音声とテキストの同期方法とJavaによる実装

柴崎　慧、小島　秀樹、宇田川 佳久

東京工芸大学工学部コンピュータ応用学科

目　次

1. 研究の背景と概要

- グローバル化が進む現在，特に英語の習得が課題になっている
- 有効な学習方法としてディクテーションがある
  - 単語から文章の構造までを正確に認識する
- ディクテーションは単純な作業になりやすく，苦手とする学習者が多い
- ディクテーション支援機能のJavaによる実装について述べる

研究の方針

- オーディ機器で採用されていることを考慮し，
  - 音声データとしてMP3データ，
  - テキストとしてlrcファイルを採用
- 音声とテキストを同期させて再生するために、Javaのマルチスレッド機能を活用した
- ディクテーションGUIを開発し、実験を実施した

2. MP3ファイルと再生方法

MP3ファイルの構成
[MPEGオーディオフレームヘッダ]
+ [音声データ]
+ [ID3タグ]

MPEGフレームヘッダは，4バイト(32ビット)
MPEG Audio のバージョンID，ビットレート，サンプリングレートなどを記憶している。

サンプリングされた音声データは，フレームと呼ばれる単位で記憶されてる。フレーム単位での再生が可能である。
ファイルサイズ * 8 = 再生時間 * ビットレート

ID3タグは，曲名，アーティスト名，アルバム名，ジャンルなどを記憶した128バイトのデータ

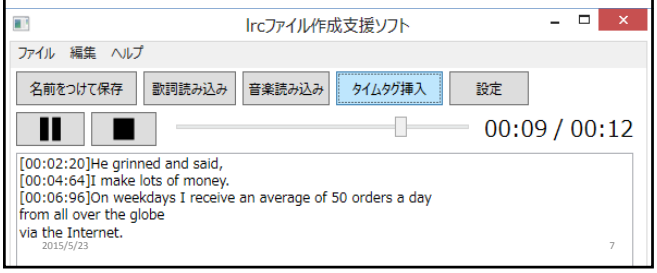
JLayerを使った音声の再生

```
1 import java.io.*;
2 import javazoom.jl.player.Player;
3 ...
4 public void run(){
5 try {
6     stream = new BufferedInputStream(
7                 (new FileInputStream(FName)));
8     play = new Player(stream);
9     int i=0;
10    boolean fn= false;
11    while ( !fn ) {
12        boolean ret = play.play(i);
13        i= i+1;
14        fn= play.isComplete();
15    }
16 } catch (Exception e) {
17    e.printStackTrace();
```

3. Lrcファイルの作成と表示

タイムタグ

[00:01:86] He grinned and said,
[00:03:49] I make lots of money.



Lrcファイルの表示処理の実装

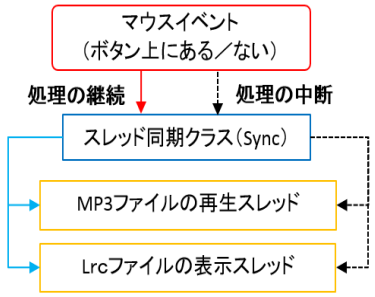
テキストの表示処理

```
import java.io.*;
...
// lrcファイル再生開始時刻の取得
start_time = System.currentTimeMillis();

// lrcファイルの表示処理
str = br.readLine();
    while(str != null){
    ...
        // タイムタグの時刻データの取得
        tm= tm1 + tm2 + tm3;
        for(int j = 0; j < 10000; j++){
            //  経過時間の計算
            length_time =
                System.currentTimeMillis() - start_time;
            if (length_time > tm ){
                System.out.println( str +
                    " 経過時間=" + length_time);
                break;
            } else {
                Thread.sleep(30);
            }
        }
        str = br.readLine();     //次の行の読み込み
    }
br.close();
```


マルチスレッドの同期：操作仕様

- 再生継続：マウスをボタン上に置く
- 再生中断：マウスをボタン以外の場所に置く



Javaのマルチスレッドの同期機能

- wait()： 他のスレッドがこのオブジェクトのnotify() メソッドまたは，notifyAll()メソッドを呼び出すまで，現在のスレッドを待機状態にする．
- notifyall()： wait()メソッドによって待機状態にあるすべてのスレッドを再開する．

図5 スレッド同期のためのSyncクラス

```
public class Sync {
    int ST= 0;    // 0：継続、 1：中断
    synchronized public void sync() {
        if(ST != 0){
            try{
                wait(); // 自分スレッドを休止させる
            } catch (InterruptedException e){
                System.out.println("    ▼   " + e);
            }
    }
    //  処理中断イベントで、このメソッドを実行する
    synchronized public void setST(){
        ST=1;
    }
    //  処理再開イベントで、このメソッドを実行する
    synchronized public void resetST(){
        ST=0;
        notifyAll();//  すべてのスレッドを起こす
    }
}
```


マウスイベント処理とSyncクラス

Sync SN= new Sync();

```
public class myListener implements MouseListener{
    public void mouseExited(MouseEvent e){
        //  マウスが出たことを通知（処理の中断）
        SN.setST();
    }
    public void mouseEntered(MouseEvent e){
        //  マウスが出たことを通知（処理の再開）
        SN.resetST();
    }
}
```

MP3ファイル再生と Syncクラス

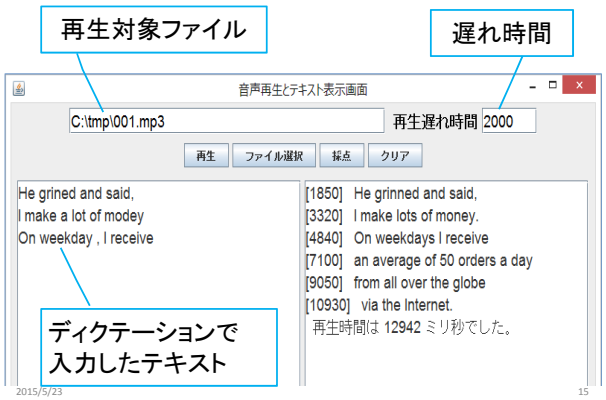
```
1 import java.io.*;
2 import javazoom.jl.player.Player;
3 ...
4 public void run(){
5 try {
6     stream = new BufferedInputStream(
7                     (new FileInputStream(FName)));
8     play = new Player(stream);
9     int i=0;
10    boolean fn= false;
11    while ( !fn ) {
12        boolean ret = play.play(i);
13        i= i+1;
14        fn= play.isComplete();
15        // １フレームの再生ごとに中断をチェックする
16        SN.sync();
17    }
18 } catch (Exception e) {
19 ...
```


図8 中断機能を追加したIrcファイル表示処理

```
4 start_time = System.currentTimeMillis();
5
6 // Ircファイルの表示処理
7 str = br.readLine();
8     while(str != null){
9     ...
10        // タイムタグの時刻データの取得
11        tm= tm1 + tm2 + tm3;
12        for(int j = 0; j < 10000; j++){
13            // 中断した時間を計測し，経過時間から除く
14            time1 = System.currentTimeMillis();
15            SN.sync();    //  スレッドの同期
16            idle_time = System.currentTimeMillis() - time1;
17            start_time= start_time + idle_time;
18            //  経過時間の計算
19            length_time =
20                System.currentTimeMillis() - start_time;
21            if (length_time > tm ){
22                System.out.println( str +
23                    " 経過時間=" + length_time);
24                break;
25            } else {
26                Thread.sleep(30);
27            }
```

中断時間を帳消しにする処理

ディクテーション支援GUIと使用例



カラオケの練習支援としての使用例

デモ

実験の結果と考察

- **ディクテーションへの適用実験**
  - 英文1フレーズの再生が終わった時点で，マウスをテキストエリアに移動し，聞き取った英文を入力する
  - テキスト入力が終わったら，マウスを「再生」ボタンのエリアに移動する
  - 以上を繰り返す
  - 短いセンテンスでディクテーションができる
    - ➡ ストレスが少ない（初級者にも向いている）
- **ディクテーション支援ツール「聞き返し」との比較**
  - キー押下による音声再生の中断・再開ができる
  - テキストの表示機能が無い
    - ➡ 行単位での確認ができない
    - ➡ 中上級者向け

実験の結果と考察

- カラオケの練習支援**の適用実験**
  - ディクテーションとは逆に，テキストを音声よりも先行して表示させる必要がある
    - ➡ 遅れ時間を負の値にする（－1800ミリ秒）
  - テキストが先行表示されるので、聞き取りやすい
    - ➡ 歌詞を覚える段階では効果がある

おわりに

今後の課題

文字列マッチングで実装できる

- ディクテーションの訂正・採点機能
- 音声再生の戻し機能
- 音声再生速度の変更機能
- 音声の再生位置の表示機能

ご清聴ありがとうございました