Hochschule Aalen

Bachelorarbeit des Studiengangs
Augenoptik / Augenoptik und Hörakustik

Entwicklung einer Android-Applikation zur Berechnung von optischen Formeln

vorgelegt von

René Nierath

Matrikelnummer: 47452

Erstprüfer: Prof. Dr. Jürgen Nolting

Zweitprüferin: Prof. Dr. Ulrike Paffrath

Aalen, 16.04.17

Eidesstattliche Erklärung

Mit dieser Erklärung versichere ich, dass die vorliegende Bachelorthesis eigenständig und ohne fremde Hilfe verfasst wurde. Es wurden ausschließlich die angegebenen Quellen und Hilfsmittel benutzt, sowie sämtliche wörtlichen wie sinngemäßen Zitate kenntlich gemacht. Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Ort, Datum

Unterschrift

Danksagung

Ich möchte mich an dieser Stelle ganz herzlich bei meinen Eltern Iris Nierath und Vladimir Nierath bedanken, die mir dieses Studium erst ermöglichten, mich während dieser Zeit finanziell unterstützten, in den Semesterferien aufnahmen und mir während des Schreibens dieser Arbeit viel Verständnis entgegenbrachten.

Bedanken möchte ich mich bei meinem Erstprüfer Herrn Prof. Dr. Jürgen Nolting, der dieses für die Augenoptik doch recht ungewöhnliche Thema absegnete, hilfreiche Kritik sowie Anregungen zu den Applikations-Prototypen äußerte und diese Arbeit begutachtet hat.

Mein Dank gilt außerdem meiner Zweitprüferin Frau Prof. Dr. Ulrike Paffrath für die stets schnellen Antworten, das positive Feedback und ebenfalls für die Betreuung meiner Arbeit.

Darüber hinaus möchte ich mich bei Herrn Prof. Dr. Peter Baumbach bedanken, da das Teilprogramm *Abbildungsfehler eines sph.-tor. Brillenglases* auf einem Grundkonzept seiner Arbeit basiert und ich zu diesem Programmabschnitt ebenfalls hilfreiche Kritik erhielt.

Zu guter Letzt möchte ich mich bei meinen Freunden Ulrich Wächtler und Rudolf Patzke bedanken, die meine ständige Meckerei ertrugen und mir wertvolle Ratschläge zur Arbeit und darüber hinaus gaben.

Inhaltsverzeichnis

Formelzeichen

Zeichen	Einheit	Bedeutung
s_m	mm	objektseitige Schnittweite der Fläche *m*
s'_m	mm	bildseitige Schnittweite der Fläche *m*
n_m	-	Brechungsindex des Materials *m*
r_m	mm	Radius der Fläche *m*
d_m	mm	Abstand zwischen den Flächen *m* und *m+1*
β	-	Lateralvergrößerung
β_t	-	Tiefenvergrößerung
β_w	-	Winkelvergrößerung
f'	mm	bildseitige Brennweite
f	mm	objektseitige Brennweite
D_{Ges}	dpt	Gesamtwirkung des optischen Systems
$s'_{H'}$	mm	Lage des bildseitigen Hauptpunkts
s_H	mm	Lage des objektseitigen Hauptpunkts
k'	mm	Lage des bildseitigen Knotenpunkts
k	mm	Lage des objektseitigen Knotenpunkts
u_m	°	Winkel zwischen dem einfallenden Lichtstrahl und der optischen Achse vor der Fläche *m*
u'_m	°	Winkel zwischen dem ausfallenden Lichtstrahl und der optischen Achse nach der Fläche *m*
ε_m	°	Winkel des einfallenden Lichtstrahls zum Lot der brechenden Fläche *m*
ε'_m	°	Winkel des ausfallenden Lichtstrahls zum Lot der brechenden Fläche *m*
Sph	dpt	Sphäre
Zyl	dpt	Zylinder
α	°	Achsangabe des Zylinderwerts

λ	nm	Wellenlänge
$A_{0/1/2/3/4/5}$		Konstanten der Dispersionsformel
p_m		Reflektionsgrad der Fläche *m*
T_m		Transmission durch die Fläche *m*
T_0		Reintransmission
d_0	mm	Referenzabstand der Reintransmission von 10 mm
P		Positions-Faktor P
$S_{komafrei}$		Shape-Faktor S für ein komafreies Glas
S_{opt}		Shape-Faktor S für ein Glas mit minimaler sphärischer Aberration
$F_{1/2/3}$	dpt	Flächenbrechkraft der Vorder- und Rückfläche; Bei torischen Gläsern entsprechend des jeweiligen Hauptschnitts
$t_{1/2/3}$	mm	Scheiteltiefe
d_r	mm	Randdicke
Ø	mm	Durchmesser
ρ	$\frac{g}{cm^3}$	Dichte des Glasmaterials
b'	mm	Abstand des Augendrehpunkts bis zur Rückfläche des Glases
h	mm	Bauhöhe des Rohglases
V_{Ges}	mm^3	Gesamtes Volumen des Rohglases
m_{Glas}	g	Masse des Rohglases
β	°	Blickwinkel
T_β	dpt	Kehrwert der tangentialen Schnittweite
S_β	dpt	Kehrwert der sagittalen Schnittweite
A_β	dpt	Astigmatismus (schiefer Bündel)
R_β	dpt	Refraktionsfehler
Vz_β	%	Verzeichnung

Erläuterungen

Begriff	Bedeutung
Open Source	Eine Software deren Programmcode vollständig einsehbar ist.
Wearable	Überbegriff für elektronische Geräte, die direkt am Körper getragen werden und den Nutzer mit zusätzlichen Informationen versorgen. So bieten Smartwatches neben ihrer ursprünglichen Funktion, die der Wiedergabe der Uhrzeit, weitere Informationen wie die Überwachung von Puls und Blutdruck, dienen als Schrittzähler oder lassen sich mit dem Smartphone verbinden.
Kompilieren	Beschreibt den Prozess des Umwandelns von Programmcode in ausführbaren Code. Im Falle von Android Studio wird der Programmcode Java zu einer APK-Datei umgewandelt die zur Installation auf Android-Geräten benötigt wird.
Entwickler-Optionen	Ein verstecktes Untermenü in den Android-Einstellungen, welches zusätzliche Konfigurationsmöglichkeiten des Gerätes ermöglicht.
Debugger-Modus	Die USB-Debugging-Einstellung bei Android-Geräten ermöglicht einen Direktzugriff auf das Gerät von Seiten des Computers. Dies ermöglicht ein Auslesen bestimmter Geräte- oder Sensorinformationen, sowie das Installieren von Software.
GridLayout	Layout-Element von Android Studio, welches ein Ausrichten von Objekten, ähnlich der einer Tabelle mit Hilfe von Zeilen und Spalten, ermöglicht.
Unbekannte Quellen	Diese Einstellung in den Sicherheitsoptionen eines Android-Gerätes ermöglicht die Installation von Applikationen die von außerhalb der Verkaufsplattform Google Play bezogen werden.

ListView	Layout-Element von Android Studio, welches die einzelnen Elemente eines Strings auflistet.
String	Objektklasse die eine bestimmte Zeichenfolge als Wert abspeichert.
Activity	Unter Activity wird der komplette Inhalt einer Bildschirmseite bestehend aus der jeweiligen Java-Klasse und dem dazugehörigen Layout zusammengefasst.
Event	Auslösen eines bestimmten Ereignisses bedingt durch einen genau definierten Reiz wie das Erfüllen einer bestimmten Bedingung oder das Drücken eines Buttons.
Button	Objekt in der Layout-Umgebung welches als Schalter zum aktivieren bestimmter Ereignisse dient. Das Objekt kann durch kurzes oder langes Drücken aktiviert werden und löst je nachdem ein unterschiedliches Event aus.
Checkbox	Objekt in der Layout-Umgebung welches durch Betätigen entweder den Status wahr oder falsch annehmen kann.
EditText	Layout-Element welches eine Eingabe von Parametern oder Text ermöglicht.
ID	Zahlen- oder Buchstabenkombination die zur Identifikation eines Layout-Elements dient.
Spinner	Layout-Element, das durch Antippen eine Liste mit auswählbaren Objekte anzeigt.
Plugin	Software-Erweiterung, die zusätzlich installiert werden muss und den Funktionsumfang einer bestehenden Software erweitert oder verändert.

Abstract (deutsch)

In dieser Bachelorarbeit wurde die Android-Applikation *Aalener Optik-Formelrechner* entwickelt. Sie soll die Studenten des Studiengangs Augenoptik bei der Berechnung einer Vielzahl von optischen Funktionen unterstützen. Zudem kann sie als Plattform für interaktive Vorlesungen verwendet werden und damit Studieninhalte effizienter vermitteln sowie den Lernerfolg steigern. Die implementierten Funktionen können semesterübergreifend verwendet werden.

Darüber hinaus werden der Aufbau und die Funktionsweise der Applikation im Detail beschrieben. Die Applikation wird von Geräten mit einer Android-Version 4.0.3 oder höher unterstützt und kann kostenlos über die Verkaufsplattform Google Play Store bezogen werden. Sie beinhaltet keinerlei Werbung und benötigt keine Berechtigungen, die auf das Gerät zugreifen.

Aktuell sind folgende Funktionen implementiert:

- Paraxiales Raytracing
- Raytracing für einen beliebig einfallenden Lichtstrahl
- Berechnung von schiefgekreuzten Zylindern mit optionalem sphärischen Anteil
- Berechnung von Brechungsindices nach der Schottgleichung
- Berechnung der relativen Transmission
- Ermittlung der Radien für komafreie Gläser und minimale sphärische Aberration
- Berechnung und grafische Darstellung von Refraktionsfehler, Astigmatismus und Verzeichnung sowie Volumen, Gewicht und Abmessungen von individuellen sphäro-torischen Brillengläsern

Bei den derzeitigen Teilprogrammen handelt es sich um in sich geschlossene Strukturen, die sich problemfrei ersetzen, tauschen oder sogar in andere Applikationen eingliedern lassen. Durch den Aufbau der Applikation lassen sich neue Funktionen sehr einfach hinzufügen.

Abstract (englisch)

In this bachelor thesis the Android application *Aalener Optik-Formelrechner* was developed. It is intended to assist optometry students with the computation of a variety of optical calculations. In addition it can be utilized as a teaching tool for interactive lectures, increasing the efficiency of the teaching process and optimizing the learning outcome. The app comprises functions suitable during different phases of the optometry studies.

The structure and functionality of the app are described in detail. The app is compatible with devices running on Android 4.0.3 or higher. It is distributed via the Google Play Store. All parts of the software are open source and free. No advertising has been included. Specific permissions are not required.

Currently the following functions are included:

- Paraxial raytracing
- Exact raytracing of a light beam starting from an axis point
- Calculation of spherocylindrical combinations by means of power vectors
- Calculation of the refractive index
- Calculation of relative transmission
- Determination of the geometric design for lenses without coma and with minimal spheric aberration
- Calculation of the refractive error, astigmatism, distortion, design height, volume, weight and dimensions of spherical and toric lenses

These functions are coded as standalone and can easily be replaced or even implemented in other applications without problems. The structured approach to programming facilitates the addition of subprograms. Thus the further enhancement of the range of application fields covered by the app can be achieved efficiently.

1 Einleitung

Das übergeordnete und persönliche Ziel dieser Bachelorthesis war die Entwicklung einer nachhaltigen und sinnvollen Smartphone-Applikation, welche den Studenten des Studiengangs Augenoptik und Hörakustik die Möglichkeit zum Selbststudium bietet. So wird man im Verlauf des Studiums mit einigen Aufgabenstellungen konfrontiert, bei denen ein durchschnittlicher Taschenrechner an die Grenzen seiner Speicherkapazität gelangt, oder aber eine schnelle und übersichtliche Berechnung ohne ein Hilfsprogramm nicht möglich ist.

Dieses Programm soll vor allem zur Kontrolle von Rechenfehlern dienen, kann aber auch die Planung von optischen Versuchen erleichtern.

Um dies einem möglichst großen Publikum komfortabel zu ermöglichen und eine hohe Verfügbarkeit zu gewährleisten, fiel die Wahl auf das Android-Betriebssystem der Firma Google Inc., welches mit 88% im dritten Quartal 2016 weltweit den höchsten Nutzeranteil besaß (Sui 2016).

Das Smartphone ist inzwischen ein fester Bestandteil der Gesellschaft und wird neben sozialen Plattformen und Gelegenheitsspielen, mittlerweile auch verstärkt in der Arbeitswelt eingesetzt. So statteten 61% der Unternehmen 2016 ihre Mitarbeiter mit einem Smartphone oder Tablet aus, mit der Intention mobiles Arbeiten zu ermöglichen (Destatis 9. Dezember 2016). Auch an Hochschulen und Universitäten werden Smartphones im größeren Umfang eingesetzt. Lerngruppen werden über Messenger organisiert, Präsentationen über PowerPoint Mobile gehalten und Skripte über das Tablet verfolgt. Da die Geräte jedes Jahr leistungsstärker werden und viele Funktionen ersetzen, die zuvor nur auf PCs möglich waren, verzeichneten Hersteller konventioneller Computer und Laptops 2016 einen weiteren Marktrückgang und damit einen bisher fünfjährigen Negativtrend (Gartner 11. Januar 2017).

Zwar bietet die Verkaufsplattform Google Play bereits einige Applikationen zum Thema Augenoptik und ebenfalls zur Berechnung von optischen Formeln, jedoch verfügt keine über den notwendigen Umfang, bietet die deutsche Sprache an oder ist auf den Studiengang und seine Bedürfnisse zugeschnitten. Die insgesamt sieben verfügbaren Teilprogramme wurden in Absprache mit dem Erstbetreuer

Herrn Prof. Dr. Jürgen Nolting, der Zweitbetreuerin Frau Prof. Dr. Ulrike Paffrath und Herrn Prof. Dr. Peter Baumbach ausgewählt. Die entwickelte Applikation ist von der aktuellen Android-Version 7.1.1 bis zur Version 5.0 vollständig kompatibel, lässt sich aber bis herunter zu Version 4.0.3 mit kleinen Einschränkungen bei der Layout-Darstellung nutzen. Damit ist die Applikation für 97 % der Geräte verfügbar, die aktuell den Google Webstore besuchen (Google 2000-2017).

2 Material und Methoden

2.1 Installation und Inbetriebnahme

Die in dieser Thesis beschriebene Applikation wurde mit Hilfe der Software Android Studio 2.3 entwickelt. Android Studio ist eine kostenlose Programmieroberfläche, welche durch Google Inc. als Entwickler des Android-Betriebssystems bereitgestellt wird. Mit ihrer Hilfe lassen sich neben der Entwicklung von Applikationen für Smartphones und Tablets ebenfalls Programme für sogenannte *Wearables* erstellen. Das Programm ist neben Windows auch für Linux und Mac verfügbar.

Als Programmiersprache dient *Java***,** welche 1995 durch das Entwicklerstudio Oracle Corp. veröffentlicht wurde (Oracle 2015). Das Java Development Kit, kurz JDK, lässt sich in der Standard Edition kostenlos über die Webseite des Herstellers beziehen und wird zum *Kompilieren* der Applikationen benötigt (Louis et al. 2016). Um die Applikationen vorab zu testen, bietet Android Studio die Möglichkeit mit Hilfe der integrierten virtuellen Maschine eine Android-Version nach Wahl zu simulieren. Ebenfalls ist es möglich, diese auf ein verbundenes Android-Smartphone zu überspielen, welches über einen Zugang zu den *Entwickler Optionen* verfügt und den *Debugger-Modus* aktiviert hat.

Bei jedem Anlegen eines neuen Projektes, d. h. einer neuen Applikation, wird zu Beginn nach dem Kompatibilitätsumfang gefragt. Neuere Android-Versionen garantieren einen größeren Funktionsumfang, decken folglich jedoch auch weniger Geräte ab. Die in dieser Thesis beschriebene Applikation ist für Geräte ab der Android-Version 4.0.3 verfügbar, bietet jedoch erst ab Version 5.0 eine vollständige Kompatibilität. So ist es erst seit dieser Version möglich, innerhalb von *GridLayouts* mit der Funktion *layout_columnWeight* zu arbeiten, die die gezielte Ausrichtung von einzelnen Spalten ermöglicht. Frühere Versionen bieten zwar den vollen Funktionsumfang, es kann jedoch an einigen Stellen zu Problemen bei der Ausrichtung der Objekte kommen.

Für die Applikation ist die Veröffentlichung einer kostenlosen Version auf der Verkaufsplattform Google Play unter dem Namen *Aalener Optik-Formelrechner* geplant. Um Applikationen zu veröffentlichen wird ein Entwickler-Konto benötigt, welches sich für den einmaligen Preis von 25 $ freischalten lässt. Eine Veröffentlichung über diese Plattform garantiert eine höhere Reichweite und einen unkomplizierten Zugang, da die Verkaufsplattform bei den meisten Android-Geräten bereits vorinstalliert ist. Eine alternative Downloadmöglichkeit böte die Veröffentlichung über die Webseite der Hochschule Aalen. Eine Installation von Applikationen außerhalb der Google eigenen Verkaufsplattform setzt allerdings voraus, dass auf dem Android-Gerät die Installation von Dateien aus *unbekannten Quellen* möglich ist. Diese Option lässt sich in den Sicherheitseinstellungen eines jeden Android-Gerätes aktivieren. Von einer permanenten Aktivierung wird an dieser Stelle eindeutig abgeraten, da dadurch der Schutz vor schädlicher Software verringert wird (Google Support - Nexus 2017).

3 Aufbau der Applikation

3.1 Startbildschirm

Nach dem Installieren der Applikation erscheint diese im App-Menü des Android-Gerätes (Abb. 1).

Abbildung 1 – Applikations-Icon im App-Menü

Die Applikation benötigt keine besonderen Berechtigungen, speichert keinerlei Daten des Benutzers und benötigt ebenfalls keine Internetverbindung um zu funktionieren. Nach dem Starten der Applikation befindet sich der Nutzer auf der Startseite (Abb. 2). Von dieser aus hat er Zugriff auf sämtliche Teilprogramme und die *Informationsseite des Programmes*. Die aufgelisteten Elemente sind

Bestandteil einer *ListView*. Diese lässt sich bei Bedarf erweitern und bietet so die Möglichkeit in späteren Versionen weitere Teilprogramme hinzuzufügen.

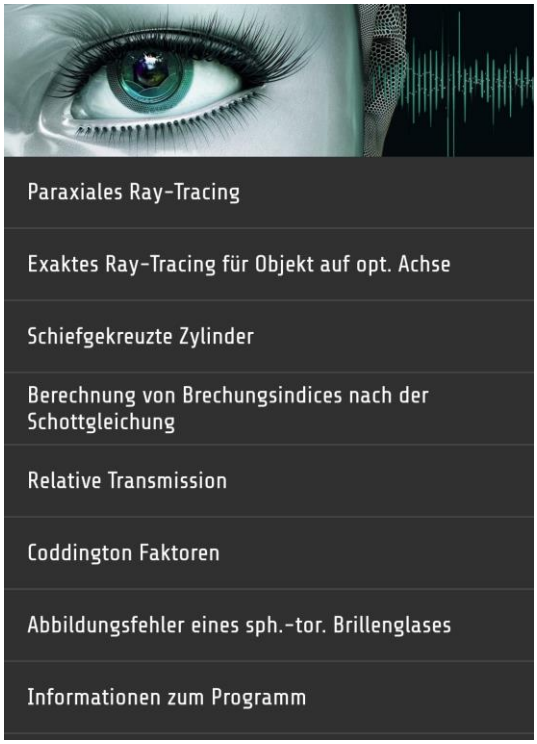

Abbildung 2 – Startseite main_activity.xml

3.2 Paraxiales Ray-Tracing

Das Teilprogramm *Paraxiales Raytracing* ist in insgesamt zwei *Activities* aufgeteilt. Die erste *Activity Raytracing_Start.java* besitzt das Layout *raytracing_start.xml* mit insgesamt vier Objekten (Abb. 3) und dient zur Erstellung des dynamischen Layouts zur weiteren Berechnung. Der Nutzer gibt die Anzahl der Flächen ein die er berechnen möchte und bestätigt seine Eingabe über den Start *Button*. Wird das Feld frei gelassen, so erstellt das Programm im nächsten Schritt automatisch ein Layout, um eine einzelne Fläche zu berechnen. Bedingt durch die maximale Speicherkapazität von Integer Operanden ist die Berechnung auf höchstens 1000 Flächen begrenzt, sollte damit allerdings für alle Aufgabenstellungen in der Optik vollkommen ausreichend sein. Weiterhin ist es möglich, den Objektabstand *s* für ein unendlich weit entferntes Objekt über die *Checkbox* $s_1 = -\infty$ zu aktivieren.

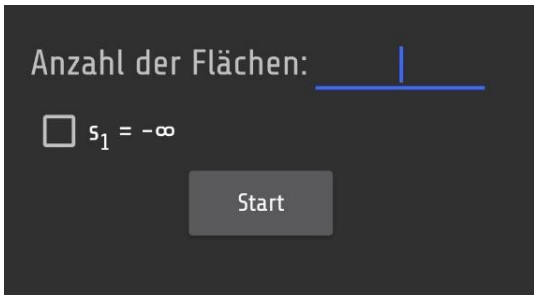

Abbildung 3 – Layout raytraycing_start.xml

Nachdem der Nutzer seine Eingabe über den Start *Button* bestätigt, wird die Anzahl der Flächen mit Hilfe der *Bundle* Funktion in die folgende *Activity Raytracing_Berechnung.java* transferiert und als *datenpaket1* abgespeichert. Der Status der *Checkbox* wird nachfolgend als *datenpaket2* weitergeleitet (Listing 1).

```
// Die Bundle Funktion ermöglicht die Weitergabe der Anzahl der zu berechnenden Flächen an die nächste Activity
// Zudem wird der Status der Checkbox ebenfalls weitergeleitet
Bundle korb = new Bundle();
korb.putString("datenpaket1", enteredText);
Intent in = new Intent(Raytracing_Start.this, Raytracing_Berechnung.class);
in.putExtras(korb);

final CheckBox checkBox = (CheckBox) findViewById(R.id.check_s1);
Bundle korb2 = new Bundle();
if (checkBox.isChecked()) {
korb2.putString("datenpaket2", "1");
} else {
korb2.putString("datenpaket2", "0");
}
in.putExtras(korb2);
startActivity(in);
```

Listing 1 - Ausschnitt aus dem Programmtext der *Raytracing_Start.java*, die den Vorgang der Informationsweitergabe an *Raytracing_Berechnung.java* und die Bundle Funktion beschreibt

Anschließend startet die Activity Raytracing_Berechnung.java und erstellt ein dynamisches Layout, welches abhängig von der Anzahl der Flächen und dem Status der Checkbox ist. Im folgenden Beispiel ist das Layout für zwei Flächen und manueller Eingabe des Objektabstands s_1 zu sehen (Abb. 4). Nach dem Ausfüllen aller *Edittext-Felder* bestätigt der Nutzer seine Eingabe über den *Berechnen-Button*. Die Felder erlauben lediglich die Eingabe von positiven oder negativen Dezimalzahlen. Sollte eines der Felder nicht ausgefüllt worden sein, so weist ein Warnhinweis darauf hin (Abb. 5).

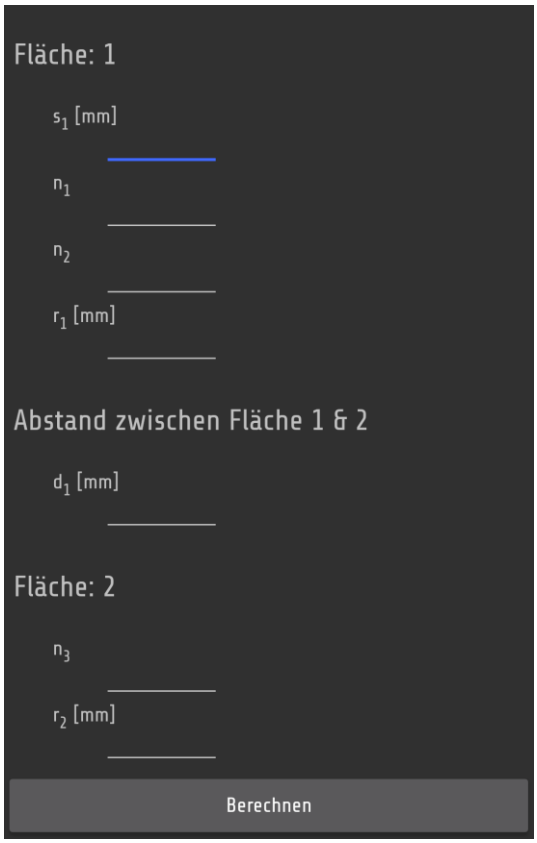

Abbildung 4 – Layout raytracing_berechnung.xml für den speziellen Fall für zwei Flächen und manueller Eingabe des Objektabstands s_1

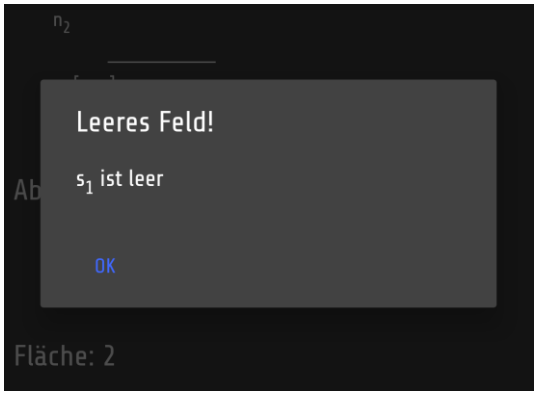

Abbildung 5 – Warnhinweis, der auf ein unausgefülltes Feld – in diesem Fall das Feld s_1 – hinweist

Das Programm führt nun eigenständig die notwendigen Rechenschritte zur Berechnung der bildseitigen Schnittweite für die letzte Fläche durch. Dafür wird jeweils die Abbildungsgleichung (1) für Einzelflächen verwendet:

$$\frac{n_{m+1}}{s'_m} = \frac{n_m}{s_m} + \frac{n_{m+1} - n_m}{r_m} \quad (1)$$

mit: n_m **Brechungsindex vor der Fläche**
n_{m+1} **Brechungsindex nach der Fläche**
s_m **Schnittweite des Gegenstands**
s'_m **Schnittweite des Bildes**
r_m **Radius der Fläche**

Mit Hilfe der berechneten Bildweite lässt sich nun über eine Transfergleichung (2) die Objektweite der folgenden Fläche berechnen:

$$s_{m+1} = s'_m - d_m \quad (2)$$

mit: d_m **Abstand zwischen den Flächen**

Das Programm führt diese Rechenschritte automatisiert über eine Schleifenfunktion bis zur letzten Fläche durch. Wurde manuell ein Objektabstand eingegeben, so berechnet das Programm zudem die Lateralvergrößerung β (3):

$$\beta = \frac{n_1}{n_{m+1}} \cdot \frac{s'_1 \cdot s'_2 \cdot \ldots \cdot s'_m}{s_1 \cdot s_2 \cdot \ldots \cdot s_m} \quad (3)$$

die Tiefenvergrößerung β_t (4):

$$\beta_t = \frac{n_1}{n_{m+1}} \cdot \beta^2 \quad (4)$$

sowie die Winkelvergrößerung β_w (5):

$$\beta_w = \frac{n_1}{n_{m+1}} \cdot \frac{1}{\beta} \quad (5)$$

Das Ergebnis wird anschließend unter dem *Berechnen-Button* sichtbar (Abb. 6).

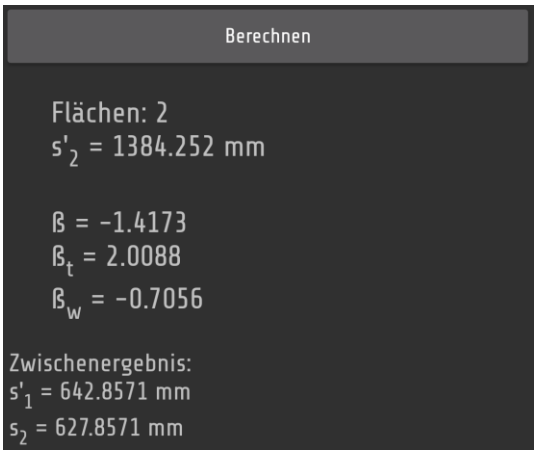

Abbildung 6 – Ergebnisse der Berechnung für insgesamt zwei Flächen mit manueller Angabe der objektseitigen Schnittweite $s_1 = -1000\,mm$, den Brechungsindices $n_1 = 1$, $n_2 = 1.5$ und $n_3 = 1$, den Radien $r_1 = 150\,mm$ und $r_2 = 300$ sowie dem Abstand $d_1 = 15\,mm$ zwischen den beiden Flächen

Wurde für $s_1 = -\infty$ gewählt, so berechnet das Programm neben der endgültigen bildseitigen Schnittweite s'_m zusätzlich die bildseitige Brennweite f' (6):

$$f' = \frac{s'_1 \cdot s'_2 \cdot \ldots \cdot s'_{m-1}}{s_2 \cdot s_3 \cdot \ldots \cdot s_m} \cdot s'_m \qquad (6)$$

die Gesamtwirkung aller Flächen D_{Ges} (7):

$$D_{Ges} = \frac{n_{m+1}}{f'} \qquad (7)$$

und die Lage des bildseitigen Hauptpunktes $s'_{H'}$ (8):

$$s'_{H'} = s'_m - f' \qquad (8)$$

Anschließend wird das System gespiegelt, um die objektseitige Schnittweite s_f (Vgl. Formel 1), die objektseitige Brennweite f (Vgl. 6), die Lage des objektseitigen Hauptpunktes s_H (Vgl. Formel 8), sowie die Knotenpunkte k und k' (9) zu berechnen (Abb. 7):

$$k = k' = f + f' \tag{9}$$

Bei dem gespiegelten System ändern sich die Reihenfolge der Brechungsindices n_m und die Vorzeichen der Flächenradien r_m . Da jedes erzeugte Layout-Element im Vorfeld eine individuelle *ID* erhält, lässt sich mit Hilfe dieser die Reihenfolge gezielt anpassen (Listing 2).

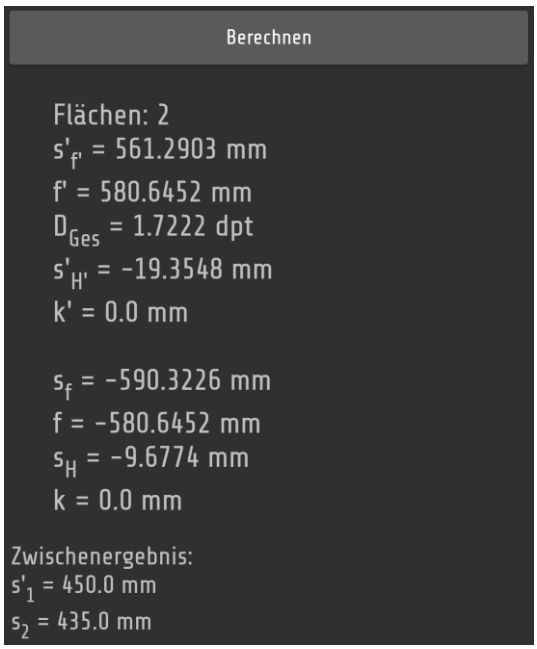

Abbildung 7 - Ergebnisse der Berechnung für insgesamt zwei Flächen mit dem Objektabstand $s_1 = -\infty$, den Brechungsindices $n_1 = 1$, $n_2 = 1.5$ und $n_3 = 1$, den Radien $r_1 = 150\,mm$ und $r_2 = 300$, sowie dem Abstand $d_1 = 15\,mm$ zwischen den beiden Flächen

```
// für den fall, dass der objektabstand -unendlich war, berechne die lage
// der hauptpunkte, die knotenpunkte, sowie f und f'
if (cb_s1 == 1 && fn > 1) {
      absolut = fn*5-1;
      //um die objektseitige brennweite f, k und die lage des objektseitigen
      // hauptpunktes zu ermitteln, muss rückwärts durch das system gerechnet werden
      for (int k = fn; k > 0; k--) {
            if (k == fn) {
                  z4 = (edittext) findviewbyid(absolut);
                  z_r1 = double.parsedouble(z4.gettext().tostring());
                  z_r1 = -z_r1;
                  absolut = absolut - 1;

                  z3 = (edittext) findviewbyid(absolut);
                  z_n1 = double.parsedouble(z3.gettext().tostring());
                  absolut = absolut - 3;

                  z1 = (edittext) findviewbyid(absolut);
                  z_d = double.parsedouble(z1.gettext().tostring());
                  absolut = absolut - 2;

                  z2 = (edittext) findviewbyid(absolut);
                  z_n2 = double.parsedouble(z2.gettext().tostring());
                  absolut = absolut + 1;

                  z_s_1 = ((z_n2-z_n1)/z_r1)/z_n2;
                  z_s_1=1/z_s_1;
                  z_b = z_n2;
                  z_a = z_s_1-z_d;
                  f = z_s_1/z_a;

            } else if (k > 1) {
                  z4 = (edittext) findviewbyid(absolut);
                  z_r1 = double.parsedouble(z4.gettext().tostring());
                  z_r1 = -z_r1;
                  absolut = absolut - 4;

                  z1 = (edittext) findviewbyid(absolut);
                  z_d = double.parsedouble(z1.gettext().tostring());
                  absolut = absolut - 2;

                  z2 = (edittext) findviewbyid(absolut);
                  z_n2 = double.parsedouble(z2.gettext().tostring());
                  absolut = absolut + 1;

                  z_s_1 = ((z_b/z_a)+((z_n2-z_b)/z_r1))/z_n2;
                  z_s_1 = 1 / z_s_1;
                  z_b = z_n2;
                  z_a = z_s_1 - z_d;
                  f = f * (z_s_1/z_a);

            } else if (k == 1) {
                  z4 = (edittext) findviewbyid(absolut);
                  z_r1 = double.parsedouble(z4.gettext().tostring());
                  z_r1 = -z_r1;
                  absolut = absolut - 2;

                  z2 = (edittext) findviewbyid(absolut);
                  z_n2 = double.parsedouble(z2.gettext().tostring());

                  z_s_1 = ((z_b/z_a)+((z_n2-z_b)/z_r1))/z_n2;
                  z_s_1 = 1 / z_s_1;
                  f = f * z_s_1;
                  z_s_1 = -z_s_1;
                  f = -f;
                  sh = z_s_1 - f;

                  knoten = f + f_1;
                  knoten_1 = f_1 + f;

                  vnm1 = (edittext) findviewbyid(((fn-1)*5)+3);
                  vgr_nm1 = double.parsedouble(vnm1.gettext().tostring());
                  d_ges = (vgr_nm1/(f_1/1000));
            }
      }
}
```

Listing 2 – Ausschnitt aus dem Programmtext *Raytracing_Berechnung.java*, der für die Spiegelung des Systems verantwortlich ist

3.3 Exaktes Ray-Tracing für ein Objekt auf der optischen Achse

Wie auch das *Paraxiale Raytracing*, ist dieses Teilprogramm in insgesamt zwei *Activities* unterteilt. Zu Beginn startet die *Activity Exaktes_Raytracing_Start.java* und fordert den Nutzer auf, die Anzahl der zu berechnenden Flächen einzugeben (Abb. 8).

Abbildung 8 – Layout exaktes_raytracing_start.xml

Nach der Bestätigung über den *Start-Button* wird der Nutzer an die *Activity Exaktes_Raytracing_Berechnung.java* weitergeleitet, in dem mit Hilfe der vorigen Eingabe dynamisch das Layout zur Berechnung erstellt wird. Zusätzlich zu den auch für das *Paraxiale Raytracing* erforderlichen Größen (Vgl. Abb. 5) wird nun ebenfalls der Winkel zwischen einfallendem Lichtstrahl und optischer Achse u_m benötigt (Abb. 9).

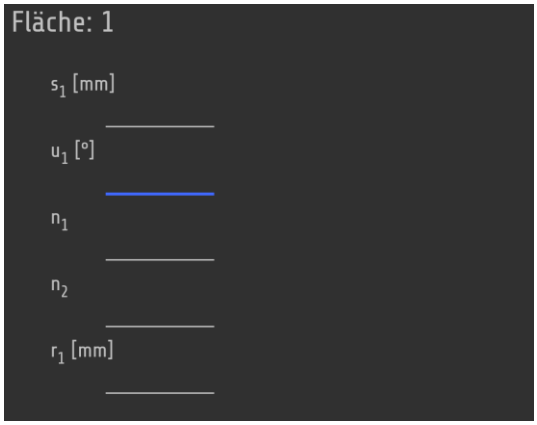

Abbildung 9 – Ausschnitt aus dem Layout exaktes_raytracing_berechnung.xml

Der grundsätzliche Aufbau des Programms ist dem des *Paraxialen Raytracings* sehr ähnlich. Die Darstellung des Layouts entspricht dem Fall mit manueller Eingabe des Objektabstandes s_m (Vgl. Abb. 6).

Neben der Berechnung der bildseitigen Schnittweite s'_m (10 bis 13), wird ebenfalls die Lateralvergrößerung (3), die Tiefenvergrößerung (4) und die Winkelvergrößerung (5) ermittelt.

$$sin\ \varepsilon_m = \left(\frac{s_m}{r_m} - 1\right) \cdot \sin u_m \qquad (10)$$

mit: $\boldsymbol{u_m}$ **Winkel zwischen dem einfallenden Lichtstrahl und der optischen Achse**

$\boldsymbol{\varepsilon_m}$ **Winkel des einfallenden Lichtstrahls zum Lot der brechenden Fläche**

$$sin\ \varepsilon'_m = \frac{n_m}{n_{m+1}} \cdot \sin \varepsilon_m \qquad (11)$$

mit: $\boldsymbol{\varepsilon'_m}$ **Winkel des ausfallenden Lichtstrahls zum Lot der brechenden Fläche**

$$u'_m = u_m + \varepsilon_m - \varepsilon'_m \qquad (12)$$

$$s'_m = r_m \cdot \left(1 + \frac{\sin \varepsilon'_m}{\sin u'_m}\right) \qquad (13)$$

Wie bereits beim *Paraxialen Raytracing* werden diese Rechenschritte in einer Schleife bis zur letzten Fläche wiederholt. Für die jeweils folgende Fläche wird erneut die Transfergleichung (Formel 2) benötigt; zusätzlich gilt $u'_m = u_{m+1}$. Die Darstellung der Ergebnisse ähnelt ebenfalls der des *Paraxialen Raytracings*. Hinzugekommen ist lediglich der jeweilige Winkel zwischen dem ausfallenden Lichtstrahl und der optischen Achse u'_m (Abb. 10).

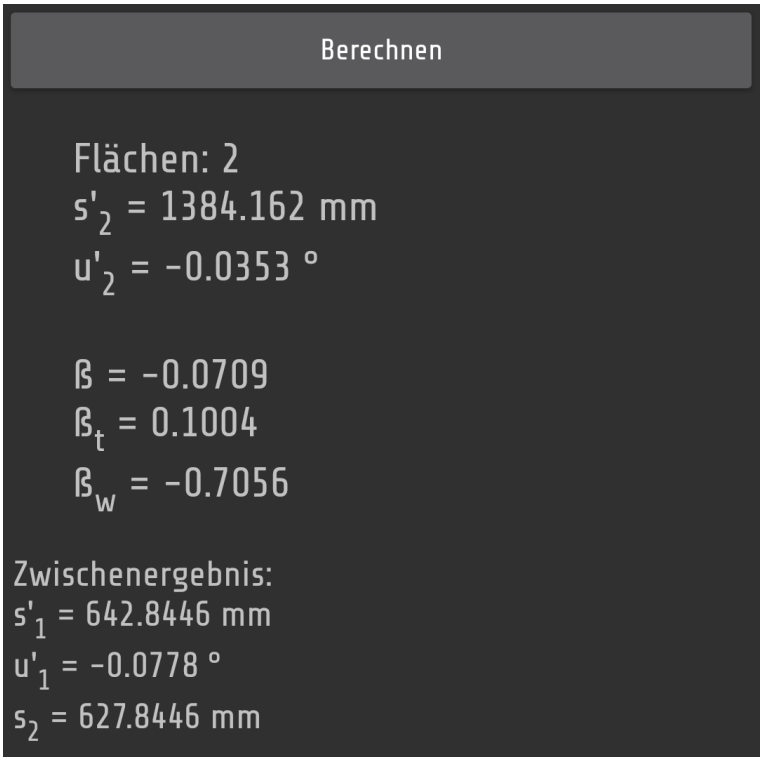

Abbildung 10 - Ergebnisse der Berechnung für insgesamt zwei Flächen mit der objektseitigen Schnittweite $s_1 = -1000\,mm$, dem Winkel zwischen dem einfallenden Lichtstrahl und der optischen Achse u_1, den Brechungsindices $n_1 = 1$, $n_2 = 1.5$ und $n_3 = 1$, den Radien $r_1 = 150\,mm$ und $r_2 = 300$, sowie dem Abstand $d_1 = 15\,mm$ zwischen den beiden Flächen

3.4 Schiefgekreuzte Zylinder

Das Unterprogramm *Schiefgekreuzte Zylinder* berechnet die Kombination von Sphäre und Zylinder zweier Gläser sowie die resultierende Achslage. Dafür werden Sphäre, Zylinder und Achse in Powervektoren umgerechnet, miteinander addiert und anschließend wieder zurücktransferiert (14 bis 20).

$$I_0 = -\frac{Zyl}{2} \cdot \cos 2\alpha \qquad (14)$$

mit: $\boldsymbol{Zyl}$ **Zylinder**

α **Achse**

$$I_{45} = -\frac{Zyl}{2} \cdot \sin 2\alpha \qquad (15)$$

$$M = Sph + \frac{Zyl}{2} \qquad (16)$$

mit: $\boldsymbol{Sph}$ **Sphäre**

$$\begin{pmatrix} I_{0_1} \\ I_{45_1} \\ M_1 \end{pmatrix} + \begin{pmatrix} I_{0_2} \\ I_{45_2} \\ M_2 \end{pmatrix} = \begin{pmatrix} I_{0_3} \\ I_{45_3} \\ M_3 \end{pmatrix} \qquad (17)$$

$$Zyl = 2\sqrt{I_0^2 + I_{45}^2} \qquad (18)$$

$$Sph = M - \frac{Zyl}{2} \qquad (19)$$

$$\alpha = \frac{1}{2} \cdot \arctan\frac{I_{45}}{I_0} + \Delta \quad | \quad I_0 > 0 \rightarrow \Delta \pm 90° \qquad (20)$$

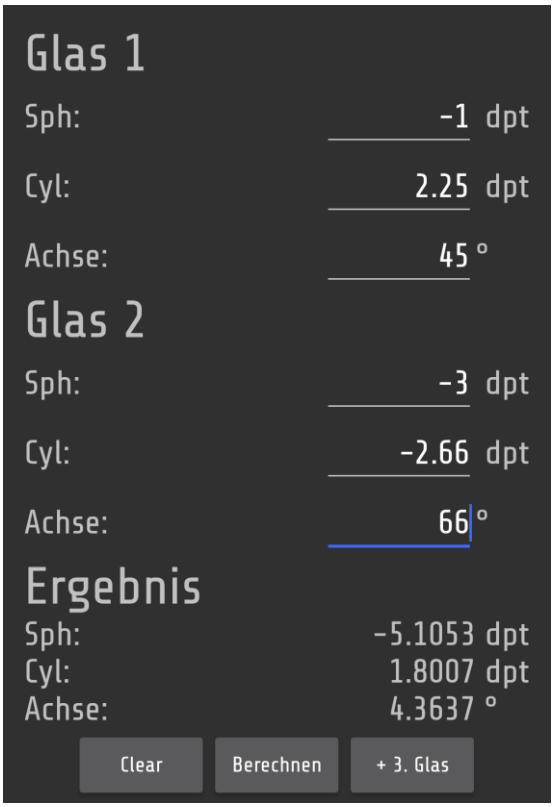

Abbildung 11 – Layout *powervektoren.xml* mit dem Ergebnis für eine sphäro-zylindrische Kombination

Das Teilprogramm besteht aus der *Activity Powervektoren.java* und dem zugehörigen Layout *powervektoren.xml* (Abb. 11). Der Nutzer gibt die Werte beider Gläser in die dafür vorgesehenen Felder ein und bestätigt anschließend seine Eingabe über den *Berechnen-Button*. Ob er dabei die Werte für den Zylinder in positiver oder negativer Schreibweise verwendet, hat keinerlei Auswirkungen auf das Ergebnis. Die Eingabefelder für die Achslage sind durch einen Filter auf Dezimalzahlen zwischen 0° und 179° beschränkt (Listing 3). Die Grundstruktur dieser Funktion stammt aus einem Forenarchiv (Sharma 2013).

Mit Hilfe des *Clear-Buttons* lassen sich die Einträge in den Feldern löschen. Möchte der Nutzer mehr als zwei Gläser berechnen, so kann durch den *+Glas-Button* das bisherige Ergebnis in die Felder für Glas 1 geladen werden (Abb. 11 & 12).

```
// Erstellt einen Filter der die Eingabe der Achsen auf 0-179° beschränkt
public class InputFilterMinMax implements InputFilter {
    private int min;
    private int max;

    public InputFilterMinMax(int min, int max) {
        this.min = min;
        this.max = max;
    }

    @Override
    public CharSequence filter(CharSequence source, int start, int end, Spanned dest,
                               int dstart, int dend) {

        try {
            double input = Double.parseDouble(dest.subSequence(0, dstart).toString() +
                    source + dest.subSequence(dend, dest.length()));
            if (isInRange(min, max, input))
                return null;
        } catch (NumberFormatException nfe) { }
        return "";
    }

    private boolean isInRange(double a, double b, double c) {
        return b > a ? c >= a && c <= b : c >= b && c <= a;
    }
}
```

Listing 3 – Ausschnitt aus dem Programmcode *Powervektoren.java*, die den Filter zeigt, der die Eingabe der Achsen auf Werte zwischen 0° und 179° beschränkt

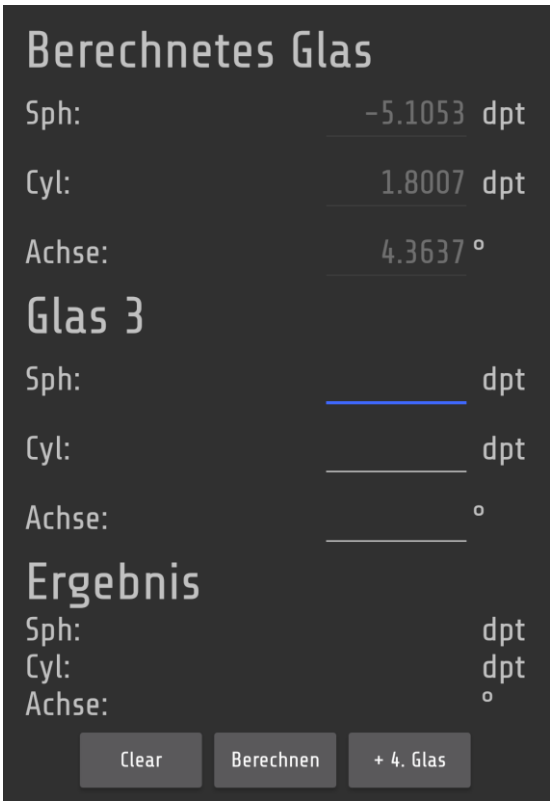

Abbildung 12 – Durch Betätigen des *+Glas-Buttons* unten rechts lassen sich die Ergebnisse in die Berechnungsfelder laden und ermöglicht somit das Hinzufügen weiterer Gläser (Vgl. Abb. 11)

3.5 Berechnung von Brechungsindices nach der Schottgleichung

Die *Activity Schott.java* dient zur Berechnung der Brechungsindices nach der Schottgleichung und bietet neben der eigenen Angabe von Wellenlängen und Glasmaterialien auch eine bereits abgespeicherte Auswahl an. Bei den bereits enthaltenen Materialien handelt es sich um die im Studiengang zur Berechnung verwendeten Glasarten BaK 4, BK 7, K 5 und SF 10 bei denen die Konstanten A_0 bis A_5 der Dispersionsformel bereits abgespeichert sind (Abb. 13 & 14).

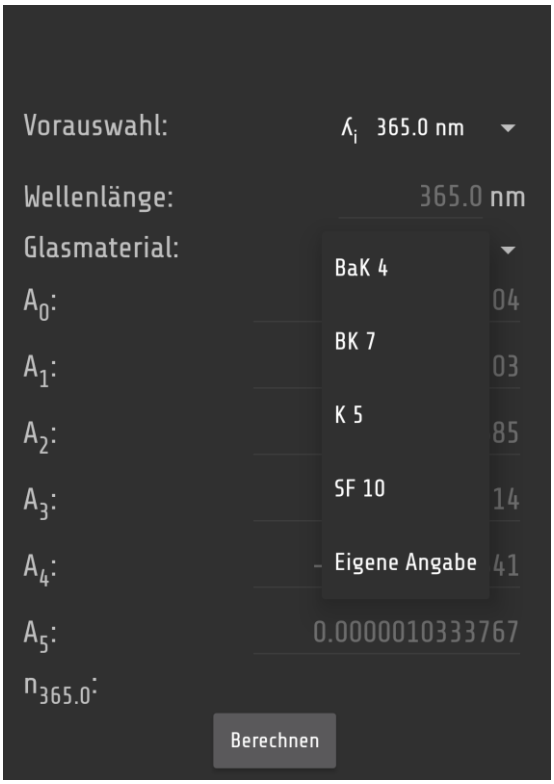

Abbildung 13 – Layout schott.xml mit der geöffneten Auswahl der bereits vorinstallierten Glasmaterialien

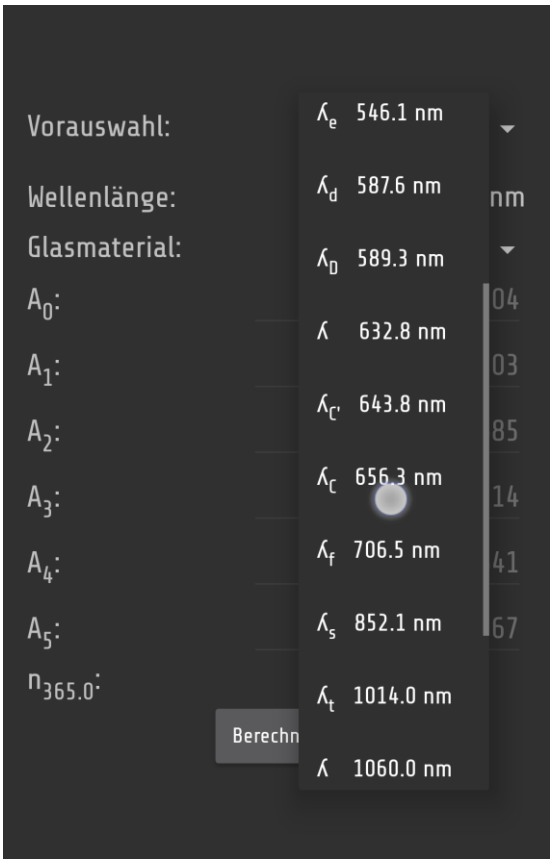

Abbildung 14 – Layout schott.xml mit einem Ausschnitt der geöffneten Auswahl der bereits vorinstallierten Wellenlängen

Nachdem der Nutzer seine Eingabe über den *Berechnen-Button* bestätigt, berechnet das Programm mit Hilfe der Schottgleichung (21) den Brechungsindex:

$$n = \sqrt{A_0 + A_1 \cdot \lambda^2 + \frac{A_2}{\lambda^2} + \frac{A_3}{\lambda^4} + \frac{A_4}{\lambda^6} + \frac{A_5}{\lambda^8}} \qquad (21)$$

mit: $\boldsymbol{\lambda}$ **Wellenlänge in µm**

$\boldsymbol{A_0 - A_5}$ **Konstanten der Dispersionsformel**

3.6 Relative Transmission

Die Berechnung der *Relativen Transmission* geschieht erneut mit Hilfe eines dynamischen Layouts, welches dem des *Paraxialen Raytracings* sehr ähnlich ist. Nach dem Öffnen des Teilprogramms startet zunächst die *Activity Transmission_Start.java* (Abb. 15).

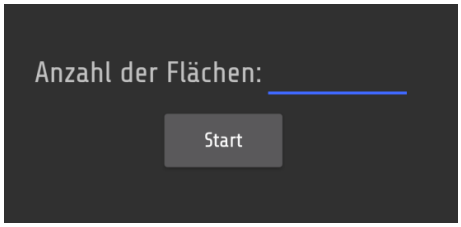

Abbildung 15 – Layout transmission_start.xml

Nach der Angabe der Anzahl der zu berechnenden Flächen erstellt anschließend die *Activity Transsmision_Berechnung.java* selbstständig das dafür notwendige Layout (Abb. 16).

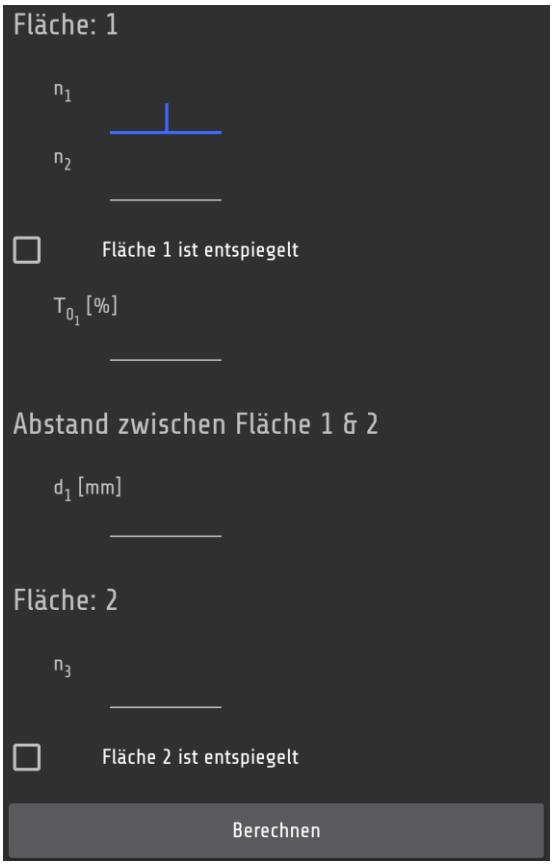

Abbildung 16 – Layout transmission_berechnung.xml für zwei Flächen

Für entspiegelte Flächen hat der Nutzer die Option, die jeweiligen *Checkboxen* zu bestätigen. Nachdem der *Berechnen-Button* betätigt wurde, fragt das Programm automatisch alle *Checkboxen* ab und erstellt gegebenenfalls die Option, den Restreflex der entspiegelten Fläche anzugeben und weist zusätzlich auf die Eingabemethode hin (Abb. 17 & 18).

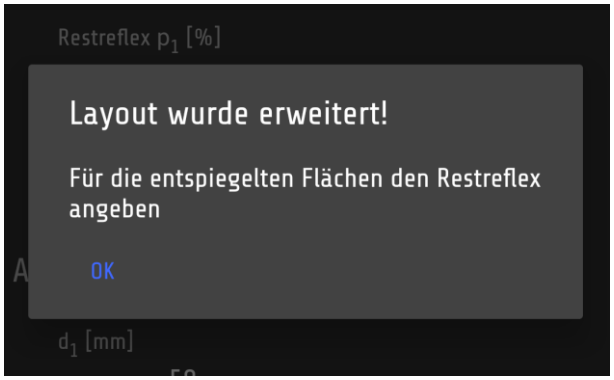

Abbildung 17 – Hinweisfenster nach Betätigen des *Berechnen-Buttons* und mindestens einer bestätigten *Checkbox*

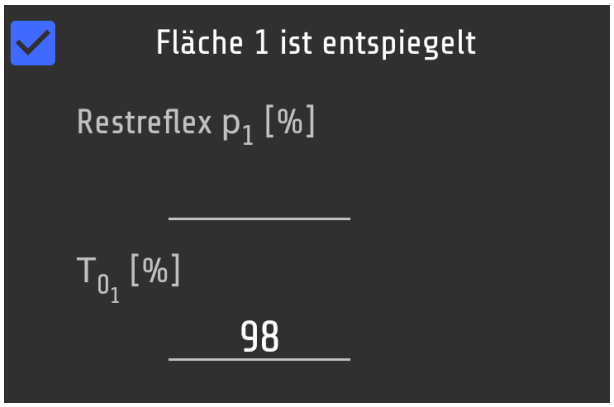

Abbildung 18 – Hinzugefügte Option zur Eingabe des Restreflexes p_1 für die Fläche 1

Dies geschieht durch den folgenden Abschnitt im Programmcode der *Transmission_Berechnung.java* (Listing 4):

```
bt.setOnClickListener (new View.OnClickListener() {
    public void onClick(View v) {

        int k = 1;
        //Überprüfe für alle erstellten Checkboxen, ob der Haken für eine entspiegelte
        // Fläche gesetzt wurde und füge gegebenfalls das entsprechende Eingabefenster
        // hinzu um den prozentualen Restreflex zu setzen
        if (cb_check == 0) {
            for (int y = 1; y <= fn; y++) {

                CB = (CheckBox) findViewById(k + 1);
                if (CB.isChecked()) {
                    B3 = (EditText) findViewById(k + 4);
                    B3.setVisibility(View.VISIBLE);
                    B3_Text = (TextView) findViewById(k);
                    B3_Text.setVisibility(View.VISIBLE);
                    cb_check = 1;
                } else {
                    CB.setEnabled(false);
                    B3 = (EditText) findViewById(k+4);
                    B3.setText("1");
                }
                k = k + 7;
            }
        }

        k = 1;

        // Sollte der Haken bei einer Checkbox gesetzt worden sein, dann weise den
        // Nutzer darauf hin, dass er nun den prozentualen Restreflex eingeben kann
            if (cb_check == 1) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Layout wurde erweitert!");
                AlarmBox.setMessage(Html.fromHtml("Für die entspiegelten Flächen " +
                        "den Restreflex angeben"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                cb_check = 2;
                return;

            } else if (cb_check == 0) {
```

Listing 4 – Ausschnitt aus dem Programmcode *Transmission_Berechnung.java*

Wurde keine *Checkbox* ausgewählt, so wird dieser Schritt einfach übersprungen und die Berechnung startet. In diesem Fall wird der Reflexionsgrad der Fläche p_m über die Brechungsindices berechnet (22):

$$p_m = \left(\frac{n_m - n_{m+1}}{n_m + n_{m+1}}\right)^2 \qquad (22)$$

Die Transmission durch das Material T_m wird mit Hilfe der Reintransmission und dem Abstand zwischen den Flächen ermittelt (23):

$$T_m = T_0^{\frac{d_m}{d_0}} \quad (23)$$

mit: T_0 **Reintransmission**

d_m **Abstand zwischen den Flächen**

d_0 **Referenzabstand der Reintransmission von 10 mm**

Die Transmission nach der letzten Fläche T_{Ges} ergibt sich nun aus der Formel (24):

$$T_{Ges} = T_1 \cdot (1 - \mathrm{p}_1) \cdot T_2 \cdot (1 - \mathrm{p}_2) \cdot \ldots \cdot T_m \cdot (1 - \mathrm{p}_m) \quad (24)$$

Neben der relativen Transmission nach der letzten Fläche werden zusätzlich auch die Reflektion und die Transmission innerhalb jeder einzelnen Fläche als Zwischenergebnis angezeigt (Abb. 19).

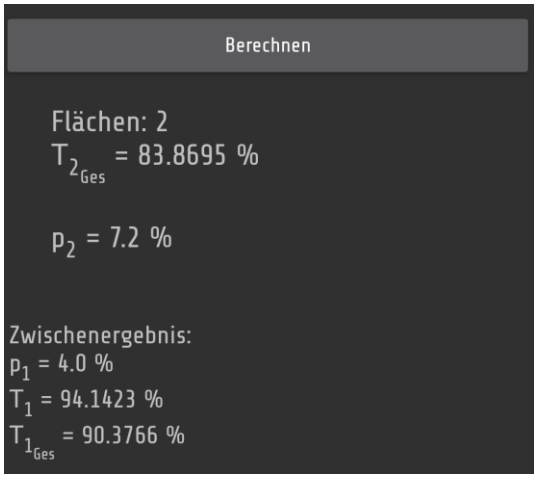

Abbildung 19 – Ergebnisse der relativen Transmission durch zwei Flächen mit den Brechungsindices $n_1 = 1$, $n_2 = 1{,}5$, $n_3 = 1$, der Reintransmission $T_{0_1} = 98{,}8\,\%$ zwischen den Flächen, dem Abstand $d_1 = 50\,mm$ zwischen den Flächen und einer entspiegelten Fläche 2 mit Restreflex $\mathrm{p}_2 = 7{,}2\,\%$

3.7 Coddington-Faktoren

Der Programmabschnitt *Coddington-Faktoren* dient zur Berechnung der Formfaktoren für komafreie Linsen und Linsen mit minimaler sphärischer Aberration. Diese Berechnung gilt ausschließlich für dünne Gläser an Luft. Das Programm selbst besteht aus der *Activity Shape.java* und dem zugehörigen Layout *shape.xml.* Nach dem Starten des Programms gibt der Nutzer den Brechungsindex *n* des Glasmaterials an, wählt im Falle eines unendlich weit entfernten Objektes eine der beiden *Checkboxen* oder gibt manuell den Objektabstand *s* an (Abb. 20). Zusätzlich hat er die Option, statt des Objektabstands die bildseitige Schnittweite *s‘* einzutragen (Abb.21).

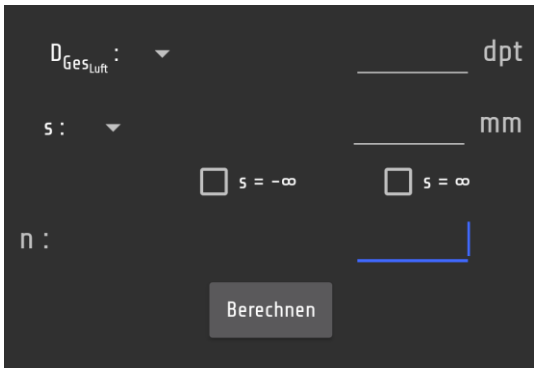

Abbildung 20 – Layout *shape.xml*

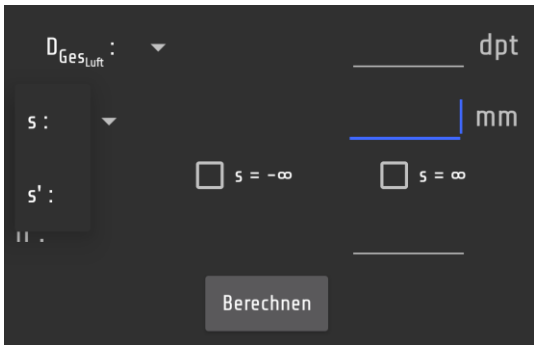

Abbildung 21 – Geöffneter *Spinner* mit der Auswahl zwischem dem Objektabstand *s* und der bildseitigen Schnittweite *s‘*

Der Nutzer hat ebenfalls die Wahl, ob er den Gesamtbrechwert $D_{Ges_{Luft}}$ oder die bildseitige Brennweite *f'* angeben möchte (Abb. 22).

Abbildung 22 – Geöffneter *Spinner* mit der Auswahl zwischen dem Gesamtbrechwert $D_{Ges_{Luft}}$ und der bildseitigen Brennweite *f'*

Befindet sich ein Wert im *Edittext-Feld* und der Nutzer ändert seine Auswahl, so rechnet das Programm automatisch um und setzt den neuen Wert in das *Edittext-Feld* ein (Abb. 23).

Abbildung 23 – Das Programm rechnet selbstständig zwischen dem Gesamtbrechwert $D_{Ges_{Luft}}$ und der bildseitigen Brennweite *f'* um

Die Umrechnung selbst geschieht mit Hilfe der *Event-Abfrage* des *Spinners,* welcher beim Wechsel zwischen dem Gesamtbrechwert und der bildseitigen Brennweite ein Signal aussendet (Listing 5).

Das Ergebnis besteht aus dem Positions-Faktor *P*, den Shape-Faktoren $S_{komafrei}$ und S_{opt} sowie den jeweils zugehörigen Radien r_1 und r_2 des Glases (Abb. 24).

Der Positions-Faktor *P* wird, je nachdem ob der Objektabstand *s* (25) oder die bildseitige Schnittweite *s'* (26) gegeben ist, auf unterschiedliche Weise ermittelt:

$$P = -\left(\frac{2 \cdot f'}{s} + 1\right) \quad (25)$$

$$P = 1 - \frac{2 \cdot f'}{s'} \quad (26)$$

Anschließend wird der Shape-Faktor $S_{komafrei}$ (27) für die komafreie Linse und der Shape-Faktor S_{opt} (28) für die Linse mit minimaler sphärischer Aberration berechnet:

$$S_{komafrei} = -\left(\frac{2n^2 - n - 1}{n + 1}\right) \cdot P \quad (27)$$

$$S_{opt} = -\frac{2 \cdot (n^2 - 1)}{n + 2} \cdot P \quad (28)$$

Mit Hilfe dieser beider Faktoren lassen sich nun die Radien r_1 (29) und r_2 (30) des jeweiligen Glases ermitteln:

$$r_1 = \frac{2 \cdot f' \cdot (n - 1)}{S_{opt/komafrei} + 1} \quad (29)$$

$$r_2 = \frac{2 \cdot f' \cdot (n - 1)}{S_{opt/komafrei} - 1} \quad (30)$$

```
shape_D = (Spinner) findViewById(R.id.spinner_shape_D);
adapter = ArrayAdapter.createFromResource(this, R.array.spinner_shape_D,
        android.R.layout.simple_list_item_1);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
shape_D.setAdapter(adapter);
shape_D.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {


        if (position == 0) {
            txt_shape_D_mm.setText(" dpt");
            D_auswahl = "D<sub><small>Ges<sub><small><small>Luft</small>" +
                    "</small></sub></small></sub>";
            String Alarm = et_DGes.getText().toString();
            if (Alarm.equals("") || Alarm.equals("0")) {
            } else {
                Umrechner = Double.parseDouble(et_DGes.getText().toString());
                Umrechner = 1 / (Umrechner / 1000);
                et_DGes.setText(Double.toString(Math.round(Umrechner * 10000) / 10000.0));
            }
        } else {
            txt_shape_D_mm.setText(" mm");
            D_auswahl = "f'";
            String Alarm = et_DGes.getText().toString();
            if (Alarm.equals("") || Alarm.equals("0")) {
            } else {
                Umrechner = Double.parseDouble(et_DGes.getText().toString());
                Umrechner = (1 / Umrechner) * 1000;
                et_DGes.setText(Double.toString(Math.round(Umrechner * 10000) / 10000.0));
            }
        }

    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

Listing 5 – Ausschnitt aus dem Programmcode *Shape.java*, der für die Umrechnung zwischen dem Gesamtbrechwert und der bildseitigen Brennweite verantwortlich ist

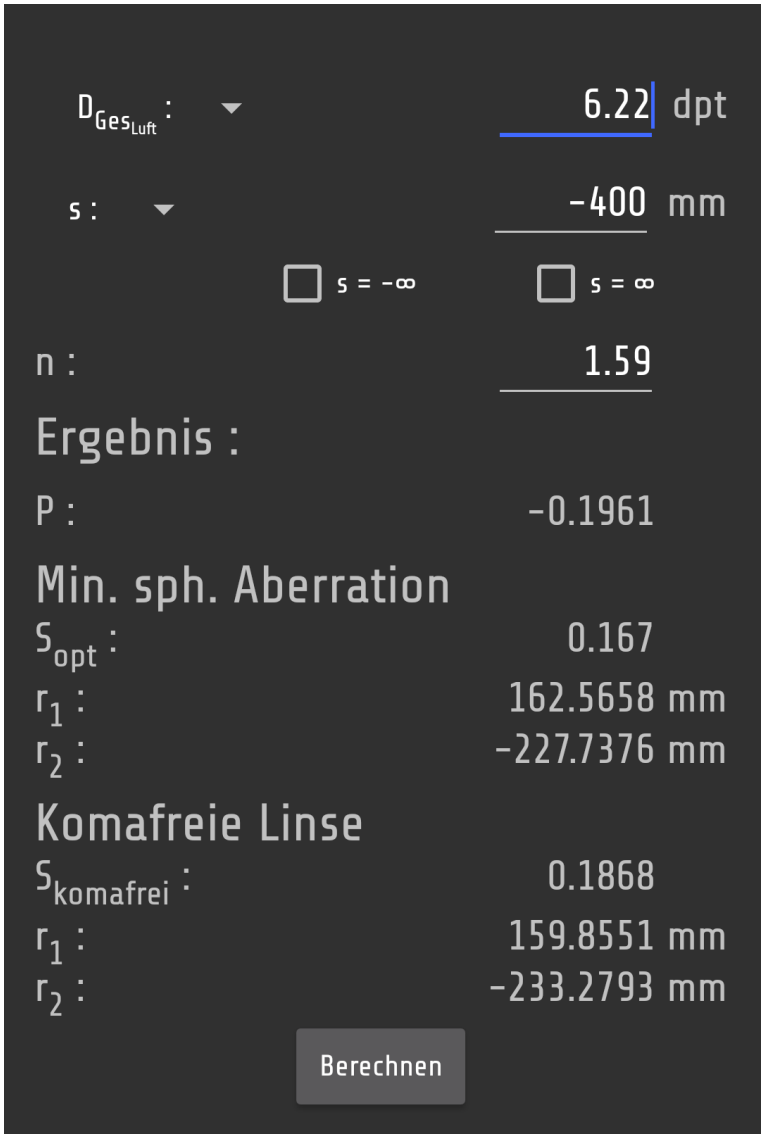

Abbildung 24 – Ergebnisse für ein Glas mit der Gesamtbrechkraft von 6,22 dpt, einem Objektabstand von 400 mm vor dem Glas und einem Brechungsindex von 1,59

3.8 Abbildungsfehler eines sph.-tor. Brillenglases

Das letzte der insgesamt sieben Teilprogramme der Applikation besteht aus dem Programmcode *Glasberechnung.java* und dem zugehörigen Layout *glasberechnung.xml.* Alle verwendeten Formeln stammen aus dem Vorlesungsskript des Wahlpflichfachs *OTB 2 – Projekt Einstärkenglasentwicklung* (Baumbach 2016). Neben dem Refraktionsfehler, Astigmatismus und der Verzeichnung lassen sich ebenfalls das Gewicht, die Bauhöhe und das Volumen des resultierenden sphäro-torischen Glases berechnen. Zusätzlich lassen sich die Radien im Werkzeugindex von $n = 1{,}525$ anzeigen und eine Rundung der Radien der Rückfläche in 0,0625 dpt Schritten ist möglich. Der maximal zu berechnende Blickwinkel und die Schrittweite lassen sich ebenfalls individuell anpassen (Abb. 25).

Abbildung 25 – Layout *glasberechnung.xml*

Sollte mindestens einer der beiden Hauptschnitte positiv oder ohne Wirkung sein, so ist die Angabe einer minimalen Randdicke d_r notwendig. Für zwei negative Hauptschnitte wird eine minimale Mittendicke d benötigt. In beiden Fällen weist das Programm automatisch darauf hin, sollte das benötigte Feld freigelassen worden sein. Um die resultierende Mittendicke bei vorhandener minimaler Randdicke zu berechnen, wird ein iteratives Verfahren angewendet. Zunächst werden die durch den Nutzer angegebenen Werte angenommen und die Flächenbrechkräfte $F_{2/3}$ berechnet (31):

$$F_{2/3} = S'_{\infty_{1/2}} - \frac{F_1}{1 - \frac{d}{n} \cdot F_1} \qquad (31)$$

mit: F_1 **Flächenbrechkraft der Vorderfläche**
F_2 **Flächenbrechkraft der Rückfläche in Hauptschnitt 1**
F_3 **Flächenbrechkraft der Rückfläche in Hauptschnitt 2**
$S'_{\infty_{1/2}}$ **Scheitelbrechwert im zugehörigen Hauptschnitt**

Wurde keine Basiskurve F_1 durch den Nutzer angegeben, so setzt das Programm automatisch einen eigenen Wert in das zugehörige *Edittext-Feld.* Für Rohgläser mit zwei negativen Hauptschnitten wird standardmäßig eine Basiskurve von 4 dpt angenommen. Sollte einer der beiden Hauptschnitte allerdings positiv sein, so gilt (32):

$$F_1 = S'_{\infty_{1/2}} + 1\ dpt \qquad (32)$$

Sind beide Hauptschnitte positiv, so wird der Scheitelbrechwert des stärker positiven verwendet.

Anschließend werden die Radien $r_{1/2/3}$ des Glases bestimmt (33):

$$r_{1/2/3} = \frac{n_{m+1} - n_m}{F_{1/2/3}} \qquad (33)$$

Mit Hilfe der Scheiteltiefen $t_{1/2/3}$ (34) lassen sich nun die Randdicken $d_{r_{2/3}}$ (35) ermitteln:

$$t_{1/2/3} = r_{1/2/3} \cdot \left(1 - \sqrt{1 - \frac{\left(\frac{Ø}{2}\right)^2}{{r_{1/2/3}}^2}} \right) \quad (34)$$

mit: Ø **Durchmesser des Rohglases**

$$d_{r_{2/3}} = d + t_{2/3} - t_1 \quad (35)$$

Nun wird die errechnete Randdicke von der gewollten minimalen Randdicke $d_{r_{min}}$ abgezogen. Ist das Ergebnis positiv, so wird es von der bisherigen Mittendicke d_{alt} abgezogen; sollte es negativ sein, so wird es addiert (36):

$$d_{neu} = d_{alt} - \Delta d_{r_{2/3}} = d_{alt} - d_{r_{min}} - d_{r_{2/3}} \quad (36)$$

Diese Berechnungsschritte werden insgesamt viermal durchgeführt. Anschließend wird das resultierende Ergebnis als hinreichend genau angesehen. Die neue Mittendicke d_{neu} ersetzt den bisherigen Wert und wird in das zugehörige *Edittext-Feld* geschrieben (Listing 6).

```
for (int i = 0; i < 4; i++) {

    […]

    t1 = r1 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
            (Math.pow(r1, 2))))));
    t2 = r2 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
            (Math.pow(r2, 2))))));

    if (cyl_in != 0) {

        t3 = r3 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
                (Math.pow(r3, 2))))));

        if (r2 >= r3) {

            dr = d_in - t1 + t2;
            ddr = dr - dr_in;
            d_in = d_in - ddr;
            d_in = Math.round(d_in*10000)/10000.0;
        } else {
            dr = d_in - t1 + t3;
            ddr = dr - dr_in;
            d_in = d_in - ddr;
            d_in = Math.round(d_in*10000)/10000.0;
        }
    } else {
        dr = d_in - t1 + t2;
        ddr = dr - dr_in;
        d_in = d_in - ddr;
        d_in = Math.round(d_in*10000)/10000.0;
    }

    et_glasrechner_d_in.setText(Double.toString(d_in));

}
```

Listing 6 – Ausschnitt aus dem Programmcode *Glasberechung.java*, der für die Bestimmung der resultierenden Mittendicke bei vorhandener minimaler Randdicke verantwortlich ist

Sollten beide Hauptschnitte des Brillenglases allerdings negativ sein, so wird lediglich eine minimale Mittendicke als notwendige Eingabe vorausgesetzt.

Die Bauhöhe h lässt sich über die Scheiteltiefen t_1 bis t_3 und der Mittendicke d bestimmen. Die Randdicke d_r des negativeren Hauptschnitts wird zur Berechnung verwendet, sofern es sich um ein torisches Glas handelt (37):

$$h = t_1 + d_r \tag{37}$$

Das gesamte Volumen V_{Ges} wird näherungsweise über beide Hauptschnitte gemittelt (38):

$$V_{Ges} = \frac{\pi \cdot \left[d_{r_1} \cdot h^2 + t_1^2 \cdot \left(r_1 - \frac{t_1}{3}\right) - t_2^2 \cdot \left(r_2 - \frac{t_2}{3}\right)\right] + \pi \cdot \left[d_{r_2} \cdot h^2 + t_1^2 \cdot \left(r_1 - \frac{t_1}{3}\right) - t_3^2 \cdot \left(r_3 - \frac{t_3}{3}\right)\right]}{2} \tag{38}$$

Anschließend wird die Masse des Rohglases m_{Glas} berechnet (39):

$$m_{Glas} = \rho \cdot V_{Ges} \tag{39}$$

mit: $\boldsymbol{\rho}$ **Dichte des Glasmaterials**

Der Astigmatismus schiefer Bündel A ergibt sich aus der Differenz der Kehrwehrte von sagittaler (S) und tangentialer (T) Schnittweite (40). Dieser wird individuell – je nach Angabe der Schrittweite – für jeden einzelnen Blickwinkel β und für beide Hauptschnitte des Glases berechnet:

$$A_{\beta_{1/2}} = T_{\beta_{1/2}} - S_{\beta_{1/2}} \tag{40}$$

mit: $\boldsymbol{T_\beta}$ **Kehrwert der tangentialen Schnittweite in dpt**

$\boldsymbol{S_\beta}$ **Kehrwert der sagittalen Schnittweite in dpt**

Der Refraktionsfehler R wird ebenfalls für jeden einzelnen Blickwinkel α berechnet und ergibt sich aus dem Mittelwert der Kehrwerte von sagittaler und tangentialer Schnittweite abzüglich des Scheitelbrechwerts des zugehörigen Hauptschnitts auf der optischen Achse (41):

$$R_{\beta_{1/2}} = \frac{T_{\beta_{1/2}} + S_{\beta_{1/2}}}{2} - S'_{\infty_{1/2}} \qquad (41)$$

Als weiterer Bestandteil des Programms wird ebenfalls die Verzeichnung Vz in Abhängigkeit des Blickwinkels β für jeden Hauptschnitt ermittelt (42):

$$Vz_{\beta_{1/2}} = \frac{\frac{\tan(u'_{2_\beta})_{1/2}}{\tan(u_{1_\beta})_{1/2}}}{\frac{\tan(u'_{2_0})_{1/2}}{\tan(u_{1_0})_{1/2}}} \cdot 100\% - 100 \qquad (42)$$

mit:

- u'_{2_β} **Winkel zwischen dem ausfallenden Lichtstrahl und der optischen Achse nach der Fläche F_2 in Abhängigkeit des Blickwinkels β**
- u_{1_β} **Winkel zwischen dem einfallenden Lichtstrahl und der optischen Achse vor der Fläche F_1 in Abhängigkeit des Blickwinkels β**
- u'_{2_0} **Winkel zwischen dem ausfallenden Lichtstrahl und der optischen Achse nach der Fläche F_2 bei annähernd 0°**
- u_{1_0} **Winkel zwischen dem einfallenden Lichtstrahl und der optischen Achse vor der Fläche F_1 bei annähernd 0°**

Nachdem der Nutzer alle notwendigen Parameter eingetragen und seine Eingabe über den *Berechnen-Button* bestätigt (Abb. 26), startet das Programm mit den oben aufgeführten Berechnungsschritten (31-42) und gibt die Ergebnisse anschließend aus (Abb. 27-30).

Dabei werden zuerst die Eigenschaften des Glases gelistet, die unabhängig des Blickwinkels sind (Abb. 27).

Der Refraktionsfehler, der Astigmatismus und die Verzeichnung werden tabellarisch in Abhängigkeit des Blickwinkels aufgelistet und zusätzlich mit Hilfe eines Graphen visualisiert (Abb. 28-30). Dies wird mit Hilfe des *Plugins MPAndroidChart* ermöglicht (Jahoda 2016).

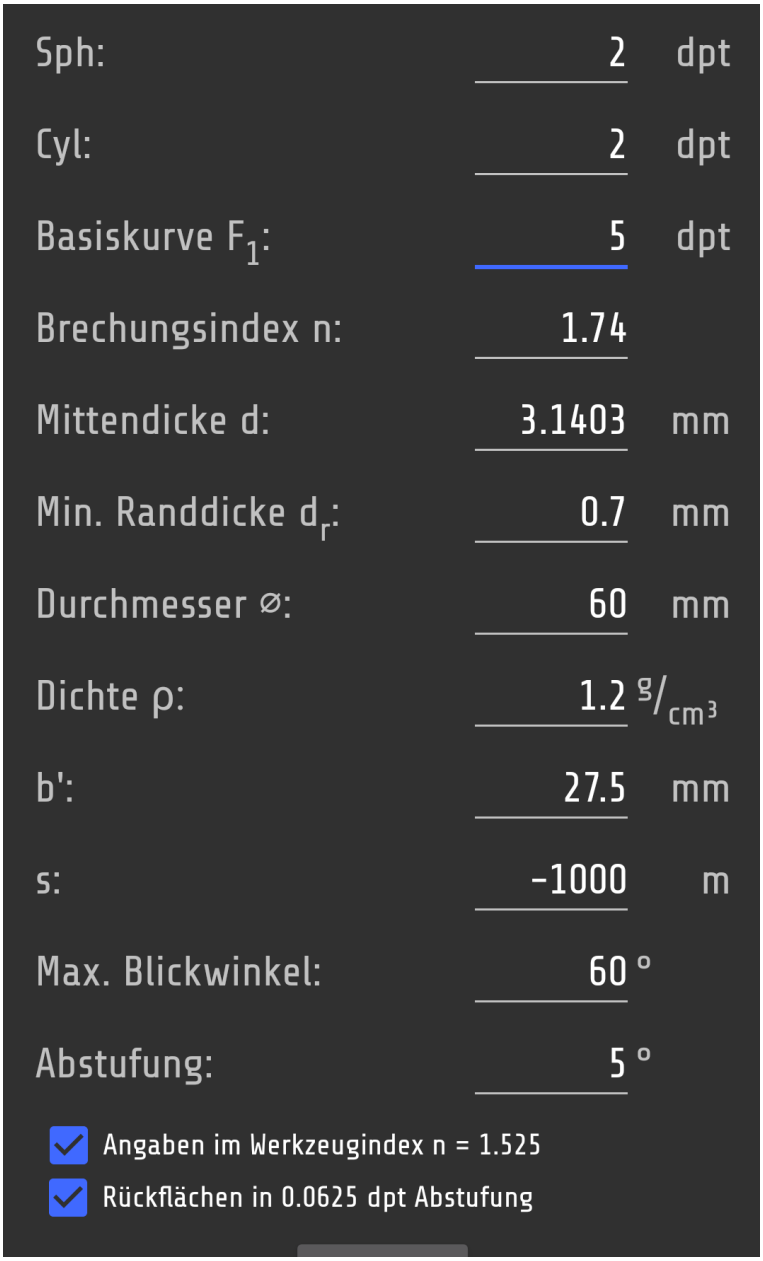

Abbildung 26 – Eingabeparameter eines sphäro-torischen Glases in der Pluszylinder-Schreibweise (mit Sphäre Sph, Zylinder Zyl, Basiskurve F_1, Brechungsindex n, Randdicke d_r, Mittendicke d, Rohglasdurchmesser $\emptyset$, Dichte des Glasmaterials ρ, Abstand des Augendrehpunktes zur Rückfläche des Glases b', Objektabstand s, maximaler Blickwinkel von 60°, Berechnungsintervall von je 5°, Eingaben und Ergebnisse unter Berücksichtigung des Werkzeugindex und einer Rundung der Rückflächenradien in 0,0625 dpt Abstufungen)

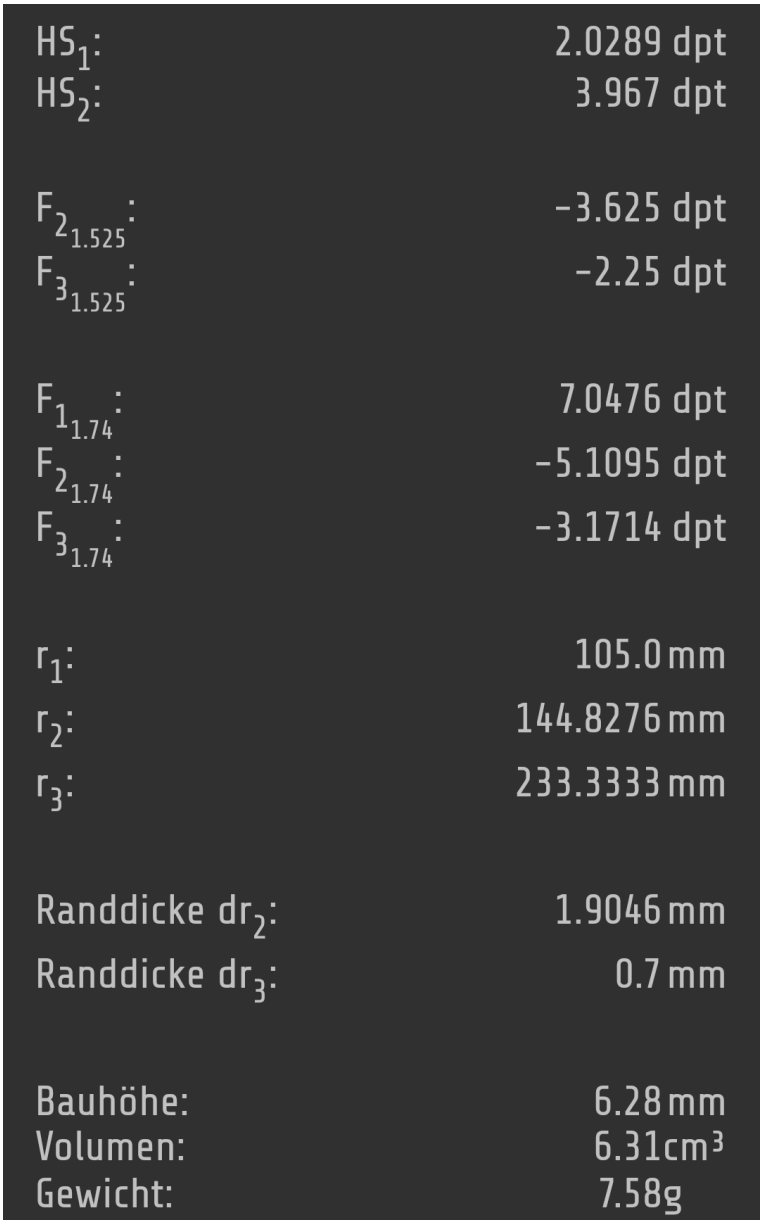

Abbildung 27 – Oberer Abschnitt des Ergebnisfensters für die Eingabeparameter aus Abbildung 26 mit den Scheitelbrechwerten im jeweiligen Hauptschnitt $HS_{1/2}$, den Flächenbrechwerten der Rückflächen für den Werkzeugindex $F_{2/3_{1,525}}$, den tatsächlichen Flächenbrechwerten für den angegebenen Brechungsindex $F_{1/2/3_{1,74}}$, den Radien $r_{1/2/3}$, den Randdicken $d_{r_{2/3}}$, der Bauhöhe, dem Volumen und dem Gewicht

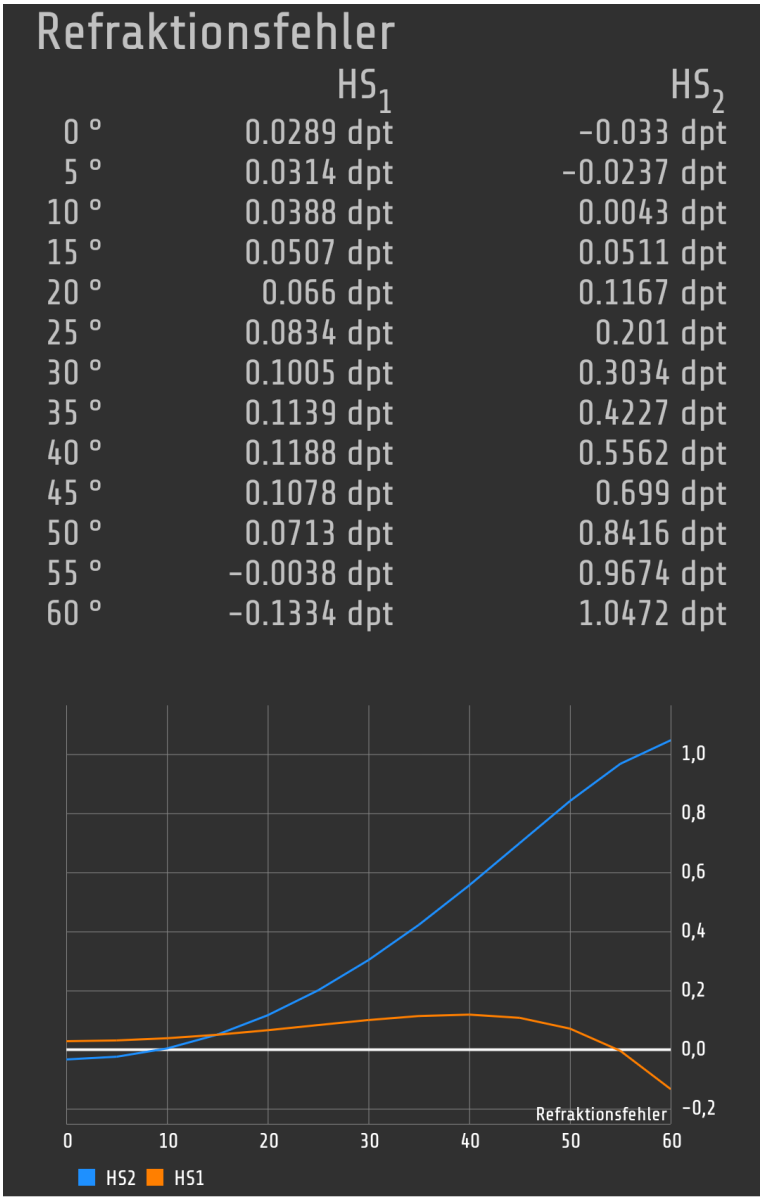

Abbildung 28 – Unterer Abschnitt des Ergebnisfensters mit der tabellarischen Auflistung des Refraktionsfehlers in Abhängigkeit des Blickwinkels für beide Hauptschnitte und die Visualisierung mit Hilfe eines Graphen

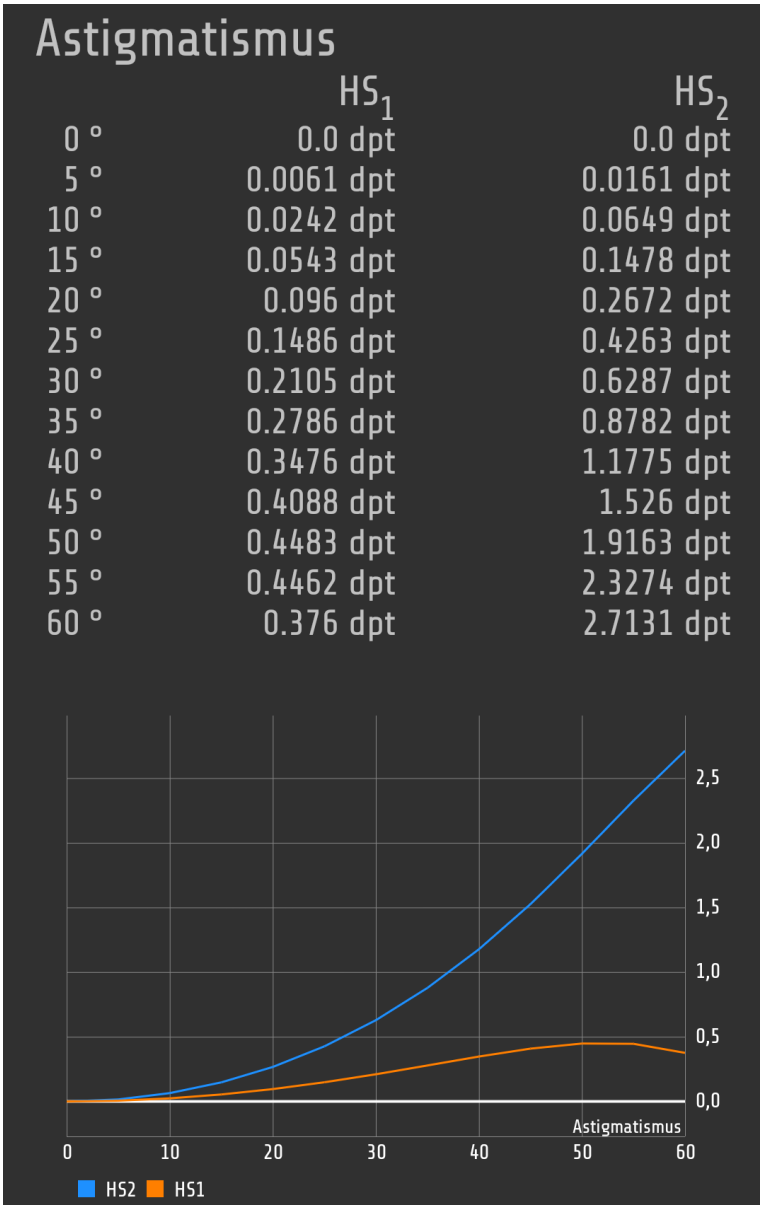

Abbildung 29 – Unterer Abschnitt des Ergebnisfensters mit der tabellarischen Auflistung des Astigmatismus in Abhängigkeit des Blickwinkels für beide Hauptschnitte und die Visualisierung mit Hilfe eines Graphen

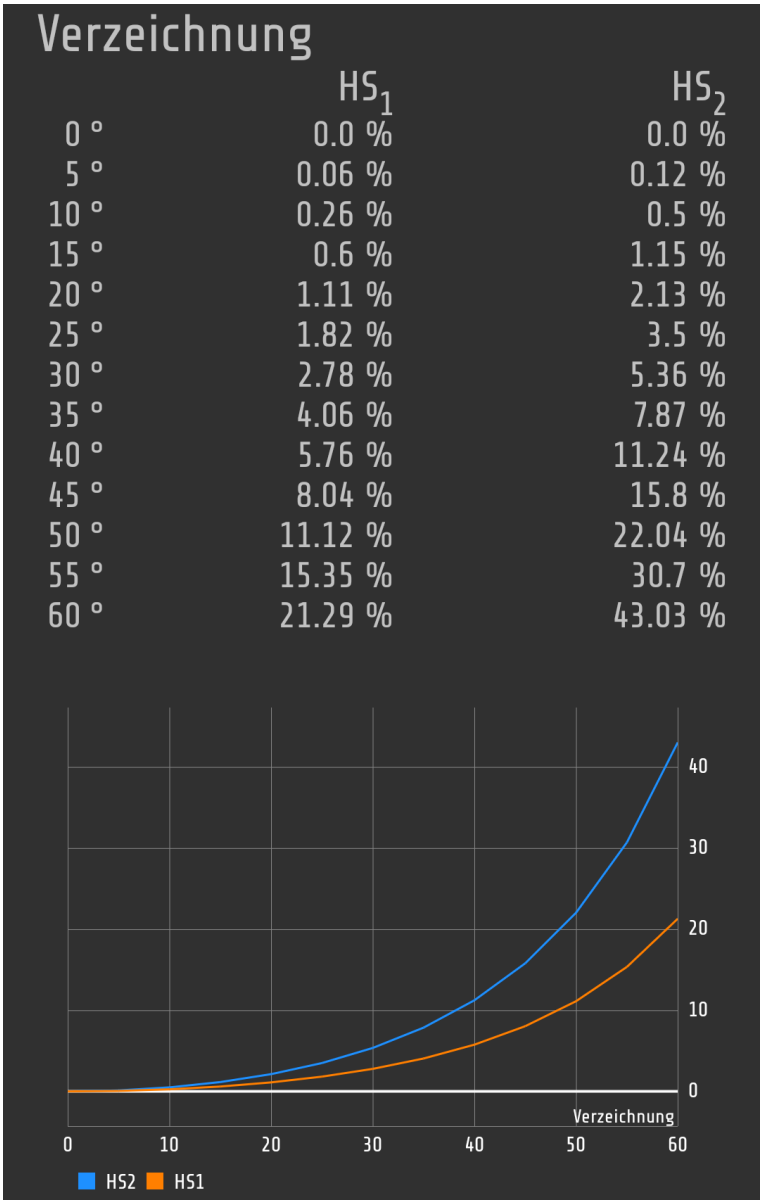

Abbildung 30 – Unterer Abschnitt des Ergebnisfensters mit der tabellarischen Auflistung der Verzeichnung in Abhängigkeit des Blickwinkels für beide Hauptschnitte und die Visualisierung mit Hilfe eines Graphen

3.9 Informationen zum Programm

Dieser Programmabschnitt besteht aus der *Activity Info.java* und dem Layout *info.xml* (Abb. 31).

Abbildung 31 – Layout und Inhalt des Programmabschnitts *Informationen zum Programm*

4 Diskussion

Die Applikation *Aalener Optik-Formelrechner* ist ein in sich abgeschlossenes Programm, das – sofern sich an der Abwärtskompatibilität nichts ändert – auch mit zukünftigen Android-Betriebssystemen einwandfrei funktioniert. Da es sich bei allen bisher implementierten *Activities* um in sich geschlossene Strukturen handelt, lassen sich diese problemfrei ersetzen, tauschen oder sogar in andere Applikationen eingliedern. Durch den Aufbau der Startseite mit Hilfe einer *ListView* kann die Applikation sehr einfach durch weitere Teilprogramme erweitert werden. Der bisherige Funktionsumfang des Programms beschränkt sich auf das Lösen von Formeln, die für den Speicher vieler Taschenrechner zu groß sind oder aber für die der Lösungsweg viel Zeit in Anspruch nimmt und fehleranfällig ist. Die verwendeten Formeln werden in gleicher Weise in den Vorlesungen des Studiengangs Augenoptik und Hörakustik verwendet, so dass die Applikation vorlesungsbegleitend verwendet werden kann. Gegebenenfalls eingesetzte Näherungen haben sich als hinreichend genau erwiesen. Nichtsdestotrotz ersetzt diese Applikation keine komplexen Rechenprogramme, sondern sollte als Hilfestellung, die Dank des Smartphones immer dabei ist, betrachtet werden.

Die Wahl von Android 4.0.3 als niedrigstes unterstütztes Betriebssystem geht aus der Meldung von Android Studio hervor, dass mindestens 97 % aller Android-Geräte mindestens über diese Version verfügen. Leider kann es bei Geräten mit einer niedrigeren Version als 5.0 zu kosmetischen Fehlern im Layout kommen, die zwar unschön, nicht aber funktionsbeeinträchtigend sind (Vgl. Einleitung).

Idealerweise wird dieses Projekt in Zukunft fortgesetzt und um weitere Funktionen erweitert. Die notwendigen Grundkenntnisse für den Einstieg in die Programmierung mit Android Studio liefern neben Büchern (Louis et al., 2016) auch eine digitale Anleitung (Google, 2000-2017). Darüber hinaus befindet sich auf der Videoplattform youtube.com ein riesiges Angebot an Anleitungen, die durch erfahrene Entwickler veröffentlicht wurden.

Literatur

Baumbach, Peter (2016): Projekt Einstärkenglasentwicklung. Unterlagen zur Theorie der Berechnung. Skript. Aalen [Stand: 15. Januar 2017]

Destatis (2016): 61 % der Unternehmen in Deutschland ermöglichen mobiles Arbeiten. Pressemitteilung Nr. 443. Wiesbaden: Destatis - Statistisches Bundesamt, 09. Dezember 2016. URL: https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2016/12/PD16_443_52911.html [Stand: 06. März 2017]

Gartner (2017): Gartner Says 2016 Marked Fifth Consecutive Year of Worldwide PC Shipment Decline. PC Industry Ended the Year with 3.7 Percent Decline in 4Q16 Shipments. Stamford: Gartner, 11. Januar 2017. URL: http://www.gartner.com/newsroom/id/3568420 [Stand: 06. März 2017]

Google (2000-2017): Android Studio 2.3

Google Support - Nexus (2017): „Schutz vor schädlichen Apps“. Google Inc. (Hg.). URL: https://support.google.com/nexus/answer/2812853?hl=de [Stand: 12. März 2017]

Jahoda, Philipp (2016): „MPAndroidChart“. URL: https://github.com/PhilJay/MPAndroidChart [Stand: 15. April 2017]

Louis, Dirk/Müller, Peter (2016): Android. Der schnelle und einfache Einstieg in die Programmierung und Entwicklungsumgebung. 2. Auflage. München: Carl Hanser Verlag GmbH & Co. KG

Oracle (2015): „Java Development Timeline“. URL: http://oracle.com.edgesuite.net/timeline/java/ [Stand: 11. März 2017]

Sharma, Pratik (2013): „Forenbeitrag. Is there a way to define a min and max value for EditText in Android?“. Stack Exchange Inc. (Hg.). URL: http://stackoverflow.com/questions/14212518/is-there-a-way-to-define-a-min-and-max-value-for-edittext-in-android [Stand: 16. April 2017]

Sui, Linda (2016): „Strategy Analytics: Android Captures Record 88 Percent Share of Global Smartphone Shipments in Q3 2016“. URL: https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-

press-releases/strategy-analytics-press-release/2016/11/02/strategy-analytics-android-captures-record-88-percent-share-of-global-smartphone-shipments-in-q3-2016#.WBteR-QwdaQ [Stand: 06. März 2017]

Abbildungsverzeichnis

Listing-Verzeichnis

Anhang - Quellentext

a) AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="de.HS_Aalen.OptikFormelrechner">

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name="de.HS_Aalen.OptikFormelrechner.MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Raytracing_Start"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Raytracing_Berechnung"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Exaktes_Raytracing_Start"></activity>
    <activity
android:name="de.HS_Aalen.OptikFormelrechner.Exaktes_Raytracing_Berechnung"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Powervektoren"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Schott"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Shape"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Glasberechnung"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Transmission_Start"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Transmission_Berechnung"></activity>
    <activity android:name="de.HS_Aalen.OptikFormelrechner.Info"></activity>
</application>
</manifest>
```

b) Don – build.gradle

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
        maven { url "https://jitpack.io" }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

c) app – build.gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "25.0.0"
    defaultConfig {
        applicationId "de.HS_Aalen.don"
        minSdkVersion 15
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile 'com.github.PhilJay:MPAndroidChart:v3.0.1'
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.1'
    testCompile 'junit:junit:4.12'
}
```

d) strings.xml

```
<resources>
    <string name="app_name">Optik-Formelrechner</string>
    <string name="Farbauswahl">Farbauswahl</string>
    <string name="Auswahlglas">Glasauswahl</string>
    <string name="glasart">Glas: </string>
    <string name="s_unendlich">s<sub><small>1</small></sub> = -∞</string>
    <string name="txt_a0">A<sub><small>0</small></sub>:</string>
    <string name="txt_a1">A<sub><small>1</small></sub>:</string>
    <string name="txt_a2">A<sub><small>2</small></sub>:</string>
    <string name="txt_a3">A<sub><small>3</small></sub>:</string>
    <string name="txt_a4">A<sub><small>4</small></sub>:</string>
    <string name="txt_a5">A<sub><small>5</small></sub>:</string>
    <string-array name="glasarten">

        <item>BaK 4</item>
        <item>BK 7</item>
        <item>K 5</item>
        <item>SF 10</item>
        <item>Eigene Angabe</item>
    </string-array>
    <string-array name="farbwahl">
        <item> Λ<sub><small>i\</small></sub>"   "365.0 nm</item>
        <item> Λ<sub><small>h\</small></sub>"   "404.7 nm</item>
        <item> Λ<sub><small>g\</small></sub>"   "435.8 nm</item>
        <item> Λ<sub><small>F\'\</small></sub>"   "480.0 nm</item>
        <item> Λ<sub><small>F\</small></sub>"   "486.1 nm</item>
        <item> Λ<sub><small>e\</small></sub>"   "546.1 nm</item>
        <item> Λ<sub><small>d\</small></sub>"   "587.6 nm</item>
        <item> Λ<sub><small>D\</small></sub>"   "589.3 nm</item>
        <item> " "Λ"     "632.8 nm</item>
        <item> Λ<sub><small>C\'\</small></sub>"   "643.8 nm</item>
        <item> Λ<sub><small>C\</small></sub>"   "656.3 nm</item>
        <item> Λ<sub><small>f\</small></sub>"   "706.5 nm</item>
        <item> Λ<sub><small>s\</small></sub>"   "852.1 nm</item>
        <item> Λ<sub><small>t\</small></sub>"   "1014.0 nm</item>
        <item> " "Λ"    "1060.0 nm</item>
        <item> " "Λ"    "1529.6 nm</item>
        <item> " "Λ"    "1970.1 nm</item>
        <item> " "Λ"    "2325.4 nm</item>
        <item> Eigene Angabe</item>
    </string-array>
   <string-array name="spinner_shape_D">

       <item> D<sub><small>Ges\<sub><small><small>Luft</small></small></sub></small></sub> :</item>
       <item> f\' :</item>
```

```
   </string-array>
    <string-array name="shape_s">

        <item> s :</item>
        <item> s\' :</item>
    </string-array>
    <string name="txt_shape_Shape_sph">S<sub><small>opt</small></sub> :</string>
    <string name="txt_shape_r1_sph">r<sub><small>1</small></sub> :</string>
    <string name="txt_shape_r2_sph">r<sub><small>2</small></sub> :</string>
    <string name="txt_shape_Shape_koma">S<sub><small>komafrei</small></sub> :</string>
    <string name="txt_shape_r1_koma">r<sub><small>1</small></sub> :</string>
    <string name="txt_shape_r2_koma">r<sub><small>2</small></sub> :</string>
    <string name="txt_glasrechner_f1">Basiskurve F<sub><small>1</small></sub>:</string>
    <string name="txt_glasrechner_dr">Min. Randdicke d<sub><small>r</small></sub>:</string>
    <string
name="txt_glasrechner_rho_einheit"><sup><small>g</small></sup>/<sub><small>cm³</small></sub>
</string>
    <string name="txt_glasrechner_HS1">HS<sub><small>1</small></sub>:</string>
    <string name="txt_glasrechner_HS2">HS<sub><small>2</small></sub>:</string>
    <string
name="txt_glasrechner_F1_real">F<sub><small>1<sub><small>real</small></sub></small></sub>:</string>
    <string
name="txt_glasrechner_F2_real">F<sub><small>2<sub><small>real</small></sub></small></sub>:</string>
    <string
name="txt_glasrechner_F3_real">F<sub><small>3<sub><small>real</small></sub></small></sub>:</string>
    <string
name="txt_glasrechner_F2_wkn">F<sub><small>2<sub><small>1.525</small></sub></small></sub>:</string>
    <string
name="txt_glasrechner_F3_wkn">F<sub><small>3<sub><small>1.525</small></sub></small></sub>:</string>
    <string name="txt_glasrechner_R1">r<sub><small>1</small></sub>:</string>
    <string name="txt_glasrechner_R2">r<sub><small>2</small></sub>:</string>
    <string name="txt_glasrechner_R3">r<sub><small>3</small></sub>:</string>
    <string name="txt_glasrechner_DR2">Randdicke dr<sub><small>2</small></sub>:</string>
    <string name="txt_glasrechner_DR3">Randdicke dr<sub><small>3</small></sub>:</string>
    <string name="txt_glasrechner_HS1_table">HS<sub><small>1</small></sub></string>
    <string name="txt_glasrechner_HS2_table">HS<sub><small>2</small></sub></string>
</resources>
```

e) Programmtext – MainActivity.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.Context;
import android.content.Intent;
import android.content.res.Configuration;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.ListView;

public class MainActivity extends AppCompatActivity {

    final Context context = this;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);

        ImageView keyvisual = (ImageView) findViewById(R.id.keyvisual);

        // Zeige das KeyVisual-Image nur im Porträtmodus und nicht im Landscape
        if (getResources().getConfiguration().orientation == Configuration.ORIENTATION_LANDSCAPE) {
            keyvisual.setVisibility(View.GONE);
        }
        else {
            keyvisual.setVisibility(View.VISIBLE);
        }

        ListView listView = (ListView) findViewById(R.id.ListView1);

        final String liste []= {
                "Paraxiales Ray-Tracing",
                "Exaktes Ray-Tracing für Objekt auf opt. Achse",
                "Schiefgekreuzte Zylinder",
                "Berechnung der Brechungsindices nach der Schottgleichung",
                "Relative Transmission",
                "Coddington Faktoren",
                "Abbildungsfehler eines sph.-tor. Brillenglases",
                "Informationen zum Programm",
        };


        ArrayAdapter <String> adapter = new ArrayAdapter<String>
                (this,android.R.layout.simple_list_item_1, liste);

        listView.setAdapter(adapter);


        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {



            @Override
            public void onItemClick( AdapterView<?> parent, View view, int i, long id) {

                //Weist jedem String Element in "liste" eine Klasse zu, die durch klicken geöffnet
                // wird

                if (i == 0) {
                    Intent s = new Intent(view.getContext(), Raytracing_Start.class);
                    startActivity(s);
                }

                if (i == 1) {
                    Intent s = new Intent(view.getContext(), Exaktes_Raytracing_Start.class);
                    startActivity(s);
                }

                if (i == 2) {
                    Intent s = new Intent(view.getContext(), Powervektoren.class);
                    startActivity(s);
                }

                if (i == 3) {
                    Intent s = new Intent(view.getContext(), Schott.class);
                    startActivity(s);
                }

                if (i == 4) {
                    Intent s = new Intent(view.getContext(), Transmission_Start.class);
```

```
                startActivity(s);
            }

            if (i == 5) {
                Intent s = new Intent(view.getContext(), Shape.class);
                startActivity(s);
            }

            if (i == 6) {
                Intent s = new Intent(view.getContext(), Glasberechnung.class);
                startActivity(s);
            }

            if (i == 7) {
                Intent s = new Intent(view.getContext(), Info.class);
                startActivity(s);
            }
        }
    });

}
}
```

f) Programmtext – Raytracing_Start.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.Intent;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.content.Context;


public class Raytracing_Start extends AppCompatActivity {

    final Context context = this;
    private EditText et;
    private Button bt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.raytracing_start);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        et = (EditText) findViewById(R.id.eT);
        bt = (Button) findViewById(R.id.bT);

        bt.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String enteredText = et.getText().toString();

                // Sollte keine Zahl eingegeben werden, so wird das Layout für eine einzelne
                // Fläche erstellt
                if (enteredText.equals("")) {
                    enteredText = "1";
                }

                // Beschränkt die Berechnung auf maximal 1000 Flächen und gibt bei einer größeren
                // Zahl einen Warnhinweis
                int Overload = Integer.parseInt(enteredText);
                if (Overload > 1000) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Obere Grenze");
                    AlarmBox.setMessage("Es können maximal 1000 Flächen berechnet werden");
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    return;
                }

                // Die Bundle Funktion ermöglicht die Weitergabe der Anzahl der zu berechnenden
                // Flächen an die nächste Activity
                // Zudem wird der Status der Checkbox ebenfalls weitergeleitet
```

```
                Bundle korb = new Bundle();
                korb.putString("datenpaket1", enteredText);
                Intent in = new Intent(Raytracing_Start.this, Raytracing_Berechnung.class);
                in.putExtras(korb);

                final CheckBox checkBox = (CheckBox) findViewById(R.id.check_s1);
                Bundle korb2 = new Bundle();
                if (checkBox.isChecked()) {
                    korb2.putString("datenpaket2", "1");
                } else {
                    korb2.putString("datenpaket2", "0");
                }
                in.putExtras(korb2);
                startActivity(in);

                }
        });
    }
}
```

g) Programmtext – Raytracing_Berechnung.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.app.ActionBar.LayoutParams;
import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.text.InputType;
import android.util.TypedValue;
import android.view.Gravity;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import static android.util.TypedValue.COMPLEX_UNIT_DIP;


public class Raytracing_Berechnung extends AppCompatActivity {

    final Context context = this;
    EditText B1,B2,B3,B4,B5,Z1,Z2,Z3,Z4,VN1,VNM1;
    TextView Ausgabe, Ausgabe_zwischenergebnisse;
    double S,S_1,S_2,N1,N2,N5,R1,D1,S1,S2,A,B,C,D2,f_1,D,f,E,Z_R1,Z_N1,Z_N2,Z_D,Z_S_1,Z_B,Z_A,
            sH,sH_1,knoten,knoten_1,Vgr,Vgr_T,Vgr_W,Vgr_N1,Vgr_Nm1,D_Ges;
    int zwischenspeicher;
    String formation;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.raytracing_berechnung);

        // Empfängt die gesendeten Inhalte aus der Bundle Funktion
        Bundle zielkorb = getIntent().getExtras();
        String text2 = zielkorb.getString("datenpaket1");
        final int fn = Integer.parseInt(text2);
        Bundle zielkorb2 = getIntent().getExtras();
        String text3 = zielkorb2.getString("datenpaket2");
        final int cb_s1 = Integer.parseInt(text3);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        // Erstellt im folgenden ein komplett dynamisches Layout auf Basis der Anzahl der Flächen
        ScrollView sv = new ScrollView(this);
        LinearLayout ll = new LinearLayout(this);

        ll.setOrientation(LinearLayout.VERTICAL);

        final LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
        params.setMargins(200,0,0,0);

        sv.addView(ll);
```

```
int k = 1;

// Erstellt an dieser Stelle die TextView und EditText Elemente die zur Eingabe benötigt
// werden und weißt jedem Element eine eigene ID zu
for (int i = 0; i < fn; i++) {
    TextView tV0 = new TextView(this);
    tV0.setPadding(20,0,0,0);
    tV0.setLines(3);
    tV0.setGravity(Gravity.CENTER_VERTICAL);
    tV0.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);
    TextView tVa = new TextView(this);
    tVa.setPadding(100,0,0,5);
    TextView tVb = new TextView(this);
    tVb.setPadding(100,0,0,5);
    TextView tVc = new TextView(this);
    tVc.setPadding(100,0,0,5);
    TextView tVd = new TextView(this);
    tVd.setPadding(100,0,0,5);
    TextView tVe = new TextView(this);
    tVe.setPadding(100,0,0,5);
    TextView tVf = new TextView(this);
    tVf.setLines(3);
    tVf.setPadding(20,0,0,0);
    tVf.setGravity(Gravity.CENTER_VERTICAL);
    tVf.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);


    EditText et1 = new EditText(this);
    et1.setLayoutParams(params);
    et1.setPadding(20,0,20,20);
    et1.setEms(4);
    et1.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
    et1.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_SIGNED |
            InputType.TYPE_NUMBER_FLAG_DECIMAL);
    et1.setId(k);
    k = k + 1;
    EditText et2 = new EditText(this);
    et2.setLayoutParams(params);
    et2.setPadding(20,0,20,20);
    et2.setEms(4);
    et2.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
    et2.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
    et2.setId(k);
    k = k + 1;
    EditText et3 = new EditText(this);
    et3.setLayoutParams(params);
    et3.setPadding(20,0,20,20);
    et3.setEms(4);
    et3.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
    et3.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
    et3.setId(k);
    k = k + 1;
    EditText et4 = new EditText(this);
    et4.setLayoutParams(params);
    et4.setPadding(20,0,20,20);
    et4.setEms(4);
    et4.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
    et4.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_SIGNED |
            InputType.TYPE_NUMBER_FLAG_DECIMAL);
    et4.setId(k);
    k = k + 1;
    EditText et5 = new EditText(this);
    et5.setLayoutParams(params);
    et5.setPadding(20,0,20,20);
    et5.setEms(4);
    et5.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
    et5.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
    et5.setId(k);
    k = k + 1;

    tV0.setText("Fläche: " + (i + 1));
    // Benötigt vor der ersten Fläche die Felder für den Objektabstand s und
    // den Brechungsindex n
    if (i == 0) {
        tVa.setText(Html.fromHtml("s" + "<sub><small>" + (i + 1) + "</small></sub> [mm]"));
        tVb.setText(Html.fromHtml("n" + "<sub><small>" + (i + 1) + "</small></sub>"));
    }
    tVc.setText(Html.fromHtml("n" + "<sub><small>" + (i + 2) + "</small></sub>"));
    tVd.setText(Html.fromHtml("r" + "<sub><small>" + (i + 1) + "</small></sub> [mm]"));
    tVe.setText(Html.fromHtml("d" + "<sub><small>" + (i + 1) + "</small></sub> [mm]"));
    tVf.setText("Abstand zwischen Fläche " + (i+1) + " & " + (i+2));


    ll.addView(tV0);
    if (i == 0) {

        //Fügt die Eingabe für den Objektabstand s nur ein, wenn der Status der
        // Checkbox in der vorigen Activity "false" war
```

```
            if (cb_s1 == 0) {
                ll.addView(tVa);
                ll.addView(et1);
            }
            ll.addView(tVb);
            ll.addView(et2);
        }
        ll.addView(tVc);
        ll.addView(et3);
        ll.addView(tVd);
        ll.addView(et4);
        //Lässt das Eingabefeld und die Beschreibung für den Abstand zwischen den Flächen
        // "d" nach der letzten Fläche weg
        if (i < fn - 1) {
            ll.addView(tVf);
            ll.addView(tVe);
            ll.addView(et5);
        }
    }

    Button bt = new Button(this);
    bt.setText("Berechnen");
    bt.setAllCaps(false);
    ll.addView(bt);

    //Erstellt das das finale TextView in der das Ergebnis anschließend dargestellt wird
    final TextView tv = new TextView(this);
    tv.setText("");
    tv.setId(0);
    tv.setVisibility(View.GONE);
    tv.setWidth(200);
    tv.setTextSize(COMPLEX_UNIT_DIP, 20);
    tv.setPadding(100,50,0,50);
    ll.addView(tv);

        //Erstellt das TextView in der die Zwischenergebnisse fargestellt werden
        final TextView zwischenergebnisse = new TextView(this);
        zwischenspeicher = fn*5+1;
        zwischenergebnisse.setId(zwischenspeicher);
        zwischenergebnisse.setMaxLines(fn*2-1);
        zwischenergebnisse.setPadding(15,0,0,15);
        zwischenergebnisse.setVisibility(View.GONE);
        zwischenergebnisse.setTextSize(COMPLEX_UNIT_DIP, 17);
        ll.addView(zwischenergebnisse);


    this.setContentView(sv);


    bt.setOnClickListener (new View.OnClickListener() {
        public void onClick(View v) {


            int j = 1;
            int absolut = 1;
            tv.setVisibility(View.VISIBLE);
            f_1 = 1;
            if (fn > 1) {
                zwischenergebnisse.setVisibility(View.VISIBLE);
                if (fn == 2) {
                    formation = "Zwischenergebnis: <br>";
                } else {
                    formation = "Zwischenergebnisse: <br>";
                }
            }

            // Berechnung wird für die Anzahl "fn" der Flächen durchgeführt
            for ( int k = 0; k < fn; k++) {

                if (k == 0) {

                    // 1. Fall => Objektabstand s ist -unendlich
                    if (cb_s1 == 0) {

                        //Abfrage ob das Eingabegeld leer ist mit Warnhinweis
                        B1 = (EditText) findViewById(j);
                        String Alarm = B1.getText().toString();
                        if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                                Alarm.equals("+")) {
                            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                            AlarmBox.setTitle("Leeres Feld!");
                            AlarmBox.setMessage(Html.fromHtml("s<sub><small>1</small>" +
                                    "</sub> ist leer<br>"));
                            AlarmBox.setNeutralButton("OK", null);

                            AlertDialog dialog = AlarmBox.create();
                            dialog.show();
                            tv.setVisibility(View.GONE);
```

```
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double S1 = Double.parseDouble(B1.getText().toString());
        j = j + 1;
        B2 = (EditText) findViewById(j);
        Alarm = B2.getText().toString();
        if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("n<sub><small>1</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double N1 = Double.parseDouble(B2.getText().toString());
        j = j + 1;
        B3 = (EditText) findViewById(j);
        Alarm = B3.getText().toString();
        if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("n<sub><small>2</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double N2 = Double.parseDouble(B3.getText().toString());
        j = j + 1;
        B4 = (EditText) findViewById(j);
        Alarm = B4.getText().toString();
        if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("r<sub><small>1</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double R1 = Double.parseDouble(B4.getText().toString());
        j = j + 1;
        // So lange noch nicht die letzte Fläche erreicht wurde, wird der
        // Abstand zwischen den Flächen "d" eingelesen
        if (k < fn - 1) {
            B5 = (EditText) findViewById(j);
            Alarm = B5.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("d<sub><small>1</small>" +
                        "</sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double D1 = Double.parseDouble(B5.getText().toString());
            j = j + 1;
        }

        S_1 = (N1 / S1 + (N2 - N1) / R1) / N2;
        B = N2;

        Vgr = (1/S_1)/S1;

        //2. Fall => Objektabstand wurde manuell eingegeben
```

```
        } else if (cb_s1 == 1) {
            j = j + 1;
            B2 = (EditText) findViewById(j);
            String Alarm = B2.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("n<sub><small>1</small>" +
                        "</sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double N1 = Double.parseDouble(B2.getText().toString());
            j = j + 1;
            B3 = (EditText) findViewById(j);
            Alarm = B3.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("n<sub><small>2</small>" +
                        "</sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double N2 = Double.parseDouble(B3.getText().toString());
            j = j + 1;
            B4 = (EditText) findViewById(j);
            Alarm = B4.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("r<sub><small>1</small>" +
                        "</sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double R1 = Double.parseDouble(B4.getText().toString());
            j = j + 1;
            if (k < fn - 1) {
                B5 = (EditText) findViewById(j);
                Alarm = B5.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                        Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("d<sub><small>1</small>" +
                            "</sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double D1 = Double.parseDouble(B5.getText().toString());
                j = j + 1;
            }

            S_1 = ((N2 - N1) / R1) / N2;
            B = N2;
            D = S_1 - D1;
            f_1 = (S_1/D);


        }
        // Nachdem die erste Fläche berechnet wurde, ist die Berechnung für die
        // restlichen Flächen sowohl für den 1. als auch für den 2. Fall identisch
    } else {
        j = j + 2;
```

```
        B3 = (EditText) findViewById(j);
        String Alarm = B3.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("n<sub><small>" + (k+2) +
                    "</small></sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double N2 = Double.parseDouble(B3.getText().toString());
        j = j + 1;
        B4 = (EditText) findViewById(j);
        Alarm = B4.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("r<sub><small>" + (k+1) +
                    "</small></sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double R1 = Double.parseDouble(B4.getText().toString());
        j = j + 1;
        if (k < fn - 1 ) {
            B5 = (EditText) findViewById(j);
            Alarm = B5.getText().toString();
            if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("d<sub><small>" + (k+1) +
                        "</small></sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double D1 = Double.parseDouble(B5.getText().toString());
            j = j + 1;

        }
        S_1 = (B / A + (N2 - B) / R1) / N2;
        B = N2;
        Vgr = Vgr * ((1/S_1)/A);

    }
    S_1 = 1 / S_1;


    A = S_1;
    // So lange nicht die letzte Fläche erreicht wurde, schreibe die Ergebnisse
    // für s' und s in die Zwischenergebnisse
    if (k < fn-1) {
        formation = formation + "s'" + "<sub><small>" + (k+1) + "</small></sub>" +
                " = " + Double.toString(Math.round(A*10000)/10000.0) + " mm<br>";
        D2 = Double.parseDouble(B5.getText().toString());
        C = S_1 - D2;
        formation = formation + "s" + "<sub><small>" + (k + 2) + "</small></sub>" +
                " = " + Double.toString(Math.round(C * 10000) / 10000.0) +
                " mm<br>";
        Ausgabe_zwischenergebnisse = (TextView) findViewById(zwischenspeicher);
        Ausgabe_zwischenergebnisse.setText(Html.fromHtml(formation));
    }

    // So lange nicht die letzte Fläche erreicht wurde, berechne den neuen
    // Objektabstand A "s" mit Hilfe des Abstandes zwischen die Flächen D1 "d"
        if (k < fn - 1) {
            double D1 = Double.parseDouble(B5.getText().toString());
            A = S_1 - D1;
            f_1 = f_1 * (S_1/A);
        }
```

```
    }

f_1 = f_1 * A;
sH_1 = A - f_1;
// Wurde der Objektabstand manuell eingegeben, berechne ebenfalls die
// Tiefenvergrößerung, latereale Vergrößerung und Winkelvergrößerung
if (cb_s1 == 0) {
    VN1 = (EditText) findViewById(2);
    Vgr_N1 = Double.parseDouble(VN1.getText().toString());
    VNM1 = (EditText) findViewById(((fn-1)*5)+3);
    Vgr_Nm1 = Double.parseDouble(VNM1.getText().toString());
    Vgr = Vgr * (Vgr_N1/Vgr_Nm1);
    Vgr_T = (Vgr*Vgr*(Vgr_Nm1/Vgr_N1));
    Vgr_W = ((1/Vgr)*(Vgr_N1/Vgr_Nm1));
}


// Für den Fall, dass der Objektabstand -unendlich war, berechne die Lage
// der Hauptpunkte, die Knotenpunkte, sowie f und f'
if (cb_s1 == 1 && fn > 1) {
    absolut = fn*5-1;
    //Um die objektseitige Brennweite f, k und die Lage des objektseitigen
    // Hauptpunktes zu ermitteln, muss rückwärts durch das System gerechnet werden
    for (int k = fn; k > 0; k--) {
        if (k == fn) {
            Z4 = (EditText) findViewById(absolut);
            Z_R1 = Double.parseDouble(Z4.getText().toString());
            Z_R1 = -Z_R1;
            absolut = absolut - 1;

            Z3 = (EditText) findViewById(absolut);
            Z_N1 = Double.parseDouble(Z3.getText().toString());
            absolut = absolut - 3;

            Z1 = (EditText) findViewById(absolut);
            Z_D = Double.parseDouble(Z1.getText().toString());
            absolut = absolut - 2;

            Z2 = (EditText) findViewById(absolut);
            Z_N2 = Double.parseDouble(Z2.getText().toString());
            absolut = absolut + 1;

            Z_S_1 = ((Z_N2-Z_N1)/Z_R1)/Z_N2;
            Z_S_1=1/Z_S_1;
            Z_B = Z_N2;
            Z_A = Z_S_1-Z_D;
            f = Z_S_1/Z_A;

        } else if (k > 1) {
            Z4 = (EditText) findViewById(absolut);
            Z_R1 = Double.parseDouble(Z4.getText().toString());
            Z_R1 = -Z_R1;
            absolut = absolut - 4;

            Z1 = (EditText) findViewById(absolut);
            Z_D = Double.parseDouble(Z1.getText().toString());
            absolut = absolut - 2;

            Z2 = (EditText) findViewById(absolut);
            Z_N2 = Double.parseDouble(Z2.getText().toString());
            absolut = absolut + 1;

            Z_S_1 = ((Z_B/Z_A)+((Z_N2-Z_B)/Z_R1))/Z_N2;
            Z_S_1 = 1 / Z_S_1;
            Z_B = Z_N2;
            Z_A = Z_S_1 - Z_D;
            f = f * (Z_S_1/Z_A);

        } else if (k == 1) {
            Z4 = (EditText) findViewById(absolut);
            Z_R1 = Double.parseDouble(Z4.getText().toString());
            Z_R1 = -Z_R1;
            absolut = absolut - 2;

            Z2 = (EditText) findViewById(absolut);
            Z_N2 = Double.parseDouble(Z2.getText().toString());

            Z_S_1 = ((Z_B/Z_A)+((Z_N2-Z_B)/Z_R1))/Z_N2;
            Z_S_1 = 1 / Z_S_1;
            f = f * Z_S_1;
            Z_S_1 = -Z_S_1;
            f = -f;
            sH = Z_S_1 - f;

            knoten = f + f_1;
            knoten_1 = f_1 + f;

            VNM1 = (EditText) findViewById(((fn-1)*5)+3);
```

```
                    Vgr_Nm1 = Double.parseDouble(VNM1.getText().toString());
                    D_Ges = (Vgr_Nm1/(f_1/1000));
                }
            }
        }



        Ausgabe = (TextView) findViewById(0);


        if (cb_s1 == 1) {
            if (fn == 1) {
                Ausgabe.setText(Html.fromHtml("s'<sub><small>f'</small></sub> = " +
                        Double.toString(Math.round(A * 10000.0) / 10000.0) + " mm"));
            } else {
                Ausgabe.setText(Html.fromHtml("Flächen: " + fn +
                        "<br>s'<sub><small>f'</small></sub> = " +
                        Double.toString(Math.round(A * 10000.0) / 10000.0) +
                        " mm<br>" + "f' = " + Double.toString(Math.round(
                                f_1 * 10000.0) / 10000.0) +" mm<br>" +
                        "D<sub><small>Ges</small></sub> = " +
                        Double.toString(Math.round(D_Ges * 10000.0) / 10000.0) +
                        " dpt<br>" + "s'<sub><small>H'</small></sub> = " +
                        Double.toString(Math.round(sH_1 * 10000.0) / 10000.0) +
                        " mm<br>" + "k' = " +
                        Double.toString(Math.round(knoten_1 * 10000.0) / 10000.0) +
                        " mm<br><br>" + "s<sub><small>f</small></sub> = " +
                        Double.toString(Math.round(Z_S_1 * 10000.0) / 10000.0) +
                        " mm<br>" + "f = " +
                        Double.toString(Math.round(f * 10000.0) / 10000.0) +
                        " mm<br>" + "s<sub><small>H</small></sub> = " +
                        Double.toString(Math.round(sH * 10000.0) / 10000.0) +
                        " mm<br>" + "k = " +
                        Double.toString(Math.round(knoten * 10000.0) / 10000.0) + " mm"));
                Ausgabe_zwischenergebnisse = (TextView) findViewById(zwischenspeicher);

            }
        } else {
            if (fn == 1) {
                Ausgabe.setText(Html.fromHtml("s'<sub><small>" + fn + "</small></sub> = " +
                        Double.toString(Math.round(A * 10000.0) / 10000.0) +
                        " mm<br><br>" + "ß = " +
                        Double.toString(Math.round(Vgr * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>t</small></sub> = " +
                        Double.toString(Math.round(Vgr_T * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>w</small></sub> = " +
                        Double.toString(Math.round(Vgr_W * 10000.0) / 10000.0)));
            } else {
                Ausgabe.setText(Html.fromHtml("Flächen: " + fn + "<br>s'<sub><small>" +
                        fn + "</small></sub> = " +
                        Double.toString(Math.round(A * 10000.0) / 10000.0) +
                        " mm<br><br>" + "ß = " +
                        Double.toString(Math.round(Vgr * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>t</small></sub> = " +
                        Double.toString(Math.round(Vgr_T * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>w</small></sub> = " +
                        Double.toString(Math.round(Vgr_W * 10000.0) / 10000.0)));
                Ausgabe_zwischenergebnisse = (TextView) findViewById(zwischenspeicher);
            }
        }
    }
    });
}

    }
```

h) Programmtext – Exaktes_Raytracing_Start.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;


public class Exaktes_Raytracing_Start extends AppCompatActivity {

    final Context context = this;
    private EditText et;
    private Button bt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.exaktes_raytracing_start);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        et = (EditText) findViewById(R.id.eT);
        bt = (Button) findViewById(R.id.bT);
        bt.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String enteredText = et.getText().toString();
                if (enteredText.equals("")) {
                    enteredText = "1";
                }

                // Beschränkt die Berechnung auf maximal 1000 Flächen und gibt bei einer
                // größeren Zahl einen Warnhinweis
                int Overload = Integer.parseInt(enteredText);
                if (Overload > 1000) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Obere Grenze");
                    AlarmBox.setMessage("Es können maximal 1000 Flächen berechnet werden");
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    return;
                }

                // Die Bundle Funktion ermöglicht die Weitergabe der Anzahl der zu
                // berechnenden Flächen an die nächste Activity
                Bundle korb3 = new Bundle();
                korb3.putString("datenpaket3", enteredText);
                Intent in = new Intent(Exaktes_Raytracing_Start.this,
                        Exaktes_Raytracing_Berechnung.class);
                in.putExtras(korb3);

                startActivity(in);

                }
        });
    }
}
```

i) Programmtext – Exaktes_Raytracing_Berechnung.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.app.ActionBar.LayoutParams;
import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.text.InputType;
import android.util.TypedValue;
import android.view.Gravity;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import static android.util.TypedValue.COMPLEX_UNIT_DIP;


public class Exaktes_Raytracing_Berechnung extends AppCompatActivity {

    final Context context = this;
    EditText B1, B1_1, B2,B3,B4,B5,Z1,Z2,Z3,Z4,VN1,VNM1;
    TextView Ausgabe, Ausgabe_zwischenergebnisse;
    double S,S_1,S_2,N1,N2,N5,R1,D1,S1,S2,A,B,C,D2,f_1,D,f,E,Z_R1,Z_N1,Z_N2,Z_D,Z_S_1,Z_B,Z_A,sH,
            sH_1,knoten,knoten_1,Vgr,Vgr_T,Vgr_W,Vgr_N1,Vgr_Nm1,D_Ges;
    double SinE1, SinE1_1, U1_1, Y;
    int zwischenspeicher;
    String formation;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.exaktes_raytracing_berechnung);

        // Empfängt die gesendeten Inhalte aus der Bundle Funktion
        Bundle zielkorb = getIntent().getExtras();
        String text2 = zielkorb.getString("datenpaket3");
        final int fn = Integer.parseInt(text2);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        ScrollView sv = new ScrollView(this);
        LinearLayout ll = new LinearLayout(this);

        ll.setOrientation(LinearLayout.VERTICAL);

        final LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
        params.setMargins(200,0,0,0);

        sv.addView(ll);

        int k = 1;

        // Erstellt an dieser Stelle die TextView und EditText Elemente die zur Eingabe benötigt
        // werden und weißt jedem Element eine eigene ID zu
        for (int i = 0; i < fn; i++) {
            TextView tV0 = new TextView(this);
            tV0.setPadding(20,0,0,0);
            tV0.setLines(3);
            tV0.setGravity(Gravity.CENTER_VERTICAL);
            tV0.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);
            TextView tVa = new TextView(this);
            tVa.setPadding(100,0,0,5);
            TextView tVa2 = new TextView(this);
            tVa2.setPadding(100,0,0,5);
            TextView tVb = new TextView(this);
            tVb.setPadding(100,0,0,5);
            TextView tVc = new TextView(this);
            tVc.setPadding(100,0,0,5);
            TextView tVd = new TextView(this);
            tVd.setPadding(100,0,0,5);
            TextView tVe = new TextView(this);
            tVe.setPadding(100,0,0,5);
            TextView tVf = new TextView(this);
            tVf.setLines(3);
            tVf.setPadding(20,0,0,0);
            tVf.setGravity(Gravity.CENTER_VERTICAL);
```

```
tVf.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);


EditText et1 = new EditText(this);
et1.setLayoutParams(params);
et1.setPadding(20,0,20,20);
et1.setEms(4);
et1.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et1.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_SIGNED |
        InputType.TYPE_NUMBER_FLAG_DECIMAL);
et1.setId(k);
k = k + 1;
EditText et1_1 = new EditText(this);
et1_1.setLayoutParams(params);
et1_1.setPadding(20,0,20,20);
et1_1.setEms(4);
et1_1.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et1_1.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_SIGNED |
        InputType.TYPE_NUMBER_FLAG_DECIMAL);
et1_1.setId(k);
k = k + 1;
EditText et2 = new EditText(this);
et2.setLayoutParams(params);
et2.setPadding(20,0,20,20);
et2.setEms(4);
et2.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et2.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et2.setId(k);
k = k + 1;
EditText et3 = new EditText(this);
et3.setLayoutParams(params);
et3.setPadding(20,0,20,20);
et3.setEms(4);
et3.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et3.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et3.setId(k);
k = k + 1;
EditText et4 = new EditText(this);
et4.setLayoutParams(params);
et4.setPadding(20,0,20,20);
et4.setEms(4);
et4.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et4.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_SIGNED |
        InputType.TYPE_NUMBER_FLAG_DECIMAL);
et4.setId(k);
k = k + 1;
EditText et5 = new EditText(this);
et5.setLayoutParams(params);
et5.setPadding(20,0,20,20);
et5.setEms(4);
et5.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et5.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et5.setId(k);
k = k + 1;

tV0.setText("Fläche: " + (i + 1));
// Benötigt vor der ersten Fläche die Felder für den Objektabstand s, den Brechungsindex
// n und den Winkel u des einfallenden Lichtstrahls
if (i == 0) {
    tVa.setText(Html.fromHtml("s" + "<sub><small>" + (i + 1) + "</small></sub> [mm]"));
    tVa2.setText(Html.fromHtml("u" + "<sub><small>" + (i + 1) + "</small></sub> [°]"));
    tVb.setText(Html.fromHtml("n" + "<sub><small>" + (i + 1) + "</small></sub>"));
}
tVc.setText(Html.fromHtml("n" + "<sub><small>" + (i + 2) + "</small></sub>"));
tVd.setText(Html.fromHtml("r" + "<sub><small>" + (i + 1) + "</small></sub> [mm]"));
tVe.setText(Html.fromHtml("d" + "<sub><small>" + (i + 1) + "</small></sub> [mm]"));
tVf.setText("Abstand zwischen Fläche " + (i+1) + " & " + (i+2));


ll.addView(tV0);

if (i == 0) {

    ll.addView(tVa);
    ll.addView(et1);
    ll.addView(tVa2);
    ll.addView(et1_1);

    ll.addView(tVb);
    ll.addView(et2);
}
ll.addView(tVc);
ll.addView(et3);
ll.addView(tVd);
ll.addView(et4);
//Lässt das Eingabefeld und die Beschreibung für den Abstand zwischen den Flächen "d"
// nach der letzten Fläche weg
if (i < fn - 1) {
```

```
            ll.addView(tVf);
            ll.addView(tVe);
            ll.addView(et5);
        }
    }

    Button bt = new Button(this);
    bt.setText("Berechnen");
    bt.setAllCaps(false);
    ll.addView(bt);

    //Erstellt das das finale TextView in der das Ergebnis anschließend dargestellt wird
    final TextView tv = new TextView(this);
    tv.setText("");
    tv.setId(0);
    tv.setVisibility(View.GONE);
    tv.setWidth(200);
    tv.setTextSize(COMPLEX_UNIT_DIP, 20);
    tv.setPadding(100,50,0,50);
    ll.addView(tv);

        //Erstellt das TextView in der die Zwischenergebnisse fargestellt werden
        final TextView zwischenergebnisse = new TextView(this);
        zwischenspeicher = fn*6+1;
        zwischenergebnisse.setId(zwischenspeicher);
        zwischenergebnisse.setMaxLines(fn*3-2);
        zwischenergebnisse.setPadding(15,0,0,15);
        zwischenergebnisse.setVisibility(View.GONE);
        zwischenergebnisse.setTextSize(COMPLEX_UNIT_DIP, 17);
        ll.addView(zwischenergebnisse);


    this.setContentView(sv);


    bt.setOnClickListener (new View.OnClickListener() {
        public void onClick(View v) {


            int j = 1;
            int absolut = 1;
            tv.setVisibility(View.VISIBLE);
            f_1 = 1;
            if (fn > 1) {
                zwischenergebnisse.setVisibility(View.VISIBLE);
                if (fn == 2) {
                    formation = "Zwischenergebnis: <br>";
                } else {
                    formation = "Zwischenergebnisse: <br>";
                }
            }

            // Berechnung wird für die Anzahl "fn" der Flächen durchgeführt
            for ( int k = 0; k < fn; k++) {

                // Für die erste Fläche wird zusätzlich der Objektabstand s mit eingelesen
                if (k == 0) {

                        //Abfrage ob das Eingabegeld leer ist mit Warnhinweis
                        B1 = (EditText) findViewById(j);
                        String Alarm = B1.getText().toString();
                        if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                                Alarm.equals("+")) {
                            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                            AlarmBox.setTitle("Leeres Feld!");
                            AlarmBox.setMessage(Html.fromHtml("s<sub><small>1</small></sub> " +
                                    "ist leer<br>"));
                            AlarmBox.setNeutralButton("OK", null);

                            AlertDialog dialog = AlarmBox.create();
                            dialog.show();
                            tv.setVisibility(View.GONE);
                            zwischenergebnisse.setVisibility(View.GONE);
                            return;
                        }
                        double S1 = Double.parseDouble(B1.getText().toString());
                        j = j + 1;

                    B1_1= (EditText) findViewById(j);
                     Alarm = B1_1.getText().toString();
                    if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                            Alarm.equals("+")) {
                        AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                        AlarmBox.setTitle("Leeres Feld!");
                        AlarmBox.setMessage(Html.fromHtml("u<sub><small>1</small></sub> " +
                                "ist leer<br>"));
                        AlarmBox.setNeutralButton("OK", null);
```

```
            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double U1 = Double.parseDouble(B1_1.getText().toString());
        j = j + 1;

            B2 = (EditText) findViewById(j);
            Alarm = B2.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("n<sub><small>1</small></sub> " +
                        "ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double N1 = Double.parseDouble(B2.getText().toString());
            j = j + 1;
            B3 = (EditText) findViewById(j);
            Alarm = B3.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("n<sub><small>2</small></sub> " +
                        "ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double N2 = Double.parseDouble(B3.getText().toString());
            j = j + 1;
            B4 = (EditText) findViewById(j);
            Alarm = B4.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("r<sub><small>1</small></sub> " +
                        "ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double R1 = Double.parseDouble(B4.getText().toString());
            j = j + 1;

        // So lange noch nicht die letzte Fläche erreicht wurde, wird der
        // Abstand zwischen den Flächen "d" eingelesen
            if (k < fn - 1) {
                B5 = (EditText) findViewById(j);
                Alarm = B5.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                        Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("d<sub><small>1</small>" +
                            "</sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double D1 = Double.parseDouble(B5.getText().toString());
                j = j + 1;
            }

        SinE1 = ((S1 / R1) - 1) *  Math.sin(Math.toRadians(U1));
```

```
        SinE1_1 = (N1/N2) * SinE1;
        U1_1 = U1 + Math.toDegrees(Math.asin(SinE1)) -
                Math.toDegrees(Math.asin(SinE1_1));

        S_1 = R1 * (1 + (SinE1_1/Math.sin(Math.toRadians(U1_1))));

            B = N2;

          Vgr = (S_1)/S1;
    } else {
        j = j + 3;
        B3 = (EditText) findViewById(j);
        String Alarm = B3.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("n<sub><small>" + (k+2) +
                    "</small></sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double N2 = Double.parseDouble(B3.getText().toString());
        j = j + 1;
        B4 = (EditText) findViewById(j);
        Alarm = B4.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("r<sub><small>" + (k+1) +
                    "</small></sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double R1 = Double.parseDouble(B4.getText().toString());
        j = j + 1;
        if (k < fn - 1 ) {
            B5 = (EditText) findViewById(j);
            Alarm = B5.getText().toString();
            if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("d<sub><small>" + (k+1) +
                        "</small></sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double D1 = Double.parseDouble(B5.getText().toString());
            j = j + 1;

        }
        SinE1 = ((A / R1) - 1) * Math.sin(Math.toRadians(U1_1));
        SinE1_1 = (B/N2) * SinE1;
        U1_1 = U1_1 + Math.toDegrees(Math.asin(SinE1)) -
                Math.toDegrees(Math.asin(SinE1_1));

        S_1 = R1 * (1 + (SinE1_1/Math.sin(Math.toRadians(U1_1))));

        B = N2;

        Vgr = Vgr * ((S_1)/A);

    }

    Y = U1_1;

    A = S_1;

    // Auflistung der Zwischenergebnisse für s', u' und s
    if (k < fn-1) {
        formation = formation + "s'" + "<sub><small>" + (k+1) + "</small></sub>" +
```

```
                        " = " + Double.toString(Math.round(A*10000)/10000.0) + " mm<br>";
                formation = formation + "u'" + "<sub><small>" + (k+1) + "</small></sub>" +
                        " = " + Double.toString(Math.round(Y*10000)/10000.0) + " °<br>";
                D2 = Double.parseDouble(B5.getText().toString());
                C = S_1 - D2;
                formation = formation + "s" + "<sub><small>" + (k + 2) + "</small></sub>" +
                        " = " + Double.toString(Math.round(C * 10000) / 10000.0) +
                        " mm<br>";
                Ausgabe_zwischenergebnisse = (TextView) findViewById(zwischenspeicher);
                Ausgabe_zwischenergebnisse.setText(Html.fromHtml(formation));
            }

                if (k < fn - 1) {
                    double D1 = Double.parseDouble(B5.getText().toString());
                    A = S_1 - D1;
                }

            }

        // Berechne die Tiefenvergrößerung, latereale Vergrößerung und
        // Winkelvergrößerung
            VN1 = (EditText) findViewById(2);
            Vgr_N1 = Double.parseDouble(VN1.getText().toString());
            VNM1 = (EditText) findViewById(((fn-1)*6)+4);
            Vgr_Nm1 = Double.parseDouble(VNM1.getText().toString());
            Vgr = Vgr * (Vgr_N1/Vgr_Nm1);
            Vgr_T = (Vgr*Vgr*(Vgr_Nm1/Vgr_N1));
            Vgr_W = ((1/Vgr)*(Vgr_N1/Vgr_Nm1));


        Ausgabe = (TextView) findViewById(0);



            if (fn == 1) {
                Ausgabe.setText(Html.fromHtml("s'<sub><small>" + fn + "</small></sub> = " +
                        Double.toString(Math.round(A * 10000.0) / 10000.0) + " mm<br>" +
                        "u'<sub><small>" + fn + "</small></sub> = " +
                        Double.toString(Math.round(Y * 10000.0) / 10000.0) + " °<br><br>" +
                        "ß = " + Double.toString(Math.round(Vgr * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>t</small></sub> = " +
                        Double.toString(Math.round(Vgr_T * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>w</small></sub> = " +
                        Double.toString(Math.round(Vgr_W * 10000.0) / 10000.0)));
            } else {
                Ausgabe.setText(Html.fromHtml("Flächen: " + fn + "<br>s'<sub><small>" +
                        fn + "</small></sub> = " +
                        Double.toString(Math.round(A * 10000.0) / 10000.0) +
                        " mm<br>" + "u'<sub><small>" + fn + "</small></sub> = " +
                        Double.toString(Math.round(Y * 10000.0) / 10000.0) +
                        " °<br><br>" + "ß = " +
                        Double.toString(Math.round(Vgr * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>t</small></sub> = " +
                        Double.toString(Math.round(Vgr_T * 10000.0) / 10000.0) +
                        "<br>ß<sub><small>w</small></sub> = " +
                        Double.toString(Math.round(Vgr_W * 10000.0) / 10000.0)));
                Ausgabe_zwischenergebnisse = (TextView) findViewById(zwischenspeicher);
            }
        }

    });
}

    }
```

j) Programmtext – Powervektoren.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.InputFilter;
import android.text.Spanned;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;

import static de.HS_Aalen.OptikFormelrechner.R.id.Glas_1;
import static de.HS_Aalen.OptikFormelrechner.R.id.Glas_2;


public class Powervektoren extends AppCompatActivity {

    EditText sph1, sph2, cyl1, cyl2, achse1, achse2;
    TextView sph3, cyl3, achse3, Glas1, Glas2;
    int Glaszaehler = 3;

    // Erstellt einen Filter der die Eingabe der Achsen auf 0-179° beschränkt
    public class InputFilterMinMax implements InputFilter {
        private int min;
        private int max;

        public InputFilterMinMax(int min, int max) {
            this.min = min;
            this.max = max;
        }

        @Override
        public CharSequence filter(CharSequence source, int start, int end, Spanned dest,
                                   int dstart, int dend) {

            try {
                double input = Double.parseDouble(dest.subSequence(0, dstart).toString() +
                        source + dest.subSequence(dend, dest.length()));
                if (isInRange(min, max, input))
                    return null;
            } catch (NumberFormatException nfe) { }
            return "";
        }

        private boolean isInRange(double a, double b, double c) {
            return b > a ? c >= a && c <= b : c >= b && c <= a;
        }
    }

    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.powervektoren);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        sph1 = (EditText) findViewById(R.id.in_sph_1);
        sph2 = (EditText) findViewById(R.id.in_sph_2);
        cyl1 = (EditText) findViewById(R.id.in_cyl_1);
        cyl2 = (EditText) findViewById(R.id.in_cyl_2);
        achse1 = (EditText) findViewById(R.id.in_achse_1);
        achse1.setFilters(new InputFilter[]{new InputFilterMinMax(0, 179)});
        achse2 = (EditText) findViewById(R.id.in_achse_2);
        achse2.setFilters(new InputFilter[]{new InputFilterMinMax(0, 179)});

        sph3 = (TextView) findViewById(R.id.out_sph_3);
        cyl3 = (TextView) findViewById(R.id.out_cyl_3);
        achse3 = (TextView) findViewById(R.id.out_achse_3);

        Glas1 = (TextView) findViewById(Glas_1);
        Glas2 = (TextView) findViewById(Glas_2);

        Button Berechnen = (Button)findViewById(R.id.button);
        final Button Clear = (Button)findViewById(R.id.bt_clear);
        final Button plus_glas = (Button)findViewById(R.id.bt_plus);

        Berechnen.setOnClickListener(
                new Button.OnClickListener(){
                    public void onClick(View v) {
                        String Alarm = sph2.getText().toString();
                        String Alarm2 = cyl2.getText().toString();
                        if (Alarm.equals("") && Alarm2.equals("")) {
```

```
            return;
        } else {
            Alarm = sph1.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                sph1.setText("0");
            }
            double b_sph1 = Double.parseDouble(sph1.getText().toString());
            Alarm = sph2.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                sph2.setText("0");
            }
            double b_sph2 = Double.parseDouble(sph2.getText().toString());
            Alarm = cyl1.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                cyl1.setText("0");
            }
            double b_cyl1 = Double.parseDouble(cyl1.getText().toString());
            Alarm = cyl2.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                cyl2.setText("0");
            }
            double b_cyl2 = Double.parseDouble(cyl2.getText().toString());
            Alarm = achse1.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                achse1.setText("0");
            }
            double b_achse1 = Double.parseDouble(achse1.getText().toString());
            Alarm = achse2.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                achse2.setText("0");
            }
            double b_achse2 = Double.parseDouble(achse2.getText().toString());

            if (b_cyl1 < 0) {
                b_sph1 = b_sph1 + b_cyl1;
                b_cyl1 = b_cyl1 * (-1);
                b_achse1 = b_achse1 + 90;
                if (b_achse1 > 179) {
                    b_achse1 = b_achse1 - 180;
                }
            }

            if (b_cyl2 < 0) {
                b_sph2 = b_sph2 + b_cyl2;
                b_cyl2 = b_cyl2 * (-1);
                b_achse2 = b_achse2 + 90;
                if (b_achse2 > 179) {
                    b_achse2 = b_achse2 - 180;
                }
            }
            double M1 = b_sph1 + (b_cyl1 / 2);
            double M2 = b_sph2 + (b_cyl2 / 2);
            double M = M1 + M2;

            double rad_achse1 = Math.toRadians(b_achse1 * 2);
            double rad_achse2 = Math.toRadians(b_achse2 * 2);

            double I0_1 = -(b_cyl1 / 2) * Math.cos(rad_achse1);
            double I0_2 = -(b_cyl2 / 2) * Math.cos(rad_achse2);
            double I0 = I0_1 + I0_2;

            double I45_1 = -(b_cyl1 / 2) * Math.sin(rad_achse1);
            double I45_2 = -(b_cyl2 / 2) * Math.sin(rad_achse2);

            double I45 = I45_1 + I45_2;

            double b_cyl3 = 2 * Math.sqrt(Math.pow(I0, 2) + Math.pow(I45, 2));
            b_cyl3 = Math.round(b_cyl3 * 10000) / 10000.0;
            double b_sph3 = M - (b_cyl3 / 2);
            b_sph3 = Math.round(b_sph3 * 10000) / 10000.0;
            double b_achse3 = (Math.atan(I45 / I0)) / 2;

            if (b_cyl3 == 0) {
                achse3.setText("0");
            } else {
                b_achse3 = Math.round(Math.toDegrees(b_achse3) * 10000) / 10000.0;
            }

            if (I0 > 0) {
                b_achse3 = b_achse3 + 90;
            }
```

```
                        if (b_achse3 < 0) {
                            b_achse3 = b_achse3 + 180;
                        }

                        sph3.setText(Double.toString(b_sph3));
                        String Ausgabe = sph3.getText().toString();
                        if (Ausgabe.length() > 9) {
                            sph3.setText(String.format("%9.4e", b_sph3).replace(",","."));
                        }

                        cyl3.setText(Double.toString(b_cyl3));
                        Ausgabe = cyl3.getText().toString();
                        if (Ausgabe.length() > 9) {
                            cyl3.setText(String.format("%9.4e" , b_cyl3).replace(",","."));
                        }
                        if (b_cyl3 != 0) {
                            achse3.setText(Double.toString(b_achse3));
                        }

                    }
                }
            }
        );
        Clear.setOnClickListener(
                new Button.OnClickListener() {
                    public void onClick(View v) {
                        sph1.setEnabled(true);
                        sph1.setText("");
                        sph2.setText("");
                        sph3.setText("");

                        cyl1.setEnabled(true);
                        cyl1.setText("");
                        cyl2.setText("");
                        cyl3.setText("");

                        achse1.setEnabled(true);
                        achse1.setText("");
                        achse1.setFilters(new InputFilter[]{new InputFilterMinMax(0, 179)});
                        achse2.setText("");
                        achse2.setFilters(new InputFilter[]{new InputFilterMinMax(0, 179)});
                        achse3.setText("");

                        plus_glas.setText("+ 3. Glas");
                        Glaszaehler = 3;
                        Glas1.setText("Glas 1");
                        Glas2.setText("Glas 2");
                    }
                }
        );
        plus_glas.setOnClickListener(
                new Button.OnClickListener() {
                    public void onClick(View v) {
                        String Alarm = sph3.getText().toString();
                        if (Alarm.isEmpty()) {
                            return;
                        } else {
                            sph1.setEnabled(true);
                            sph1.setText(sph3.getText());
                            sph1.setEnabled(false);
                            cyl1.setEnabled(true);
                            cyl1.setText(cyl3.getText());
                            cyl1.setEnabled(false);
                            achse1.setEnabled(true);
                            achse1.setFilters(new InputFilter[] {});
                            achse1.setText(achse3.getText());
                            achse1.setFilters(new InputFilter[]{new InputFilterMinMax(0, 179)});
                            achse1.setEnabled(false);
                            sph2.setText("");
                            sph3.setText("");
                            cyl2.setText("");
                            cyl3.setText("");
                            achse2.setText("");
                            achse2.setFilters(new InputFilter[]{new InputFilterMinMax(0, 179)});
                            achse3.setText("");
                            Glas1.setText("Berechnetes Glas");
                            Glas2.setText("Glas " + Glaszaehler);
                            Glaszaehler = Glaszaehler + 1;
                            plus_glas.setText("+ " + Glaszaehler + ". Glas");

                        }

                    }
                }
        );
    }
}
```

k) Programmtext – Schott.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.view.View;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;


public class Schott extends AppCompatActivity {

    Spinner Auswahlglas, Farbwahl;
    ArrayAdapter<CharSequence> adapter;


    final Context context = this;
    EditText et_wellenlaenge, et_a0, et_a1, et_a2, et_a3, et_a4, et_a5,  et;
    TextView n, txt_result;
    Button bt;
    double a0, a1, a2, a3, a4, a5, lambda, result;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.schott);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        Auswahlglas = (Spinner) findViewById(R.id.Auswahlglas);
        adapter = ArrayAdapter.createFromResource(this, R.array.glasarten,
                android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Auswahlglas.setAdapter(adapter);
        Auswahlglas.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {

                et_a0 = (EditText) findViewById(R.id.et_a0);
                et_a1 = (EditText) findViewById(R.id.et_a1);
                et_a2 = (EditText) findViewById(R.id.et_a2);
                et_a3 = (EditText) findViewById(R.id.et_a3);
                et_a4 = (EditText) findViewById(R.id.et_a4);
                et_a5 = (EditText) findViewById(R.id.et_a5);
                if (position == 0) {
                    et_a0.setText("2.4218304");
                    et_a1.setText("-0.0092167103");
                    et_a2.setText("0.013821685");
                    et_a3.setText("0.00032955714");
                    et_a4.setText("-0.0000121536 41");
                    et_a5.setText("0.0000010333767");

                    et_a0.setEnabled(false);
                    et_a1.setEnabled(false);
                    et_a2.setEnabled(false);
                    et_a3.setEnabled(false);
                    et_a4.setEnabled(false);
                    et_a5.setEnabled(false);
                } else if (position == 1) {
                    et_a0.setText("2.2718929");
                    et_a1.setText("-0.010108077");
                    et_a2.setText("0.010592509");
                    et_a3.setText("0.00020816965");
                    et_a4.setText("-0.0000076472538");
                    et_a5.setText("0.00000049240991");

                    et_a0.setEnabled(false);
                    et_a1.setEnabled(false);
                    et_a2.setEnabled(false);
                    et_a3.setEnabled(false);
                    et_a4.setEnabled(false);
                    et_a5.setEnabled(false);
                } else if (position == 2) {
                    et_a0.setText("2.2850299");
                    et_a1.setText("-0.0086010725");
                    et_a2.setText("0.011806783");
```

```
                et_a3.setText("0.00020765657");
                et_a4.setText("-0.0000021314913");
                et_a5.setText("0.000000321312345");

                et_a0.setEnabled(false);
                et_a1.setEnabled(false);
                et_a2.setEnabled(false);
                et_a3.setEnabled(false);
                et_a4.setEnabled(false);
                et_a5.setEnabled(false);
            } else if (position == 3) {
                et_a0.setText("2.8784725");
                et_a1.setText("-0.0105656453");
                et_a2.setText("0.033279420");
                et_a3.setText("0.0020551378");
                et_a4.setText("-0.00011396226");
                et_a5.setText("0.000016340021");

                et_a0.setEnabled(false);
                et_a1.setEnabled(false);
                et_a2.setEnabled(false);
                et_a3.setEnabled(false);
                et_a4.setEnabled(false);
                et_a5.setEnabled(false);
            } else {
                et_a0.setText("");
                et_a1.setText("");
                et_a2.setText("");
                et_a3.setText("");
                et_a4.setText("");
                et_a5.setText("");

                et_a0.setEnabled(true);
                et_a1.setEnabled(true);
                et_a2.setEnabled(true);
                et_a3.setEnabled(true);
                et_a4.setEnabled(true);
                et_a5.setEnabled(true);
            }

        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });
    Farbwahl = (Spinner) findViewById(R.id.Farbauswahl);
    adapter = ArrayAdapter.createFromResource(this, R.array.farbwahl,
            android.R.layout.simple_list_item_1);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    Farbwahl.setAdapter(adapter);
    Farbwahl.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {

            if (position == 18) {
                et_wellenlaenge = (EditText) findViewById(R.id.et_wellenlaenge);
                et_wellenlaenge.setEnabled(true);
                et_wellenlaenge.setText("");
            } else {
                et_wellenlaenge = (EditText) findViewById(R.id.et_wellenlaenge);

                if (position == 0) {
                    et_wellenlaenge.setText("365.0");
                } else if (position == 1) {
                    et_wellenlaenge.setText("404.7");
                } else if (position == 2) {
                    et_wellenlaenge.setText("435.8");
                } else if (position == 3) {
                    et_wellenlaenge.setText("480.0");
                } else if (position == 4) {
                    et_wellenlaenge.setText("486.1");
                } else if (position == 5) {
                    et_wellenlaenge.setText("546.1");
                } else if (position == 6) {
                    et_wellenlaenge.setText("587.6");
                } else if (position == 7) {
                    et_wellenlaenge.setText("589.3");
                } else if (position == 8) {
                    et_wellenlaenge.setText("632.8");
                } else if (position == 9) {
                    et_wellenlaenge.setText("643.8");
                } else if (position == 10) {
                    et_wellenlaenge.setText("656.3");
                } else if (position == 11) {
                    et_wellenlaenge.setText("706.5");
                } else if (position == 12) {
```

```
                et_wellenlaenge.setText("852.1");
            } else if (position == 13) {
                et_wellenlaenge.setText("1014.0");
            } else if (position == 14) {
                et_wellenlaenge.setText("1060.0");
            } else if (position == 15) {
                et_wellenlaenge.setText("1529.6");
            } else if (position == 16) {
                et_wellenlaenge.setText("1970.1");
            } else if (position == 17) {
                et_wellenlaenge.setText("2325.4");
            }
            et_wellenlaenge.setEnabled(false);
            n = (TextView) findViewById(R.id.txt_n);
            n.setText(Html.fromHtml("n<sub><small>" + et_wellenlaenge.getText().toString() +
                    "</small></sub>:"));

        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});

bt = (Button) findViewById(R.id.bT);
bt.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {

        String Alarm = et_a0.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("A<sub><small>0</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            return;
        }
        Alarm = et_a1.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("A<sub><small>1</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            return;
        }
        Alarm = et_a2.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("A<sub><small>2</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            return;
        }
        Alarm = et_a3.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("A<sub><small>3</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            return;
        }
        Alarm = et_a4.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("A<sub><small>4</small>" +
```

```
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            return;
        }
        Alarm = et_a5.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("A<sub><small>5</small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            return;
        }

        Alarm = et_wellenlaenge.getText().toString();
        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("Das Feld für die Wellenlänge ist leer"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            return;
        }

        lambda = Double.parseDouble(et_wellenlaenge.getText().toString());
        n.setText(Html.fromHtml("n<sub><small>" +
                Double.toString(Math.round(lambda*10)/10.0) + "</small></sub>:"));
        lambda = lambda/1000;
        a0 = Double.parseDouble(et_a0.getText().toString());
        a1 = Double.parseDouble(et_a1.getText().toString());
        a2 = Double.parseDouble(et_a2.getText().toString());
        a3 = Double.parseDouble(et_a3.getText().toString());
        a4 = Double.parseDouble(et_a4.getText().toString());
        a5 = Double.parseDouble(et_a5.getText().toString());

        result = Math.sqrt(a0 + a1 * Math.pow(lambda, 2) + (a2 / Math.pow(lambda, 2)) +
                (a3 / Math.pow(lambda, 4)) + (a4 / Math.pow(lambda, 6)) +
                (a5 / Math.pow(lambda, 8)));

        result = Math.round(result * 100000) / 100000.0;

        txt_result = (TextView) findViewById(R.id.txt_result);
        txt_result.setText(Double.toString(result));

        }
    });
  }
}
```

I) Programmtext – Transmission_Start.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;


public class Transmission_Start extends AppCompatActivity {

    final Context context = this;
    private EditText et;
    private Button bt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.transmission_start);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        et = (EditText) findViewById(R.id.eT);
        bt = (Button) findViewById(R.id.bT);
        bt.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String enteredText = et.getText().toString();
                if (enteredText.equals("")) {
                    enteredText = "1";
                }

                // Beschränkt die Berechnung auf maximal 1000 Flächen und gibt bei einer
                // größeren Zahl einen Warnhinweis
                int Overload = Integer.parseInt(enteredText);
                if (Overload > 1000) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Obere Grenze");
                    AlarmBox.setMessage("Es können maximal 1000 Flächen berechnet werden");
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    return;
                }

                // Die Bundle Funktion ermöglicht die Weitergabe der Anzahl der zu berechnenden
                // Flächen an die nächste Activity
                Bundle korb4 = new Bundle();
                korb4.putString("datenpaket4", enteredText);
                Intent in = new Intent(Transmission_Start.this, Transmission_Berechnung.class);
                in.putExtras(korb4);

                startActivity(in);

                }
        });
    }
}
```

m) Programmtext – Transmission_Berechnung.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.app.ActionBar.LayoutParams;
import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.text.InputType;
import android.util.TypedValue;
import android.view.Gravity;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import static android.util.TypedValue.COMPLEX_UNIT_DIP;


public class Transmission_Berechnung extends AppCompatActivity {

    final Context context = this;
    EditText B1,B2,B3,B4,B5;
    TextView Ausgabe, Ausgabe_zwischenergebnisse, B3_Text;
    CheckBox CB;

    double Reflexion, Transmission, Ergebnis;
    double Transmission_zwischen, Reflexion_zwischen, Ergebnis_zwischen;
    double n_neu;

    int zwischenspeicher;
    String formation = "";

    int Abfrage = 1;
    int cb_check = 0;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.transmission_berechnung);

        // Einlesen der Daten aus der Bundle Funktion aus der vorigen Activity
        Bundle zielkorb = getIntent().getExtras();
        String text2 = zielkorb.getString("datenpaket4");
        final int fn = Integer.parseInt(text2);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        ScrollView sv = new ScrollView(this);
        LinearLayout ll = new LinearLayout(this);

        ll.setOrientation(LinearLayout.VERTICAL);

        final LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
        params.setMargins(200,0,0,0);

        sv.addView(ll);

        Transmission = 1;
        Reflexion = 0;
        Ergebnis =1;

        int k = 1;

        // Erstelle die Eingabefenster und Beschreibungen für die Anzahl "fn" an Flächen
        for (int i = 0; i < fn; i++) {
            TextView tV0 = new TextView(this);
            tV0.setPadding(20,0,0,0);
            tV0.setLines(3);
            tV0.setGravity(Gravity.CENTER_VERTICAL);
            tV0.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);
            TextView tV1 = new TextView(this);
            tV1.setPadding(100,0,0,5);
            TextView tV2 = new TextView(this);
            tV2.setPadding(100,0,0,5);
            TextView tV3 = new TextView(this);
            tV3.setPadding(100,0,0,35);
```

```
tV3.setId(k);
k = k + 1;
TextView tV4 = new TextView(this);
tV4.setPadding(100,15,0,25);
TextView tV5 = new TextView(this);
tV5.setPadding(100,0,0,15);
TextView tV6 = new TextView(this);
tV6.setLines(3);
tV6.setPadding(20,0,0,0);
tV6.setGravity(Gravity.CENTER_VERTICAL);
tV6.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);

// Füge eine Checkbox hinzu, ob eine Fläche entspiegelt ist
CheckBox cb_ent = new CheckBox(this);
cb_ent.setPadding(100,50,0,50);
cb_ent.setId(k);
k = k + 1;

EditText et1 = new EditText(this);
et1.setLayoutParams(params);
et1.setPadding(20,0,20,20);
et1.setEms(4);
et1.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et1.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et1.setId(k);
k = k + 1;
EditText et2 = new EditText(this);
et2.setLayoutParams(params);
et2.setPadding(20,0,20,20);
et2.setEms(4);
et2.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et2.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et2.setId(k);
k = k + 1;
EditText et3 = new EditText(this);
et3.setLayoutParams(params);
et3.setPadding(20,0,20,20);
et3.setEms(4);
et3.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et3.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et3.setId(k);
et3.setVisibility(View.GONE);
k = k + 1;
EditText et4 = new EditText(this);
et4.setLayoutParams(params);
et4.setPadding(20,0,20,20);
et4.setEms(4);
et4.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et4.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et4.setId(k);
k = k + 1;
EditText et5 = new EditText(this);
et5.setLayoutParams(params);
et5.setPadding(20,0,20,20);
et5.setEms(4);
et5.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);
et5.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_FLAG_DECIMAL);
et5.setId(k);
k = k + 1;

tV0.setText("Fläche: " + (i + 1));
// Der Brechungindex n1 vor der ersten Fläche wird nur ein einziges Mal benötigt
if (i == 0) {
    tV1.setText(Html.fromHtml("n" + "<sub><small>" + (i + 1) + "</small></sub>"));
}
tV2.setText(Html.fromHtml("n" + "<sub><small>" + (i + 2) + "</small></sub>"));
cb_ent.setText("Fläche " + (i + 1) + " ist entspiegelt");
tV3.setText(Html.fromHtml("Restreflex p" + "<sub><small>" + (i + 1) +
        "</small></sub> [%]"));
tV3.setVisibility(View.GONE);
tV4.setText(Html.fromHtml("T<sub><small>0" + "<sub><small>" + (i + 1) +
        "</small></sub></small></sub> [%]"));
tV5.setText(Html.fromHtml("d" + "<sub><small>" + (i + 1) + "</small></sub> [mm]"));
tV6.setText("Abstand zwischen Fläche " + (i+1) + " & " + (i+2));


ll.addView(tV0);
if (i == 0) {

    ll.addView(tV1);
    ll.addView(et1);
}
ll.addView(tV2);
ll.addView(et2);
ll.addView(cb_ent);
ll.addView(tV3);
ll.addView(et3);
if (i < fn - 1) {
```

```
            ll.addView(tV4);
            ll.addView(et4);
            ll.addView(tV6);
            ll.addView(tV5);
            ll.addView(et5);
        }
    }

    Button bt = new Button(this);
    bt.setText("Berechnen");
    bt.setAllCaps(false);
    ll.addView(bt);

    // Erstellen der TextView für das Endergebniss
    final TextView tv = new TextView(this);
    tv.setText("");
    tv.setId(0);
    tv.setVisibility(View.GONE);
    tv.setWidth(200);
    tv.setTextSize(COMPLEX_UNIT_DIP, 20);
    tv.setPadding(100,50,0,50);
    ll.addView(tv);

        // Erstellen der TextView für die Zwischenergebnisse
        final TextView zwischenergebnisse = new TextView(this);
        zwischenspeicher = fn*7+1;
        zwischenergebnisse.setId(zwischenspeicher);

        zwischenergebnisse.setPadding(15,0,0,15);
        zwischenergebnisse.setVisibility(View.GONE);
        zwischenergebnisse.setTextSize(COMPLEX_UNIT_DIP, 17);
        ll.addView(zwischenergebnisse);


    this.setContentView(sv);



    bt.setOnClickListener (new View.OnClickListener() {
        public void onClick(View v) {

            int k = 1;
            //Überprüfe für alle erstellten Checkboxen, ob der Haken für eine entspiegelte
            // Fläche gesetzt wurde und füge gegebenfalls das entsprechende Eingabefenster
            // hinzu um den prozentualen Restreflex zu setzen
            if (cb_check == 0) {
                for (int y = 1; y <= fn; y++) {

                    CB = (CheckBox) findViewById(k + 1);
                    if (CB.isChecked()) {
                        B3 = (EditText) findViewById(k + 4);
                        B3.setVisibility(View.VISIBLE);
                        B3_Text = (TextView) findViewById(k);
                        B3_Text.setVisibility(View.VISIBLE);
                        cb_check = 1;
                    } else {
                        CB.setEnabled(false);
                        B3 = (EditText) findViewById(k+4);
                        B3.setText("1");
                    }
                    k = k + 7;
                }
            }

            k = 1;

                // Sollte der Haken bei einer Checkbox gesetzt worden sein, dann weise den
            // Nutzer darauf hin, dass er nun den prozentualen Restreflex eingeben kann
                if (cb_check == 1) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Layout wurde erweitert!");
                    AlarmBox.setMessage(Html.fromHtml("Für die entspiegelten Flächen " +
                            "den Restreflex angeben"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    cb_check = 2;
                    return;

                } else if (cb_check == 0) {

                    if (fn > 1) {
                        zwischenergebnisse.setVisibility(View.VISIBLE);
                        if (fn == 2) {
                            formation = "Zwischenergebnis: <br>";
                        } else {
                            formation = "Zwischenergebnisse: <br>";
```

```
        }
    }

    // Berechnung der Transmission
    for (int i = 1; i <= fn; i++) {
        if (i == 1) {

            // Abfrage ob Felder alle einen Wert haben
            B1 = (EditText) findViewById(k+2);
            String Alarm = B1.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("n<sub><small>1</small>" +
                        "</sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double N1 = Double.parseDouble(B1.getText().toString());

            B2 = (EditText) findViewById(k+3);
            Alarm = B2.getText().toString();
            if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                AlarmBox.setTitle("Leeres Feld!");
                AlarmBox.setMessage(Html.fromHtml("n<sub><small>2</small>" +
                        "</sub> ist leer<br>"));
                AlarmBox.setNeutralButton("OK", null);

                AlertDialog dialog = AlarmBox.create();
                dialog.show();
                tv.setVisibility(View.GONE);
                zwischenergebnisse.setVisibility(View.GONE);
                return;
            }
            double N2 = Double.parseDouble(B2.getText().toString());

            if (i < fn) {
                B4 = (EditText) findViewById(k+5);
                Alarm = B4.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") ||
                        Alarm.equals("-") || Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox =
                            new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("T<sub><small>0<sub>" +
                            "<small>1</small></sub></small>" +
                            "</sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double T0 = Double.parseDouble(B4.getText().toString());

                B5 = (EditText) findViewById(k+6);
                Alarm = B5.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") ||
                        Alarm.equals("-") || Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox =
                            new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("d<sub><small>1</small>"+
                            "</sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double d = Double.parseDouble(B5.getText().toString());

                T0 = T0/100;
                Transmission = Math.pow(T0, (d/10));
                Transmission_zwischen = Transmission*100;
            }
```

```
                k = k + 7;

                N1 = N1/100;
                N2 = N2/100;

                Reflexion = Math.pow(((N1-N2)/(N1+N2)),2);
                Reflexion_zwischen = Reflexion*100;
                Reflexion = 1-Reflexion;

                n_neu = N2;

                Ergebnis = Transmission*Reflexion;
                Ergebnis_zwischen = Ergebnis*100;

                if (i < fn) {
                    formation = formation + "p" + "<sub><small>1</small></sub>" +
                            " = " + Double.toString(
                                    Math.round(Reflexion_zwischen*10000)/10000.0) +
                            " %<br>T<sub><small>1</small></sub> = " +
                            Double.toString(Math.round(
                                    Transmission_zwischen*10000)/10000.0) +
                            " %<br>T<sub><small>1<sub><small>Ges</small>" +
                            "</sub></small></sub> = " + Double.toString(Math.round(
                                    Ergebnis_zwischen*10000)/10000.0) +
                            " %<br><br>";
                    Ausgabe_zwischenergebnisse = (TextView)
                            findViewById(zwischenspeicher);
                    Ausgabe_zwischenergebnisse.setText(Html.fromHtml(formation));
                }

            } else {

                B2 = (EditText) findViewById(k+3);
                String Alarm = B2.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                        Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("n<sub><small>"+(i+1)+
                            "</small></sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double N2 = Double.parseDouble(B2.getText().toString());

                if (i < fn) {
                    B4 = (EditText) findViewById(k+5);
                    Alarm = B4.getText().toString();
                    if (Alarm.equals("") || Alarm.equals(".") ||
                            Alarm.equals("-") || Alarm.equals("+")) {
                        AlertDialog.Builder AlarmBox =
                                new AlertDialog.Builder(context);
                        AlarmBox.setTitle("Leeres Feld!");
                        AlarmBox.setMessage(Html.fromHtml(
                                "T<sub><small>0<sub><small>" + i +
                                        "</small></sub></small>" +
                                        "</sub> ist leer<br>"));
                        AlarmBox.setNeutralButton("OK", null);

                        AlertDialog dialog = AlarmBox.create();
                        dialog.show();
                        tv.setVisibility(View.GONE);
                        zwischenergebnisse.setVisibility(View.GONE);
                        return;
                    }
                    double T0 = Double.parseDouble(B4.getText().toString());

                    B5 = (EditText) findViewById(k+6);
                    Alarm = B5.getText().toString();
                    if (Alarm.equals("") || Alarm.equals(".") ||
                            Alarm.equals("-") || Alarm.equals("+")) {
                        AlertDialog.Builder AlarmBox =
                                new AlertDialog.Builder(context);
                        AlarmBox.setTitle("Leeres Feld!");
                        AlarmBox.setMessage(Html.fromHtml("d<sub><small>"+i+
                                "</small> ist leer<br>"));
                        AlarmBox.setNeutralButton("OK", null);

                        AlertDialog dialog = AlarmBox.create();
                        dialog.show();
                        tv.setVisibility(View.GONE);
                        zwischenergebnisse.setVisibility(View.GONE);
```

```
                        return;
                    }
                    double d = Double.parseDouble(B5.getText().toString());

                    T0 = T0/100;
                    Transmission = Math.pow(T0, (d/10));
                    Transmission_zwischen = Transmission*100;
                }

                N2 = N2/100;
                Reflexion = Math.pow(((n_neu-N2)/(n_neu+N2)),2);
                Reflexion_zwischen = Reflexion*100;
                Reflexion = 1-Reflexion;
                n_neu = N2;

                if (i < fn) {
                    Ergebnis = Ergebnis * Transmission * Reflexion;

                // Nach der letzten Fläche wird keine Transmission benötigt
                } else {
                    Ergebnis = Ergebnis * Reflexion;
                }
                Ergebnis_zwischen = Ergebnis*100;

                k = k + 7;

                // So lange die letzte Fläche nicht erreicht wurde,
                // schreibe die Ergebnisse als Zwischenergebnisse
                if (i < fn) {
                    formation = formation + "p" + "<sub><small>"+i+
                            "</small></sub>" +" = " +
                            Double.toString(Math.round(
                                    Reflexion_zwischen*10000)/10000.0) +
                            " %<br>T<sub><small>"+i+"</small></sub> = " +
                            Double.toString(Math.round(
                                    Transmission_zwischen*10000)/10000.0) +
                            " %<br>T<sub><small>"+i+"<sub><small>Ges</small>" +
                            "</sub></small></sub> = " +
                            Double.toString(Math.round(
                                    Ergebnis_zwischen*10000)/10000.0) +
                            " %<br><br>";
                    Ausgabe_zwischenergebnisse = (TextView)
                            findViewById(zwischenspeicher);
                    Ausgabe_zwischenergebnisse.setText(Html.fromHtml(formation));
                }
            }
        }
    } else if (cb_check == 2) {

        if (fn > 1) {
            zwischenergebnisse.setVisibility(View.VISIBLE);
            if (fn == 2) {
                formation = "Zwischenergebnis: <br>";
            } else {
                formation = "Zwischenergebnisse: <br>";
            }
        }

        for (int i = 1; i <= fn; i++) {

            if (i == 1) {

                B1 = (EditText) findViewById(k + 2);
                String Alarm = B1.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                        Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("n<sub><small>1</small>" +
                            "</sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double N1 = Double.parseDouble(B1.getText().toString());

                B2 = (EditText) findViewById(k + 3);
                Alarm = B2.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                        Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("n<sub><small>2</small>" +
                            "</sub> ist leer<br>"));
```

```
        AlarmBox.setNeutralButton("OK", null);

        AlertDialog dialog = AlarmBox.create();
        dialog.show();
        tv.setVisibility(View.GONE);
        zwischenergebnisse.setVisibility(View.GONE);
        return;
    }
    double N2 = Double.parseDouble(B2.getText().toString());

    B3 = (EditText) findViewById(k + 4);
    Alarm = B3.getText().toString();
    if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
            Alarm.equals("+")) {
        AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
        AlarmBox.setTitle("Leeres Feld!");
        AlarmBox.setMessage(Html.fromHtml("p<sub><small>" + i +
                "</small></sub> ist leer<br>"));
        AlarmBox.setNeutralButton("OK", null);

        AlertDialog dialog = AlarmBox.create();
        dialog.show();
        tv.setVisibility(View.GONE);
        zwischenergebnisse.setVisibility(View.GONE);
        return;
    }

    double R_flx = Double.parseDouble(B3.getText().toString());

    if (i < fn) {
        B4 = (EditText) findViewById(k + 5);
        Alarm = B4.getText().toString();
        if (Alarm.equals("") || Alarm.equals(".") ||
                Alarm.equals("-") || Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox =
                    new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("T<sub><small>0<sub>" +
                    "<small>1</small></sub></small>" +
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double T0 = Double.parseDouble(B4.getText().toString());

        B5 = (EditText) findViewById(k + 6);
        Alarm = B5.getText().toString();
        if (Alarm.equals("") || Alarm.equals(".") ||
                Alarm.equals("-") || Alarm.equals("+")) {
            AlertDialog.Builder AlarmBox =
                    new AlertDialog.Builder(context);
            AlarmBox.setTitle("Leeres Feld!");
            AlarmBox.setMessage(Html.fromHtml("d<sub><small>1</small>"+
                    "</sub> ist leer<br>"));
            AlarmBox.setNeutralButton("OK", null);

            AlertDialog dialog = AlarmBox.create();
            dialog.show();
            tv.setVisibility(View.GONE);
            zwischenergebnisse.setVisibility(View.GONE);
            return;
        }
        double d = Double.parseDouble(B5.getText().toString());

        T0 = T0/100;
        Transmission = Math.pow(T0, (d/10));
        Transmission_zwischen = Transmission*100;
    }

    k = k + 7;


    N1 = N1/100;
    N2 = N2/100;

    CB = (CheckBox) findViewById(k - 6);
    if (CB.isChecked()) {
        Reflexion = 1-(R_flx/100);
        Reflexion_zwischen = R_flx;
    } else {
        Reflexion = Math.pow(((N1 - N2) / (N1 + N2)), 2);
        Reflexion_zwischen = Reflexion * 100;
        Reflexion = 1 - Reflexion;
```

```
                }

                n_neu = N2;

                Ergebnis = Transmission*Reflexion;
                Ergebnis_zwischen = Ergebnis*100;

                if (i < fn) {
                    formation = formation + "p" + "<sub><small>1</small></sub>" +
                            " = " + Double.toString(Math.round(
                                    Reflexion_zwischen*10000)/10000.0) +
                            " %<br>T<sub><small>1</small></sub> = " +
                            Double.toString(Math.round(
                                    Transmission_zwischen*10000)/10000.0) +
                            " %<br>T<sub><small>1<sub><small>Ges</small>" +
                            "</sub></small></sub> = " +
                            Double.toString(Math.round(
                                    Ergebnis_zwischen*10000)/10000.0) +
                            " %<br><br>";
                    Ausgabe_zwischenergebnisse = (TextView)
                            findViewById(zwischenspeicher);
                    Ausgabe_zwischenergebnisse.setText(Html.fromHtml(formation));
                }

            } else {

                B2 = (EditText) findViewById(k + 3);
                String Alarm = B2.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                        Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("n<sub><small>" +
                            (i + 1) + "</small></sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double N2 = Double.parseDouble(B2.getText().toString());

                B3 = (EditText) findViewById(k + 4);
                Alarm = B3.getText().toString();
                if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                        Alarm.equals("+")) {
                    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
                    AlarmBox.setTitle("Leeres Feld!");
                    AlarmBox.setMessage(Html.fromHtml("p<sub><small>" + i +
                            "</small></sub> ist leer<br>"));
                    AlarmBox.setNeutralButton("OK", null);

                    AlertDialog dialog = AlarmBox.create();
                    dialog.show();
                    tv.setVisibility(View.GONE);
                    zwischenergebnisse.setVisibility(View.GONE);
                    return;
                }
                double R_flx = Double.parseDouble(B3.getText().toString());

                if (i < fn) {
                    B4 = (EditText) findViewById(k + 5);
                    Alarm = B4.getText().toString();
                    if (Alarm.equals("") || Alarm.equals(".") ||
                            Alarm.equals("-") || Alarm.equals("+")) {
                        AlertDialog.Builder AlarmBox =
                                new AlertDialog.Builder(context);
                        AlarmBox.setTitle("Leeres Feld!");
                        AlarmBox.setMessage(Html.fromHtml("T<sub><small>0<sub>" +
                                "<small>" + i + "</small></sub></small>" +
                                "</sub> ist leer<br>"));
                        AlarmBox.setNeutralButton("OK", null);

                        AlertDialog dialog = AlarmBox.create();
                        dialog.show();
                        tv.setVisibility(View.GONE);
                        zwischenergebnisse.setVisibility(View.GONE);
                        return;
                    }
                    double T0 = Double.parseDouble(B4.getText().toString());

                    B5 = (EditText) findViewById(k + 6);
                    Alarm = B5.getText().toString();
                    if (Alarm.equals("") || Alarm.equals(".") ||
                            Alarm.equals("-") || Alarm.equals("+")) {
                        AlertDialog.Builder AlarmBox =
```

```
                                        new AlertDialog.Builder(context);
                                AlarmBox.setTitle("Leeres Feld!");
                                AlarmBox.setMessage(Html.fromHtml("d<sub><small>" + i +
                                        "</small> ist leer<br>"));
                                AlarmBox.setNeutralButton("OK", null);

                                AlertDialog dialog = AlarmBox.create();
                                dialog.show();
                                tv.setVisibility(View.GONE);
                                zwischenergebnisse.setVisibility(View.GONE);
                                return;
                            }
                            double d = Double.parseDouble(B5.getText().toString());

                            T0 = T0/100;
                            Transmission = Math.pow(T0, (d/10));
                            Transmission_zwischen = Transmission*100;
                        }

                        k = k + 7;

                        N2 = N2/100;

                        CB = (CheckBox) findViewById(k - 6);
                        if (CB.isChecked()) {
                            Reflexion = 1-(R_flx/100);
                            Reflexion_zwischen = R_flx;
                        } else {
                            Reflexion = Math.pow(((n_neu - N2) / (n_neu + N2)), 2);
                            Reflexion_zwischen = Reflexion * 100;
                            Reflexion = 1 - Reflexion;
                        }
                        n_neu = N2;

                        if (i < fn) {
                            Ergebnis = Ergebnis * Transmission * Reflexion;
                        } else {
                            Ergebnis = Ergebnis * Reflexion;
                        }
                        Ergebnis_zwischen = Ergebnis*100;

                        if (i < fn) {
                            formation = formation + "p" + "<sub><small>"+i+
                                    "</small></sub>" +" = " +
                                    Double.toString(Math.round(
                                            Reflexion_zwischen*10000)/10000.0) +
                                    " %<br>T<sub><small>"+i+"</small></sub> = " +
                                    Double.toString(Math.round(
                                            Transmission_zwischen*10000)/10000.0) +
                                    " %<br>T<sub><small>"+i+"<sub><small>Ges</small>" +
                                    "</sub></small></sub> = " +
                                    Double.toString(Math.round(
                                            Ergebnis_zwischen*10000)/10000.0) +
                                    " %<br><br>";
                            Ausgabe_zwischenergebnisse = (TextView)
                                    findViewById(zwischenspeicher);
                            Ausgabe_zwischenergebnisse.setText(Html.fromHtml(formation));
                        }
                    }
                }
            }


        tv.setVisibility(View.VISIBLE);

            Ausgabe = (TextView) findViewById(0);

            if (fn == 1) {
                Ausgabe.setText(Html.fromHtml("T<sub><small>"+fn+
                        "<sub><small>Ges</small></sub></small></sub> = " +
                        Double.toString(Math.round(Ergebnis_zwischen*10000)/10000.0) +
                        " %<br><br>p" + "<sub><small>"+fn+"</small></sub>" +" = " +
                        Double.toString(Math.round(Reflexion_zwischen*10000)/10000.0) +
                        " %<br>"));
            } else {
                Ausgabe.setText(Html.fromHtml("Flächen: " + fn +
                        "<br>T<sub><small>"+fn+
                        "<sub><small>Ges</small></sub></small></sub> = " +
                        Double.toString(Math.round(Ergebnis_zwischen*10000)/10000.0) +
                        " %<br><br>p" + "<sub><small>"+fn+"</small></sub>" +" = " +
                        Double.toString(Math.round(Reflexion_zwischen*10000)/10000.0) +
                        " %<br>"));
            }
        }

    });
  }
}
```

n) Programmtext – Shape.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.view.View;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.GridLayout;
import android.widget.Spinner;
import android.widget.TextView;


public class Shape extends AppCompatActivity {

    Spinner shape_s, shape_D;
    CheckBox cb_s_munendlich, cb_s_unendlich;
    EditText et_DGes, et_s, et_n;
    TextView txt_shape_s_mm, txt_shape_D_mm, P_result, S_sph_result, S_koma_result,
            r1_sph_result, r2_sph_result, r1_koma_result, r2_koma_result;
    ArrayAdapter<CharSequence> adapter;
    GridLayout grid_result;
    String s_auswahl, D_auswahl;
    double P, f_1, s, S_sph, S_koma, r1_sph, r2_sph, r1_koma, r2_koma, n, Umrechner;


    final Context context = this;
    Button bt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.shape);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        grid_result = (GridLayout) findViewById(R.id.grid_result);

        cb_s_munendlich = (CheckBox) findViewById(R.id.cb_shape_s_munendlich);
        cb_s_unendlich = (CheckBox) findViewById(R.id.cb_shape_s_unendlich);
        et_DGes = (EditText) findViewById(R.id.et_DGes);
        et_DGes.requestFocus();
        et_n = (EditText) findViewById(R.id.et_n);
        et_s = (EditText) findViewById(R.id.et_s);
        txt_shape_s_mm = (TextView) findViewById(R.id.txt_shape_s_mm);
        txt_shape_D_mm = (TextView) findViewById(R.id.txt_D_dpt);

        shape_D = (Spinner) findViewById(R.id.spinner_shape_D);
        adapter = ArrayAdapter.createFromResource(this, R.array.spinner_shape_D,
                android.R.layout.simple_list_item_1);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        shape_D.setAdapter(adapter);
        shape_D.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {


                if (position == 0) {
                    txt_shape_D_mm.setText(" dpt");
                    D_auswahl = "D<sub><small>Ges<sub><small><small>Luft</small>" +
                            "</small></sub></small></sub>";
                    String Alarm = et_DGes.getText().toString();
                    if (Alarm.equals("") || Alarm.equals("0")) {
                    } else {
                        Umrechner = Double.parseDouble(et_DGes.getText().toString());
                        Umrechner = 1 / (Umrechner / 1000);
                        et_DGes.setText(Double.toString(Math.round(Umrechner * 10000) / 10000.0));
                    }
                } else {
                    txt_shape_D_mm.setText(" mm");
                    D_auswahl = "f'";
                    String Alarm = et_DGes.getText().toString();
                    if (Alarm.equals("") || Alarm.equals("0")) {
                    } else {
                        Umrechner = Double.parseDouble(et_DGes.getText().toString());
```

```
                    Umrechner = (1 / Umrechner) * 1000;
                    et_DGes.setText(Double.toString(Math.round(Umrechner * 10000) / 10000.0));
                }
            }

        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });

    shape_s = (Spinner) findViewById(R.id.shape_s);
    adapter = ArrayAdapter.createFromResource(this, R.array.shape_s,
            android.R.layout.simple_list_item_1);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    shape_s.setAdapter(adapter);
    shape_s.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {


            if (position == 0) {
                cb_s_munendlich.setChecked(false);
                cb_s_unendlich.setChecked(false);
                cb_s_munendlich.setVisibility(View.VISIBLE);
                cb_s_unendlich.setVisibility(View.VISIBLE);
                s_auswahl = "s";
            } else {
                cb_s_munendlich.setChecked(false);
                cb_s_unendlich.setChecked(false);
                cb_s_munendlich.setVisibility(View.GONE);
                cb_s_unendlich.setVisibility(View.GONE);
                s_auswahl = "s'";
            }

        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });

    cb_s_munendlich.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

            if (cb_s_munendlich.isChecked()) {
                cb_s_unendlich.setVisibility(View.GONE);
                et_s.setText("");
                et_s.setVisibility(View.GONE);
                txt_shape_s_mm.setVisibility(View.GONE);
            } else {
                cb_s_unendlich.setVisibility(View.VISIBLE);
                et_s.setVisibility(View.VISIBLE);
                txt_shape_s_mm.setVisibility(View.VISIBLE);
            }
        }
    });

    cb_s_unendlich.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

            if (cb_s_unendlich.isChecked()) {
                cb_s_munendlich.setVisibility(View.GONE);
                et_s.setText("");
                et_s.setVisibility(View.GONE);
                txt_shape_s_mm.setVisibility(View.GONE);
            } else {
                cb_s_munendlich.setVisibility(View.VISIBLE);
                et_s.setVisibility(View.VISIBLE);
                txt_shape_s_mm.setVisibility(View.VISIBLE);
            }
        }
    });


    bt = (Button) findViewById(R.id.bT);
    bt.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {

            String Alarm = et_DGes.getText().toString();
            if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                    Alarm.equals("+")) {
                AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
```

```
        AlarmBox.setTitle("Leeres Feld!");
        AlarmBox.setMessage(Html.fromHtml(D_auswahl + " ist leer<br>"));
        AlarmBox.setNeutralButton("OK", null);

        AlertDialog dialog = AlarmBox.create();
        dialog.show();
        return;
    }

    Alarm = et_s.getText().toString();
    if (cb_s_munendlich.isChecked() || cb_s_unendlich.isChecked()) {
    } else {

        if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
                Alarm.equals("+")) {
        AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
        AlarmBox.setTitle("Leeres Feld!");
        AlarmBox.setMessage(s_auswahl + " ist leer");
        AlarmBox.setNeutralButton("OK", null);

        AlertDialog dialog = AlarmBox.create();
        dialog.show();
        return;
        }

    }


    Alarm = et_n.getText().toString();
    if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
            Alarm.equals("+")) {
        AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
        AlarmBox.setTitle("Leeres Feld!");
        AlarmBox.setMessage("n ist leer");
        AlarmBox.setNeutralButton("OK", null);

        AlertDialog dialog = AlarmBox.create();
        dialog.show();
        return;
    }

    grid_result.setVisibility(View.VISIBLE);

    if (cb_s_munendlich.isChecked()) {
        if (D_auswahl.equals("f'")) {
            f_1 = Double.parseDouble(et_DGes.getText().toString());
        } else {
            f_1 = Double.parseDouble(et_DGes.getText().toString());
            f_1 = (1 / f_1) * 1000;
        }
        P = -1;
    } else if (cb_s_unendlich.isChecked()) {
        if (D_auswahl.equals("f'")) {
            f_1 = Double.parseDouble(et_DGes.getText().toString());
        } else {
            f_1 = Double.parseDouble(et_DGes.getText().toString());
            f_1 = (1 / f_1) * 1000;
        }
        P = 1;
    } else {
        if (s_auswahl.equals("s")) {
            if (D_auswahl.equals("f'")) {
                f_1 = Double.parseDouble(et_DGes.getText().toString());
                s = Double.parseDouble(et_s.getText().toString());
                P = -((( 2 * f_1) / s) + 1);
            } else {
                f_1 = Double.parseDouble(et_DGes.getText().toString());
                f_1 = (1 / f_1) * 1000;
                s = Double.parseDouble(et_s.getText().toString());
                P = -(((2 * f_1) / s) + 1);
            }
        } else {
            if (D_auswahl.equals("f'")) {
                f_1 = Double.parseDouble(et_DGes.getText().toString());
                s = Double.parseDouble(et_s.getText().toString());
                P = 1 - ((2 * f_1) / s);
            } else {
                f_1 = Double.parseDouble(et_DGes.getText().toString());
                f_1 = (1 / f_1) * 1000;
                s = Double.parseDouble(et_s.getText().toString());
                P = 1 - ((2 * f_1) / s);
            }
        }
    }

    n = Double.parseDouble(et_n.getText().toString());

    S_sph = -((2 * (Math.pow(n, 2) - 1)) / (n + 2)) * P;
```

```
                r1_sph = (2 * f_1 *(n - 1)) / (S_sph + 1);
                r2_sph = (2 * f_1 *(n - 1)) / (S_sph - 1);

                S_koma = -(((2 * Math.pow(n, 2)) - n - 1) / (n + 1)) * P;
                r1_koma = (2 * f_1 *(n - 1)) / (S_koma + 1);
                r2_koma = (2 * f_1 *(n - 1)) / (S_koma - 1);

                P_result = (TextView) findViewById(R.id.txt_P_result);
                S_sph_result = (TextView) findViewById(R.id.txt_shape_Shape_sph_result);
                S_koma_result = (TextView) findViewById(R.id.txt_shape_Shape_koma_result);

                r1_sph_result = (TextView) findViewById(R.id.txt_shape_r1_sph_result);
                r2_sph_result = (TextView) findViewById(R.id.txt_shape_r2_sph_result);

                r1_koma_result = (TextView) findViewById(R.id.txt_shape_r1_koma_result);
                r2_koma_result = (TextView) findViewById(R.id.txt_shape_r2_koma_result);

                P_result.setText(Double.toString(Math.round(P * 10000) / 10000.0));
                String Ausgabe = P_result.getText().toString();
                if (Ausgabe.length() > 12) {
                   P_result.setText(String.format("%9.4e", P).replace(",","."));
                }

                S_sph_result.setText(Double.toString(Math.round(S_sph * 10000) / 10000.0));
                Ausgabe = S_sph_result.getText().toString();
                if (Ausgabe.length() > 12) {
                  S_sph_result.setText(String.format("%9.4e", S_sph).replace(",","."));
                }

                S_koma_result.setText(Double.toString(Math.round(S_koma * 10000) / 10000.0));
                Ausgabe = S_koma_result.getText().toString();
                if (Ausgabe.length() > 12) {
                    S_koma_result.setText(String.format("%9.4e", S_koma).replace(",","."));
                }

                r1_sph_result.setText(Double.toString(Math.round(r1_sph * 10000) / 10000.0));
                Ausgabe = r1_sph_result.getText().toString();
                if (Ausgabe.length() > 12) {
                    r1_sph_result.setText(String.format("%9.4e", r1_sph).replace(",","."));
                }

                r2_sph_result.setText(Double.toString(Math.round(r2_sph * 10000) / 10000.0));
                Ausgabe = r2_sph_result.getText().toString();
                if (Ausgabe.length() > 12) {
                    r2_sph_result.setText(String.format("%9.4e", r2_sph).replace(",","."));
                }

                r1_koma_result.setText(Double.toString(Math.round(r1_koma * 10000) / 10000.0));
                Ausgabe = r1_koma_result.getText().toString();
                if (Ausgabe.length() > 12) {
                    r1_koma_result.setText(String.format("%9.4e", r1_koma).replace(",","."));
                }

                r2_koma_result.setText(Double.toString(Math.round(r2_koma * 10000) / 10000.0));
                Ausgabe = r2_koma_result.getText().toString();
                if (Ausgabe.length() > 12) {
                    r2_koma_result.setText(String.format("%9.4e", r2_koma).replace(",","."));
                }

                }
        });
    }
}
```

o) Programmtext – Glasberechnung.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.Context;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.GridLayout;
import android.widget.TextView;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.LimitLine;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;

import java.util.ArrayList;


public class Glasberechnung extends AppCompatActivity {


    final Context context = this;


    // Beschreibung Visibility gone
    private TextView txt_HS2_wunsch, txt_glasrechner_hs2_out_einheit, txt_F2_wkn,
            txt_glasrechner_f2_wkn_out_einheit, txt_F3_wkn, txt_glasrechner_f3_wkn_out_einheit,
            txt_F3_real, txt_glasrechner_f3_real_out_einheit, txt_R3,
            txt_glasrechner_r3_out_einheit, txt_DR3, txt_glasrechner_dr3_out_einheit,
            txt_glasrechner_hs2_refra, txt_glasrechner_hs2_asti, txt_glasrechner_hs2_vz;

    // Ausgabe Visibility gone
    private TextView txt_glasrechner_hs2_out, txt_glasrechner_f2_wkn_out,
            txt_glasrechner_f3_wkn_out, txt_glasrechner_f3_real_out, txt_glasrechner_r3_out,
            txt_glasrechner_dr3_out, txt_glasrechner_refra_hs2_out,
            txt_glasrechner_asti_hs2_out, txt_glasrechner_vz_hs2_out;

    // Ausgabe Visibility true
    private TextView txt_glasrechner_hs1_out, txt_glasrechner_f1_real_out,
            txt_glasrechner_f2_real_out, txt_glasrechner_r1_out, txt_glasrechner_r2_out,
            txt_glasrechner_dr2_out, txt_glasrechner_bauhoehe_out, txt_glasrechner_volumen_out,
            txt_glasrechner_gewicht_out, txt_glasrechner_grad_refra_out,
            txt_glasrechner_refra_hs1_out, txt_glasrechner_grad_asti_out,
            txt_glasrechner_asti_hs1_out, txt_glasrechner_grad_vz_out, txt_glasrechner_vz_hs1_out,
            txt_glasrechner_d_einheit, txt_glasrechner_d, txt_glasrechner_dr_einheit,
            txt_glasrechner_dr, txt_F1_real, txt_F2_real;

    private EditText et_glasrechner_sph_in, et_glasrechner_cyl_in, et_glasrechner_f1_in,
            et_glasrechner_n_in, et_glasrechner_d_in, et_glasrechner_dr_in,
            et_glasrechner_durchmesser_in, et_glasrechner_rho_in, et_glasrechner_b_1_in,
            et_glasrechner_s_1_in, et_glasrechner_blickwinkel_in, et_glasrechner_abstufung_in;
    private CheckBox cb_glasrechner_werkzeugindex, cb_glasrechner_flaechen_gerundet;
    private GridLayout grid_glasrechner_result;
    private Button bt;

    private LineChart lc_glasrechner_refra_out, lc_glasrechner_asti_out, lc_glasrechner_vz_out;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.glasberechnung);

        // Verhindert das Öffnen des Tastaturlayouts beim Starten der Activity
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

        et_glasrechner_sph_in = (EditText) findViewById(R.id.et_glasrechner_sph_in);
        et_glasrechner_sph_in.requestFocus();

        cb_glasrechner_werkzeugindex = (CheckBox) findViewById(R.id.cb_glasrechner_werkzeugindex);
        cb_glasrechner_flaechen_gerundet =
                (CheckBox) findViewById(R.id.cb_glasrechner_flaechen_gerundet);
        cb_glasrechner_flaechen_gerundet.setEnabled(false);
```

```
cb_glasrechner_werkzeugindex.setOnCheckedChangeListener
        (new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (cb_glasrechner_werkzeugindex.isChecked()) {
            cb_glasrechner_flaechen_gerundet.setEnabled(true);
        } else {
            cb_glasrechner_flaechen_gerundet.setChecked(false);
            cb_glasrechner_flaechen_gerundet.setEnabled(false);
        }
    }
});

bt = (Button) findViewById(R.id.bT);
bt.setOnClickListener(new View.OnClickListener() {
   public void onClick(View v) {

        txt_HS2_wunsch = (TextView) findViewById(R.id.txt_HS2_wunsch);
        txt_HS2_wunsch.setVisibility(View.GONE);
        txt_glasrechner_hs2_out_einheit = (TextView)
                findViewById(R.id.txt_glasrechner_hs2_out_einheit);
        txt_glasrechner_hs2_out_einheit.setVisibility(View.GONE);
        txt_F1_real = (TextView) findViewById(R.id.txt_F1_real);
        txt_F2_real = (TextView) findViewById(R.id.txt_F2_real);
        txt_F2_wkn = (TextView) findViewById(R.id.txt_F2_wkn);
        txt_F2_wkn.setVisibility(View.GONE);
        txt_glasrechner_f2_wkn_out_einheit = (TextView)
                findViewById(R.id.txt_glasrechner_f2_wkn_out_einheit);
        txt_glasrechner_f2_wkn_out_einheit.setVisibility(View.GONE);
        txt_F3_wkn = (TextView) findViewById(R.id.txt_F3_wkn);
        txt_F3_wkn.setVisibility(View.GONE);
        txt_glasrechner_f3_wkn_out_einheit = (TextView)
                findViewById(R.id.txt_glasrechner_f3_wkn_out_einheit);
        txt_glasrechner_f3_wkn_out_einheit.setVisibility(View.GONE);
        txt_F3_real = (TextView) findViewById(R.id.txt_F3_real);
        txt_F3_real.setVisibility(View.GONE);
        txt_glasrechner_f3_real_out_einheit = (TextView)
                findViewById(R.id.txt_glasrechner_f3_real_out_einheit);
        txt_glasrechner_f3_real_out_einheit.setVisibility(View.GONE);
        txt_R3 = (TextView) findViewById(R.id.txt_R3);
        txt_R3.setVisibility(View.GONE);
        txt_glasrechner_r3_out_einheit = (TextView)
                findViewById(R.id.txt_glasrechner_r3_out_einheit);
        txt_glasrechner_r3_out_einheit.setVisibility(View.GONE);
        txt_DR3 = (TextView) findViewById(R.id.txt_DR3);
        txt_DR3.setVisibility(View.GONE);
        txt_glasrechner_dr3_out_einheit = (TextView)
                findViewById(R.id.txt_glasrechner_dr3_out_einheit);
        txt_glasrechner_dr3_out_einheit.setVisibility(View.GONE);
        txt_glasrechner_hs2_refra = (TextView) findViewById(R.id.txt_glasrechner_hs2_refra);
        txt_glasrechner_hs2_refra.setVisibility(View.GONE);
        txt_glasrechner_hs2_asti = (TextView) findViewById(R.id.txt_glasrechner_hs2_asti);
        txt_glasrechner_hs2_asti.setVisibility(View.GONE);
        txt_glasrechner_hs2_vz = (TextView) findViewById(R.id.txt_glasrechner_hs2_vz);
        txt_glasrechner_hs2_vz.setVisibility(View.GONE);

        txt_glasrechner_hs2_out = (TextView) findViewById(R.id.txt_glasrechner_hs2_out);
        txt_glasrechner_hs2_out.setVisibility(View.GONE);
        txt_glasrechner_f2_wkn_out = (TextView)
                findViewById(R.id.txt_glasrechner_f2_wkn_out);
        txt_glasrechner_f2_wkn_out.setVisibility(View.GONE);
        txt_glasrechner_f3_wkn_out = (TextView)
                findViewById(R.id.txt_glasrechner_f3_wkn_out);
        txt_glasrechner_f3_wkn_out.setVisibility(View.GONE);
        txt_glasrechner_f3_real_out = (TextView)
                findViewById(R.id.txt_glasrechner_f3_real_out);
        txt_glasrechner_f3_real_out.setVisibility(View.GONE);
        txt_glasrechner_r3_out = (TextView)
                findViewById(R.id.txt_glasrechner_r3_out);
        txt_glasrechner_r3_out.setVisibility(View.GONE);
        txt_glasrechner_dr3_out = (TextView) findViewById(R.id.txt_glasrechner_dr3_out);
        txt_glasrechner_dr3_out.setVisibility(View.GONE);
        txt_glasrechner_refra_hs2_out = (TextView)
                findViewById(R.id.txt_glasrechner_refra_hs2_out);
        txt_glasrechner_refra_hs2_out.setVisibility(View.GONE);
        txt_glasrechner_asti_hs2_out = (TextView)
                findViewById(R.id.txt_glasrechner_asti_hs2_out);
        txt_glasrechner_asti_hs2_out.setVisibility(View.GONE);
        txt_glasrechner_vz_hs2_out = (TextView)
                findViewById(R.id.txt_glasrechner_vz_hs2_out);
        txt_glasrechner_vz_hs2_out.setVisibility(View.GONE);


        txt_glasrechner_hs1_out = (TextView) findViewById(R.id.txt_glasrechner_hs1_out);
        txt_glasrechner_f1_real_out = (TextView)
                findViewById(R.id.txt_glasrechner_f1_real_out);
        txt_glasrechner_f2_real_out = (TextView)
                findViewById(R.id.txt_glasrechner_f2_real_out);
```

```
txt_glasrechner_r1_out = (TextView) findViewById(R.id.txt_glasrechner_r1_out);
txt_glasrechner_r2_out = (TextView) findViewById(R.id.txt_glasrechner_r2_out);
txt_glasrechner_dr2_out = (TextView) findViewById(R.id.txt_glasrechner_dr2_out);
txt_glasrechner_bauhoehe_out = (TextView)
        findViewById(R.id.txt_glasrechner_bauhoehe_out);
txt_glasrechner_volumen_out = (TextView)
        findViewById(R.id.txt_glasrechner_volumen_out);
txt_glasrechner_gewicht_out = (TextView)
        findViewById(R.id.txt_glasrechner_gewicht_out);
txt_glasrechner_grad_refra_out = (TextView)
        findViewById(R.id.txt_glasrechner_grad_refra_out);
txt_glasrechner_refra_hs1_out = (TextView)
        findViewById(R.id.txt_glasrechner_refra_hs1_out);
txt_glasrechner_grad_asti_out = (TextView)
        findViewById(R.id.txt_glasrechner_grad_asti_out);
txt_glasrechner_asti_hs1_out = (TextView)
        findViewById(R.id.txt_glasrechner_asti_hs1_out);
txt_glasrechner_grad_vz_out = (TextView)
        findViewById(R.id.txt_glasrechner_grad_vz_out);
txt_glasrechner_vz_hs1_out = (TextView)
        findViewById(R.id.txt_glasrechner_vz_hs1_out);

lc_glasrechner_refra_out = (LineChart) findViewById(R.id.lc_glasrechner_refra_out);
lc_glasrechner_asti_out = (LineChart) findViewById(R.id.lc_glasrechner_asti_out);
lc_glasrechner_vz_out = (LineChart) findViewById(R.id.lc_glasrechner_vz_out);


et_glasrechner_cyl_in = (EditText) findViewById(R.id.et_glasrechner_cyl_in);
et_glasrechner_f1_in = (EditText) findViewById(R.id.et_glasrechner_f1_in);
et_glasrechner_d_in = (EditText) findViewById(R.id.et_glasrechner_d_in);
et_glasrechner_dr_in = (EditText) findViewById(R.id.et_glasrechner_dr_in);
et_glasrechner_n_in = (EditText) findViewById(R.id.et_glasrechner_n_in);
et_glasrechner_durchmesser_in = (EditText)
        findViewById(R.id.et_glasrechner_durchmesser_in);
et_glasrechner_rho_in = (EditText) findViewById(R.id.et_glasrechner_rho_in);
et_glasrechner_b_1_in = (EditText) findViewById(R.id.et_glasrechner_b_1_in);
et_glasrechner_s_1_in = (EditText) findViewById(R.id.et_glasrechner_s_1_in);
et_glasrechner_blickwinkel_in =(EditText)
        findViewById(R.id.et_glasrechner_blickwinkel_in);
et_glasrechner_abstufung_in = (EditText)
        findViewById(R.id.et_glasrechner_abstufung_in);

grid_glasrechner_result = (GridLayout) findViewById(R.id.grid_glasrechner_result);

String Alarm = et_glasrechner_sph_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+")) {
    et_glasrechner_sph_in.setText("0");
}

double sph_in = Double.parseDouble(et_glasrechner_sph_in.getText().toString());

Alarm = et_glasrechner_cyl_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+")) {
    et_glasrechner_cyl_in.setText("0");
}

double cyl_in = Double.parseDouble(et_glasrechner_cyl_in.getText().toString());

Alarm = et_glasrechner_f1_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+")) {
    double fix;
    fix = 4;
    if (sph_in > 0 || sph_in+cyl_in > 0){
        if (cyl_in > 0) {
            fix = sph_in + cyl_in + 1;
        } else {
            fix = sph_in + 1;
        }

    }
    et_glasrechner_f1_in.setText(Double.toString(Math.round((fix)*100)/100.0));
}

double f1_in = Double.parseDouble(et_glasrechner_f1_in.getText().toString());

Alarm = et_glasrechner_n_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+")) {
    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
    AlarmBox.setTitle("Leeres Feld!");
    AlarmBox.setMessage("n ist leer");
    AlarmBox.setNeutralButton("OK", null);

    AlertDialog dialog = AlarmBox.create();
    dialog.show();
    return;
}

double n_in = Double.parseDouble(et_glasrechner_n_in.getText().toString());
```

```
Alarm = et_glasrechner_d_in.getText().toString();
if((Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+")
        || Alarm.equals("0")) && ((sph_in+cyl_in < 0) && (sph_in < 0))) {
    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
    AlarmBox.setTitle("Leeres Feld!");
    AlarmBox.setMessage("Wenn beide HS negativ sind muss die Mittenddicke d " +
            "angegeben werden");
    AlarmBox.setNeutralButton("OK", null);

    AlertDialog dialog = AlarmBox.create();
    dialog.show();
    return;
} else if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
        Alarm.equals("+")) {
    et_glasrechner_d_in.setText("1.2");
}

double d_in = Double.parseDouble(et_glasrechner_d_in.getText().toString());

Alarm = et_glasrechner_dr_in.getText().toString();
if((Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+"))
        && ((sph_in+cyl_in >= 0) || (sph_in >= 0))) {
    AlertDialog.Builder AlarmBox = new AlertDialog.Builder(context);
    AlarmBox.setTitle("Leeres Feld!");
    AlarmBox.setMessage(Html.fromHtml("Wenn min. ein HS positiv ist muss die min. " +
            "Randdicke d<sub><small>r</small></sub> angegeben werden <br>"));
    AlarmBox.setNeutralButton("OK", null);

    AlertDialog dialog = AlarmBox.create();
    dialog.show();
    return;
} else if (Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") ||
        Alarm.equals("+")) {
    et_glasrechner_dr_in.setText("0.7");
}

double dr_in = Double.parseDouble(et_glasrechner_dr_in.getText().toString());

Alarm = et_glasrechner_durchmesser_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+")
        || Alarm.equals("0")) {
    et_glasrechner_durchmesser_in.setText("60");
}

double durchmesser_in =
        Double.parseDouble(et_glasrechner_durchmesser_in.getText().toString());

Alarm = et_glasrechner_rho_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+")) {
    et_glasrechner_rho_in.setText("1.2");
}

double rho_in = Double.parseDouble(et_glasrechner_rho_in.getText().toString());

Alarm = et_glasrechner_b_1_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+") ||
        Alarm.equals("0")) {
    et_glasrechner_b_1_in.setText("27.5");
}

double b_1_in = Double.parseDouble(et_glasrechner_b_1_in.getText().toString());

Alarm = et_glasrechner_s_1_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+") ||
        Alarm.equals("0")) {
    et_glasrechner_s_1_in.setText("-1000");
}

double s_1_in = Double.parseDouble(et_glasrechner_s_1_in.getText().toString());

Alarm = et_glasrechner_blickwinkel_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+") ||
        Alarm.equals("0")) {
    et_glasrechner_blickwinkel_in.setText("1");
}

int blickwinkel_in =
        Integer.parseInt(et_glasrechner_blickwinkel_in.getText().toString());

Alarm = et_glasrechner_abstufung_in.getText().toString();
if(Alarm.equals("") || Alarm.equals(".") || Alarm.equals("-") || Alarm.equals("+") ||
        Alarm.equals("0")) {
    et_glasrechner_abstufung_in.setText("1");
}

int abstufung_in =
        Integer.parseInt(et_glasrechner_abstufung_in.getText().toString());
```

```
            for (int i=0; abstufung_in > blickwinkel_in; i++) {
                abstufung_in = abstufung_in - 1;
            }
            et_glasrechner_abstufung_in.setText(Integer.toString(abstufung_in));

            grid_glasrechner_result.setVisibility(View.VISIBLE);


            double f1_real, f2_real, f2_wkn, f3_real, f3_wkn;
            double hs1, hs2;
            double r1, r2, r3;

            double dr, ddr;
            double t1, t2, t3, dr1, dr2;
            r1 = 100;
            r2 = 100;
            r3 = 100;

        if (sph_in >= 0 || sph_in+cyl_in >= 0) {

            for (int i = 0; i < 4; i++) {

                if (cb_glasrechner_werkzeugindex.isChecked()) {
                    if (cb_glasrechner_flaechen_gerundet.isChecked()) {
                        f1_real = ((n_in - 1) / (1.525 - 1)) * f1_in;
                        f2_real = (sph_in - (f1_real / (1 - ((d_in * 0.001) / n_in) *
                                f1_real)));
                        f2_wkn = Math.round((((1.525 - 1) / (n_in - 1)) * f2_real) / 0.0625) *
                                0.0625;
                        f2_real = ((n_in - 1) / (1.525 - 1)) * f2_wkn;

                        txt_glasrechner_f1_real_out.setText(
                                Double.toString(Math.round(f1_real * 10000) / 10000.0));
                        txt_F1_real.setText(Html.fromHtml("F<sub><small>1<sub><small>"+
                                Double.toString(Math.round(n_in*1000)/1000.0)+
                                "</small></sub></small></sub>:"));
                        txt_glasrechner_f2_real_out.setText(
                                Double.toString(Math.round(f2_real * 10000) / 10000.0));
                        txt_F2_real.setText(Html.fromHtml("F<sub><small>2<sub><small>"+
                                Double.toString(Math.round(n_in*1000)/1000.0)+
                                "</small></sub></small></sub>:"));
                        txt_glasrechner_f2_wkn_out.setText(Double.toString(f2_wkn));

                        hs1 = f2_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                        txt_glasrechner_hs1_out.setText(
                                Double.toString(Math.round(hs1 * 10000) / 10000.0));

                        r1 = 1000 * (n_in - 1) / f1_real;
                        r2 = 1000 * (1 - n_in) / f2_real;

                        txt_glasrechner_r1_out.setText(
                                Double.toString(Math.round(r1 * 10000) / 10000.0));
                        txt_glasrechner_r2_out.setText(
                                Double.toString(Math.round(r2 * 10000) / 10000.0));

                        if (cyl_in != 0) {
                            f3_real = ((sph_in + cyl_in) -
                                    (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real)));
                            f3_wkn = Math.round((((1.525 - 1) / (n_in - 1)) * f3_real) / 0.0625)
                                    * 0.0625;
                            f3_real = ((n_in - 1) / (1.525 - 1)) * f3_wkn;

                            txt_glasrechner_f3_real_out.setText(
                                    Double.toString(Math.round(f3_real * 10000) / 10000.0));
                            txt_F3_real.setText(Html.fromHtml("F<sub><small>3<sub><small>"+
                                    Double.toString(Math.round(n_in*1000)/1000.0)+
                                    "</small></sub></small></sub>:"));
                            txt_glasrechner_f3_wkn_out.setText(Double.toString(f3_wkn));

                            hs2 = f3_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                            txt_glasrechner_hs2_out.setText(Double.toString(
                                    Math.round(hs2 * 10000) / 10000.0));

                            r3 = 1000 * (1 - n_in) / f3_real;

                            txt_glasrechner_r3_out.setText(Double.toString(
                                    Math.round(r3 * 10000) / 10000.0));
                        }
                    } else {
                        f1_real = ((n_in - 1) / (1.525 - 1)) * f1_in;
                        f2_real = (sph_in - (f1_real / (1 - ((d_in * 0.001) / n_in) *
                                f1_real)));
                        f2_wkn = (((1.525 - 1) / (n_in - 1)) * f2_real);


                        txt_glasrechner_f1_real_out.setText(
```

```
                Double.toString(Math.round(f1_real * 10000) / 10000.0));
            txt_F1_real.setText(Html.fromHtml("F<sub><small>1<sub><small>"+
                Double.toString(Math.round(n_in*1000)/1000.0)+
                "</small></sub></small></sub>:"));
            txt_glasrechner_f2_real_out.setText(
                Double.toString(Math.round(f2_real * 10000) / 10000.0));
            txt_F2_real.setText(Html.fromHtml("F<sub><small>2<sub><small>"+
                Double.toString(Math.round(n_in*1000)/1000.0)+
                "</small></sub></small></sub>:"));
            txt_glasrechner_f2_wkn_out.setText(
                Double.toString(Math.round(f2_wkn * 10000) / 10000.0));

            hs1 = f2_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

            txt_glasrechner_hs1_out.setText(
                Double.toString(Math.round(hs1 * 10000) / 10000.0));

            r1 = 1000 * (n_in - 1) / f1_real;
            r2 = 1000 * (1 - n_in) / f2_real;

            txt_glasrechner_r1_out.setText(
                Double.toString(Math.round(r1 * 10000) / 10000.0));
            txt_glasrechner_r2_out.setText(
                Double.toString(Math.round(r2 * 10000) / 10000.0));

            if (cyl_in != 0) {
                f3_real = ((sph_in + cyl_in) - (f1_real / (1 - ((d_in * 0.001) /
                    n_in) * f1_real)));
                f3_wkn = (((1.525 - 1) / (n_in - 1)) * f3_real);


                txt_glasrechner_f3_real_out.setText(
                    Double.toString(Math.round(f3_real * 10000) / 10000.0));
                txt_F3_real.setText(Html.fromHtml("F<sub><small>3<sub><small>"+
                    Double.toString(Math.round(n_in*1000)/1000.0)+
                    "</small></sub></small></sub>:"));
                txt_glasrechner_f3_wkn_out.setText(
                    Double.toString(Math.round(f3_wkn * 10000) / 10000.0));

                hs2 = f3_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                txt_glasrechner_hs2_out.setText(
                    Double.toString(Math.round(hs2 * 10000) / 10000.0));

                r3 = 1000 * (1 - n_in) / f3_real;

                txt_glasrechner_r3_out.setText(
                    Double.toString(Math.round(r3 * 10000) / 10000.0));
            }
        }
    } else {
        f1_real = ((n_in - 1) / (1.525 - 1)) * f1_in;
        f2_real = (sph_in - (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real)));

        txt_glasrechner_f1_real_out.setText(
            Double.toString(Math.round(f1_real * 10000) / 10000.0));
        txt_F1_real.setText(Html.fromHtml("F<sub><small>1<sub><small>"+
            Double.toString(Math.round(n_in*1000)/1000.0)+
            "</small></sub></small></sub>:"));
        txt_glasrechner_f2_real_out.setText(
            Double.toString(Math.round(f2_real * 10000) / 10000.0));
        txt_F2_real.setText(Html.fromHtml("F<sub><small>2<sub><small>"+
            Double.toString(Math.round(n_in*1000)/1000.0)+
            "</small></sub></small></sub>:"));

        hs1 = f2_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

        txt_glasrechner_hs1_out.setText(
            Double.toString(Math.round(hs1 * 10000) / 10000.0));

        r1 = 1000 * (n_in - 1) / f1_real;
        r2 = 1000 * (1 - n_in) / f2_real;

        txt_glasrechner_r1_out.setText(
            Double.toString(Math.round(r1 * 10000) / 10000.0));
        txt_glasrechner_r2_out.setText(
            Double.toString(Math.round(r2 * 10000) / 10000.0));

        if (cyl_in != 0) {
            f3_real = ((sph_in + cyl_in) - (f1_real / (1 - ((d_in * 0.001) / n_in) *
                f1_real)));

            txt_glasrechner_f3_real_out.setText(
                Double.toString(Math.round(f3_real * 10000) / 10000.0));
            txt_F3_real.setText(Html.fromHtml("F<sub><small>3<sub><small>"+
                Double.toString(Math.round(n_in*1000)/1000.0)+
                "</small></sub></small></sub>:"));
```

```
                    hs2 = f3_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                    txt_glasrechner_hs2_out.setText(
                            Double.toString(Math.round(hs2 * 10000) / 10000.0));

                    r3 = 1000 * (1 - n_in) / f3_real;

                    txt_glasrechner_r3_out.setText(
                            Double.toString(Math.round(r3 * 10000) / 10000.0));
                }
            }

            t1 = r1 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
                    (Math.pow(r1, 2)))));
            t2 = r2 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
                    (Math.pow(r2, 2)))));

            if (cyl_in != 0) {

                t3 = r3 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
                        (Math.pow(r3, 2))))));

                if (r2 >= r3) {

                    dr = d_in - t1 + t2;
                    ddr = dr - dr_in;
                    d_in = d_in - ddr;
                    d_in = Math.round(d_in*10000)/10000.0;
                } else {
                    dr = d_in - t1 + t3;
                    ddr = dr - dr_in;
                    d_in = d_in - ddr;
                    d_in = Math.round(d_in*10000)/10000.0;
                }
            } else {
                dr = d_in - t1 + t2;
                ddr = dr - dr_in;
                d_in = d_in - ddr;
                d_in = Math.round(d_in*10000)/10000.0;
            }

            et_glasrechner_d_in.setText(Double.toString(d_in));

        }
    } else {
        if (cb_glasrechner_werkzeugindex.isChecked()) {
            if (cb_glasrechner_flaechen_gerundet.isChecked()) {
                f1_real = ((n_in - 1) / (1.525 - 1)) * f1_in;
                f2_real = (sph_in - (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real)));
                f2_wkn = Math.round((((1.525 - 1) / (n_in - 1)) * f2_real) / 0.0625) *
                        0.0625;
                f2_real = ((n_in - 1) / (1.525 - 1)) * f2_wkn;

                txt_glasrechner_f1_real_out.setText(
                        Double.toString(Math.round(f1_real * 10000) / 10000.0));
                txt_F1_real.setText(Html.fromHtml("F<sub><small>1<sub><small>"+
                        Double.toString(Math.round(n_in*1000)/1000.0)+
                        "</small></sub></small></sub>:"));
                txt_glasrechner_f2_real_out.setText(
                        Double.toString(Math.round(f2_real * 10000) / 10000.0));
                txt_F2_real.setText(Html.fromHtml("F<sub><small>2<sub><small>"+
                        Double.toString(Math.round(n_in*1000)/1000.0)+
                        "</small></sub></small></sub>:"));
                txt_glasrechner_f2_wkn_out.setText(Double.toString(f2_wkn));

                hs1 = f2_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                txt_glasrechner_hs1_out.setText(
                        Double.toString(Math.round(hs1 * 10000) / 10000.0));

                r1 = 1000 * (n_in - 1) / f1_real;
                r2 = 1000 * (1 - n_in) / f2_real;

                txt_glasrechner_r1_out.setText(
                        Double.toString(Math.round(r1 * 10000) / 10000.0));
                txt_glasrechner_r2_out.setText(
                        Double.toString(Math.round(r2 * 10000) / 10000.0));

                if (cyl_in != 0) {
                    f3_real = ((sph_in + cyl_in) - (f1_real / (1 - ((d_in * 0.001) /
                            n_in) * f1_real)));
                    f3_wkn = Math.round((((1.525 - 1) / (n_in - 1)) * f3_real) / 0.0625) *
                            0.0625;
                    f3_real = ((n_in - 1) / (1.525 - 1)) * f3_wkn;

                    txt_glasrechner_f3_real_out.setText(
                            Double.toString(Math.round(f3_real * 10000) / 10000.0));
                    txt_F3_real.setText(Html.fromHtml("F<sub><small>3<sub><small>"+
```

```
                        Double.toString(Math.round(n_in*1000)/1000.0)+
                        "</small></sub></small></sub>:"));
                txt_glasrechner_f3_wkn_out.setText(Double.toString(f3_wkn));

                hs2 = f3_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                txt_glasrechner_hs2_out.setText(
                        Double.toString(Math.round(hs2 * 10000) / 10000.0));

                r3 = 1000 * (1 - n_in) / f3_real;

                txt_glasrechner_r3_out.setText(
                        Double.toString(Math.round(r3 * 10000) / 10000.0));
            }
        } else {
            f1_real = ((n_in - 1) / (1.525 - 1)) * f1_in;
            f2_real = (sph_in - (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real)));
            f2_wkn = (((1.525 - 1) / (n_in - 1)) * f2_real);


            txt_glasrechner_f1_real_out.setText(
                    Double.toString(Math.round(f1_real * 10000) / 10000.0));
            txt_F1_real.setText(Html.fromHtml("F<sub><small>1<sub><small>"+
                    Double.toString(Math.round(n_in*1000)/1000.0)+
                    "</small></sub></small></sub>:"));
            txt_glasrechner_f2_real_out.setText(
                    Double.toString(Math.round(f2_real * 10000) / 10000.0));
            txt_F2_real.setText(Html.fromHtml("F<sub><small>2<sub><small>"+
                    Double.toString(Math.round(n_in*1000)/1000.0)+
                    "</small></sub></small></sub>:"));
            txt_glasrechner_f2_wkn_out.setText(
                    Double.toString(Math.round(f2_wkn * 10000) / 10000.0));

            hs1 = f2_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

            txt_glasrechner_hs1_out.setText(
                    Double.toString(Math.round(hs1 * 10000) / 10000.0));

            r1 = 1000 * (n_in - 1) / f1_real;
            r2 = 1000 * (1 - n_in) / f2_real;

            txt_glasrechner_r1_out.setText(
                    Double.toString(Math.round(r1 * 10000) / 10000.0));
            txt_glasrechner_r2_out.setText(
                    Double.toString(Math.round(r2 * 10000) / 10000.0));

            if (cyl_in != 0) {
                f3_real = ((sph_in + cyl_in) - (f1_real / (1 - ((d_in * 0.001) /
                        n_in) * f1_real)));
                f3_wkn = (((1.525 - 1) / (n_in - 1)) * f3_real);


                txt_glasrechner_f3_real_out.setText(
                        Double.toString(Math.round(f3_real * 10000) / 10000.0));
                txt_F3_real.setText(Html.fromHtml("F<sub><small>3<sub><small>"+
                        Double.toString(Math.round(n_in*1000)/1000.0)+
                        "</small></sub></small></sub>:"));
                txt_glasrechner_f3_wkn_out.setText(
                        Double.toString(Math.round(f3_wkn * 10000) / 10000.0));

                hs2 = f3_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                txt_glasrechner_hs2_out.setText(
                        Double.toString(Math.round(hs2 * 10000) / 10000.0));

                r3 = 1000 * (1 - n_in) / f3_real;

                txt_glasrechner_r3_out.setText(
                        Double.toString(Math.round(r3 * 10000) / 10000.0));
            }
        }
    } else {
        f1_real = ((n_in - 1) / (1.525 - 1)) * f1_in;
        f2_real = (sph_in - (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real)));

        txt_glasrechner_f1_real_out.setText(
                Double.toString(Math.round(f1_real * 10000) / 10000.0));
        txt_F1_real.setText(Html.fromHtml("F<sub><small>1<sub><small>"+
                Double.toString(Math.round(n_in*1000)/1000.0)+
                "</small></sub></small></sub>:"));
        txt_glasrechner_f2_real_out.setText(
                Double.toString(Math.round(f2_real * 10000) / 10000.0));
        txt_F2_real.setText(Html.fromHtml("F<sub><small>2<sub><small>"+
                Double.toString(Math.round(n_in*1000)/1000.0)+
                "</small></sub></small></sub>:"));

        hs1 = f2_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));
```

```
                txt_glasrechner_hs1_out.setText(
                        Double.toString(Math.round(hs1 * 10000) / 10000.0));

                r1 = 1000 * (n_in - 1) / f1_real;
                r2 = 1000 * (1 - n_in) / f2_real;

                txt_glasrechner_r1_out.setText(
                        Double.toString(Math.round(r1 * 10000) / 10000.0));
                txt_glasrechner_r2_out.setText(
                        Double.toString(Math.round(r2 * 10000) / 10000.0));

                if (cyl_in != 0) {
                    f3_real = ((sph_in + cyl_in) - (f1_real / (1 - ((d_in * 0.001) / n_in) *
                            f1_real)));

                    txt_glasrechner_f3_real_out.setText(
                            Double.toString(Math.round(f3_real * 10000) / 10000.0));
                    txt_F3_real.setText(Html.fromHtml("F<sub><small>3<sub><small>"+
                            Double.toString(Math.round(n_in*1000)/1000.0)+
                            "</small></sub></small></sub>:"));

                    hs2 = f3_real + (f1_real / (1 - ((d_in * 0.001) / n_in) * f1_real));

                    txt_glasrechner_hs2_out.setText(
                            Double.toString(Math.round(hs2 * 10000) / 10000.0));

                    r3 = 1000 * (1 - n_in) / f3_real;

                    txt_glasrechner_r3_out.setText(
                            Double.toString(Math.round(r3 * 10000) / 10000.0));
                }
            }
        }

        ArrayList<Entry> yAXES_refra_hs1 = new ArrayList<>();
        ArrayList<Entry> yAXES_refra_hs2 = new ArrayList<>();
        ArrayList<Entry> yAXES_asti_hs1 = new ArrayList<>();
        ArrayList<Entry> yAXES_asti_hs2 = new ArrayList<>();
        ArrayList<Entry> yAXES_vz_hs1 = new ArrayList<>();
        ArrayList<Entry> yAXES_vz_hs2 = new ArrayList<>();

        double s_2, u_2_rad, Sin_e_2, Sin_e2, e_2, e2, u2, s2, alpha2,
                s_1, u_1, Sin_e_1, e_1, Sin_e1, e1, alpha1, u1;
        double l_t1, Ast_Konst_1, Hilfe1, l_t_1, L_t_1, l_s1, Hilfe2, l_s_1;
        double p1, p2, ds;
        double l_t2, Ast_Konst_2, Hilfe21, l_t_2, l_s2, Hilfe22, l_s_2;
        double delta_s_1, t_sk_1, s_sk_1, T_sk_1, S_sk_1, mitt_Brechkraft, Astigmatismus,
                Refraktionsfehler, Verzeichnung;

        double u1_absolut;

        int Grad;

        String Grad_Anzeige = "";
        String hs1_refra_list, hs1_asti_list, hs1_vz_list;
        String hs2_refra_list, hs2_asti_list, hs2_vz_list;

        String Anhang;

        hs1_asti_list = "";
        hs1_refra_list = "";
        hs1_vz_list = "";

        hs2_asti_list = "";
        hs2_refra_list = "";
        hs2_vz_list = "";

        u1_absolut = 0.001;
        double u1_absolut2 = 0.001;

        Grad = 0;
        u_2_rad = Math.toRadians(0.001);

        for (int i=0; Grad <= blickwinkel_in; i++) {
            Grad_Anzeige = Grad_Anzeige + Integer.toString(Grad) + " ° <br>";
            s_2 = b_1_in;
            Sin_e_2 = (s_2-r2)/r2*Math.sin(u_2_rad);
            Sin_e2 = (1/n_in)*Sin_e_2;
            e_2 = Math.asin(Sin_e_2);
            e2 = Math.asin(Sin_e2);
            u2 = u_2_rad+e_2-e2;
            s2 = r2*(1+(Sin_e2/Math.sin(u2)));
            alpha2 = u_2_rad+e_2;

            s_1 = d_in+s2;
            u_1 = u2;
            Sin_e_1 = (s_1-r1)/r1*Math.sin(u_1);
            e_1 = Math.asin(Sin_e_1);
```

```
                    Sin_e1 = (n_in/1)*Sin_e_1;
                    e1 = Math.asin(Sin_e1);
                    alpha1 = u_1+e_1;
                    u1 = u_1+e_1-e1;
                    if (Grad == 0) {
                        u1_absolut = u1;
                    }

                    l_t1 = s_1_in*1000;
                    Ast_Konst_1 = (n_in*Math.cos(e_1)-1*Math.cos(e1))/r1;
                    Hilfe1 = 1*Math.cos(e1)*Math.cos(e1)/l_t1;
                    l_t_1 = n_in*Math.cos(e_1)*Math.cos(e_1)/(Ast_Konst_1+Hilfe1);

                    l_s1 = l_t1;
                    Hilfe2 = 1/l_s1;
                    l_s_1 = n_in/(Ast_Konst_1+Hilfe2);

                    p1 = r1*(1-Math.cos(alpha1));
                    p2 = r2*(1-Math.cos(alpha2));
                    ds = (d_in+p2-p1)/Math.cos(u2);

                    l_t2 = l_t_1-ds;
                    Ast_Konst_2 = (1*Math.cos(e_2)-n_in*Math.cos(e2))/r2;
                    Hilfe21 = n_in*Math.cos(e2)*Math.cos(e2)/l_t2;
                    l_t_2 = 1*Math.cos(e_2)*Math.cos(e_2)/(Ast_Konst_2+Hilfe21);

                    l_s2 = l_s_1-ds;
                    Hilfe22 = n_in/l_s2;
                    l_s_2 = 1/(Ast_Konst_2+Hilfe22);

                    delta_s_1 = (r2*(Math.sin(alpha2)/Math.sin(u_2_rad)))-b_1_in;

                    t_sk_1 = l_t_2-delta_s_1;
                    s_sk_1 = l_s_2-delta_s_1;
                    T_sk_1 = 1000/t_sk_1;
                    S_sk_1 = 1000/s_sk_1;
                    mitt_Brechkraft = 0.5*(T_sk_1+S_sk_1);
                    Astigmatismus = T_sk_1-S_sk_1;
                    Anhang = Double.toString(Math.round(Astigmatismus*10000)/10000.0);
                    if (Anhang.length()>10) {
                        Anhang = String.format("%6.4e", Astigmatismus).replace(",",".");
                    }
                    hs1_asti_list = hs1_asti_list + Anhang + " dpt<br>";

                    float y_data_asti_hs1 = Float.parseFloat(String.valueOf(Astigmatismus));
                    yAXES_asti_hs1.add(new Entry(Grad,y_data_asti_hs1));

                    Refraktionsfehler = mitt_Brechkraft-sph_in-1000/l_t1;
                    Anhang = Double.toString(Math.round(Refraktionsfehler*10000)/10000.0);
                    if (Anhang.length()>10) {
                        Anhang = String.format("%6.4e", Refraktionsfehler).replace(",",".");
                    }
                    hs1_refra_list = hs1_refra_list + Anhang + " dpt<br>";

                    float y_data_refra_hs1 = Float.parseFloat(String.valueOf(Refraktionsfehler));
                    yAXES_refra_hs1.add(new Entry(Grad,y_data_refra_hs1));

                    Verzeichnung                                                                    =
(Math.tan(u_2_rad)/Math.tan(u1))/(Math.tan(Math.toRadians(0.001))/
                            Math.tan(u1_absolut))*100-100;
                    Anhang = Double.toString(Math.round(Verzeichnung*100)/100.0);
                    if (Anhang.length()>10) {
                        Anhang = String.format("%6.4e", Verzeichnung).replace(",",".");
                    }
                    hs1_vz_list = hs1_vz_list + Anhang + " %<br>";

                    float y_data_vz_hs1 = Float.parseFloat(String.valueOf(Verzeichnung));
                    yAXES_vz_hs1.add(new Entry(Grad,y_data_vz_hs1));

                    if (cyl_in != 0) {
                        s_2 = b_1_in;
                        Sin_e_2 = (s_2-r3)/r3*Math.sin(u_2_rad);
                        Sin_e2 = (1/n_in)*Sin_e_2;
                        e_2 = Math.asin(Sin_e_2);
                        e2 = Math.asin(Sin_e2);
                        u2 = u_2_rad+e_2-e2;
                        s2 = r3*(1+(Sin_e2/Math.sin(u2)));
                        alpha2 = u_2_rad+e_2;

                        s_1 = d_in+s2;
                        u_1 = u2;
                        Sin_e_1 = (s_1-r1)/r1*Math.sin(u_1);
                        e_1 = Math.asin(Sin_e_1);
                        Sin_e1 = (n_in/1)*Sin_e_1;
                        e1 = Math.asin(Sin_e1);
                        alpha1 = u_1+e_1;
                        u1 = u_1+e_1-e1;
                        if (Grad == 0) {
```

```
                    u1_absolut2 = u1;
                }

                l_t1 = s_1_in*1000;
                Ast_Konst_1 = (n_in*Math.cos(e_1)-1*Math.cos(e1))/r1;
                Hilfe1 = 1*Math.cos(e1)*Math.cos(e1)/l_t1;
                l_t_1 = n_in*Math.cos(e_1)*Math.cos(e_1)/(Ast_Konst_1+Hilfe1);

                l_s1 = l_t1;
                Hilfe2 = 1/l_s1;
                l_s_1 = n_in/(Ast_Konst_1+Hilfe2);

                p1 = r1*(1-Math.cos(alpha1));
                p2 = r3*(1-Math.cos(alpha2));
                ds = (d_in+p2-p1)/Math.cos(u2);

                l_t2 = l_t_1-ds;
                Ast_Konst_2 = (1*Math.cos(e_2)-n_in*Math.cos(e2))/r3;
                Hilfe21 = n_in*Math.cos(e2)*Math.cos(e2)/l_t2;
                l_t_2 = 1*Math.cos(e_2)*Math.cos(e_2)/(Ast_Konst_2+Hilfe21);

                l_s2 = l_s_1-ds;
                Hilfe22 = n_in/l_s2;
                l_s_2 = 1/(Ast_Konst_2+Hilfe22);

                delta_s_1 = (r3*(Math.sin(alpha2)/Math.sin(u_2_rad)))-b_1_in;

                t_sk_1 = l_t_2-delta_s_1;
                s_sk_1 = l_s_2-delta_s_1;
                T_sk_1 = 1000/t_sk_1;
                S_sk_1 = 1000/s_sk_1;
                mitt_Brechkraft = 0.5*(T_sk_1+S_sk_1);
                Astigmatismus = T_sk_1-S_sk_1;
                Anhang = Double.toString(Math.round(Astigmatismus*10000)/10000.0);
                if (Anhang.length()>10) {
                    Anhang = String.format("%6.4e", Astigmatismus).replace(",",".");
                }
                hs2_asti_list = hs2_asti_list + Anhang + " dpt<br>";

                float y_data_asti_hs2 = Float.parseFloat(String.valueOf(Astigmatismus));
                yAXES_asti_hs2.add(new Entry(Grad,y_data_asti_hs2));

                Refraktionsfehler = mitt_Brechkraft-(sph_in+cyl_in)-1000/l_t1;
                Anhang = Double.toString(Math.round(Refraktionsfehler*10000)/10000.0);
                if (Anhang.length()>10) {
                    Anhang = String.format("%6.4e", Refraktionsfehler).replace(",",".");
                }
                hs2_refra_list = hs2_refra_list + Anhang + " dpt<br>";

                float y_data_refa_hs2 = Float.parseFloat(String.valueOf(Refraktionsfehler));
                yAXES_refra_hs2.add(new Entry(Grad,y_data_refa_hs2));

                Verzeichnung = (Math.tan(u_2_rad)/Math.tan(u1))/
                        (Math.tan(Math.toRadians(0.001))/Math.tan(u1_absolut2))*100-100;
                Anhang = Double.toString(Math.round(Verzeichnung*100)/100.0);
                if (Anhang.length()>10) {
                    Anhang = String.format("%6.4e", Verzeichnung).replace(",",".");
                }
                hs2_vz_list = hs2_vz_list + Anhang + " %<br>";

                float y_data_vz_hs2 = Float.parseFloat(String.valueOf(Verzeichnung));
                yAXES_vz_hs2.add(new Entry(Grad,y_data_vz_hs2));

            }

            Grad = Grad + abstufung_in;
            u_2_rad = Math.toRadians(Grad);
        }

            ArrayList<ILineDataSet> lineDataSets_refra = new ArrayList<>();
            ArrayList<ILineDataSet> lineDataSets_asti = new ArrayList<>();
            ArrayList<ILineDataSet> lineDataSets_vz = new ArrayList<>();

            LineDataSet lineDataSet_refra_hs1 = new LineDataSet(yAXES_refra_hs1,"HS1");
            lineDataSet_refra_hs1.setDrawCircles(false);
            lineDataSet_refra_hs1.setDrawValues(false);
            lineDataSet_refra_hs1.setColor(Color.rgb(255,127,0));
            lineDataSet_refra_hs1.setHighlightEnabled(false);
            lineDataSet_refra_hs1.setLineWidth(1f);

        if (cyl_in != 0) {
            LineDataSet lineDataSet_refra_hs2 = new LineDataSet(yAXES_refra_hs2, "HS2");
            lineDataSet_refra_hs2.setDrawCircles(false);
            lineDataSet_refra_hs2.setDrawValues(false);
            lineDataSet_refra_hs2.setColor(Color.rgb(30, 144, 255));
            lineDataSet_refra_hs2.setHighlightEnabled(false);
            lineDataSet_refra_hs2.setLineWidth(1f);
            lineDataSets_refra.add(lineDataSet_refra_hs2);
```

```
}

    lineDataSets_refra.add(lineDataSet_refra_hs1);

    lc_glasrechner_refra_out.setData(new LineData(lineDataSets_refra));
    lc_glasrechner_refra_out.setVisibleXRangeMaximum(blickwinkel_in);
    lc_glasrechner_refra_out.getDescription().setText("Refraktionsfehler");
    lc_glasrechner_refra_out.getDescription().setTextColor(Color.rgb(255,255,255));
    lc_glasrechner_refra_out.getDescription().setTextSize(9f);
    lc_glasrechner_refra_out.getLegend().setTextColor(Color.rgb(255,255,255));

    LimitLine Zero = new LimitLine(0f);
    Zero.setLineColor(Color.rgb(255,255,255));
    Zero.setLineWidth(1f);

    XAxis xAxis = lc_glasrechner_refra_out.getXAxis();
    xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
    xAxis.setTextColor(Color.rgb(255,255,255));
    xAxis.setDrawLabels(true);

    YAxis yleft = lc_glasrechner_refra_out.getAxisLeft();
    yleft.setEnabled(false);

    YAxis yright = lc_glasrechner_refra_out.getAxisRight();
    yright.setTextColor(Color.rgb(255,255,255));
    yright.setDrawLabels(true);
    yright.addLimitLine(Zero);
    yright.setDrawLimitLinesBehindData(true);

    lc_glasrechner_refra_out.invalidate();


    LineDataSet lineDataSet_asti_hs1 = new LineDataSet(yAXES_asti_hs1,"HS1");
    lineDataSet_asti_hs1.setDrawCircles(false);
    lineDataSet_asti_hs1.setDrawValues(false);
    lineDataSet_asti_hs1.setColor(Color.rgb(255,127,0));
    lineDataSet_asti_hs1.setHighlightEnabled(false);
    lineDataSet_asti_hs1.setLineWidth(1f);

if (cyl_in != 0) {
    LineDataSet lineDataSet_asti_hs2 = new LineDataSet(yAXES_asti_hs2, "HS2");
    lineDataSet_asti_hs2.setDrawCircles(false);
    lineDataSet_asti_hs2.setDrawValues(false);
    lineDataSet_asti_hs2.setColor(Color.rgb(30, 144, 255));
    lineDataSet_asti_hs2.setHighlightEnabled(false);
    lineDataSet_asti_hs2.setLineWidth(1f);
    lineDataSets_asti.add(lineDataSet_asti_hs2);
}

    lineDataSets_asti.add(lineDataSet_asti_hs1);


    lc_glasrechner_asti_out.setData(new LineData(lineDataSets_asti));
    lc_glasrechner_asti_out.setVisibleXRangeMaximum(blickwinkel_in);
    lc_glasrechner_asti_out.getDescription().setText("Astigmatismus");
    lc_glasrechner_asti_out.getDescription().setTextColor(Color.rgb(255,255,255));
    lc_glasrechner_asti_out.getDescription().setTextSize(9f);
    lc_glasrechner_asti_out.getLegend().setTextColor(Color.rgb(255,255,255));

    Zero.setLineColor(Color.rgb(255,255,255));
    Zero.setLineWidth(1f);

    xAxis = lc_glasrechner_asti_out.getXAxis();
    xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
    xAxis.setTextColor(Color.rgb(255,255,255));
    xAxis.setDrawLabels(true);

    yleft = lc_glasrechner_asti_out.getAxisLeft();
    yleft.setEnabled(false);

    yright = lc_glasrechner_asti_out.getAxisRight();
    yright.setTextColor(Color.rgb(255,255,255));
    yright.setDrawLabels(true);
    yright.addLimitLine(Zero);
    yright.setDrawLimitLinesBehindData(true);

    lc_glasrechner_asti_out.invalidate();


    LineDataSet lineDataSet_vz_hs1 = new LineDataSet(yAXES_vz_hs1,"HS1");
    lineDataSet_vz_hs1.setDrawCircles(false);
    lineDataSet_vz_hs1.setDrawValues(false);
    lineDataSet_vz_hs1.setColor(Color.rgb(255,127,0));
    lineDataSet_vz_hs1.setHighlightEnabled(false);
    lineDataSet_vz_hs1.setLineWidth(1f);

if (cyl_in != 0) {
    LineDataSet lineDataSet_vz_hs2 = new LineDataSet(yAXES_vz_hs2, "HS2");
```

```
                    lineDataSet_vz_hs2.setDrawCircles(false);
                    lineDataSet_vz_hs2.setDrawValues(false);
                    lineDataSet_vz_hs2.setColor(Color.rgb(30, 144, 255));
                    lineDataSet_vz_hs2.setHighlightEnabled(false);
                    lineDataSet_vz_hs2.setLineWidth(1f);
                    lineDataSets_vz.add(lineDataSet_vz_hs2);
                }

                    lineDataSets_vz.add(lineDataSet_vz_hs1);


                    lc_glasrechner_vz_out.setData(new LineData(lineDataSets_vz));
                    lc_glasrechner_vz_out.setVisibleXRangeMaximum(blickwinkel_in);
                    lc_glasrechner_vz_out.getDescription().setText("Verzeichnung");
                    lc_glasrechner_vz_out.getDescription().setTextColor(Color.rgb(255,255,255));
                    lc_glasrechner_vz_out.getDescription().setTextSize(9f);
                    lc_glasrechner_vz_out.getLegend().setTextColor(Color.rgb(255,255,255));

                    Zero.setLineColor(Color.rgb(255,255,255));
                    Zero.setLineWidth(1f);

                    xAxis = lc_glasrechner_vz_out.getXAxis();
                    xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
                    xAxis.setTextColor(Color.rgb(255,255,255));
                    xAxis.setDrawLabels(true);

                    yleft = lc_glasrechner_vz_out.getAxisLeft();
                    yleft.setEnabled(false);

                    yright = lc_glasrechner_vz_out.getAxisRight();
                    yright.setTextColor(Color.rgb(255,255,255));
                    yright.setDrawLabels(true);
                    yright.addLimitLine(Zero);
                    yright.setDrawLimitLinesBehindData(true);

                    lc_glasrechner_vz_out.invalidate();


                 // Ausgabe der Abbildungsfehler für HS 1

                txt_glasrechner_grad_refra_out.setText(Html.fromHtml(Grad_Anzeige));
                txt_glasrechner_grad_asti_out.setText(Html.fromHtml(Grad_Anzeige));
                txt_glasrechner_grad_vz_out.setText(Html.fromHtml(Grad_Anzeige));

                txt_glasrechner_refra_hs1_out.setText(Html.fromHtml(hs1_refra_list));
                txt_glasrechner_asti_hs1_out.setText(Html.fromHtml(hs1_asti_list));
                txt_glasrechner_vz_hs1_out.setText(Html.fromHtml(hs1_vz_list));


                 // Berechnung uns Ausgabe der endgültigen Randdicken und der Bauhöhe

                t1 = r1 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
                        (Math.pow(r1, 2))))));
                t2 = r2 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
                        (Math.pow(r2, 2))))));

                dr1 = d_in - t1 + t2;

                double V1, V2, VGes, Gewicht;

                V1 = Math.PI * (dr1 * Math.pow((durchmesser_in/2), 2) + Math.pow(t1, 2) *
                        (r1 - (t1/3)) - Math.pow(t2, 2) * (r2 - (t2/3)));

                if (cyl_in != 0) {
                    t3 = r3 * (1 - Math.sqrt(1 - ((Math.pow((durchmesser_in / 2), 2) /
                            (Math.pow(r3, 2))))));


                    dr2 = d_in - t1 + t3;

                    txt_glasrechner_dr3_out.setText(Double.toString(Math.round(dr2*10000)/10000.0));

                    V2 = Math.PI * (dr2 * Math.pow((durchmesser_in/2), 2) + Math.pow(t1, 2) *
                            (r1 - (t1/3)) - Math.pow(t3, 2) * (r3 - (t3/3)));

                    VGes = ((V1 + V2) / 2) / 1000;


txt_glasrechner_volumen_out.setText(Double.toString(Math.round(VGes*100)/100.0));

                    Gewicht = VGes * rho_in;

                    txt_glasrechner_gewicht_out.setText(
                            Double.toString(Math.round(Gewicht*100)/100.0));

                    if (dr1 > dr2) {
                        txt_glasrechner_bauhoehe_out.setText(
                                Double.toString(Math.round((t1+dr1)*100)/100.0));
```

```
                    } else {
                        txt_glasrechner_bauhoehe_out.setText(
                                Double.toString(Math.round((t1+dr2)*100)/100.0));
                    }
                } else {
                    txt_glasrechner_bauhoehe_out.setText(
                            Double.toString(Math.round((t1+dr1)*100)/100.0));
                    V1 = V1/1000;
                    txt_glasrechner_volumen_out.setText(Double.toString(Math.round(V1*100)/100.0));
                    Gewicht = V1 * rho_in;
                    txt_glasrechner_gewicht_out.setText(
                            Double.toString(Math.round(Gewicht*100)/100.0));
                }

                txt_glasrechner_dr2_out.setText(Double.toString(Math.round(dr1*10000)/10000.0));


                if (cyl_in != 0) {

                     // Ausgabe der Abbildungsfehler für HS 2

                    txt_glasrechner_refra_hs2_out.setText(Html.fromHtml(hs2_refra_list));
                    txt_glasrechner_asti_hs2_out.setText(Html.fromHtml(hs2_asti_list));
                    txt_glasrechner_vz_hs2_out.setText(Html.fromHtml(hs2_vz_list));

                     // Sichtbarkeit für sämtliche Bezeichungs- und Einheitenobjekte für HS 2

                    txt_HS2_wunsch.setVisibility(View.VISIBLE);
                    txt_glasrechner_hs2_out_einheit.setVisibility(View.VISIBLE);
                    txt_F3_real.setVisibility(View.VISIBLE);
                    txt_glasrechner_f3_real_out_einheit.setVisibility(View.VISIBLE);
                    txt_R3.setVisibility(View.VISIBLE);
                    txt_glasrechner_r3_out_einheit.setVisibility(View.VISIBLE);
                    txt_DR3.setVisibility(View.VISIBLE);
                    txt_glasrechner_dr3_out_einheit.setVisibility(View.VISIBLE);
                    txt_glasrechner_hs2_refra.setVisibility(View.VISIBLE);
                    txt_glasrechner_hs2_asti.setVisibility(View.VISIBLE);
                    txt_glasrechner_hs2_vz.setVisibility(View.VISIBLE);

                    txt_glasrechner_hs2_out.setVisibility(View.VISIBLE);
                    txt_glasrechner_f3_real_out.setVisibility(View.VISIBLE);
                    txt_glasrechner_r3_out.setVisibility(View.VISIBLE);
                    txt_glasrechner_dr3_out.setVisibility(View.VISIBLE);
                    txt_glasrechner_refra_hs2_out.setVisibility(View.VISIBLE);
                    txt_glasrechner_asti_hs2_out.setVisibility(View.VISIBLE);
                    txt_glasrechner_vz_hs2_out.setVisibility(View.VISIBLE);
                }
                if (cb_glasrechner_werkzeugindex.isChecked()) {
                    txt_F2_wkn.setVisibility(View.VISIBLE);
                    txt_glasrechner_f2_wkn_out.setVisibility(View.VISIBLE);
                    txt_glasrechner_f2_wkn_out_einheit.setVisibility(View.VISIBLE);
                    if (cyl_in != 0) {
                        txt_F3_wkn.setVisibility(View.VISIBLE);
                        txt_glasrechner_f3_wkn_out.setVisibility(View.VISIBLE);
                        txt_glasrechner_f3_wkn_out_einheit.setVisibility(View.VISIBLE);
                    }
                }
            }
        });
    }
}
```

p) Programmtext – Info.java

```
package de.HS_Aalen.OptikFormelrechner;

import android.content.res.Configuration;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;


public class Info extends AppCompatActivity {

    TextView impressum_inhalt, datenschutz_inhalt, haftung_inhalt, opensource_inhalt;
    ImageView banner_hs;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.info);

        banner_hs = (ImageView) findViewById(R.id.banner_hs);

        // Zeige das Bild des Hochschullogos "banner_hs" nur in Porträt- und nicht im Landscapemodus
        if (getResources().getConfiguration().orientation == Configuration.ORIENTATION_LANDSCAPE) {
            banner_hs.setVisibility(View.GONE);
        }
        else {
            banner_hs.setVisibility(View.VISIBLE);
        }

        impressum_inhalt = (TextView) findViewById(R.id.impressum_inhalt);
        datenschutz_inhalt = (TextView) findViewById(R.id.datenschutz_inhalt);
        haftung_inhalt = (TextView) findViewById(R.id.haftung_inhalt);
        opensource_inhalt = (TextView) findViewById(R.id.opensource_inhalt);

        // Da Zeilenumbrüche in einem TextView direkt nicht möglich sind, wird an dieser Stelle
        // manuell mit Hilfe von Html-Code der Text und das Textlayout erstellt
        impressum_inhalt.setText(Html.fromHtml("Entwickler: René Nierath<br><br>Studiengang " +
                "Augenoptik und Hörakustik der Hochschule Aalen<br><br>Versionsnummer: " +
                "1.0<br><br>Impressum:<br>https://www.hs-aalen.de/pages/impressum"));
        datenschutz_inhalt.setText(Html.fromHtml("Diese App speichert keinerlei Daten und " +
                "benötigt keine Berechtigungen die auf das Gerät zugreifen."));
        haftung_inhalt.setText(Html.fromHtml("Trotz sorgfältiger Kontrolle aller Formeln haften" +
                " weder der Entwickler noch die Hochschule Aalen für die Richtigkeit der " +
                "berechneten Ergebnisse."));
        opensource_inhalt.setText(Html.fromHtml("Copyright 2016 Philipp Jahoda<br><br>" +
                "https://github.com/PhilJay/MPAndroidChart<br><br>Licensed under the Apache " +
                "License, Version 2.0 (the \"License\"); you may not use this file except in" +
                " compliance with the License. You may obtain a copy of the License at<br><br>" +
                "http://www.apache.org/licenses/LICENSE-2.0<br><br>Unless required by " +
                "applicable law or agreed to in writing, software distributed under the License " +
                "is distributed on an \"AS IS\" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF " +
                "ANY KIND, either express or implied. See the License for the specific language " +
                "governing permissions and limitations under the License.<br>"));


    }
}
```

q) Layout – main_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">


    <ImageView
        android:id="@+id/keyvisual"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:cropToPadding="false"
        app:srcCompat="@drawable/keyvisual" />

    <ListView
        android:id="@+id/ListView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:footerDividersEnabled="false"
        android:headerDividersEnabled="false"></ListView>


</LinearLayout>
```

r) Layout – raytracing_start.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <GridLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true">

        <TextView
            android:text="Anzahl der Flächen:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/textView"
            android:layout_weight="1"
            android:textSize="20dp"
            android:layout_row="0"
            android:layout_column="0" />

        <EditText
            android:layout_height="wrap_content"
            android:inputType="number"
            android:ems="10"
            android:id="@+id/eT"
            android:layout_row="0"
            android:layout_column="1"
            android:layout_width="102dp"
            android:layout_gravity="right"
            android:textAlignment="center" />

        <CheckBox
            android:text="@string/s_unendlich"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/check_s1"
            android:textSize="16dp"
            android:layout_row="1"
            android:layout_column="0"
            android:layout_columnSpan="2" />

        <Button
            android:text="Start"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:id="@+id/bT"
            android:layout_weight="1"
            android:layout_row="2"
            android:layout_column="0"
            android:layout_columnSpan="2"
            android:textAlignment="center"
            android:textAllCaps="false"
            android:layout_gravity="center_horizontal" />
    </GridLayout>

</RelativeLayout>
```

s) Layout – raytracing_berechnung.xml

```
<?xml version="1.0" encoding="utf-8"?>

 <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/scrollView">


    </ScrollView>
```

t) Layout – exaktes_raytracing_start.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    >

    <GridLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true">

        <TextView
            android:text="Anzahl der Flächen:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/textView"
            android:layout_weight="1"
            android:textSize="20dp"
            android:layout_row="0"
            android:layout_column="0" />

        <EditText
            android:layout_height="wrap_content"
            android:inputType="number"
            android:ems="10"
            android:id="@+id/eT"
            android:layout_row="0"
            android:layout_column="1"
            android:layout_width="102dp"
            android:layout_gravity="right"
            android:textAlignment="center" />

        <Button
            android:text="Start"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/bT"
            android:layout_weight="1"
            android:layout_row="1"
            android:layout_column="0"
            android:layout_columnSpan="2"
            android:textAlignment="center"
            android:textAllCaps="false"
            android:layout_gravity="center_horizontal" />
    </GridLayout></RelativeLayout>
```

u) Layout – exaktes_raytracing_berechnung.xml

```
<?xml version="1.0" encoding="utf-8"?>

    <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/scrollView"
        >

    </ScrollView>
```

v) Layout – powervektoren.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    >

    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        >

        <GridLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:id="@+id/dub"
            >

            <TextView
                android:text="Glas 1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/Glas_1"
                android:layout_row="0"
                android:layout_column="0"
                android:layout_columnSpan="3"
                android:textSize="34sp" />

            <TextView
                android:text="Sph:"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/Sph_1"
                android:layout_row="1"
                android:layout_column="0"
                android:textSize="22sp" />

            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:ems="10"
                android:id="@+id/in_sph_1"
                android:layout_row="1"
                android:layout_column="1"
                android:inputType="numberSigned|numberDecimal"
                android:textSize="22sp"
                android:width="100dp"
                android:textAlignment="textEnd"
                android:layout_gravity="end"
                android:layout_columnWeight="2" />

            <TextView
                android:text="dpt"
                android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/dpt1"
        android:layout_row="1"
        android:layout_column="2"
        android:layout_gravity="right"
        android:textSize="22sp"
        android:textAlignment="gravity" />

    <TextView
        android:text="Cyl:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Cyl_1"
        android:layout_row="2"
        android:layout_column="0"
        android:textSize="22sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:ems="10"
        android:id="@+id/in_cyl_1"
        android:layout_row="2"
        android:layout_column="1"
        android:textSize="22sp"
        android:width="100dp"
        android:textAlignment="textEnd"
        android:layout_gravity="end"
        android:layout_columnWeight="2" />

    <TextView
        android:text="dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/dpt2"
        android:layout_row="2"
        android:layout_column="2"
        android:layout_gravity="right"
        android:textSize="22sp"
        android:textAlignment="gravity" />

    <TextView
        android:text="Achse:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Achse_1"
        android:layout_row="3"
        android:layout_column="0"
        android:textSize="22sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:ems="10"
        android:id="@+id/in_achse_1"
        android:layout_row="3"
        android:layout_column="1"
        android:textSize="22sp"
        android:width="100dp"
        android:textAlignment="textEnd"
        android:layout_gravity="end"
        android:layout_columnWeight="2" />

    <TextView
        android:text="°"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/dpt3"
        android:layout_row="3"
        android:layout_column="2"
        android:layout_gravity="start"
        android:textSize="22sp"
        android:textAlignment="textStart" />

    <TextView
        android:text="Glas 2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Glas_2"
        android:layout_row="4"
        android:layout_column="0"
        android:layout_columnSpan="3"
        android:textSize="34sp"
        android:layout_gravity="start" />

    <TextView
```

```
        android:text="Sph:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Sph_2"
        android:layout_row="5"
        android:layout_column="0"
        android:textSize="22sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:ems="10"
        android:id="@+id/in_sph_2"
        android:layout_row="5"
        android:layout_column="1"
        android:textSize="22sp"
        android:width="100dp"
        android:textAlignment="textEnd"
        android:layout_gravity="end"
        android:layout_columnWeight="2" />

    <TextView
        android:text="dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/dpt4"
        android:layout_row="5"
        android:layout_column="2"
        android:layout_gravity="right"
        android:textSize="22sp"
        android:textAlignment="gravity" />

    <TextView
        android:text="Cyl:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Cyl_2"
        android:layout_row="6"
        android:layout_column="0"
        android:textSize="22sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:ems="10"
        android:id="@+id/in_cyl_2"
        android:layout_row="6"
        android:layout_column="1"
        android:textSize="22sp"
        android:width="100dp"
        android:textAlignment="textEnd"
        android:layout_gravity="end"
        android:layout_columnWeight="2" />

    <TextView
        android:text="dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/dpt5"
        android:layout_row="6"
        android:layout_column="2"
        android:layout_gravity="right"
        android:textSize="22sp"
        android:textAlignment="gravity" />

    <TextView
        android:text="Achse:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Achse_2"
        android:layout_row="7"
        android:layout_column="0"
        android:textSize="22sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:ems="10"
        android:id="@+id/in_achse_2"
        android:layout_row="7"
        android:layout_column="1"
        android:textSize="22sp"
        android:width="100dp"
        android:textAlignment="textEnd"
        android:layout_gravity="end"
```

```
        android:layout_columnWeight="2" />

    <TextView
        android:text="°"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/dpt6"
        android:layout_row="7"
        android:layout_column="2"
        android:layout_gravity="start"
        android:textSize="22sp"
        android:textAlignment="textStart" />

    <TextView
        android:text="Ergebnis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Result"
        android:layout_row="8"
        android:layout_column="0"
        android:layout_columnSpan="3"
        android:textSize="34sp" />

    <TextView
        android:text="Sph:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Sph_3"
        android:layout_row="9"
        android:layout_column="0"
        android:textSize="22sp" />

    <TextView
        android:text=""
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/out_sph_3"
        android:layout_row="9"
        android:layout_column="1"
        android:textSize="22sp"
        android:layout_gravity="end"
        android:maxLength="10"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2" />

    <TextView
        android:text=" dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/dpt7"
        android:layout_row="9"
        android:layout_column="2"
        android:layout_gravity="right"
        android:textSize="22sp"
        android:textAlignment="gravity" />

    <TextView
        android:text="Cyl:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Cyl_3"
        android:layout_row="10"
        android:layout_column="0"
        android:textSize="22sp" />

    <TextView
        android:text=""
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/out_cyl_3"
        android:layout_row="10"
        android:layout_column="1"
        android:textSize="22sp"
        android:layout_gravity="end"
        android:maxLength="10"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2" />

    <TextView
        android:text=" dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/dpt8"
        android:layout_row="10"
        android:layout_column="2"
        android:layout_gravity="right"
        android:textSize="22sp"
        android:textAlignment="gravity" />
```

```
            <TextView
                android:text="Achse:"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/Achse_3"
                android:layout_row="11"
                android:layout_column="0"
                android:textSize="22sp" />

            <TextView
                android:text=""
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/out_achse_3"
                android:layout_row="11"
                android:layout_column="1"
                android:textSize="22sp"
                android:layout_gravity="end"
                android:textAlignment="textEnd"
                android:layout_columnWeight="2" />

            <TextView
                android:text=" °"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/dpt9"
                android:layout_row="11"
                android:layout_column="2"
                android:layout_gravity="start"
                android:textSize="22sp"
                android:textAlignment="textStart" />

            <GridLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_row="12"
                android:layout_column="0"
                android:layout_columnSpan="3"
                android:layout_gravity="center">

                <Button
                    android:text="Clear"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:layout_row="0"
                    android:layout_column="0"
                    android:id="@+id/bt_clear"
                    android:textAllCaps="false" />

                <Button
                    android:text="Berechnen"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:layout_row="0"
                    android:layout_column="1"
                    android:id="@+id/button"
                    android:textAllCaps="false" />

                <Button
                    android:text="+ 3. Glas"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:layout_row="0"
                    android:layout_column="3"
                    android:id="@+id/bt_plus"
                    android:textAllCaps="false" />

            </GridLayout>


        </GridLayout>


    </ScrollView>


</RelativeLayout>
```

w) Layout – schott.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        >

    <GridLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:text="Vorauswahl:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/txt_farbe"
            android:layout_weight="1"
            android:textSize="20sp"
            android:layout_row="0"
            android:layout_column="0" />

        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/Farbauswahl"
            android:layout_row="0"
            android:layout_column="1"
            android:layout_columnSpan="3"
            android:layout_gravity="right"
            android:gravity="end"
            android:textAlignment="viewEnd"
            />


        <TextView
            android:text="Glasmaterial:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/glasart"
            android:layout_weight="1"
            android:textSize="20sp"
            android:layout_columnSpan="2"
            android:layout_row="2"
            android:layout_column="0" />

        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/Auswahlglas"
            android:layout_row="2"
            android:layout_column="2"
            android:layout_columnSpan="2"
            android:layout_gravity="right"
            android:gravity="end"
            android:textAlignment="viewEnd"
            />

        <TextView
            android:text="Wellenlänge:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/txt_wellenlaenge"
            android:layout_weight="1"
            android:textSize="20sp"
            android:layout_columnSpan="2"
            android:layout_row="1"
            android:layout_column="0"
            android:layout_gravity="center_vertical" />

        <GridLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_row="1"
            android:layout_column="2"
```

```
        android:layout_columnSpan="2"
        android:layout_gravity="end"
        >

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal"
        android:ems="10"
        android:id="@+id/et_wellenlaenge"
        android:layout_column="0"
        android:layout_width="102dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:textSize="20sp" />

    <TextView
        android:text="nm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/nm2"
        android:textSize="20sp"
        android:layout_column="1"
        android:layout_gravity="end" />
    </GridLayout>


    <TextView
        android:text="@string/txt_a0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_a0"
        android:textSize="20sp"
        android:layout_columnSpan="1"
        android:layout_row="3"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <TextView
        android:text="@string/txt_a1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_a1"
        android:textSize="20sp"
        android:layout_columnSpan="1"
        android:layout_row="4"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <TextView
        android:text="@string/txt_a2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_a2"
        android:textSize="20sp"
        android:layout_columnSpan="1"
        android:layout_row="5"
        android:layout_column="0"
        android:paddingBottom="2dp" />

    <TextView
        android:text="@string/txt_a3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_a3"
        android:textSize="20sp"
        android:layout_columnSpan="1"
        android:layout_row="6"
        android:layout_column="0"
        android:paddingBottom="2sp"
        />

    <TextView
        android:text="@string/txt_a4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_a4"
        android:textSize="20sp"
        android:layout_columnSpan="1"
        android:layout_row="7"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <TextView
        android:text="@string/txt_a5"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_a5"
        android:textSize="20sp"
        android:layout_columnSpan="1"
        android:layout_row="8"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_a0"
        android:layout_row="3"
        android:layout_column="1"
        android:layout_width="182dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="3"
        android:textSize="20sp" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_a1"
        android:layout_row="4"
        android:layout_column="1"
        android:layout_width="182dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="3"
        android:textSize="20sp" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_a2"
        android:layout_row="5"
        android:layout_column="1"
        android:layout_width="182dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="3"
        android:textSize="20sp" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_a3"
        android:layout_row="6"
        android:layout_column="1"
        android:layout_width="182dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="3"
        android:textSize="20sp" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_a4"
        android:layout_row="7"
        android:layout_column="1"
        android:layout_width="182dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="3"
        android:textSize="20sp" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_a5"
        android:layout_row="8"
        android:layout_column="1"
        android:layout_width="182dp"
```

```
            android:layout_gravity="end"
            android:gravity="end"
            android:textAlignment="viewEnd"
            android:layout_columnSpan="3"
            android:textSize="20sp" />

        <TextView
            android:text="n:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/txt_n"
            android:textSize="20sp"
            android:layout_columnSpan="1"
            android:layout_row="9"
            android:layout_column="0"
            android:paddingBottom="2dp"
            />

        <TextView
            android:text=""
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/txt_result"
            android:textSize="20sp"
            android:layout_columnSpan="1"
            android:layout_row="9"
            android:layout_column="3"
            android:layout_gravity="right"
            android:gravity="right"
            />


        <Button
            android:text="Berechnen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/bT"
            android:layout_row="10"
            android:layout_column="0"
            android:layout_columnSpan="4"
            android:textAlignment="center"
            android:textAllCaps="false"
            android:layout_gravity="center_horizontal" />

    </GridLayout>

</ScrollView>

</RelativeLayout>
```

x) Layout – transmission_start.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <GridLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true">

        <TextView
            android:text="Anzahl der Flächen:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/textView"
            android:layout_weight="1"
            android:textSize="20dp"
            android:layout_row="0"
            android:layout_column="0" />

        <EditText
            android:layout_height="wrap_content"
            android:inputType="number"
            android:ems="10"
            android:id="@+id/eT"
            android:layout_row="0"
            android:layout_column="1"
            android:layout_width="102dp"
            android:layout_gravity="right"
            android:textAlignment="center" />

        <Button
            android:text="Start"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/bT"
            android:layout_weight="1"
            android:layout_row="1"
            android:layout_column="0"
            android:layout_columnSpan="2"
            android:textAlignment="center"
            android:textAllCaps="false"
            android:layout_gravity="center_horizontal" />
    </GridLayout>

</RelativeLayout>
```

y) Layout – transmission_berechnung.xml

```
<?xml version="1.0" encoding="utf-8"?>


    <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/scrollView"
        >

    </ScrollView>
```

z) Layout – shape.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_alignParentRight="true"
        android:layout_centerHorizontal="false">

    <GridLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        >

        <Spinner
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/spinner_shape_D"
            android:layout_row="0"
            android:layout_column="0"
            />

        <EditText
            android:layout_height="wrap_content"
            android:inputType="number|numberDecimal|numberSigned"
            android:ems="10"
            android:id="@+id/et_DGes"
            android:layout_row="0"
            android:layout_column="1"
            android:layout_width="80dp"
            android:layout_gravity="end"
            android:gravity="end"
            android:textAlignment="viewEnd"
            android:layout_columnWeight="2"
            android:textSize="20dp" />

        <TextView
            android:id="@+id/txt_D_dpt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20dp"
            android:layout_row="0"
            android:layout_column="2"
            android:layout_gravity="end"
            />

        <GridLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_row="1"
            android:layout_column="0"
            android:layout_columnSpan="3"
            >

            <Spinner
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/shape_s"
                android:layout_row="0"
                android:layout_rowSpan="2"
                android:layout_column="0"
                android:layout_columnSpan="1"
                android:layout_gravity="start"
                android:gravity="start"
                android:textAlignment="gravity"
                />

            <EditText
                android:layout_height="wrap_content"
                android:inputType="number|numberDecimal|numberSigned"
                android:ems="10"
                android:id="@+id/et_s"
                android:layout_row="0"
                android:layout_column="1"
                android:layout_width="80dp"
                android:layout_gravity="end"
```

```
            android:gravity="end"
            android:textAlignment="viewEnd"
            android:layout_columnSpan="2"
            android:textSize="20dp" />

        <TextView
            android:text=" mm"
            android:id="@+id/txt_shape_s_mm"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20dp"
            android:layout_row="0"
            android:layout_column="3" />

        <CheckBox
            android:text="s = -∞"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/cb_shape_s_munendlich"
            android:layout_row="1"
            android:layout_column="1"
            android:visibility="gone"
            android:layout_gravity="end|center_vertical"
            android:textAlignment="textEnd"
            android:layout_columnWeight="3" />

        <CheckBox
            android:text="s = ∞"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/cb_shape_s_unendlich"
            android:layout_row="1"
            android:layout_column="2"
            android:visibility="gone"
            android:layout_gravity="end|center_vertical"
            android:textAlignment="textEnd"
            android:layout_columnWeight="4"/>

    </GridLayout>

    <TextView
        android:text="n :"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_n"
        android:paddingBottom="2dp"
        android:textSize="20dp"
        android:layout_row="2"
        android:layout_column="0" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal"
        android:ems="10"
        android:id="@+id/et_n"
        android:layout_row="2"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:textSize="20dp" />

    <GridLayout
        android:id="@+id/grid_result"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_row="3"
        android:layout_column="0"
        android:layout_columnSpan="3"
        android:visibility="gone" >

        <TextView
            android:text="Ergebnis :"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/txt_Ergebnis"
            android:textSize="25dp"
            android:layout_row="0"
            android:layout_column="0"
            android:layout_columnSpan="3"
            android:paddingBottom="10dp"
            />

        <TextView
            android:text="P :"
            android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/txt_P"
        android:textSize="20dp"
        android:layout_row="1"
        android:layout_column="0"
        android:paddingBottom="10dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_P_result"
        android:textSize="20dp"
        android:layout_row="1"
        android:layout_column="1"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2"
        />

    <TextView
        android:text="Min. sph. Aberration"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_sph_Ab"
        android:textSize="25dp"
        android:layout_row="2"
        android:layout_column="0"
        android:layout_columnSpan="3"
        />

    <TextView
        android:text="@string/txt_shape_Shape_sph"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_Shape_sph"
        android:textSize="20dp"
        android:layout_row="3"
        android:layout_column="0"
        android:paddingBottom="6dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_Shape_sph_result"
        android:textSize="20dp"
        android:layout_row="3"
        android:layout_column="1"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2"
        />

    <TextView
        android:text="@string/txt_shape_r1_sph"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_r1_sph"
        android:textSize="20dp"
        android:layout_row="4"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_r1_sph_result"
        android:textSize="20dp"
        android:layout_row="4"
        android:layout_column="1"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2"
        />

    <TextView
        android:text=" mm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:layout_row="4"
        android:layout_column="2" />

    <TextView
        android:text="@string/txt_shape_r2_sph"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_r2_sph"
        android:textSize="20dp"
        android:layout_row="5"
        android:layout_column="0"
        android:paddingBottom="10dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_r2_sph_result"
        android:textSize="20dp"
        android:layout_row="5"
        android:layout_column="1"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2"
        />

    <TextView
        android:text=" mm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:layout_row="5"
        android:layout_column="2" />



    <TextView
        android:text="Komafreie Linse"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_koma_Ab"
        android:textSize="25dp"
        android:layout_row="6"
        android:layout_column="0"
        android:layout_columnSpan="3"
        />

    <TextView
        android:text="@string/txt_shape_Shape_koma"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_Shape_koma"
        android:textSize="20dp"
        android:layout_row="7"
        android:layout_column="0"
        android:paddingBottom="6dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_Shape_koma_result"
        android:textSize="20dp"
        android:layout_row="7"
        android:layout_column="1"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2"
        />

    <TextView
        android:text="@string/txt_shape_r1_koma"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_r1_koma"
        android:textSize="20dp"
        android:layout_row="8"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_shape_r1_koma_result"
        android:textSize="20dp"
        android:layout_row="8"
        android:layout_column="1"
        android:layout_gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2"
        />
```

```
            <TextView
                android:text=" mm"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20dp"
                android:layout_row="8"
                android:layout_column="2" />

            <TextView
                android:text="@string/txt_shape_r2_koma"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/txt_shape_r2_koma"
                android:textSize="20dp"
                android:layout_row="9"
                android:layout_column="0"
                android:paddingBottom="2dp"
                />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/txt_shape_r2_koma_result"
                android:textSize="20dp"
                android:layout_row="9"
                android:layout_column="1"
                android:layout_gravity="end"
                android:textAlignment="textEnd"
                android:layout_columnWeight="2"
                />

            <TextView
                android:text=" mm"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20dp"
                android:layout_row="9"
                android:layout_column="2" />
        </GridLayout>



        <Button
            android:text="Berechnen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/bT"
            android:layout_row="4"
            android:layout_column="0"
            android:layout_columnSpan="3"
            android:textAlignment="center"
            android:textAllCaps="false"
            android:layout_gravity="center_horizontal" />

    </GridLayout>

</ScrollView>

</RelativeLayout>
```

aa) Layout – glasberechnung.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_alignParentRight="true"
        android:layout_centerHorizontal="false">

        <GridLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">

            <TextView
                android:text="Sph:"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20sp"
                android:layout_row="0"
                android:layout_column="0" />

            <EditText
                android:layout_height="wrap_content"
                android:inputType="number|numberDecimal|numberSigned"
                android:ems="10"
                android:id="@+id/et_glasrechner_sph_in"
                android:layout_row="0"
                android:layout_column="1"
                android:layout_width="80dp"
                android:layout_gravity="end"
                android:gravity="end"
                android:textAlignment="viewEnd"
                android:layout_columnSpan="1"
                android:layout_columnWeight="2"
                android:textSize="20sp" />

            <TextView
                android:text="dpt"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20sp"
                android:layout_row="0"
                android:layout_column="2"
                android:layout_gravity="end"
                />

            <TextView
                android:text="Cyl:"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20sp"
                android:layout_row="1"
                android:layout_column="0" />

            <EditText
                android:layout_height="wrap_content"
                android:inputType="number|numberDecimal|numberSigned"
                android:ems="10"
                android:id="@+id/et_glasrechner_cyl_in"
                android:layout_row="1"
                android:layout_column="1"
                android:layout_width="80sp"
                android:layout_gravity="end"
                android:gravity="end"
                android:textAlignment="viewEnd"
                android:layout_columnSpan="1"
                android:layout_columnWeight="2"
                android:textSize="20sp" />

            <TextView
                android:text="dpt"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20sp"
                android:layout_row="1"
                android:layout_column="2"
```

```
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_f1"
        android:id="@+id/txt_glasrechner_f1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="2"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_glasrechner_f1_in"
        android:layout_row="2"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="2"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="Brechungsindex n:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="3"
        android:layout_column="0" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal"
        android:ems="10"
        android:id="@+id/et_glasrechner_n_in"
        android:layout_row="3"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="Mittendicke d:"
        android:id="@+id/txt_glasrechner_d"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="4"
        android:layout_column="0" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal"
        android:ems="10"
        android:id="@+id/et_glasrechner_d_in"
        android:layout_row="4"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
```

```
        android:id="@+id/txt_glasrechner_d_einheit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="4"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:id="@+id/txt_glasrechner_dr"
        android:text="@string/txt_glasrechner_dr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="5"
        android:layout_column="0"
        android:paddingBottom="2dp"
        />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal"
        android:ems="10"
        android:id="@+id/et_glasrechner_dr_in"
        android:layout_row="5"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
        android:id="@+id/txt_glasrechner_dr_einheit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="5"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="Durchmesser ø:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="6"
        android:layout_column="0" />

    <EditText
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/et_glasrechner_durchmesser_in"
        android:layout_row="6"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="6"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="Dichte ρ:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:layout_row="7"
        android:layout_column="0" />

    <EditText
```

```
        android:text="1.2"
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal"
        android:ems="10"
        android:id="@+id/et_glasrechner_rho_in"
        android:layout_row="7"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="@string/txt_glasrechner_rho_einheit"
        android:id="@+id/txt_glasrechner_rho_einheit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="7"
        android:layout_column="2"
        android:layout_gravity="end"
        android:paddingBottom="3dp"
        />

    <TextView
        android:text="b':"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="8"
        android:layout_column="0" />

    <EditText
        android:text="27.5"
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal"
        android:ems="10"
        android:id="@+id/et_glasrechner_b_1_in"
        android:layout_row="8"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="8"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="s:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="9"
        android:layout_column="0" />

    <EditText
        android:text="-1000"
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal|numberSigned"
        android:ems="10"
        android:id="@+id/et_glasrechner_s_1_in"
        android:layout_row="9"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="m"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="9"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="Max. Blickwinkel:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="10"
        android:layout_column="0" />

    <EditText
        android:text="60"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/et_glasrechner_blickwinkel_in"
        android:layout_row="10"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="°"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="10"
        android:layout_column="2"
        android:layout_gravity="left"
        />

    <TextView
        android:text="Abstufung:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="11"
        android:layout_column="0" />

    <EditText
        android:text="5"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/et_glasrechner_abstufung_in"
        android:layout_row="11"
        android:layout_column="1"
        android:layout_width="80dp"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="°"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="11"
        android:layout_column="2"
        android:layout_gravity="left"
        />

    <CheckBox
        android:text="Angaben im Werkzeugindex n = 1.525"
        android:id="@+id/cb_glasrechner_werkzeugindex"
        android:layout_row="12"
        android:layout_column="0"
        android:layout_columnSpan="3"
        />

    <CheckBox
        android:text="Rückflächen in 0.0625 dpt Abstufung"
        android:id="@+id/cb_glasrechner_flaechen_gerundet"
        android:layout_row="13"
        android:layout_column="0"
```

```
        android:layout_columnSpan="3"
        />


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/bT"
        android:text="Berechnen"
        android:textAllCaps="false"
        android:layout_gravity="center_horizontal"
        android:layout_row="14"
        android:layout_column="0"
        android:layout_columnSpan="3"
        />



    <GridLayout
        android:id="@+id/grid_glasrechner_result"
        android:layout_row="15"
        android:layout_column="0"
        android:layout_columnSpan="3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:visibility="gone"
        >

        <TextView
            android:text="@string/txt_glasrechner_HS1"
            android:id="@+id/txt_HS1_wunsch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_row="0"
            android:layout_column="0"
            android:paddingTop="30dp"
            android:paddingBottom="2dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/txt_glasrechner_hs1_out"
            android:layout_row="0"
            android:layout_column="1"
            android:layout_gravity="end"
            android:gravity="end"
            android:textAlignment="viewEnd"
            android:layout_columnSpan="1"
            android:layout_columnWeight="2"
            android:textSize="20sp" />

        <TextView
            android:text="dpt"
            android:id="@+id/txt_glasrechner_hs1_out_einheit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_row="0"
            android:layout_column="2"
            android:layout_gravity="end"
            />

        <TextView
            android:text="@string/txt_glasrechner_HS2"
            android:id="@+id/txt_HS2_wunsch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_row="1"
            android:layout_column="0"
            android:paddingBottom="2dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/txt_glasrechner_hs2_out"
            android:layout_row="1"
            android:layout_column="1"
            android:layout_gravity="end"
            android:gravity="end"
            android:textAlignment="viewEnd"
            android:layout_columnSpan="1"
            android:layout_columnWeight="2"
            android:textSize="20sp"
            />
```

```
<TextView
    android:text="dpt"
    android:id="@+id/txt_glasrechner_hs2_out_einheit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="1"
    android:layout_column="2"
    android:layout_gravity="end"
    />

<TextView
    android:text="@string/txt_glasrechner_F2_wkn"
    android:id="@+id/txt_F2_wkn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="2"
    android:layout_column="0"
    android:paddingTop="30dp"
    android:paddingBottom="8dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/txt_glasrechner_f2_wkn_out"
    android:layout_row="2"
    android:layout_column="1"
    android:layout_gravity="end"
    android:gravity="end"
    android:textAlignment="viewEnd"
    android:layout_columnSpan="1"
    android:layout_columnWeight="2"
    android:textSize="20sp"
    />

<TextView
    android:text="dpt"
    android:id="@+id/txt_glasrechner_f2_wkn_out_einheit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="2"
    android:layout_column="2"
    android:layout_gravity="end"
    />

<TextView
    android:text="@string/txt_glasrechner_F3_wkn"
    android:id="@+id/txt_F3_wkn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="3"
    android:layout_column="0"
    android:paddingBottom="8dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/txt_glasrechner_f3_wkn_out"
    android:layout_row="3"
    android:layout_column="1"
    android:layout_gravity="end"
    android:gravity="end"
    android:textAlignment="viewEnd"
    android:layout_columnSpan="1"
    android:layout_columnWeight="2"
    android:textSize="20sp"
    />

<TextView
    android:text="dpt"
    android:id="@+id/txt_glasrechner_f3_wkn_out_einheit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="3"
    android:layout_column="2"
    android:layout_gravity="end"
    />

<TextView
    android:text="@string/txt_glasrechner_F1_real"
    android:id="@+id/txt_F1_real"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="4"
        android:layout_column="0"
        android:paddingTop="30dp"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_f1_real_out"
        android:layout_row="4"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp"
        />

    <TextView
        android:text="dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="4"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_F2_real"
        android:id="@+id/txt_F2_real"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="5"
        android:layout_column="0"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_f2_real_out"
        android:layout_row="5"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="dpt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="5"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_F3_real"
        android:id="@+id/txt_F3_real"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="6"
        android:layout_column="0"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_f3_real_out"
        android:layout_row="6"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
```

```
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="dpt"
        android:id="@+id/txt_glasrechner_f3_real_out_einheit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="6"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_R1"
        android:id="@+id/txt_R1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="7"
        android:layout_column="0"
        android:paddingTop="30dp"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_r1_out"
        android:layout_row="7"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="7"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_R2"
        android:id="@+id/txt_R2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="8"
        android:layout_column="0"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_r2_out"
        android:layout_row="8"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="8"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_R3"
        android:id="@+id/txt_R3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:textSize="20sp"
        android:layout_row="9"
        android:layout_column="0"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_r3_out"
        android:layout_row="9"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
        android:id="@+id/txt_glasrechner_r3_out_einheit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="9"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_DR2"
        android:id="@+id/txt_DR2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="10"
        android:layout_column="0"
        android:paddingTop="30dp"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_dr2_out"
        android:layout_row="10"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="mm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="10"
        android:layout_column="2"
        android:layout_gravity="end"
        />

    <TextView
        android:text="@string/txt_glasrechner_DR3"
        android:id="@+id/txt_DR3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="11"
        android:layout_column="0"
        android:paddingBottom="8dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_dr3_out"
        android:layout_row="11"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />
```

```
<TextView
    android:text="mm"
    android:id="@+id/txt_glasrechner_dr3_out_einheit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="11"
    android:layout_column="2"
    android:layout_gravity="end"
    />

<TextView
    android:text="Bauhöhe:"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="12"
    android:layout_column="0"
    android:paddingTop="30dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/txt_glasrechner_bauhoehe_out"
    android:layout_row="12"
    android:layout_column="1"
    android:layout_gravity="end"
    android:gravity="end"
    android:textAlignment="viewEnd"
    android:layout_columnSpan="1"
    android:layout_columnWeight="2"
    android:textSize="20sp" />

<TextView
    android:text="mm"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="12"
    android:layout_column="2"
    android:layout_gravity="end"
    />

<TextView
    android:text="Volumen:"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="13"
    android:layout_column="0"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/txt_glasrechner_volumen_out"
    android:layout_row="13"
    android:layout_column="1"
    android:layout_gravity="end"
    android:gravity="end"
    android:textAlignment="viewEnd"
    android:layout_columnSpan="1"
    android:layout_columnWeight="2"
    android:textSize="20sp" />

<TextView
    android:text="cm³"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="13"
    android:layout_column="2"
    android:layout_gravity="end"
    />

<TextView
    android:text="Gewicht:"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="14"
    android:layout_column="0"
    />

<TextView
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/txt_glasrechner_gewicht_out"
        android:layout_row="14"
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="viewEnd"
        android:layout_columnSpan="1"
        android:layout_columnWeight="2"
        android:textSize="20sp" />

    <TextView
        android:text="g"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="14"
        android:layout_column="2"
        android:layout_gravity="left"
        />

    <GridLayout
        android:layout_row="15"
        android:layout_column="0"
        android:layout_columnSpan="3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        >


        <TextView
            android:text="Refraktionsfehler"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="25sp"
            android:layout_row="0"
            android:layout_column="0"
            android:paddingTop="30dp"
            android:layout_columnSpan="3"
            />

        <TextView
            android:text="@string/txt_glasrechner_HS1_table"
            android:id="@+id/txt_glasrechner_hs1_refra"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_row="1"
            android:layout_column="1"
            android:paddingBottom="2dp"
            android:layout_gravity="end"
            android:gravity="start"
            android:textAlignment="textStart"
            />

        <TextView
            android:text="@string/txt_glasrechner_HS2_table"
            android:id="@+id/txt_glasrechner_hs2_refra"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_row="1"
            android:layout_column="2"
            android:paddingBottom="2dp"
            android:layout_gravity="end"
            android:gravity="start"
            android:textAlignment="textStart"
            />

        <TextView
            android:text="Grad"
            android:id="@+id/txt_glasrechner_grad_refra_out"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="17sp"
            android:layout_row="2"
            android:layout_column="0"
            android:gravity="end"
            android:textAlignment="textEnd"
            />

        <TextView
            android:text="Refra HS 1"
            android:id="@+id/txt_glasrechner_refra_hs1_out"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="17sp"
            android:layout_row="2"
```

```
        android:layout_column="1"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="2"
        />

    <TextView
        android:text="Refra HS 2"
        android:id="@+id/txt_glasrechner_refra_hs2_out"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="17sp"
        android:layout_row="2"
        android:layout_column="2"
        android:layout_gravity="end"
        android:gravity="end"
        android:textAlignment="textEnd"
        android:layout_columnWeight="3"
        />

    <RelativeLayout
        android:layout_row="3"
        android:layout_column="0"
        android:layout_columnSpan="3"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent">

        <com.github.mikephil.charting.charts.LineChart
            android:id="@+id/lc_glasrechner_refra_out"
            android:layout_width="fill_parent"
            android:layout_height="250dp"
            ></com.github.mikephil.charting.charts.LineChart>
    </RelativeLayout>



    <TextView
        android:text="Astigmatismus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="25sp"
        android:layout_row="4"
        android:layout_column="0"
        android:paddingTop="30dp"
        android:layout_columnSpan="3"
        />

    <TextView
        android:text="@string/txt_glasrechner_HS1_table"
        android:id="@+id/txt_glasrechner_hs1_asti"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="5"
        android:layout_column="1"
        android:paddingBottom="2dp"
        android:layout_gravity="end"
        android:gravity="start"
        android:textAlignment="textStart"
        />

    <TextView
        android:text="@string/txt_glasrechner_HS2_table"
        android:id="@+id/txt_glasrechner_hs2_asti"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_row="5"
        android:layout_column="2"
        android:paddingBottom="2dp"
        android:layout_gravity="end"
        android:gravity="start"
        android:textAlignment="textStart"

        />

    <TextView
        android:text="Grad"
        android:id="@+id/txt_glasrechner_grad_asti_out"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="17sp"
        android:layout_row="6"
        android:layout_column="0"
        android:gravity="end"
        android:textAlignment="textEnd"
        />
```

```
<TextView
    android:text="Asti HS 1"
    android:id="@+id/txt_glasrechner_asti_hs1_out"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="17sp"
    android:layout_row="6"
    android:layout_column="1"
    android:layout_gravity="end"
    android:gravity="end"
    android:textAlignment="textEnd"
    android:layout_columnWeight="2"
    />

<TextView
    android:text="Asti HS 2"
    android:id="@+id/txt_glasrechner_asti_hs2_out"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="17sp"
    android:layout_row="6"
    android:layout_column="2"
    android:layout_gravity="end"
    android:gravity="end"
    android:textAlignment="textEnd"
    android:layout_columnWeight="3"
    />

<RelativeLayout
    android:layout_row="7"
    android:layout_column="0"
    android:layout_columnSpan="3"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent">

    <com.github.mikephil.charting.charts.LineChart
        android:id="@+id/lc_glasrechner_asti_out"
        android:layout_width="fill_parent"
        android:layout_height="250dp"
        ></com.github.mikephil.charting.charts.LineChart>
</RelativeLayout>

<TextView
    android:text="Verzeichnung"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:layout_row="8"
    android:layout_column="0"
    android:paddingTop="30dp"
    android:layout_columnSpan="3"
    />

<TextView
    android:text="@string/txt_glasrechner_HS1_table"
    android:id="@+id/txt_glasrechner_hs1_vz"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="9"
    android:layout_column="1"
    android:paddingBottom="2dp"
    android:layout_gravity="end"
    android:gravity="start"
    android:textAlignment="textStart"
    />

<TextView
    android:text="@string/txt_glasrechner_HS2_table"
    android:id="@+id/txt_glasrechner_hs2_vz"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_row="9"
    android:layout_column="2"
    android:paddingBottom="2dp"
    android:layout_gravity="end"
    android:gravity="start"
    android:textAlignment="textStart"
    />

<TextView
    android:text="Grad"
    android:id="@+id/txt_glasrechner_grad_vz_out"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="17sp"
```

```
                    android:layout_row="10"
                    android:layout_column="0"
                    android:gravity="end"
                    android:textAlignment="textEnd"
                    />

                <TextView
                    android:text="VZ HS 1"
                    android:id="@+id/txt_glasrechner_vz_hs1_out"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textSize="17sp"
                    android:layout_row="10"
                    android:layout_column="1"
                    android:layout_gravity="end"
                    android:gravity="end"
                    android:textAlignment="textEnd"
                    android:layout_columnWeight="2"
                    />

                <TextView
                    android:text="VZ HS 2"
                    android:id="@+id/txt_glasrechner_vz_hs2_out"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textSize="17sp"
                    android:layout_row="10"
                    android:layout_column="2"
                    android:layout_gravity="end"
                    android:gravity="end"
                    android:textAlignment="textEnd"
                    android:layout_columnWeight="3"
                    />

                <RelativeLayout
                    android:layout_row="11"
                    android:layout_column="0"
                    android:layout_columnSpan="3"
                    android:layout_height="fill_parent"
                    android:layout_width="fill_parent">

                    <com.github.mikephil.charting.charts.LineChart
                        android:id="@+id/lc_glasrechner_vz_out"
                        android:layout_width="fill_parent"
                        android:layout_height="250dp"
                        ></com.github.mikephil.charting.charts.LineChart>
                </RelativeLayout>

            </GridLayout>


        </GridLayout>

    </GridLayout>

</ScrollView>

</RelativeLayout>
```

bb) Layout – info.xml

```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Space
        android:layout_width="match_parent"
        android:layout_height="10dp"
        android:visibility="visible" />

    <ImageView
        android:id="@+id/banner_hs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitXY"
        app:srcCompat="@drawable/logo"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        />

    <Space
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:visibility="visible" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:text="Herausgeber"
        android:textSize="22sp"
        android:textStyle="bold" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="10dp"
        android:visibility="visible" />

    <TextView
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:id="@+id/impressum_inhalt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Impressum"
        android:textSize="12sp" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:visibility="visible" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:text="Datenschutzbestimmungen"
        android:textSize="22sp"
        android:textStyle="bold" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="10dp"
        android:visibility="visible" />

    <TextView
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:id="@+id/datenschutz_inhalt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:text="Datenschutz"
        android:textSize="12sp" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:visibility="visible" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:text="Haftungshinweis"
        android:textSize="22sp"
        android:textStyle="bold" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="10dp"
        android:visibility="visible" />

    <TextView
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:id="@+id/haftung_inhalt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Haftung"
        android:textSize="12sp" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:visibility="visible" />

    <TextView
        android:id="@+id/textView10"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:text="Open Source Lizenzen"
        android:textSize="22sp"
        android:textStyle="bold" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="10dp"
        android:visibility="visible" />

    <TextView
        android:id="@+id/textView12"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:text="MPAndroidChart"
        android:textSize="16sp"
        android:textStyle="bold" />

    <TextView
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:id="@+id/opensource_inhalt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OpenSource"
        android:textSize="12sp" />

</LinearLayout>

</ScrollView>
```