallunterscheidung: „Wenn u1 größer als 90 ist, dann 1, sonst 0“
```
e[51] = grf2;     measure;    calculate;  "if (calculate(u1) < 270) then (1) else (0)"
```

// Fallunterscheidung: „Wenn u1 kleiner als 270 ist, dann 1, sonst 0“

```
e[52] = grf3;     measure;    calculate;  "if (calculate(grf1) & calculate(grf2)) then
(1) else (0)"
```

// Fallunterscheidung: „Wenn grf1 und grf2 1 sind, dann 1, sonst 0“

```
e[53] = grf4;     measure;    calculate;  "if (calculate(u1) > calculate(gre2)) then
(1) else (0)"
```

// Fallunterscheidung: „Wenn u1 größer als gre2 ist, dann 1, sonst 0“

```
e[54] = grf5;     measure;    calculate;  "if (calculate(u1) < 360-calculate(gre2))
then (1) else (0)"
e[55] = grf6;     measure;    calculate;  "if (calculate(grf4) & calculate(grf5)) then
(1) else (0)"
e[56] = grf7;     measure;    calculate;  "if (calculate(grf6) & calculate(n1zun2))
then (1) else (0)"
e[57] = grf8;     measure;    calculate;  "if (calculate(grf7) | calculate(grf3)) then
(1) else (0)"
```

// Fallunterscheidung: „Wenn grf7 oder grf3 1 ist, dann 1, sonst 0“

```
e[58] = grf8b;    measure;    calculate;  "if (calculate(grf7) | calculate(grf3)) then
(0) else (1)"
e[59] = grf9;     measure;    calculate;  "if (calculate(grf8) | calculate(grf3)) then
(1) else (0)"
e[60] = grf10;    measure;    calculate;  "if (calculate(grf8b) | calculate(grf3))
then (1) else (0)"
e[62] = h3yel;    line; ray;  T, P2;      0;yellow;yellow;yellow
```

// das ist der reflektierte Strahl in gelb

```
  //
hidden[1] = "if (calculate(grf3)) hide (h1)"
```

// „Wenn grf3 1 ist, dann verstecke h1“ (h1, der obere Lichtstrahl, ist dann also unsichtbar, weil der Punkt Tob ins untere Medium gezogen wurde)

```
hidden[2] = "if (calculate(grf8) = 1) hide (h2)"
```

// Wenn die Totalreflexion eintritt oder Tob zu weit unten ist wird der gebrochene Lichtstrahl h2 versteckt.

```
hidden[3] = "if (calculate(grf9) = 1) hide (h3)"
hidden[4] = "if (calculate(grf10) = 1) hide (h3yel)"
```

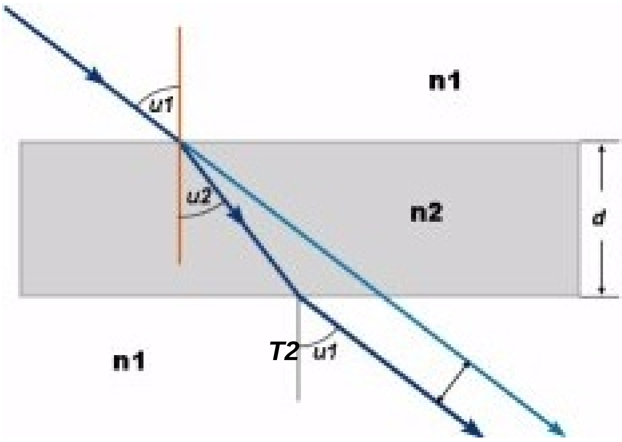
// h3 ist grau und h3yel ist gelb; beide sind die reflektierenden Strahlen (liegen eigt. Aufeinander, werden aber
// machmal versteckt, manchmal nicht):
// Wenn Totalreflexion eintritt oder Tob zu weit unten ist, wird h3 versteckt.
// Wenn Totalreflexion nicht eintritt oder Tob zu weit unten ist, wird h3yel versteckt.

Mit `<textbox> ... </textbox>` werden die Textboxen, in denen die Messwerte ausgegeben werden, beschrieben.

Das ist das Applet mit dem leichtesten/kürzesten Quellode aus meiner Arbeit. Die Applets die ich am noch erklären werde gehen teilweise bis über e[200] und nicht *nur* bis e[60]. Außerdem sind sie noch wesentlich komplizierter und komplexer aufgebaut (Sie können sich den Quelltext ja mal durchlesen, indem Sie mit dem Notepad die *.script-Dateien öffnen..).
Jetzt aber wieder zurück zur Physik:

3.3. Brechung an einer planparallelen Platte

In diesem Abschnitt möchte ich zeigen wie das Licht an einer planparallelen Platte gebrochen wird:
Wie man in Abb.2 sehen kann sind u1, n1 und n2 bekannt, was bedeutet, dass ich nur noch u2 nach *(F1b)* berechnen muss, um die Brechung an der ersten Fläche zu kennen. Die Brechung an der zweiten Fläche erhält man indem man durch den Punkt T2 die Parallele zum ursprünglichen Strahl einzeichnet. Genau das habe ich im Applet in _ap3.html gemacht, nur dass die gesamte Figur um 90° gedreht ist.

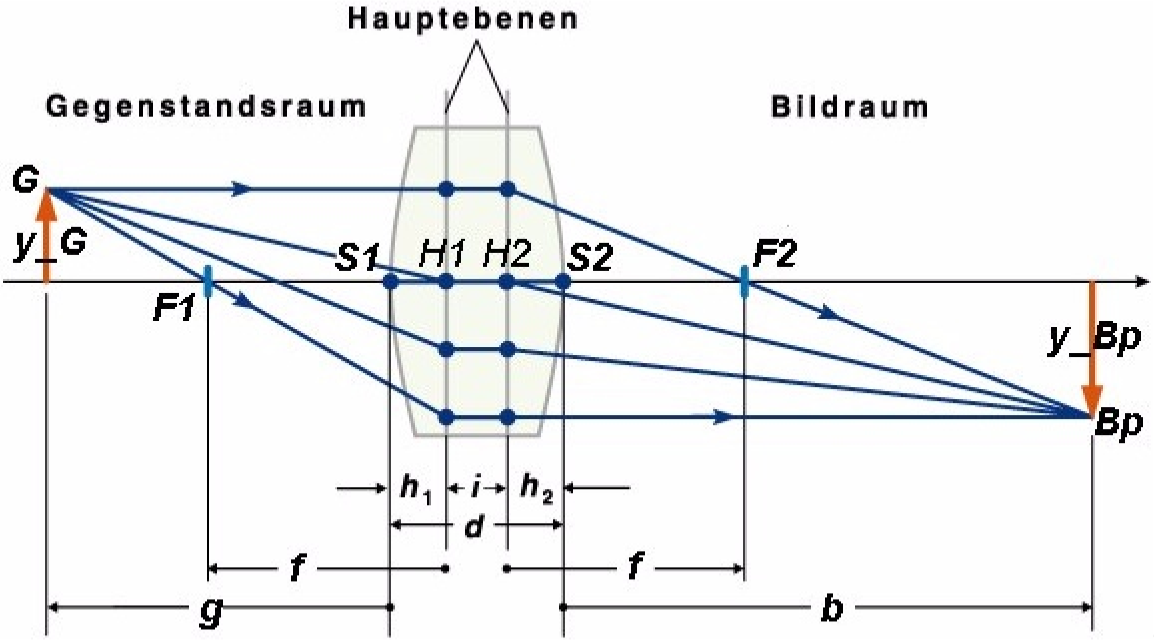


(Abb. 2)

In diesem Applet werden, wie in allen folgenden die Reflexion und die Totalreflexion nicht mehr berechnet, weil sie für meine Zwecke bedeutungslos sind. Bei dem Applet ist noch folgendes zu beachten: Tli muss immer links von Tre liegen und Tob muss immer links von der Platte liegen, weil sonst die Winkel für das Brechungsgesetz falsch angezeigt werden.

3.4. Brechung an einer dicken Linse (mit Gaußschen Hauptebenen)

Um den Strahlenverlauf durch dicke Linsen zu berechnen bin ich zuerst auf die Gaußschen Hauptebenen gestoßen, und obwohl mir einige wegen der Komplexität der Hauptebenen davon abrieten, habe ich versucht die Hauptebenen in einer dicken Linse zu berechnen. Dieses Applet ist das erste das ich geschrieben habe.
Hier die Bildkonstruktion:



(Abb. 3)

das dazugehörige Applet ist in der **_ap4.html** eingebettet.
Zum Aufbau des Applets:

Durch Schieberegler sind gegeben: Brechzahl des Linsenmaterials (Linse befindet sich in Luft)
Radien der Linse r1 und r2 (r1 links und r2 rechts)
Dicke der Linse d (über d_vonMpMp)

S1 und S2 sind die Schnittpunkte der Kugelflächen mit der optischen Achse (auch: Scheitelpunkte).
Durch Ziehen ist außerdem der Punkt G und somit g und y_G gegeben.

Nun muss ich alle anderen Werte berechnen:
Die Brennweite f wird wie folgt berechnet (Linse sei in Luft):

$$\frac{1}{f} = (n-1)\left[\frac{1}{r1} - \frac{1}{r2} + \frac{(n-1)\cdot d}{n\cdot r1\cdot r2}\right] \qquad (F3)$$

Im Grenzfall d → 0 geht diese Gleichung in die Linsenschleiferformel für dünne Linsen über, weil die beiden Hauptebenen dann aufeinander liegen:

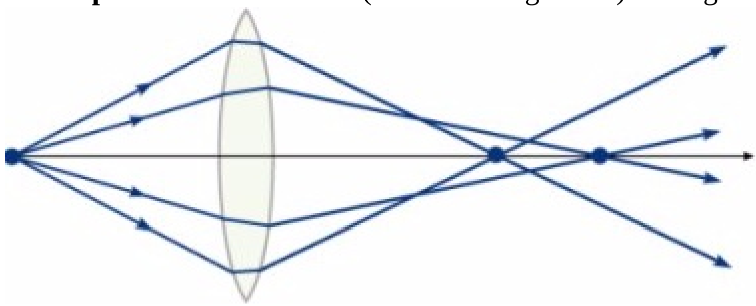
$$\frac{1}{f} = D = (n-1)\left(\frac{1}{r1} - \frac{1}{r2}\right) \qquad (F4)$$

Für die Abstände h1 und h2 der Hauptebenen von ihren jeweiligen Scheitelpunkten gilt:

$$h1 = -\frac{f\cdot(n-1)\cdot d}{r2\cdot n} \qquad (F5a)$$

$$h2 = -\frac{f\cdot(n-1)\cdot d}{r1\cdot n} \qquad (F5b)$$

wobei die Hauptebene rechts von ihrem Scheitelpunkt liegt wenn h1 bzw. h2 positiv ist. Somit sind die Punkte F1, F2, H1 und H2 exakt berechnet. Um nun den Strahlengang zu zeigen werden diese Punkte wie in Abb.3 gezeigt miteinander verbunden, womit auch Bp berechnet und y_Bp berechnet sind.
Jetzt kann ich also den Strahlengang durch dicke Linsen berechnen. Allerdings werden Linsenfehler nicht mit eingerechnet, was v.a. durch die **sphärische Aberration** (auch: Öffnungsfehler) zu Ungenauigkeiten führen kann:



(Abb. 4)

Wenn Strahlen parallel zur optischen Achse, aber nicht mehr paraxial (= achsennah) auf eine Linse fallen, werden sie nicht mehr im idealen Brennpunkt gesammelt. Das heißt, dass der Brennpunkt nicht ein durch die obigen Formeln festgelegter Punkt ist, sondern eigentlich eine „Fläche“, deren Mittelpunkt mit F bezeichnet wird. D.h. dass auch der Bildpunkt, der ja direkt von der Brennweite abhängt, vom Abstand zwischen G und der optischen Achse abhängig ist. Die Abb.4 zeigt diesen Linsenfehler stark übertrieben. In dem Applet in _ap4.html wird das - im Gegensatz zu den Applets mit dem Brechungsgesetz - nicht beachtet, weil es dafür, dass die Abweichung bei achsennahen Strahlen nur sehr klein ist zu aufwendig wäre diese Wölbung mit einzubeziehen.
Alle anderen Linsenfehler sind für dieses Applet von keiner bzw. nur geringer Bedeutung.
Es sei noch gesagt, dass dieser Linsenfehler zwar an einer sphärischen, aber nicht an einer parabolischen Linse, welche wohl im Allgemeinen gebräuchlicher sind, vorkommt, weil solche Linsen genau dafür konzipiert sind möglichst ohne Linsenfehler zu arbeiten.

Hinweise zum Arbeiten mit dem Applet:

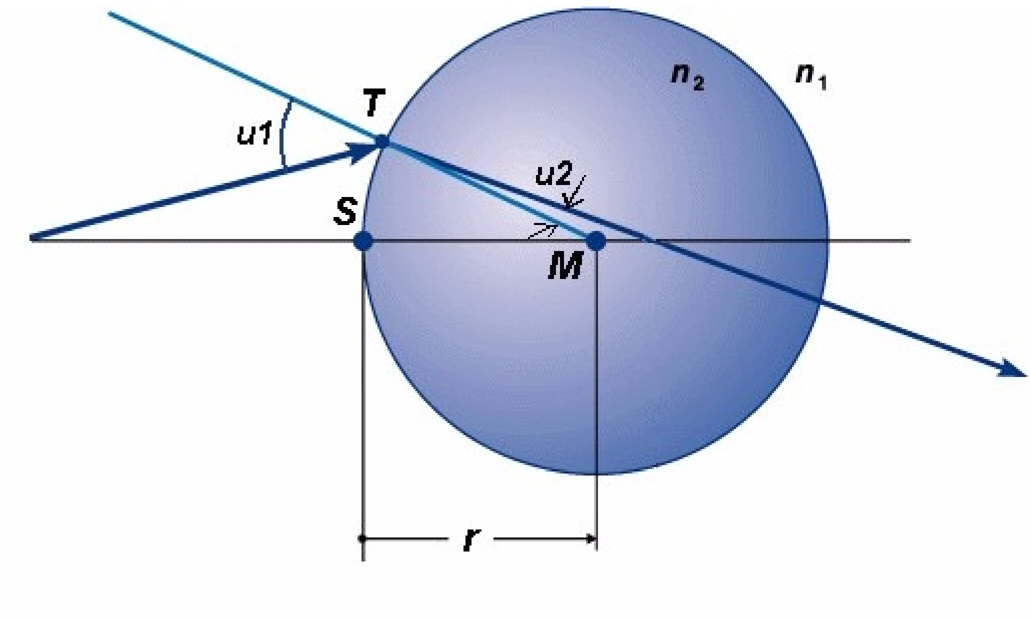
– Wenn man die Brennweite über die <u>Schieberegler verändert</u>, so ist zu beachten, dass der Punkt G immer links der Brennweite liegen muss; wenn die Gerade durch F1 (durch Verändern der Schieberegler) den Punkt G „einholt“ kann das zu Problemen in der Strahlenkonstruktion führen.
– Wenn man die Radien der Linse über die <u>Schieberegler r1 (links) und r2 (rechts) verändert</u>, so muss die Linse trotzdem „real bleiben“. Weil ich ganz zu Beginn meiner gesamten Arbeit einen Fehler bei der Konstruktion der Linsenflächen gemacht habe muss man die Dicke der Linse über den Schieberegler „<u>d_vonMpMp</u>“ einstellen. Es wird dadurch der Abstand zwischen dem Mittelpunkt vom Kreis für r2 und dem Mittelpunkt des Kreises für r1 verändert, was bedeutet, dass die Linsenflächen nicht fest an einem Punkt verankert sind, sondern dass sie sich bei Verändern der Radien die Linsenflächen von einander entfernen bzw. nähern. Deshalb muss man beim Verändern der Radien immer auch d_vonMpMp so verändern, dass sich die **<u>Linsenflächen noch schneiden</u>**. Am Anfang erscheint das zwar umständlich, aber mit ein wenig Übung ist das kein Problem. (Es waren wirklich die ersten 20

Konstruktionsschritte meiner gesamten Arbeit; eine Behebung dieses Fehlers ist jetzt im Nachhinein leider kaum noch möglich..).

Es können nur bikonvexe Linsen dargestellt und berechnet werden (rein physikalisch kann man mit den Hauptebenen zwar auch konkave Linsen berechnen - allerdings ist das 1. über das Brechungsgesetz einfacher und 2. relativ schwer zu programmieren).

3.5. Brechung an einer sphärischen Fläche (Brechungsgesetz)

Wegen der angesprochenen Probleme mit den Hauptebenen habe ich mich entschlossen, den Strahlenverlauf durch eine Linse auch über das schon bekannte Brechungsgesetz zu zeigen:



(Abb. 5)

wobei wie in Abschnitt 3.2 u1, n1 und n2 bekannt sind und u2 wieder wie folgt berechnet wird:

$$u2 = \arcsin\left(\frac{\sin u1 \cdot n1}{n2}\right) \qquad (F1b)$$

das dazugehörige Applet ist in der _ap5.html eingebettet. Diese Lösung hat gegenüber der mit den Hauptebenen mehrere Vorteile:

- Es können alle Linsenarten (also auch konkave) berechnet werden.
- Es kann der Übergang von jedem Material in jedes Material berechnet werden (muss außen also nicht zwangsläufig Luft sein wie in der Hauptebenen-Lösung).
- Linsenfehler werden mit beachtet, das Ergebnis ist also exakter.
- Die gesamte Konstruktion ist einfacher und es ist somit eher möglich die Brechung an mehr als 2 Kugelflächen zu zeigen, indem man den Quelltext kopiert.

Um die Brennweite der Kugelfläche (von den Scheitelpunkten gemessen) zu berechnen bedarf es einer weiteren Formel:

$$f1 = \frac{n1 \cdot r}{n2 - n1} \quad (F6a) \qquad\qquad f2 = \frac{n2 \cdot r}{n2 - n1} \quad (F6b)$$

wobei hier bei den Vorzeichen folgendes zu beachten ist: Aus der Formel geht direkt hervor, dass f1 und f2 das gleiche Vorzeichen haben müssen, weil sie sich nur im Zähler in n1 und n2, welche ja beide positiv sein müssen (wenn -n1 = n2 würde es zu Reflexion kommen - wird hier aber nicht benötigt) unterscheiden.

Jetzt lege ich fest folgendes fest: Wenn f1 und f2 positiv sind, ist f1 die objektseitige Brennweite (im Applet also links) und f2 die bildseitige. Sind f1 und f2 negativ, so ist f1 die bildseitige Brennweite (also rechts) und f2 die objektseitige.

Wenn man auf die Kugelfläche im Applet einen zur optischen Achse parallelen Strahl fallen lässt, erkennt man den bildseitigen Brennpunkt. Lässt man den Strahl so auftreffen, dass er die Kugelfläche parallel zur opt. Achse verlässt kennt man den bildseitige Brennpunkt. Diese Entfernungen werden in dem Applet links unten mit „f1 =" und „f2 =" angezeigt.
Diese Werte sind über *(F6)* berechnet. Weil es jedoch nicht so einfach ist die Brennpunkte grafisch in das Applet einzuzeichnen (nicht nur die Brennpunkte selbst können links oder rechts liegen, sondern auch die Scheitelpunkte sind vom Vorzeichen des Radius abhängig !), habe ich das bis jetzt (22.1.) wegen extremen Zeitdrucks nicht geschafft. Bis Mitte Februar in Knetzgau sollte ich es aber haben..

Hinweise zum Arbeiten mit dem Applet:

- Es lassen sich die Punkte i_Tli, i_T, i_M und P1 (links von i_S auf der optischen Achse) verschieben.
- Die Kugelfläche ist links von ihrem Mittelpunkt i_M, wenn der Radius positiv ist.
- i_Gs und i_Bs sind die Schnittpunkte der beiden Lichtstrahlen mit der optische Achse, wobei i_Gs der objektseitige und i_Bs der bildseitige Schnittpunkt ist. Wenn i_Gs links von der Linse liegt ist der Wert rechts unten im Applet positiv. Der andere Wert rechts unten wird positiv wenn i_Bs rechts von der Linse liegt.
- die berechneten Werte sind nahezu perfekt exakt, weil die Linsenfehler mitbeachtet werden. Die einzige Abweichung besteht jetzt noch darin, dass der Weg des Lichts nicht ausschließlich durch 'Linien' dargestellte werden darf (wie es in der geometrischen Optik üblich ist), wenn das Ergebnis wirklich ganz exakt sein muss. Dann kommt es nämlich wegen der (in der geometrischen Optik mehr auftretenden) Welleneigenschaft des Lichtes zu weiteren Abweichungen. Auf diese Abweichungen will ich hier nicht weiter eingehen, weil diese Abweichungen wohl nur für ein paar wenige große Firmen eine Rolle spielen.

die berechneten Werte gelten nur für sphärische Flächen, also wirklich kugelförmige und nicht parabolische, welche die Linsenfehler ausgleichen würden (also: für kugelförmige: Brechungsgesetz, sonst evtl. Hauptebenen !).

3.6. Brechung an mehreren sphärischen Flächen (Brechungsgesetz)

Mit dem Brechungsgesetz ist es nun also möglich durch jede beliebige gekrümmte Fläche den Strahlenverlauf des Lichtes zu berechnen. Wie Sie sicherlich schon bemerkt haben ist in dem Applet in 3.5. jeder Punkt mit „i_" am Anfang beschriftet. Das hat folgenden Grund: Wenn die Brechung an einer Fläche klappt (und das tut sie ja zum Glück, s. oben), warum soll es dann nicht auch mit 2, 3 oder 20 Flächen klappen ? Wenn ich einfach den Quelltext für eine Fläche dupliziere und „i_" durch „ii_" ersetze, habe ich schon eine Linse. Wiederhole ich das kann ich auch 2 oder 3 oder noch mehr Linsen zeigen und das ziemlich exakt !
So habe ich mir das anfangs vorgestellt. Aber der Computer ist ja bekanntlich immer für Überraschungen gut und deshalb habe ich den Quelltext nur auf insgesamt 2 Flächen kopieren können, siehe _ap6.html. Bevor ich das Problem mit mehr als 2 Flächen zeige, möchte ich das Applet mit den 2 Flächen, das auch nicht ohne ist, noch beschreiben:

In den Grundzügen entspricht dieses Applet dem aus 3.5., nur dass es wegen der Programmierung einige Besonderheiten gibt:
Das erste Problem ist es, ii_T und ii_Tli festzulegen. Während diese Punkte in (i) noch mit der Maus bewegbar sind, müssen sie in (ii) als Schnittpunkte festgelegt sein, was dadurch, dass beide Flächen nach links oder nach rechts gebogen sein können (Schieberegler positiv/negativ) eine relativ unangenehmes Unterfangen ist. Mit ein paar if-Schleifen habe ich dieses Problem aber lösen können.
Dazu kommen noch etliche andere programmiertechnische Probleme, die aber meistens auch lösbar waren. Manchmal musste ich aber auch auf Kompromisse eingehen, denn dadurch dass man in diesem Applet ungeheuer viel verändern kann (6 Schieberegler und 5 ziehbare Punkte !) ist es schwierig alle Möglichkeiten offen zu lassen, und so kann es dann dazu kommen, dass man zwar z.B. den Abstand der Flächen verändern darf, dabei aber der Lichtstrahl nahe der

optischen Achse verlaufen muss (nur während des Änderns der Entfernung; ist der Abstand eingestellt kann der Strahl wieder überall hin). So gibt es noch einige weitere Kompromisse, die ich treffen musste:

Hinweise zum Arbeiten mit dem Applet, (i) sei die 1. Fläche und (ii) die 2.:

- Der Strahl, der aus (i) austritt muss die 2. Fläche in jedem Fall schneiden !! (viele der folgenden Hinweise resultieren daraus). Dieser Schnittpunkt heißt ii_T.
- Wird die Position von i_T oder i_Tli verändert, darf ii_T nicht von (ii) verschwinden !
- Wird die Position von i_M oder ii_M geändert sollte der Lichtstrahl auf bzw. möglichst nahe an der optischen Achse entlang verlaufen (ansonsten wäre es möglich, dass man die 2. Fläche aus dem austretenden Lichtstrahl der 1. Fläche bewegt, was bedeutet dass es hier keinen Schnittpunkt ii_T gäbe).
- Werden die Radien oder die Brechzahlen über die Schieberegler geändert, darf 1.) der Schnittpunkt auch nicht verschwinden und 2.) sollte die Fläche die schwarze Zeichenfläche nicht verlassen !
- Wird der Radius von (i) verändert, sollte der Punkt i_Tli nicht von der Fläche ‘eingeholt’ werden.
- Der Scheitelpunkt von (i) (bei positivem Radius i_S, sonst i_P8) muss immer links vom Scheitelpunkt von (ii) (bei positivem Radius ii_S, sonst ii_P8) liegen !

In der **_ap6-gali.txt** ist genau beschrieben wie man z.B. ein Galileisches Fernrohr mit dem Applet „bauen“ kann. Dazu wird die einzelne Fläche wie eine dünne Linse betrachtet.
Jetzt können Sie Ihr System noch genauer studieren, in dem Sie die Messwerte betrachten:
Bei den Messwerten ist zu dem Applet in 3.5. hier nur folgendes neu: Unter „**SYSTEM**“ können Sie den Abstand der Scheitelpunkte (= Schnittpunkt zw. Kugelfläche und opt. Achse) ablesen. Der 2. Wert gibt den Abstand der Mittelpunkt voneinander an. Unter „**BILD**“ werden die Abstände der Schnittweiten zu den jew. Scheitelpunkten angezeigt. Diese Werte sind durch die Eingaben (Position der Mittelpunkte, Radien) gegeben.
Analog dazu lassen sich auch andere Systeme darstellen. Will man z.B. eine dicke Linse, müssen die Flächen vergleichsweise nahe aneinander sein. Will man eher 2 dünne Linsen (z.B. Fernrohre, Lupe - Auge, Mikroskop,..) zeigen, so verhält sich das Licht durch eine Kugelfläche mit bestimmter Brennweite f genauso wie durch eine dünne Linse mit der selben Brennweite

Sollte ii_T trotz aller ‘Vorsicht’ einmal verschwinden und der Rechner ziemlich lang rechnen und knören, müssen Sie wohl oder übel wieder die Ausgangsposition neu laden, in dem Sie auf „Aktualisieren“ im Browser klicken (wenn Sie Glück haben reicht es auch die Schieberegler/Punkte so zu verschieben, dass ii_T wieder existiert - mein Prozessor ist mit 233MHz dazu aber leider zu langsam).

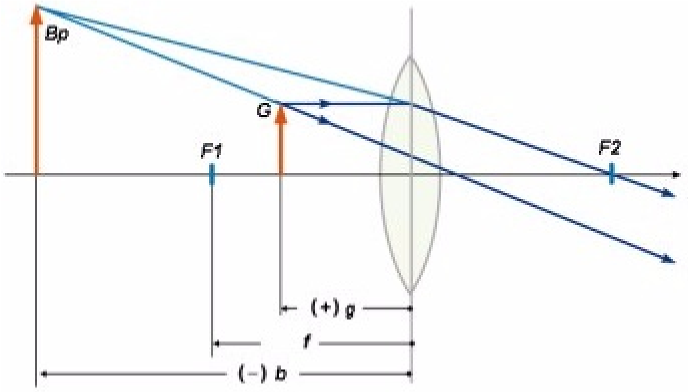
Dieses Applet ist wohl das von der theoretischen Physik her gesehen das beste das ich (bis jetzt) geschrieben habe. Weil es aber unter enormem Zeitdruck entstanden ist, fehlen noch ein paar Details. So sind z.B. die Brennweiten nur berechnet, aber nicht eingezeichnet, weil das wegen dem Vorzeichenschlamassel, das ich in 3.5 beschrieben habe, nicht ganz so einfach hinzubekommen ist. Es ist aber durhaus möglich, dass ich die Brennweiten bis Mitte Februar in Knetzgau mit eingebracht haben werde. Vielleicht lassen sich die Hauptebenen dann auch noch einbinden..

Jetzt aber wieder zurück zu dem Problem mit mehr als 2 Flächen:
In dem oben beschriebenen Applet ist eine Fallunterscheidung mit einprogrammiert, die überprüft ob ii_T existiert und das gegebenenfalls ausgleicht. Der letzte Schritt dieser Fallunterscheidung ist der, dass der Punkt ii_T als von anderen Werten abhängiger Punkt konstruiert wird. Dazu benötigt man ‘point; functionDepend; „x-coordinate“, „y-coordinate“’. Und genau da kommt eine wirre Fehlermeldung von der Java-VirtualMachine, mit der ich nichts anfangen kann. Deshalb habe ich mit Herrn Dr. Ehmke (dem Autor dieser Sprache) telefonisch Kontakt aufgenommen und ihn gefragt, wo der Fehler liegen könnte. Inzwischen sind beinahe 2 Wochen vergangen und ich habe noch ein paar mal mit Herrn Ehmke telefoniert, aber weder er noch ich weiß wo der Fehler liegt - auch Herr Ehmke ist absolut ratlos ! So muss ich mich eben mit dem Kompromiss zufrieden geben, dass ii_T nie verschwinden darf.

Weil sich dieser Fehler schon bei 2 Kugelflächen nicht beheben lässt, ist an ein System aus 3 Flächen kaum zu denken. Vielleicht kommen Herr Ehmke und ich aber auch noch auf den Fehler und ich kann das dann womöglich in Knetzgau zeigen..

3.7. Die Lupe

In dem Applet in der _ap7.html wird der Strahlengang durch eine Lupe gezeigt:



(Abb. 6)

Die Brennweite f ist durch den Schieberegler gegeben und die Gegenstandsweite ist durch die Position von G (ziehbar) gegeben. Anhand der bekannten Linsengleichung

$$\frac{1}{g}+\frac{1}{b}=\frac{1}{f} \qquad (F7)$$

kann ich also die Bildweite b berechnen (die Lupe ist eine dünne Sammellinse, weshalb man diese Formel ohne Bedenken anwenden darf). Bp ergibt sich aus der Geometrie, in dem man die Konstruktion in Abb.6 anwendet (der untere blaue Lichtstrahl ist im Bild etwas daneben geraten, er müsste eigtl. Genau durch die Mitte verlaufen, siehe Applet).

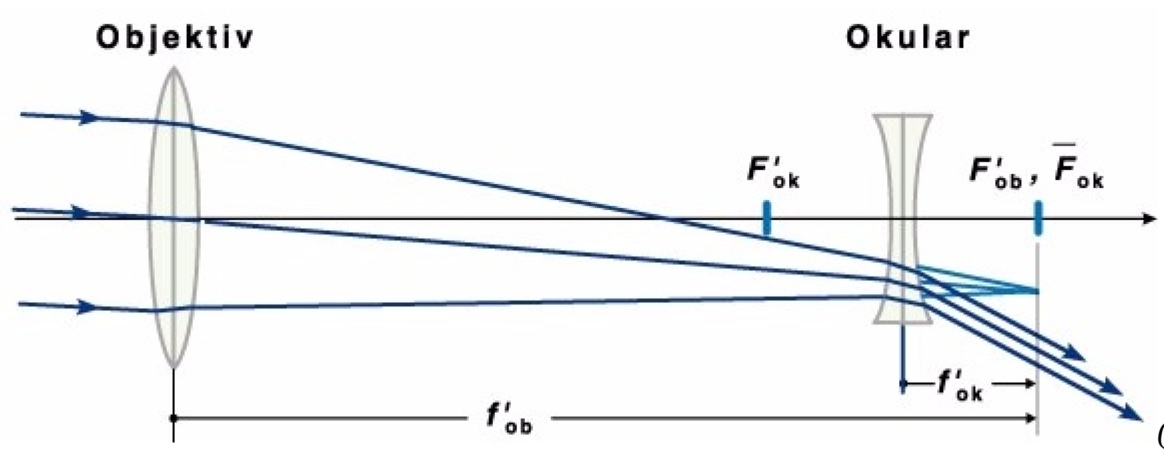
Die Brechkraft der Lupe D ist der reziproke Wert der Brennweite (in Meter). Die Vergrößerung der Lupe v ergibt sich aus folgender Formel:

$$v=\frac{b}{f}-1 \qquad (F8)$$

Wenn man durch eine Lupe einen Gegenstand betrachtet, der in der Brennebene oder zwischen Brennebene und Lupe liegt, so fallen auf das Auge parallele Lichtstrahlen. Das Auge 'denkt' sich dass der Gegenstand weit weg sein muss (weil Strahlen parallel) und akkomodiert auf unendlich (Augenlinse entspannt, so dass paralleles Licht scharf auf die Netzhaut trifft; wenn die Lichtstrahlen nicht parallel sind akkomodiert das Auge so, dass das Bild trotzdem scharf auf die Netzhaut fällt). Deshalb sehen wir ein vergrößertes Bild. Das Auge wird also nur getäuscht. In dem Applet ist links unten eine Linie die die Vergrößerung noch einmal darstellt; der reale Gegenstand ist so groß wie der Abstand der beiden hellblauen Punkte zueinander und die grüne Strecke zeigt die Größe des durch die Lupe gesehenen Bildes.

3.8. Das Galileische oder holländische Fernrohr

In diesem Abschnitt möchte ich zeigen wie das Galileische oder auch holländische Fernrohr (H. Lipperhey, 1608 und G. Galilei, 1609) funktioniert:



(Abb. 7)

Wie in dem Bild zu sehen ist besteht das Galileische Fernrohr aus einer Sammellinse mit großer Brennweite, die als Objektiv dient, und einer Zerstreuungslinse mit kleiner Brennweite, die als Okular dient. Außerdem liegt die Brennebene des Okulars in der des Objektives (siehe Zeichnung). Weil man mit einem Fernrohr nicht wie bei einer Lupe einen relativ nahen Gegenstand vergrößern, sondern die Ferne (z.B. den Himmel) beobachten will, trifft das Licht nahezu parallel auf das Fernrohr. Ein Fernrohr hat jetzt zwei *Aufgaben*: 1. Sehr entfernte Gegenstände unter einem größeren Sehwinkel erscheinen zu lassen und 2. Ein helleres Bild auf das Auge zu werfen.
Wenn man sich das Okular wegdenkt entsteht das Bild in der Brennebene vom Objektiv. Weil die Brennweite des Okulars in der Brennebene des Objektives liegt, entsteht das Bild also auch auf der Brennebene des Okulars. Wie allgemein bekannt ist (und auch durch das Brechungsgesetz-Applets leicht nachvollzogen werden kann) verlässt das Licht, das von einem Gegenstand, der in der Brennebene liegt, ausgeht, eine Linse als paralleler Lichtstrahl. Das heißt dass z.B. ein Stern zuerst auf die Brennebene des Objektives abgebildet wird. Dieses Bild kann dann als Gegenstand für das Okular gesehen werden. Das Bild dieses Gegenstandes liegt im unendlichen rechts vom Okular.
Dabei wird der Strahl ‘dünner’, es trifft also mehr Licht auf das Auge, was bedeutet, dass der Stern unter einem größeren Sehwinkel erscheint (vgl. *Aufgaben* des Fenrohrs). So viel zum Praktischen.

Jetzt zum Theoretischen: Das dazugehörige Applet befindet sich in der _ap8.html. In dem Applet sind die Brennweiten der beiden Linsen durch Schiebregler gegeben, was in der Praxis oft sinnvoller ist, als die Eingabe von Radien, Dicke und Brechzahl. Weil dieses Applet sehr praxisorientiert sein soll, habe ich mich dazu entschlossen den Strahlengang nicht über Brechungsgesetz, Hauptebenen oder Matrixmethoden zu berechnen, sondern das Ergebnis (nämlich dass der Strahl am Ende wieder parallel ist) schon vorwegzunehmen. Dass das tatsächlich so ist kann man in dem Brechungsgesetz-Applet auch erkennen, nur dass man da nicht direkt die Brennweite eingibt, sondern Radien und Brechzahl; in der Praxis steht gewöhlich auf einer Linse nur die Brennweite.
Der Winkel des einfallenden Strahls zur optischen Achse ist durch die Position des dicke roten Punktes gegeben. Diesen Winkel nenne ich φ (im Applet ‘phi’). φ ist also der Einfallswinkel des Strahles vor dem Fernrohr und somit gleichzeitig der Winkel unter dem das Licht das Auge treffen würde, wenn kein Fernrohr vor dem Auge wäre.
Den Winkel zwischen der optischen Achse und dem aus dem Fernrohr austretenden Lichtstrahl nenne ich γ (im Applet ‘gamma’). Diesen Winkel erhalte ich , indem ich den Bildpunkt, der vom Objektiv erzeugt wird, mit dem Mittelpunkt des Okulars verbinde (im Applet also Gerade durch i_Bp und ii_M). Jetzt kommt der Lichtstrahl also von links unter dem Winkel φ auf das Objektiv und wird in Richtung i_Bp (im Applet) gebrochen. Dann trifft der Strahl auf das Okular, wo er unter dem Winkel γ (zur optischen Achse !) wieder gebrochen wird. Die Größe dieser beiden Winkel wird im Applet angezeigt.
Die Vergrößerung v ergibt sich aus folgender Formel:

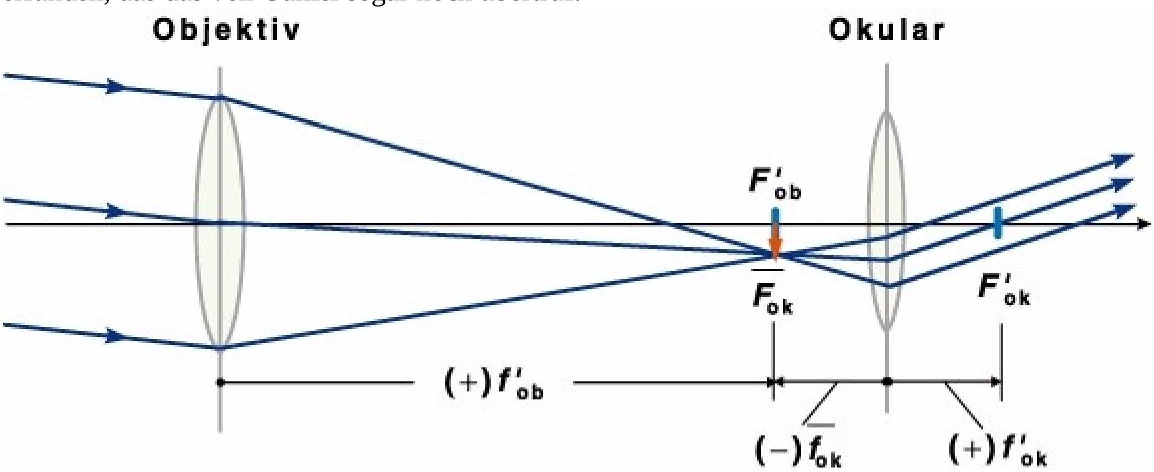
$$v = \frac{\tan\gamma}{\tan\varphi} \qquad (F9)$$

wie in dem Applet schön zu erkennen ist, ist das auch der Faktor zwischen der Breite Strahls vor und nach dem Fernrohr. Damit das ganze Prinzip auch richtig klar wird, habe ich die Linsen als Bilder in das Applet eingefügt. Außerdem kann man auch erkennen wie das Auge funktioniert.

Hinweis zum Applet:
Wenn man mit einem Fernrohr in den Himmel schaut treffen alle Strahlen parallel zur optische Achse bzw. unter einem sehr kleinen Winkel zur optischen Achse auf das Objektiv, was bedeutet, dass in der Realität der Winkel φ sehr klein ist . Nun ist es so, dass man in dem Applet den Winkel φ auch auf 50° oder mehr verschieben kann. Ich habe kein Limit von diesem Winkel (z.B. bei 10°) eingebaut, weil der Strahlenverlauf unter einem größeren Winkel eher deutlich wird. Allerdings sind dann die unten berechneten Werte recht ungenau. Wenn man nun also selbst ein solches Linsensystem experimentell aufbauen will (was ich in Knetzgau anhand dieses Applets machen werde), muss man den Einfallswinkel möglichst klein, am besten auf 0°, halten.
Dieses Applet ist von der theoretischen Physik nicht so anspruchsvoll wie z.B. das Hauptebenen- oder Brechungsgesetz-Applet, dafür in der praktischen Physik aber wesentlich sinnvoller und man muss nichts beim Konstruieren beachten, weil nicht so viele Möglichkeiten enstehen können. Analog dazu verhält es sich mit dem nächsten Applet:

3.9. Das Keplersche oder astronomische Fernrohr

Als Johannes Kepler vom Galileischen Fernrohr gehört hatte, bat er Galilei um ein Exemplar davon. Doch Galilei ging mit seinem Fernrohr zum Vatikan - und es ist bekannt was dort mit ihm geschah. Kepler hat 1611 ein eigenes Fernrohr erfunden, das das von Galilei sogar noch übertraf:



(Abb. 8)

Es besteht wie man sehen kann aus 2 Sammellinsen (Objektiv große Brennweite und Okular kleine Brennweite), wobei der Abstand der beiden Linsen voneinander der Summe der Brennweiten entspricht (F`_ob = F_ok in der Zeichnung). Wenn jetzt wieder paralleles Licht auf das Objektiv fällt, ist der Bildpunkt davon auf der Brennebene des Objektives. Weil diese Brennebene mit der Brennebene des Okulars übereinstimmt, liegt der Bildpunkt auch auf der Brennebene des Okulars. Und ein Punkt der in der Brennebene liegt wird durch eine Linse bekanntlich ins unendliche abgebildet, der Lichtstrahl verlässt das Okular also wieder parallel. Während das Endbild im Galileischen Okular jedoch aufrecht ist, ist das Endbild des Keplerschen Fernrohres umgekehrt, weil sich ja die Lichtstrahlen in der Brennebene kreuzen. In der Astronomie spielt das aber kaum eine Rolle. Wenn man irdische Objekte beobachten will, benutzt man ein *terrestrisches Fernrohr*. In einem solchen Fernrohr wird das Bild einfach noch einmal umgekehrt, in dem man eine dritte Linse mit einbaut. Auch heute noch sind viele Teleskope nach diesem Prinzip aufgebaut.
Wegen Platzmangels und Zeitdrucks kann ich leider nicht genauer auf die Funktionsweise eingehen und gehe deshalb jetzt zur Beschreibung des Applets über. Dieses Applet ist in der _ap9.html einegbettet. Hierzu sei nur noch gesagt, dass sich die Vergrößerung aus f1/f2 ergibt. Die weiße unterbrochene Linie links vom Objektiv soll eine Blende darstellen. In diesem Applet ist es jetzt auch möglich die Position des Auges zu verändern.

4. Ergebnisse

Auf die Ergebnisse meiner Arbeit im Detail bin ich schon in 3. eingegangen. Trotzdem sei hier noch einmal erwähnt, dass ich Applets zu folgenden Themen geschrieben habe:
Brechung des Lichtes, Totalreflexion, Brechung an einer planparallelen Platte, Brechung an dicker Linse (Hauptebenen), Brechung an einer Kugelfläche (Brechungsgesetz), Brechung an mehreren Kugelflächen (Brechungsgesetz), Lupe, Galileisches und Keplersches Fernrohr.
Das Ergebnis ist also eine ganze Palette von Java-Applets die teilweise das Prinzip des Strahlengangs verdeutlichen und teilweise den Strahlengang exakt berechnen. Obwohl es schon Programme gibt, die das ansatzweise können, gibt es im Internet keine Java-Applets, die höhere optische Zusammenhänge darstellen können (und wenn, müssen sie extrem gut versteckt sein, weil ich Stunden lang danach gesucht habe). Im Normalfall gehen die Applets die ich gefunden habe nicht über die Ansätze einer dünnen Linse hinaus.

5. Diskusssion

Neben den Möglichkeiten mit dem Brechungsgesetz und den Hauptebenen den Strahlengang (auch: raytracing) durch Linsen/-systeme zu berechnen, bin ich auf eine dritte, sehr handliche Lösung gestoßen: **Matrizen**.
Ich habe hier bei mir daheim etliche Kopien und Schriftstücke von Physikern (über Mailkontakte zu Unis) auf denen deutlich wird wie diese Matrixmethode funktioniert. Sie ist eigentlich - verglichen mit den Hauptebenen-Formeln - sehr kurz und primitiv. Es lässt sich der Strahlengang durch ganze Linsensysteme mit nur ein paar Inputwerten sehr einfach berechnen, so dass ich mir sicher bin, dass diese Methode die schnellste ist wenn man keinen PC besitzt. Doch auch der Computer kann davon profitieren, weil die Programme nicht so kompliziert wären. Ich habe lange Zeit vergeblich nach einer Möglichkeit gesucht, dem Computer mit Java bzw. JavaScript beizubringen wie er Matrizen miteinander multiplizieren muss. Es wäre wohl möglich gewesen, diese Schritte wirklich auch nur mit +,-,*,/ und Klammern durchzuführen, nur hatte ich leider nicht die Zeit dazu, weil ich erst mal die Applets mit dem Brechungsgesetz vervollständigen musste, so dass ich mit dieser schriftlichen Arbeit erst eine Woche vor Abgabetermin, also dem 16.1., begonnen habe und die Matrixmetode leider vorerst über den Haufen werfen musste.

Vielleicht hätte ich die Hauptebenen wirklich nicht machen sollen und dafür mehr Systeme mit dem Brechungsgesetz. Von meiner gesamten Arbeitszeit ging wohl etwa ein Drittel auf das Hauptebenen-Applet ! Weil es mein erstes Applet war, habe ich noch ziemlich viele Fehler gemacht und es hat eine Ewigkeit gedauert bis das so ausgesehen hat, wie es jetzt ist. Die Applets mit dem Brechungsgesetz sind später entstanden - auf diese Möglichkeit bin ich erst kurz vor Weihnachten gekommen. Aber durch die Erfahrung die man langsam sammelt entstehen die Applets immer schneller und werden immer besser und schöner, wie man an den Applets auf der CD gut erkennen kann (Hauptebenen das erste und Fernrohre die letzten).

Wie Sie im Literaturverzeichnis sehen können habe ich mit nur wenigen Büchern gearbeitet - aber die kann ich fast auswendig. Dafür habe ich mich über das Internet bzw. über email-Kontakte sehr gut informiert. Dabei habe ich v.a. Mitarbeiter von Universitäten und Firmen angeschrieben und ihnen meine Fragen gestellt. Gerade am Anfang war es sehr nützlich jmd. zu kennen der sich wirklich auskennt. Später sind dann durch Java-Probleme auch internationale Kontakte hinzugekommen. Insgesamt zählt der Schüler-Experimentieren-Ordner in meinem Mailprogramm zur Zeit 426 eMails ! Das ist eben ein riesengroßer Vorteil vom Internet - man kann extrem gut mit richtigen Experten kommunizieren, egal ob sie aus Kiel, Dresden oder Indien kommen.

Es sei auch gesagt, dass es schon Programme gibt, die den Strahlenverlauf durch Linsen/-systeme berechnen können. Z.B. verwenden die großen Optikfirmen solche Software. Diese Software arbeitet teilweise über die angesprochene Matrixmethode, weil man dann auch windschiefe Lichtstrahlen durch ein System rechnen kann. Wie ich aber vor

kurzem gehört habe, verwenden diese Firmen dazu teilweise auch das Brechungsgesetz bzw. daraus resultierende Gesetze. Aber für solche Zwecke habe ich das hier ja nicht geschrieben. Es soll eher eine Veranschaulichung der geometrischen Optik sein. Wenn man z.B. in einem Buch liest, dass sich wenn x immer größer wird y verkleinert. Solche Dinge kann man an den Applets sehr gut nachvollziehen. Auch kann es z.B. dabei helfen ein Fernrohr oder ähnliches selbst zu bauen.
Wenn ich diese Arbeit ins Internet gestellt haben werde, kann sich auch wirklich jeder diese Applets direkt im Browser ansehen (ich weiß noch nicht ob netscape navigator geht - aber internet explorer geht auf jeden Fall). Auch wäre der Einsatz im Physikunterricht in Schulen denkbar.
Aber das wichigste ist immer noch, dass mir die Programmierung der Applets sehr viel Spaß gemacht hat und man wirklich ungeheur viel - sowohl in der Physik, als auch in der Programmierkenntnis und der Mathematik) dabei dazulernt...

6. Literaturverzeichnis

hier die von mir verwendete Literatur:

– Bergmann/Schaefer: Lehrbuch der Experimentalphysik Band III Optik, Berlin, 1974 (mehrere Autoren, herausgegeben von H. Gobrecht)
– etliche Kopien, Mails und schriftl. Erklärungen von den folgenden Personen (zufällige Reihenfolge):

Elmar Baumann (Hobbyfotograf - hat mir ganz am Anfang sehr viel helfen können), Daniel Bublitz (Doktorand Uni Jena - Hilfe bei Matrixmethode), Dr. Timo Ehmke (Autor von geometria, Uni Kiel, bei Fragen zu geometria immer sehr nützlich), Prof. Raimund Girwidz (Uni Würzburg - Autor einiger guter geometrie-optik Programme), Holger Heidrich (TU Dresden, hat Applets auf Fehler überprüft und Tipps gegeben, Martin Kamp (Uni Würzburg, sehr große Hilfe bei Hauptebenen und Infomaterial !), Armin Kiessling (Uni Jena, riesige Hilfe zu Matrixmethode), Peter Krahmer (mein Physiklehrer - immer sehr hilfreiche Tipps und sehr große Hilfe), Dr. Maxim Darscht (früher Uni Jena, jetzt Dortmund - hat raytrace-Programm über Linsenfehler geschrieben, bei Fragen zur Optik immer sehr große Hilfe !), Roland Mechling (Autor von dynegeo Euklid - bei Fragen zu Euklid immer hilfreich), Thomas Schott (Uni und Glasmuseum in Jena - Hilfe bei Brechungsgesetz und Hauptebenen), Dr. Volker Tautz (optische Berechnungen für Hensoldt AG Zeiss Gruppe - sehr hilfreich), Jörg Tischer (Chefredakteur Dt. Optiker Zeitung DOZ - sehr nützliche Tipps), Jürgen Kramer (Hilfe bei Brechungsgesetz), B. Surendranath (Indien, Java-Hilfe), Prof. Fu-Kwun Hwang (Dept. of Physics, National Taiwan Normal Univ. - Applet-Hilfe), Sergey Kiselev (Russland, Java-Hilfe).

VIELEN DANK an alle für die Hilfe und die Geduld !!!

Die Bilder sind mir vom Harri-Deutsch Verlag (www.harri-deutsch.de/verlag) großzügigerweise zur Verfügung gestellt worden.

Marcel Schmittfull
Salierstr.10
Tel: 09721/82727
Mail: marcel-sl@gmx.de
97505 Geldersheim

Schüler Experimentieren 2001/2002