Projektbericht VL Virtuelle Realität

Hannes Bischoff

Inhaltsverzeichnis

Beleg

Motivation

Projektidee war eine App für ein Android Smartphone zu entwickeln die Röhrenglocken abbildet. Das Instrument der Röhrenglocken gehört zu den Idiophonen wird seit Ende des 19. Jahrhunderts auch in Orchestern verwendet. Um es zu spielen schlägt der Schlagwerker mit einem kleinen Hämmerchen gegen eines der Röhren. Diese sind so aufgehangen das diese frei schwingen können und dabei klingen. Die meisten Röhrenglocken bestehen aus 8, 12 oder 20 Röhren. Ziel diese Projektes ist es eine einfache, günstige Simulation dieses Spielverhaltens zu erzeugen. Die verschiedenen Sensoren sollen dabei als Trigger dienen. Imitiert man mit dem Smartphone einen Schlag, so soll analog zum anschlag mit dem Hammer ein Ton erklingen. Wird eine andere Röhre angeschlagen ändert sich die Tonhöhe, auch das soll abgebildet werden. Das wird so gelöst das abhängig von der Richtung in der das Smartphone beim Schlag gehalten wird sich die Tonhöhe ändert.

Ein Schlagwerker an den Röhrenglocken 1917
https://de.wikipedia.org/wiki/R%C3%B6hrenglocken

Technologie

Grundlagen

Zentrales Element der Simulation ist das Smartphone. Dank seiner kompakten größe, handlichkeit und der eingebauten Sensorik lässt es sich vielseitig verwenden und ist hierfür ideal geeignet. Außerdem ist keine Anschaffung spezieller Hardware nötig wenn man handelsübliches Android Smartphone besitzt.

Die Wahl fiel auf die Android Plattform, da diese im Kern ein open-source source Betriebssystem ist und es unzählig freie Software und Tools für die Entwicklung von Apps gibt.

Der Programmcode dieses Projektes ist in Java geschrieben. Es wird als Entwicklungsumgebung das freie Android Studio mit den dazugehörigen Android Developer Kits genutzt. Layoutdateien und Design des Projektes sind androidtypisch in xml.

Um die Notwendigen und recht umfangreichen Vorarbeiten einer Android app zu verkürzen habe ich auf ein bereits existierendes Projekt mit offenem Quellcode zur auswertung der verschiedene Sensoren zurückgegriffen. Das Projekt Lemur auf github, schien dafür ideal geeignet zu sein. Es bedarf natürlich einiger Anpassungen aber verkürzte die Arbeitsaufwand immens. Layout, Grundeinstellungen, App interne Navigation und Berechtigungen systemseitig anzufragen können mitunter bei einer App, die von grund auf neu entwickelt wird mehrere Tage arbeitsaufwand bedeuten bevor auch nur eine Zeile des gewünschten Algorithmuses geschrieben werden kann.

Kernaufgaben

Von allem Überbau jeglicher Android Projekte abgesehen, lassen sich zwei Schwerpunkte herauskristallisieren.

Das verwendete Basisprojekt liest diverse Sensorwerte über die Systemschnittstellen aus und kann diese grafisch anschaulich plotten. Die grafische darstellung der werte ist besonders nützlich um Charakteristik und Wertebereich eines ereignisses zu erkennen.

Die meisten heutzutage verkauften Smartphones verfügen über mehrere eingebaute Bewegungssensoren. In der Android Entwicklerdokumentation https://developer.android.com/guide/topics/sensors/sensors_motion sind diese beschrieben. Einige Sensoren liefern physikalische Messwerte wie z.B. Beschleunigungssensor und Gyroskop. Andere Sensoren existiere nur virtuell. der Rotationsvektor oder Schrittzähler sind Softwareimplementierungen und nutzen die Daten der anderen Sensoren und kombinieren diese und vorverarbeiten die daten.

1. Erkennen des Anschlags

Dazu wird das Telefon in einer Kippbewegung um die Z-Achse bzw in der von der Y und X-Achsen aufgespannten Displayebene nach links bewegt. Dieses Ereignis kann mit den im Smartphone eingebauten Beschleunigungssensoren erfasst werden. Der Sensor liefert Werte für Beschleunigung in der X,Y, und Z-Achse. Um die Ausgelesenen Werte zu validieren genügt ein simpler Test. Steht das Gerät senkrecht auf mit der Unterkante auf einem Tisch und wird nicht

bewegt liefert der Sensor im idealfall für die Y-Achse 9,8m/s² entsprechend der lokalen Erdbeschleunigung und die Sensoren für die X und Z-Achse je 0 m/s².

Graphisch ist der Ausschlag sehr gut zu sehen. Im nächsten Schritt muss eine geeignete Methode entwickelt und als Algorithmus umgesetzt werden, um in der Zeitreihe das Muster zuverlässig zu erkennen. Das vom Sensor gelieferte Signal ist zeitdiskret mit konstanter Abtastzeit und wertediskret. Die Quantisierungsschritte sind sehr klein im Java Datentyp 'Double'.

Als praktikabel und zuverlässig hat sich folgende Methode herausgestellt.

Der Signalwert der X-Achse wird bei jeder Iteration mit dem Wert der vorherigen Iteration verglichen. Wenn der Aktuelle größer ist, die Werte also ansteigen und unterhalb der unteren Schranke liegen, wird Trigger 1 ausgelöst. Dieser Trigger öffnet ein 300ms langes Zeitfenster (grün). Wird Trigger 1 mehrfach ausgelöst spielt das keine Rolle, dieser überschreibt den Vorherigen. Der Trigger 1 ist notwendig aber noch nicht hinreichend um den Anschlag zu erkennen. Im geöffneten Zeitfenster, wird das Signal nun auf eine fallende Flanke untersucht. Sobald das Signal fällt also der vorherige Signalwert größer als der Aktuelle ist und sich die Werte oberhalb der oberen Schranke befinden, wird dies als Maximum detektiert. Die Festlegung der Schranken dienen dazu sehr kleine Ausschläge zu ignorieren.

2. Richtungserkennung

Mithilfe des virtuellen Richtungsvektor-Sensors ist eine Bestimmung der Orientierung des Gerätes im Raum möglich. Wird das smartphone gedreht und verharrt dann in dieser position wird dies trotzdem erkannt und es werden Winkel vom Gerät relativ zur angenommenen Erdoberfläche und der Magnetischen orientierung angegeben. Der Softwaresensor bezieht Daten aus verschiedenen Quellen wie Beschleunigungsmesser, Magnetfelddetektor und Gyroskop. Fehlt einer dieser Sensoren, fällt die Genauigkeit dementsprechend deutlich schlechter aus.

Für diese Simulation wird die X-Z-Ebene in 7 gleichgroße Felder zu je 51,4° eingeteilt, welche jeweils einen Ton repräsentieren. Je nachdem welchen Winkel der Sensor für die Drehung um die Y-Achse liefert wird der Ton gespielt.

Wird ein Maximum detektiert, wird abhängig davon welchen Sektor der aktuellen Richtungsvektor liefert, der dazugehörige Ton gespielt.

Sound

Sound leider war es mir nicht möglich eine Tonleiter auf echten Röhrenglocken auf zu nehmen. Auch eine qualitativ hochwertige frei verwendbare Audiodatei im Interneht habe ich nicht finden können. Deshalb habe ich mich entschieden statdessen die Tonleiter einer Klangschale zu

verwenden. Die verwendete stereo Audiodatei ist von der Website freesound.
https://freesound.org/people/mooncubedesign/sounds/347975/

Die Aufnahme der 12-Teiligen chromatischen Tonleiter geht in Halbtonschritten von $C\#^3$ bis $C\#^4$ und weist eine sehr gute Qualität auf. Die Frequenzanalyse offenbart, dass die Aufnahme frei von Rauschen ist und es einen markanten Grund- und mehrere harmonische Obertöne gibt. Dies erzeugt einen reinen, klaren Klang. Es tritt fast kein rauschen auf, wie im Spektrum zu sehen ist.

 image

Die Klangschale klingt im original für ca 12 Sekunden, was im ersten versuch für die App etwas störend war. In Audacity wurde das Audiosignal getrennt und die einzelnen Töne herausgeschnitten um jeden Ton in einer seperaten mp3-Datei abzuspeichern und in die App zu Integrieren. Außerdem wurde die Hüllkurve des Sounds so moduliert, dass der Ton ca nach 2,5 sek ausklingt ohne abrupt zu enden.

 image

In Versuchen habe ich herausgefunden, dass es schwierig ist die Richtug beim Schlag konstant zu halten. Dadurch ist es nicht sinnvoll möglich eine umdrehung von 360° in 12 Sektoren zu teilen. Deshalb sind in dieser Version leider nicht alle Töne der chromatischen Tonleiter untergebracht.

Die verwendeten Töne im aktuellen Projekt sind:
$c\#^3$, d^3, $d\#^3$, e^3, f^3, $f\#^3$, g^3.

7 Sektoren erwies sich als gerade noch praktikable Umsetzung um gezielt einen gewünschten Ton zu treffen und zu spielen. Eine weitere Unterteilung für mehr Töne könnte eventuell mit der unterscheidung ob ein Anschlag nach ober oder unten geschiet unterschieden werden können, nicht jedoch durch weitere sektoren.

App Layout

Die Activity mit der Röhrenglocken Simulation ist in das Konstrukt der zugrundeliegenden Basis-App integriert. Das Layout ist schlicht gehalten und zeigt je nach Orientierung des Smartphones an, welcher Ton gespielt würde, wenn das Gerät jetzt zum Schlag angesetzt wird. Dabei hilft zusätzlich eine farbliche Unterstützung in verschiedenen Orangetönen. Die Gradangabe ist rein informativer Natur und war während des Entwicklungsprozesses unerlässlich. Gleiches gilt für das Diagramm im unteren Bereich, es dient der Information und Veranschaulichung der Beschleunigungskräfte. Mitunter können Beschleunigungen von 3 bis 4 g gemessen werden.

 image

Ausblick

Das rotationssymmetrische System erweist sich mitunter als unpraktisch, da zum Teil große drehbewegungen nötig sind, um von einem zum andern Ton zu wechseln. In einer weiteren

Version könnten mehr Töne z.B. durch Einführung einer virtuellen Ebene geschaffen werden (wie anschlag nach oben, anschlag nach unten). Auch denkbar wären variable Tonlängen, abhängig davon wie schnell Smartphone nach dem Schlag zurück gezogen wird. In diesem Atemzug wäre auch denkbar das Tonsignal direkt auf dem Gerät unter Einbeziehung der Sensorwerte zu syntehisieren anstatt ein fertiges Audiofile abzuspielen.

Dies bedarf größerer Änderungen und wäre zum Beispiel eine Idee für eine Projektarbei im nächsten Jahr.

Referenz

Projektverzeichnis der Simulation auf Github.

Hier kann der Gesamte Quellcode eingesehen werden und auch reproduziert werden:
https://github.com/hablix/Lemur

Quellen

Basis projekt welches die grundlage für die simulation ist auf Github:
https://github.com/Roslund/Lemur

verwendeter soundfile:
https://freesound.org/people/mooncubedesign/sounds/347975/

Entwicklerdoku Senroren:
https://developer.android.com/guide/topics/sensors/sensors_motion

Entwicklerdoku Kordinatensystem Sensoren:
https://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords

Entwicklingsumgebung Android studio:
https://developer.android.com/studio

Audio editor Audacity:
https://www.audacityteam.org/

Bildnachweis

Achsen smartphone:
https://jp.mathworks.com/help/supportpkg/android/ref/gyroscope.html

Röhrenglocken:
https://de.wikipedia.org/wiki/R%C3%B6hrenglocken#/media/Datei:Chimes_(IMSO_pp55).jpg