

平成24年度 特別研究報告書

マウスとソフトキーボードによらない
文字入力手法の検討

龍谷大学 理工学部 情報メディア学科

学籍番号 T090471 森山 慶一

指導教員 三好 力 教授

内容梗概

近年、インターネットの普及に伴って、情報機器に触れる機会が増えた。手の不自由な障がい者がコンピュータに文字入力を行うための従来の手法では、画面上の文字表を見続けないといけないため、視覚や身体に負担がかかる。より容易に文字入力を行うためには、入力速度の向上、視覚のみに頼らない、意識を集中しないで気軽に入力できる、どのコンピュータや携帯端末にも利用できるという汎用性、などが必要である。そこで視覚、聴覚、発声、首の運動など複数の感覚を複合的に利用し、キーボードで入力キーを見ないで文字入力を行うブラインドタッチのような入力を実現するための手法を検討した。

目次

第1章	はじめに	1
第2章	既存技術	2
	2.1 一般的な入力手法	2
	2.2 障がい者向けの入力手法	5
	2.3 分析	8
第3章	提案方式	10
	3.1 ジョイスティックによる文字入力システム	10
	3.2 発声による文字入力システム	12
	3.3 加速度センサを用いた文字入力システム	15
第4章	実験結果と考察	17
	4.1 実験方法	17
	4.2 実験 1：仮名打ちの入力速度	19
	4.3 実験 2：ローマ字打ちの入力速度	20
	4.4 実験 3：提案手法の入力速度	21
	4.5 実験全体の考察	22
第5章	全体の考察	23

第1章 はじめに

近年、インターネットの普及に伴って、情報機器に触れる機会が増えた。コンピュータの一般的な入力手法としてキーボードがあるが、キーボードにも様々な種類があり、例えば **qwerty** 配列、**50** 音配列、親指シフトなどがある。またキー配列を記憶することで、手元を見ずに文字入力を行うブラインドタッチができるようになる。しかしキーボード入力は健常者には容易であるが、手の不自由な障がい者には難しい。

手の不自由な障がい者がコンピュータに文字入力を行うための従来の手法として、ワンスイッチを用いて画面に出る文字表から範囲を絞って文字を選択していくシステムや、ジョイスティックを用いて画面にソフトキーボードを出して、ジョイスティックにより入力したい文字にカーソルを合わせて入力していくというものがある。どちらの方法も画面上の文字表を見続けないといけないため、視覚や身体に負担がかかる。より容易に文字入力を行うためには、入力速度の向上、視覚のみに頼らない、意識を集中しないで気軽に入力できる、どのコンピュータや携帯端末にも利用できるという汎用性、などが必要である。

そこで視覚、聴覚、発声、首の運動など複数の感覚を複合的に利用し、キーボードで入力キーを見ないで文字入力を行うブラインドタッチのような入力を実現するための手法を以下で検討した。文字入力速度が速くなると、文書作成が容易になり、手の不自由な障がい者の仕事の分野が広がる。また言語にも障がいがある人には、新たなコミュニケーションツールとなることが期待できる。

第2章 既存技術

2.1 一般的な入力手法

2.1.1 qwerty キーボードと 50 音キーボード

qwerty 配列は元々タイプライタに用いられていて、一説として技術的な限界から打鍵速度を落としてアームの衝突を防ぐために考え出された配列だと言われる。

図 2-1 に qwerty キーボードを示す。



図 2-1 qwerty キーボード

図 2-2 に 50 音キーボードを示す。



図 2-2 50 音キーボード

2.1.2 親指シフトキーボード

かな打ちを速くするために、親指の近くのキーの状態でかな文字を一意に入力できるようにしたもの。

・かなの入力方法

- ◆下側に刻字されている「かな」→文字キー単独で打鍵する
- ◆上側に刻字されている「かな」→文字キーと同じ側の親指キーを同時打鍵する
- ◆濁音（が、ぎ、だ、ば、等）→文字キーと反対側の親指キーを同時打鍵する
- ◆半濁音（ぱ、ぴ、ぷ、ぺ、ぽ）→シフトキーを押しながら文字キーを打鍵する
または、親指左キーと文字キーを同時打鍵する

図 2-3 に 50 音キーボードを示す。

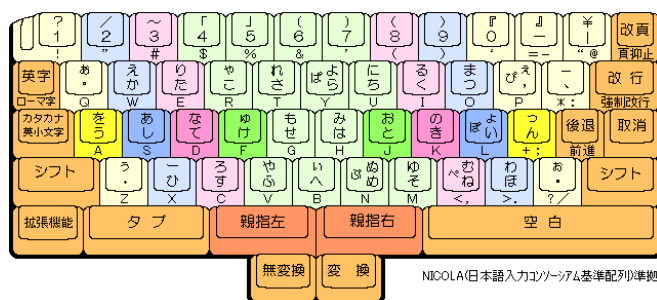


図 2-3 親指シフトキーボード

2.1.3 ソフトキーボード

コンピュータの画面に 50 音表や qwerty、アルファベットの配列のキーボードを表示し、カーソルで入力したい文字を選択していく。

実施例 「もりやま けいいち」を入力








図 2-4 入力例

2.1.4 手書き入力

携帯端末やタブレットのキーボードは、特に外出中などには使いにくいことがある。その場合、Google.com の手書き入力がある。この機能では、指を使って検索キーワードを入力できる。書き進めると、検索ボックスで手書きの文字が単語に変換される。

手順

1. 携帯端末の画面の右下隅にある手書きアイコン  をタップする。
2. 画面の任意の場所で、指を使って検索キーワードを書き始める。
3. 書いている間に、手書きのテキストが検索ボックスで単語に変換される。文字を削除するときは、バックスペース アイコン  をタップする。一度に複数の文字を削除す

- るには、バックスペース アイコン  を押し続ける。始めから入力し直すときは、検索ボックスで **[X]** をクリックする。
4. 入力途中の検索ボックスの下には、予測キーワードのリストが表示される。いずれかのキーワードをタップしてその予測を検索するか、右にある矢印  をクリックして検索キーワードを作成し、さらに入力を続ける。
 5. 検索アイコン  をタップして検索する。

手書き入力の実行画面を図 2-5 に示す。



図 2-5 実行画面

2.1.5 トグル打ち(携帯電話)

1つのキーに五十音の1行やアルファベット3-4文字など複数の文字が対応し、ボタンを押すごとに次の文字が現れる(トグル機構を持つ)文字の打ち方である。例えば「1」のキーを押すごとに「あ→い→う→え→お→…」という順で文字が現れるようになっている。文字の出現順を逆順にする「逆トグル」と呼ばれる機能や「大文字小文字変換」を備えたものもあり、2011年現在携帯電話の文字入力システムとして一般的なATOK・iWnn・POBox Proのどれでも逆トグルや大文字小文字変換に対応している。

2.1.6 2タッチ入力(携帯電話)

かな・英文字表示可能なポケットベル(ポケベル)へ電話機からメッセージを送る際、かな・英文字1文字を送信するため2つの数字の組み合わせで打ち込む方式が日本で多く採用されていた。この文字入力方法をそのまま携帯電話端末で行えるようにしたのが「2タッチ入力」モードである。

携帯電話端末の通常のかな入力方法では、1文字を入力するのに1~5回のキー操作が必要であるのに比べ、必ず2回のキー操作で入力できることが特徴である。

ポケベルにてかな・英文字メッセージを多用していた人にとっては、文字と数字との対応パターンをあらかじめ記憶しているため、一定のキー操作で入力できるメリットが高いとされてきた。

2011年現在では、ATOK・iWnn・POBox Pro など日本国内で発売されている携帯電話に採用されている文字入力システムのほとんど標準で使用することができるようになっている。QWERTY キーボードを搭載した機種種の増加や、シングルタップ・フリック入力などの他の効率的な入力方法の導入もされているが、2タッチ入力にはテンキー搭載機が広く対応している、汎用的な入力方法である点がメリットの1つである。

特徴

- ・必ず2回のキー操作で入力することができる。

2.2 障がい者向けの入力手法

2.2.1 ハーティラーダー



図 2-6 実行画面

仮名なら五十音表が表示して、まず「あ」「か」「さ」「た」「な」など、各行を枠が順次移動する。入力したい行でスイッチを押すと、次はその行内を枠が移動する。

例えば、「さ」行でスイッチを押すと、「さ」「し」「す」「せ」「そ」と移動していき、入力したい文字のところでスイッチを押すと、その文字が決定される。これを繰り返して、単語を入力していく。

2.2.2 ワンスイッチ



図 2-7 ワンスイッチ

使用方法はハーティラーダーなどで文字を選択するときにワンスイッチを押し、入力をする。また上下左右の項目をワンスイッチで選択すると、カーソルはその方向に動き、もう一度押すと止まる。

2.2.3 文字入力装置

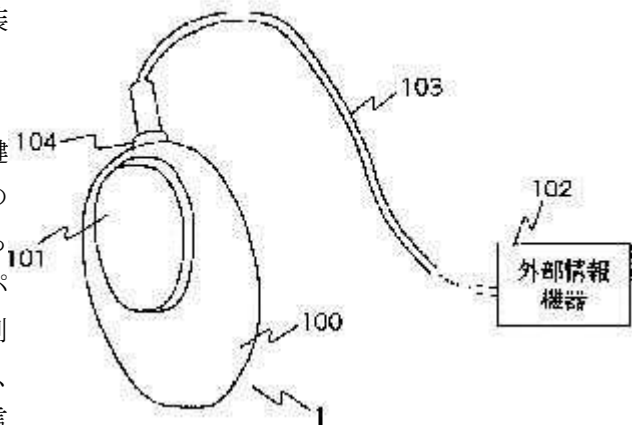
出願日：1998年12月03日

【課題】片手で操作が可能な文字入力手段を有し、小型で携帯性に優れた文字入力装置を提供する。

【解決手段】手のひらに納まるケースと、文字入力用の一つのボタンと、ボタン打鍵モニター用のブザーと、外部情報機器との通信用コネクタを備えており、ボタンから入力されたモールス符号に基づいた打鍵パターンを認識して文字コードに変換する制御部と、文字コードを格納する記憶装置と、外部情報機器に文字コードを送信する通信装置で構成される。

特徴：一つのボタンでモールス信号を表すことができる。

図 1



2.2.4 文字入力システム及びポインティングデバイス

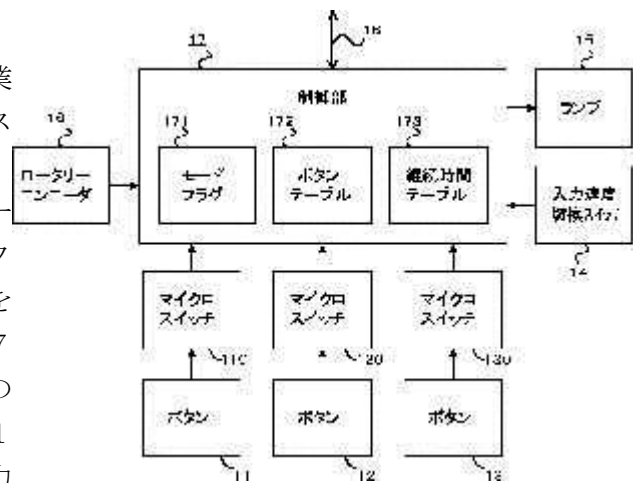
出願日：1999年01月25日

(57) 【要約】

【課題】 キーボードと同等の文字入力作業が可能で、かつ、操作性の良い文字入力システム等を提供する。

【解決手段】 操作者がマウス1の動作モードを切り替えるためのアイコンをクリックすると、コンピュータ2はモード切替信号をマウス1に送信する。マウス1の制御部17は、内部メモリに例えばキーボードモードのフラグを設定する。以後、操作者がマウス1のボタンを操作すると、モールス符号が入力される。ボタンは、ボタンテーブル172の設定に従い、一方が短点入力用、他方が長点入力用として機能する。制御部17は、ボタン操作に応じてマイクロスイッチ110、120がオンすると、継続時間テーブル173を参照して、所定の長さの短点信号又は長点信号をコンピュータ2に送信する。コンピュータ2は、受信した短点信号と長点信号の組み合わせについてモールス符号表を検索し、対応する文字コードを判別する。

特徴：モールス信号での入力と他の入力方法を併用できる。



2.2.5 文字入力装置およびその方法

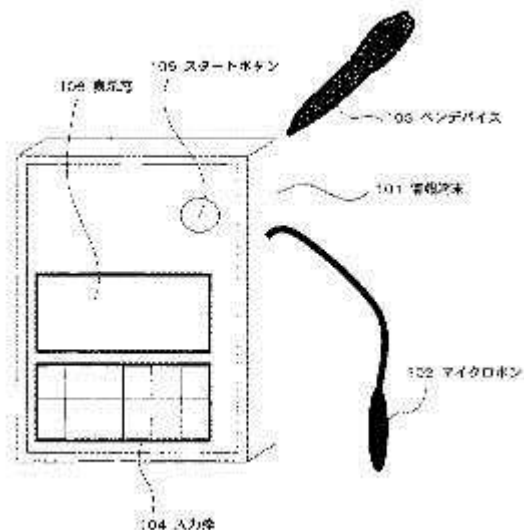
出願日： 2003年08月25日 (72)発明者： 丸山 剛

【要約】

【課題】 ペンデバイスを用いて入力する際に、キーボードと同等の時間で入力でき、且つ文字認識率を向上させた文字入力装置およびその方法を提供する。

【解決手段】 座標入力手段により音素を入力する第一の音素入力手段（ペンデバイス103と入力枠104）と、音声入力手段により音素を入力する第二の音素入力手段（マイクロホン102）とを備え、前記第一の音素入力手段により得られた音素と、前記第二の音素入力手段により得られた音素とから文字を認識し、当該文字入力装置に入力する構成としてある。

【選択図】 図1



特徴：ペンデバイスと音声を併用することで文字入力の効率を上げる。

2.2.6 息で文字入力できる障害者向け Twitter 投稿 iPad アプリ「息鳥」

障害を持つユーザーでも Twitter を利用しやすい iPad アプリ「息鳥」（いきどり）を App Store で無料配信開始した。キーボード画面上のカーソル移動に従って iPad 用外付けマイクに息を吹きかけて文字の選択/入力をする「ブレスコントロール機能」を搭載しており、手指や言語に障害をもつユーザーも Twitter の投稿と閲覧が行える。

マイクは市販の iPad 対応のものを利用でき、ブレスコントロール機能だけでなく、通常のキーボード入力も可能。タイムラインを自動更新して定期的なスクロールを行ったり、「更新」選択で更新する機能も搭載。

なお、同アプリは、公独立行政法人情報通信研究機構（NICT）による平成22年度「高齢者・チャレンジド向け通信・放送サービス充実研究開発助成金」事業として、2010年7月よりテックファームと慶應義塾大学らが産学共同で進めている「スマートフォンを活用した障害者支援 ICT 技術」の研究開発の成果の一部をテックファームが製品化したもの。本研究開発では、今後より広い用途で使用できるスマートフォンアプリを開発する予定。



利用イメージ



図 2-8 実行画面

特徴：手を使わずに文字入力することができる。

2.3 分析

手の不自由な人には手書き入力で文字を直接入力することは難しいが、技術を応用して簡単に縦線を描くと短点、横線を描くと長点としたらモールス信号を利用することができる。

qwerty 配列はタイプライタからの継承により、現在のコンピュータのキーボードとして主流となっている。したがって qwerty 配列はコンピュータへの文字入力の利便性と関係はないと思われる。日本人にとって 50 音配列の方が使いやすいと思うがそうではない。何故なら qwerty 配列が先に普及されているためである。ゆえに利便性は配列に依存しない。

トグル打ちは母音が o である文字を選択するとき、ボタンを 5 回押す必要がある。親指シフトキーボードと 2 タッチ入力は一文字を打つための操作回数は 2 回で済む。2 タッチ入力の方が更にキー数が少ないので文字入力速度は速い。どちらもキー配置を記憶するとブラインドタッチが可能になり、視覚負担の軽減になる。また場所を取るが 10 個のワンスイッチで 2 タッチ入力を実現できれば、身体的な負担の軽減にもなる。

モールス信号を用いたシステムは一文字を打つための操作回数は文字によって異なり、親指シフトキーボードと 2 タッチ入力に比べ、文字入力速度は下がるが、モールス信号を記憶すると視覚負担の軽減になる。しかしボタンを用いているので、1 回押すつもりが 2 回押してしまうような誤操作が起きる。

「文字入力装置およびその方法」は 2 つのデバイスを用いているので、一つの操作に集中できない。

「息鳥」とハーティラーダーは範囲を絞っていく入力手法なので、かなり文字入力速度は遅い。その上スイッチを押したり、息を吹くタイミングを逃すと、再び順番が来るのを待たなければいけない。また入力が終わるまで画面を見続けなければならないので、視覚への負担がかなり掛かる。その点ソフトキーボードは待つ時間がないので、「息鳥」やハーティラーダーよりは視覚への負担が軽減される。正確性の観点からソフトキーボードは隣の文字を誤って打つ可能性があるが、ハーティラーダーは範囲を絞っていく入力手法なので誤字は少ない。

これらのことを考慮し、新たな手法を検討した。

第3章 提案方式

文字入力時の目の負担軽減と入力速度の向上のために、ソフトキーボードでの入力時の視覚フィードバックの要らない文字入力システムを開発した。開発したシステムは3つあり、どちらもモールス信号を用いている。モールス信号とは短点と長点の二つの信号の組み合わせにより、一文字を表す。またどちらのシステムもC#言語でプログラミングし、win32apiメソッドによりwordなどに出力する。

3.1 ジョイスティックによる文字入力システム

障がい者用入力デバイスとして既存技術の2.2.1の文字入力装置と2.2.2の文字入力システム及びポインティングデバイスがあるが、モールス信号入力に用いるには以下の問題点がある。

- **文字入力装置**

一つのボタンで長点と短点を使い分ける事が難しい。

- **文字入力システム及びポインティングデバイス**

モールス信号を用いる場合、その都度マウスの動作モードを切り替える必要がある。

そこであえて一般的なジョイスティックを用いてモールス信号を入力することを考えた。手法の概要を以下に示す。

1. ジョイスティックを左に倒すと1秒おきに1(短点)、右に倒すと1秒おきに2(長点)を文字列として追加される。
2. ジョイスティックを左に倒すと1秒おきに1(短点)、右に倒すと1秒おきに2(長点)が文字列として追加される。
3. ジョイスティックを放した状態で4秒おくと1と2の文字列をもとに、モールス信号の規則に従って文字(アルファベットまたは仮名)に変換される。続けて5回長点を入力、またはモールス信号の規則にない1と2の文字列になるとリセットされる。
4. 続けて6回短点を入力するとモード切替(アルファベット⇄仮名)となる。
5. 音または視覚によるフィードバックで、短点と長点の数を確認し、入力を行う。

工夫点：

- 始めジョイスティックの倒す回数を短点と長点の数としようとしたが、連続打ちを速くするために倒す時間数を短点と長点の数とした。
- ジョイスティックを素早く左右に切り替えても、短点と長点を区別できるようにした。

3.1.1 詳細

グローバル変数

joystick : ジョイスティックの信号を扱うためのもの

joy : ジョイスティックを左に倒すと1、右に倒すと2、手を放すと0を格納

count1 : モードの切り替えに用いる

subst102 : 1秒おきにjoyに格納されたものを文字列として並べる

st : 1秒おきにjoyに格納された左右の情報のみを文字列として並べる

opt : word ソフトなどに出力する文字を格納

主要関数の外部仕様

`bool HasString(string target, string word)`

使用目的：第1引数に第2引数が含まれるか調べる

`void timer1_Tick(object sender, EventArgs e)`

使用目的：モールス信号により文字入力を行う

主要関数の内部仕様

`bool HasString(string target, string word)`

処理手順

1. 第2引数に文字が格納されていなければ `false` を返す
2. もし第1引数に第2引数が含まれているなら `true` を返す
3. もし第1引数に第2引数が含まれていないなら `false` を返す

`void timer1_Tick(object sender, EventArgs e)`

変数

`pov` : ジョイスティックの倒す角度を示す(上 : 0度 右 : 90度 下 : 180度 左 : 270度)

`wavePlayer` : 音を鳴らす

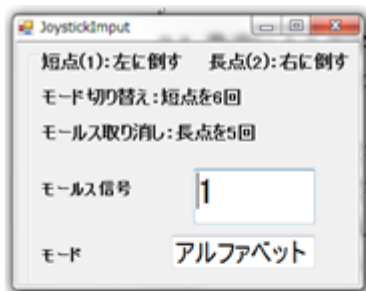
`path, input` : 音源ファイルを指定

処理手順

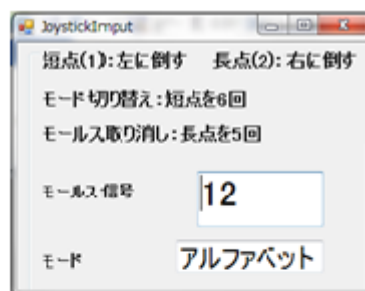
1. `pov` の値が 270 から 360 の間なら以下の処理を行う
 - 1.1 `joy` に 1 を格納し、`st` に `joy` を追加
 - 1.2 `subst102` に 1 が入っていないなら `subst102` に `joy` を代入
 - 1.3 `subst102` が空でないなら `subst102` に `joy` を追加
 - 1.4 短点用の音を鳴らす
2. `pov` の値が 0 から 90 の間なら以下の処理を行う
 - 2.1 `joy` に 2 を格納し、`st` に `joy` を追加
 - 2.2 `subst102` に 2 が入っていないなら `subst102` に `joy` を代入
 - 2.3 `subst102` が空でないなら `subst102` に `joy` を追加
 - 2.4 長点用の音を鳴らす
3. `pov` の値が 0(ジョイスティックを上を倒す、もしくは触れていない状態)なら以下の処理を行う
 - 3.1 `joy` に 0 を格納し、`subst102` に `joy` を追加
4. `subst102` に "00000" が含まれていれば、`st` に "\$End" を追加
5. `st` が "\$End" または "\$End\$End" なら `st` を初期化
6. `st` に "22222" が含まれていれば、`st` を初期化し取り消し音を鳴らす
7. `subst102` の文字列の長さが 10 になれば `subst102` を初期化
8. `subst102` に "111111" が含まれていれば、以下の処理を行う
 - 8.1 `subst102` と `st` を初期化
 - 8.2 `count1` に 1 を足す
 - 8.3 取り消し音を鳴らす
9. `joy`、`subst102`、`st` をトレース出力
10. `st` を文字に変換(`count1` を 2 で割って余りが 0 ならアルファベット、1 なら仮名)

3.1.2 実施例

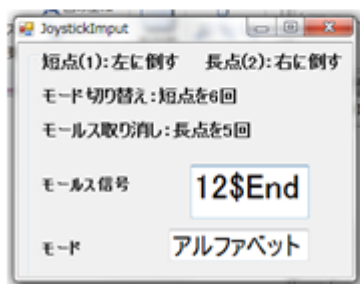
例 「a」を入力



① ジョイスティックを左に1秒倒す



② ジョイスティックを右に1秒倒す



③ ジョイスティックから手を放す

図 3-1 実行画面

3.1.3 利点と課題点

画面を見ずに文字入力可能であるが、マウスの代わりと文字入力用の二台のジョイスティックが必要となる。

3.2 発声による文字入力システム

障がい者用入力デバイスとして既存技術の 2.2.4 の息鳥と 2.2.2 の文字入力システム及びポインティングデバイスがあるが、利便性の観点から以下の問題点がある。

- 息鳥

1. 文字表で画面の大部分を占める
2. 文字を選択するときに画面を見ていないといけない

- 文字入力システム及びポインティングデバイス

モールス信号を用いる場合、その都度モードを切り替える必要がある。

そこで視覚と手の負担を減らすために、発声を用いてモールス信号を入力することを考えた手法の概要を以下に示す。

1. 発声を1回すると1(短点)、1秒以内に2回連続すると2(長点)を文字列として追加される。
2. 無音状態で3秒おくと1と2の文字列をもとに、モールス信号の規則に従って文字(アルファベット・仮名)に変換される。1秒間続けて発声すると1と2の文字列がリセットされる。また2秒間続けて発声するとモード切替(ロック⇒アルファベット⇒仮名)。
3. 音または視覚によるフィードバックで、短点と長点の数、1と2の文字列のリセット、モード切替を確認。

3.2.1 詳細

グローバル変数

- opt : wordソフトなどに出力する文字を格納
- input : 文字入力に用いる
- subinput, subcount : モード切り替えに用いる
- st : 1,2から成る文字列を格納
- i : 発声間隔を調べるためのもの
- ms : 1 か 2 を格納

主要関数の外部仕様

- bool HasString(string target, string word)
使用目的 : 第 1 引き数に第 2 引き数が含まれるか調べる
- void timer1_Tick(object sender, EventArgs e)
使用目的 : モールス信号により文字入力を行う

主要関数の内部仕様

- bool HasString(string target, string word)
処理手順
 1. 第 2 引き数に文字が格納されていなければ false を返す
 2. もし第 1 引き数に第 2 引き数が含まれているなら true を返す
 3. もし第 1 引き数に第 2 引き数が含まれていないなら false を返す

- void timer1_Tick(object sender, EventArgs e)
処理手順
 1. 無音なら以下の処理を行う
 - 1.1 subcount が 1 のときテキストボックスに"locking!"を表示
 - 1.2 subcount が 2 または 3 のとき input に 0 を追加
 - 1.3 subinput に 0 を追加
 2. 音を認識したら以下の処理を行う
 - 2.1 テキストボックスに"○"を表示
 - 2.2 input に"1"が含まれていないなら 1 を追加
 - 2.3 input に"10"または"3"が含まれていないなら 2 を追加
 - 2.4 input に"2"が含まれているなら 3 を追加
 - 2.5 subinput に 1 を追加
 - 3 テキストボックスに input と subinput を表示
 - 4 input の何文字目に 1,2 があるか求める
 - 5 input の 2 文字目に 1 がある,または input の文字列の長さが 40 になれば input を初期化
 - 6 subinput の文字列の長さが 25 になれば subinput を初期化
 - 7 input の 1 と 2 の間の 0 の数を数える
 - 8 subcount が 2 または 3 のとき以下の処理を行う
 - 8.1 無音が続いたら input を初期化し,st に終点を追加

- 8.2 imput に "10000"が含まれているなら以下の処理を行う
 - 8.2.1 短点用の音を鳴らす
 - 8.2.2 imput を初期化し,st に 2 を追加
- 8.3 st に"\$End\$End",またはsubimput に"111111"が含まれているなら以下の処理を行う
 - 8.3.1 st と imput を初期化
 - 8.3.2 取り消し音を鳴らす
- 8.4 subimput に"111111111111111111"が含まれているなら以下の処理を行う
 - 8.4.1 モード切り替え音を鳴らす
 - 8.4.2 subcount に 1 を足す
 - 8.4.3 subimput を初期化
- 9 stを文字に変換(subcountが2ならアルファベット、3なら仮名)

3.2.2 実施例

例 「a」を入力



図 3-2 実行画面

工夫点：

始め 2 回発声した間隔で短点と長点を区別しようとしたが、少しでも発声回数を減らして楽に文字入力するために、1 回発声すると短点、2 回発声すると長点とした。

利点：

画面から離れて他の作業をしながら文字入力可能。

課題点：

吐息に反応、スピーカとマイクが近いと音によるフィードバックを発声と認識してしまう。

3.3 加速度センサを用いた文字入力システム

近年普及してきているスマートフォンを利用し、外出先でも文字入力が容易になればメールを打ちやすくなる。手法として内蔵されている加速度センサを用いて、タブレットを水平にした状態で上に倒すと短点、下に倒すと長点としてモールス信号による文字入力を行う。

3.3.1 詳細

グローバル変数

s : 1(短点)、2(長点)を格納

morse : "\$ST" (初期値)、"\$End" (終点)、1(短点)、2(長点)を格納

mo : 文字に変換するために用いる

stext,morsetext,morseLtext,motext : s、morse、moの内容を表示するために用いる

主要関数の外部仕様

bool HasString(string target, string word)

使用目的 : 第1引数に第2引数が含まれるか調べる

void onCreate(Bundle savedInstanceState)

使用目的 : 文字などの大きさや位置の指定

void onSensorChanged(SensorEvent e)

使用目的 : モールス信号により文字入力を行う

主要関数の内部仕様

bool HasString(string target, string word)

処理手順

1. 第2引数に文字が格納されていなければ false を返す
2. もし第1引数に第2引数が含まれているなら true を返す
3. もし第1引数に第2引数が含まれていないなら false を返す

void onCreate(Bundle savedInstanceState)

処理内容

stext,morsetext,morseLtext,motext をインスタンス化し、フォント指定する

void onSensorChanged(SensorEvent e)

処理手順

- 1 e.values[1](加速度センサの値)が1から6の間ならsに1を追加
- 2 e.values[1](加速度センサの値)が-1から-6の間ならsに2を追加
- 3 e.values[1](加速度センサの値)が6以上ならsに0を追加
- 4 sの文字列の長さが30になればsを初期化
- 5 morseの文字列の長さが10になればmorseを初期化
- 6 sを表示
- 7 sに"11111111111111111111"が含まれているならsを初期化し、morseに"1"を追加

- 8 sに" 2222222222222222"が含まれているならsを初期化し、morseに"2"を追加
- 9 sに" 0000"が含まれているならsを初期化し、morseに"\$End"を追加
- 10 morseを表示
- 11 morseに"\$End\$End"または"\$End1"または"\$End2"が含まれているならmorseを初期化
- 12 morseを文字に変換

時間の都合により、実験はしなかった。

第4章 実験結果と考察

4.1 実験方法

提案手法の入力速度と目の疲れを既存技術と比較するため、「もりやまけいいち」を表 4-1 で示す文字列で入力した時のそれぞれの手法の入力時間と目の疲れの計測を行った。

かなキーボードはかな、アルファベットキーボードは英字、提案手法は数字で示すモールス信号を用いて入力した。サイズやキー配列の異なるソフトキーボードと提案手法で入力を 10 回ずつ繰り返し、その平均値を求めた。

手始めに普段利用している 50 音配列でサイズの違いによるかな入力速度の違いを調べ、その後アルファベットで配列と入力速度の関係性と提案手法を検証した。

独自で制作した文字入力時間計測プログラムの実行画面を図 4-1 に示す。

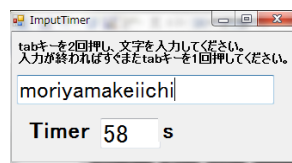


図 4-1 実行画面

1 を短点、2 を長点として各文字とモールス信号の対応を表 4-1 に示す。

表 4-1

も		り		や		ま		け		い	い	ち		
m	o	r	i	y	a	m	a	k	e	i	i	c	h	i
22	222	121	11	121	12	22	12	212	1	11	11	2121	1111	11

IME パッドのソフトキーボードを図 4-2 に示す。



図 4-2 ソフトキーボード(横 : 400 ピクセル 縦 : 145 ピクセル)

こどもキーボードを図 4-3 に示す。

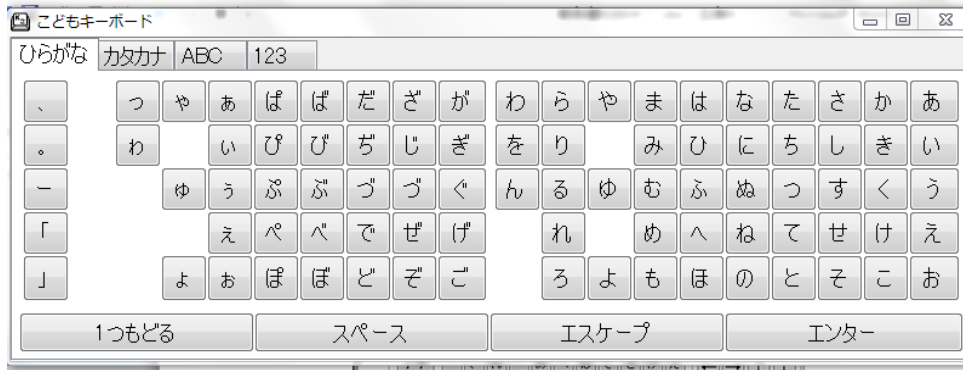


図 4-3 こどもキーボード(横 : 800 ピクセル 縦 : 300 ピクセル)

スクリーンキーボードを図 4-4 に示す。



図 4-4 スクリーンキーボード(横 : 800 ピクセル 縦 : 300 ピクセル)

4.2 実験1：仮名打ちの入力速度

50音表の配列のキーボードを画面に表示し、「もりやま けいいち」の入力時間を10回計測する。

こどもキーボードとソフトキーボードの計測時間を表4-2に示す。

表4-2 仮名打ちの入力速度

単位[秒]	50音配列(こどもキーボード)	50音配列(ソフトキーボード)
1回目	29	30
2回目	26	28
3回目	27	34
4回目	29	30
5回目	28	33
6回目	28	33
7回目	27	26
8回目	28	27
9回目	27	28
10回目	26	30
平均	27.50	29.90
秒/文字	3.44	3.74

4.2.1 考察

一文字の入力速度を見ると、こどもキーボードが3.44秒/文字、ソフトキーボードが3.74秒/文字となっており、キーボードサイズの大きなこどもキーボードが少し早くなっている。これは、サイズが大きいためカーソルの移動距離が長くなることよりも、目的のキーにカーソルを合わせるときの精度を要求されないことの方が入力速度への影響が大きいためと考えられる。またキーボードが大きいほど目の疲れが小さいと感じた。

4.3 実験2：ローマ字打ちの入力速度

配列とサイズの異なるキーボードを画面に表示し、「moriyama keiichi」の入力速度を10回計測する。こどもキーボードとソフトキーボードのqwerty、abc配列の計測時間を表4-3に示す。

表 4-3 ローマ字打ちの入力速度

単位{秒}	qwerty 配列(スクリーンキーボード)	abc 配列(こどもキーボード)	qwerty 配列(ソフトキーボード)	abc 配列(ソフトキーボード)
1回目	65	64	71	67
2回目	61	64	73	70
3回目	50	60	59	69
4回目	52	59	59	69
5回目	59	60	66	68
6回目	57	54	67	68
7回目	55	57	61	69
8回目	61	59	61	55
9回目	56	62	62	60
10回目	53	55	60	63
平均	56.90	59.40	63.90	65.80
秒/文字	3.79	3.96	4.26	4.39

4.3.1 考察

ソフトキーボードのサイズは横が400ピクセル、縦が145ピクセル、スクリーンキーボードとこどもキーボードのサイズは横が800ピクセル、縦が300ピクセルで、スクリーンキーボードとこどもキーボードのサイズはソフトキーボードのおよそ2倍である。一文字の入力速度を見ると、ソフトキーボードのqwerty配列が4.26秒/文字、abc配列が4.39秒/文字で、大きいキーボードのqwerty配列が3.79秒/文字、abc配列が3.96秒/文字である。

キーボードサイズの大小では、配列に関わらず大きいサイズの方が少し速くなっている。これは4.2.1と同様にサイズが大きいためカーソルの移動距離が長くなることよりも、目的のキーにカーソルを合わせる際の精度を要求されないことの方が入力速度への影響が大きいためと考えられる。キー配列での比較では、キーボードサイズに関わらずabc配列よりもqwerty配列の方が入力速度が速い。これはabc配列で文字を打つ場合、普段利用していなくてアルファベットの順番を辿る必要があるためである。一方、qwerty配列は記憶しているので辿る必要がない事が入力速度に影響したものとする。ブラインドタッチまでいかなくともキー配列を記憶することである程度目の負担が軽減されることが分かる。

また配列に関係なくサイズの大きなキーボードの方が実験1と同様に目の疲れが小さいと感じた。

4.4 実験3：提案手法の入力速度

ジョイスティック、発声を用いたモールスシステムの計測時間を表 4-4 に示す。

表 4-4 提案手法の入力速度

単位[秒]	モールス(ジョイスティック)	モールス(発声)
1 回目	75	163
2 回目	71	152
3 回目	74	147
4 回目	78	146
5 回目	76	164
6 回目	75	159
7 回目	79	142
8 回目	81	165
9 回目	74	169
10 回目	85	177
平均	76.80	158.40
秒/信号	2.02	4.17

4.4.1 考察

モールス信号の一信号の入力速度を見ると、ジョイスティックを用いたシステムは 2.02 秒/信号、発声によるシステムは 4.17 秒/信号である。発声によるシステムはジョイスティックを用いたシステムのおよそ倍になった。この一つの要因として、長点がジョイスティックの場合 1 回の入力に対し、発声の場合 2 回の入力が必要となることが挙げられる。

また音によるフィードバックでほとんど画面を凝視する必要がないため、目の負担が軽減された。

4.5 実験全体の考察

表 4-5 に入力手法と入力速度、ジョイスティックを用いたモールスシステムの入力速度を 100 としたときの他の手法の入力速度の比率、目の負担(主観的)を示す。

表 4-5 入力速度

入力手法	入力速度[秒/1 入力]	入力速度の比率	目の負担
モールス(ジョイスティック)	2.02	100	小
モールス(発声)	4.17	206	小
50 音キーボード(大)	3.44	170	中
50 音キーボード(小)	3.74	185	大
qwerty キーボード(大)	3.79	188	中
qwerty キーボード(小)	4.26	211	大
abc キーボード(大)	3.96	196	中
abc キーボード(小)	4.39	217	大

表 4-5 からモールス信号の一つの信号を一文字とすると、最も速く一文字を入力する、かつ目の負担軽減することが出来るのはジョイスティックによるモールスシステムである。ゆえに速さと視覚負担軽減の総合的な観点から現時点ではジョイスティックによるモールスシステムが有効であることが分かる。また最も一文字の入力速度が遅く、目の負担があるのは小さいキーボードの abc 配列である。この要因としてキーボードサイズが小さい事と、abc 配列に慣れていない事が考えられる。

第5章 全体の考察

手の不自由な障がい者が容易に文字入力を行うためには、入力速度の向上、視覚のみに頼らない、意識を集中しないで気軽に入力できる、どのコンピュータや携帯端末にも利用できるという汎用性、などが必要である。そこで視覚、聴覚、発声、首の運動など複数の感覚を複合的に利用し、キーボードで入力キーを見ないで文字入力を行うブラインドタッチのような入力を実現するためにモールス信号を用いた入力システムを提案した。そしてこどもキーボードとソフトキーボードの **qwerty**、**abc** 配列と提案手法による文字入力時間の比較実験を行った。

モールス信号の発声による文字入力時間はジョイスティックを用いたシステムのおよそ倍になった。目と身体の負担が軽減されるのは発声によるモールスシステムである。

文字をキーボード以外で入力を行うには、一文字一文字を区別するためにモールス信号のように多くの情報がある。しかし **BS** や **ENTER** などの特殊キーは使用頻度は高いが、種類は少なく区別するための情報は少なくて済む。したがって文字入力はソフトキーボードで、特殊キーを発声によるモールスシステムで行えば、文章作成の効率が上がると思われる。

二つのモールスシステムよりも画面キーボードの方の入力速度が速い要因として慣れもあるが、操作回数の違いもある。画面キーボードの場合、全ての文字に対して入力時の操作回数は目的の文字にカーソルを合わせるために 1~2 回、クリックを 1 回の合計 2~3 回である。しかしモールスシステムの操作回数は文字によって異なり、3 回を超えるものもある。従ってモールス信号の規則をやめ、少ない信号で文字を表すことができれば、さらに文字入力速度が上がると思われる。

二つのモールスシステムの利点は、聴覚フィードバックや発声を用いることで目の負担軽減や、キーボードのように隣のキーを誤って入力するということはない。すなわち精密な操作性は必要にならず、身体の特定の部位の負担軽減にも繋がる。

謝辞

本研究を進めるにあたり、様々なご指導を頂きました三好力教授に深謝いたします。
また発表を通じて多くの示唆を頂いた三好研究室の皆様に感謝します。

参考文献

[1] 「公開特許公報フロントページ検索」

<http://www1.ipdl.inpit.go.jp/FP1/cgi-bin/FP1INIT?1347469472795>

[2] 「息で文字入力できる障害者向け Twitter 投稿 iPad アプリ「息鳥」 - インターネットコム」

<http://japan.internet.com/allnet/20110713/4.html>

[3] 「親指シフトキーボード」

<http://homepage2.nifty.com/LM/nicola/nicola.htm>

[4] 「google 検索サービス」

<http://support.google.com/websearch/bin/answer.py?hl=ja&answer=2649515>

[5] 「モールス符号 一覧表」

<http://jr7ibw.com/CW/sign.html>

付録 ソースリスト

ジョイスティックを用いたモジュールシステム

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.DirectInput;
using System.Media;

using System.Runtime.InteropServices;
using System.Threading;
namespace DirectXInput
{
    // Win32API を呼び出すためのクラス
    public class win32api
    {
        [DllImport("user32.dll")]
        public static extern uint keybd_event(byte bVlc, byte bScan, uint
dwFlags, UIntPtr dwExtralInfo);
    }

    public partial class Form1 : Form
    {
        Device joystick;

        int joy = 0, count1 = 0;
        string subst102, st = "";
        char opt;

        static bool HasString(string target, string word)
        {
            if (word == "")
                return false;
            if (target.IndexOf(word) >= 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //create joystick device.
            foreach (
                DeviceInstance di in
                Manager.GetDevices(
                    DeviceClass.GameControl,
                    EnumDevicesFlags.AttachedOnly))
            {
                joystick = new Device(di.InstanceGuid);
                break;
            }

            if (joystick == null)
            {
                //Throw exception if joystick not found.
                throw new Exception("No joystick found.ないよ");
            }

            //Set joystick axis ranges.
            foreach (DeviceObjectInstance doi in joystick.Objects)
            {
                if ((doi.ObjectId & (int)DeviceObjectTypeFlags.Axis) != 0)
                {
                    joystick.Properties.SetRange(
                        ParameterHow.ById,
                        doi.ObjectId,
                        new InputRange(-5000, 5000));
                }
            }

            //Set joystick axis mode absolute.
            joystick.Properties.AxisModeAbsolute = true;

            //set cooperative level.
            joystick.SetCooperativeLevel(
                this,
                CooperativeLevelFlags.NonExclusive |
                CooperativeLevelFlags.Background);

            //Acquire devices for capturing.
            joystick.Acquire();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            //Get
            JoystickState state = joystick.CurrentJoystickState;
            byte[] buttons = state.GetButtons();
            int[] pov = state.GetPointOfView();
            SoundPlayer wavePlayer;
            string path = null, input = @"C:\WINDOWS\Media\input.wav";

            //Output
            trackBar1.Value = state.X;
            trackBar2.Value = state.Y;
            trackBar3.Value = state.Z;
            trackBar4.Value = state.Rz;
            checkBox1.Checked = (buttons[0] > 0);
            checkBox2.Checked = (buttons[1] > 0);
            checkBox3.Checked = (buttons[2] > 0);
            checkBox4.Checked = (buttons[3] > 0);

            if (pov[0] / 100 >= 270 && pov[0] / 100 < 360)
            {
                joy = 1; st += joy;
                if (HasString(subst102, "1") == false) subst102 =
joy.ToString();
                subst102 += joy;

                path = @"C:\WINDOWS\Media\short.wav";
                wavePlayer = new SoundPlayer(path);
                wavePlayer.PlaySync();
            }
            else if (pov[0] / 100 > 0 && pov[0] / 100 <= 90)
            {
                joy = 2; st += joy;
                if (HasString(subst102, "2") == false) subst102 =
joy.ToString();
                subst102 += joy;

                path = @"C:\WINDOWS\Media\long.wav";
                wavePlayer = new SoundPlayer(path);
                wavePlayer.PlaySync();
            }
            else if (pov[0] / 100 == 0) { joy = 0; subst102 += joy; }

            if (HasString(subst102, "00000") st += "$End";
            if (st == "$End" || (HasString(st, "$End$End") == true)) st = "";

            if (HasString(st, "22222") == true)
            {
                st = "";
                path = @"C:\WINDOWS\Media\clear.wav";
                wavePlayer = new SoundPlayer(path);
                wavePlayer.PlaySync();
            }

            if (subst102.Length == 10) subst102 = "";
            if (HasString(subst102, "111111") == true) {
                subst102 = " ";
                st = "";
                count1++;
                path = @"C:\WINDOWS\Media\clear.wav";
                wavePlayer = new SoundPlayer(path);
                wavePlayer.PlaySync();
            }
        }
    }
}

```

```

}
Console.WriteLine("joy = " + joy);
Console.WriteLine("subst102 =" + subst102);
Console.WriteLine("st =" + st);

textBox3.Text = "12$End";

wavePlayer = new SoundPlayer(input);

if (count1%2 == 0)
{ textBox4.Text = "アルファベット";

    switch (st)
    {
        case null:
            opt = '0';

            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            break;
        case "12$End":
            opt = 'A';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);

            wavePlayer.PlaySync();
            st = "";
            break;
        case "2111$End":
            opt = 'B';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "2121$End":
            opt = 'C';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "211$End":
            opt = 'D';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "1$End":
            opt = 'E';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "1121$End":
            opt = 'F';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "221$End":
            opt = 'G';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "1111$End":
            opt = 'H';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "11$End":
            opt = 'I';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "1222$End":
            opt = 'J';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);

            wavePlayer.PlaySync(); st = "";
            break;
        case "212$End":
            opt = 'K';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "1211$End":
            opt = 'L';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "22$End":
            opt = 'M';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "21$End":
            opt = 'N';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "222$End":
            opt = 'O';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "1221$End":
            opt = 'P';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "2212$End":
            opt = 'Q';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "121$End":
            opt = 'R';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "111$End":
            opt = 'S';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "2$End":
            opt = 'T';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "112$End":
            opt = 'U';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "1112$End":
            opt = 'V';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
            win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
            wavePlayer.PlaySync(); st = "";
            break;
        case "122$End":
            opt = 'W';
            win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
    }
}

```



```

        wavePlayer.PlaySync();
        st = "";
        break;
    case "1222$End":
        opt = 'W';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        opt = 'O';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync();
        st = "";
        break;
    case "12121$End":
        opt = 'N';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync();
        st = "";
        break;
    default:
        //Console.WriteLine("Default case");
        break;
    }
}
}
}
private void trackBar1_Scroll(object sender, EventArgs e)
{
}
}
}
}

```

発声によるモルルスシステム

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using System.Media;
using System.Runtime.InteropServices;
using System.Threading;

using System.Diagnostics;
using LIB;

```

```

namespace MusicalVoice
{
    // Win32API を呼び出すためのクラス
    public class win32api
    {
        [DllImport("user32.dll")]
        public static extern uint keybd_event(byte bVk, byte bScan, uint dwFlags, UIntPtr dwExtralInfo);
    }
}

```

```

public partial class Form1 : Form
{
    char opt;
    string input = "0", subinput = "", test, st = "";
    int i = 0, ms=0, subcount=1;
    SoundPlayer wavePlayer;
    string path = null, input = @"C:\WINDOWS\Media\input.wav";

    static bool HasString(string target, string word)
    {
        if (word == "")
            return false;
        if (target.IndexOf(word) >= 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

```

```

public Form1()
{
    InitializeComponent();
}
//-----
const int SAMPLING_RATE = 22050; //サンプリングレート (Hz)
const int BUF_SIZE = 2048; //一回の収集サンプル数。

const int FFT_SIZE = 4096; //一回の FFT 計算サンプル数。

const int FFT_START = 13; //TICK(5.3833)*13=70Hz
const int FFT_STOP = 211; // *211=1135Hz
//音階に応じた周波数のテーブルを準備
KeyInterval mi = new KeyInterval();
//周波数を計算する準備
Frequency freq = new Frequency(SAMPLING_RATE, FFT_SIZE,
FFT_START, FFT_STOP);
//PCM データの読み込み
InPCM inPCM;
bool bDone = false;
Int16[] pcm; //2 面のバッファ
int pcm_index; //2 面のどちらを使うか順番を制御
bool bFirstTime = true; //最初に 2 面分のデータが埋まるのを待つのに
使用

```

```

int hmo = 0;

double limiter;

//-----
private void Form1_Load(object sender, EventArgs e)
{
    //タイトルバーの設定

    this.Text = App.AppTitleBar();
    //PCM 受信バッファの確保、受信準備
    pcm = new Int16[2];
    inPCM = new InPCM(SAMPLING_RATE, BUF_SIZE);
    //インタバールタイマの設定

    timer1.Interval = 100;
    timer1.Enabled = true;
    //入力レベル表示用プログレスバーの初期化

    toolStripProgressBar1.Minimum = 0;
    toolStripProgressBar1.Maximum = 120;
    toolStripProgressBar1.Value = 0;
    //表示の初期化

    toolStripComboBoxLimiter.Text = "-80";
    label1.Text = "";
    //MIDI 再生の準備
    delegate TimerProc = new WinMIDI.TimerProc(mmTimerProc);
    WinMIDI.midiOutOpen(ref hmo, -1, 0, 0, 0);
    SetIdle();
}
//-----
void RestartRecvPCM()
{
    bFirstTime = true;
    pcm_index = 0; //2 面のどちらを使うか順番を制御
    inPCM.Resume();
}
//-----
//ボタンなどの状態

enum _STATE { IDLE, PLAYING, RECORDING };
_STATE state;
void SetIdle()
{
    state = _STATE.IDLE;
    toolStripButtonStart.Enabled = true;
    toolStripButtonStop.Enabled = false;
    toolStripButtonPlay.Enabled = true;
    RestartRecvPCM(); //PCM 受信再開
}
void SetRecording()
{
    state = _STATE.RECORDING;
    toolStripButtonStart.Enabled = false;
    toolStripButtonStop.Enabled = true;
    toolStripButtonPlay.Enabled = false;
    RestartRecvPCM(); //PCM 受信再開
}
void SetPlaying()
{
    inPCM.Pause(); //PCM 受信の停止
}

```

```

state = _STATE.PLAYING;
toolStripButtonStart.Enabled = false;
toolStripButtonStop.Enabled = false;
toolStripButtonPlay.Enabled = false;
}
//-----
string old_mn = "-";
double old_power = 0.0;
int old_tick = 0;
//-----
void RecordScore(string mn, double power)
{
    if (old_mn == mn)
    {
        old_tick++;
        old_power += power;
    }
    else
    {
        double p=0.0;
        if (old_tick > 0) p = old_power / old_tick;
        textBoxList.Text += old_mn + " " + old_tick.ToString() + " " +
p.ToString("#0") + "¥¥¥\n";
        //カーソルを最後に移動してスクロール
        textBoxList.SelectionStart = textBoxList.Text.Length;
        textBoxList.Focus();
        textBoxList.ScrollToCaret();
        old_mn = mn;
        old_power = power;
        old_tick = 1;
    }
}
//-----
//閾値のコンボボックスの値の取り込み
void SetLimiterValue()
{
    if (toolStripComboBoxLimiter.SelectedIndex < 0) return;
    limiter = Convert.ToDouble(toolStripComboBoxLimiter.Text);
}
//-----
int ClearKeyTiming = 0;
double max_power = double.MinValue;
//-----
//インターバルタイマ
private void timer1_Tick(object sender, EventArgs e)
{
    if (state == _STATE.PLAYING)
    {
        //再生中
        if (bMMTimerDone)
        {
            SetIdle();
        }
        return;
    }
    //再生中以外はPCMを受信
    while (!bDone)
    {
        //PCM データをキューから取り出す
        int qc = inPCM.GetQueueCount();
        toolStripStatusLabelQueueCount.Text = "Queue:" +
qc.ToString();
        if (qc <= 0) break;//無ければ復帰
        pcm[pcm_index++] = inPCM.Read();
        pcm_index %= 2;
        //一回目のときは、2面のパuffersが埋まっていない
        if (bFirstTime)
        {
            bFirstTime = false;
            break;
        }
        //最大パワーの箇所の周波数を計算
        int first = ((pcm_index + 1) % 2) / pcm_index は、古いほうの
データを示している
        double hz = freq.GetFrequency(pcm[first], pcm[pcm_index]);
        double power = freq.GetMaxPower(); //0 から -100
        if (power > max_power) max_power = power;
        toolStripStatusLabel1.Text = max_power.ToString("#0.0") + "
" + power.ToString("#0.0");
        toolStripProgressBar1.Value = (int)(power+100.0);//プログレ
スバーに入力パワー表示
        if ((hz <= 0.0) || (power < limiter)

```

```

{
    //入力パワーなし
    // if (imput.Length == 40) imput = "0";
    ClearKeyTiming++;
    if (ClearKeyTiming > 10)
    {
        //音程表示をクリア
        textBoxKey.Text = "";
        ClearKeyTiming = 0;
        //label1.Text = "";
    }
    if (subcount == 2|| subcount == 3) imput += '0';
    else textBox3.Text = "locking!";

    subimput += '0';
}
else
{
    //入力パワーあり
    ClearKeyTiming = 0;
    int i = mi.GetScaleIndex(hz);
    textBoxKey.Text = "°";

    if (HasString(imput, "1") == false) imput += '1';
    if (HasString(imput, "3") == false && HasString(imput,
"10") == true) imput += "2";
    if ( HasString(imput, "2") == true) imput += "3";

    //if (HasString(subimput, "1") == false) subimput = "1";
    subimput += '1';

    //label1.Text = "入力 : " + hz.ToString("#0") + "Hz¥¥¥\n 定
格 : " + mi.Hz(i).ToString("#0") + "Hz";
    //if (state == _STATE.RECORDING)
    RecordScore(textBoxKey.Text, power);
}
textBoxList.Text = "imput=" + imput;
textBox2.Text = "subimput=" + subimput;
test = "11100000000111100";

int i1 = imput.IndexOf('1'), i2 = imput.IndexOf('2'), count = 0;

if (i1 == 1||imput.Length==40) imput = "0";
if (subimput.Length == 25) subimput = "0";

if (i1 > 0 && i2 > 0 && imput.Length > i2)//Substring の引数は
正数
{
    //1 と 2 の間にある 0 を数える(実際より-1 個)
    for (i = i1; i < i2-1; i++)
    {
        //i 番目が 0、かつその右隣も 0 のとき count++
        if (imput.Substring(i, 1) == "0" &&
imput.Substring(i, 1) == imput.Substring(i + 1,
1))
            count++;
        //例 1000002 なら count=4
        // i1 i2
    }
}

if (subcount == 2|| subcount == 3)
{
    //無信号が続いたときに imput をリセット
    if (imput.Length == 20 && HasString(imput, "1") == false)
    {
        imput = "0";
        st += "$End";
    }
}

// 短点
if (HasString(imput, "10000"))
{
    path = @"C:\WINDOWS¥Media¥short.wav";
    wavePlayer = new SoundPlayer(path);
    wavePlayer.PlaySync();
    imput = "0";
    st += 1;
}
// 長点

```

```

        else if (((count > 0 && count < 6) || HasString(imput, "102")) &&
HasString(imput, "30") == true)
        {
            path = @"C:\WINDOWS\Media\long.wav";
            wavePlayer = new SoundPlayer(path);
            wavePlayer.PlaySync();
            imput = "0";
            st += 2;
        }
    }

    if (subimput.Length == 25) subimput = "0";

    textBox1.Text = st;
    wavePlayer = new SoundPlayer(input);

    if (subcount == 2 || subcount == 3)
    {
        //無信号が続いたり、間違えたときに st と imput をリセット
        if (HasString(st, "$End$End") == true || HasString(subimput,
"1111111") == true)
        {
            st = "";
            imput = "0";
            path = @"C:\WINDOWS\Media\clear.wav";
            wavePlayer = new SoundPlayer(path);
            wavePlayer.PlaySync();
        }
    }
    if (HasString(subimput, "11111111111111111") == true)
    {
        path = @"C:\WINDOWS\Media\onoff.wav";
        wavePlayer = new SoundPlayer(path);
        wavePlayer.PlaySync();
        subcount++;
        subimput = "0";
    }

    //textBox3.Text = subcount.ToString();
    //モース変換 アルファベット
    if (subcount == 2)
    {
        textBox3.Text = "アルファベット";
        switch (st)
        {
            case "$End":
                st = "";
                break;
            case "12$End":
                opt = 'A';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "2111$End":
                opt = 'B';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "2121$End":
                opt = 'C';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "211$End":
                opt = 'D';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "1$End":
                opt = 'E';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "1121$End":
                opt = 'F';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "221$End":
                opt = 'G';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "1111$End":
                opt = 'H';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "11$End":
                opt = 'I';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "1222$End":
                opt = 'J';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "212$End":
                opt = 'K';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "1211$End":
                opt = 'L';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "22$End":
                opt = 'M';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "21$End":
                opt = 'N';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "222$End":
                opt = 'O';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "1221$End":
                opt = 'P';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "2212$End":
                opt = 'Q';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;
        }
    }
}

```

```

        wavePlayer.PlaySync(); st = "";
        break;

    case "121$End":
        opt = 'R';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;
    case "111$End":
        opt = 'S';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "2$End":
        opt = 'T';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "112$End":
        opt = 'U';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "1112$End":
        opt = 'V';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "122$End":
        opt = 'W';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "2112$End":
        opt = 'X';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "2122$End":
        opt = 'Y';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "2211$End":
        opt = 'Z';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "22111$End":
        opt = '7';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync(); st = "";
        break;

    case "2222$End":
        opt = 'K';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        opt = 'E';
        win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
        win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
        wavePlayer.PlaySync();
        st = "";
        break;
    default:
        //Console.WriteLine("Default case");
        break;
}

}

    }

    //平仮名変換 50音表の行と段で選択
    if (subcount == 3)
    {
        textBox3.Text = "仮名";
        switch (st)
        {
            case "$End":
                st = "";
                break;

            //あ行
            case "22122$End":
                opt = 'A';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "12$End":
                opt = 'I';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "112$End":
                opt = 'U';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "21222$End":
                opt = 'E';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "12111$End":
                opt = 'O';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            //か行
            case "1211$End":
                opt = 'K';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                opt = 'A';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "21211$End":
                opt = 'K';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                opt = 'I';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;

            case "1112$End":
                opt = 'K';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                opt = 'U';
                win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
                win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
                wavePlayer.PlaySync();
                st = "";
                break;
        }
    }
}

```



```

win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
opt = 'A';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

case "221$End":
opt = 'R';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
opt = '!';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

case "21221$End":
opt = 'R';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
opt = 'U';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

case "222$End":
opt = 'R';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
opt = 'E';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

case "1212$End":
opt = 'R';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
opt = 'O';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

//お行
case "212$End":
opt = 'W';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
opt = 'A';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

case "1222$End":
opt = 'W';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
opt = 'O';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

case "12121$End":
opt = 'N';
win32api.keybd_event((byte)opt, 0, 0, (UIntPtr)0);
win32api.keybd_event((byte)opt, 0, 2, (UIntPtr)0);
wavePlayer.PlaySync();
st = "";
break;

default:
//Console.WriteLine("Default case");
break;
}
}

if (input.Length == 40) input = "0";
if (subinput.Length == 25) subinput = "0";

if (subcount == 4) subcount = 1;

textBox4.Text = "subcount = " + subcount.ToString();
}

//-----
//マルチメディアタイマによる再生処理

string[] score;
int scoreIndex;
const int MM_TIMER_INTERVAL = 96;
WinMIDI.TimerProc delegateTimerProc;
uint timerID = 0;
int countTick = 0;
bool bMMTimerDone = false;
void mmTimerProc(uint ulID, uint uMsg, ref int dwUser, int dw1, int
dw2)
{
if (scoreIndex >= score.Length)
{
//演奏終了

WinMIDI.timeKillEvent(timerID);
bMMTimerDone = true;
}
else
{
if (countTick > 0)
{
countTick--;//同じ状態の継続
}
else
{
string[] sa = score[scoreIndex++].Split(new char[] { ' ' });
if (sa.Length > 2)
{
int count=Convert.ToInt32(sa[1]);
if (sa[0] != "-")
{
PlayMIDI(sa[0], sa[2]);//MIDI で単音を出す
}
countTick = count - 1;
}
}
}
return;
}
//-----
void PlayMIDI(string key, string volume)
{
try
{
int n = mi.MidiIndex(key);
int v = Convert.ToInt32(volume);
if (v < 0)
{
if (v < -100) v = -100;
v = 120 + v;
}
if ((v < 0) || (v > 127)) v = 60;
int shortMessage = v;
shortMessage <<= 8;
shortMessage |= n;
shortMessage <<= 8;
shortMessage |= 0x90;
WinMIDI.midiOutShortMsg(hmo, shortMessage);
}
catch
{
}
}
//-----
//Form が閉じられる
private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
{
}

```



```

        bDone = true;
    }
    //コンボボックスで閾値が変更された

    private void
toolStripComboBoxLimiter_SelectedIndexChanged(object sender, EventArgs
e)
    {
        SetLimiterValue();
    }
    //開始ボタン
private void toolStripButtonStart_Click(object sender, EventArgs e)
{
    textBoxList.Text = "";
    old_tick = 0;
    old_power = 0.0;
    old_mn = ".";
    SetRecording();
}
//停止ボタン
private void toolStripButtonStop_Click(object sender, EventArgs e)
{
    SetIdle();
}
//再生ボタン
int dwUser = 0; //ダミー。未使用
private void toolStripButtonPlay_Click(object sender, EventArgs e)
{
    score = textBoxList.Text.Split(new char[] { '\r', '\n' });
    if (score.Length <= 0) return; //エディットボックスが空
    //再生開始

    SetPlaying();
    scoreIndex = 0;
    countTick = 0;
    bMMTimerDone = false;
    WinMIDI.timeSetEvent(MM_TIMER_INTERVAL,
MM_TIMER_INTERVAL, delegateTimerProc, ref dwUser, 1);
}

private void textBoxList_TextChanged(object sender, EventArgs e)
{
}

private void textBoxKey_TextChanged(object sender, EventArgs e)
{
}
}

タブレットを用いたモールスシステム
package com.example.jaiormorse1;

import com.example.jaiormorse1.JairoMorse1.SampleSensorEventListener;

import android.R.bool;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.app.Activity;
import android.text.Layout.Alignment;
import android.view.Menu;
import android.widget.LinearLayout;
import android.widget.TextView;

public class JairoMorse1 extends Activity {
    private SensorManager manager;
    private Sensor sensor;
    private SensorEventListener sample_listener;
    String s="", morse="$ST";
    //char opt= ' ';

    public String mo="";
    int l=morse.length();

    TextView tv_x;
    TextView tv_y;
    TextView tv_z;
    TextView stext, morsetext, morseLtext, motext;
    static boolean HasString(String target, String word)
    {
        if (word == "")

```

```

        return false;
    if (target.indexOf(word) >= 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    LinearLayout ll = new LinearLayout(this);
    ll.setOrientation(LinearLayout.VERTICAL);
    setContentView(ll);
    tv_x = new TextView(this);
    tv_y = new TextView(this);
    tv_z = new TextView(this);

    stext = new TextView(this);
    stext.setTextSize(50);
    ll.addView(stext);

    morsetext = new TextView(this);
    morsetext.setTextSize(50);
    ll.addView(morsetext);

    morseLtext = new TextView(this);
    morseLtext.setTextSize(50);
    ll.addView(morseLtext);

    motext = new TextView(this);
    motext.setTextSize(70);
    ll.addView(motext);

    ll.addView(tv_x);
    ll.addView(tv_y);
    ll.addView(tv_z);
    sample_listener = new SampleSensorEventListener();
}

@Override
protected void onResume(){
    super.onResume();
    manager =
(SensorManager) getSystemService(SENSOR_SERVICE);
    sensor =
manager.getDefaultSensor( Sensor.TYPE_ACCELEROMETER );
    manager.registerListener(sample_listener, sensor,
SensorManager.SENSOR_DELAY_NORMAL );
}

@Override
protected void onPause(){
    super.onPause();
    manager.unregisterListener(sample_listener);
}

class SampleSensorEventListener implements
SensorEventListener{
    public void onSensorChanged(SensorEvent e) {
        if(e.sensor.getType() ==
Sensor.TYPE_ACCELEROMETER ){
            String str_x = "X 軸の加速度:" + e.values[0];
            tv_x.setText( str_x );
            String str_y = "Y 軸の加速度:" + e.values[1];
            tv_y.setText( str_y );
            String str_z = "Z 軸の加速度:" + e.values[2];
            tv_z.setText( str_z );

            if(e.values[1]>1&& e.values[1]<6)s+=1;
            else if(e.values[1]<-1&&
e.values[1]>-6)s+=-2;

            else if(-1<=e.values[1] &&
e.values[1]<=1)s+="";

            else if (e.values[1]>6)s+=0;

            if(s.length()==30)s="";
            if(morse.length()==10)morse="$ST";
            stext.setText(s);

```

