KARLSTAD UNIVERSITET

beatr

Real time, interactive Android app for creation and composition of electronic music.

Kristian Sällberg

8/8/2012

This document contains technical information and background reasons concerning the beatr project.

Index

Introduction

First of all, I would like to thank the University of Karlstad for providing this very open ended and free of constrains summer course. And I would like to thank Martin for accepting my request to make a sound application even though I didn't know anything about the field. I have a learned a lot this summer.

I have been interested in creating some kind of music generation tool for a long time. My personal background is in professional Action Script development. I have extensive experience in programming user interfaces. In school, Computer Science at University of Gothenburg, I have studied java in many courses. Before this project started, I knew nothing about sound programming. I realized early on that I would not have time to create the actual sound generating (synth) code myself. I needed to find a software synthesizer that would work out of the box on the platform of choice.

At first, I wanted to build the tool in C++, using an open source framework such as openFrameworks or cinder, building an old school desktop application. But roughly a week before I started programming, it appeared to me that a mobile app would be very interesting for a few reasons. Mainly, because the course is short, seven weeks of programming is not a lot of time compared to the years of development time that have been invested by the major desktop music application creators. Another reason is that now when everyone has a smart phone in their pocket all day long, it makes sense to provide some kind of simple music application for quickly prototyping beats or just kill time. Yet another reason is that smart phones usually support multi touch, making the application feel more like a real instrument. On top of this, there are not many real time music generation apps for the Android platform; this point will be further deliberated later on in this document.

I moved to Berlin to work on this project. Berlin is a very entrepreneurial city and a technological hotspot. It houses many companies focused on sound development.

Background

For a long time, Google has done nothing at all to satisfy the developers community's desire to gain control of the sound card though the SDK. In 2010, Google embedded a library called SoniVox Jetcreator to make it easier to control and schedule sound interactively. With that API developers can schedule what midi files or slices of midi files to play and what to play after that. But it has (to the time of writing this, at least before API version 4.1 which is brand new) never been possible to feed sound data that you have generated yourself to the soundcard through the SDK.

It is possible to write data to a stream that sends it to the sound card. Using the NDK (Native Development Kit), a few of the existing software synthesizers, open source libraries that have been developed over the course of many years, have ported their C libraries to work with the Android platform. They have successfully managed to connect Java with their C synth code.

These solutions are not very easy to find, and when you find them, they are hard to compile and hard to get running within the Android framework. It took me a good week of searching through music forums and even synth forums before finding a couple of Android ports of existing software synths that did not lag too much and that weren't too awkward to compile.

The combination of Android being an awkward platform to develop sound apps for and it being reasonably difficult for developers to find the ported synths has scared away most developers from this segment. Many sound developers have chosen to make their apps in the IOS platform instead. Even major players like Propellerhead (creator of Reason, Tractor etc) have chosen not to make an Android version of their IOS hit Propellerhead Figure. Their CEO stating the screen size fragmentation of the android ecosystem as a major obstacle for creating good apps.

Who are then left in the segment? There are a couple of developers who made the move and created interactive sound generation apps for the Android platform. I have been in touch with the creator of the most popular Android music generation app, Caustic. Caustic does not officially support the low end Android devices because the app is too advanced for Processors like the AMV6 ???????????? Caustic uses it's own synth code. It's specifically tailored and optimized for the Android platform. I've also been in touch with a computer science professor who successfully compiled fluid synth to Android. He has not yet published an app based on the synth to the market.

All of the problems stated above combined, leave the market for real time, software synth backed, sound generating apps for low-end devices pretty much empty. And that's what I wanted to change with beatr.

Overview

The application is basically a collection of instruments playing simultaneously. The user can add any amount of instruments. There are no restrictions as to how many instances of an instrument users can add, because different android devices will be able to handle different amounts of instrument instances without performance issues and there is no way I can know what each and every device can handle.

Beatr has two modes. The overview mode, which shows all existing instrument instances in a list, is what users first see when opening the app. The amount of instruments that fit into one window vary depending on what device the user is running beatr on. In the bottom of this view, there are arrows that allow the user to see the next or previous n amount of instrument instances. Then, every instrument instance has an edit mode, which will allow the user to edit sound settings and record what they want the instrument to be playing.

Beatr is focused on creating simple repetitive beats very quickly without any required knowledge of music and composition. To make it easier to control what sounds play when, I decided to abandon the timeline concept that's an integral part of almost every desktop sound production tool. Instead of a timeline, it is possible to control what instrument plays when by changing the volume bar that's connected to every instrument.

It has a metronome that the user can specify to dispatch "tick" events in the ranges 80 to 200 beats per minute. All drum instances are then synched to the metronome, they make their drum sounds when the metronome dispatches its event.

There are three instrument types in the app; the bass and the synth are essentially the same instrument. But the bass is restricted to low keys and the synth starts from roughly where the bass ends and then continues to higher pitches. They both have a master volume property and individual volume properties as well as a property to control the frequency of the oscillator,

the measure of how often the curve turns from low to high in the speaker (all these musical words are totally new to me and I've picked them up during the course).

The third instrument is completely independent of the synth and bass. It has seven settings, on top of individual volume and master volume. The root property controls the pitch of the drum; it's possible to make it sound like a high-hat and also like a bass drum. F01 and F02 control ??????. Clip controls ??????. Shape controls ????? Decay controls the length of the sound duration. Mod controls ???? The drum also features nine drum pads. They can be pressed to be set to active and pressed again to be set to inactive. They are toggled between by the metronome, 1-9. If they are active when focused by the metronome, the drum plays its sound according to all the settings.

The user is able to record the synth and bass instruments, when entering the edit mode, only the current instrument will on, after recording and going back to the overview view, all

Detailed Description

Applikationen använder sig för nuvarande av ett mycket enkelt ramverk för MVC-modellen.

aMVC:

En vän har sammanställd detta MVC-ramverk som jag använder i min applikation. Det sätter controllern i centrum, controllern skapar views och models.

Views fungerar på så sätt att de är en samling som kan ta upp många andra nativa android-views i en aMVC-View. Detta för att man ibland i android behöver kunna nestla views i varandra för att uppnå vissa utseenden, och det skulle vara mycket opraktiskt att behöva ha en View för varje Layout i appen t.ex.

Model håller som vanligt på data som presenteras i vyn.

Pure Data:

Pure Data är en software synthesizer som finns tillgänglig i många olika plattformar. Den finns även portad till androidplattformen och det är alltså denna jag kommer använda. Man kan skapa instrument med "grafisk programmering" och sedan använda dessa instrument, och ändra parametrar i realtid för att kunna få en interaktiv upplevelse.

Pure data har inte stöd för att kunna ändra namn på objekt från kommandon från synthen, lösningen är att ändra namnen i .pd-filerna innan dom skickas in i pure data.

Conclusions/Result

I did not know about any of these limitations when I picked my project for this course. If I had – I would probably have chosen to do something else, or at least to do it in another platform, probably IOS.

The move to Berlin was a risky bet as I had nowhere to live and nowhere to work. But after having fixed all the everyday issues such as a place to live and an office space, it turned out Berlin gave a lot of inspiration back. Weekly programming meetups for a multitude of programming languages and countless techno clubs gave inspiration I hope will last for a long

time after this course has ended. I also made new friends and business contacts I hope will prove valuable in the future.

List of References

Referens till Propellerheads CEO som säger att dom inte ska utveckla för android

Bilagor

Här vill jag t.ex. ha in bilderna på mitt fysiska anteckningsblock för projektet

[Se beatr/docs/design/beatr.png]

Systemskiss över arkitekturen