ボイスレコーダーの開発

奥野　悠太

１．研究概要

Unity と呼ばれる統合型開発環境を用いて,C#というプログラミング言語とJavaScript というスクリプト言語を用いてプログラミングをする。それにより音の取り扱いの基本であるWAVEファイルをA/D変換し,録音・再生をする。

また、取り込んだオーディオデータをフーリエ変換とよばれるもので解析した後、解析したオーディオデータを波形表示させる。

２．研究の具体的内容

研究の流れを次に示す。

A) Unityのインストール
B) Unity を使い,JavaScript 言語,C#言語でプログラミングをする。
C) マイクから音声信号（声）を入力して音声が正しく入力されているか確認する。
D) 音声信号を入力→録音できるまでデバッグ。
E) アプリケーションとして出力する。

（１）Unityについて

今回,このボイスレコーダーを開発するにあたって,Unityと呼ばれる統合開発環境を用いたが,これは厳密には「ゲーム開発エンジン」である。Unityには,ゲーム制作における重要な要素である「音」を取り扱うライブラリが充実していたため,アプリケーション制作に応用できないかと思い使用した。

WAVEファイルを読み込むと,オーディオクリップと呼ばれるコンポーネントが作成され,オーディオデータを簡単に使用することができる。このオーディオクリップを用いてプログラムしていくことで比較的簡単にオーディオデータを取り扱うことができる。

（２）Unityでのプログラミングについて

Unityでのプログラミングは,まずアプリ内でのボタンなどを「オブジェクト」(GameObject)というものに置き換えてプログラムを組むという考え方である。GameObjectにどのような動きをさせたいか,ボタン(オブジェクト)を押すとどう動くのかをGameObjectに持たせることができるのが「Component」である。

Unityの場合Componentの動作は「C#」「JavaScript」などのプログラミング言語で記述することができる。

今回は「C#」を用いてプログラミングを記述した。

（３）WAVEファイルの取り扱い

WAVEまたはWAVファイルとは,マイクロソフト社とIBM社によって開発されたWindows標準のオーディオデータのファイル形式である。

WAVEファイルそのものは,RIFFチャンクと呼ばれるデータの集合体で構成されている。

RIFFチャンクにはサンプリングレートなどの情報を格納するfmt（フォーマット）チャンクと,実際の波形データを格納しておくdata(波形データ)チャンクとよばれるものがある。

WAVEファイルでは,これらのチャンクの中身を読み込むことでデータとして扱えるようになる。表１にWAVEファイルの構造を示す。

表１ WAVE ファイルの構造

バイト数	内容	説明
4	'R' 'I' 'F' 'F'	RIFFヘッダー
4	これ以降のサイズ	
4	'W' 'A' 'V' 'E'	
ｆｍｔチャンク		
4	'f' 'm' 't' ' '	fmtチャンク
4	チャンクサイズ	
2	フォーマットID	
2	チャンネル数	
4	サンプリングレート	Hz単位
4	データ速度	バイト/秒
2	ブロックサイズ	バイト/サンプル×チャンネル
2	ビットレート	ビット/サンプル
2	拡張領域のサイズ	
n	拡張領域	
ｄａｔａチャンク		
4	'd' 'a' 't' 'a'	dataチャンク
4	チャンクサイズ	
n	波形情報	

WAVE ファイルからチャンネル数とサンプリングレート,ビットレート,ブロックサイズを読みだすことで扱えるようになる。

ステレオの場合は左右のチャンネルに分かれるのでチャンネル数が２となる。

（４）マイクでの録音・再生について

マイクでオーディオデータ（声）を入力する際にはオーディオデバイスとよばれる,オーディオ機器の入出力を制御するソフトウェアの情報を,取得する必要がある。その後オーディオ入力デバイスをオープンし,マイクから入力するオーディオデータを格納するデータブロックを使用できるようにする。マイク入力が開始されるとキュー（図１に構造を示す）に入力データをフレーム単位で格納していく。格納されたオーディオデータをサンプリング（A/D 変換）し,メモリに書き込んでいくことで録音ができる。

再生では「出力バッファ」を用意し,先ほど入力データを格納したメモリからオーディオデータを,出力バッファに格納していくことで再生できる。

バッファは,1 つの波形の再生が終われば,現在の再生を中止して新しい再生を繰り返したりはしない。

そこで図１のように,キューが空になる前に新しいバッファを次々と書き込んでいけば,途切れることのない音を再生できる。

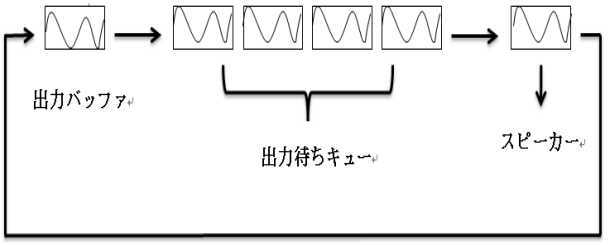

図１　再生処理

次に,録音・再生処理のプログラムを示す。

```
Microphone.Start (null, false, 10, 44100);
                yield return new
//デフォルトのデバイスで録音を開始する。
WaitForSeconds (10.0f);
//指定した秒数待つ。
        }
void EndRecord(){
        Microphone.End (null);
        //録音を停止
        StopCoroutine("StartRecord");
        //コルーチンを終了
}
IEnumerator PlayClip(){
        if(audio1.clip.length > 0){
                audio1.Play();
                while(true){
        audio1.GetSpectrumData(data,0,FFTW
indow.BlackmanHarris);
```

（５）波形表示について

波形表示をさせるには,入力されたオーディオデータの周波数成分を取得する必要がある。周波数とは,１秒間に繰り返される波のことである。この周波数成分の解析に欠かせないのが FFT（高速フーリエ変換）である。この高速フーリエ変換を行うと,オーディオデータの横軸である「時間」が「周波数」へと変換される。

普通,波形を見ただけで周波数を観測することは難しいと思われる。だが,このフーリエ変換を用いると「この波形には〇Hz と〇Hz と・・・〇Hz の波が含まれている」ということを解析することができる。

50Hz の波形グラフを図２に示す。
このグラフだと,１秒間にいくつ波が繰り返されているか数えるのは難しい。

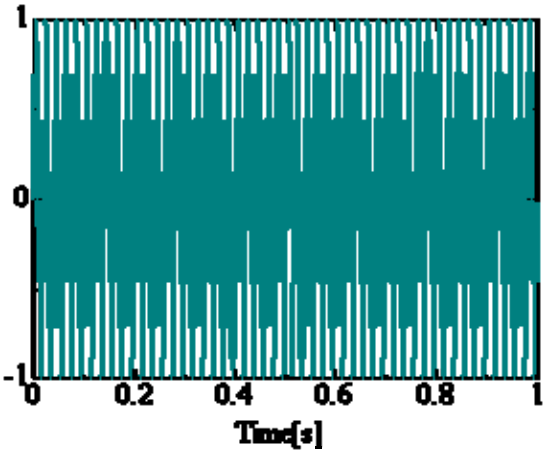

図２　波形グラフ

しかし,このグラフをフーリエ変換すると,図３ようになる。

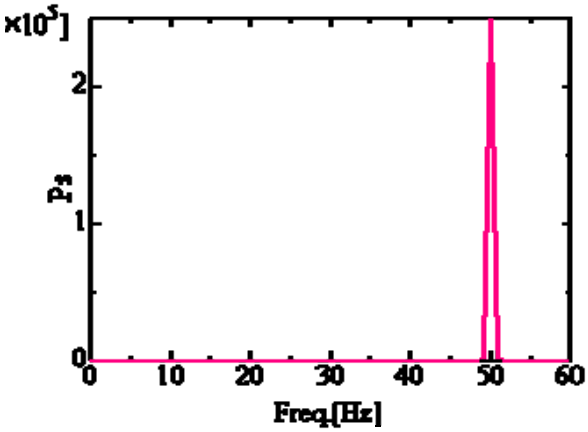

図３　フーリエ変換グラフ

このように周波数成分だけを取り出して出力することができる。
Unity では
「audio1.GetSpectrumData(data,0,FFTWindow.BlackmanHarris);」
と呼ばれるメソッド(操作)を使用することで,フーリエ変換をした周波数成分を取得することができる。

波形表示では,フーリエ変換で取得した周波数成分を毎秒更新し保存,数値として値を出力して「DrawLine」と呼ばれるメソッドを使用すると,その数値を点として描画し,点どうしを結ぶと波形として表示される。

図４実行した際の画面を示しておく。

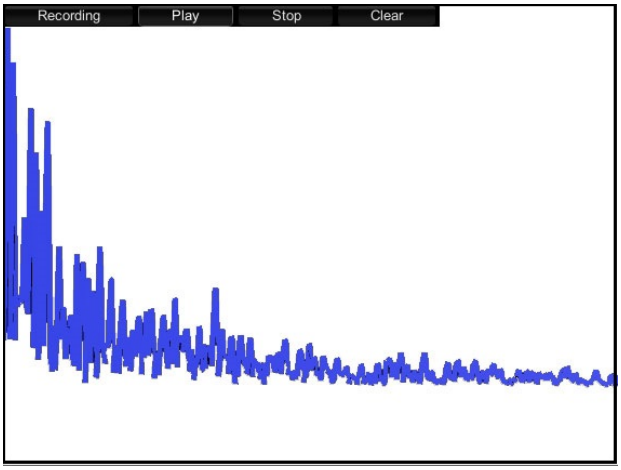

図４　実際の波形表示画面

３．研究のまとめ

今回はUnityを使ってボイスレコーダーの開発を目標に行ってきた。当初はボイスチェンジャーの開発を試みていたが,音声信号処理や,プログラミングの知識の無さを考慮し,ボイスチェンジャーを開発することは断念した。ボイスレコーダーの開発は岡工祭までに「録音」「再生」「削除」まで進むことができ,アプリとして実行できるまでに至った。

Unityでのプログラミングは,Unity付属のライブラリを用いて比較的簡単にプログラミングすることができた。音声信号処理についてはwaveファイルの取り扱い等,困難な個所がいくつもあったが,なんとか「録音」「再生」「削除」することが出来た。

４．感想

最初は,年間計画やアプリのデザインなどにかなり時間を費やし「まぁ,出来るだろう」ぐらいの,軽い気持ちで臨んでいた。だが,いざ開発しようと始めるとソフトウェアの開発をしたことがないので,右も左もわからず,4か月程無駄に過ごした記憶がある。また,音声信号処理という分野は参考文献が少なく,理論からプログラミングを再現していったのでとても難しかった。勿論,多少ソースは参考文献に載っていたので参考文献に頼る形となった。

1年間課題研究を通して,ソフトウェアの開発の大変さを学んだ。中々思い通りに作ることができず苦労だらけだったが,最終的に動くものが出来てよかったと思う。普段何気なく使っているアプリを作るのがこんなに難しいとは思わなかった。大学でも今回の課題研究の経験を生かして頑張っていきたいと思う。

５．参考文献・サイト・使用器具

● マイクロフォン
情報技術科の丸山氏による提供
WiiU付属マイクを使用

● サウンドプログラミング入門 - 音響合成の基本とC言語による実装
青木 直史 著
http://floor13.sakura.ne.jp/book06/book06.html

● サウンド処理のプログラミング
http://nis-ei.eng.hokudai.ac.jp/~aoki/laboratory04.html

● Unity マニュアル-オーディオコンポーネント
http://docs.unity3d.com/ja/current/Manual/comp-AudioGroup.html

● ワンダープラネット株式会社
http://wonderpla.net/blog/engineer/Unity_Co-routine/

● SOFTIST
http://www.softist.com/programming/programming.html

● Unity公式
http://japan.unity3d.com/