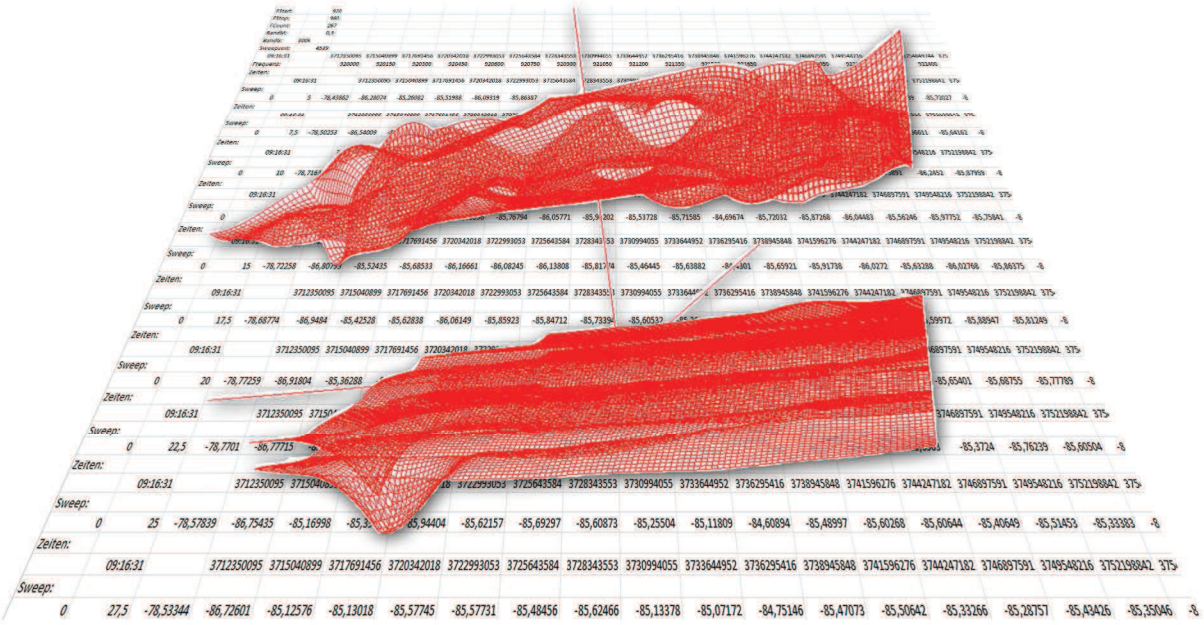
Methoden zur Visualisierung von Messdaten eines Radioteleskops



Wettbewerb „Jugend Forscht“ 2014

Lukas Amann (16 Jahre)

Arbeitsgemeinschaft „Jugend Forscht“
des Christian - Gymnasiums Hermannsburg
Leitung: StD Thomas Biedermann

Inhaltsverzeichnis

1. Einleitung

Seit nun schon über einem Jahr existiert an meiner Schule die Radioastronomie-AG, die es sich zum Ziel gesetzt hat, ein funktionsfähiges Radioteleskop zu bauen, um damit radioastronomische Messungen durchführen zu können. Aus diesem Grund möchte ich ein Programm schreiben, das die Daten, die später von dem Teleskop empfangen werden sollen, mit verschiedenen Methoden, u.a. einer 3D-Darstellung, visualisieren und Kenngrößen bestimmen kann, damit die spätere Auswertung vereinfacht wird. Als Orientierung diente mir ein Programm, das bereits von einer Jungforscherin geschrieben wurde.

Des Weiteren ist erst eine Messung, die wir mit einem unserer Teleskope auf der IdeenExpo 2013, gemacht haben vorhanden. Deshalb verwende ich diese neue und einige umformatierte alte Messungen.

Ich nutze die Entwicklungsoberfläche „Visual Studio 2010“ mit dem dazugehörigen „.Net Framework“ und die Programmiersprache C# zur Programmierung.

1.1 Messdaten

Das Radioteleskop misst in einem definierbaren Frequenzbereich. Die Messung eines Radioteleskopes ist so aufgebaut, dass es eine bestimmte Position anfährt, eine Messung durchführt und die Position ändert, z.B. eine Schrittweite in der Höhe (Elevation). Hier wird das eingestellte Frequenzspektrum durchgemessen. Daher sind nun zwei solcher Messsätze vorhanden. Das Diagramm wird nun um eine Achse erweitert: Die Position im Azimut. Dieser Prozess wird fortgeführt, bis der höchste Elevationswert erreicht ist. In Abb. 2 sind z.B. Messsätze an vier Positionen dargestellt. Hinzu kommt, dass sich das Teleskop nicht nur in der Elevation, sondern auch in horizontaler Richtung (Azimut) dreht. Das führt dazu, dass jeder Position eine x- (Azimut) und y-Koordinate (Elevation) und wie an dem Beispiel meiner Daten, 267 Messwerte für die entsprechenden Frequenzen zugeordnet sind. Eine solche Darstellung aller Daten ist in einem normalen Diagramm nicht möglich. Daher ordne ich einer Position nur einen auswählbaren Wert zu und stelle sie in einem dreidimensionalen System dar.

Abb. 3 zeigt einen Ausschnitt aus einer Messdatei. Der Inhalt jeder Zeile wird durch ein Schlüsselwort kenntlich gemacht.

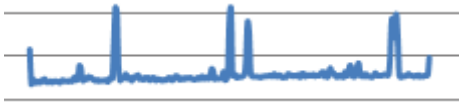


Abb. 1: Pegel in Abhängigkeit zur Frequenz

Abb. 2: Pegel in Abhängigkeit zur Frequenz an vier Positionen

```
FStart: ;920
FStop: ;960
FCount: ;267
BandM: ;0,3
BandG: ;300k
Sweepzeit: ;4539
09:16:31;;3712350095;3715040899;3717691456;3720342018;372
57192473;3959942465;3962593026;3965243936;3967894483;3970
4893376;4207543871;4210194369;4212844822;4215495542;42181
Frequenz: ;;920000;920150;920300;920450;920600;920750;920
00;941750;941900;942050;942200;942350;942500;942650;94280
Zeiten:
;09:16:31;;3712350095;3715040899;3717691456;3720342018;37
957192473;3959942465;3962593026;3965243936;3967894483;397
04893376;4207543871;4210194369;4212844822;4215495542;4218
Sweep:
000,00;005,00;-78,43862;-86,28074;-85,26082;-85,51988;-86,
-85,26764;-85,28474;-84,86947;-85,06259;-85,01678;-84,834
64518;-84,28808;-84,66482;-84,30453;-84,74323;-83,55533;-
Zeiten:
;09:16:31;;3712350095;3715040899;3717691456;3720342018;37
957192473;3959942465;3962593026;3965243936;3967894483;397
04893376;4207543871;4210194369;4212844822;4215495542;4218
Sweep:
000,00;007,50;-78,50253;-86,54009;-85,23116;-85,65262;-86,
```

Abb. 3: Ausschnitt einer Messdatei im Editor

2.Vorüberlegungen

Zu Beginn muss festgelegt werden, welche Eigenschaften das neue Programm haben soll.

Zum Einen sollte es .csv-Dateien einlesen können, weil die Messungen in diesem Format gespeichert werden. Das bereits existierende Programm war dazu zwar in der Lage, allerdings musste zuerst noch eine Konvertierung der Dateien vorgenommen werden, was nun nicht mehr von Nöten sein sollte.

Wie schon beschrieben sollte es zudem die Messwerte dreidimensional darstellen können. Da zu jeder Position nur ein Wert dargestellt werden kann, sollte wählbar sein, wie dieser Wert berechnet wird: der Mittelwert der Messdaten einer Position, Maximum oder Minimum, Wert einer wählbaren Frequenz und Mittelwerte eines definierbaren Frequenzbereiches. Dazu sollte über eine Farbe die empfangene Leistung in dB erkennbar sein, um einen schnellen Überblick über den Wert der dritten Achse und seine Farbe zu ermöglichen.
Außerdem sollte das Programm die Maxima und Minima der Signalstärke für bestimmte Frequenzen oder das globale Maximum bzw. Minimum ermitteln können.

3. Software

Das Programm muss zu Beginn die Datei einlesen. Anschließend müssen dann die Daten zum Zeichnen umsortiert, nach der entsprechenden Darstellungsmethode analysiert und schließlich dargestellt werden.

3.1 Einlesen der Messparameter

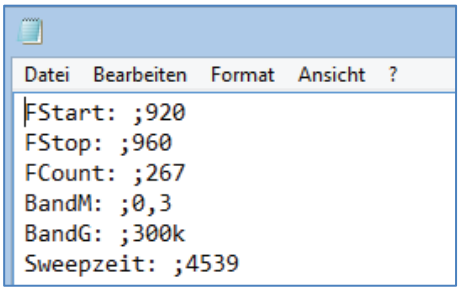


Abb. 4: .csv-Datei im Texteditor

Abb. 5 zeigt den C#-Programmcode zum Einlesen der Messparameter. Da die Messparameter am Anfang der Datei stehen und nach jedem eine neue Zeile beginnt, werden sie zeilenweise eingelesen. Der eingelesene String wird zuerst in einer Variablen (tmp) zwischengespeichert. Da jeder Parameter durch ein Semikolon von seinem Namen getrennt ist (Abb. 4), kann mit Hilfe der Funktion „Substring“ der Parameter isoliert werden. Anschließend wird der Wert des Parameters in einer globalen Variable (z.B. „FCount“ für die Anzahl der Frequenzen) gespeichert, damit sie zur späteren Verwendung verfügbar sind. Dieser Vorgang wird für alle Parameter wiederholt (Z. 104-127 (Abb. 5)).
Anschließend wird noch die Schrittweite aus der Startfrequenz (FStart), der Endfrequenz (FStop) und der Anzahl der Frequenzen (FCount) bestimmt (Z. 129).

```
104                 tmp = Re.ReadLine();
105                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
106                 FStart =Convert.ToDouble(tmp);
107
108                 tmp = Re.ReadLine();
109                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
110                 FStop = Convert.ToDouble(tmp);
111
112                 tmp = Re.ReadLine();
113                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
114                 FCount = Convert.ToDouble(tmp);
115
116                 tmp = Re.ReadLine();
117                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
118                 BandM = Convert.ToDouble(tmp);
119
120                 tmp = Re.ReadLine();
121                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
122                 tmp = tmp.Substring(0, tmp.IndexOf("k"));
123                 BandG = Convert.ToDouble(tmp);
124
125                 tmp = Re.ReadLine();
126                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
127                 Sweepzeit = Convert.ToDouble(tmp);
128
129                 Schrittweite = (FStop - FStart) / FCount;
```

Abb. 5: C# Code: Einlesen der Messparameter

3.2 Einlesen der Zeiten und Frequenzen

Nach den Messparametern folgt ein Absatz, in dem die Zeitstempel der Messung festgehalten sind. Zu Beginn wird der komplette Absatz eingelesen. Da die Anzahl der Zeitstempel mit der Anzahl der Frequenzen übereinstimmt, ist es möglich ein Array für diese Zeiten zu initialisieren. Ähnlich wie bei den Parametern dienen hier die Semikolons als Beschränkung für jeden Zeitstempel. Unter erneutem einlesen einer Zeile und der Benutzung der „Substring-Methode“ können nun die einzelnen Zeitstempel bestimmt und in dem Array (Zeiten) gespeichert werden.

Da die Frequenzen in der Datei in kHz und die Parameter in MHz angegeben sind, habe ich mich entschieden die Frequenzen in MHz anzugeben, weshalb ich den eingelesenen Double (freq) durch 1000 dividieren muss (Z.153).

Das Einlesen der Frequenzen funktioniert grundsätzlich auf die gleiche Weise (Abb. 5). Allerdings ist es möglich, dass die Frequenzen in der Datei nicht in der richtigen Reihenfolge gespeichert wurden, wenn der Sweep (die Messung) abgebrochen und später fortgesetzt wurde. Aus diesem Grund wird zunächst ein korrigierter Index (k) ermittelt (Z.154). Dadurch ist es möglich im späteren Programmverlauf die Messwerte richtig zu ordnen. Damit nur ganzzahlige Indexe vorliegen, müssen die berechneten Ergebnisse mit Hilfe der „Math.Round-Funktion“ gerundet werden (Z.156). „AwayFromZero“ gibt hierbei an, dass bei fünf aufgerundet wird.

Anschließend werden die Frequenzen und die dazugehörigen Indexe in einem zweidimensionalen Array (Frequenzen) gespeichert.

```
149                 for (i = 0; i < FCount; i++)
150                 {
151                     tmp2 = tmp.Substring(0, tmp.IndexOf(";"));
152                     freq = Convert.ToDouble(tmp2);
153                     freq = freq / 1000;
154                     k = (freq - FStart) / Schrittweite;
155                     Frequenzen[0, i] = freq;
156                     Frequenzen[1, i] = Math.Round(k,MidpointRounding.AwayFromZero);
157                     tmp = tmp.Substring(tmp.IndexOf(";") + 1);
158 
159                     if (Frequenzen[1, i] == 0)
160                     {
161                         indexff = i;
162                     }
163                 }
164
```

Abb. 6: C#-Code zum Einlesen der Frequenzen

3.3 Bestimmen der Messpunkte

Die Messung des Teleskopes ist so aufgebaut, dass jeder Position des Teleskopes eine horizontale und vertikale Koordinate x- und y-Koordinate, die Messwerte für die jeweiligen Frequenzen und die Zeitstempel zugeordnet sind. Um diese Informationen in einem Datensatz zu speichert wird ein eigener Struct nötigt. Dieser Typ heißt in dem Programm „Data“. Jedem Objekt von dem Typ „Data“ werden die x- und y-Koordinaten (Az und El), ein Array mit den Messdaten (Sweep), das jeweilige Maxima (Max) bzw. Minima (Min) der aktuellen Messdaten und ein Array für die Zeiten (Times) zugeordnet. Das nebenstehende Struktogramm zeigt, wie die Datensätze ermittelt werden. Zu Beginn der in Abb. 7 dargestellten While-Schleife wird geprüft, ob sich der „Streamreader“ bereits am Ender der Datei befindet. Ist das nicht der Fall, so wird einer Liste (complMess), die später die Datensätze enthalten soll, ein Objekt vom Typ Data (tmpData) übergeben. Anschließend liest er eine weitere Zeile ein. Steht am Anfang dieser Zeile das Wort „Zeiten“, so wird, ähnlich wie beim vorherigen Einlesen der Zeiten bzw. der Frequenzen das Array des Objekts vom Typ „Data“ gefüllt. Anschließend wird eine weitere Zeile eingelesen. Steht am Anfang dieser Ziele das Wort „Sweep“ so werden die ersten beiden Werte in Data.Az bzw. Data.El gespeichert, da es sich dabei um die Korrdinaten handelt und anschließend Data.Sweep mit den Messwerten gefüllt. Anschließend werden das Maxima (Max) bzw. das Minima (Min) dieser Messdaten bestit.

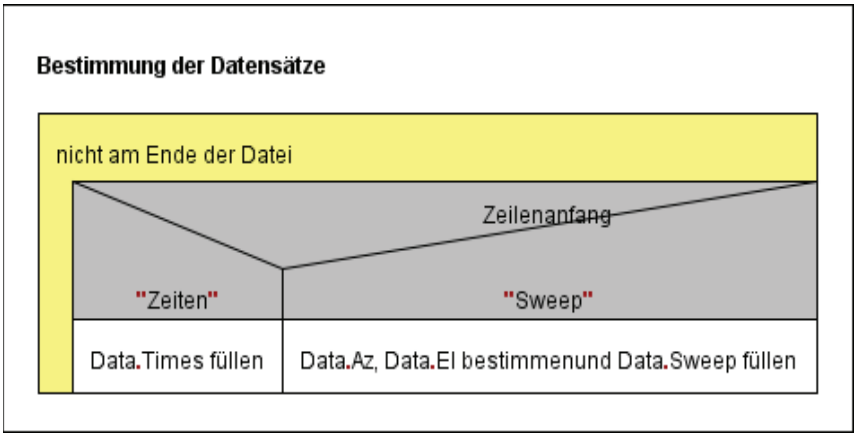


Abb. 7: Struktogramm zur Bestimmung der Datensätze

Die „Case-Abfrage“ ermöglicht das Einlesen der Daten auch dann, wenn diese Reinfolge bei dem Schreiben der Datei nicht eingehalten wurde. Sollte das Ende der Datei erreicht sein, bricht an dieser Stelle die While-Schliefe ab, ansonsten beginnt der Prozess von vorn.

Für das spätere Zeichnen werden noch die y- und x-Schrittweiten benötigt. Diese werden aus den kleinsten bzw. größten x- (xMax und xMin) und y-Werten (yMax und yMin) und ihrer Anzahl (xCount bzw. yCount) ermittelt. Die jeweiligen Schrittweiten und die Anzahl werden für die spätere Nutzung in globalen Variablen gespeichert und die Schrittweiten werden direkt auf der Benutzeroberfläche ausgegeben.

3.4 Bestimmen der Punkte

Da mit C# ohne weitere Frameworks 3D-Darstellungen nur für „Windows Presentation Foundation-Anwendungen“ (WPF) möglich sind und es zusätzliche Zeit beansprucht hätte, sich die Art der Programmierung für diese Anwendungen anzueignen, suchte ich im Internet nach einer einfacheren Variante und stieß auf eine Bibliothek namens „Drawing3d“ [1].

	x →											
y ↓	Data 0,0	Data 1,0	Data 2,0	Data 3,0	Data 4,0	Data 5,0	Data 6,0	Data 7,0	Data 8,0	Data 9,0	Data 10,0	...Data xCount, 0
	Data 0,1	Data 1,1	Data 2,1	Data 3,1	Data 4,1	Data 5,1	Data 6,1	Data 7,1	Data 8,1	Data 9,1	Data 10,1	
	Data 0,2	Data 1,2	Data 2,2	Data 3,2	Data 4,2	Data 5,2	Data 6,2	Data 7,2	Data 8,2	Data 9,2	Data 10,2	
	Data 0,3	Data 1,3	Data 2,3	Data 3,3	Data 4,3	Data 5,3	Data 6,3	Data 7,3	Data 8,3	Data 9,3	Data 10,3	
	Data 0,4	Data 1,4	Data 2,4	Data 3,4	Data 4,4	Data 5,4	Data 6,4	Data 7,4	Data 8,4	Data 9,4	Data 10,4	
	Data 0,5	Data 1,5	Data 2,5	Data 3,5	Data 4,5	Data 5,5	Data 6,5	Data 7,5	Data 8,5	Data 9,5	Data 10,5	
	Data 0,6	Data 1,6	Data 2,6	Data 3,6	Data 4,6	Data 5,6	Data 6,6	Data 7,6	Data 8,6	Data 9,6	Data 10,6	
	Data 0,7	Data 1,7	Data 2,7	Data 3,7	Data 4,7	Data 5,7	Data 6,7	Data 7,7	Data 8,7	Data 9,7	Data 10,7	
	Data 0,8	Data 1,8	Data 2,8	Data 3,8	Data 4,8	Data 5,8	Data 6,8	Data 7,8	Data 8,8	Data 9,8	Data 10,8	
	Data 0,9	Data 1,9	Data 2,9	Data 3,9	Data 4,9	Data 5,9	Data 6,9	Data 7,9	Data 8,9	Data 9,9	Data 10,9	
	Data 0,10	Data 1,10	Data 2,10	Data 3,10	Data 4,10	Data 5,10	Data 6,10	Data 7,10	Data 8,10	Data 9,10	Data 10,10	
	...Data 0,YCount											

Abb. 8: Aufbau von "DataArray"

In dieser Bibliothek ist eine Klasse namens „BezierSurfaces“ vorhanden, der ein 4x4 Array übergeben wird und die daraus ein Netz berechnet. Aus diesem Grund müssen die Messdatensäte zerlegt werden. Um die einzelnen Arrays später verwenden zu können, werden sie in einem neuen Struct „Punkt“ in „Punkt.part“ gespeichert. Diese Objekte vom Typ „Punkt“ werden dann einer Liste („Punkte“) übergeben.

Um diesen Prozess zu vereinfachen werden zunächst die Messdatensetze aus der Liste, in der sich gespeichert sind (complMess), in einem Array(DataArray) so gespeichert, dass sie nach ihren x- bzw. y-Koordinaten sortiert sind. Dadurch wird eine „Karte“ erstellt, die das Auswählen der zu zeichnenden Punkte vereinfacht.

Das Zerlegen der Messung übernimmt eine eigene Funktion namens „calcPart“. Für jede spätere Darstellungsmethode müssen diese Punkte unterschiedlich ausgewählt werden.

Um nun die 4x4 Arrays („tmp.part“) aus den vorhanden Daten für die Darstellung der Mittelwerte zu erstellen, wird der in Abb. 9 rot umrahmte Bereich bestimmt, indem die Stelle von DataArray [0,0] als Startpunkt gewählt wird. Ausgehend von diesem Punkt wird in einer for-Schleife bei einem Durchlauf die Objekte vier Spalten (00 - 3,0) einer Zeile ermittelt. Im nächsten Durchlauf wird die nächste Zeile (0,0 – 0,3) bestimmt und die gleichen Spalten eingelesen. Damit das später zu Netz zusammenhängt, muss dem nächsten Array das Ende des letzten Arrays als Startpunkt

	x →											
y ↓	Data 0,0	Data 1,0	Data 2,0	Data 3,0	Data 4,0	Data 5,0	Data 6,0	Data 7,0	Data 8,0	Data 9,0	Data 10,0	...Data xCount, 0
	Data 0,1	Data 1,1	Data 2,1	Data 3,1	Data 4,1	Data 5,1	Data 6,1	Data 7,1	Data 8,1	Data 9,1	Data 10,1	
	Data 0,2	Data 1,2	Data 2,2	Data 3,2	Data 4,2	Data 5,2	Data 6,2	Data 7,2	Data 8,2	Data 9,2	Data 10,2	
	Data 0,3	Data 1,3	Data 2,3	Data 3,3	Data 4,3	Data 5,3	Data 6,3	Data 7,3	Data 8,3	Data 9,3	Data 10,3	
	Data 0,4	Data 1,4	Data 2,4	Data 3,4	Data 4,4	Data 5,4	Data 6,4	Data 7,4	Data 8,4	Data 9,4	Data 10,4	
	Data 0,5	Data 1,5	Data 2,5	Data 3,5	Data 4,5	Data 5,5	Data 6,5	Data 7,5	Data 8,5	Data 9,5	Data 10,5	
	Data 0,6	Data 1,6	Data 2,6	Data 3,6	Data 4,6	Data 5,6	Data 6,6	Data 7,6	Data 8,6	Data 9,6	Data 10,6	
	Data 0,7	Data 1,7	Data 2,7	Data 3,7	Data 4,7	Data 5,7	Data 6,7	Data 7,7	Data 8,7	Data 9,7	Data 10,7	
	Data 0,8	Data 1,8	Data 2,8	Data 3,8	Data 4,8	Data 5,8	Data 6,8	Data 7,8	Data 8,8	Data 9,8	Data 10,8	
	Data 0,9	Data 1,9	Data 2,9	Data 3,9	Data 4,9	Data 5,9	Data 6,9	Data 7,9	Data 8,9	Data 9,9	Data 10,9	
	Data 0,10	Data 1,10	Data 2,10	Data 3,10	Data 4,10	Data 5,10	Data 6,10	Data 7,10	Data 8,10	Data 9,10	Data 10,10	
	...Data 0,YCount											

Abb. 9: erstes 4x4 Array

	x →											
y ↓	Data 0,0	Data 1,0	Data 2,0	Data 3,0	Data 4,0	Data 5,0	Data 6,0	Data 7,0	Data 8,0	Data 9,0	Data 10,0	...Data xCount, 0
	Data 0,1	Data 1,1	Data 2,1	Data 3,1	Data 4,1	Data 5,1	Data 6,1	Data 7,1	Data 8,1	Data 9,1	Data 10,1	
	Data 0,2	Data 1,2	Data 2,2	Data 3,2	Data 4,2	Data 5,2	Data 6,2	Data 7,2	Data 8,2	Data 9,2	Data 10,2	
	Data 0,3	Data 1,3	Data 2,3	Data 3,3	Data 4,3	Data 5,3	Data 6,3	Data 7,3	Data 8,3	Data 9,3	Data 10,3	
	Data 0,4	Data 1,4	Data 2,4	Data 3,4	Data 4,4	Data 5,4	Data 6,4	Data 7,4	Data 8,4	Data 9,4	Data 10,4	
	Data 0,5	Data 1,5	Data 2,5	Data 3,5	Data 4,5	Data 5,5	Data 6,5	Data 7,5	Data 8,5	Data 9,5	Data 10,5	
	Data 0,6	Data 1,6	Data 2,6	Data 3,6	Data 4,6	Data 5,6	Data 6,6	Data 7,6	Data 8,6	Data 9,6	Data 10,6	
	Data 0,7	Data 1,7	Data 2,7	Data 3,7	Data 4,7	Data 5,7	Data 6,7	Data 7,7	Data 8,7	Data 9,7	Data 10,7	
	Data 0,8	Data 1,8	Data 2,8	Data 3,8	Data 4,8	Data 5,8	Data 6,8	Data 7,8	Data 8,8	Data 9,8	Data 10,8	
	Data 0,9	Data 1,9	Data 2,9	Data 3,9	Data 4,9	Data 5,9	Data 6,9	Data 7,9	Data 8,9	Data 9,9	Data 10,9	
	Data 0,10	Data 1,10	Data 2,10	Data 3,10	Data 4,10	Data 5,10	Data 6,10	Data 7,10	Data 8,10	Data 9,10	Data 10,10	
	...Data 0,YCount											

Abb. 10: erstes 4x4 Array in zweiter Reihe

gegeben werden (Abb. 10). Um sicherzustellen, dass dem „BezierSurface“ mindestens ein 3x4 Array übergeben werden kann, wird geprüft, ob noch mindestens drei Spalten zum Ende übrig sind. Ist dies nicht der Fall, also das Ende der Messreihe erreicht wird y um drei erhöht (Abb. 11) und der gesamte Prozess wird wiederholt, solange bis wieder keine drei Spalten übrig sind und y um drei erhöht wird. Dadurch werden zuerst die Punkte in x-Richtung und anschließend in y-Richtung bestimmt.
Für die einzelnen Darstellungsmöglichkeiten muss unterschieden werden, welche Messwerte übergeben werden. Will man sich die Mittelwerte anzeigen lassen, so werden von „Data. Sweep“ die Mittelwerte gebildet. Möchte man die Frequenzen durchfahren, wird der Wert aus „Data.Sweeep“ übergeben, der denselben Index, wie die gewählte Frequenz besitzt. Dieser Index, d.h. die Frequenz ist wählbar durch einen Schieberegler auf der graphischen Oberfläche des Programmes. Lässt man sich eine gewählte Frequenz mit gewählter Bandbreite anzeigen, so wird der Bereich der Werte in „Data.Sweep“ über die Indizes der Frequenzen in dem Frequenzbereich bestimmt („calcMittel“). Anschließend wird daraus der Mittelwert, der dann an „tmp.part“ übergeben wird.

3.5 Zeichnen der Daten

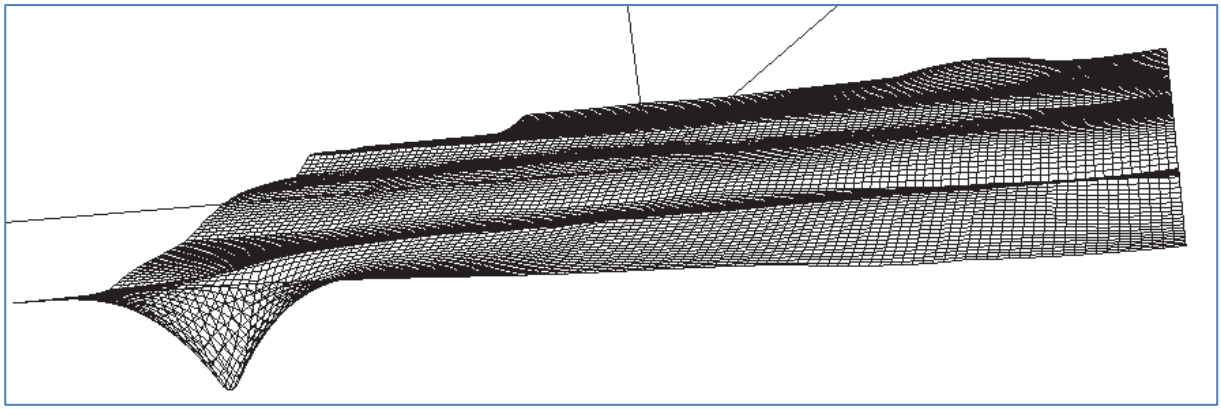


Abb. 11: 3D-Darstellung der Mittelwerte einer Messung

Um die Daten zu zeichnen, muss der „BezierSurface“-Klasse die 4x4 Arrays als ControllPoints übergeben werden. Zwischen je zwei angegebenen Punkten interpoliert die Klasse zwei weitere Punkte und zeichnet anschließend das so entstandene Netz.

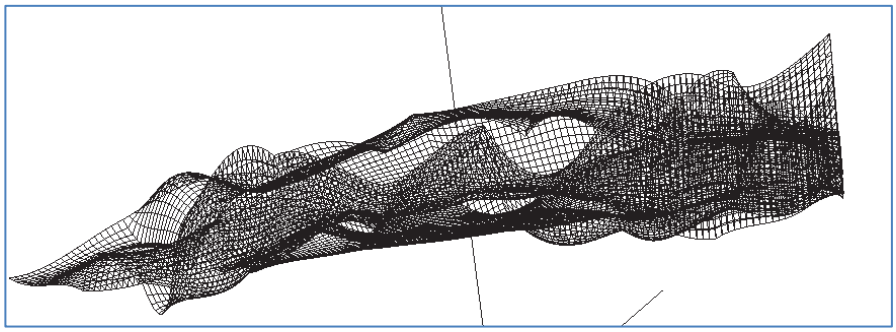


Abb. 12: 3D-Darstellung der Mittelwerte mit Maxima und Minima

Es werden immer kleine Netze gezeichnet, die sich überlappen und dadurch ein großes Netz bilden. Abb. 11 zeigt ein solches Netz, bei dem die Mittelwerte dargestellt werden. Dieses kann man mit Hilfe der Eigenschaft des verwendeten OpenGL-Devices „Navigate“ rotieren, heranzoomen und verschieben. Durch die zusätzliche Interpolation der „BezierSurface“-Klasse wird das Bild glatter. In Abb. 12 sind sowohl die Mittelwerte, als auch die Maxima und Minima der jeweiligen Position dargestellt. Diese Darstellungen sind noch ohne Farbzuweisung gezeichnet.
Damit man zwischen der Darstellung eines Gitternetzes und einer durchgehenden Ebene wechseln kann, sind zwei Radiobuttons vorhanden, die zwischen den beiden Modi hin- und herwechseln.

3.6 Farben

Abhängig von der Größe eines Wertes sollte dem Messpunkt eine Farbe zugewiesen werden. Der Ansatz ist:

Es ist möglich eine Textur zu übergeben, die über das Gezeichnete gelegt wird. Da es möglich ist mit C# selbst eine Bitmap zu erstellen, sollten so die Farben zugewiesen werden.

Damit nicht nur die Linien, sondern das gesamte Netz, also auch die Leerräume, mit der Textur bedeckt sind, muss der Benutzer den Radiobutton „Fillmode“ wählen. Sollte das der Fall sein, wird bei dem Erstellen der 4x4 Arrays zusätzlich für jedes dieser Arrays ein Bitmap erzeugt, welches in einer Liste (Farbe) gespeichert wird.

Das Bitmap lässt sich ähnlich wie das Netz mit der Klasse „Graphics“ erzeugen. Die Farbe wird durch eine eigene Klasse (Palette) berechnet. Dieser Klasse muss eine Farbskalar übergeben werden und sie berechnet dann aus einem gegebenen Wert die dazugehörige Farbe.

Dieses Erstellen der Bitmap befindet sich ebenfalls in der for-Schleife, in der auch die Arrays erstellt werden, weshalb dieser Prozess ebenfalls viermal Durchlaufen wird. Dadurch wird für jedes einzelne Array ein entsprechendes Bild gezeichnet. Auch hier werden zuerst die Punkte in x-Richtung bestimmt.

Das Zeichnen der Netze funktioniert grundsätzlich noch immer gleich. Allerdings werden jetzt nicht nur die ControllPoints, sondern auch die Bitmaps der zugehörigen Arrays werden einer Textur (Farben) übergeben. Die Eigenschaft des Devices „texture“ bekommt dann Farben zugewiesen und wird über das aktuelle Netz gelegt und anschließend gezeichnet.

Allerdings scheint es bei diesem Ansatz einen Fehler zugeben, da das Programm sich bei dem Versuch, die Bitmaps der Texturen zu übergeben aufhängt. Nach mehreren Tests habe ich festgestellt, dass die Zeichnenfunktion („Onpaint“) sehr viel Speicher benötigt (noch mehr, wenn auch farben zugewiesen werden sollen), den bereits andere Prozesse einnehmen, weil sehr viele Daten verarbeitet werden müssen.

4. Benutzeroberfläche

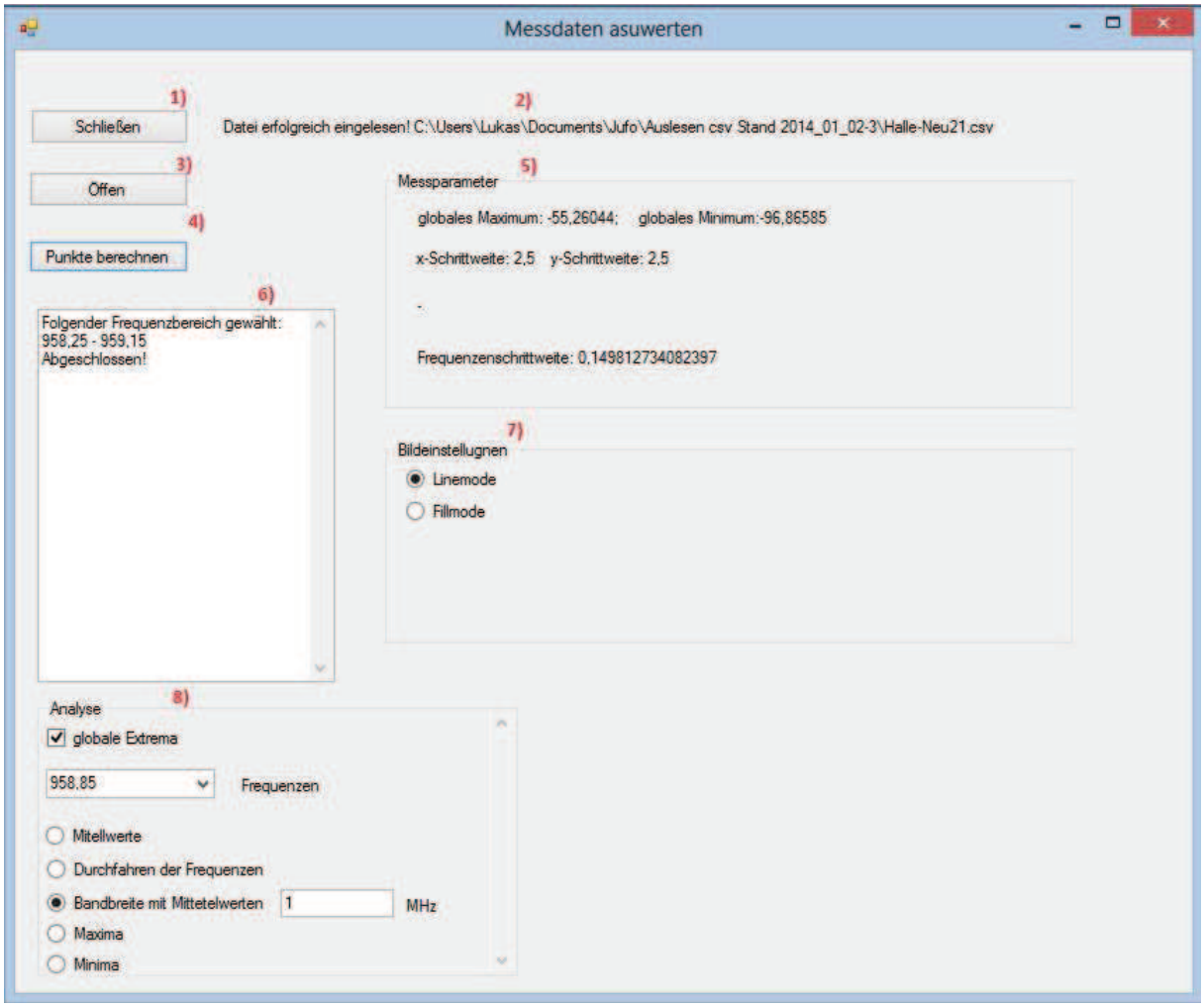


Abb. 13: Benutzeroberfläche

Abb. 13 zeigt die Benutzeroberfläche. Sie sollte möglichst benutzerfreundlich und selbsterklärend sein.

1) Mit dem Button „Schließen“ lässt sich das Programm beenden. Damit lässt sich sowohl das aktuelle, als auch das Fenster der Graphik schließen und alle Hintergrundprozesse werden beendet.

2) Hier wird ausgegeben, welche Datei gelesen wurde und, ob das Einlesen erfolgreich war.

3) Mit dem „Öffnen“-Button lassen sich die „.csv“-Dateien öffnen.

4) Der Button „Punkte berechnen“ führt die Funktion „calcPart“ aus (Datenaufbereitung, s. 3.4).

5) In dieser GroupBox werden die berechneten Größen, wie z.B. das globale Maxima und Minima, die Frequenzschrittweite oder die x- und y-Schrittweite ausgegeben. Außerdem wird ein zweites Fenster geöffnet, in dem die Bildausgabe stattfindet.

6) In diesem Abschnitt werden Meldungen, wie z.B. der aktuelle Stand laufender Prozesse, ausgegeben.

7) Hier sind alle Möglichkeiten zur Einstellung von Darstellungsoptionen aufgelistet.

8) In der GroupBox „Analyse“ sind die Möglichkeiten der Analyse der Messdaten aufgelistet. Hier kann eingestellt werden, ob die Mittelwerte, Werte bestimmter Frequenzen, Maxima und Minima der Positionen oder Mittelwerte von Frequenzen mit gewählter Bandbreite angezeigt werden sollen.

5. Erfahrung mit dem Programm

Mit Hilfe meines Programmes bin ich nun in der Lage, verschiedene Messungen graphisch darzustellen. Ich habe jedoch festgestellt, dass das Programm relativ lang zum Zeichen bzw. Rotieren und Zoomen braucht, was das Arbeiten damit erschwert. Nach mehreren Tests habe ich festgestellt, dass die Zeichenfunktion („Onpaint“) sehr viel Speicher benötigt (noch mehr, wenn auch Farben zugewiesen werden sollen), den bereits andere Prozesse einnehmen, weil sehr viele Daten verarbeitet werden müssen. Das könnte daran liegen, dass die Bibliothek nicht für so große Datenmengen ausgelegt ist. Ein weiteres Indiz dafür ist, dass kein Beispielprogramm mit so vielen Daten arbeitet. Deshalb stößt der Computer schnell an seine Grenzen.

6. Fazit

Ich konnte einen Großteil der Anforderungen an das Programm erfüllen: Es ist in der Lage „.csv“-Dateien ohne zusätzliche Konvertierung einlesen und sie in einer 3D-Darstellung visualisieren. Zudem können verschiedene Analysemethoden verwendet werden: es können Mittelwerte, Werte bestimmter Frequenzen, Mittelwerte wählbarer Frequenzbereiche und Maxima und Minima der einzelnen Positionen angezeigt werden. Zudem ist das Programm in der Lage, das globale Maximum und das globale Minimum zu bestimmen.
Es ist zwar auch prinzipiell in der Lage, verschiedenen empfangenen Leistungen eine Farbe zuzuweisen, bzw. diese darzustellen, jedoch dauert dieser Prozess sehr lang.
Des Weiteren konnte ich feststellen, dass die Bibliothek nicht für so viele Daten ausgelegt ist, weil der Computer bei großen Datenmengen schnell an seine Grenzen stößt. Ein weiteres Indiz dafür ist, dass auch bei den Beispielprogrammen nur mit wenigen Daten gearbeitet wird.

7. Ausblick

Das Programm kann noch weiter verbessert werden: es wäre z.B. hilfreich, wenn man in der Lage wäre, von der Benutzeroberfläche die „Kamera“ (die Richtung aus der der Beobachter die Daten sieht) zu steuern, also direkt Positionen über z.B. eine Textbox angeben. Zudem ist das Programm noch recht langsam, wodurch das Finden der richtigen Position erschwert wird. Dieses Problem könnte z.B. dadurch behoben werden, dass mit Klicken in das Bild sich erst nur die Achsen drehen können, was deutlich schneller gehen würde. Hat man seine Position gefunden, so wird der Rest dementsprechend gedreht.
Zudem hoffe ich, dass ich bis zum Wettbewerb in der Lage bin, die Messdaten auch mit den entsprechenden Farben richtig darstellen zu können.

8. Danksagung

Besonderer Dank gilt meinen beiden Betreuern Julia Bienert und Thomas Biedermann, ohne die dieses Projekt nicht möglich gewesen wäre, die mir mit Rat und Tat zur Seite standen und die mir geholfen haben in so kurzer Zeit in die Welt des Programmierens einzusteigen. Auch möchte ich mich bei Susanne Biedermann für die tolle Verpflegung bedanken, weshalb ich nicht verhungert bin.

9. Quellen

[1] „drawing3d“ http://drawing3d.de/Default.aspx
Tutorials und Beispiele von “drawing3d”
Microsoft Visual C# 2010 –Das Entwicklerbuch, Dirk Louis, Shinja Strasser, Thorsten Kansy

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using Drawing3d;
using Drawing3d.Math;
using Drawing3d.Devices;
using Drawing3d.Surfaces;
using Drawing3d.Editors;
using Drawing3d.Curves;

namespace Auslesen_csv
{
    public partial class Form1 : Form
    {
        public CurveEditor2d CurveEditor = new CurveEditor2d();
        public CoordinateAxes CoordinateAxes = new CoordinateAxes();
        double FStart;          //Messparameter
        double FStop;
        double FCount;
        double BandM;
        double BandG;
        double Sweepzeit;
        Double[] Zeiten;
        Double [ , ] Frequenzen;
        Data aktdata = new Data();
        Boolean Datain = false;
        double gMax;            //globales Maximum
        double gMin;            //globales Minimum
        double xStep = 0;       //x-Schrittweite
        double yStep = 0;       //y-Schrittweite
        double xCount;          //Anzahl der x-Werte
        double yCount;          //Anzahl der y-Werte
        xyz viewpoint = new xyz(0, 0, 0);   //Position der Kamera
        double x = 10;                          // Position der Achsen(bzw.Richtung)
        double y = 10;
        double z = 10;
        Boolean Dataon = false;
        int indexff;                                //Index der ersten Frequenz
        int gfreq;                                  //ausgewählte frequenz
        double Schrittweite;                        //der Frequenzen
        Boolean changed = false;                            //inhalt der bandbreitentextbox ändert sich
        Form Form2 = new Form();
        Texture Farbe = new Texture();
        List<Bitmap> Farben = new List<Bitmap>();



        BezierSurface bsf = new BezierSurface();        //BeizierSurface Initialisierung
        CustomSurface sf = new CustomSurface();
        Palette p = new Palette();

        struct Punkt
        {
            public xyz[,] part;
        }
        List<Punkt> Punkte = new List<Punkt>();
        Punkt tmp;

        struct Data                     // Struct für die Messdaten: Winkelkordinaten, Zeiten und
        {                               // zugehörige Messdaten
            public double Az;
            public double El;
            public double Max;
            public double Min;
            public double[] Times;
            public double[] Sweep;
        }
```



```
        List<Data> complMess=new List<Data>();
         Data[,] DataArray;


        CoordinateAxes Achse = new CoordinateAxes();
        Graphics gr = null;

        public OpenGlDevice Device = new OpenGlDevice();

        public Form1()
        {
            InitializeComponent();
        }

        private void cmdClose_Click(object sender, EventArgs e)
        {
            this.Close();
            Form2.Close();
        }

        private void cmdOpen_Click(object sender, EventArgs e)            // Algorithmus zum Einlesen der Daten
        {
            int i;
            string tmp;
            string tmp2;
            double freq;
            double time;
            double k;                                                   // Korrigierter Index
            Boolean FirstPoint=true;                        // Information über Beginn des Datenabsatzes


            OpenDlg.Filter = "Csv-Datei |*.csv";
            OpenDlg.Title = "csv Datei öffnen";

            if (OpenDlg.ShowDialog()==DialogResult.OK)
            {
                StreamReader Re = new StreamReader (OpenDlg.FileName);

                tmp = Re.ReadLine();
                tmp = tmp.Substring(tmp.IndexOf(";") + 1);
                FStart =Convert.ToDouble(tmp);                              // Einlesen der Paremeter

                tmp = Re.ReadLine();
                tmp = tmp.Substring(tmp.IndexOf(";") + 1);
                FStop = Convert.ToDouble(tmp);

                tmp = Re.ReadLine();
                tmp = tmp.Substring(tmp.IndexOf(";") + 1);
                FCount = Convert.ToDouble(tmp);

                tmp = Re.ReadLine();
                tmp = tmp.Substring(tmp.IndexOf(";") + 1);
                BandM = Convert.ToDouble(tmp);

                tmp = Re.ReadLine();
                tmp = tmp.Substring(tmp.IndexOf(";") + 1);  // BandG = bandbreite gegeben, BandM = bandbreite Messung
                tmp = tmp.Substring(0, tmp.IndexOf("k"));      //lies bis "k" ein
                BandG = Convert.ToDouble(tmp);

                tmp = Re.ReadLine();
                tmp = tmp.Substring(tmp.IndexOf(";") + 1);
                Sweepzeit = Convert.ToDouble(tmp);

                Schrittweite = (FStop - FStart) / FCount;          // Schrittweite der Frequenzen bestimmen
                txtshrittw.Text = "Frequenzenschrittweite: " + Convert.ToString(Schrittweite);

                tmp = Re.ReadLine();                                // Zeiten einlesen
                Zeiten = new Double[(int)FCount];
                tmp = tmp.Substring(tmp.IndexOf(";") + 2);

                for (i = 0; i < FCount; i++)
```

```
146             {
147                 tmp2= tmp.Substring(0, tmp.IndexOf(";"));
148                 time = Convert.ToDouble(tmp2);
149                 Zeiten[i] = time;
150                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
151 
152             }
153 
154             tmp = Re.ReadLine();                                    // Frequenzen einlesen
155             Frequenzen = new Double[2,(int)FCount];                 // Größe von Array festlegen
156             tmp = tmp.Substring(tmp.IndexOf(";") + 2);
157 
158             for (i = 0; i < FCount; i++)
159             {
160                 tmp2 = tmp.Substring(0, tmp.IndexOf(";"));              //Einlesen der Frequenzen,
161                 freq = Convert.ToDouble(tmp2);
162                 freq = freq / 1000;
163                 k = (freq - FStart) / Schrittweite;                   // korrigierten Index
    bestimmen
164                 Frequenzen[0, i] = freq;                              // in Array schreiben
165                 Frequenzen[1, i] = Math.Round(k,MidpointRounding.AwayFromZero);
166                 tmp = tmp.Substring(tmp.IndexOf(";") + 1);
167 
168                 if (Frequenzen[1, i] == 0)
169                 {
170                     indexff = i;
171                 }
172             }
173 
174             int t =indexff +1;
175 
176             while ( t!= indexff)
177             {
178                 combofreq.Items.Add(Convert.ToString(Frequenzen[0, t])); //Combobox mit Frequenzen
    in richtiger
179                 t++;                                                    //Reihnfolge füllen
180 
181                 if (t == Frequenzen.Length/2)
182                 {
183                     t = 0;
184                 }
185 
186             }
187 
188 
189 
190             Data tmpData = new Data();   // Zwischenspeicher vom Typ Data fürs Einlesen der
    Messdaten
191             int l;
192             int corIndex;
193             double tmpMax;
194             double tmpMin;
195             double x1;
196             double y1;
197             Boolean notendoflist = true;
198             double xMax = -1000;
199             double yMax = -1000;
200             double xMin = 1000;
201             double yMin = 1000;
202 
203 
204             while (!Re.EndOfStream)
205             {
206 
207                 tmp = Re.ReadLine();                    // Keywords stehen immer in einzelner Zeile
208 
209                 switch (tmp)
210                 {
211                     case "Zeiten:":
212                         l = 0;
213 
214                         // Zeiten markiert den Beginn eines neuen Datenpunktes --> Vorherigen Punkt
    zunächst an Liste anfügen
215 
```



```
216                         if (!FirstPoint)
217                         {
218                             complMess.Add(tmpData);
219                         }
220 
221                         tmp = Re.ReadLine();                            // Datenzeile einlesen
222                         tmp = tmp.Substring(tmp.IndexOf(";")+1);          // Zeit in Klarschrift
    überspringen
223                         tmp = tmp.Substring(tmp.IndexOf(";") + 2);
224                         tmpData = new Data();
225                         tmpData.Times = new double[(int)FCount];    // Arrays von tmpData mit Größe
    FCount initalisieren
226                         tmpData.Sweep = new double[(int)FCount];
227 
228                         while (tmp != "")                               // Zeiten in Zeit-Array von
    tmpData schreiben
229                         {
230                             corIndex = (int) Frequenzen[1, l];
231                             tmpData.Times[corIndex] = Convert.ToDouble(tmp.Substring(0,tmp.IndexOf("
    ;")));
232                             tmp = tmp.Substring(tmp.IndexOf(";") + 1);
233                             l++;
234                         }
235 
236                         FirstPoint = false;
237 
238                         break;
239 
240                     case "Sweep:":
241                         l = 0;
242                         tmp = Re.ReadLine();                            // Datenzeile einlesen
243 
244                         tmpData.Az = Convert.ToDouble(tmp.Substring(0,tmp.IndexOf(";")));
245                         tmp = tmp.Substring(tmp.IndexOf(";") + 1);
246                         tmpData.El = Convert.ToDouble(tmp.Substring(0,tmp.IndexOf(";")));
247                         tmp = tmp.Substring(tmp.IndexOf(";") + 1);
248                         tmpMax = -10000;
249                         tmpMin = 10000;
250 
251                         while (tmp !="")
252                         {
253                             corIndex = (int)Frequenzen[1, l++];
254 
255                             if (Convert.ToDouble(tmp.Substring(0, tmp.IndexOf(";"))) > tmpMax)
    //Maximum Bestimmen
256                             {
257                                 tmpData.Max = corIndex;
258                                 tmpMax = Convert.ToDouble(tmp.Substring(0, tmp.IndexOf(";")));
259                             }
260 
261                             if (Convert.ToDouble(tmp.Substring(0, tmp.IndexOf(";"))) < tmpMin)
    //Minimum bestimmen
262                             {
263                                 tmpData.Min = corIndex;
264                                 tmpMin = Convert.ToDouble(tmp.Substring(0, tmp.IndexOf(";")));
265                             }
266 
267                             tmpData.Sweep[corIndex] = Convert.ToDouble(tmp.Substring(0, tmp.IndexOf
    (";")));
268                             tmp = tmp.Substring(tmp.IndexOf(";") + 1);
269                         }
270 
271                         break;
272 
273                 }
274             }
275 
276             Re.Close();
277             lblInfo.Text = "Datei erfolgreich eingelesen! " + OpenDlg.FileName;
278             Datain = true;
279             Dataon = true;
280             x1 = complMess[0].Az;
281             y1 = complMess[0].El;
282 
```

```
283
284
285             while (!(xStep != 0 && yStep != 0) && notendoflist)     // falls x step und y step nicht
     mehr null oder
286             {                                                       //ende der liste(az oder el
    ändern sich nicht)
287
288                 foreach (Data d in complMess)                       //bestimmen von x und y Step
289                 {
290                     if (d.Az != x1 && xStep == 0)
291                     {
292                         xStep = d.Az - x1;
293                     }
294
295                     if (d.El != y1 && yStep == 0)
296                     {
297                         yStep = d.El - y1;
298                     }
299                     if (complMess.IndexOf(d) == complMess.Count-1)
300                     {
301                         notendoflist = false;
302                     }
303                 }
304             }
305             if (xStep == 0)
306             {
307                 xStep = 1;
308                 txtLog.Text += Environment.NewLine;
309                 txtLog.Text += "Keine Änderung von x-Step";
310             }                                           //zur überprüfung und Standbildmessung
311             if (yStep == 0)                             //(x-Step bzw. y-Step darf nicht null
    werden)
312             {
313                 yStep = 1;
314                 txtLog.Text += Environment.NewLine;
315                 txtLog.Text += "Keine Änderung von y-Step";
316             }
317
318
319             txtStep.Text = "x-Schrittweite: " + Convert.ToString(xStep) + "    " + "y-Schrittweite:
    "+ Convert.ToString(yStep);
320
321
322             foreach (Data dp in complMess)  //Bestimmen von x bzw. yMax und Min zur späteren
    bestimmung der Datenarraygröße
323             {
324                 if (dp.Az > xMax)
325                 {
326                     xMax = dp.Az;
327                 }
328
329                 if (dp.Az < xMin)
330                 {
331                     xMin = dp.Az;
332                 }
333
334                 if (dp.El > yMax)
335                 {
336                     yMax = dp.El;
337                 }
338
339                 if (dp.El < yMin)
340                 {
341                     yMin = dp.El;
342                 }
343             }
344
345             xCount = (xMax -xMin)/xStep;   //Hilft beim Ermitteln der Position der gesuchten
    Messpunkte bei Darstellung
346             yCount = (yMax - yMin)/yStep;
347
348              DataArray = new Data[(int)xCount +1, (int)yCount +1];
349
350             foreach (Data dp in complMess)
```



```
351             {
352                 DataArray[(int)((dp.Az-xMin)/xStep), (int)((dp.El-yMin)/yStep)] = dp;
353             }
354
355
356 // ------------------------------------------------------------------------------
357         }
358     }
359
360     private void Form1_Load(object sender, EventArgs e)
361     {
362         Device.WinControl = Form2;                      //gibt "Ort" der Darstellung an
363         Form2.Height = 1000;                        //Länge und Breite von Form2
364         Form2.Width = 1000;
365         Device.Navigate = NavigateKind.ZoomRotateTrans;
366         Device.Lighting = false;                            // Licht wird ausgeschaltet
367         Device.BackColor = Color.White;                //Hintergrundfarbe
368         Device.PenColor = Color.Black;
369         Device.OnPaint += new EventHandler(Device_OnPaint);
370         Device.FieldOfView = 50;    //Sichtfeld
371
372
373
374
375
376     }
377
378     public void Device_OnPaint(object sender, EventArgs e)
379     {
380
381         Tools.drawArrow(Device, new xyz(1, 0, 0), new xyz(x, 0, 0), 0.4);   //Zeichnen der Pfeile
382         Device.drawLine(viewpoint, viewpoint+ new xyz(x, 0, 0));
383
384         Tools.drawArrow(Device, new xyz(0, 1, 0), new xyz(0, y, 0), 0.4);
385         Device.drawLine(viewpoint, viewpoint+ new xyz(0, y, 0));
386
387         Tools.drawArrow(Device, new xyz(0, 0, 1), new xyz(0, 0, z), 0.4);
388         Device.drawLine(viewpoint,  viewpoint +new xyz(0, 0, z));
389
390
391
392         if (Datain)     //wenn Daten eingelsen wurden
393         {
394             viewpoint = new xyz(DataArray[(int)(xCount / 2), (int)(yCount / 2)].Az, DataArray[(int)
    (xCount / 2), (int)(yCount / 2)].El, DataArray[(int)(xCount / 2), (int)(yCount / 2)].Sweep.Average
    ());
395             x = 100;        //Kamera neu psitoinieren und Pfeile anpassen
396             y = 100;
397             z = 100;
398
399             if (Dataon) //nur einmal kamereinstellung ändern
400             {
401                 Device.Camera.LookAt(new xyz(-10, -5, 1), viewpoint, new xyz(0, 0, -1));
402                 Device.Camera.SetRelativSystem();
403                 Dataon = false;
404             }
405             //Device.texture = Farbe;
406
407
408             foreach (Punkt p  in Punkte)
409             {
410
411                 if (radiofill.Checked)  //wenn im Fillmodus
412                 {
413                     Device.PushMatrix();
414                     Farbe.SetBitmap(Farben[Punkte.IndexOf(p)], Color.White);    //Textur zugehörige
415                     Device.texture = Farbe;                                     //farbe übergeben
416                     bsf.ControlPoints = p.part; //Controllpoints übergeben
417                     Device.drawSurface(bsf);    //Surface zeichnen
418                     Device.PopMatrix();
419                 }
420                  else
421                 {
422                     bsf.ControlPoints = p.part;
```

```
                        Device.drawSurface(bsf);
                    }
                }
            }


        }

        private void calcPart(Data[,] Data) //Bestimmen der 4x4 Arrays
        {
            int y = 0;
            int x = 0;
            tmp = new Punkt();
            tmp.part = new xyz[4, 4];
            Bitmap btp = new Bitmap(4, 4);  //zu erzeugendes Bild
            p.OpenPal(Application.StartupPath + "\\" + "Pal1.col"); //Palette, nach der Farben berechnet
            txtLog.Text += Environment.NewLine;                     //werden sollen
            txtLog.Text += "Berechne Punkte...";    //Information an Benutzer

            Farben.Clear();



            while (y < yCount - 3)
            {

                if (tmp.part[0, 0] != new xyz (0,0,0))  //wenn nicht am Anfang
                {
                    Punkte.Add(tmp);    //Liste Punkt hinzufügen


                }
                tmp = new Punkt();  //neuen Punkt initialisieren
                tmp.part = new xyz[4, 4];

                if (radioMittel.Checked)
                {
                    for (int f = 0; f < 4; f++)
                    {
                        tmp.part[0, f] = new xyz(Data[x, y + f].Az, Data[x, y + f].El, Data[x, y + f].
Sweep.Average());//erstellen der 4x4 Arrays
                        tmp.part[1, f] = new xyz(Data[x + 1, y + f].Az, Data[x + 1, y + f].El, Data[x +
1, y + f].Sweep.Average()); //für Mittelwerte
                        tmp.part[2, f] = new xyz(Data[x + 2, y + f].Az, Data[x + 2, y + f].El, Data[x +
2, y + f].Sweep.Average());
                        tmp.part[3, f] = new xyz(Data[x + 3, y + f].Az, Data[x + 3, y + f].El, Data[x +
3, y + f].Sweep.Average());

                            gr = Graphics.FromImage(btp);   //Bilder erzeugen
                            Pen stift = new Pen(p.GetCol(Data[x, y + f].Sweep.Average()));  //Farbe
bestimmen
                            gr.DrawRectangle(stift, 0, f, btp.Height / 4, btp.Height / 4);  //Position
und Größe
                            stift = new Pen(p.GetCol(Data[x + 1, y + f].Sweep.Average()));
                            gr.DrawRectangle(stift, 1, f, btp.Height / 4, btp.Height / 4);
                            stift = new Pen(p.GetCol(Data[x + 2, y + f].Sweep.Average()));
                            gr.DrawRectangle(stift, 2, f, btp.Height / 4, btp.Height / 4);
                            stift = new Pen(p.GetCol(Data[x + 3, y + f].Sweep.Average()));
                            gr.DrawRectangle(stift, 3, f, btp.Height / 4, btp.Height / 4);



                    }

                }

                if (Scrollfreq.Enabled)
                {
                    for (int f = 0; f < 4; f++) //4x4 Arrays für gewählte Frequenz bestimmen
                    {
                        tmp.part[0, f] = new xyz(Data[x, y + f].Az, Data[x, y + f].El, Data[x, y + f].
Sweep[Scrollfreq.Value]);
                        tmp.part[1, f] = new xyz(Data[x + 1, y + f].Az, Data[x + 1, y + f].El, Data[x +
1, y + f].Sweep[Scrollfreq.Value]);
```



```
                        tmp.part[2, f] = new xyz(Data[x + 2, y + f].Az, Data[x + 2, y + f].El, Data[x +
2, y + f].Sweep[Scrollfreq.Value]);
                        tmp.part[3, f] = new xyz(Data[x + 3, y + f].Az, Data[x + 3, y + f].El, Data[x +
3, y + f].Sweep[Scrollfreq.Value]);

                            gr = Graphics.FromImage(btp);
                            Pen stift = new Pen(p.GetCol(Data[x, y + f].Sweep[Scrollfreq.Value]));
                            gr.DrawRectangle(stift, 0, f, 1, 1);
                            stift = new Pen(p.GetCol(Data[x + 1, y + f].Sweep[Scrollfreq.Value]));
                            gr.DrawRectangle(stift, 1, f, 1, 1);
                            stift = new Pen(p.GetCol(Data[x + 2, y + f].Sweep[Scrollfreq.Value]));
                            gr.DrawRectangle(stift, 2, f, 1, 1);
                            stift = new Pen(p.GetCol(Data[x + 3, y + f].Sweep[Scrollfreq.Value]));
                            gr.DrawRectangle(stift, 3, f, 1, 1);
                    }
                }

                if (textBand.Enabled)   //wenn Bandtext entspeert wurde
                {
                    if (changed)        //und sich Inhalt ändert
                    {
                        for (int f = 0; f < 4; f++) //4x4 Arrays für Frequenzen mit bandbreite
                        {
                            tmp.part[0, f] = new xyz(Data[x, y + f].Az, Data[x, y + f].El, calcMittel
(Data[x , y + f].Sweep, gfreq, Convert.ToInt32(textBand.Text)));
                            tmp.part[1, f] = new xyz(Data[x + 1, y + f].Az, Data[x + 1, y + f].El,
calcMittel(Data[x + 1, y + f].Sweep, gfreq, Convert.ToInt32(textBand.Text)));
                            tmp.part[2, f] = new xyz(Data[x + 2, y + f].Az, Data[x + 2, y + f].El,
calcMittel(Data[x + 2, y + f].Sweep, gfreq, Convert.ToInt32(textBand.Text)));
                            tmp.part[3, f] = new xyz(Data[x + 3, y + f].Az, Data[x + 3, y + f].El,
calcMittel(Data[x + 3, y + f].Sweep, gfreq, Convert.ToInt32(textBand.Text)));

                                gr = Graphics.FromImage(btp);
                                Pen stift = new Pen(p.GetCol(calcMittel(Data[x, y + f].Sweep, gfreq,
Convert.ToInt32(textBand.Text))));
                                gr.DrawRectangle(stift, 0, f, 1, 1);
                                stift = new Pen(p.GetCol(calcMittel(Data[x + 1, y + f].Sweep, gfreq,
Convert.ToInt32(textBand.Text))));
                                gr.DrawRectangle(stift, 1, f, 1, 1);
                                stift = new Pen(p.GetCol(calcMittel(Data[x + 2, y + f].Sweep, gfreq,
Convert.ToInt32(textBand.Text))));
                                gr.DrawRectangle(stift, 2, f, 1, 1);
                                stift = new Pen(p.GetCol(calcMittel(Data[x + 3, y + f].Sweep, gfreq,
Convert.ToInt32(textBand.Text))));
                                gr.DrawRectangle(stift, 3, f, 1, 1);
                        }


                    }
                    if (changed == false)   //Stoppt Schleife
                    {
                        y = (int)FCount;
                    }
                }

                if (radiomax.Checked)   //Arrays für Betrachtung der Maxima
                {
                    for (int f = 0; f < 4; f++)
                    {
                        tmp.part[0, f] = new xyz(Data[x, y + f].Az, Data[x, y + f].El, Data[x, y + f].
Sweep[(int)Data[x, y +f].Max]);
                        tmp.part[1, f] = new xyz(Data[x + 1, y + f].Az, Data[x + 1, y + f].El, Data[x +
1, y + f].Sweep[(int)Data[x + 1, y + f].Max]);
                        tmp.part[2, f] = new xyz(Data[x + 2, y + f].Az, Data[x + 2, y + f].El, Data[x +
2, y + f].Sweep[(int)Data[x + 2, y + f].Max]);
                        tmp.part[3, f] = new xyz(Data[x + 3, y + f].Az, Data[x + 3, y + f].El, Data[x +
3, y + f].Sweep[(int)Data[x + 3, y + f].Max]);

                            gr = Graphics.FromImage(btp);
                            Pen stift = new Pen(p.GetCol(Data[x, y + f].Sweep[(int)Data[x, y + f].Max]))
;
                            gr.DrawRectangle(stift, 0, f, 1, 1);
                            stift = new Pen(p.GetCol(Data[x + 1, y + f].Sweep[(int)Data[x + 1, y + f].
Max]));
```

```
547                             gr.DrawRectangle(stift, 1, f, 1, 1);
548                             stift = new Pen(p.GetCol(Data[x + 2, y + f].Sweep[(int)Data[x + 2, y + f].
    Max]));
549                             gr.DrawRectangle(stift, 2, f, 1, 1);
550                             stift = new Pen(p.GetCol(Data[x + 3, y + f].Sweep[(int)Data[x + 3, y + f].
    Max]));
551                             gr.DrawRectangle(stift, 3, f, 1, 1);
552                         }
553                     }
554
555                     if (radiomin.Checked)   //Arrays für Betrachtung der Minima
556                     {
557                         for (int f = 0; f < 4; f++)
558                         {
559                             tmp.part[0, f] = new xyz(Data[x, y + f].Az, Data[x, y + f].El, Data[x, y + f].
    Sweep[(int)Data[x, y + f].Min]);
560                             tmp.part[1, f] = new xyz(Data[x + 1, y + f].Az, Data[x + 1, y + f].El, Data[x +
    1, y + f].Sweep[(int)Data[x + 1, y + f].Min]);
561                             tmp.part[2, f] = new xyz(Data[x + 2, y + f].Az, Data[x + 2, y + f].El, Data[x +
    2, y + f].Sweep[(int)Data[x + 2, y + f].Min]);
562                             tmp.part[3, f] = new xyz(Data[x + 3, y + f].Az, Data[x + 3, y + f].El, Data[x +
    3, y + f].Sweep[(int)Data[x + 3, y + f].Min]);
563
564                             if (radiofill.Checked)
565                             {
566                                 gr = Graphics.FromImage(btp);
567                                 Pen stift = new Pen(p.GetCol(Data[x, y + f].Sweep[(int)Data[x, y + f].Min]))
    ;
568                                 gr.DrawRectangle(stift, 0, f, 1, 1);
569                                 stift = new Pen(p.GetCol(Data[x + 1, y + f].Sweep[(int)Data[x + 1, y + f].
    Min]));
570                                 gr.DrawRectangle(stift, 1, f, 1, 1);
571                                 stift = new Pen(p.GetCol(Data[x + 2, y + f].Sweep[(int)Data[x + 2, y + f].
    Min]));
572                                 gr.DrawRectangle(stift, 2, f, 1, 1);
573                                 stift = new Pen(p.GetCol(Data[x + 3, y + f].Sweep[(int)Data[x + 3, y + f].
    Min]));
574                                 gr.DrawRectangle(stift, 3, f, 1, 1);
575                             }
576                         }
577                     }
578                     picbtp.Image = btp; //Bild in Testpicturebox anzeigen
579                     Farben.Add(btp);    //Bild der Liste hinzufügen
580                     x += 3;
581
582
583                     if (x > xCount - 3) //wenn Ender der x_Reihe erreicht
584                     {
585                         y += 3;
586                         x = 0;
587                     }
588
589
590                 }
591                 txtLog.Text += Environment.NewLine;
592                 txtLog.Text += "Abgeschlossen!";    //Information für Benutzer
593             }
594
595
596
597
598             private void cmd_draw_Click(object sender, EventArgs e) //Punkte bestimmen (Button)
599             {
600
601                 calcPart(DataArray);    //calcPart ausführen
602                 Form2.Show();           //zweites Fenster öffnen
603
604             }
605
606
607             private void checkgExtrema_CheckedChanged_1(object sender, EventArgs e)
608             {
609                 gMax = -100000;     //gMax und gMin setzen, damit sie zu Beginn gelich geändert werden
    müssen
```



```
610                 gMin = 100000;
611
612                 if (Datain)
613                 {
614                     if (checkgExtrema.Checked == true)
615                     {
616
617                         foreach (Data d in complMess)                           //Bestimmen des globalen
    Maximas bzw. Minimas
618                         {
619                             if (d.Sweep[(int)d.Max] > gMax)
620                             {
621                                 gMax = d.Sweep[(int)d.Max];
622                             }
623
624                             if (d.Sweep[(int)d.Min] < gMin)
625                             {
626                                 gMin = d.Sweep[(int)d.Min];
627                             }
628
629                         } //Werte ausgeben
630                         txtgExtrema.Text = "globales Maximum: " + Convert.ToString(gMax)+";" + "     " +
    "globales Minimum:" + Convert.ToString(gMin);
631                     }
632
633                     else
634                     {
635                         txtgExtrema.Text = "-";
636                     }
637
638                 }
639             }
640
641             private void combofreq_SelectedIndexChanged(object sender, EventArgs e)
642             {
643                 if (Datain)
644                 {
645                     gfreq = combofreq.SelectedIndex;    //Index der gewählten Frequenz bestimmen
646                     textBand.Enabled = true;            //Textfeld "Bandbreite" freischalten
647                     Device.Refresh();
648                 }
649
650             }
651
652             public double calcMittel(Double [] freq, int index, double band)    //Mittelwerte für Bereich
    bestimmen
653             {
654                 double mitlw =0;
655                 int g;
656
657                 g = (int)(band/Schrittweite)/2;
658
659                 if (g > index)  //Falls Bereich nicht auswählbar
660                 {
661                     txtLog.Text += Environment.NewLine;
662                     txtLog.Text += "Eingegebene bandbreite nicht möglich!";
663                     changed = false;
664
665                 }
666                 else
667                 {
668
669                     for (int i = 0; i < 2 * g - 1; i++)
670                     {
671                         mitlw += freq[index - g + i];
672                     }
673
674                     txtLog.Text ="Folgender Frequenzbereich gewählt: " + Convert.ToString(Frequenzen[0,
    index -g]) + " - " + Convert.ToString(Frequenzen[0, index + g]);
675
676                     mitlw = mitlw / (2 * g);
677                 }
678                 return mitlw;
679             }
```

```
680
681
682         private void radiofreq_CheckedChanged(object sender, EventArgs e)
683         {
684             if (radiofreq.Checked)
685             {
686                 Scrollfreq.Enabled = true;
687             }
688
689             else
690             {
691                 Scrollfreq.Enabled = false;
692             }
693             Device.Refresh();   //Zeichnung aktualisieren
694
695         }
696
697         private void radioMittel_CheckedChanged(object sender, EventArgs e)
698         {
699             Device.Refresh();
700         }
701
702         private void radioBand_CheckedChanged(object sender, EventArgs e)
703         {
704             if (radioBand.Checked)
705             {
706                 combofreq.Enabled = true;
707             }
708
709             else
710             {
711                 combofreq.Enabled = false;
712                 textBand.Enabled = false;
713             }
714
715             Device.Refresh();
716         }
717
718         private void Scrollfreq_Scroll(object sender, ScrollEventArgs e)
719         {
720             Scrollfreq.Maximum =(int)(FCount -1);
721             Scrollfreq.Minimum = 0;
722             Device.Refresh();
723             //gewählte Frequenz ausgeben
724             txtscrvalue.Text ="Gewählte Frequenz: " + Convert.ToString(Frequenzen[0, Scrollfreq.Value])
    + "MHz";
725         }
726
727         private void textBand_TextChanged(object sender, EventArgs e)
728         {
729             changed = true;
730         }
731
732         private void radioline_CheckedChanged(object sender, EventArgs e)
733         {
734             if (radioline.Checked)
735             {
736                 Device.PolygonMode = PolygonMode.Line; //Gitternetz zeichnen
737             }
738         }
739
740         private void radiofill_CheckedChanged(object sender, EventArgs e)
741         {
742             if (radiofill.Checked)
743             {
744                 Device.PolygonMode = PolygonMode.Fill; //ausgefüllte Ebene zeichnen
745             }
746         }
747
748         private void radiomax_CheckedChanged(object sender, EventArgs e)
749         {
750             calcPart(DataArray);
751             Device.Refresh();
752         }
```



```
753
754         private void radiomin_CheckedChanged(object sender, EventArgs e)
755         {
756             calcPart(DataArray);
757             Device.Refresh();
758         }
759
760
761     }
762 }
763
```