平成 27 年度卒業論文

身のまわりの音を楽器にする
音楽知育アプリの開発

函館工業高等専門学校　情報工学科　5 年

東海林研究室　高石莉穂

目次

1 章　序論

1.1 英文アブストラクト

Today, a lot of musical and cognitive education applications exist. However, almost all applications for children play existing instruments. Therefore, we have thought over such a way for children to enjoy the sound more without the existing instruments and have developed a musical education application that makes the musical instrument from sound around the user in this study. First, we have created a program to increase the pitch by pitch-shifting. Then, we have applied the linear interpolation estimation and have created a program to make 12 different pitches by time-stretching. Finally, the pitches are placed on the screen so that the user can play the created scale freely.

Key words : Pitch-shifting, Time-stretching, the linear interpolation estimation

1.2 目的

本研究では音について楽しんでもらうことを目的に、子供を対象とした音楽知育 Android アプリケーションを作成する。端末のマイクに入力された肉声や環境音から音階を作成し、作成した音階をピアノのような鍵盤に割り当てて表示し、自由に演奏できるアプリケーションを作成する。

1.3 背景

現在、スマートフォンやタブレット向けの音楽アプリケーションや知育アプリケーションが多数存在している。しかし子供向けの音楽アプリケーションは知育を目的としたものがほとんどであり、これらは既存の楽器を演奏するものばかりであった。そこで、既存の楽器ではなく、身のまわりの音などが楽器になれば子供たちはより音を楽しむことが出来るのではないかと考えた。

2 章　アプリケーションの概要

2.1 仕様

まず初めに、肉声や環境音を端末のマイクへ入力する[1]。次に、マイクへ入力した音から音階(ドレミ・・・)を作成する。作成した音階をピアノのような鍵盤に割り当てて画面上に表示し[2]、その鍵盤を使って自由に演奏できるようにする [3]。

2.2 アプリケーションの説明

開発したアプリケーションの画面を図 1 に示す。

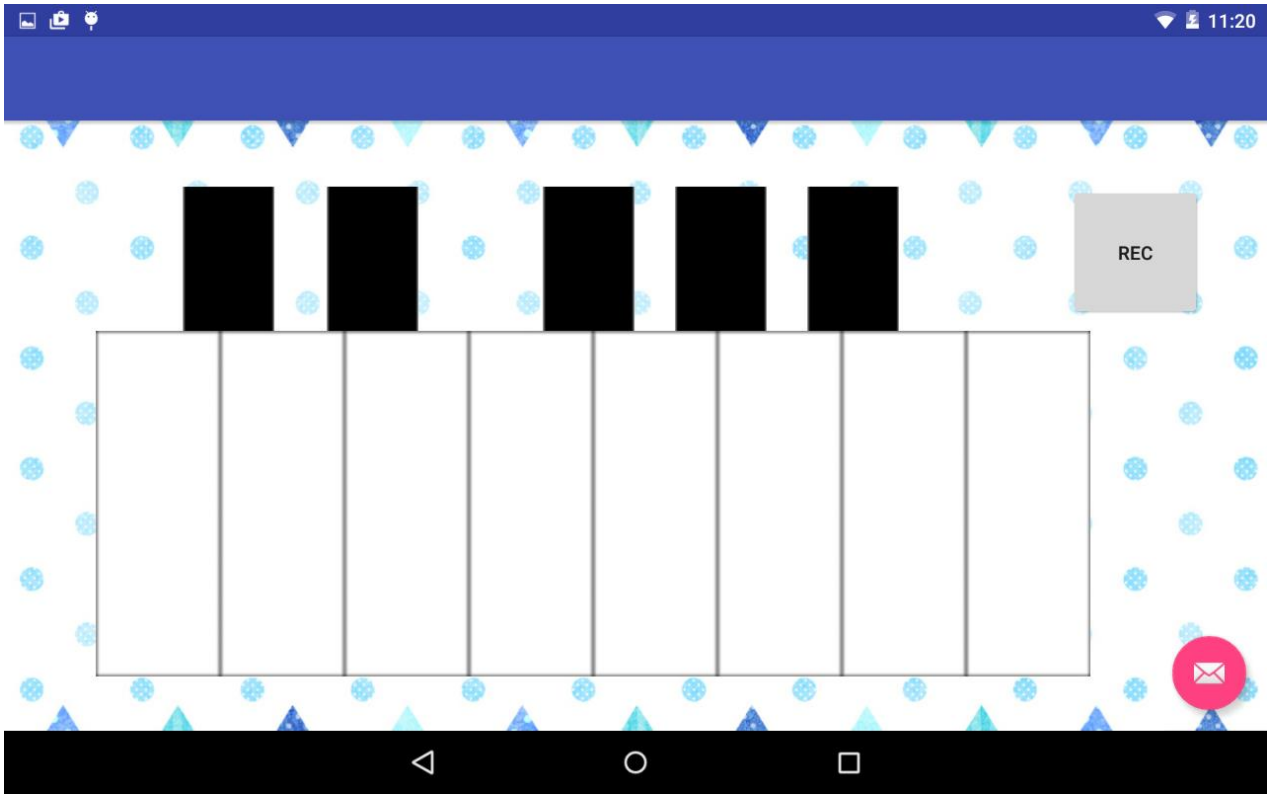

図 1　アプリケーション画面

REC のボタンを押すと 1 秒間録音することが出来る。録音した音を初めのドの音高(ピッチ)とし、それを元に 1 オクターブ上のドまでの音程に変更する。

また鍵盤にはそれぞれの音高が割り当てられていて、鍵盤をタップすると対応した音が鳴る。再生されている間に再度鍵盤をタップすると、前に再生されていたものが止まり、新たに音が鳴る仕様になっている。

2.3 開発環境

開発環境は以下の通りである。

・開発用 OS ･･･ Ubuntu 15.04
・アプリケーション開発環境 ･･･ Android studio 1.4
・開発言語 ･･･ JAVA version 1.8.0_66
・実験端末(図 2) ･･･ Nexus 7 (2012)　Android バージョン 5.1.1

図 2　Nexus 7(2012)

3 章　アルゴリズムについて

3.1　アルゴリズムの概要

音階を作成するために、再生速度を変えずにそれぞれの音階に合ったピッチに録音した音のピッチを変える処理(ピッチシフト)を用いる必要がある。ピッチシフトを行うには「波形の時間方向への拡大･縮小」と「タイムストレッチ」の処理を組み合わせて行う[4]。単純に波形を時間方向に拡大・縮小すると音程も再生速度も変わってしまうため「タイムストレッチ」を行う必要がある。

3.2　タイムストレッチ

タイムストレッチとは、音程を保ったまま再生速度だけを変える処理のことをいう。波形を小さなブロックに分割し、再配置することで可能になる[4]。

例えば図 3 のように録音した音源を小さなブロックに分割し、コピー元の位置を 0.5 ブロックずつずらしながら配置すると再生速度が半分になり、元の音程のままゆっくりと再生される。

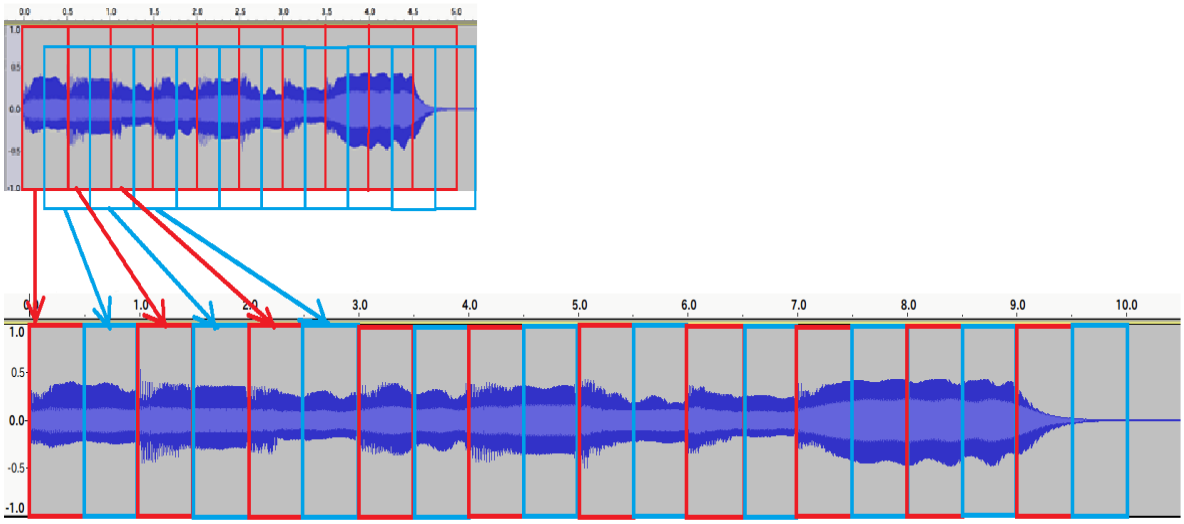

図 3　タイムストレッチ(再生速度半分)の仕組み

また図 4 のように録音した音源を小さなブロックに分割し、ブロックを 1 つ飛ばしながら配置すると音程はそのままの状態で再生速度が 2 倍速く再生される。

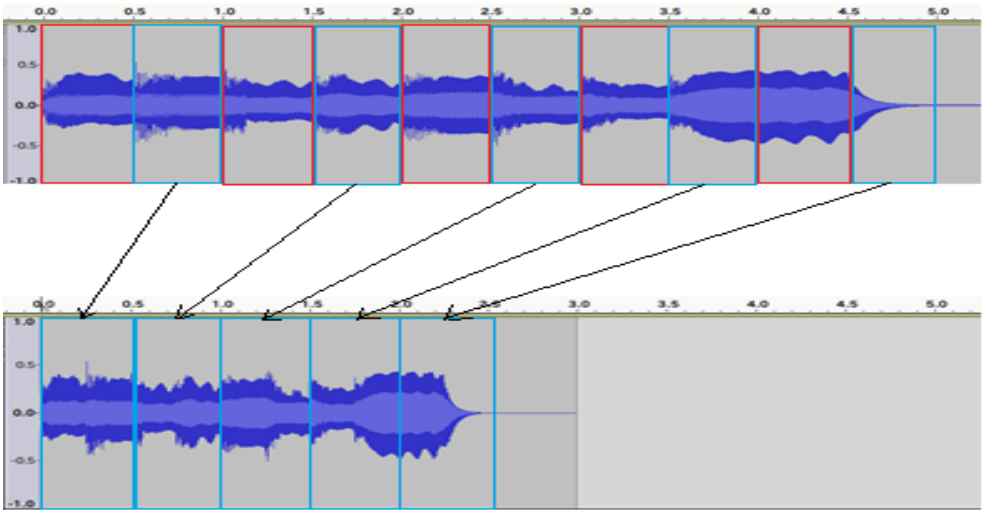

図 4　タイムストレッチ(再生速度 2 倍)の仕組み

これらのようにブロックの配置の仕方によって、音程を保ったままの状態で自由に再生速度を変えることが出来る。

3.3 ピッチシフト

ピッチシフトとは、再生速度を保ったまま音程だけを変える処理のことをいう。ピッチシフトは「波形の時間方向への拡大・縮小」と「タイムストレッチ」を組み合わせることによって可能になる[4][5]。

例として音程を 1 オクターブ上げる処理の説明をする（図 5）。図 5 の 3 つの波形の縦軸は振幅、横軸は時間を表している。まず、波形を単純に時間方向で半分に縮小すると再生速度が 2 倍になりピッチが 1 オクターブ上がる。次に、タイムストレッチを使って再生速度を 1/2 倍にすることにより、元の音の再生速度を保ったまま音程だけが 1 オクターブ上がるピッチシフトが完成する。

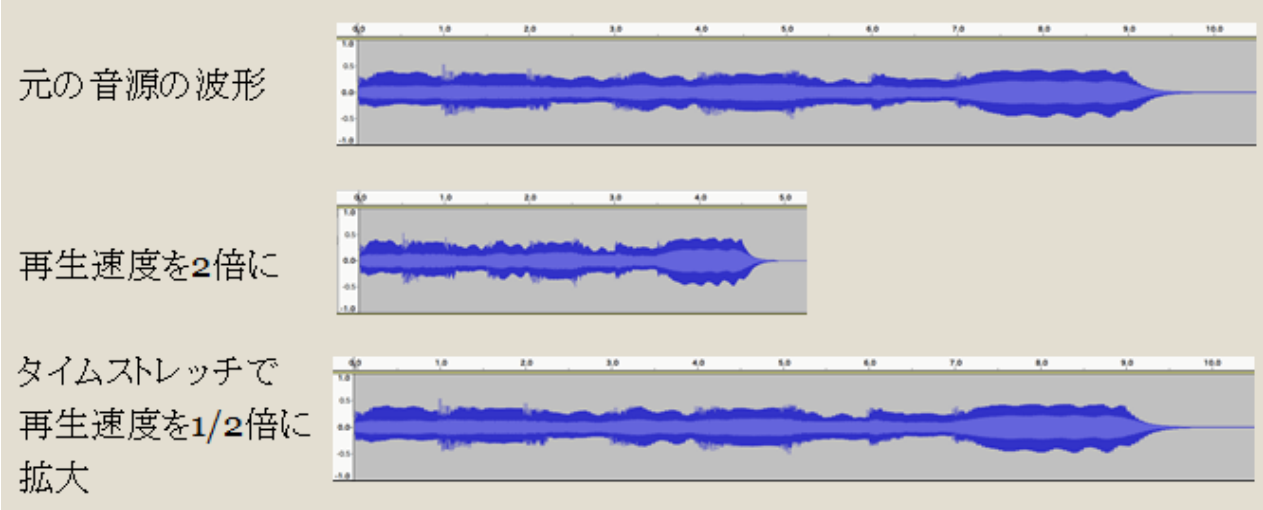

図 5　1 オクターブ上げる処理の仕組み

図 6、7 はそれぞれ元の音のスペクトラムと、1 オクターブ上げるピッチシフトを行った後のスペクトラムの例である。縦軸は周波数、横軸は時間を表している。

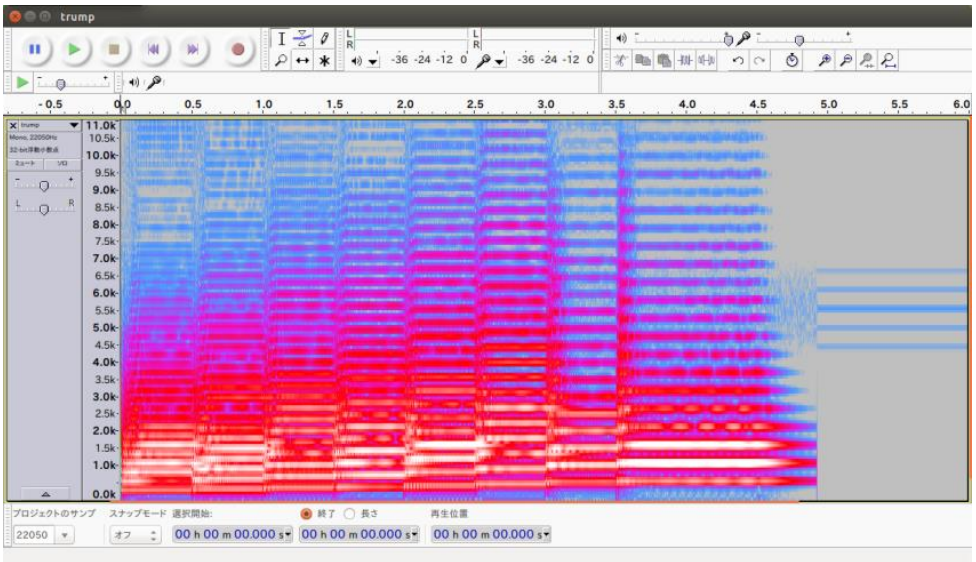

図 6　元の音のスペクトラム

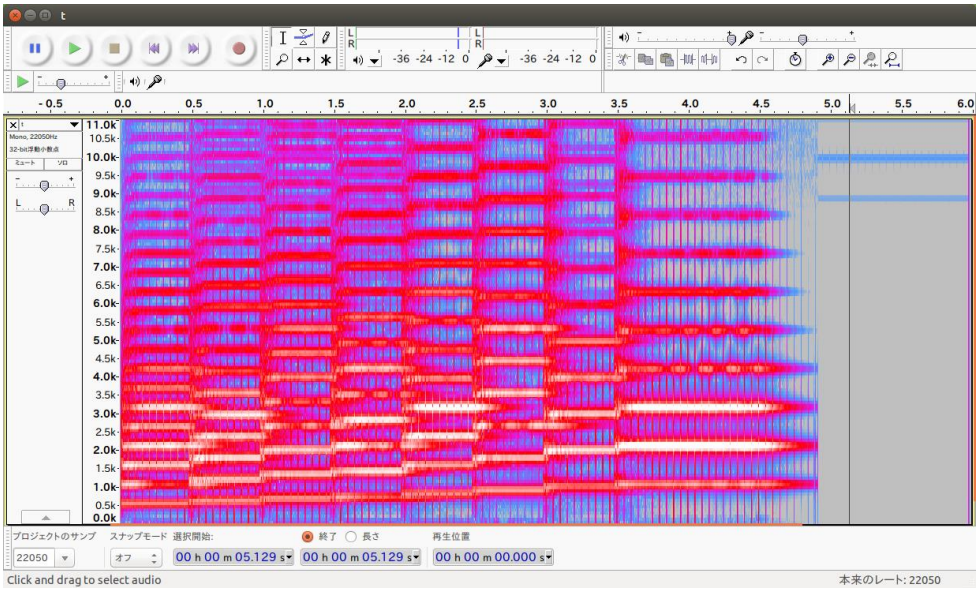

図 7　1 オクターブ上げるピッチシフト後のスペクトラム

1 オクターブ上げるということは周波数を 2 倍にすることを意味する[6]。この 2 つのスペクトラムを見比べてみると、時間軸方向はそのままの状態だが縦に 2 倍に引き延ばされているため周波数が 2 倍になっている。よって 1 オクターブ上げる処理がきちんと行えていることが分かる。

3.4　線形補間による推定を用いた音程変更

補間とは、ある既知の数値データ列を基にして、そのデータ列の各区間の範囲内を埋める数値を求めること、またはそのような関数を与えることである。補間は様々な方法が存在しており、場合によって適した方法を選択する必要がある。代表的なものとして、0 次補間や線形補間、放物線補間、多項式補間、キュービック補間と呼ばれる 3 次補間、ラグランジュ補間、スプライン補間などがある[7]。

本研究で使った補間は線形補間で、これはとある 2 点を直線で結び、間にある任意の値を補間するものである。図 8 でいうと、(x0,y0)と(x1,y1)を直線で結び、その間にある(x,y)を求める。1 次補間としても知られており、数学の世界やコンピュータグラフィックスを含む多くの分野で使われている。計算量が非常に少なく、補間の非常に単純な形式である[8][9]。

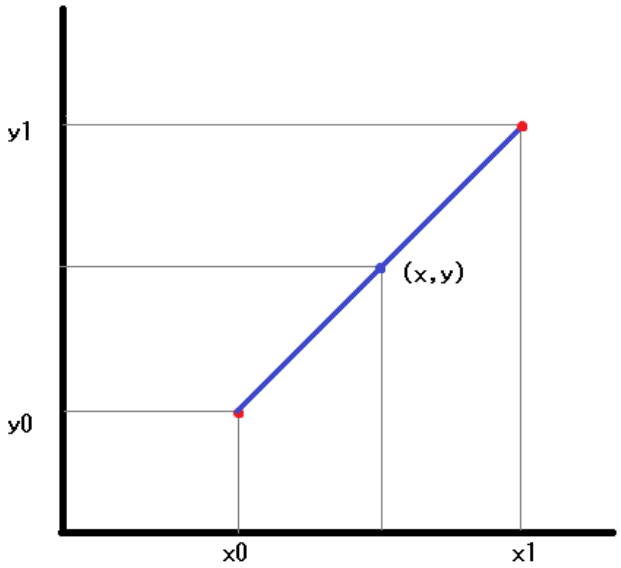

図 8　2 点間の線形補間

音階を作成するにあたって、半音上げたい場合などは波形を時間軸方向に小数倍だけ波形を拡大・縮小する必要がある。そのような場合は元の音声データに含まれていない時点のデータ値の推定を行う必要があり、本研究では線形補間によって補間を行った。

例えば音階を半音上げるには、まず波形を時間方向に $1/(2^{1/12})$ 倍に縮小し、再生速度を $2^{1/12}$ 倍速くしてピッチを半音上げる。そして、タイムストレッチによって音程を変えずに再生速度を $1/(2^{1/12})$ 倍に遅くする必要がある。ここでピッチを半音上げるために波形を時間方向に $1/(2^{1/12})$ 倍に縮小して再生速度を $2^{1/12}$ 倍の速さにする理由は、ドから 1 オクターブ上のドの間には 12 音が存在するためである。もし半音上げるためには再生速度を α 倍しなければならないとすると、1 オクターブ上げる = 再生速度 2 倍に相当することから以下の式が成り立つ。

$$\alpha^{12} = 2$$

したがって、$\alpha = 2^{1/12}$ となり、この値を使うことによって半音上げる処理を行えるようになる[10]。

ここで時刻 $2^{1/12} \fallingdotseq 1.059$ における(デジタルの)音声データ（配列 in に格納したものとする）のデータ値は時刻 1 と時刻 2 の間の値であるためそのままでは取り出すことが不可能である。そこでこの in[1]と in[2]の値を使って in[1.059]の値を推定し、次に in[2]と in[3]のデータを使って in[2.118]のデータを推定し、といったように順番に出力データの推定を行う。そうして推定で求めた新たな音声データの再生速度をタイムストレッチにより $1/(2^{1/12})$ 倍に遅くすることで半音上げるピッチシフト処理が完成する。

3.5　音階の作成

タイムストレッチと波形の時間軸方法の拡大・縮小を用いて 1 オクターブ分および一つ上のドの 13 音階を作成する。具体的には入力された音を下のド音とし、ドからド#に音程を変更するために 3.4 で説明したように、波形を時間方向に $1/(2^{1/12})$倍に縮小して再生速度を$2^{1/12}$倍してから、タイムストレッチを使って再生速度を $1/(2^{1/12})$倍にする。同様に、ドからレにするには波形を時間方向に縮小し再生速度を$2^{1/6}$倍してから、タイムストレッチで再生速度を $1/(2^{1/6})$倍する。図 9 に示すようにこの処理を繰り返すことで 13 音階を作成することが出来る[11]。

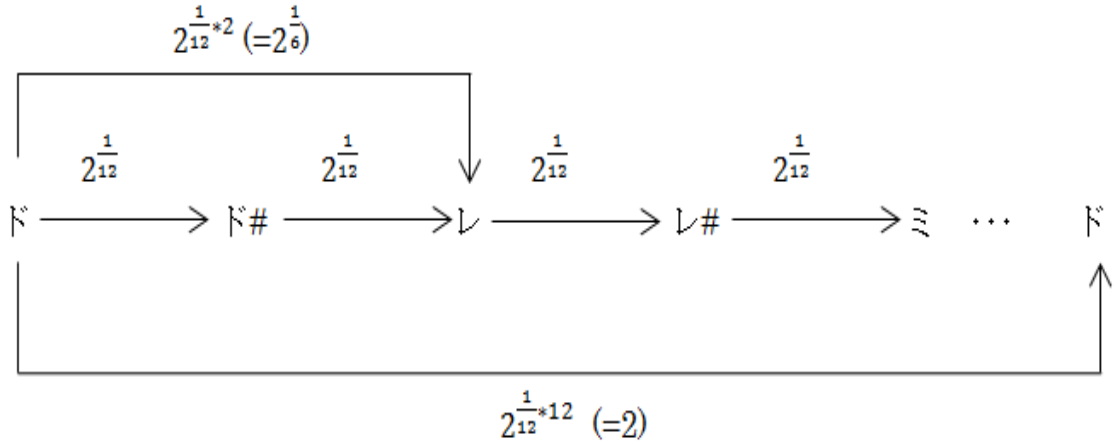

図 9　音程を変更する仕組み

4 章　評価実験

4.1　実験概要

最初にアプリケーションを起動したあと、以下の項目について動作確認を行った。

動作確認項目
(1). 録音ボタンが正常に動作しているか。
(2). 繰り返し録音可能になっているか。
(3). 録音ごとに音階がきちんと作成されているか。
(4). それぞれの音程にあった鍵盤が正常に動作しているか。
(5). 鍵盤を連打したときにもきちんと音がなるようになっているか。

次に実際に 10 人に使ってもらい、以下の項目についてのアンケートを取った。評価は(1)～(6)を 5 段階評価で、(7)を記述で行う。「5」をとても思う、「4」を思う、「3」をどちらとも言えない、「2」をあまり思わない、「1」を思わないとする。

アンケート内容
(1). 子供にもわかりやすい仕様になっている。
(2). 子供向けのデザインになっている。
(3). 鍵盤を押した時の反応に不満はない。
(4). 作成された音階に不満はない。
(5). 録音時間は適切だと思う。
(6). 音について楽しめた。
(7). 感想

4.2　実験結果

動作確認については全ての項目をほぼ満たしていた。(5)の項目に関して、少し間を置いて連打する分には問題はなかったが、間を置かず同時に鍵盤を連打した場合などアプリケーション自体が落ちてしまうことがあった。

また、アンケート結果については以下に記載する。最上段の 5 から 1 は 5 段階評価、それ以降の数字はアンケート内容に対しその評価をした人数である。

アンケート内容	5	4	3	2	1	平均
(1). 子供にもわかりやすい仕様になっている。	1	8	1	0	0	4.0
(2). 子供向けのデザインになっている。	4	6	0	0	0	4.4
(3). 鍵盤を押した時の反応に不満はない。	3	4	2	1	0	3.9
(4). 作成された音階に不満はない。	10	0	0	0	0	5.0
(5). 録音時間は適切だと思う。	6	2	2	0	0	4.4
(6). 音について楽しめた。	8	2	0	0	0	4.8

アンケート項目(7)についても抜粋して記載する。

・使用してみて楽しかった。
・音の変化を楽しめた。
・デザインが可愛い。
・録音ボタンについて、子供向けなら英語ではなく日本語にした方がいい。
・2 つの音階を同時に出せないところを改善して欲しい。
・鍵盤に「ド」「レ」などそれぞれの音程に合った表記をした方がいい。
・録音時間が少し短い。
・録音したとき、最初の無音のところがどうしても出来てしまうのでそれをカットして欲しい。
・音程を上にしかシフトしていないので、下にもして欲しい。
・録音開始・終了の合図があるといい。
・録音せずに鍵盤を押すとアプリケーションが終了するので改善して欲しい。

4.3　アルゴリズムに関する考察

今回任意の音程に変更するにあたって線形補間を使ったが、簡易的な補間方法であることから繋ぎ目のあたりに若干のノイズが発生していた。このノイズを消すにはもっと精度の良い方法で補間を行うと良いと思われる。

また動作確認の結果、録音をする前に鍵盤をタップするとアプリケーションが落ちてしまう問題があったので、この処理も落ちないように改善する必要がある。

更にアンケート結果から、鍵盤を押した時の反応をより良くする必要がある。

4.4　アプリケーションに関する考察

キーボードの鍵盤の配置が簡素なのでそれぞれの鍵盤の形にそった画像を用意し、よりキーボードらしくデザインを整えるべきである。そして、アンケート結果にもあったにもあったように、録音ボタンを英語表記ではなく日本語表記にするべきである。さらに鍵盤にそれぞれの音程の表記もすると良いと思われる。また、知育アプリなのでもう少し知育要素を取り込むべきであると思われる。

5章　まとめ

本研究では、身のまわりの音を楽器にする音楽知育アプリの開発を行った。端末のマイクに入力された肉声や環境音から音階を作成し、作成した音階をピアノのような鍵盤に割り当てて表示し、自由に演奏できるアプリケーションを作成した。

評価実験を行った結果、本研究の目的は達成されていることは分かったが、まだまだ改善点があるため、今後はノイズの少ない音階の作成や、より子供たちが楽しめるアプリケーション開発のために改善を重ねるべきである。

参考文献

[1] teruu のブログ　Android で録音
http://steavevaivai.hatenablog.com/entry/2015/01/03/225505

[2] Android アプリ開発で ImageButton (イメージボタン)を追加する方法【初心者向け】
http://techacademy.jp/magazine/3618

[3] 音声を使って楽しげなものを作りたい。再生をする。
http://seesaawiki.jp/w/moonlight_aska/d/%A5%C7%A1%BC%A5%BF%A5%B9%A5%C8%A5%EA%A1%BC%A5%E0%A4%C8%A4%B7%A4%C6%B2%BB%C0%BC%A5%C7%A1%BC%A5%BF%A4%F2%C6%C9%A4%DF%B9%FE%A4%E0

[4] タイムストレッチ、ピッチシフトのアルゴリズム
http://ackiesound.ifdef.jp/tech/timestretch.html

[5] 音楽プログラミングの超入門(仮)
(2)音のタイムストレッチとピッチシフト:【波形領域】
http://yukara-13.hatenablog.com/entry/2014/02/16/093556

[6] オクターブと周波数/音階名
http://www.ari-web.com/sound/measurement/spl-04.htm

[7] 補間方法と補間精度
http://www.cannula.jp/hokan.html

[8] 線形補間
http://7ujm.net/etc/senkei.html

[9] 第 14 章　補間と最小二乗法
http://na-inet.jp/nasoft/chap14.pdf#search='%E7%B7%9A%E5%BD%A2%E8%A3%9C%E9%96%93'

[10] 倍音とはなにか
http://www.page.sannet.ne.jp/hirasho/sound/baion.html

[11] 平均律について
http://stby.jp/heikinritu.html

付録

short[] in;　//録音したデータを入れる
int LEN;　　//録音時間の長さ

ブロックの長さは 50msec がよいとされている。サンプリング周波数 22050Hz 、サンプリング間隔 1000/22050 、50msec 間に音声データは 50/サンプリング間隔 =50/1000*22050=1102.5≒1100 ということで、1100 ごとに区切っている。

【1 オクターブ上げるピッチシフト】

```
//再生時間*1/2
        short[] data2;
        data2 = new short[len/2];       //再生時間を 1/2 にしたデータを入れる配列
        for(int i=0,j=0;i<len/2;i++,j+=2){
            data2[i]=in[j];
        }

//再生速度*2
         short[] data3;
         int n=0;
         data3 = new short[len];          //タイムストレッチ後のデータを入れる配列
         Arrays.fill(data3,(short)0);

         for(int i=0;i<len;i=i+1100){
               for(int m=0;m<=1100;m++) {
                   if(i+m < len && n+m < len/2)    data3[i + m] = data2[n + m];
                }
                n=n+550;
         }
```

【1 音上げるピッチシフト】

```
//再生時間*2^1/12
        int j=0;
        int len2;
        int n=0;
        double k;
        double p = Math.pow(2,1.0/12);
        short[] data2;

        data2 = new short[(int)(len/(Math.pow(2,(double)1/12)))];
        len2 = (int)(len/p);
        for(int i=0;i<len2;i++){
             k = i * p;
             j = (int)k;
            data2[i] = (short) ((in[j+1] - in[j]) * ( k - j ) + in[j]); //推定
        }

  //再生速度*(1/2^1/12)
        short[] data3;
        data3 = new short[len];
        Arrays.fill(data3,(short)0);

        for(int i=0;i<len;i=i+1100){
                for(int m=0;m<=1100;m++) {
                    if(i+m < len && n+m < (int)(len/p) ) data3[i + m] = data2[n + m];
                }
                n=n+(int)(1100*(1/ p));
         }
```