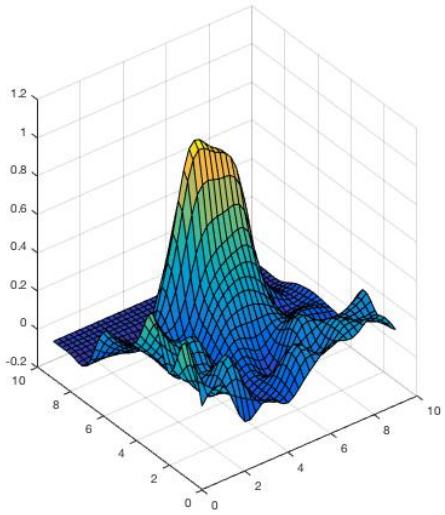
Schweizer Jugend forscht

Wettbewerbsarbeit 2016



Akustische Linsen

Beugungsmuster im Raum

Autoren:

Lewis Beauchamp

Waltersgrabenweg 8
4125 Riehen

lewis.beachamp@edubs.ch

Stanislaw Zytynski

Schäublinstrasse 101
4059 Basel

stanislaw.zytynska@edubs.ch

Gymnasium Kirschgarten Basel

Betreut von Reinhard Weiss, Physiklehrperson am GKG Basel &

Hermann Klein, Schülerforschungszentrum Lörrach (D)

Basel, 23. Oktober 2016

Inhaltsverzeichnis

1 Vorwort

Letzten Herbst haben wir beim Internationalen Wettbewerb METAKSI am PHAENOVUM in Lörrach (D) teilgenommen. Durch diesen Wettbewerb wurden wir auf das Schülerforschungs-Zentrum aufmerksam. Bei einer Exkursion mit unserem Physiklehrer zum PHAENOVUM sind wir auf die Idee gekommen eine eigene Arbeit zu schreiben. Als Thema der Arbeit haben wir eine Aufgabenstellung von IYPT, Problems for the 30th „IYPT 2016“, ausgewählt. Diese offenen Probleme eignen sich für eine *Schweizer Jugend forscht* Arbeit. Das Thema „ACOUSTIC LENS“ sprach uns besonders an. Es lautete:

> *Fresnel lenses with concentric rings are widely used in optical applications, however a similar principle can be used to focus acoustic waves. Design and produce an acoustic lens and investigate its properties, such as amplification, as a function of relevant parameters.*

Da wir im Physikunterricht gerade Beugung am Gitter besprochen hatten, haben wir uns schnell für dieses Thema entschieden. Das Thema erwies sich bald komplexer als vorerst erwartet. Mit der Zeit hat uns die Arbeit mehr und mehr gefordert und wir stiessen an die Grenze unseres physikalischen Grundwissens und mussten uns mehr als Anfangs erwartet, dafür einsetzen. Wir wollen uns vor allem beim Schülerforschungszentrum PHAENOVUM und Hermann Klein bedanken. Sie stellten uns die Räumlichkeiten und das Material für unsere Arbeit zur Verfügung. Auch unser Physiklehrer Reinhard Weiss hat uns immer wieder motiviert, Tipps gegeben und stand z.B. bei der Auswertung der Messergebnisse zur Seite. Insbesondere half er uns auch die Arbeit in LaTeX aufzusetzen und hatte am Text oder an den Graphiken immer etwas auszusetzen. Auch ein Dank an unsere Klassenkameradin Aashi Kalra. Sie half uns bei der Erstellung der Graphiken mit MatLab.

2 Zusammenfassung

In der Arbeit befassen wir uns mit der Ausbreitung von Schallwellen im Raum. Da Schallwellen sich an Objekten im Raum beugen lassen, kann man sie mit einer akustischen Zonenplatte fokussieren. Zonenplatten bestehen aus konzentrischen Ringen, die zwischen schalldurchlässig und -undurchlässig Zonen abwechseln. Diese erkannte bereits Augustin Jean Fresnel.

Wir wollen untersuchen, inwiefern sich Schallwellen mit einer Zonenplatte fokussieren lassen. Dabei wollten wir das Beugungsmuster im Raum erfassen. Im Zentrum der Arbeit stand deshalb unser selbst gebauter akustischer 3D-Scanner und dessen Programmierung. Der 3D Scanner wird durch ein Arduino-Board gesteuert.

Die erhaltenen Daten haben wir mit MatLab ausgewertet. Die Intensitätsverteilung im Raum lieferte uns faszinierende Bilder und Erkenntnisse über die Funktionsweise von akustischen Zonenplatten.

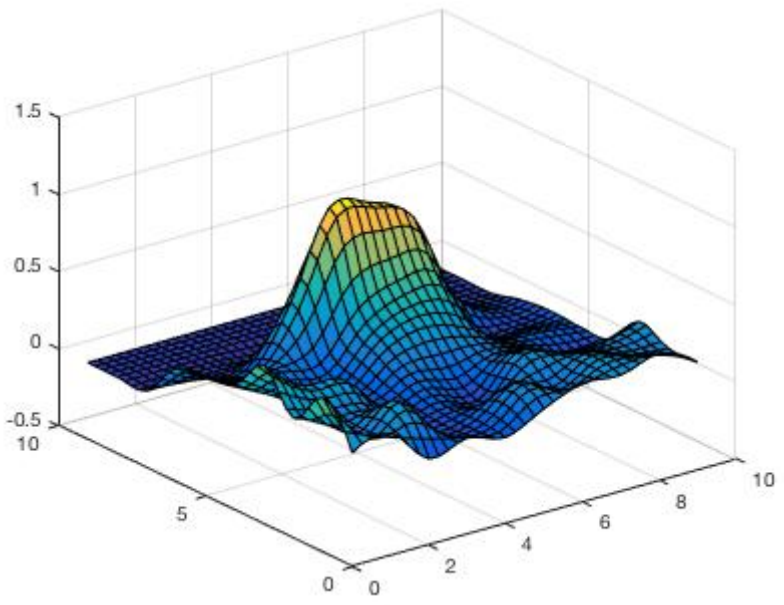


Abb. 1: Intensitätsverteilung des Schalls ohne Linse

3 Einleitung

Beugung ist ein schon lang verstandenes Phänomen. Bereits der niederländische Physiker Christiaan Huygens (* 14. April 1629; † 8. Juli 1695) und der englische Physiker Augustin Jean Fresnel (* 10. Mai 1788; † 14. Juli 1827) erkannten, dass wenn man einer Welle ein Hindernis in den Weg stellt, sich neue Elementarwellen bilden. Diese Elementarwellen können sich gegenseitig überlagern. Es kommt zu Interferenzerscheinungen[1].

Die Zonenplatte wurde von Augustin Jean Fresnel erfunden. Mit ihr kann man mithilfe von Interferenz einer Frequenz die Schallwellen hinter der Linse fokussieren. Die Formel welche Fresnel aufstellte lautete[Wik16]:

$$r_n = \sqrt{n \cdot \lambda \cdot f + \frac{n^2 \cdot \lambda^2}{4}}. \tag{1}$$

Zunächst haben wir die Intensitätsverteilung von Schall an einem Spalt und Doppelspalt mit dem Mikrofon eines Handys gemessen und betrachtet. Wir konnten zwar keine grossen Schwankungen feststellen, aber ein Beugungsmuster war durchaus erkennbar. Deshalb überlegten wir uns folgende Fragestellung:

1. Können wir mithilfe von Fresnels Formel eine Zonenplatte herstellen und Schall beugen?

2. Ist es möglich mithilfe der Zonenplatte den Schall zu fokussieren und/oder zu verstärken?

Damit wir weitere Fragen beantworten konnten, mussten wir zuerst die Formel (1) verstehen und unsere erste Linse bauen. Die von Fresnel aufgestellte Formel beruht auf der Charakteristik von Wellen, sich an Hindernissen zu beugen. Wir haben seine Formel für den Bau einer Zonenplatte übernommen, in Form eines kurzen Programmes, welches die Schnittstellen zwischen Spalten bestimmte[a]. Dabei konnten wir die Zonenplatten anhand vom Brennpunkt und der Frequenz der Wellen eindeutig festlegen.

Abb. 2: Zonenplatte

[a] siehe Anhang, Quellcode 1

[1] Physik für Mittelschulen, S. 327[KM14]

4 Vorbereitungen für das Experiment

Um die Zonenplatte herzustellen, haben wir ein kleines Programm geschrieben, welches uns die richtigen Werte für die Zonenabstände lieferte. Mit diesem Algorithmus konnten wir die Graphen in Abb. 3 berechnen. Hier wird der Zusammenhang zwischen Brennweite und Radius der Zonenringe bei einer konstanten Frequenz von 15 kHz dargestellt. Mithilfe dieses Programms und den gewonnenen Daten konnten wir eine optimale Grösse für unsere Linse bestimmen.

Dabei sollte die ganze Apparatur transportierbar sein und die Brennweite unter 2 Meter liegen. Dies war notwendig, damit wir die Messwerte in unserer schallisolierten Box überhaupt messen konnten. Mittels dessen kamen wir zum Schluss, dass ein höherer Frequenzbereich (um 15 kHz) optimal wäre. In Abb. 3 erkennt man, dass die Zonenabstände nicht den gleichen Abstand haben, d.h. dass die Ringdicken nach aussen hin immer kleiner werden.

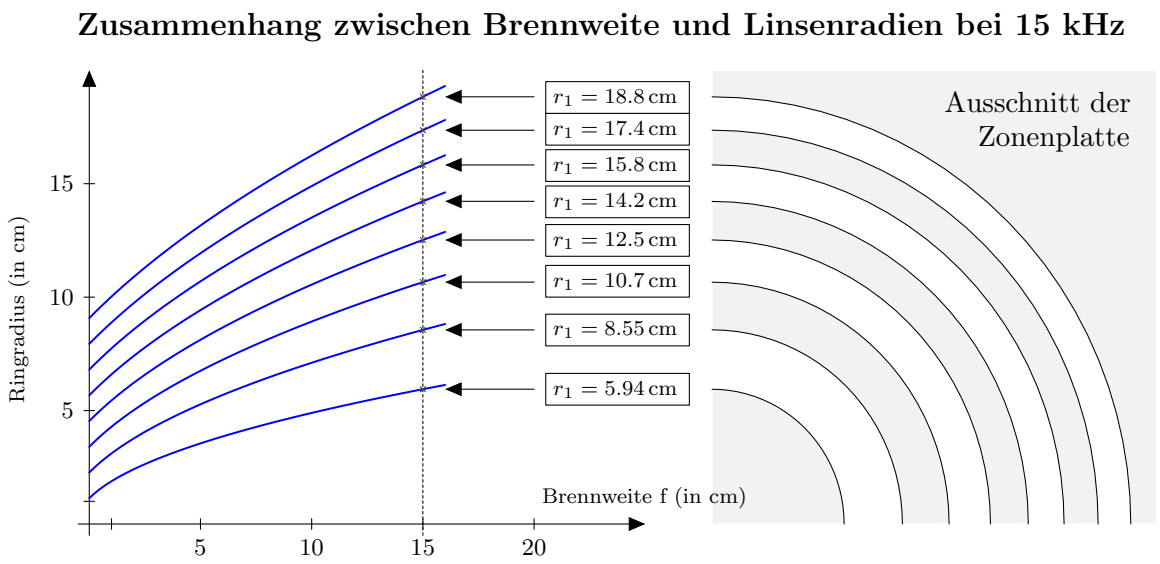


Abb. 3: Berechnung der Radien für die einzelnen Zonen

Die erste Zonenplatte haben wir aus einer Kunststoffplatte gefertigt, später aus Karton. Karton lässt sich einfacher bearbeiten und dadurch konnten wir die Linse genauer herstellen.

Abb. 4: Fresnellinse, Eigenbau

4.1 Die Linse

Wir haben mit der Kunststofflinse unsere ersten Messungen erfasst. Dabei haben wir ein Mikrofon hinter der Zonenplatte positioniert in der Brennebene hin und her bewegt (Abb. 4). Mithilfe einer bereits existierenden Software (PASCO Capstone) konnten wir die Amplitude auf dem PC ablesen.

In Abb. 4 sieht man unsere erste Zonenplatte. Im Zentrum beginnt Sie mit einer ausgefüllten Zone. Es wäre auch denkbar, dass Sie im Zentrum mit einem Loch beginnen würde.

Für weitere Experimente haben wir deshalb bei einer zweiten Zonenplatte die Zonenbereiche umgedreht und dadurch die „invertierte Zonenplatte“ erhalten.

4.2 Versuchsaufbau

Der erste Aufbau war sehr primitiv. Wir konnten noch keine aussagekräftigen Messungen erzielen. Eine Analyse der Daten hat jedoch darauf schliessen lassen, dass eine Steigerung der Amplitude relativ zu der nahen Umgebung stattgefunden hat, dies im Vergleich zu den Messungen ohne Zonenplatte. Als wir die Messung ohne Zonenplatte wiederholten ist uns jedoch aufgefallen, dass die Amplitude ohne Platte höher war als mit Zonenplatte, dies auch im Brennpunkt selber.

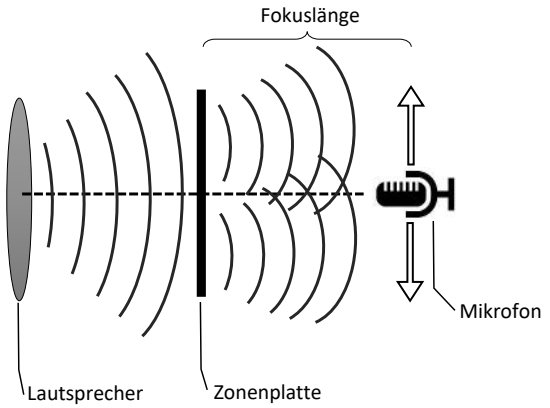


Abb. 5: Darstellung des Versuchsaufbaus

Das war jedoch auch zu erwarten, da etwa die Hälfte des Schalls von der Oberfläche der Zonenplatte reflektiert wird. Das Ziel einer solchen Zonenplatte ist also folglich nicht zwingend die Fokussierung von Schall, sondern eher die Isolierung davon. Da diese Messungen als Vorarbeit dienen, haben wir danach mit unserem eigentlichen Versuchsaufbau begonnen. Dieser musste die folgenden Kriterien erfüllen:

- Die Messdaten müssen viel genauer erfasst werden können.
- Die Messwerte müssen automatisch aufgenommen und dargestellt werden.

- Der äussere Lärm muss abgeschirmt werden.

Der akustischer 3D-Scanner

Wir entwickeln ein System, welches einen Raum abfährt und aufzeichnet, indem wir ein Mikrofon an vier Schnüren aufgespannt haben. Diese werden in einer schallisolierten Kiste untergebracht, mit vier Motoren verbunden und können mit einer Spule aufgerollt werden. Die Abbildung zeigt eine raytracing Rendering von unserer ursprünglichen Vision für das System. Im rechten Bild (Abb. 7) sehen Sie den fertigen Scanner.

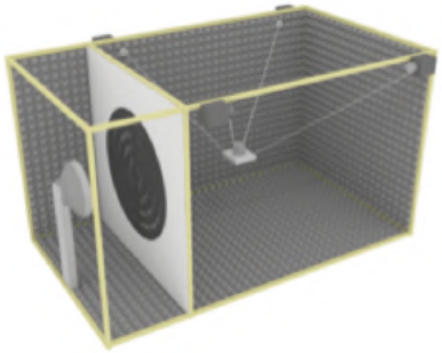


Abb. 6: Modell unseres 3D-Scanners

Abb. 7: Der 3D-Scanner

Der Ton für diesen Aufbau wurde von einem Funktionsgenerator mit eingebautem Lautsprecher erzeugt. Um die Motoren anzusteuern haben wir ein Arduino Mega 2560 benutzt und mit einem A4988 Stepper Treiber gekoppelt. Wir mussten die Platine eigens herstellen, um alle Stepper Treiber zusammenzuschliessen.

Der Arduino wurde mit den standard Libraries programmiert. Dabei haben wir nur diejenigen benutzt, die mit dem IDE zusammen kamen. Der Arduino nimmt die gewünschte Länge der Schnüre über eine USB Schnittstelle vom PC auf und bewegt die Schrittmotoren dementsprechend in die richtige Position. Da kein System vorhanden ist, um die momentane Lage zu bestimmen, müssen die Motoren beim starten immer neu kalibriert werden.

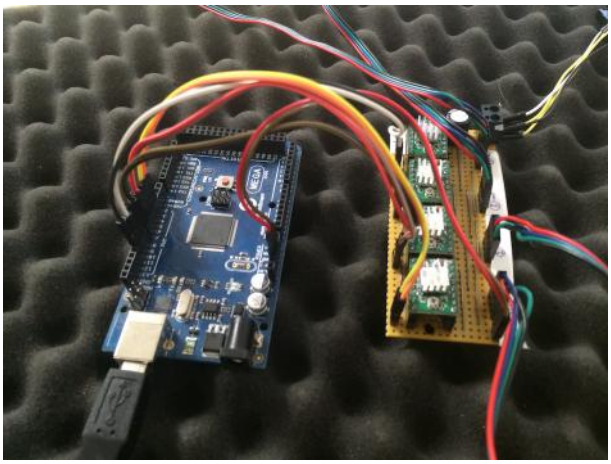


Abb. 8: Platine mit Bord

4.3 Aufnahme und FFT

Vorerst wurde ein Mikrofon bestellt, welches die Amplitude über das ADC vom Arduino hätte einlesen sollen. Weil wir jedoch das falsche Mikrofon bestellten, das über keinen Stromspannungskonvertierer verfügte, um den logischen Pegel vom Mikrofon auf den des Mikrokontrollers zu bringen, mussten wir diesen durch ein anderes kurzfristig ersetzen. Wir haben ein USB Mikrofon ausgewählt, das direkt an den PC gekoppelt werden kann.

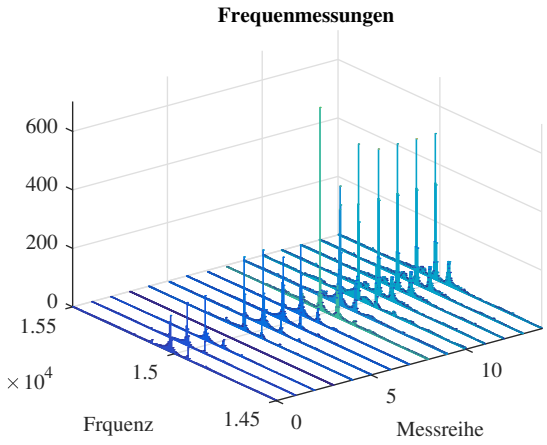


Abb. 9: Messreihen für die Optimierung der Sample-Rate

Um die Amplitude vom Schall genau zu bestimmen und Hintergrundgeräusche zu vermindern, haben wir einen FFT-Algorithmus angewendet. Dieser konnte die Daten, die kontinuierlich vom Mikrofon eintrafen so verarbeiten, dass wir einzelne Frequenzen isolieren konnten. Die Daten wurden mit den Audio Libraries von Java JDK erfasst und trafen im Form von einem 44.1kHz stream von Amplituden-Daten ein. Wir haben immer ein Zeitfenster von etwa 256 Samples gewählt. Bald haben wir jedoch gemerkt, dass der Peak um 15kHz nicht genau auf ein Bin gefallen ist. Dies lag dem zugrunde, dass die Frequenzverteilung vom Schall kürzer war, als die Auflösung vom FFT Algorithmus. Um dieses Problem zu lösen haben wir die sample rate vergrössert, damit die Peaks besser getroffen wurden. Ebenfalls wurde der Peak gesucht, d.h. es wurde im Bereich um 15kHz gesucht, für den Fall, dass er nicht genau bei der gewünschten

Frequenz liegen sollte. Diese Veränderungen sind in der Graphik in Abb. 9 ersichtlich, wobei die gefundenen Peaks eine geringere und höhere Sample-Rate dargestellten. Dadurch konnte der Peak von einer Genauigkeit von etwa 20dB auf etwa 3dB minimiert werden.

4.4 Die Software

Einen wesentlichen Teil der Arbeit bildete das Schreiben der PC Software. Es besteht aus zwei Teilen, dem Teil, der auf dem PC läuft und dem Teil, der auf dem Arduino ausgeführt wird. Der erste Teil ist in Java geschrieben besteht aus etwa 2500 Zeilen. Der Arduino Code wurde mit dem Arduino IDE in C erstellt. Das PC Programm umfasst die folgende und Hauptfunktionen:

1. Berechnung der Schnurlänge und Position des Mikrofons.
2. Berechnung des Pfades zum Abscannen eines Bereiches.
3. Kommunikation mit dem Arduino und Organisierung und Aneinanderreihung der Kommandos.
4. Mikrofon Aufnahme über USB und Analyse der FFT Library.
5. Datenspeicherung und Visualisierung von Schnittflächen.

Der FFT Algorithmus funktioniert zusammen mit der Schalldichtung, um die Hintergrundgeräusche zu minimieren. Es wurden zwei Versionen von der Software erstellt. Die erste Version hatte ein GUI, damit man die Daten auch direkt anzeigen konnte und die Position visuell veranschaulichen konnte. Diese Version wurde jedoch bald ersetzt, da die Swing Elemente absolut positioniert wurden und sich das GUI nicht auf hochauflösenden Displays darstellen liess. Dazu hatte die Struktur vom Code ein paar grundlegende Fehler in der Objektorientierten Struktur. Beispielsweise wurde kein Observer Pattern, sondern nur statische Variablen benutzt, um Action Events aufzufassen. Deshalb wurde eine zweite Version grösstenteils von Grund auf neu geschrieben, welcher mit einem CLI funktioniert. Dies ist weniger intuitiv, war jedoch schneller umsetzbar. Die Funktionen des Programms sind wie folgt umgesetzt: Die Schnurlängen werden mit dem Satz des Pythagoras berechnet und jeweils gespeichert. Das Arduino nimmt also lediglich die korrekte Länge entgegen. Um einen Bereich abzuscannen kann man einen Quader um den Raum mit Punkten bestimmen. Danach kann man auch die gewünschte Auflösung festlegen, je nachdem, wie viele Datenpunkte aufgenommen werden sollten. Das Programm berechnet danach einen Pfad, nach dem der Bereich abgescannt wird. Die generierten Kommandos werden

aneinandergereiht und nacheinander per USB an das Arduino geschickt. Dabei wird immer zuerst eine Position angegeben und gewartet, bis diese eingenommen wird. Danach wird über das Mikrofon die Amplitude aufgenommen und der Prozess geht weiter, bis alle Punkte gescannt sind. Die Amplitude wird danach ermittelt und mit der Position gespeichert. Sobald das Programm fertig ist, können die Daten in einer Liste ausgeschrieben werden. Dazu kann eine einfache Visualisierung gemacht werden, indem Schnittflächen durch den gescannten Bereich angezeigt werden.

4.5 Arduino Software

Der Arduino Code hat die folgende Funktion:

- Bewegung der Motoren an die angegebene Position

Dafür wartet das Bord auf die neue Position, die über ein USB übermittelt wird. Diese trifft immer in Form der gewünschten Schnurlängen ein. Ein Problem, das dabei aufgekommen ist, war die Bewegung der Seilrollen in der korrekten Reihenfolge. Dafür haben wir einen Algorithmus, welcher zuerst die Schnüre verlängert und danach einzieht, geschrieben. Um die momentane Schnurlänge immer zu wissen, merkt sich das Arduino wie weit die Schnüre verlängert sind. Bei dem Startup müssen sie jeweils die maximale Länge haben. Da die Schrittmotoren generell wenige Fehler machen, kann man davon ausgehen, dass sie ihre Position beibehalten.

5 Resultate

Wir haben für die Messungen verschiedene Anordnungen und Zonenplatten ausprobiert. Dabei haben wir immer Vertikale Schnittflächen mit der normalen Zonenplatte gemessen. Wir haben für die drei Konfigurationen: „keine Zonenplatte“, „normale Zonenplatte“ und „invertierte Zonenplatte“ jeweils eine Messung bei normaler und doppelter Brennweite gemacht. Der Lautsprecher wurde immer mit derselben Abstand vor der Zonenplatte aufgestellt. Die Graphik zeigt die Konfigurationen:

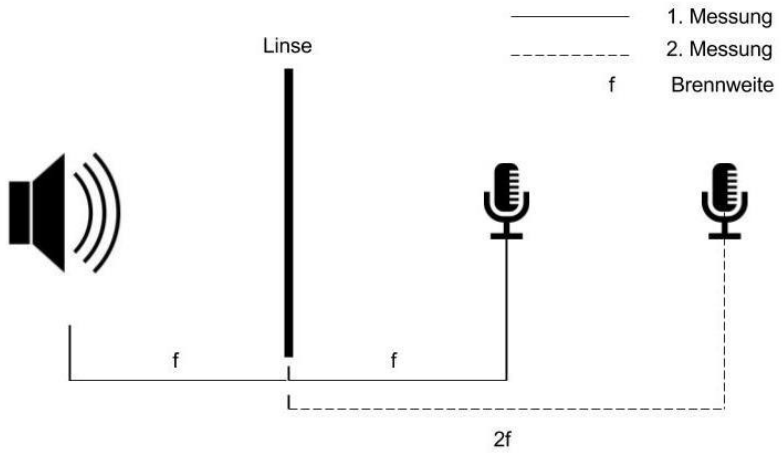


Abb. 10: Versuchaufbau

Wir erhielten brauchbare Messergebnisse, die wir in Abb. 11 und 12 mithilfe von MatLab dargestellt haben. Die Zwischenwerte wurden dabei interpoliert.

Betrachtet man in Abb. 11 die Graphik ohne Zonenplatte, erkennt man, wie die Amplitude in der Mitte der Schnittfläche am höchsten ist. Dies führen wir auf die Positionierung des Lautsprechers zurück, der sehr nahe und ebenfalls in der Mitte der Schnittfläche positioniert war. Hellgelb steht für starke Intensität und Dunkelblau für eine schwache Intensität. Diese Grafik nehmen wir als Normalzustand und betrachten die folgenden Ergebnisse im Zusammenhang mit dieser Messung.

In Abb. 11 stellt die oberste Grafik dieselbe Schnittfläche dar, jedoch mit akustischer Zonenplatte (wie gezeigt in Abb. 4). Dabei erkennt man viele kleine Maxima und eine unregelmässige Verteilung der Amplitude Maxima und Minima. Man sieht wie die Schallwellen der Zonenplatte gestreut werden. Auf der untersten Grafik sehen wir was passiert, wenn man eine invertierte Zonenplatte benutzt. Man kann klar erkennen wie sich die Intensität fokussiert.

In der Abb. 12 sind die Beugungsmuster in der Ebene mit doppelter Brennweite abgebildet. Hier erkennt man ein ähnliches Muster wie bei der Abb. 11. Die Grafik ohne Zonenplatte

hat eine relativ hohe Amplitude in der Mitte, die Richtung aussen mit einzelnen Dellen im Beugungsmuster stetig abnimmt.

Bei der Grafik mit normaler Zonenplatte sieht man wie sich das Beugungsmuster ähnlich wie bei Abb. 11 verhält: sehr rapide Änderungen von Maxima zu Minima auf kleiner Fläche. Jedoch gibt es hier auch einen zentralen Peak in der Amplitude, der sich deutlich vom restlichen Beugungsmuster abhebt. Betrachtet man nun die invertierte Zonenplatte so sieht man, im Vergleich zur Grafik ohne Zonenplatte, dass die Amplitude fokussiert und verstärkt wurde. Ebenfalls erkennt man, dass das Beugungsmuster im Vergleich zum Beugungsmuster der normalen Zonenplatte sanfter wurde.

5.1 Beugungsmuster in der Brennebene

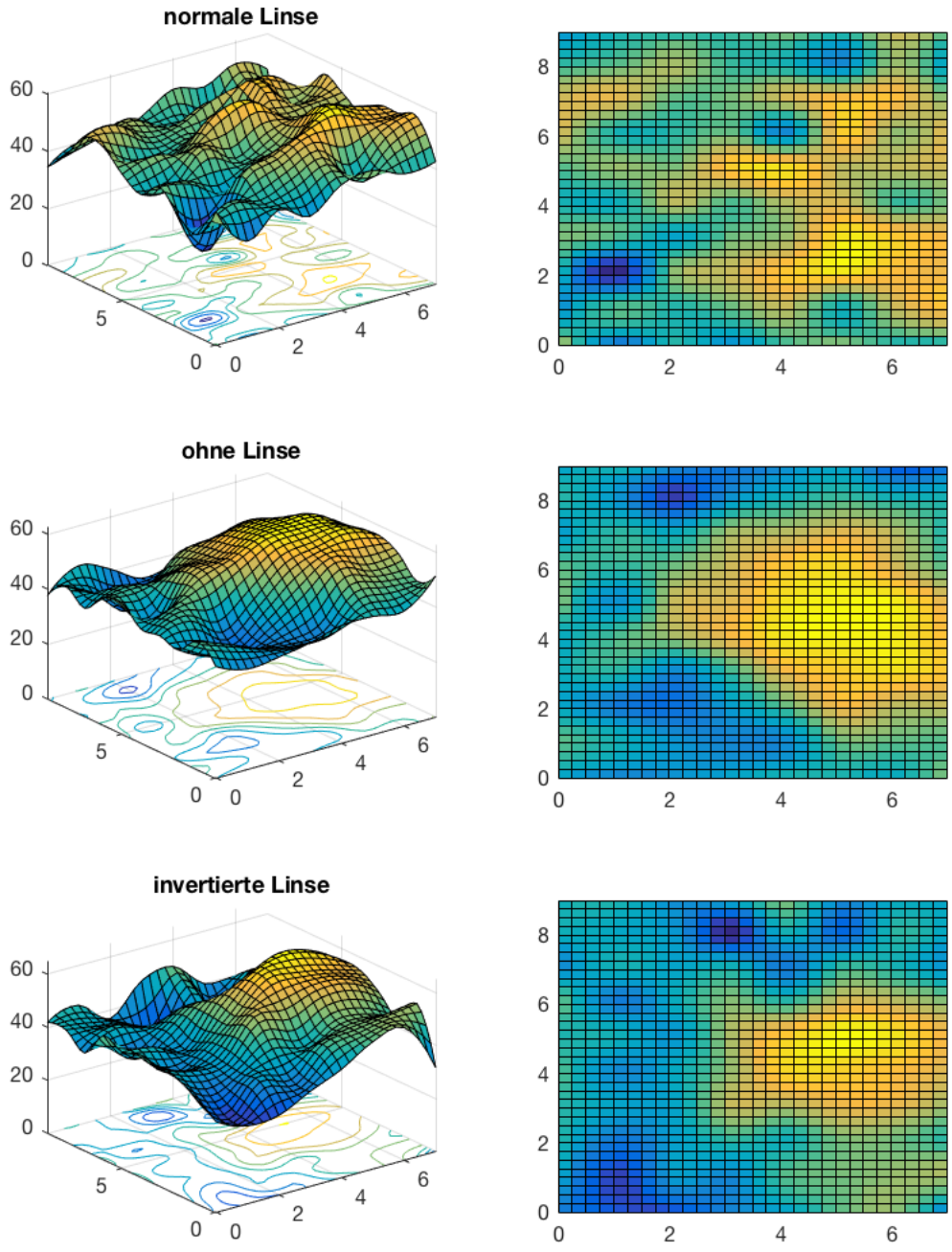


Abb. 11: Interferenzbilder beim Linsenabstand 16 cm, $f = 16\,\mathrm{kHz}$

5.2 Beugungsmuster in der Ebene bei doppelter Brennweite

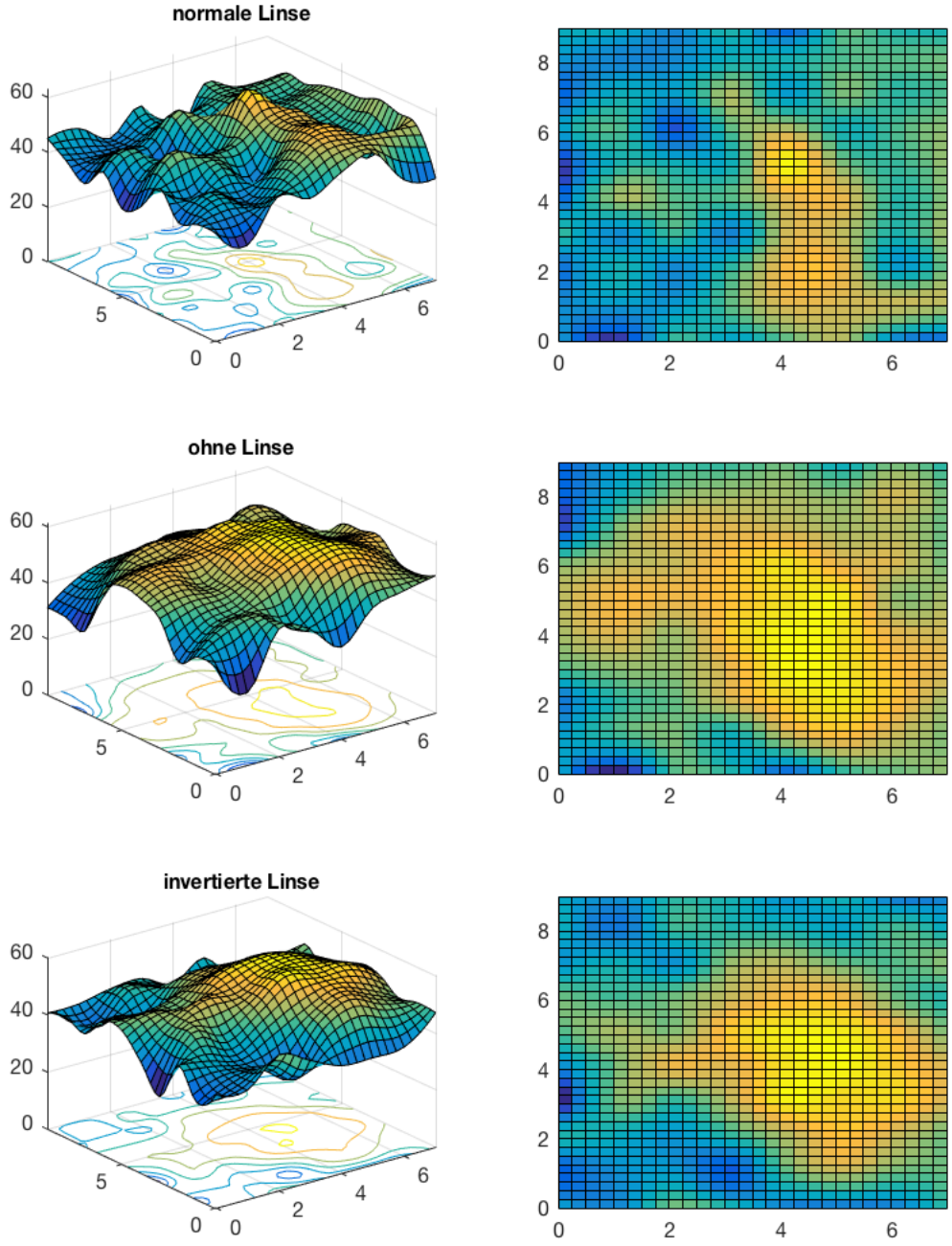


Abb. 12: Interferenzbilder beim Linsenabstand 32 cm, $f = 16\,\mathrm{kHz}$

6 Diskussion

Die Messergebnisse haben unsere Vermutungen teilweise bestätigt. Die Intensität des Schalles ist ohne Zonenplatte immer noch ein wenig höher. Dies ist darauf zurückzuführen, dass ein wesentlicher Teil des Schalls von der Zonenplatte reflektiert wird, bevor er an das Mikrofon gelangen kann. Trotzdem kann man jedoch sagen, dass bei der Messung mit invertierter Zonenplatte eine Bündelung der Schallwellen stattgefunden hat, gegenüber der Messung ohne Zonenplatte. Dies ist darauf zurückzuführen, dass sich die Wellen im Raum überlagern. Eine weitere Erkenntnis ist, dass sich die Wellen bei der ersten Zonenplatte nicht offensichtlich bündeln. Dies ist ein interessanter Punkt, den wir in der Zukunft genauer untersuchen möchten.

Anhand der Formel von Fresnel war es einfach zu glauben, dass bei einer invertierten Zonenplatte, bei der jedoch die Grenzflächen identisch sind, dieselben Resultate gemessen werden. Bei der normalen Zonenplatte sind Muster erkennbar, die wir als Interferenzmuster interpretieren. Bei der invertierten Zonenplatte zeigen sich die erwarteten Resultate. Eine andere wichtige Erkenntnis ist, dass sich der Schall gegen hinten, also bei doppelter Brennweite, nicht mehr so stark bündelt. Dies sieht man bei der invertierten Zonenplatte genau wie beim Versuch ohne Zonenplatte.

Vergleicht man die beiden Messungen mit Zonenplatte bei Abb. 12 (also Normal und Invertiert), im Bezug auf die Maxima und Minima der Wellen im Beugungsmuster, so sieht man, dass bei der Grafik der normalen Zonenplatte deutlich mehr Chaos und Unterschiede zwischen Maxima und Minima auf engem Raum vorhanden sind. Bei der Invertierten Zonenplatte gibt es hingegen eine deutlich abgeflachte Form. Dies deuten wir so, dass hier der innere Zonenring bei der Invertierten Zonenplatte schalldurchlässig ist. Dies führt dazu, dass ein grosser Teil der Wellen nicht direkt reflektiert wird. Wohingegen bei der normalen Zonenplatte dieser Teil reflektiert und Schallwellen nur durch Brechung die Zonenplatte überwinden können. Was dazu führt, dass viele Maxi- und Minima entstehen welche es bei der invertierten Zonenplatte auch gäbe, die aber durch die starke Amplitude, die durch den Mittelring zustande kommt, überdeckt wird. Die effektive Bündelung geschieht mit Hilfe der restlichen Zonenringen.

6.1 Zukünftige Pläne für Messungen

Unser Lehrer hat uns darauf hin gewiesen, dass es eigentlich keinen Sinn macht, den Lautsprecher in den Brennpunkt zu stellen, wenn wir den Schall fokussieren möchten. Deshalb wollen wir als nächstes einen Versuch machen, bei dem wir den Lautsprecher weiter hinten positionieren. Dadurch sollten die Schallwellen parallel auf die Zonenplatte auftreffen, wodurch sich

vielleicht eine bessere oder andere Bündelung des Schalls ergibt. Weiterhin möchten wir unser Programm so verbessern, dass wir einen grösseren Raum einfach nach seinem Akustischen Muster durchscannen können. Hinzu kommt, dass wir unsere Kiste verbessern wollen, damit wir genauere Messungen an Zonenplatten und anderen Gegenständen durchführen können, um deren Beugungsmuster zu vermessen. Aus den momentanen Schnittflächen lassen sich interessante Erkenntnisse schliessen, die Schnittflächen sind jedoch immer zweidimensional. Dadurch, dass wir Messungen im kompletten Raum, d.h dreidimensionale Messungen machen könnten, würden wir auch eine bessere Vorstellung vom Verlauf des Schalls bekommen. Wir möchten auch das Problem lösen, dass ca. 50 Prozent des Schalls reflektiert wird. In der Zukunft wollen wir Methoden erforschen, um grössere Proportionen umzulenken. Dadurch könnte man eventuell nicht nur eine Bündelung, sondern auch eine tatsächliche Fokussierung von Schallwellen erzielen, deren Maximum grösser ist als ohne Linse.

6.2 Fazit

Es ist möglich mit Hilfe der Fresnel Formel Akustische Wellen zu beugen. Wir können eine Fresnel Zonenplatte mit Hilfe der Fresnel Formel und einem einfachen Programm berechnen und herstellen.

Unsere Messungen können nicht bestätigen, dass man mit einer Zonenplatte eine höhere Amplitude messen kann. Jedoch erkennt man eine Fokussierung mit Zonenplatte und ebenfalls eine stärkere Amplitude im Vergleich zur unmittelbaren Umgebung.

7 Abkürzungsverzeichnis und Glossar

USB Universal Serial Bus

RXTX Receive/Transceive used for USB communication

IDE Integrated Development Environment

FFT Fast Fourier Transform

IYPT International Young Physicists' Tournament

GUI Graphical User Interface

CLI Command Line Interface

ADC Analog to Digital Converter

8 Quellenangaben und Verzeichnisse

Literatur

[KM14] Hans Kammer and Irma Mgeladze. *Physik für Mittelschulen*, volume 1. Auflage. hep-Verlag, 2014.

[MSGD14] M. Moleryn, M. Serra-Garcia, and C. Daraio. Acoustic fresnel lenses with extraordinary transmission. *Appl. Phys. Lett. 105, 114109*, 2014.

[Wik16] Wikipedia. *Fresnel-Zonenplatte*. https://en.wikipedia.org/wiki/Zone_plate, Eingesehen am 01.07.2016.

Abbildungsverzeichnis

Libraries

Für das Programm wurden die folgenden Java Libraries benutzt:

RXTX http://rxtx.qbang.org/wiki/index.php/Main_Page

JTransforms https://sites.google.com/site/piotrwendykier/software/jtransforms

9 Anhang

9.1 Quellcode

9.1.1 Programmcode für das Arduino

```
String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete

const int delayTime = 500;

const int step1 = 3;
const int motor1 = 4;
const int step2 = 5;
const int motor2 = 6;
const int step3 = 7;
const int motor3 = 8;
const int step4 = 9;
const int motor4 = 10;

const int micInPin = A1;

const int microstepping = 16;
const int fullRotationMM = 147;

int motor1Pos = 1690;
int motor2Pos = 2270;
int motor3Pos = 1670;  //+100
int motor4Pos = 2280;  //-30

void setup() {
  Serial.begin(115200);
  inputString.reserve(200);


  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);

  pinMode(A0, OUTPUT);
  digitalWrite(A0, HIGH);
}

void loop() {
if (stringComplete) {
    String command = inputString;
    String message = inputString;
    command.remove(2);

    if(command == "G1"){
      //Has the format G1_BXXXXX_XXXXX_XXXXX_XXXXX;
      //Eg: G1_00100_00100_00100_00100_1111;

      int  s1 = message.substring(3,8).toInt();
      int  s2 = message.substring(9,14).toInt();
```

```
55        int  s3 = message.substring(15,20).toInt();
          int  s4 = message.substring(21,26).toInt();
57
          /*char b1 = message.charAt(27);
59        char b2 = message.charAt(28);
          char b3 = message.charAt(29);
61        char b4 = message.charAt(30);*/

63        /*if(b1 == '0'){
            s1 = s1*-1;
65        }
          if(b2 == '0'){
67          s2 = s2*-1;
          }
69        if(b3 == '0'){
            s3 = s3*-1;
71        }
          if(b4 == '0'){
73          s4 = s4*-1;
          }
75        */

77        moveTo(s1, s2, s3, s4);

79      }else if(command == "S1"){

81        int value = analogRead(0);
          while(true){
83          int value = analogRead(A1);
            Serial.println(value);
85        }


87
        }
89

91      inputString = "";
        stringComplete = false;
93      Serial.println("nexts");
    }
95  }

97
    void serialEvent() {
99    while (Serial.available()) {
        char inChar = (char)Serial.read();
101     inputString += inChar;
        if (inChar == ';') {
103       stringComplete = true;
        }
105   }
    }
107

109 void moveTo(double m1, double m2, double m3, double m4){

111   bool m1D;
      bool m2D;
113   bool m3D;
      bool m4D;
115
      /*while(!m1D || !m2D || !m3D || !m4D){
```

```
117
      for(long i = -500;i < 2500;i++){
119     if(m1-motor1Pos > i && !m1D){
          steps(m1-motor1Pos, step1, motor1);
121       m1D = true;
        }else if(m2-motor2Pos > i && !m2D){
123       steps(m2-motor2Pos, step2, motor2);
          m2D = true;
125     }else if(m3-motor3Pos > i && !m3D){
          steps(m3-motor3Pos, step3, motor3);
127       m3D = true;
        }else if(m4-motor4Pos > i && !m4D){
129       steps(m4-motor4Pos, step4, motor4);
          m4D = true;
131     }
      }
133
    }*/
135
    int order[4];
137
    boolean m1Set = false;
139 boolean m2Set = false;
    boolean m3Set = false;
141 boolean m4Set = false;

143 for(int i = 0;i < 4;i++){
      int smallest = -2000;
145   if(m1-motor1Pos >= smallest && !m1Set){
          smallest = m1-motor1Pos;
147       order[i] = 0;
      }
149   if(m2-motor2Pos >= smallest && !m2Set){
          smallest = m2-motor2Pos;
151       order[i] = 1;
      }
153   if(m3-motor3Pos >= smallest && !m3Set){
          smallest = m2-motor3Pos;
155       order[i] = 2;
      }
157   if(m4-motor4Pos >= smallest && !m4Set){
          smallest = m4-motor4Pos;
159       order[i] = 3;
      }
161
      if(order[i] == 0){
163     m1Set = true;
      }else if(order[i] == 1){
165     m2Set = true;
      }else if(order[i] == 2){
167     m3Set = true;
      }else if(order[i] == 3){
169     m4Set = true;
      }
171
    }
173

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

177
      if(order[i] == 0){
```

```
179         steps(m1-motor1Pos, step1, motor1);
          }else if(order[i] == 1){
181         steps(m2-motor2Pos, step2, motor2);
          }else if(order[i] == 2){
183         steps(m3-motor3Pos, step3, motor3);
          }else if(order[i] == 3){
185         steps(m4-motor4Pos, step4, motor4);
          }
187
      }
189


191


193


195     motor1Pos = m1;
        motor2Pos = m2;
197     motor3Pos = m3;
        motor4Pos = m4;
199

201 }

203
    void steps(double mm, int stepPin, int motorPin){
205


207
        if(mm > 0){
209      digitalWrite(stepPin, HIGH);
        }else if(mm < 0){
211      mm = mm * -1;
         digitalWrite(stepPin, LOW);
213     }

215     long stepNumber = (200*16)*(mm/fullRotationMM);

217     for(long i = 0;i < stepNumber;i++){
         digitalWrite(motorPin, HIGH);
219      delayMicroseconds(delayTime);              // wait for a second
         digitalWrite(motorPin, LOW);    // turn the LED off by making the voltage LOW
221      delayMicroseconds(delayTime);
        }
223

    }
```

9.1.2 Javaquellcode

./JavaCode/AddValue.java

```
1   import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
3
    import javax.swing.JButton;
5   import javax.swing.JFrame;
    import javax.swing.JLabel;
7   import javax.swing.JTextField;

9
    public class AddValue extends JFrame{
11
        private static final long serialVersionUID = 1L;
```

```
13      private Data data = new Data();
        //private Frame frame = new Frame();
15
        private JTextField text1 = new JTextField("0");
17      private JTextField text2 = new JTextField("0");
        private JTextField text3 = new JTextField("0");
19      private JTextField text4 = new JTextField("0");

21      public AddValue(double amplitude){
            this.setSize(200,200);
23          this.setLayout(null);

25          text4.setText(Double.toString(amplitude));

27          JLabel label1 = new JLabel("x:");
            label1.setBounds(10,13,40,10);
29          this.add(label1);

31          JLabel label2 = new JLabel("y:");
            label2.setBounds(10,53,40,10);
33          this.add(label2);

35          JLabel label3 = new JLabel("z:");
            label3.setBounds(10,93,40,10);
37          this.add(label3);

39          JLabel label4 = new JLabel("Value:");
            label4.setBounds(10,133,40,10);
41          this.add(label4);

43          text1.setBounds(50,7,40,21);
            this.add(text1);
45
            text2.setBounds(50,47,40,21);
47          this.add(text2);

49          text3.setBounds(50,87,40,21);
            this.add(text3);
51
            text4.setBounds(50,127,40,21);
53          this.add(text4);

55          JButton btnSet = new JButton("Set");
            btnSet.setBounds(130,127,50,21);
57          this.add(btnSet);

59          btnSet.addActionListener(new ActionListener(){
                @Override
61              public void actionPerformed(ActionEvent e){
                    data.set(Integer.parseInt(text1.getText()), Integer.parseInt(
                        text2.getText()),
63                          Integer.parseInt(text3.getText()), Double.parseDouble(
                                text4.getText()));
                    //frame.repaintViewer();
65                  //System.out.println(data.getAmplitude(0, 0, 0));
                }
67          });

69          this.setResizable(false);
            this.setVisible(true);
71      }
```

```
73


75 }
```

./JavaCode/AnalyseWindow.java

```
1  import java.awt.BorderLayout;
   import javax.swing.JFrame;
3
   public class AnalyseWindow extends JFrame{
5
       private static final long serialVersionUID = 1L;
7      private double sampleRadiusMultiplier = 1;
       private int dataSampleOpacity = 150;
9
       private boolean cubes;
11     private boolean crosses;

13     public AnalyseWindow(){

15     }


17
       void setOpacity(int opacity){
19         this.dataSampleOpacity = opacity;
       }
21
       void setRadius(double radius){
23         this.sampleRadiusMultiplier = radius;
       }
25
       void useCubes(boolean b){
27         this.cubes = b;
       }
29
       void showCrosses(boolean b){
31         this.crosses = b;
       }
33
       void createWindow(double[][] data, String plane, int layer){
35         this.setTitle("Analyse - Plane: " + plane + " Layer: " + layer);
           this.setSize(500,500);
37         Viewer3D viewer = new Viewer3D(500,500,data.length,data[0].length);
           viewer.paintMap(data, sampleRadiusMultiplier, dataSampleOpacity, cubes,
               crosses);
39         this.add(viewer, BorderLayout.CENTER);
           this.setVisible(true);
41     }


43
   }
```

./JavaCode/Command.java

```
   /**
2   * Created by lewis on 29/09/2016.
    */
4  public class Command {

6      private String command;
       private Point3D point3D;
8
       public Command(String string){
10         setString(string);
```

```
    }

    public Command(String string, Point3D point3D){
        setString(string);
        setPoint(point3D);
    }

    private void setPoint(Point3D point){
        this.point3D = point;
    }

    Point3D getPoint(){
        return point3D;
    }

    private void setString(String string){
        this.command = string;
    }

    String getString(){
        return command;
    }
}
```

./JavaCode/CommandMain.java

```
import com.sun.javaws.jnl.ResourcesDesc;
import gnu.io.CommPortIdentifier;

import javax.swing.*;
import java.util.Observable;
import java.util.Observer;
import java.util.Scanner;

/**
 * Created by Lewis on 18/05/2016.
 */

public class CommandMain implements Observer {

    private static Scanner scanner;
    private static NewConnection connection;
    private static CommandQueue commandsQueue;
    private static Settings settings;


    private static Data data;
    private static AnalyseWindow analyseWindow;
    private static Scan scan;
    private static ScannerObject scannerObject;

    MicrophoneObserver microphoneObserver;

    public void update(Observable observable, Object object){

        if(observable == microphoneObserver){

            Point3D point3D = microphoneObserver.getPoint3D();
            int x = point3D.getX();
            int y = point3D.getY();
            int z = point3D.getZ();
            Microphone microphone = new Microphone(settings.getSamples());
            double amplitude = microphone.calculateAmplitude(settings.getSamples
```

```
                (), settings.getExpectedFrequency(), 5);
            data.set(x,y,z,amplitude);

            System.out.println("Set the data point " + x + ", " + y + ", " + z +
                 " to an amplitude of " + amplitude);

        }
    }

    public CommandMain(MicrophoneObserver microphoneObserver){
        this.microphoneObserver = microphoneObserver;
    }

    public void start(){

        scanner = new Scanner(System.in);

        System.out.println("Welcome to Fresnel Connect+ 2.0 CLI Edition");
        System.out.println("For help, type help");

        //The loop waiting for serial communication
        connection = new NewConnection();

        commandsQueue = new CommandQueue(connection, microphoneObserver);
        commandsQueue.start();

        settings = new Settings();
        data = new Data();
        analyseWindow = new AnalyseWindow();
        scan = new Scan();
        scannerObject = new ScannerObject(settings);


        while(true){

            System.out.println("Ready:");
            String input = scanner.next();
            if(input.toLowerCase().equals("help") || input.toLowerCase().equals(
                "h")){
                help();
            }else if(input.substring(0,1).toLowerCase().equals("g")){
                movement(input);
            }else if(input.substring(0,1).toLowerCase().equals("f")){
                settings(input);
            }else if(input.substring(0,1).toLowerCase().equals("s")){
                scanning(input);
            }else if(input.substring(0,1).toLowerCase().equals("d")){
                data(input);
            }else if(input.substring(0,1).toLowerCase().equals("t")){
                fft(input);
            }else if(input.substring(0,1).toLowerCase().equals("c")){
                communication(input);
            }else if(input.toLowerCase().equals("exit")){
                System.exit(0);
            }else{
                System.out.println("Command not recognized");
            }
        }

    }

    private static void help(){
```

```
97
            System.out.println("Please choose one of the following options:");
99          System.out.println("G - Movement");
            System.out.println("F - Commands");
101         System.out.println("S - Scanning");
            System.out.println("F - Settings");
103         System.out.println("D - Data Visualisation");
            System.out.println("T - FFT Settings");
105         System.out.println("C - Communication");

107         String input = scanner.next();
            if(input.toLowerCase().equals("g")){
109             System.out.println(
                        "G1 Move Relative \n" +
111                     "G2 Move Absolute \n" +
                        "G3 Center Position \n" +
113                     "G4 Reset select motors \n" +
                        "G5 Set speed \n" +
115                     "G6 Get position \n" +
                        "G7 Get speed \n" +
117                     "G8 Reset all \n" +
                        "G9 Set step \n" +
119                     "G10 Left \n" +
                        "G11 Right \n" +
121                     "G12 Forward \n" +
                        "G13 Back \n" +
123                     "G14 Up \n" +
                        "G15 Down \n");
125         }else if(input.toLowerCase().equals("f")){
                System.out.println(
127                     "F1 Get settings\n" +
                        "F2 Set size\n" +
129                     "F3 Set resolution\n" +
                        "F4 Snap resolution\n" +
131                     "F5 Set aspect ratio\n"+
                        "F6 Setup process");
133         }else if(input.toLowerCase().equals("s")){
                System.out.println(
135                     "S1 Set scan resolution\n" +
                        "S2 Set scan type\n" +
137                     "S3 Set scan points\n" +
                        "S4 Get setup scan info\n" +
139                     "S5 Start scan\n" +
                        "S6 Stop scan\n" +
141                     "S7 Scan current position");
            }else if(input.toLowerCase().equals("d")){
143             System.out.println(
                        "D1 Toggle crosses on data\n" +
145                     "D2 Toggle squares\n" +
                        "D3 Fill random\n" +
147                     "D4 Set radius multiplier\n" +
                        "D5 Set sample opacity\n" +
149                     "D6 Set view\n" +
                        "D7 Set layer\n" +
151                     "D8 Open window\n" +
                        "D9 Clear data\n" +
153                     "D10 Save data");
            }else if(input.toLowerCase().equals("t")){
155             System.out.println(
                        "T1 Set sample rate \n" +
157                     "T2 Set sample length\n" +
                        "T3 Set expected peak\n" +
```

```
159                     "T4 Snap peak\n" +
                        "T5 Set peak fluctuation\n" +
161                     "T6 Analyse custom file\n");
            }else if(input.toLowerCase().equals("c")){
163             System.out.println(
                        "C1 Create new connection\n" +
165                     "C2 Display connections\n" +
                        "C2 Display connections\n" +
167                     "C3 Disconnect\n" +
                        "C4 Send custom message\n" +
169                     "C5 Toggle print responses\n" +
                        "C6 Toggle print sent messages\n"+
171                     "C7 Show pending commands\n"+
                        "C8 Number of pending commands\n"+
173                     "C9 Send random data\n");
            }
175     }

177     private static void movement(String input){
            try{
179             int i = Integer.parseInt(input.substring(1,2));
                int x;
181             int y;
                int z;
183             switch (i){
                    case 1: //Move relative
185                     System.out.println("Enter x movement");
                        x = scanner.nextInt();
187                     System.out.println("Enter y movement");
                        y = scanner.nextInt();
189                     System.out.println("Enter z movement");
                        z = scanner.nextInt();
191                     commandsQueue.addCommand(new Command(scannerObject.
                            relativeMove(x,y,z)));
                        break;
193                 case 2: //Move absolute
                        System.out.println("Enter x position");
195                     x = scanner.nextInt();
                        System.out.println("Enter y position");
197                     y = scanner.nextInt();
                        System.out.println("Enter z position");
199                     z = scanner.nextInt();
                        commandsQueue.addCommand(new Command(scannerObject.moveTo(x,
                            y,z)));
201                     break;
                    case 3: //Center position
203                     String s = scannerObject.moveTo(settings.getXLength()/2,
                            settings.getYLength()/2,settings.getZLength()/2);
                        commandsQueue.addCommand(new Command(s, scannerObject.
                            getPosition()));
205                     System.out.println("Position centered");
                        break;
207                 case 4: //Reset select motors
                        System.out.println("Enter a number from 1-5(1-m1, 2-m2...
                            etc");
209                     System.out.println("That motor will then be assumed to be
                            fully retracted,the others extended appropriately.");
                        System.out.println("m1--x--m2");
211                     System.out.println("|       |");
                        System.out.println("y       y");
213                     System.out.println("|       |");
                        System.out.println("m4--x---m3");
```

```
215                     System.out.println("");
                        switch (scanner.nextInt()){
217                         case 1:
                                System.out.println("Motor positions reset.");
219                             scannerObject.setX(0);
                                scannerObject.setY(settings.getYLength());
221                             scannerObject.setZ(0);
                                break;
223                         case 2:
                                System.out.println("Motor positions reset.");
225                             scannerObject.setX(settings.getXLength());
                                scannerObject.setY(settings.getYLength());
227                             scannerObject.setZ(0);
                                break;
229                         case 3:
                                System.out.println("Motor positions reset.");
231                             scannerObject.setX(settings.getXLength());
                                scannerObject.setY(0);
233                             scannerObject.setZ(0);
                                break;
235                         case 4:
                                System.out.println("Motor positions reset.");
237                             scannerObject.setX(0);
                                scannerObject.setY(0);
239                             scannerObject.setZ(0);
                                break;
241                         default:
                                System.out.println("Aborted");
243                             break;
                        }
245                     break;
                    case 5: //Set speed
247                     commandsQueue.addCommand(new Command("G3"));
                        break;
249                 case 6: //Get position
                        System.out.println("X: " + scannerObject.getX() + " Y: " +
                            scannerObject.getY() + " Z: "+ scannerObject.getZ());
251                     break;
                    case 7: //Get speed
253                     System.out.println("Non applicable");
                        break;
255                 case 8: //Reset all
                        System.out.println("Ask someone else");
257                     break;
                    case 9: //Set step
259                     System.out.println("What is this meant to do? Try asking
                            google.");
                        break;
261                 case 10: //Left
                        commandsQueue.addCommand(new Command(scannerObject.
                            incrementMove(10,0,0)));
263                     break;
                    case 11: //Right
265                     commandsQueue.addCommand(new Command(scannerObject.
                            incrementMove(10,0,0)));
                        break;
267                 case 12: //Forward
                        commandsQueue.addCommand(new Command(scannerObject.
                            incrementMove(0,10,0)));
269                     break;
                    case 13: //Back
271                     commandsQueue.addCommand(new Command(scannerObject.
```

```
                        incrementMove(0,-10,0)));
                    break;
                case 14: //Up
                    commandsQueue.addCommand(new Command(scannerObject.
                        incrementMove(0,0,10)));
                    break;
                case 15: //Down
                    commandsQueue.addCommand(new Command(scannerObject.
                        incrementMove(0,0,-10)));
                    break;

            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    private static void settings(int number){
        settings("C" + Integer.toString(number));
    }

    private static void settings(String input){
        try{
            int i = Integer.parseInt(input.substring(1,2));
            String x;
            String y;
            String z;
            switch (i){
                case 1: //Get settings
                    System.out.println("Current settings:");
                    System.out.println("Size:");
                    System.out.println("X:" + settings.getXLength());
                    System.out.println("Y:" + settings.getYLength());
                    System.out.println("Z:" + settings.getZLength());
                    System.out.println("Resolution:");
                    System.out.println("X:" + settings.getXRes());
                    System.out.println("Y:" + settings.getYRes());
                    System.out.println("Z:" + settings.getZRes());
                    System.out.println("Aspect ratio");
                    System.out.println("1:" + settings.getYRatio() + ":" +
                        settings.getZRatio());
                    break;
                case 2: //Set size
                    System.out.println("Enter x size in mm(double)");
                    x = scanner.next();
                    System.out.println("Enter y size in mm(double)");
                    y = scanner.next();
                    System.out.println("Enter z size in mm(double)");
                    z = scanner.next();
                    try {
                        settings.setXLength(Double.parseDouble(x));
                        settings.setYLength(Double.parseDouble(y));
                        settings.setZLength(Double.parseDouble(z));
                        scannerObject.setSize(settings.getXLength(), settings.
                            getYLength(), settings.getZLength());
                    }catch(Exception e){
                        e.printStackTrace();
                    }
                    break;
                case 3: //Set resolution
                    System.out.println("Enter x resolution(integer)");
                    x = scanner.next();
```

```
329                     System.out.println("Enter y resolution(integer)");
                        y = scanner.next();
331                     System.out.println("Enter z resolution(integer)");
                        z = scanner.next();
333                     try {
                            settings.setDataResolution(Integer.parseInt(x),
335                                 Integer.parseInt(y),
                                    Integer.parseInt(z));
337                     }catch(Exception e){
                            e.printStackTrace();
339                     }
                        break;
341                 case 4: //Snap resolution
                        try {
343                         settings.setDataResolution(
                                    settings.getXRes(),
345                                 (int)(settings.getXRes()*settings.getYRatio()),
                                    (int)(settings.getXRes()*settings.getZRatio()));
347                     }catch(Exception e){
                            e.printStackTrace();
349                     }
                        System.out.println("Resolution: X: " + settings.getXRes() +
                            "Y: " +
351                             settings.getYRes() + "Z: " + settings.getZRes());
                        break;
353                 case 5: //Set aspect ratio
                        System.out.println("Setup aspect ratio:");
355                     System.out.println("Enter the yAspect:");
                        y = scanner.next();
357                     System.out.println("Enter the zAspect:");
                        z = scanner.next();
359                     try{
                            settings.setYRatio(Double.parseDouble(y));
361                         settings.setZRatio(Double.parseDouble(z));
                        }catch(Exception e){
363                         e.printStackTrace();
                        }
365                     break;
                    case 6: //Setup process
367                     settings(2);
                        System.out.println("Continue? (y/n)");
369                     if (scanner.next().equals("n")){
                            System.out.println("aborted");
371                         break;
                        }
373                     settings(3);
                        System.out.println("Continue? (y/n)");
375                     if (scanner.next().equals("n")){
                            System.out.println("aborted");
377                         break;
                        }
379                     settings(4);
                        System.out.println("Continue? (y/n)");
381                     if (scanner.next().equals("n")){
                            System.out.println("aborted");
383                         break;
                        }
385                     settings(5);
                        System.out.println("Continue? (y/n)");
387                     if (scanner.next().equals("n")){
                            System.out.println("aborted");
389                         break;
```

```
                    }
391                 settings(6);
                    settings(1);
393                 break;
            }
395     }catch (Exception e){
            e.printStackTrace();
397     }
    }
399
    private static void scanning(String input){
401     try{
            int i = Integer.parseInt(input.substring(1,2));
403         switch (i){
                case 1: //Set scan resolution
405                 System.out.println("Please set from the main settings");
                    break;
407             case 2: //Set scan type
                    System.out.println("Default: Cuboid");
409                 break;
                case 3: //Set scan points
411
                    int x;
413                 int y;
                    int z;
415                 System.out.println("Enter point one");
                    System.out.println("X:");
417                 x = scanner.nextInt();
                    System.out.println("Y:");
419                 y = scanner.nextInt();
                    System.out.println("Z:");
421                 z = scanner.nextInt();
                    Point3D p1 = new Point3D(x,y,z);
423                 System.out.println("Enter point two");
                    System.out.println("X:");
425                 x = scanner.nextInt();
                    System.out.println("Y:");
427                 y = scanner.nextInt();
                    System.out.println("Z:");
429                 z = scanner.nextInt();
                    Point3D p2 = new Point3D(x,y,z);
431                 scan.setSize(p1,p2);
                    System.out.println("Boundaries set");
433
                    break;
435             case 4: //Get setup scan info
                    System.out.println("No info available");
437                 break;
                case 5: //Start scan
439
                    data.setSize(settings.getXRes(), settings.getYRes(),
                        settings.getZRes());
441
                    scan.setResolution(settings.getXRes(), settings.getYRes(),
                        settings.getZRes());
443                 scan.startScan();
                    Point3D[] dividedPoints = scan.getDividedPoints();
445                 Point3D[] points = scan.getPoints();
                    Command[] stringPoints = scannerObject.addWayPoints(points,
                        dividedPoints);
447
                    for(Command c:stringPoints){
```

```
449                         commandsQueue.addCommand(c);
                            commandsQueue.addCommand(new Command("S1", c.getPoint())
                                );
451                     }
                        System.out.println("Scan started, commands being generated")
                            ;
453                     break;
                    case 6: //Stop scan
455                     System.out.println("Please clear the command list to cancel.
                            ");
                        break;
457                 case 7:
                        Microphone microphone = new Microphone(settings.getSamples()
                            );
459                     double amplitude = microphone.calculateAmplitude(settings.
                            getSamples(), settings.getExpectedFrequency(), 5);
                        System.out.println(amplitude);
461
                }
463         }catch (Exception e){
                e.printStackTrace();
465         }
        }
467
        private static void data(String input){
469         try{
                int i = Integer.parseInt(input.substring(1,2));
471             switch (i){
                    case 1: //Toggle crosses on data
473                     System.out.println("Show crosses? (y/n)");
                        if(scanner.next().equals("y")){
475                         analyseWindow.showCrosses(true);
                        }else if(scanner.next().equals("n")){
477                         analyseWindow.showCrosses(false);
                        }
479                     break;
                    case 2: //Toggle squares
481                     System.out.println("Use cubes? (y/n)");
                        if(scanner.next().equals("y")){
483                         analyseWindow.useCubes(true);
                        }else if(scanner.next().equals("n")){
485                         analyseWindow.useCubes(false);
                        }
487                     break;
                    case 3: //Fill random
489                     //data.setSize(10,10,10);
                        //data.fillRandom();
491                     double[][][] amplitudes = data.getDoubleArray();

493                     for (int u = 0;u < amplitudes.length;u++){
                            for (int e = 0;e < amplitudes[0].length;e++){
495                             for (int o = 0;o < amplitudes[0][0].length;o++){
                                    System.out.println(u + " " + e + " " + o + " " +
                                        amplitudes[u][e][o]);
497                             }
                            }
499                     }
                        break;
501                 case 4: //Set radius multiplier
                        System.out.println("Enter radius multiplier:");
503                     analyseWindow.setRadius(Double.parseDouble(scanner.next()));
                        break;
```

```
505                 case 5: //Set sample opacity
                        System.out.println("Enter sample opacity:");
507                     analyseWindow.setOpacity(Integer.parseInt(scanner.next()));
                        break;
509                 case 6: //Set view
                        System.out.println("Enter plane (xy, xz, yz):");
511                     String s = scanner.next();
                        if(s.equals("xy") || s.equals("xz") || s.equals("yz"))data.
                            setPlane(s);
513                     break;
                    case 7: //Set layer
515                     System.out.println("Enter desired layer");
                        data.setLayer(Integer.parseInt(scanner.next()));
517                     break;
                    case 8: //Open window
519                     System.out.println("Opening window...");
                        double[][] dataArray = data.getCut();
521                     analyseWindow.useCubes(true);
                        analyseWindow.createWindow(dataArray, data.getPlaneString(),
                             data.getLayerInt());
523                     break;
                    case 9: //Clear data
525                     data.clear();
                        System.out.println("Data cleared");
527                     break;
                    case 10: //Save
529                     System.out.println("saving");
                        break;
531
                }
533         }catch (Exception e){
                e.printStackTrace();
535         }
        }
537
        private static void fft(String input){
539         try{
                int i = Integer.parseInt(input.substring(1,2));
541             switch (i){
                    case 1: //Set sample rate
543                     System.out.println("Set sample rate");
                        System.out.println("Enter desired frequency in Hz:");
545                     settings.setSampleRate(Integer.parseInt(scanner.next()));
                        break;
547                 case 2: //Set sample length
                        System.out.println("Set sample length");
549                     System.out.println("Enter desired samples");
                        settings.setSamples(Integer.parseInt(scanner.next()));
551                     break;
                    case 3: //Set expected peak
553                     System.out.println("Set expected peak");
                        System.out.println("Enter desired peak frequency in Hz");
555                     settings.setExpectedPeakFrequency(Integer.parseInt(scanner.
                            next()));
                        break;
557                 case 4: //Snap peak
                        int samples = settings.getSamples();
559                     int sampleRate = settings.getSampleRate();
                        int peak = settings.getExpectedFrequency();
561                     FFT fft = new FFT();
                        int[] frequencies = fft.getFrequencies(sampleRate, samples);
563                     System.out.println(frequencies[100]);
```

```
                    boolean found = false;
565                 int peakIncrease = 0;
                    while(!found){
567                     for(int f:frequencies){
                            if(f == peak){
569                             found = true;
                            }
571                     }
                        if(!found){
573                         peak++;
                            peakIncrease++;
575                         if(peakIncrease>1000){
                                peak = 0;
577                             found = true;
                            }
579                     }
                    }
581                 settings.setExpectedPeakFrequency(peak);
                    System.out.println("Peak set to " + peak + "Hz");
583                 break;
                case 5: //Set peak fluctuation
585                 System.out.println("Enter peak fluctuation in Hz:");
                    settings.setPeakFluctuation(Integer.parseInt(scanner.next())
                        );
587                 break;
                case 6: //Analyse custom file
589                 System.out.println("Coming soon...");
                    break;
591         }
        }catch (Exception e){
593         e.printStackTrace();
        }
595 }
    private static void communication(String input){
597     try{
            int i = Integer.parseInt(input.substring(1,2));
599         switch (i){
                case 1: //Create new connection
601                 connection = new NewConnection();

603                 String[] ports = connection.getPorts();
                    for(String s:ports){
605                     System.out.println(s);
                    }
607                 System.out.println("Enter number of connection");
                    int selection = scanner.nextInt();
609                 if(selection != 999 && selection <= ports.length &&
                        selection >= 0){
                        System.out.println("Attempting to connect to " + ports[
                            selection]);
611                     connection.openPort(selection);
                        System.out.println("Probably connected");
613                 }
                    break;
615             case 2: //Display Connections
                    System.out.println(connection.toString());
617                 break;
                case 3: //Disconnect
619                 connection.close();
                    break;
621             case 4://Send custom message
                    //commandsQueue.addCommand("S1");
```

```
623                     commandsQueue.addCommand(new Command(input.substring(3)));
                        break;
625                 case 5: //Toggle print responses
                        System.out.println("Coming soon...");
627                     break;
                    case 6: //Toggle print send messages
629                     System.out.println("Coming soon...");
                        break;
631                 case 7: //Show pending commands
                        String[] commands = commandsQueue.pendingCommandsString();
633                     for(String s: commands){
                            System.out.println(s);
635                     }
                        break;
637                 case 8: //Number of pending commands
                        System.out.println(commandsQueue.pendingCommands());
639                     break;
                    case 9: //send random data
641                     for(int e = 0;i < 20;i++){
                            commandsQueue.addCommand(new Command("Hello World"));
643                     }
                        break;
645             }
            }catch(Exception e){
647             e.printStackTrace();
            }
649     }

651 }
```

./JavaCode/CommandQueue.java

```
1  import java.util.ArrayList;
   import java.util.List;
3
   /**
5   * Created by Lewis on 18/05/2016.
    */
7  public class CommandQueue {

9       private static List<Command> commands = new ArrayList<>();
        private static List<Point3D> points = new ArrayList<>();
11
        private NewConnection connection;
13      private MicrophoneObserver microphoneObserver;

15      public CommandQueue(NewConnection connection, MicrophoneObserver
            microphoneObserver){
            this.connection = connection;
17          this.microphoneObserver = microphoneObserver;
        }
19
        void addCommand(Command s) {
21          commands.add(s);

23      }


25
        Command getCommand(){
27          Command s = commands.get(0);
            commands.remove(0);
29          return s;
        }
```

```
31
    void start(){
33      Thread t = new Thread(){
            @Override
35          public void run(){
                while (true){
37                  //System.out.println(connection.getClass());
                    try{
39                      sleep(250);
                    }catch (Exception e){
41                      e.printStackTrace();
                    }
43                  if (connection.isReady() && commands.size() > 0){
                        Command c = getCommand();
45                      if(c.getString().equals("S1")){
                            microphoneObserver.setPoint3D(c.getPoint());
47                          microphoneObserver.setAmplitude();
                            //microphoneObserver.notifyObservers();
49                      }else{
                            connection.write(c.getString());
51                      }

53

                    }
55              }
            }
57      };
        t.start();
59  }

61  int pendingCommands(){
        return commands.size();
63  }

65  String[] pendingCommandsString(){
        return commands.toArray(new String[0]);
67  }

69



71
}
```

./JavaCode/Connection.java

```
import java.io.BufferedReader;
2 import java.io.InputStreamReader;
import java.io.PrintStream;
4 import com.fazecast.jSerialComm.*;

6
public class Connection {
8
    /**
10   * Due to the nature of the machine to be controlled, I have decided to
         create a gCode similar style group of commands
     * for sending information and requesting the machine do specific things.
         Here is a complete list of all commands:
12   *
     * Prefixes:
14   * M - Request info
     * G - Control device
16   *
```

```
         * M0 - Pings
18       * M1 - returns numbers 0 - 1000 (5ms interval)
         * M2 -
20       *
         * G0 - home all axis
22       * G1 - set all axis' current position to their home
         * G2 - move to position [x,y,z]
24       * G3 - move to relative position[x,y,z]
         */
26
        private static SerialPort comPort;
28
        private static boolean ready = true;
30      private static int portNum;
        //All measurements in mm
32
        /**
34       * Creates the connection
         * @param port
36       */
        void openPort(final int port) {
38
            portNum = port;
40          Thread t = new Thread() {
                public void run() {
42                  SerialPort[] ports = SerialPort.getCommPorts();
                    if (ports.length > port) {
44                      comPort = ports[port];
                        comPort.setBaudRate(115200);
46                      comPort.openPort();
                        if (comPort.isOpen()) {
48                          try {

50                              BufferedReader reader = new BufferedReader(new
                                    InputStreamReader(comPort.getInputStream()));
                                try {
52                                  while (comPort.isOpen()) {
                                        while (!reader.ready()) Thread.sleep(100);
54                                      try {
                                            String string = reader.readLine();
56                                          string = string.replace('_', ' ');
                                            receive(string);
58                                      } catch (Exception e) {
                                            comPort.closePort();
60                                          System.out.println("Connection lost");
                                        }
62                                  }
                                } catch (Exception e) {
64                                  e.printStackTrace();
                                    retry();
66                                  comPort.closePort();
                                    System.out.println("Connection lost");
68                              }
                            } catch (Exception e) {
70                              e.printStackTrace();
                                retry();
72                              comPort.closePort();
                                System.out.println("Connection lost");
74                          }
                        } else {
76                          retry();
                            comPort.closePort();
```

```
78                          System.out.println("Connection lost");
                        }
80                  } else {
                        retry();
82                      comPort.closePort();
                        System.out.println("Connection lost");
84                  }
                }
86          };
            t.start();
88
        }
90

92      void retry(){
            openPort(portNum);
94          System.out.println("retrying");

96      }
        /**
98       * Tells whether it is ready to write again
         */
100
        boolean isReady(){
102         return ready;
        }
104
        /**
106      * Handles incoming strings
         * @param s
108      */
        private void receive(String s){
110         if(s.equals("nexts")){
                ready = true;
112         }else if(s.substring(0,1).equals("S")){

114         }else{
                System.out.println("Incoming data: " + s);
116         }


118
        }
120
        /**
122      * Writes the string to the com port
         * @param string
124      */
        void write(String string){
126         ready = false;
            System.out.println(comPort.getSystemPortName());
128         if(comPort != null){
                if(comPort.isOpen()){
130                 PrintStream printer = new PrintStream(comPort.getOutputStream())
                        ;
                    System.out.println("Writing \"" + string + "\" to port");
132                 printer.print(string + ";");
                }
134         }
        }
136
        /**
138      * Closes the com port and connection
```

```
     */
    void close(){
        try{
            comPort.closePort();
        }catch(Exception e){
        }

    }

    /**
     * Method returns a string array containing the possible ports
     * @return
     */
    String[] getPorts(){

        SerialPort[] ports = SerialPort.getCommPorts();
        String[] portsString = new String[ports.length];
        for(int i = 0;i < ports.length;i++){
            portsString[i] = ports[i].getDescriptivePortName();
        }

        return portsString;

    }


}
```

./JavaCode/Data.java

```

public class Data {

    private double[][][] amplitudes;
    private double[][][] amplitudesdB;

    private double maxValue = 20;

    private static String plane = "xy";
    private static int layer = 0;


    void setLayer(int layer){
        this.layer = layer;
    }

    void setPlane(String plane){
        this.plane = plane;

    }

    void setSize(int x, int y, int z){
        amplitudes = new double[x][y][z];
        amplitudesdB = new double[x][y][z];
        maxValue = 0;
    }

    void fillRandom(){
        /*
        for(int i = 0;i < amplitudes.length;i++){
            for(int e = 0;e < amplitudes[0].length;e++){
                for(int f = 0;f < amplitudes[0][0].length;f++){
                    amplitudes[i][e][f] = Math.random();
```

```
                    amplitudesdB[i][e][f] = amplitudes[i][e][f] * 20;
                }
            }
        }
        */

    }

    void set(int x, int y, int z, double data){



        amplitudesdB[x][y][z] = data;

        if(data > maxValue){
            maxValue = data;
            for(int i = 0;i < amplitudes.length;i++){
                for(int e = 0;e < amplitudes[0].length;e++){
                    for(int f = 0;f < amplitudes[0][0].length;f++){
                        amplitudes[i][e][f] = amplitudesdB[i][e][f]/maxValue;
                    }
                }
            }
            amplitudes[x][y][z] = data/maxValue;
        }else{
            amplitudes[x][y][z] = data/maxValue;
        }
    }

    void clear(){
        amplitudes = new double[amplitudes.length][amplitudes[0].length][
            amplitudes[0][0].length];
        amplitudesdB = new double[amplitudes.length][amplitudes[0].length][
            amplitudes[0][0].length];
        maxValue = 20;
    }


    double[][] getCut(){

        int x = amplitudes.length;
        int y = amplitudes[0].length;
        int z = amplitudes[0][0].length;

        if(plane.equals("xy")){
            double[][] data = new double[y][x];
            for(int i = 0;i < y;i++){
                for(int e = 0;e < x;e++){
                    data[i][e] = amplitudes[x-1-e][y-1-i][layer];
                }
            }
            return data;
        }else if(plane.equals("xz")){
            double[][] data = new double[x][z];
            for(int i = 0;i < x;i++){
                for(int e = 0;e < z;e++){
                    data[i][e] = amplitudes[i][layer][e];
                }
            }
            return data;

        }else if(plane.equals("yz")){
```

```
            double[][] data = new double[y][z];
            for(int i = 0;i < y;i++){
                for(int e = 0;e < z;e++){
                    data[i][e] = amplitudes[layer][y-1-i][e];
                }
            }
            return data;
        }
        return null;

    }

    double[][][] getDoubleArray(){
        return amplitudes;
    }


    String getPlaneString(){
        return plane;
    }

    int getLayerInt(){
        return layer;
    }

    double getAmplitude(int x, int y, int z){
        return amplitudes[x][y][z];
    }
}
```

./JavaCode/FFT.java

```
import org.jtransforms.fft.DoubleFFT_1D;

public class FFT {


    double getAmplitude(double[] samples, int frequency){

        DoubleFFT_1D fftDo = new DoubleFFT_1D(samples.length);
        double[] fft = new double[samples.length * 2];
        System.arraycopy(samples, 0, fft, 0, samples.length);
        fftDo.realForwardFull(fft);
        double[] magnitudes = new double[fft.length/2];
        for(int i = 0;i < fft.length/2;i++){
            double real = fft[i*2];
            double imaginary = fft[i*2+1];
            magnitudes[i] = Math.sqrt(real*real+imaginary*imaginary);
        }
        double amplitude = 0;
        for(int i = 0;i < magnitudes.length;i++){
            if(i*44100/magnitudes.length == frequency){
                amplitude = magnitudes[i];
            }
        }
        return amplitude;
    }

    int[] getFrequencies(int sampleRate, int samples){
        int[] frequencies = new int[samples];
        for(int i = 0;i < samples;i++){
            frequencies[i] = i*sampleRate/samples;
        }
```

```
32          return frequencies;
        }
34
}
```

./JavaCode/Frequency.java

```
1   /**
     * Created by lewis on 02/10/2016.
3    */
    public class Frequency {
5       private int frequency;
        private double amplitude;
7
        public Frequency(double amplitude, int frequency){
9           this.frequency = frequency;
            this.amplitude = amplitude;
11      }

13      double getAmplitude() {
            return amplitude;
15      }

17      int getFrequency() {
            return frequency;
19      }

21      void print(){
            System.out.print(frequency + "  ");
23          System.out.println(amplitude);
        }
25  }
```

./JavaCode/Main.java

```
    import java.io.*;
2   import java.lang.reflect.Array;
    import java.util.*;
4   import java.util.zip.DeflaterOutputStream;

6   /**
     * Created by lewis on 28/09/2016.
8    */
    public class Main {
10
        public static void main(String[] args) {
12
            MicrophoneObserver microphoneObserver = new MicrophoneObserver();
14          CommandMain commandMain = new CommandMain(microphoneObserver);
            microphoneObserver.addObserver(commandMain);
16          commandMain.start();

18      }

20  }
```

./JavaCode/Microphone.java

```
    import sun.nio.cs.ext.MacHebrew;
2
    import javax.sound.sampled.*;
4   import javax.xml.crypto.dsig.spec.ExcC14NParameterSpec;
    import java.io.ByteArrayOutputStream;
6   import java.io.FileWriter;
    import java.util.ArrayList;
```

```
8  import java.util.Collections;
   import java.util.Comparator;
10
   /**
12  * Created by lewis on 25/09/2016.
    */
14 public class Microphone {

16     public static final int DEFAULT_SAMPLE_LENGTH = 1024;
       private int sampleLength = 0;
18     //private boolean stopped;

20     public static void main(String[] args) {
           Microphone mic = new Microphone(DEFAULT_SAMPLE_LENGTH);
22         mic.record();
       }
24
       public Microphone(int sampleLength) {
26         this.sampleLength = sampleLength;

28     }

30     double[] record() {

32         AudioFormat format = getAudioFormat();
           DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);
34
           // checks if system supports the data line
36         if (!AudioSystem.isLineSupported(info)) {
               System.out.println("Line not supported");
38         }
           try {
40             TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(info)
                   ;
               dataLine.open(format);
42
               byte[] data;// = new byte[dataLine.getBufferSize()];
44
               ByteArrayOutputStream out = new ByteArrayOutputStream();
46             int numBytesRead;
               data = new byte[dataLine.getBufferSize()];
48
               // Begin audio capture.
50             dataLine.start();

52             Thread stopper = new Thread() {
                   @Override
54                 public void run() {
                       try {
56                         sleep(500);
                       } catch (InterruptedException e) {
58                         e.printStackTrace();
                       }
60                     //stopped = true;
                   }
62             };
               //stopper.start();
64             /*int k = 0;
               while (dataLine.isActive()){
66                 k++;
                   k = 0;
68             }*/
```

```
            //System.out.println(k);
70          //while (!stopped) {
                // Read the next chunk of data from the TargetDataLine.
72          numBytesRead = dataLine.read(data, 0, dataLine.getBufferSize());
            //System.out.println(numBytesRead);
74              // Save this chunk of data.
            //System.out.println(data.length);
76          out.write(data, 0, numBytesRead);

78          //}
            /*
80          FFT fft = new FFT();
            int[] frequencies = fft.getFrequencies(44100, 512);
82
            for (int i:frequencies){
84              System.out.println(i);
            }
86          */
            //}
88
            //for ()
90
            dataLine.stop();
92          dataLine.close();

94          byte[] audioData = out.toByteArray();
            double[] audioDataDouble = new double[audioData.length];
96
            byte largest = 0;
98          for (byte d : audioData) {
                if (d > largest)
100                 largest = d;
            }
102
            for (int i = 0; i < audioData.length; i++) {
104             audioDataDouble[i] = (1.0 * audioData[i] / 127);
                /*double val;
106             if (audioDataDouble[i] == 0) {
                    val = 0;
108             }else{
                    if (audioDataDouble[i] < 0){
110                     val = audioDataDouble[i]*-1;
                        val = Math.log10(val)*20;
112                     val = val*-1;
                    }else {
114                     val = Math.log10(audioDataDouble[i])*20;
                    }
116
                }
118
                audioDataDouble[i] = val;
                */
120
122             //System.out.println(audioDataDouble[i]);
                //System.out.println(Math.log10(audioDataDouble[i])*20);
124             //audioDataDouble[i] = Math.log10(audioDataDouble[i])*20;
                //System.out.println(audioDataDouble[]);
126         }

128         //System.out.println(audioDataDouble[0]);
            return audioDataDouble;
130
```

```
        } catch (LineUnavailableException e) {
            e.printStackTrace();
            System.out.println("line unavailable");
        }

        return null;


    }


    private AudioFormat getAudioFormat() {

        float sampleRate = 44100;
        int sampleSizeInBits = 8;
        int channels = 1;
        boolean signed = true;
        boolean bigEndian = true;

        AudioFormat format = new AudioFormat(sampleRate, sampleSizeInBits,
            channels, signed, bigEndian);
        return format;
    }

    double calculateAmplitude(int sampleLength, int frequency, int cycles){
        double amplitude = 0;
        cycles = 1;

        for (int i = 0;i < cycles;i++){
            amplitude += calculateAmplitude(sampleLength, frequency);
        }
        return amplitude/cycles;
    }

    private double calculateAmplitude(int sampleLength, int frequency) {

        FFT fft = new FFT();
        int[] amplitudes = fft.getFrequencies(44100, sampleLength);
        Microphone microphone = new Microphone(sampleLength);
        double[] values = microphone.record();
        ArrayList<Frequency> list = new ArrayList<>();

        for (int i = 0; i < amplitudes.length; i++) {
            if (amplitudes[i] > frequency - 200 && amplitudes[i] < frequency +
                200) {
                list.add(new Frequency(fft.getAmplitude(values, amplitudes[i]),
                    amplitudes[i]));
                //System.out.print(amplitudes[i] + "   ");
                //System.out.println(list.get(i).getAmplitude());
            }
        }

        Collections.sort(list, new FrequencyComparator());
        Collections.reverse(list);

        double amplitude = list.get(0).getAmplitude();

        amplitude = 20 * Math.log10(amplitude);


        /*
        double p1 = list.get(0).getAmplitude();
```

```
190             double p2 = list.get(1).getAmplitude();
                double p3 = list.get(2).getAmplitude();
192
                double peakPos = .5*(p1-p3)/(p1-2*p2+p3);
194             System.out.println(peakPos);
                double peakMagnitude = p3-.75*(p1-p3)*peakPos;
196             System.out.println();
                System.out.println(list.get(0).getFrequency());
198             System.out.println(list.get(0).getAmplitude());
                System.out.println(peakMagnitude);
200             */
                return amplitude;
202         }

204 }


206
    class FrequencyComparator implements Comparator<Frequency> {
208         @Override
            public int compare(Frequency d1, Frequency d2) {
210             if (d1.getAmplitude() < d2.getAmplitude()) {
                    return -1;
212             } else if (d1.getAmplitude() > d2.getAmplitude()) {
                    return 1;
214             } else {
                    return 0;
216             }
            }
218 }
```

./JavaCode/MicrophoneObserver.java

```
1   import java.util.Observable;

3   /**
     * Created by lewis on 29/09/2016.
5    */
    public class MicrophoneObserver extends Observable {
7
        private double amplitude;
9       private Point3D point3D;

11      public MicrophoneObserver(){
            point3D = new Point3D(0,0,0);
13          amplitude = -1;
        }
15
        void setAmplitude(){
17          setChanged();
            notifyObservers();
19      }


21
        void setPoint3D(Point3D point3D){
23          this.point3D = point3D;
        }
25
        Point3D getPoint3D(){
27          return point3D;
        }
29
    }
```

./JavaCode/NewConnection.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.List;
import java.util.Scanner;

import gnu.io.*;


/**
 * Code copied from arduino site
 * Uses Rxtx library from arduino site
 */
public class NewConnection implements SerialPortEventListener {

    private static final String PORT_NAMES[] = {
            "COM0", "COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7", "
               COM8", "COM9", "COM10", "COM11", "COM12",};
    static SerialPort serialPort;

    private static BufferedReader input;
    /**
     * The output stream to the port
     */
    private static OutputStream output;

    /**
     * The print stream for writing strings to the output
     */
    private static PrintStream printStream;
    /**
     * Milliseconds to block while waiting for port open
     */
    private static final int TIME_OUT = 2000;
    /**
     * Default bits per second for COM port.
     */
    private static final int DATA_RATE = 115200;

    /**
     * Ready to receive or send information
     */
    private static boolean ready = true;

    public static void main(String[] args) throws Exception {
        NewConnection connection = new NewConnection();

        //connection.getPortIdentifiers();
        String[] positions = connection.getPorts();

        Scanner scanner = new Scanner(System.in);
        System.out.println(positions[0]);

        connection.initialize(0);


    }
```

```
60
        /**
62       * Tells whether it is ready to write again
         */
64
        boolean isReady(){
66          return ready;
        }
68
        /**
70       * Retries to connect
         */
72      void retry(){
            //openPort(portNum);
74          System.out.println("doesn't currently retry");

76      }


78
        /**
80       * Writes the string to the com port
         * @param string
82       */
        void write(String string){
84          ready = false;
            try {
86              if(serialPort != null){
                    System.out.println("Writing \"" + string + "\" to port");
88                  printStream.print(string + ";");
                }else{
90                  System.out.println("Serial port couldn't be reached");
                }
92          }catch (Exception e){
                e.printStackTrace();
94              System.out.println("Problem writing to port");
            }
96
        }
98
        /**
100      * Method returns a string array containing the possible ports
         * @return
102      */
        CommPortIdentifier[] getPortIdentifiers(){
104
            List<CommPortIdentifier> list = new ArrayList<>();
106
            CommPortIdentifier portId = null;
108         Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

110         //First, Find an instance of serial port as set in PORT_NAMES.
            while (portEnum.hasMoreElements()) {
112             CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.
                    nextElement();
                list.add(currPortId);
114         }
            if (list == null) {
116             System.out.println("Could not find COM port.");
                return null;
118         }

120         CommPortIdentifier[] identifiers = new CommPortIdentifier[list.size()];
```

```
        identifiers = list.toArray(identifiers);
122     int i = identifiers.length;
        return identifiers;
124 }

126 String[] getPorts(){
        CommPortIdentifier[] identifiers = getPortIdentifiers();
128     if (identifiers != null){
            String[] idArray = new String[identifiers.length];
130         for (int i = 0;i < idArray.length;i++){
                idArray[i] = identifiers[i].getName();
132         }
            return idArray;
134     }else{
            return null;
136     }

138 }

140 void openPort(final int port){
        initialize(port);
142 }

144 private void initialize(int position) {

146     CommPortIdentifier portId = getPortIdentifiers()[position];

148     try {
            // open serial port, and use class name for the appName.
150         serialPort = (SerialPort) portId.open(this.getClass().getName(),
                    TIME_OUT);
152
            // set port parameters
154         serialPort.setSerialPortParams(DATA_RATE,
                    SerialPort.DATABITS_8,
156                 SerialPort.STOPBITS_1,
                    SerialPort.PARITY_NONE);
158
            // open the streams
160         input = new BufferedReader(new InputStreamReader(serialPort.
                getInputStream()));
            output = serialPort.getOutputStream();
162         printStream = new PrintStream(output);

164         // add event listeners
            serialPort.addEventListener(this);
166         serialPort.notifyOnDataAvailable(true);
        } catch (Exception e) {
168         System.err.println(e.toString());
        }
170 }

172 /**
     * This should be called when you stop using the port.
174  * This will prevent port locking on platforms like Linux.
     */
176 public synchronized void close() {
        if (serialPort != null) {
178         serialPort.removeEventListener();
            serialPort.close();
180     }
    }
```

```
182
    /**
184  * Handle an event on the serial port. Read the data and print it.
     */
186 public synchronized void serialEvent(SerialPortEvent oEvent) {
        if (oEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
188         try {
                String string = input.readLine();
190             string = string.replace('_', ' ');
                receive(string);
192         } catch (Exception e) {
                System.err.println(e.toString());
194         }
        }
196     // Ignore all the other eventTypes, but you should consider the other
            ones.
    }
198

200 private void receive(String s){
        if(s.equals("nexts")){
202         ready = true;
        }else if(s.substring(0,1).equals("S")){
204
        }else{
206         System.out.println("Incoming data: " + s);
        }
208

210 }

212 }
```

./JavaCode/Point3D.java

```
2  public class Point3D {

4      private int x, y, z;

6      public Point3D(int x, int y, int z){
           this.x = x;
8          this.y = y;
           this.z = z;
10     }

12     int getX(){
           return x;
14     }
       int getY(){
16         return y;
       }
18     int getZ(){
           return z;
20     }

22     void print(){
           System.out.println("X: " + x + "  Y: " + y + "  Z: " + z);
24     }

26 }
```

./JavaCode/Scan.java

```
import java.util.ArrayList;
import java.util.List;

public class Scan {

    //Point3D point1 = new Point3D
    private static Point3D p1;
    private static Point3D p2;

    private static boolean paused = false;
    private static boolean finished = false;

    private static int zRes;
    private static int yRes;
    private static int xRes;

    private static String type;

    private static ScanPath path;



    void setSize(Point3D p1, Point3D p2){
        Scan.p1 = p1;
        Scan.p2 = p2;
    }

    /**void setType(String type){
        Scan.type = type;
    }**/

    void startScan(){
        path = new ScanPath(p1, p2, type, xRes, yRes, zRes);
        path.createPath();

    }

    Point3D[] getPoints(){
        return path.getPoints();
    }

    Point3D[] getDividedPoints(){
        return path.getDividedPoints();
    }

    void setResolution(int x, int y, int z){
        this.xRes = x;
        this.yRes = y;
        this.zRes = z;
    }


}
```

./JavaCode/ScannerObject.java

```
public class ScannerObject {

    /*private static double m1;
    private static double m2;
    private static double m3;
```

```
    private static double m4;*/

    private double speed;
    private double step;

    private double xPos = 0;
    private double yPos = 0;
    private double zPos = 0;

    private double width;
    private double depth;
    private double height;

    private Settings settings;


    //Deprecated due to incompatibility
    /**boolean movePos(double x, double y, double z){

        if(x<= width && y <= depth && z <= height){
            double g1 = Math.sqrt((y*y+x*x)+z*z) -
                    Math.sqrt((yPos*yPos+xPos*xPos)+zPos*zPos);
            double g2 = Math.sqrt((((width-x)*(width-x))+y*y)+z*z) -
                    Math.sqrt((((width-xPos)*(width-xPos))+yPos*yPos)+zPos*zPos)
                        ;
            double g3 = Math.sqrt((((depth-y)*(depth-y))+x*x)+z*z) -
                    Math.sqrt((((depth-yPos)*(depth-yPos))+xPos*xPos)+zPos*zPos)
                        ;
            double g4 = Math.sqrt((((width-x)*(width-x))+((depth-y)*(depth-y)))+
                z*z) -
                    Math.sqrt((((width-xPos)*(width-xPos))+((depth-yPos)*(depth-
                        yPos)))+zPos*zPos);
            System.out.println(g1);
            System.out.println(g2);
            System.out.println(g3);
            System.out.println(g4);
            xPos = x;
            yPos = y;
            zPos = z;
            return true;
        }else{
            return false;
        }
        //return false;
    }*/

    public ScannerObject(Settings settings){
        this.speed = settings.getSpeed();
        this.step = settings.getStep();

        this.width = settings.getXLength();
        this.depth = settings.getYLength();
        this.height = settings.getZLength();

        this.settings = settings;

    }


    private int[] getStringLength(double x, double y, double z){
        double g1 = Math.sqrt((y*y+x*x)+z*z);
        double g2 = Math.sqrt((((width-x)*(width-x))+y*y)+z*z);
```

```
        double g3 = Math.sqrt((((width-x)*(width-x))+((depth-y)*(depth-y)))+z*z)
            ;
66      double g4 = Math.sqrt((((depth-y)*(depth-y))+x*x)+z*z);
        int[] vals = {new Double(g1).intValue(), new Double(g2).intValue(), new
            Double(g3).intValue(), new Double(g4).intValue()};
68
        return vals;
70  }

72  Point3D getPosition(){
        System.out.println(xPos);
74      return new Point3D((int)((xPos/width)*settings.getXRes()),
                           (int)((yPos/depth)*settings.getYRes()),
76                         (int)((zPos/height)*settings.getZRes()));
    }
78

80  String moveTo(double x, double y, double z){
        if(x <= width && y <= depth && z <= height){
82          xPos = x;
            yPos = y;
84          zPos = z;
            int values[] = getStringLength(x,y,z);
86          String spaces[] = new String[4];
            boolean[] directions = new boolean[4];
88          for (int i = 0;i < directions.length;i++){
                if (values[i] < 0){
90                  directions[i] = true;
                    values[i] = values[i] * -1;
92              }
                int spaceLength = 5-String.valueOf(values[i]).length();
94              spaces[i] = "";
                for (int e = 0; e < spaceLength;e++){
96                  spaces[i] = spaces[i] + "0";
                }
98          }

100         //String out = "G1_" + spaces[0] + values[0] + "_" + spaces[1]  +
                values[1] + "_"  + spaces[2] + values[2] + "_" + spaces[3] +
                values[3] + "_" +
            //        directions[0] + directions[1] + directions[2] + directions
                [3];
102
            String out = "G1_" + spaces[0] + values[0] + "_" + spaces[1]  +
                values[1] + "_"  + spaces[2] + values[2] + "_" + spaces[3] +
                values[3];
104
            //System.out.println(out);
106         return out;
        }else{
108         return null;
        }
110
    }
112
    String relativeMove(double x, double y, double z) {
114     xPos = xPos + x;
        yPos = yPos + y;
116     zPos = zPos + z;
        return moveTo(xPos, yPos, zPos);
118
    }
```

```
120
        String incrementMove(int x, int y, int z) {
122         return relativeMove(x*step, y*step, z*step);
        }
124

126     Command[] addWayPoints(Point3D[] points, Point3D[] dividedPoints){
            Command[] wayPoints = new Command[points.length];
128         int counter = 0;

130         for(Point3D p: points){
                wayPoints[counter] = new Command(moveTo(p.getX(), p.getY(), p.getZ()
                    ), dividedPoints[counter]);
132             counter++;
            }
134         return wayPoints;
        }
136


138
        private boolean isActive(){
140         return true;
        }
142
        double[] getPos(){
144         double[] pos = {xPos, yPos, zPos};
            return pos;
146     }

148     boolean setX(double value){
            xPos = value;
150         return isActive();
        }
152     boolean setY(double value){
            yPos = value;
154         return isActive();
        }
156     boolean setZ(double value){
            zPos = value;
158         return isActive();
        }
160
        double getX(){
162         return xPos;
        }
164     double getY(){
            return yPos;
166     }
        double getZ(){
168         return zPos;
        }
170
        void setSpeed(double speed){
172         this.speed = speed;
        }
174     void setStep(double step){
            this.step = step;
176     }


178
        void setSize(double width, double depth, double height){
180         this.width = width;
```

```
        this.depth = depth;
182     this.height = height;
    }
184
}
```

./JavaCode/ScannerPanel.java

```
1  import java.awt.Color;
   import java.awt.Graphics;
3  import javax.swing.JPanel;

5  public class ScannerPanel extends JPanel{

7      private static final long serialVersionUID = 1L;
       private static ScannerObject scanner = new ScannerObject(new Settings());
9      private static Settings settings = new Settings();
       private static double[] xPositions;
11     private static double[] yPositions;
       private static boolean pathEnabled = false;
13

15     public ScannerPanel(Settings settings){
           setSize(250,350);
17     }

19     void updatePanel(){
           repaint();
21     }

23     @Override
       public void paintComponent(Graphics g){
25         super.paintComponent(g);

27         double x = scanner.getX();
           double y = scanner.getY();
29         double xMax = settings.getXLength();
           double yMax = settings.getYLength();
31         int width = getWidth();
           int height = getHeight();
33
           int xPos = (int)((x/xMax)*width);
35         int yPos = (int)((y/yMax)*height);
           g.setColor(Color.blue);
37         g.drawLine(xPos, yPos, width, 0);
           g.drawLine(0, 0, xPos, yPos);
39         g.drawLine(0, height, xPos, yPos);
           g.drawLine(width, height, xPos, yPos);
41         g.setColor(Color.gray);
           g.fillRect(xPos-5,yPos-5, 10, 10);
43         g.setColor(Color.BLACK);

45         g.drawString("X: " + x + " Y: " + y + " Z: " + scanner.getZ(), 10,
               height-10);

47         if(pathEnabled){
               g.setColor(Color.cyan);
49             for(int i = 0;i < xPositions.length-1;i++){
                   g.drawLine((int)(xPositions[i]*getWidth()), (int)(yPositions[i]*
                       getHeight()),
51                         (int)(xPositions[i+1]*getWidth()), (int)(yPositions[i
                               +1]*getHeight()));
               }
```

```
53          }

55

57      }

59      void paintPath(double[] xPositions, double[] yPositions){
            ScannerPanel.yPositions = yPositions;
61          ScannerPanel.xPositions = xPositions;
            pathEnabled = true;
63          repaint();
        }
65      void unpaintPath(){
            xPositions = null;
67          yPositions = null;
            pathEnabled = false;
69          repaint();
        }
71
}
```

./JavaCode/ScanPath.java

```
import java.util.ArrayList;
2 import java.util.List;

4 public class ScanPath {

6       private Point3D p1;
        private Point3D p2;
8
        private int x;
10      private int y;
        private int z;
12
        List<Point3D> scanPoints = new ArrayList<Point3D>();
14      List<Point3D> scanPointsDivided = new ArrayList<>();

16      //private String type;

18      public ScanPath(Point3D p1, Point3D p2, String type, int x, int y, int z){
            //this.type = type;
20          this.p1 = p1;
            this.p2 = p2;
22
            this.x = x;
24          this.y = y;
            this.z = z;
26
        }
28
        int createPath(){
30
            //clearPath();
32          int counter = 0;
            //All p1 axis MUST have lower values than (or be equal to) p2!!!
34          for(int i = 0;i < x;i++){
                for(int e = 0;e < y;e++){
36                  for(int o = 0;o < z;o++){
                        scanPoints.add(new Point3D((int)(p1.getX()+(1.0*i/x)*(p2.
                            getX()-p1.getX()))
38                                                  , (int)(p1.getY()+(1.0*e/y)*(p2.
                                                      getY()-p1.getY())))
```

```
                                                    , (int)(p1.getZ()+(1.0*o/z)*(p2.
                                                        getZ()-p1.getZ())));
40                      scanPointsDivided.add(new Point3D(i,e,o));
                        counter++;
42                  }
                }
44          }
            counter ++;
46          return counter;

48      }

50      private void clearPath(){
            scanPoints.clear();
52          scanPointsDivided.clear();
        }
54

56      Point3D[] getPoints(){
            Point3D[] points = new Point3D[scanPoints.size()];
58          int counter = 0;
            for (Point3D p: scanPoints){
60              points[counter] = p;
                counter++;
62          }
            return points;
64      }

66      Point3D[] getDividedPoints(){
            Point3D[] points = new Point3D[scanPointsDivided.size()];
68          int counter = 0;
            for (Point3D p: scanPointsDivided){
70              points[counter] = p;
                counter++;
72          }
            return points;
74      }

76  }
```

./JavaCode/Settings.java

```
2   public class Settings {

4

        /**
6

8        0;scantype
         0;scanx1
10       0;scany1
         0;scanz1
12       0;scanx2
         0;scany2
14       0;scanz2
         0;crosses
16       0;squares
         0;radius
18       0;opacity
         0;view
20       0;layer
         0;samplerate
```

```
22        0;samplelength
          0;expectedpeak
24        0;peakfluctuation
          0;printresponses
26        */

28      private static int xLayers = 10;
        private static int yLayers = 10;
30      private static int zLayers = 10;

32      private static double sampleRadiusMultiplier = 2;
        private static int dataSampleOpacity = 127;
34      private static boolean crossesEnabled = true;
        private static boolean aspectLocked = false;
36      private static boolean useCubes = false;
        private static boolean calculatePoints = true;
38
        private static double yRatio = 2;
40      private static double zRatio = 1;

42      private static double xLength = 785;  //All units in mm
        private static double yLength = 1030;
44      private static double zLength = 800;

46      private static int samples = 16384;
        private static int sampleRate = 44100;
48      private static int expectedPeakFrequency = 15000;
        private static int peakFluctuation = 10;
50
        private static double speed = 10;
52      private static double step = 1;

54      private static double width = 100;
        private static double depth = 100;
56      private static double height = 100;


58
        /*
60       * Settings for the visualisation
         */
62
        double getSpeed(){
64          return speed;
        }
66      double getStep(){
            return step;
68      }
        double getWidth(){
70          return width;
        }
72      double getDepth(){
            return depth;
74      }
        double getHeight(){
76          return height;
        }
78      void setXLength(double length){
            xLength = length;
80      }
        void setYLength(double length){
82          yLength = length;
        }
```

```
84      void setZLength(double length){
            zLength = length;
86      }
        double getXLength(){
88          return xLength;
        }
90      double getYLength(){
            return yLength;
92      }
        double getZLength(){
94          return zLength;
        }
96      void setYRatio(double ratio){
            yRatio = ratio;
98      }
        double getYRatio(){
100         return yRatio;
        }
102     void setZRatio(double ratio){
            zRatio = ratio;
104     }
        double getZRatio(){
106         return zRatio;
        }
108     void setSampleRadiusMultiplier(double multiplier){
            sampleRadiusMultiplier = multiplier;
110     }
        double getSampleRadiusMultiplier(){
112         return sampleRadiusMultiplier;
        }
114     void setDataSampleOpacity(int opacity){
            dataSampleOpacity = opacity;
116     }
        int getDataSampleOpacity(){
118         return dataSampleOpacity;
        }
120     void setCrossesEnabled(boolean b){
            crossesEnabled = b;
122     }
        boolean crossesEnabled(){
124         return crossesEnabled;
        }
126     void setCalculatePoints(boolean b){
            calculatePoints = b;
128     }
        boolean getCalculatePoints(){
130         return calculatePoints;
        }
132     void setAspectLocked(boolean b){
            aspectLocked = b;
134     }
        boolean aspectLocked(){
136         return aspectLocked;
        }
138     void setUseCubes(boolean b){
            useCubes = b;
140     }
        boolean useCubes(){
142         return useCubes;
        }
144     void setDataResolution(int x, int y, int z){
            xLayers = x;
```

```
146         yLayers = y;
            zLayers = z;
148
        }
150     int getLayers(String plane){
            if(plane.equals("xy"))return zLayers-1;
152         if(plane.equals("xz"))return yLayers-1;
            if(plane.equals("yz"))return xLayers-1;
154         return 0;
        }
156
        int getXRes(){
158         return xLayers;
        }
160     int getYRes(){
            return yLayers;
162     }
        int getZRes(){
164         return zLayers;
        }
166
        void setPeakFluctuation(int x){
168         Settings.peakFluctuation = x;
        }
170     void setSampleRate(int x){
            Settings.sampleRate = x;
172     }
        void setSamples(int x){
174         Settings.samples = x;
        }
176     void setExpectedPeakFrequency(int x){
            Settings.expectedPeakFrequency = x;
178     }

180     int getExpectedFrequency(){
            return Settings.expectedPeakFrequency;
182     }
        int getSamples(){
184         return Settings.samples;
        }
186     int getSampleRate(){
            return Settings.sampleRate;
188     }
        int getPeakFluctuation(){
190         return Settings.peakFluctuation;
        }
192

194 }
```

./JavaCode/UIListener.java

```
/**
2  * Created by Lewis on 09/05/2016.
 */
4 interface UIListener{

6

    /**
8    * UI Listeners for the movement in the control panel
     */
10  void btnLeftPressed();
    void btnRightPressed();
```

```
12      void btnBackPressed();
        void btnForwardPressed();
14      void btnUpPressed();
        void btnDownPressed();
16      void btnHomePressed();
        void btnCenterPressed();
18      void btnResetPressed();

20      /**
         * UI Listeners for the scan buttons in the control panel
22       */
        void btnSaveScanPressed();
24      void btnStartScanPressed();
        void btnPauseScanPressed();
26      void btnCancelScanPressed();

28      /**
         * UI Listeners for the MenuItems in the MenuBar
30       */
        void comConnect(int port);
32      void comDisconnect();
        void comRefresh();
34      void comWindow();

36      void pathWindowPressed();
        void controlWindowPressed();
38
        void exitPressed();
40      void savePressed();
        void newPressed();
42      void preferencesPressed();


44
        /**
46       * Listeners for the com connection
         */
48
        void comClosed();
50


52
}
```

./JavaCode/Viewer3D.java

```
1   import java.awt.Color;
    import java.awt.Graphics;
3   import java.awt.Graphics2D;
    import java.awt.RadialGradientPaint;
5   import java.awt.geom.Point2D;
    import java.awt.image.BufferedImage;
7
    import javax.swing.JPanel;
9
    public class Viewer3D extends JPanel{
11
        private static final long serialVersionUID = 1L;
13
        private double[][] data;     //Values from 0 - 1 representing the amplitude
15      private double[] xPositions;
        private double[] yPositions;
17      private double spacingX;
        private double spacingY;
```

```
19      private boolean useCubes = true;
        private boolean crossesEnabled = true;
21
        double sampleRadiusMultiplier = 1;
23      int dataSampleOpacity = 150;

25
        public Viewer3D(int width, int height, int dataX, int dataY){
27          this.setSize(width,height);
            data = new double[dataX][dataY];
29      }

31      void paintMap(double[][] data, double radius, int opacity, boolean useCubes,
             boolean showCrosses){
            this.useCubes = useCubes;
33          this.crossesEnabled = showCrosses;
            this.data = data;
35          this.sampleRadiusMultiplier = radius;
            this.dataSampleOpacity = opacity;
37          repaint();
        }
39

41      @Override
        public void paintComponent(Graphics g){
43          super.paintComponent(g);

45          Graphics2D g2d = (Graphics2D) g.create();

47          //The following creates boring squares:
            if(useCubes){
49              int spacingX = (int) (this.getSize().getWidth()/data.length);
                int spacingY = (int) (this.getSize().getHeight()/data[0].length);
51              int x = 0;
                int y = 0;
53
                for(int i = 0;i < data.length;i++){
55                  for(int e = 0;e < data[i].length;e++){
                        if(data[i][e]!=0){
57
                            g.setColor(new Color(255,255-((int) (data[i][e]*255)),0)
                                );
59                          g.fillRect(spacingX*i, spacingY*e, spacingX, spacingY);
                            g.setColor(Color.black);
61                          if(crossesEnabled){
                                x = spacingX*i+spacingX/2-2;
63                              y = spacingY*e+spacingY/2-2;
                                g.drawLine(x, y, x+5, y+5);
65                              g.drawLine(x+5, y, x, y+5);
                            }
67                      }
                    }
69              }
            }else if(true){ //!settings.getCalculatePoints()
71
                spacingX = this.getSize().getWidth()/data.length;
73              spacingY = this.getSize().getHeight()/data[0].length;

75              xPositions = new double[data.length];
                yPositions = new double[data[0].length];
77
                for(int i = 0;i < data.length;i++){
```

```
79                  xPositions[i] = spacingX*i+spacingX/2;
                }
81              for(int i = 0;i < data[0].length;i++){
                    yPositions[i] = spacingY*i+spacingY/2;
83              }

85              if(data != null){

87                  //BufferedImage image = new BufferedImage(width, height,
                        BufferedImage.TYPE_3BYTE_BGR);

89                  for(int i = 0;i < xPositions.length;i++){
                        for(int e = 0;e < yPositions.length;e++){
91                          //g.fillOval(, , ovalWidth, ovalHeight);
                            float radius = (int) (spacingX*sampleRadiusMultiplier);
93                          Point2D center = new Point2D.Double((int) (xPositions[i
                                ]),(int) (yPositions[e]));
                            float[] distances = {0.0f, 1.0f};
95                          Color[] colours = {new Color(255,255-(int)(data[i][e
                                ]*255),dataSampleOpacity), new Color(0,0,0,0)};
                            if(data[i][e] == 0){
97                              colours[0] = new Color(255,255-(int)(data[i][e]*255)
                                    ,0,0);
                            }
99                          RadialGradientPaint rgp = new RadialGradientPaint(center
                                , radius, distances, colours);
                            g2d.setPaint(rgp);
101                         g2d.fillRect(0, 0, getWidth(), getHeight());
                        }
103                 }

105                 //g.drawImage(image, 0, 0, this.getWidth(), this.getHeight(),
                        null);

107                 if(crossesEnabled){
                        int x = 0;
109                     int y = 0;
                        for(int i = 0;i < data.length;i++){
111                         for(int e = 0;e < data[i].length;e++){
                                if(data[i][e]!=0){
113                                 g.setColor(Color.black);
                                    x =(int) (xPositions[i])-2;
115                                 y = (int) (yPositions[e])-2;
                                    g.drawLine(x, y, x+5, y+5);
117                                 g.drawLine(x+5, y, x, y+5);
                                }
119                         }
                        }
121                 }
            }
123         }else if(false){ //settings.getCalculatePoints()

125

            spacingX = this.getSize().getWidth()/data.length;
127         spacingY = this.getSize().getHeight()/data[0].length;

129         xPositions = new double[data.length];
            yPositions = new double[data[0].length];
131

            for(int i = 0;i < data.length;i++){
133             xPositions[i] = spacingX*i+spacingX/2;
            }
```

```
135             for(int i = 0;i < data[0].length;i++){
                    yPositions[i] = spacingY*i+spacingY/2;
137             }

139             BufferedImage image = new BufferedImage(getWidth(),getHeight(),
                    BufferedImage.TYPE_INT_RGB);
                Graphics2D graphics = image.createGraphics();
141             graphics.setColor(Color.white);
                graphics.fillRect(0,0,getWidth(),getHeight());
143             g.drawImage(image,0,0, null);

145         }
        }
147 }
```

„Wir erklären hiermit, dass wir die vorliegende Projektarbeit selbständig und ohne unerlaubte fremde Hilfe erstellt haben und dass alle Quellen, Hilfsmittel und Internetseiten wahrheitsgetreu verwendet wurden und belegt sind.“

Lewis Beauchamp

Stanislaw Zytynski