Prof. Dr. Jürgen Nolting
Dipl.-Ing. (FH) Peggy Menne

Simulation geometrisch-optischer Versuche mit Java

Gehören Sie auch zu denjenigen, die hin und wieder durch das World Wide Web surfen? Dann sind Sie sicher schon öfter auf den Begriff Java gestoßen. Wer jetzt denkt, dass es sich bei diesem Begriff um eine Insel im Indischen Ozean handelt, liegt völlig falsch. Denn tatsächlich ist Java eine objektorientierte Programmiersprache, die in den letzten Jahren vor allem im Zusammenhang mit dem Internet an Bedeutung gewonnen hat.

Im Rahmen einer Diplomarbeit im Studiengang Augenoptik der FH Aalen entstanden mit Hilfe von Java einige internetfähige Anwendungen (sogenannte Applets) zur Simulation geometrisch-optischer Versuche.

1. Einleitung

Begleitend zur Vorlesung „Geometrische Optik 1 und 2" werden von den Augenoptik-Studenten der FH Aalen im Grundstudium zur Veranschaulichung der Theorie verschiedene Versuchsreihen in einem Laborpraktikum durchgeführt. Die neun Versuche befassen sich vorwiegend mit den Gesetzmäßigkeiten bei optischen Abbildungen: Neben der Bestimmung der Brechzahl einer Planplatte durch Messung des Parallelversatzes und verschiedenen Verfahren zur Brennweitenmessung dünner Plus- und Minuslinsen werden Versuche zur Wirkungsweise eines Ophthalmometers, zum photometrischen Grundgesetz, zu Beleuchtungssystemen inklusive Köhlerscher Beleuchtung und zur sphärischen Aberration durchgeführt. Abbildung 1 zeigt einige Impressionen aus dem Laborpraktikum. Um auch Interessierten außerhalb der FH Aalen einen Einblick in die Thematik zu gewähren, sind auf der Homepage des Studiengangs Augenoptik *(http://www.fh-aalen.de/sga/)* alle Versuche ausführlich beschrieben. Folgen Sie einfach dem Link „Was machen wir? → Optik → Geometrische Optik I (oder II) / Labor".

Abbildung 1: Das Laborpraktikum Geometrische Optik

Nun verfügt allerdings (leider) nicht jeder, der sich für diese Seiten interessiert, über ein optisches Labor. Aus dieser Überlegung entsprang die Idee, internetfähige Anwendungen zur Simulation dieser Versuche zu entwickeln. Die Programmiersprache Java bot für die Verwirklichung dieser Idee die idealen Voraussetzungen. Dank Java und ein „bisschen" Programmierarbeit bietet sich nun jedem Internet-User die Möglichkeit, die ersten vier Versuche ohne großen Aufwand schematisch in einer virtuellen Umgebung nachzuvollziehen. Dabei bildeten folgende Versuche die Grundlage für die Java-Anwendungen:

- **Versuch 1:** Bestimmung der Brechzahl einer Planplatte
- **Versuch 2:** Brennweitenbestimmung dünner Pluslinsen aus Objekt- und Bildweite
- **Versuch 3:** Brennweitenbestimmung dünner Minuslinsen aus Objekt- und Bildweite
- **Versuch 4:** Brennweitenbestimmung dünner Pluslinsen nach Bessel.

Sie finden die virtuellen Versuche entweder über einen Link auf den Praktikumsseiten oder, viel einfacher, über einen Link direkt von der Startseite der Homepage des Studiengangs Augenoptik. Die Abbildung 2 zeigt Fotos der Versuchsaufbauten auf der optischen Bank, wie sie im Laborpraktikum eingesetzt werden.

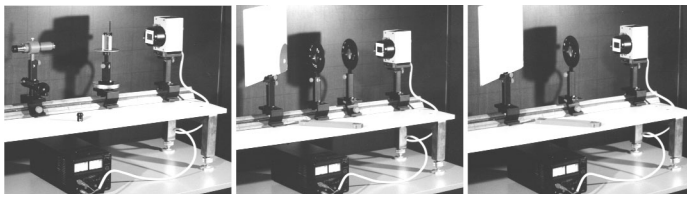

Abbildung 2: Versuchsaufbauten auf der optischen Bank für Versuch 1 (Planplatte, links), Versuch 2 und 3 (Brennweitenbestimmung, Mitte – hier mit Hilfslinsen zur Erzeugung virtueller Objekte oder zur reellen Abbildung virtueller Zwischenbilder) und Versuch 4 (Besselverfahren, rechts)

2. Was ist Java?

1991 wurde bei der Firma Sun Microsystems ein Projektteam mit dem Codenamen „Green Project" gegründet. Durch die zunehmende Integration von Mikroprozessoren in die Haushaltselektronik kam man auf die Idee, für Elektronikgegenstände ein gemeinsames Betriebssystem zu schaffen, das für die Hersteller und Benutzer zu Hause gleichermaßen einfach zu handhaben sein sollte. Das System musste nicht nur unabhängig von einer konkreten Hardware, klein und effizient sein, sondern auf jeden Fall auch sicher, um Abstürze oder Störungen zu vermeiden. Bill Joy, James Gosling, Mike Sheridan und weitere Mitarbeiter des Projektteams nahmen sich dieser Aufgabe an. Der erste Schritt in die Richtung des anvisierten Zieles war die objektorientierte Programmiersprache „Oak" aus der Feder von James Gosling.

Doch Oak ließ sich nicht als Entwicklungstool für die Steuerung von Geräten und interaktiven Fernsehern vermarkten. Dafür erkannte man bei Sun die Vorteile dieser sicheren plattformübergreifenden Programmierumgebung für das Internet. Ziel war nun, Oak als Standardprogrammiersprache für neue Internet-Anwendungen zu etablieren. Doch zunächst musste Oak aus rechtlichen Gründen umbenannt werden. Das Entwicklerteam einigte sich auf den Namen Java, der als Synonym für das Lieblingsgetränk der Programmierer steht – Java ist ein amerika-

nischer Slang-Ausdruck für Kaffee. Um das Potential von Java zu demonstrieren und das Forschungsprojekt vor der Zurückstellung zu bewahren, wurde 1994 von den Programmierern Patrick Naughton und Jonathan Payne ein Web-Browser entwickelt, der Java-Applets ausführen konnte. Ursprünglich hieß dieser Browser WebRunner, bis man ihn in HotJava umtaufte. Java und der Browser HotJava zogen zwar sehr viel Aufmerksamkeit in der Web-Gemeinde auf sich, doch ihren eigentlichen Höhenflug trat die Sprache erst an, als Netscape als erste Firma im August 1995 die Sprache lizenzierte. Die Internet-Anwender nahmen Java mit offenen Armen auf. Nicht zuletzt deswegen, weil Sun den Java-Compiler, die Dokumentation (soweit damals schon fertig) und eine bereits umfangreiche Klassenbibliothek frei verfügbar ins Internet stellte.

Zunächst soll nun die Frage geklärt werden, ob Java eine Compiler- oder eine Interpreter-Sprache ist. Um die Antwort vorweg zu nehmen: man kann zwar vielerorts lesen, Java sei eine Interpretersprache, aber das wird der Lage nicht ganz gerecht, denn tatsächlich ist Java beides. Doch was ist eine Compiler- und was eine Interpreter-Sprache? Diese beiden Typen unterscheiden sich in der Art und Weise, wie mit dem Quelltext des erstellten Programms umgegangen wird: Eine Compiler-Sprache erfordert das Übersetzen des Quelltextes in Maschinenanweisungen. Dies wird von einem speziellen Programm, dem Compiler übernommen. Bekannte Vertreter dieses Sprach-Typs sind C, C++ und Pascal. Bei Interpreter-Sprachen, wie zum Beispiel BASIC wird bei Ausführung des Programms der Quelltext Zeile für Zeile eingelesen, übersetzt und ausgeführt.

Die zeilenweise Übersetzung des Quelltextes erfordert ebenfalls ein spezielles Programm, den Interpreter. Bei Java werden nun beide Konzepte kombiniert: Der Java-Quellcode wird zunächst mit einem Compiler (er heißt JAVAC) übersetzt, allerdings nicht in den Maschinencode des jeweiligen Prozessors, sondern in sogenannten Byte-Code. Man kann sich diesen Byte-Code als den Maschinencode eines virtuellen Prozessors vorstellen. Damit der Byte-Code nun von einem echten Prozessor ausgeführt werden kann, muss er während des Programmlaufs in dessen Maschinencode übersetzt werden, das heißt, ein Interpreter ist zum Ausführen von Java-Programmen notwendig. Den Interpreter hat man einfach wie die Sprache getauft, also JAVA. Häufig findet sich auch die Bezeichnung Java Virtual Machine (JVM) oder Java Runtime Environment.

Der Compiler-/Interpreter-Ansatz bringt einige Vorteile mit sich: Java-Programme sind Hardware-unabhängig und somit portabel. Das kompilierte Programm, das im Byte-Code vorliegt, kann auf jeder beliebigen Plattform beziehungsweise unter jedem beliebigen Betriebssystem ausgeführt werden, das über eine Java Virtual Machine verfügt – das ist heutzutage fast jeder Computer, denn die Virtual Machine ist zumeist in den Web-Browser integriert. Auch auf Quellebene ist Java plattformunabhängig. Es genügt eine Version einer Quelltextdatei. Dabei ist es unerheblich, auf welcher Plattform diese erzeugt wurde – sie kann auf allen Plattformen kompiliert werden. Gleichzeitig bietet der Compiler-/Interpreter-Ansatz ein hohes Maß an Sicherheit vor Manipulationen.

Ein weiterer Pluspunkt für Java ist seine Objektorientiertheit. Als objektorientierte Programmierung, auch OOP genannt, wird eine Methode bezeichnet, bei der Computerprogramme als eine Reihe von Objekten aufgebaut werden, die miteinander interagieren. Eine andere Eigenschaft, die für Java spricht, besteht darin, dass Java klein und gegenüber vergleichbaren Sprachen leichter erlernbar ist, denn ursprünglich stand ja nicht der Standard-PC als Anwendung für Java im Vordergrund, sondern ganz normale Haushaltselektronik. Java sollte daher einfacher zu schreiben, kompilieren, debuggen und zu lernen sein. Die Sprache wurde sehr stark nach dem Vorbild von C++ modelliert, wobei die Programmierer von Java aber die komplexesten und fehlerträchtigsten Aspekte von C++ wegließen. Zusammenfassend lässt sich sagen, dass Java **klein, sicher, plattformunabhängig** und **objektorientiert** ist.

Es gibt zwei verschiedene Arten von Java-Programmen: Applets und Applikationen. Applet bedeutet soviel wie „little application", und das deutet auch schon an, was darunter zu verstehen ist. Applets sind keine selbstständigen Programme, sondern brauchen immer eine bestimmte Umgebung, in der sie existieren und ablaufen können. Diese Umgebung stellt in der Regel ein Web-Browser zur Verfügung (der natürlich Java-fähig sein muss).

Alle modernen Browser wie der Netscape Navigator, der Internet Explorer, HotJava und andere bieten diese Unterstützung. Im Gegensatz zu Applets sind Applikationen vollwertige und eigenständige Programme, die zu ihrer Ausführung keinen Browser und keine besondere Umgebung benötigen (bis auf den Java-Interpreter). Java Applikationen stellen also ein Pendant zu den üblichen C-, Pascal- oder BASIC-Programmen dar.

Alle Programme zum Laborpraktikum sind Applets und können somit direkt im Web-Browser ausgeführt werden. Sie wurden mit dem JDK (Java Developers Kit) von Sun erstellt. Das JDK besteht aus einer Sammlung aller für die Programmierung erforderlichen Tools (Compiler, Interpreter, Debugger, AppletViewer etc.), den zu Java gehörenden Standardbibliotheken und einer Reihe von Beispielprogrammen. Zwar bietet das JDK keine Automatikfunktionen zur Erzeugung von Bedienoberflächen, dafür ist es kostenlos auf der Website von JavaSoft unter *http://java.sun.com* verfügbar. Außerdem unterstützt das JDK stets als erstes Entwicklungstool neue Java-Versionen.

3. Kurze Beschreibung der Versuche

Versuch 1: Bestimmung der Brechzahl einer Planplatte

Bei Versuch 1 geht es darum, die Brechzahl eines Materials zu bestimmen. Das Verfahren, das dazu angewendet wird, basiert auf folgender Tatsache: Ein waagerechter Lichtstrahl, der auf eine schräg gestellte Planplatte trifft, erfährt beim Durchgang durch diese einen Parallelversatz Δ. Der Versatz ist dabei von der Plattendicke d, der Brechzahl n und dem Einfallswinkel ε abhängig. Wenn also d, Δ und ε gemessen werden, kann daraus n mit folgender Formel bestimmt werden:

$$n = \sqrt{\left(\frac{\cos\varepsilon}{1 - \Delta/(d\sin\varepsilon)}\right)^2 + \sin^2\varepsilon}$$

In Abbildung 3 sind die zur Bestimmung der Brechzahl notwendigen Größen dargestellt.

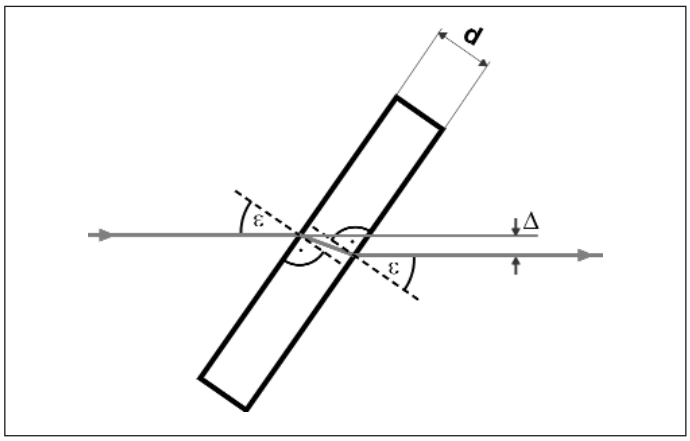

Abbildung 3: Strahldurchgang durch eine Planplatte

Versuch 2: Brennweitenbestimmung dünner Pluslinsen aus Objekt- und Bildweite

Dieser Versuch dient zur Untersuchung der Abbildung durch dünne Pluslinsen. Eine Linse wird dann als dünn bezeichnet, wenn ihre Mittendicke vernachlässigbar ist und beide Hauptebenen näherungsweise mit der Scheitelfläche zusammenfallen. Die Abbildung eines Gegenstandes durch eine dünne Linse wird durch die Abbildungsgleichung beschrieben:

$$\frac{1}{a'} - \frac{1}{a} = \frac{1}{f'} \quad \text{oder in Vergenzform:} \quad A' = A + D$$

Hierbei kennzeichnet a die Objektweite, a' die Bildweite und f' die bildseitige Brennweite. Die Brechkraft D ist definiert als der Kehrwert von f'. Gemäß der Abbeschen Vorzeichenkonvention werden alle Strecken ausgehend vom abbildenden Element gemessen und in Lichtrichtung positiv gezählt. Bei Versuch 2 soll die Bildweite für verschiedene Objektweiten gemessen werden. Aus diesen beiden Größen lässt sich dann die Brennweite ermitteln. Bei der Durchführung des Versuchs sind drei verschiedene Abbildungsfälle zu beachten:

1. Objekt reell, Bild reell
2. Objekt reell, Bild virtuell (die Bildweite kann nur über eine Zwischenabbildung durch eine Hilfslinse ermittelt werden)
3. Objekt virtuell, Bild reell (hier dient eine Hilfslinse zur Erzeugung des virtuellen Objekts)

Versuch 3: Brennweitenbestimmung dünner Minuslinsen aus Objekt- und Bildweite

Der Versuch gestaltet sich analog zu Versuch 2, nur dass diesmal die Abbildung an einer Minuslinse analysiert werden soll. In diesem Versuch sind ebenfalls drei verschiedene Abbildungsfälle zu untersuchen:

1. Objekt reell, Bild virtuell
2. Objekt virtuell, Bild reell (wie in Versuch 2 dient hier eine Hilfslinse zur Erzeugung des virtuellen Objekts)
3. Objekt virtuell, Bild virtuell (hier benötigt man eine Hilfslinse zur Erzeugung des virtuellen Objekts und eine weitere zur Bestimmung der Bildweite)

Versuch 4: Brennweitenbestimmung dünner Pluslinsen nach Bessel

Das Bessel-Verfahren dient der Brennweitenbestimmung bei Pluslinsen. Es basiert darauf, dass es immer zwei Linsenstellungen zwischen Objekt und Schirm gibt, für die sich eine scharfe Abbildung ergibt, sofern der Abstand zwischen Objekt und Schirm größer ist als die vierfache Brennweite. Dies folgt aus einer Symmetrieüberlegung: Hat man eine Stellung gefunden, so kann man die zweite Stellung erhalten, indem der Lichtweg umgekehrt wird, das heißt Objekt und Schirm vertauscht werden. Zu jeder verkleinernden Abbildung lässt sich eine vergrößernde finden und umgekehrt. Abbildung 4 verdeutlicht diese Situation.

Die Brennweite lässt sich nach folgender Formel berechnen:

$$f' = \frac{l^2 - p^2}{4l}$$

Hierbei bedeuten: f': Brennweite
l: Abstand zwischen Objekt und Schirm
p: Abstand der beiden Linsenpositionen voneinander

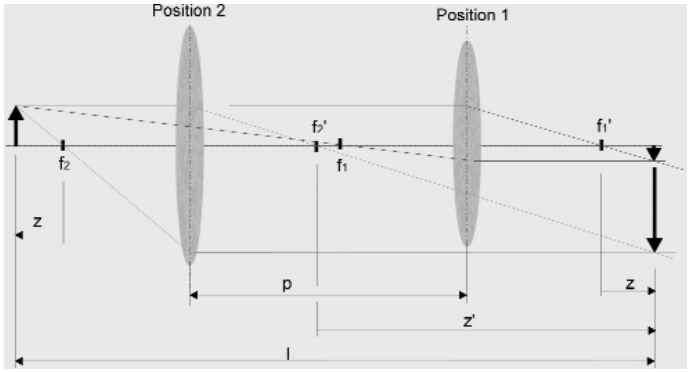

Abbildung 4: Die beiden Linsenstellungen beim Bessel-Verfahren

4. Die Realisierung der Versuche mit Java

Die Applets, die zur Darstellung der beschriebenen Versuche entwickelt wurden, beschränken sich auf die wesentlichen Aspekte. Das heißt, dass in jedem Applet nur das eigentliche abbildende Element, die optische Achse, der Strahlengang und Objekt und Bild visualisiert wurden. Hilfslinsen, Lampen oder die optische Bank fanden keine graphische Umsetzung. Bei allen Versuchssimulationen ist die Anzahl der Messwerte auf fünf beschränkt. Man kann natürlich mehr Messungen durchführen, aber der Mittelwert bezieht sich immer auf fünf Argumente. Die Benutzeroberflächen der Applets sind jeweils in zwei Teile unterteilt: ein Teil beinhaltet die Darstellung der optischen Abbildung, der andere Teil enthält Buttons (Schalter) zur Auslösung von Mess- und Rechenvorgängen sowie Label für Messwerte und Beschriftungen. Die farbliche Gestaltung der einzelnen Elemente in der Grafik wurde einheitlich auf jedes Applet angewandt. Für alle Applets, die eine Linse als abbildendes Element enthalten, beschränkt sich die Darstellung des Strahlenganges auf die Parallel- und Brennpunktstrahlen, sowie auf den Mittelpunktstrahl.

Applet Planplatte

Das erste Applet weist im Gegensatz zu den anderen Applets eine Besonderheit auf: es zeigt Versuch 1 in der Draufsicht. Auf der linken Seite der Benutzeroberfläche befindet sich die Grafik mit der Planplatte. Diese steht beim Start des Applets senkrecht,

und der Einfallsstrahl fällt von links waagerecht auf die Platte. Während der Einfallsstrahl immer waagerecht bleibt, ist die Platte dagegen um bis zu 85° nach rechts drehbar. Ausgelöst wird das durch zwei Buttons, mit denen die Platte jeweils 5° nach rechts oder 5° nach links gekippt werden kann. Ein Label (oben rechts) zeigt dabei immer den aktuellen Wert des Kippwinkels an. Der Kippwinkel entspricht übrigens dem Einfallswinkel. Auf der rechten Seite des Applet-Fensters befinden sich außerdem je fünf tabellenförmig angeordnete Label zur Angabe der gemessenen Parallelversetzungen und der errechneten Brechzahlen. Alle Tabellenlabel haben einen Anfangswert von 0,0. Durch Betätigen der Messung-Buttons (in beliebiger Reihenfolge) wird der Parallelversatz für den gerade aktuellen Kippwinkel in das zugehörige Tabellenlabel aufgenommen. Um möglichst praxisnahe Verhältnisse zu schaffen, baut das Programm beim Versatz einen zufälligen „Messfehler" ein, der zwischen den Werten +0,05 und -0,05 liegt. Für jeden gemessenen Parallelversatz wird automatisch die Brechzahl ausgerechnet und in das entsprechende Tabellenlabel eingetragen. Hat man alle 5 Messungsbuttons benutzt, erfolgt die Freischaltung des Auswertungs-Buttons. Bei Betätigung desselben erscheint in den dafür vorgesehenen Labeln das Ergebnis für den Mittelwert und die Standardabweichung der Brechzahl. Die Abbildungen 5 und 6 zeigen das Applet Planplatte in Grundstellung und während einer Messung.

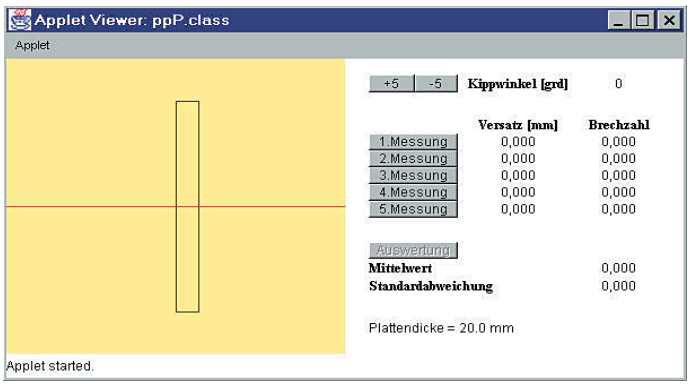

Abbildung 5: Applet Planplatte in Ausgangsposition

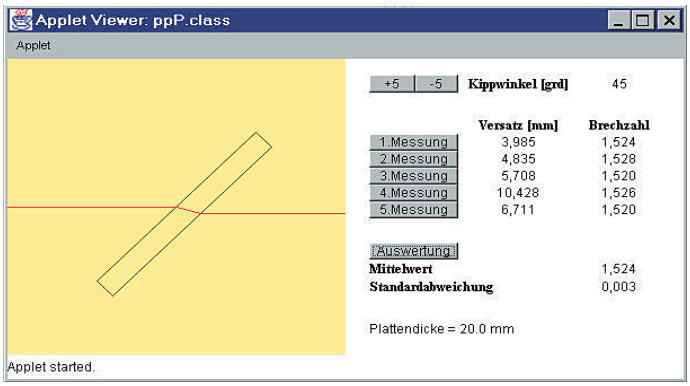

Abbildung 6: Applet Planplatte während einer Messung

Applet Pluslinse

In der oberen Hälfte des Applet-Fensters befindet sich eine Grafik, welche schematisch die Abbildung an einer Pluslinse (f' = 100 mm) zeigt. Diese Grafik beinhaltet eine mittig angeordnete Pluslinse mit Hauptebene, optischer Achse und Brennpunkten. Objekt und Bild sind als Pfeile dargestellt. Die Objektweite lässt sich auf drei verschiedene Arten variieren:

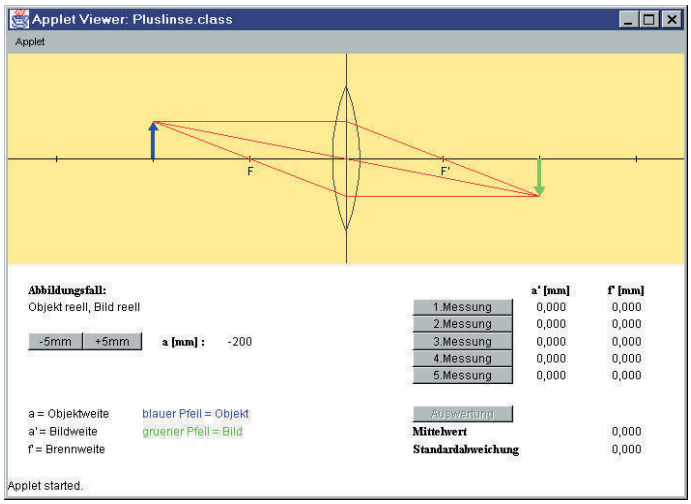

Abbildung 7: Applet Pluslinse in Ausgangsposition

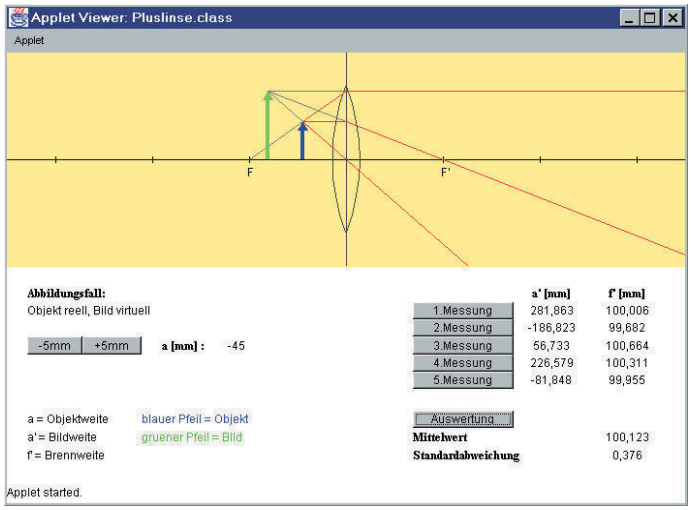

Abbildung 8: Applet Pluslinse während einer Messung

- Erste Möglichkeit: Man klickt eine bestimmte Stelle auf der optischen Achse an – das Objekt springt daraufhin zu dieser Stelle.
- Zweite Möglichkeit: Man klickt mit dem Mauszeiger direkt auf das Objekt und zieht es mit gedrückter Maustaste an die gewünschte Stelle.
- Dritte Möglichkeit: Man stellt die Objektweite mit Hilfe der Plus- und Minus-Buttons ein.

Die Schrittweite ist in allen drei Fällen gleich: sie beträgt 5 mm. Bei einer Positionsänderung des Objektes bewegt sich der Strahlengang stets mit. Ein Label zeigt dabei immer den aktuellen Wert der Objektweite an. Außerdem existiert noch ein weiteres Label, das den jeweils aktuellen Abbildungsfall angibt. Der Anfangswert für die Objektweite beträgt -200 mm, dies entspricht dem Abbildungsfall „Objekt reell, Bild reell", der ebenfalls angezeigt wird. In der unteren rechten Hälfte des Applet-Fensters befinden sich je fünf tabellenförmig angeordnete Label zur Angabe der gemessenen Bildweiten und der errechneten Brennweiten. Alle Tabellenlabel haben einen Anfangswert von 0,0. Ein Klick auf einen der Messung-Buttons bewirkt, dass die aus der gerade aktuellen Objektweite resultierende Bildweite in das zugehörige Tabellenlabel aufgenommen wird. Um den realen Versuchsbedingungen möglichst nahe zu kommen, baut das Programm bei der Bildweite einen zufälligen „Messfehler" von maximal 1 Prozent ein. Für jede

gemessene Bildweite wird automatisch die Brennweite ausgerechnet und in das entsprechende Tabellenlabel eingetragen. Hat man alle 5 Messungs-Buttons benutzt (die Reihenfolge spielt dabei keine Rolle), erfolgt die Freischaltung des Auswertungs-Buttons. Bei Betätigung desselben erscheint in den dafür vorgesehenen Labeln das Ergebnis für den Mittelwert und die Standardabweichung der Brennweite. Übrigens können die Messungs-Buttons nicht immer betätigt werden: Wenn die Objektweite gleich Null ist, oder wenn sie der objektseitigen Brennweite entspricht, werden die Buttons gesperrt. In diesen beiden Fällen wäre nämlich im praktischem Versuch keine Messung der Bildweite möglich. Die Abbildungen 7 und 8 zeigen das Applet Pluslinse in Grundstellung und während einer Messung.

Applet Minuslinse

Dieses Applet verhält sich wie das Pluslinse-Applet. Eingesetzt wird hier eine Minuslinse der Brennweite -200 mm. Die anfängliche Objektweite ist -400 mm, dem entspricht der Abbildungsfall „Objekt reell, Bild virtuell". Die Abbildungen 9 und 10 zeigen das Applet Minuslinse in Ausgangsposition und während einer Messung.

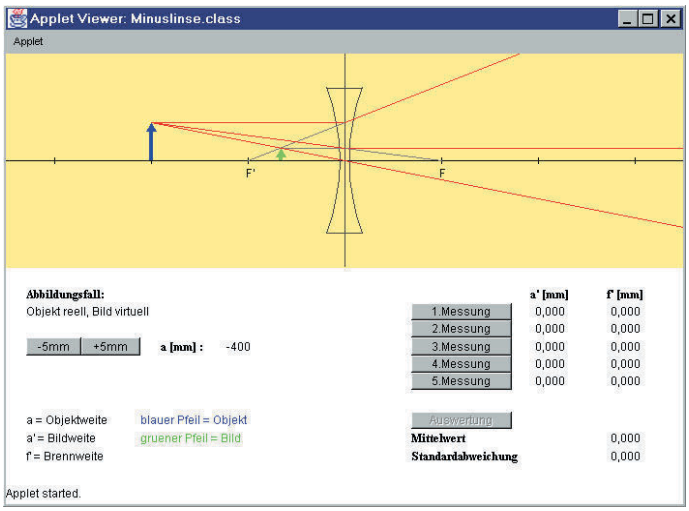

Abbildung 9: Applet Minuslinse in Ausgangsposition

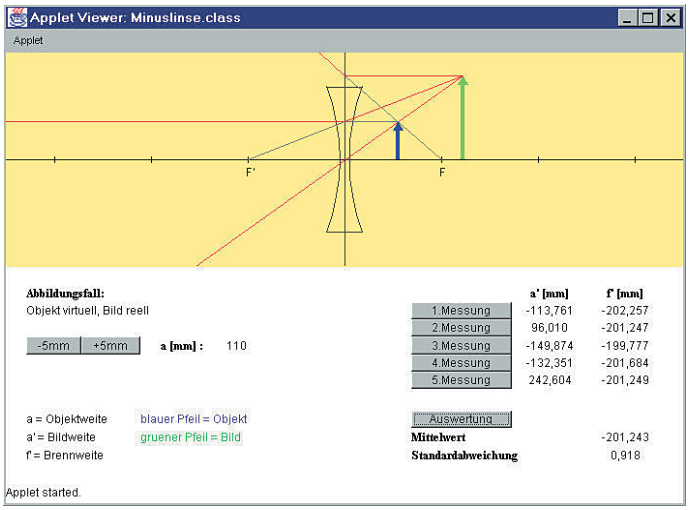

Abbildung 10: Applet Minuslinse während einer Messung

Applet Bessel

Die Grafik im oberen Teil des Bessel-Applets zeigt eine Pluslinse (f' = 100 mm) mit Hauptebene, optischer Achse und Brennpunkten. Außerdem wurde ein Schirm in den Strahlengang gebracht. Das Objekt besitzt diesmal einen festen Standort. Dafür kann der Benutzer die Linse verschieben. Der Strahlengang, das Bild und auch die Brennpunkte bewegen sich entsprechend mit. Ein Label zeigt dabei fortwährend die aktuelle Objektweite an. Bilder, die hinter dem Schirm entstehen würden, sind durch die Farbe Grau gekennzeichnet, da es sich nicht um eine reale Abbildung handelt. Auch die hinter dem Schirm verlaufenden Strahlen sind in diesem Fall grau, um zu unterstreichen, dass sie nicht wirklich hinter dem Schirm weiter laufen. Abgesehen von dieser Ausnahme enden alle Strahlen auf dem Schirm. Die Linsenposition kann entweder durch Klicken auf eine Stelle der optischen Achse erfolgen – die Linse springt daraufhin zu dieser Stelle. Alternativ kann die Linse mit dem Mauszeiger direkt auf die gewünschte Stelle gezogen werden. Die Feineinstellung der Linsenposition kann dann mit Hilfe der Plus- und Minus-Buttons erfolgen. Die Schrittweite beim Verschieben der Linse beträgt 1 mm. Zusätzlich zum Linsenstandort ist auch der Abstand des Schirmes vom Objekt einstellbar. Der Benutzer kann dazu aus einer Liste zwischen fünf verschiedenen Abständen wählen. Jedem Abstand ist dabei ein bestimmter Messungs-Button zugeteilt. Dieser wird erst freigeschaltet, wenn das Bild in die Nähe des Schirmes gelangt. Hat man das Bild durch Verschieben der Linse „scharf gestellt", das heißt auf den Schirm gebracht, muss der freigeschaltete

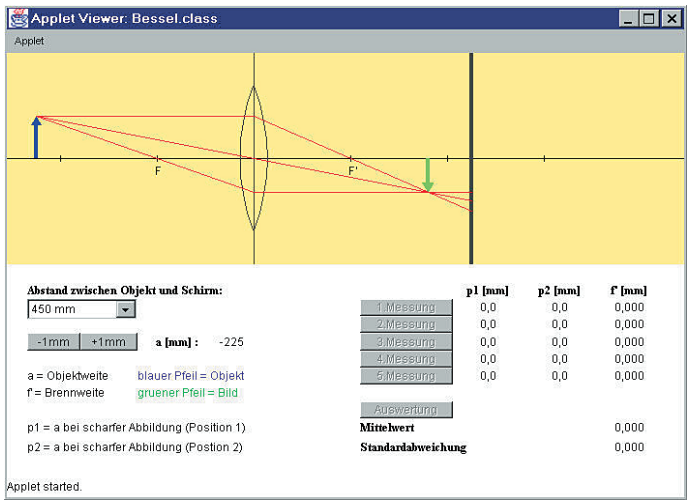

Abbildung 11: Applet Bessel in Ausgangsposition

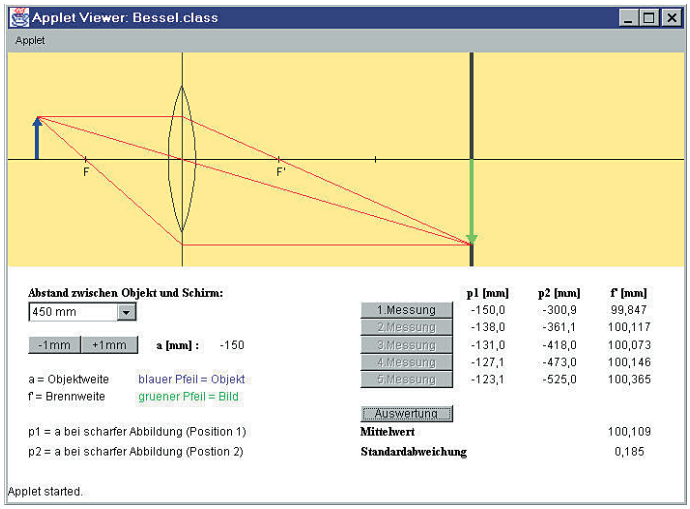

Abbildung 12: Applet Bessel während einer Messung

Messungs-Button betätigt werden. Danach verschiebt man die Linse in die andere Richtung, bis sich das Bild erneut mit dem Schirm deckt und klickt nochmals auf den Button. Beim Klick auf einen der Messungs-Buttons wird die jeweils aktuelle Objektweite in das entsprechende Tabellen-Label übertragen. Dabei existieren zwei Tabellenspalten für die Objektweiten, wobei jede Spalte immer für eine bestimmte Position der Linse vorgesehen ist. Dadurch kann es nicht passieren, dass bei zweimaliger Messung derselben Linsenposition beide Tabellenfelder ausgefüllt werden und es somit zu einer Verfälschung des Ergebnisses kommt. Nach der Bestimmung der beiden Objektweiten errechnet das Programm automatisch die Brennweite und zeigt diese in dem dafür vorgesehenen Tabellen-Label an. Um die Sperrung des Auswertungs-Buttons aufzuheben, müssen zunächst alle Tabellenfelder für die Objektweiten mit Werten gefüllt sein. Ein Klick auf den Auswertungs-Button löst dann die Berechnung des Mittelwertes und der Standardabweichung der Brennweite aus. Die Ergebnisse sind schließlich in den zwei Labeln rechts unten zu sehen. Die Abbildungen 11 und 12 zeigen das Applet Bessel in Ausgangsposition und während einer Messung.

5. Fazit

Java hat sich in den letzten Jahren als die Sprache für das Internet etabliert. Wer jetzt allerdings glaubt, Java sei nur gut, um seine Homepage etwas aufzupeppen, der hat weit gefehlt. Denn Java bietet auch alles, was für die Erstellung eigenständiger Programme nötig ist. Wenn Sie jetzt Lust bekommen haben und mehr über Java erfahren möchten, brauchen Sie sich nicht unbedingt teure Bücher anzuschaffen. Reichlich Informationen finden sie nämlich im World Wide Web, zum Beispiel unter diesen Adressen:

- *http://www.pinstore.purespace.de/books/java/inhalt.htm#TopOfPage* Java 1.1 in 21 Tagen
- *http://duke.htl-leonding.ac.at/javabuch/cover.html* Go To Java 2 – Handbuch der Java-Programmierung
- *http://java.sun.com/products/jdk/1.2/docs/api/index.html* JavaTM 2 Platform, Standard Edition, v1.2.2 API Specification (Standard-Bibliothek)

Für uns hat sich jedenfalls der Programmieraufwand bezahlt gemacht: die Homepage des Fachbereichs Augenoptik der FH Aalen ist wieder um eine kleine Attraktion reicher. Es lohnt sich also, einmal vorbeizuschauen. Wir würden uns jedenfalls freuen.

Anschrift des Autors
Prof. Dr. Jürgen Nolting, Aalen
Dipl.-Ing. (FH) Peggy Menne
FH Aalen – Fachbereich Augenoptik,
Gartenstraße 135, 73430 Aalen

Literatur:

[1] Lemay, Laura/Cadenhead, Rogers: „JAVA 2 in 21 Tagen", Markt & Technik Buch- und Software-Verlag GmbH, München, 1999

[2] Louis, Dirk/Müller, Peter: „Jetzt lerne ich JAVA", Markt & Technik Buch- und Software-Verlag GmbH, München, 1998

[3] Jesse, Ralf: „JAVA 2 – Das Taschenbuch", bhv Verlags GmbH, Kaarst, 2000

[4] Universität Hannover „JAVA – Begleitmaterial zu Vorlesungen/Kursen", Regionales Rechenzentrum für Niedersachsen, 1999

[5] Prof. Dr. Jürgen Nolting/Dipl.-Ing. (FH) Michael Hermann „Praktikum Geometrische Optik" (Skript), Fachbereich Augenoptik, Fachhochschule Aalen, 1997