移動する音による図形の輪郭提示

鈴木淳也

目 次

はじめに

全盲の視覚障碍者は，視覚情報を取得出来ないため，文字や画像情報の取得が困難である．しかし，スクリーンリーダや点字ディスプレイなどのインタフェースの洗練に伴い，現在ではPCを介した独力での文字情報の受発信が可能となっている．

一方，ウェブなどでのグラフィカル情報の増加に伴い，視覚障碍者にとっても画像の認識が重要になっている．これまで，視覚障碍者が図形を認識する手段として，触覚を利用するものが多く開発されてきた．触覚は，視覚に次いで空間分解能がよく，グラフィカルな空間情報の提示に適しているからである．PCのグラフィカル情報を，触覚に提示するものとして点図ディスプレイがある．点図ディスプレイは，画面上に表示された画像や文字などの形を，点の列で表現するものである．

しかし，点図ディスプレイは，図形全体の輪郭情報は提示できるが，文字の書き順や，物体の移動などを表現したアニメーションなど，移動を伴うグラフィカル情報の場合，その動きを指で追うことが困難である．また，点図ディスプレイは，表示面積がはがきサイズ程度と小さいため，複数の人が同時に触ることができない．さらに，これら触覚を利用して図形を認識させる装置の特徴は，点が「出ている状態」か「出ていない状態」かの二値で表現することである．このためコントラストや明るさの違いを表示することはできない．この制約は，さまざまな色を表現するPCの画面を認識する装置として見過ごすことのできない制約であると考えた．コントラストや明るさの違いを区別して表示するには，表示装置が多段階の触覚提示ができなければならない．しかしながら，多段階の高さ提示を行う点図ディスプレイは実用化されておらず，他の感覚で補うなどの方法が求められる．

触覚提示では，線の形を点の配列で提示できる．これに対して，聴覚提示では，線の形を音の配列で同時には提示できない．聴覚情報は視覚情報に比べて空間分解能が低いからである．一方，時間分解能は，触覚より聴覚の方が３～４倍高いことが報告されている．そこで，筆者は，聴覚における高い時間分解能に着目し，書き順などの時間的な位置の遷移を伴う図形提示をでき，かつ複数の人が同時に図形情報を共有できる「音響式図形提示装置」を創出することを目標に研究を進めた．

まず，スピーカマトリクスを用いて同一始点から同一の音源数で任意の図形を提示したときに，どの程度正しく知覚出来るか調べた．本実験で用いた圧電スピーカマトリクスでは毎秒8パルスのインパルス列が最も効果的に図形提示が行えることを確認した．しかし，提示した輪郭の角があいまいに認識されるという問題が明らかになった．(第2章)

次に，上記の結果を受け，より広帯域の再生を可能とするダイナミックスピーカを使って,「図形の輪郭に沿って移動する音」による図形提示の可能性を検証した．検証にあたっては，単なる，○，△，□といった単純な形状ではなく，縦線，横線，斜線，曲線を含み，それらの位置関係を認識して文字を判断

する必要があるアルファベットの大文字を書き順に従って提示し，認識実験を行った．その結果，平均正答率90%以上が11文字，80%以上が20文字，70%以上が24文字であった．本実験の結果，直線，斜線，曲線を含む図形を，移動する音で提示することで認識できることを明らかにし，この手法の応用により，任意の図形を，移動する音で提示できる可能性が示された．（第3章）

次に,「音による図形提示方法」の新たな手法として，画面に描かれた線をペンタブレットを用いてなぞり，なぞった位置の線の方向を音の移動で提示することで，図形の全体像を認識できるか調べた．その結果，ペンで触れた部分の線の方向を音の移動で提示することで輪郭をたどり，全体像を認識できることを確認した．また，３点の音の逐次提示により，90°， 45°など大まかな角度を提示できることを確認した．さらに，音でベクトルや任意の２辺挟角を繰り返し提示する際，提示の終点と，次の提示の支店の間に，無音を挟むことで，始点，終点を明確にでき，線分の傾きだけでなく，方向も示すことができることがわかった．(第4章)

これまでの研究により,「視覚の空間分解能」を「聴覚の時間分解能」に変換することで，図形を認識できることを明らかにした．今後，本方式を，様々な図形やグラフの表示に活用できるように，提示スピードなどを検討していく．さらに,16行16列のスピーカ・マトリクスを試作し，より詳細な角度，曲線の提示をした場合，その情報量をどの程度認識できるか検討していく．また，色やコントラストの情報を音響情報にマッピングする検討をしていく．

本論文の構成

第1章では，視覚障碍者の図形認識の現状を述べ，解決すべき問題を提起する．その上で，本研究の新規性，有効性について述べる．また，図形の触覚提示の課題について議論する．その上で，図形の聴覚提示に着目した理由を述べる．第2章では，図形の聴覚提示の基礎検討として行った「基本図形の認識」について述べる．第3章では，第２章での結果を受け，より広大域の周波数再生を可能とするダイナミック・スピーカを使用した図形提示について述べる．第4章では，より小型の聴覚式図形提示を目標とした実験について述べ，インタラクションと聴覚フィードバックの融合作用による図形認識について述べる．第5章では，これまでの実験結果を総括し，聴覚での図形認識の可能性について述べる．さらに，今後の課題について述べる．第6章では，本研究での業績一覧を示す．第7章では，本研究でご指導，ご協力いただいた先生方，共同研究者の皆様への謝意を示す．

第1章　研究の背景と目的

1.1　本研究の背景

全盲の視覚障碍者は，視覚情報を取得出来ないため，文字や画像情報の取得が困難である．しかし，スクリーンリーダや点字ディスプレイなどのインタフェースの洗練に伴い，現在ではPCを介した独力での文字情報の受発信が可能となっている [1][2] ．文字の意味情報の認識だけであれば，音声読み上げソフトや点字ディスプレイの利用が簡便である．

1.2　これまでに開発された視覚障碍者用画像認識装置

一方，ウェブなどでのグラフィカル情報の増加に伴い，視覚障碍者にとっても画像の認識が重要になっている．代表的な視覚障碍者用画像認識装置について述べる．

オプタコン　Fig.1.1 がオプタコンである．特殊カメラで読み取られた画像を触知盤上の点（細いピン）の振動として提示するものである．

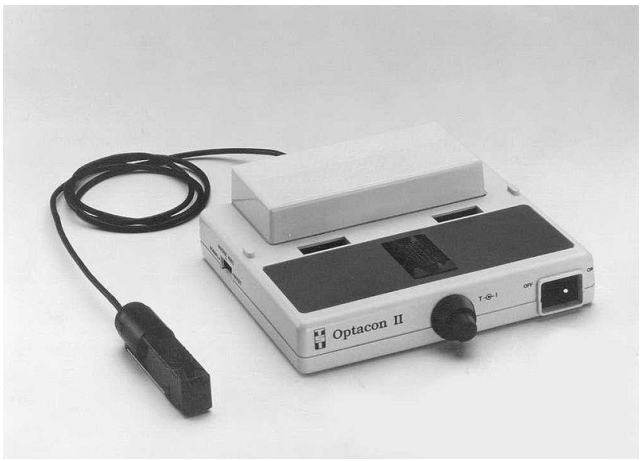


Fig. 1.1: Tactile to Optical converter

点図ディスプレイ　従来より，視覚障碍者のグラフィカルな情報の理解のためには，平面に線や面を凸状に浮き出させ，これを指で触って理解できるようにした触図が利用されてきた [3]．これを電子化したものとして点図ディスプレイが挙げられる (Fig.1.2) [5]．点図ディスプレイは，画面上に表示された画像や文字などを点の上下動を使って表現するものである．これまで，この点図ディスプレイを用いた，視覚障碍者への触覚による図形提示手法の研究が数多く行われてきた [6] [7] [8] [9] [10]．

Fig. 1.2: Dot View II

1.3　これまでに開発された装置の特徴

点の上下で輪郭線を提示
⇓
二値表現のため，
色やコントラストの表現ができない
⇓
PC の画面上で目的の図形を探せない

視覚障碍者が図形を認識する手段として，触覚を利用するものが多く開発されてきたことを述べた．触覚は，視覚に次いで空間分解能が高く，グラフィカルな空間情報の提示に適しているからである．これらの機器は，ユーザが確認したい部分にある線の形を，凸点のマトリクスで提示する．ユーザは，凸点で描かれた図形を触ることで図形を認識する．

1.4　凸点による触覚提示の課題

視覚障碍者が図形を触覚で認識する手段として，オプタコン，点図ディスプレイが開発されてきたことを述べた．いずれの装置も，指先に触れた部分の線の方向を確認することで輪郭をたどり，全体像を認識できる．これら触覚を利用して図形を認識させる装置に共通している特徴は，点が，”出ている状態”か”出ていない状態”かの二値で表現することである．このためコントラストや明るさの違いを区別して表示することはできない．この制約は，コンピュータ画面などを認識する装置として見過ごすことのできない制約であると考えた．コントラストや明るさの違いを区別して表示するには，表示装置が多段階の触覚提示ができなければならない．しかしながら，多段階の高さ提示を行う触覚ディスプレイは実用化されておらず，他の感覚で補うなどの方法が求められる．また，点図ディスプレイは，表示面積がはがきサイズ程度と小さいため，複数の人が同時に触ることができない．さらに，図形全体の輪郭情報は提示できるが，文字の書き順や，物体の移動などを表現したアニメーションなど，移動を伴うグラフィカル情報の場合，その動きを指で追うことが困難である．

多段階の点の高さ　　多段階の点の太さ
⇓　　⇓
機器の巨大化　　機器の精密化
⇓　　⇓
高価な機器となってしまい，普及が進まない
⇑
機器の値段が抑えられる，多彩な画像情報の表現が可能な手法が必要

1.5　本研究の目的

そこで，本研究では，図形の触覚提示ではなしえない，下記の項目の実現を目的に，「音を活用した図形提示」を試みてきた．

1. 図形の全体像の迅速な把握
2. 図形の動きの認識
3. 多人数での同時認識
4. 色やコントラストの提示

これまでの研究では，上記 1. と 2. の解決を目的に実験を重ねてきた．

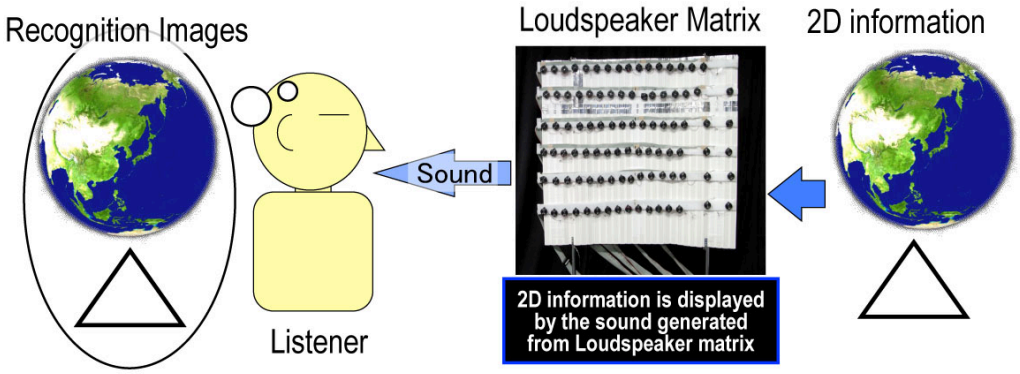


Fig. 1.3: Concept of the loudspeaker matrix which shows two-dimensional patterns

1.6 音で図形を提示する装置のコンセプト

多くの視覚障碍者は日常的に音で空間認知ができ，晴眼者よりも空間認知の精度が高いことが知られている [8, 11] ．この能力を活用して，音による図形提示ができれば，既存の方法よりも豊富な画像表現が可能となる．

触覚の場合は，線の形をピンの配列で同時に提示できる．これに対して，聴覚の場合は，線の形を音の配列で同時には提示できない．聴覚情報は視覚情報に比べて空間分解能が低いからである．一方，時間分解能は，触覚より聴覚の方が ３～４倍高いことが報告されている [11] ．

そこで，筆者は，聴覚における高い時間分解能に着目し，複数の人が同時に図形情報を共有でき，かつ書き順などの時間的な位置の連続的または不連続な遷移を伴う図形提示を可能にする「音響式図形提示装置」を創出することを目標に研究を進めた．Fig.1.3 に，音で図形を提示する装置のコンセプトを示す．

目的： 色やコントラストなどを伝達可能な視覚障碍者用の図形提示機器の開発

着目点： 聴覚の場合，音階や音色など、多段階で認識できるパラメタがある

1.7 本研究が完成した際に期待される応用分野

空間分解能は触覚提示より低下するが，多人数が同時に図形情報を認識できる黒板への応用や，時間分解能を生かしたグラフの高速提示への展開が考えられる．また，音圧，周波数の音の弁別閾に着目し，さまざまな色に対応しうる音響的画像提示装置を目標として，筆者は研究を進めてきた．

- 視覚障碍者への図形提示，図形描画支援
- 3D オーディオシステム
- 音を活用したゲーム

1.8　視覚と聴覚の分解能

ここで，図形情報（視覚情報）を音情報（聴覚情報）に変換するに当たり，各刺激の分解能の特性を整理しておく．

ヒトの眼の分解能

空間分解能: 視野角にして 1/60 度

時間分解能: 数 10Hz

ヒトの耳の分解能 Green (1971) の実験

音の時間的な変化を捉えられる最短時間
⇓
12ms 程度

ヒトの耳の分解能 Hirsch および Hirsh and Sherrick

高さや，到来方向の異なる二つの音でどのくらいの時間差があれば前後が知覚されるか？
⇓
20 ms の時間差
聴覚に限らず
視覚，触覚においても同程度の値
共通の時間知覚の仕組みが働いていることを示唆

1.9　空間音響に関する知見の整理

音の動きで図形を表現するに当たり，音の持つ性質について整理することにする．

音を聞くとは　空間において，音源から放射された音響信号は境界面 (壁や天井) による反射や散乱や回折等の影響を表す空間伝達関数 の影響を受けて受聴者の位置に到達する．この音響信号は頭部の影響を表す頭部伝達関数 (HRTF :head-related transfer function) の影響を受けて受聴者の左右の外耳道入口 (entrance of external ear canal) に到達し耳入力信号 (ear input signal) となる．この信号は，受聴者の聴覚器官への入力すなわち心理空間への入力である音刺激 (acoustic stimulus) となる．

心理空間において，受聴者は次に示す 3 つの性質に大別できる様々な要素感覚をもった音像 (sound image) を知覚する．

時間的性質: 残響感，リズム感持続感など

空間的性質: 方向感，距離感，広がり感など

質的性質: 大きさ，高さ，音色など

受聴者はそれぞれの要素感覚に対して個人の嗜好に基づいて主観評価を行い，さらに，各要素感覚の主観評価を再び個人の嗜好に基づいて重み付け，それらを統合し総合評価を下す.

音源と音像　音響信号を発するものを「音源」，発せられた音響信号によって受聴者が知覚するものを「音像」という用語を用いて記述している．方向感についても，音響信号の性質によっては，音源の方向と音像の方向が一致しないことがしばしば見られる.「正しい方向定位」という表現がよく使われるが，これは,

1. 音響信号を発した音源の方向と, 受聴者が知覚した音像の方向とが一致する場合

2. 音源の方向に関係なく, 頭部伝達関数のたたみこみなどによってシミュレーションした方向に音像を知覚できる場合

の 2 つの解釈が可能である.

左右方向の知覚　ヒトの両耳は頭部の両側についているので，音が側方から入射すると両耳への到達時間および音圧に差が生じる．左右の方向知覚の手がかりは，このような頭部伝達関数により生じる両耳間差情報，すなわち両耳間時間差 (ITD:interaural time difference) および両耳間レベル差 (ILD :interaural level difference) であることが古くから知られている [15] ．ただし，両耳への入力信号そのものの時間差が左右方向の知覚の手がかりになっているのは，約 1600 Hz 以下に限られる [16] [17] ．それ以上の周波数帯域では，両耳間レベル差が方向知覚の手がかりになっている．両耳間レベル差を工学的に応用したものがステレオシステムである．ステレオシステムで, ある音を定位させる場合には，その音の信号を 2 つのチャネルにレベル差を付けてかつ同相で（すなわち時間差を付けずに）振り分ける．こうすると，2 つのスピーカから各耳に音波が到達する際に音波が合成されて，両耳間レベル差と両耳間時間差が生成される [18] [19] .

上下方向の知覚　ヒトの上下方向の定位精度は左右方向と比較して低く，日常生活においてもしばしば誤判定が生じる．上下方向の知覚の手がかりは，左右方向のように単純ではなく，また頑健でもない.

方向知覚の弁別限　ヒトは，どの程度正確に方向を知覚できるのだろうか．水平面にある音源に対する方向知覚の弁別限に関する研究は数多くあるが，その

結果は，音源方向が正面の場合に最小となり，側方で大きく，後方で再び小さくなることで一致している．

黒澤らは前方及び後方の音源に対する弁別能力を調べ，前方及び後方の方位角の弁別限は1°程度であること，側方は10°程度であることを示し，さらに，仰角 45°付近では，仰角方向の弁別限が正中面だと5°以下，横断面だと2°以下であることを示している [20] ．このように，方向知覚の弁別限は音源方向に強く依存している．

音の時系列提示の性質: 第 1 波面の法則　2つのスピーカで，同一信号を同時に同レベルで放射すれば音像は正面に生じる．ここで，片方の信号に遅れ時間を加えると，音像は遅れ時間に応じて先に信号を放射しているスピーカのほうへ徐々に移動し，遅れ時間が約 1ms になると，先に放射しているスピーカの方向に生じる．さらに遅れ時間が増加しても，音像は先に放射したスピーカ方向のままである．このように，先行音と後続音が異なる方向からある時間間隔で入射した場合，受聴者は先行音の入射方向にのみ音像を知覚する．この現象に関する実験結果は1950年前後に Cremer [14] , Wallach et al. [15] , Haas [21] により相次いで報告され，第1波面の法則 (the low of the flrst wave front), あるいは先行音効果 (the precedence effect) と呼ばれている．この法則が成立する範囲内であれば，音像の方向を先行音方向に保ちながら後続音により受聴音圧を増強することが可能である．第1波面の法則が成立する最小遅れ時間は合成音像との境界，すなわち約 1ms である．

聴覚のマスキング効果　騒音下では，話が聴きとりにくかったり，他のすべての音がかき消されてしまうことがある．このような現象を「聴覚のマスキング」と呼び，音が騒音でマスクされたという．マスキングの程度を表すには，マスクする音があるときのその音 (マスクされる音) の最小可聴限とマスクする音のない静寂時の最小可聴限との音圧レベル差，すなわち最小可聴限の移動量で表し，これをマスキング量という．

マスキングはマスクする音とマスクされる音が同時に加えられない場合にも起きる．これを継時マスキング (temporal masking) と呼ぶ [22] [23] [24] [25] [26] ．加えられていたマスキング音を止めた場合，マスキング音による感覚は急激に消滅するのではなく，わずかの時間ではあるが徐々に減少して約200 ms で消滅する．その間に第2音が与えられた場合には，その時刻におけるマスキング音の感覚に応じたマスキングを受けるわけである．このように，先行する第1音によるマスキングを前向性マスキング (forward masking) という．また，先行する大きな音が後続の小さな音をマスクするばかりでなく，後続の大きな音が先行した小さな音をマスクする場合がある．これを後向性マスキング (backward masking) という．すなわち，時間をさかのぼってマスクするわけである．前向性のマスキングのほうが，後向性よりも時間的に離れた音にもマスキングが起きるが，マスキング音のごく近くでは，後向性マスキングのほうがマスキング量が大きい．継時マスキングを聴覚神経系で行われる現象とし

てとらえてみると次のように考えられる．前向性マスキングでは，先行したマスキング音による神経活動がしだいにおさまっていく過程で，小さな第 2 音に対する神経活動が生じている状態であって，第 2 音の活動が前者の中に埋もれてしまって区別ができなければ，第 2 音に対する感覚も生じないことは容易に推察できる．後向性マスキングで，時間をさかのぼってマスクすることは，神経のもつ潜時 (刺激が加わってから神経が発火するまでのおくれ時間) に起因する．刺激が加わって神経が応答するまでには多少の時間おくれがある．この潜時は一定の時間ではなく，刺激が強ければ短く，刺激が弱ければ長くなるのがふつうである．シナプスによる接続回数の多くなる上位の神経核ほど，刺激の強さの違いによる潜時の差が大きくなっている．マスクする音は，マスクされる音に比べて非常に大きいので，両者の潜時の差は大きく，聴覚末梢から上位の神経核をとおっているうちに，マスキング音による神経活動が先行のマスクされる音の神経活動に追いついて，両者を区別できないようにマスクしてしまうものと考えられる．継時マスキングは日常の聴こえと縁遠いものと思われるかもしれないが，日常の音としては絶えず変動している音に接しているのであるから，継時マスキングは常に密接な関係がある．また，聴覚の動的性質を知る上にも重要な現象である．

参考文献

[1] 渡辺 哲也, 岡田 伸一, 伊福部 達, ”GUIに対応した視覚障害者用スクリーンリーダの設計,” 電子情報通信学会論文誌D-II, Vol.J81-D-II, No.1, pp.137-145, 1998.

[2] 渡辺 哲也, ”視覚障害者のパソコン・インターネット・携帯電話 利用状況調査 2007”, 2008.

[3] 北林裕，青木成美，” 視覚障碍者の支援機器の現状と課題 ”，障害者職業総合センター研究部門 調査報告書No.18 第5章，pp.58–67，(Mar 1997)

[4] 点図ディスプレイ ドットビュー DV-2 http://www.kgs-jpn.co.jp/b_dv2.html

[5] 渡辺哲也，久米祐一郎，伊福部達:触覚マウスによる図形情報の識別，映像情報学会誌, Vol.54, No.6, pp.840–847, (June 2000)

[6] 小林真，” 触覚ディスプレイによる視覚障碍者用エンタテインメントシステム ”，知能と情報：日本情報ファジィ学会誌：journal of Japan Society for Fuzzy Theory and Intelligent In- formatics 16(6)，pp.492–497，(Dec 2004)

[7] 渡辺哲也，須貝克美，為近哲夫，” グラフィカル情報提示のための触覚ディスプレイシステムに関する研究 ”，日本障害者雇用促進協会障害者職業総合センター 調査研究報告書，No.41，(May 2001)

[8] 大山正，今井省吾，和気典二，” 空間的分解能と受容器の密度 ”，新編 感覚・知覚 心理学ハンドブック，第?部，2.4.5，pp.1184，(1994)

[9] 林 大作, 荻原 貴文, 遠西 学, 中村 直人 点図ディスプレイを用いた触図図形提示方法の検討 (インタフェース技術と学習支援システム/一般) 電子情報通信学会技術研究報告. ET, 教育工学 110(334), pp.53–56, (2010–12–03)

[10] 関 喜一, ”視覚障害者にやさしい街の音創り”, 日本音響学会誌, pp.387-392, 1998.

[11] T. Miura, T. Muraoka, and T. Ifukube, ”Comparison of obstacle sense ability between the blind and the sighted -a basic psychophysical study for designs of acoustic assistive devices-.”, Acoustical Science and Technology, 2009.

[12] 伊福部達:"AM 音によるマスキングとそのシュミレーション", 音響会誌, 31, 4, pp.237–245 (1975)

[13] Lord Rayleigh : On our perception of sound direction, Phil. Mag. 13, 6th series, pp.214-232 (1907)

[14] L. Cremer: Die wissenschaftlichen Grundlagen der Raumakustik, 1, p.126, S. Hirzel (1948)

[15] H. Wallach, E. B. Newman and M. R. Rosenzweig:The precedence effect in sound localization, Am. J. Psychol., 52, pp.315-336 (1949)

[16] F.Rumsey, "Spatial Audio," Focal Press (Elsevier) (2001)

[17] W. Snow, "Basic Principles of Stereophonic Sound," Journal of SMPTE Vol.61, pp.567-589 (1953)

[18] A. D. Blumlein, "Improvement in and relating to Sound-transmission, Sound-recorfing and Sound-reproducing Systems," British Patent Specification 394325 (1931) あるいは "An anthology of reprinted articles on stereophonic techniques," AES, pp.32-40 (1986) .

[19] H. A. M. Clark, G. F. Dutton and P. B. Vanderlyn : The ‘ stereophonic ’ recording and reproducing system. - A Two-channel system for domestic tape records, Journal of AES, Vol.6, no.2, pp.102-117 (1958).

[20] 黒澤明，都木徹，山口善司，“頭部伝達関数と方向弁別能力について，” 日本音響学会誌. 38(3), pp.145-151, 1982.

[21] H. Haas : ""U"ber den Einfluss eines Einfachechos auf die H6rsamkeit von Sprache, ACUSTICA, 1, pp.49-58 (1951)

[22] D. H. Raab: Forward and backward masking between acoustic clicks, J. Acoust. S0C. Am., 33, 137, (1961).

[23] L. L. Elliot Backward and forward masking of probe tones of different frequencies, J. Acoust. SOC. Am., 34, 1116, (1962).

[24] L. L. Elliot Backward masking:Monotonic and dichotic conditions, J, Acoust. SOC. Am., 34, 1108, (1962).

[25] H.N. Wright remporal summation and backward masking, J. Acoust. soc. Am., 36, 927, (1964).

[26] H.Fastl Temporal masking effects: I. Broad band noise masker, Acustica, 35,287, (1976).

第2章　圧電スピーカマトリックスを用いた音源の移動による基本図形の輪郭提示

2.1　背景と目的

本研究の共同研究者の伊福部は，頭部を覆う球面状のスピーカマトリクスを用いて，音源定位による文字認識に関して調査をしている [1, 2, 3]．この際，5文字程度の仮名文字パタンを提示し，1秒間に3〜4個までを90%程度の識別率で伝達可能という結果を得ている [2]．しかし，上記の実験の場合は始点位置や音を出すスピーカ数が異なるなど，音源定位以外の手がかりが多かった．そのため，音源定位だけでどの程度細かい空間情報を提示出来るか分からない．

そこで，スピーカマトリクスを用いて同一始点から同一の音源数で任意の図形を提示したときに，どの程度正しく知覚出来るか調べた．

2.2　図形パタンの判断実験

2.2.1　被験者

被験者は，晴眼者3名 (男性2名，女性1名, 平均33.0歳)，視覚障碍者1名 (男性1名, 42歳) である．全ての被験者は，オージオメータ (Rion AA-75) による純音聴力検査の結果，4分法で20dB HL以下であった．また，実験の際は目隠しをさせ，視覚情報を遮断した．

2.2.2　実験装置

実験は聴覚検査室 (RION AT-81) 内で行われた．壁から0.5mの所に被験者用の椅子が配置してある．壁には，被験者の頭の高さにスピーカマトリクスを配置したパネル (以下，刺激図形) を設置できるようになっている．

刺激図形のサイズは縦40cm〜横40cmである．その範囲に内接する円，正三角形および正方形を描き，各刺激図形の辺上に24個の圧電スピーカ (村田製作所:PKM34EWH1201C, 直径3.5cm) を等間隔に配置した (全3種類, Fig.2.1参照)．なお，実験に使用したスピーカは，周波数特性のばらつきの少ないものを使用した．

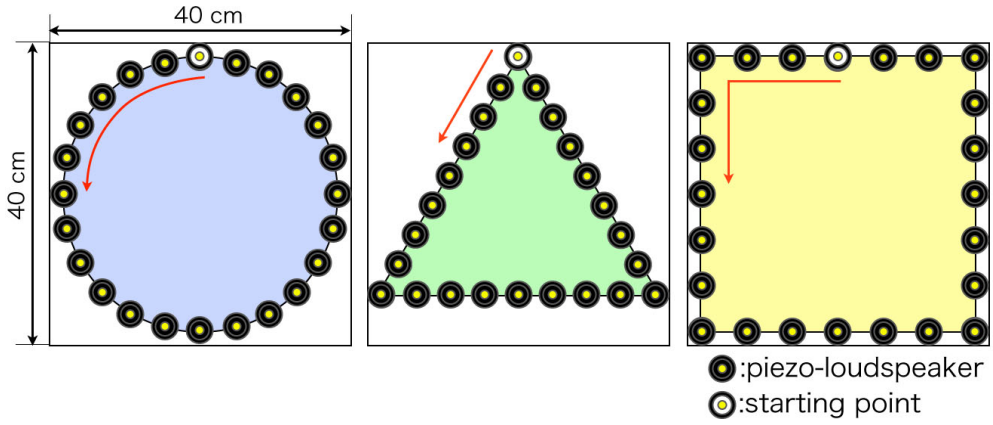


Fig. 2.1: Stimuli diagrams composed of loudspeakers

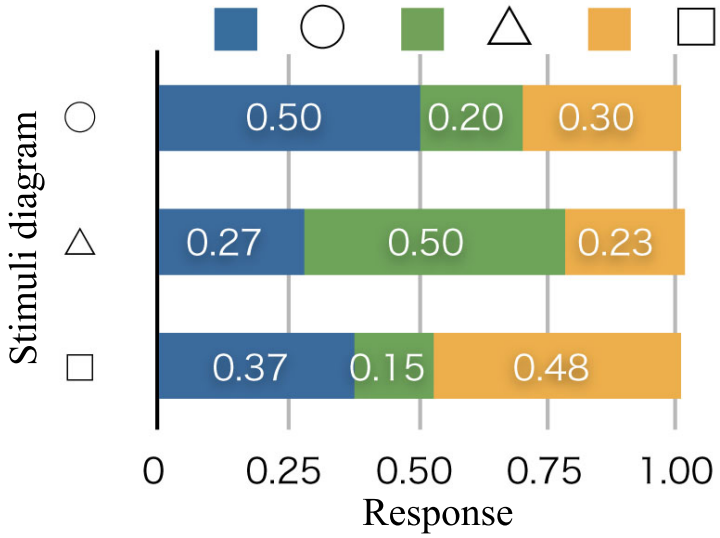


Fig. 2.2: Confusion matrix of the determination experiment for stimuli diagrams

これら圧電スピーカは，各スピーカマトリクスごとに 3 台のオーディオ・インターフェース Presonus FP10 を経由して PC(Matlab) から制御した．

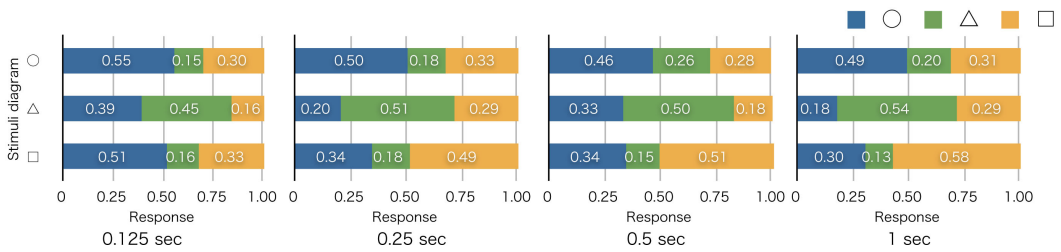


Fig. 2.3: Confusion matrices of the determination experiment for stimuli diagrams; per sound duration

2.2.3 刺激図形

スピーカマトリクスからの音の提示に当たっては，Fig.2.1 の中白の点から半時計回りの順に一つ一つのスピーカから逐次的に同一音を出した後，もう一度中白の点から音を出した．

刺激音はホワイトノイズ (周波数帯域 : 20-20kHz)，3 種類のインパルス列 (4, 8, 16 [pulses/sec]) である．これらのインパルス列は，人間が正しく数えられる回数 (4)，正しく数えるのが困難になる回数 (8)，正しく数えられない回数 (16) を想定している．また，スピーカ一つ辺りの提示時間 (以下，スピーカ提示時間) は 0.125, 0.25, 0.5, 1.0 sec である．全てのスピーカから提示する音圧は一定値 (概ね 60 dB) に揃えてあり，またスピーカからスピーカへの遷移時間は 0 sec とした．

2.2.4 実験手順

本実験の前に，3 種類の刺激図形を被験者にパタン A, B, C として聞かせる (実験前提示)．この際，パタン A, B, C がどの図形であるかは教示せず，音を聞いてスピーカ列の空間的な配置を意識して聞くよう教示した．

まず，1) 実験者が 3 種類の刺激図形のうちの 1 つを，被験者の頭部中央と刺激図形の中央が一致するよう，0.5m 離れた壁に配置する．次に，2) 刺激図形に配置されたスピーカマトリクスから音を提示する．実験前提示の段階では，スピーカ提示時間は 0.5 sec で一定とし，被験者は何度でもこれら刺激図形を聞く事ができるようにした．

実験前提示のあとに，練習をはさまずに本番を行った．手順は実験前提示の 1, 2 のあとに，3) 被験者に A,B,C のいずれであるかを，それぞれ対応するテンキーのボタン 4, 5, 6 で応答させた．この際，スピーカ提示時間は可変であり，被験者は一度しか刺激図形を聞く事ができないようにした．被験者には，音提示されている間，声などの音を発しないよう，所定の位置から動かないよう教示した．

実験セットは刺激音ごとに作成し，各刺激図形につき 5 回ずつ提示するようにした．したがって合計提示回数は 4(刺激音) × (3(刺激図形)×4(提示時間)×5(回数)) = 240 回である．

2.2.5 結果

全体的な傾向　刺激音，スピーカ提示時間ごとのグラフをまとめた結果を Fig.2.2 に示す．縦軸は刺激図形の種類であり，横軸は図形ごとの応答率を示す．青は円，緑は正三角形，橙は正方形としての応答率を示す．この結果を見ると，概ねどの図形に関しても約半分は正しく応答出来ていたことが分かる．ただし，誤応答に注目すると，円に対しては正方形と，正方形に対しては円と応答する場合が多かった．この結果は, 角があいまいに認識されたことを表している．

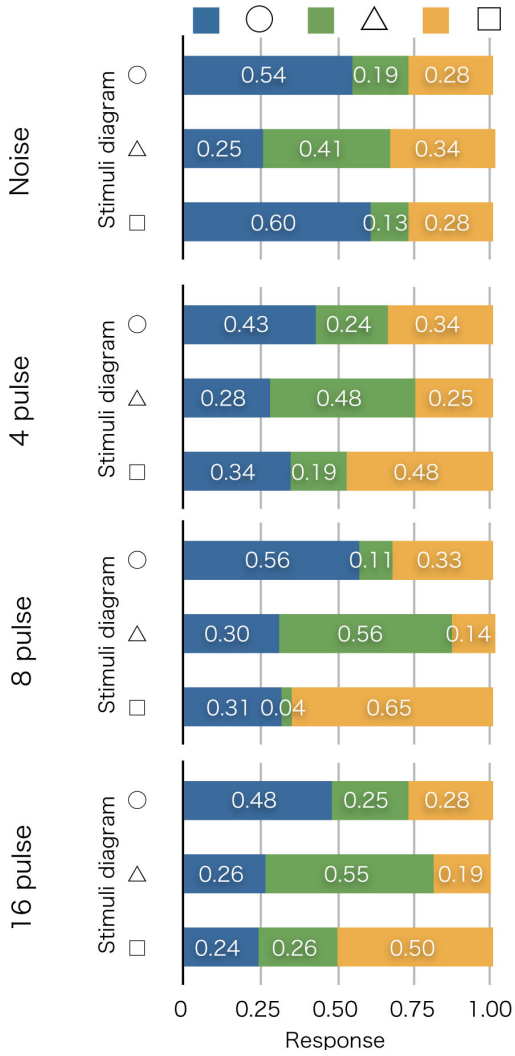


Fig. 2.4: Confusion matrices of the determination experiment for stimuli diagrams; per sound kind

刺激音ごと，スピーカ提示時間ごとの傾向　刺激音ごと，もしくはスピーカ提示時間ごとに平均化したグラフをそれぞれ Figs.2.3 および 2.4 に示す．Fig.2.3 を見ると，スピーカ提示時間が長くなるほど，刺激図形に対する応答が正しくなされた．一方で，スピーカ提示時間が短くなると，刺激図形が円の場合は正答率，誤答率に大きな変化は見られなかったが，角のある刺激図形を円として誤応答する場合が増加した．特に，0.125 sec のときは，正方形を円として応答する場合が多くなった．

Fig. 2.4 を見ると，ホワイトノイズと比較して，インパルス列の方が角のある図形で高い正答率を得られた．ホワイトノイズの場合，正方形に対しての応答が悪くなり，円としての応答率が顕著に高い．一方で，インパルス列の回数に関して比較すると，$8 > 16 > 4$ [times/sec] の順で刺激図形に対する正答率が高かった．ただ，刺激図形が正方形のときの円と誤答する率を見ると，インパルス列の回数が多い程，誤答率が減少した．刺激図形が正三角形の場合，刺

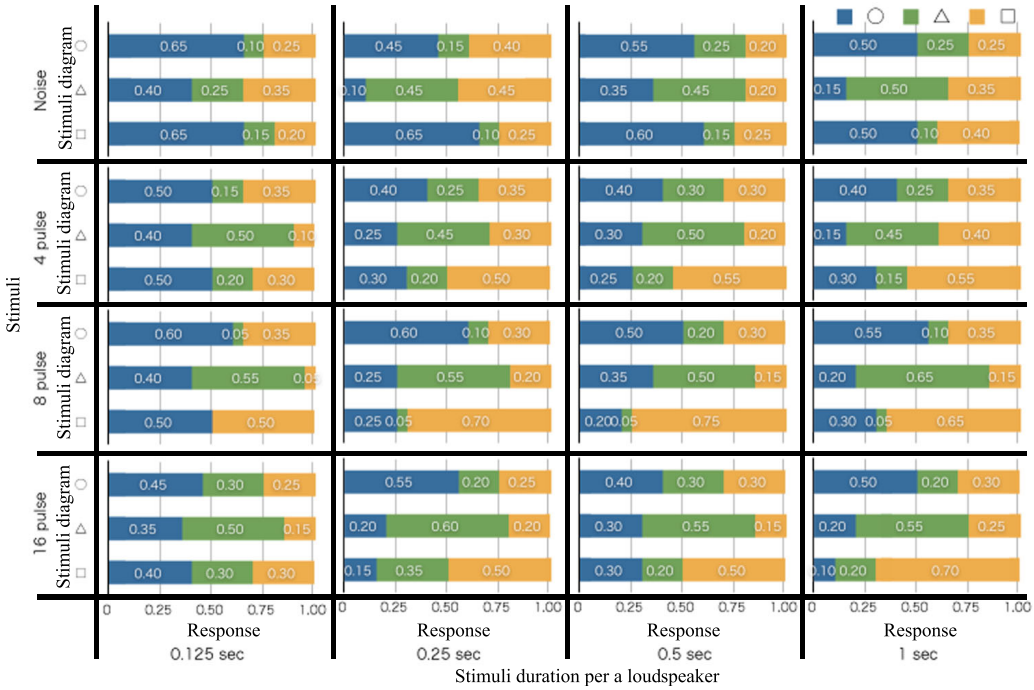


Fig. 2.5: Confusion matrices of the determination experiment for stimuli diagrams

激音ごとの顕著な傾向の変化は見受けられなかった.

各刺激セットに対する傾向　Fig. 2.5 に刺激音およびスピーカ提示時間ごとのコンフュージョンマトリクスを示す．これらのグラフは，横列は刺激音ごと，縦行はスピーカ一つ辺りの提示時間 (以下，スピーカ提示時間) ごとに並べてある.

この結果を見ると，どの刺激音についてもスピーカ提示時間が短くなると，角のある刺激図形に対して円として誤応答されることが多くなり，スピーカ提示時間が長くなると，角のある刺激図形に対する正答が増える傾向が見受けられる．ただし，ホワイトノイズについては，スピーカ提示時間が長くなっても，刺激図形が正方形であるときに正しく応答出来る率がインパルス列に比べて大きくはならなかった.

次に，インパルス列間で比較する．8 times/sec の場合，刺激図形が正方形であるときに正しく応答出来る率が大きかった．一方で，刺激図形が正方形であるときの誤答に着目すると，4 および 8 times/sec のときは円として誤答するのが多いのに対し，16 times/sec のときは正三角形と誤答することが多かった．刺激図形が円のときに着目すると，4, 16 times/sec のときと比較して 8 times/sec のときが最も正答率が高かった.

次に，刺激音が 8 times/sec のインパルス列であるときに着目する．2.2.5 節で得られた傾向と同様，スピーカ提示時間が長くなるほど，刺激図形を角のある図形として応答する場合が増加している．また，単純な正答率の平均で比

較すると，スピーカ提示時間において 0.25 = 1.0 > 0.5 > 0.125 [sec] の順となった．

被験者群ごとの傾向　Figs.2.6 に，それぞれ 2 被験者群の刺激音，スピーカ提示時間ごとの正答率を示す．ただし，エラーバーのうち晴眼者群 (sighted) のものは最大もしくは最小の正答率，全数 (all) のものは標準偏差を意味する．今回の実験では，スピーカ提示時間が 0.125 sec のときを除いて，視覚障碍者群，晴眼者群に大きな差異は確認出来なかった．

ただし，全被験者群内での正答率のばらつきは刺激音によって大きく違いが見受けられた (Fig.2.6)．インパルス列ごとに見ていくと，ばらつきの大きさは 4 > 16 > 8 [times/sec] の順で確認された．ホワイトノイズは最もばらつきは小さくなっていたが，正答率が最も低かった．

また，スピーカ提示時間が長くなるほど，概ねばらつきが大きくなった (Fig.2.7)．ただし，正答率はスピーカ提示時間が短い方が低かったことから，スピーカ提示時間が長い場合，被験者によっては正しく応答しやすい条件になったといえる．

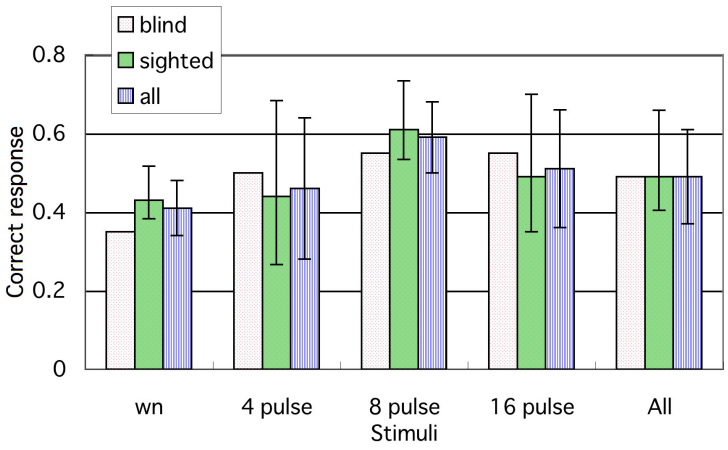


Fig. 2.6: Correct response of blind and sighted groups in each sound stimuli

2.2.6　考察

2.2.5 節にて，刺激音がホワイトノイズの際は，角のある刺激図形に対して円と誤答する場合が多かったと述べた．これは，インパルス列よりもホワイトノイズの方が音源定位に十分なスペクトル情報が含まれていないからと想定出来る．

一方で，スピーカ提示時間が短くなると，角のある刺激図形に対して円と誤答する場合が多かった．これに関して，Pollack は空間音響における両耳受聴信号処理に関連して，時定数 T_1 があることを結論付けている [4]．具体的には，ある音源から異なる音源へ連続的にスイッチする間隔 (本実験ではスピーカ提

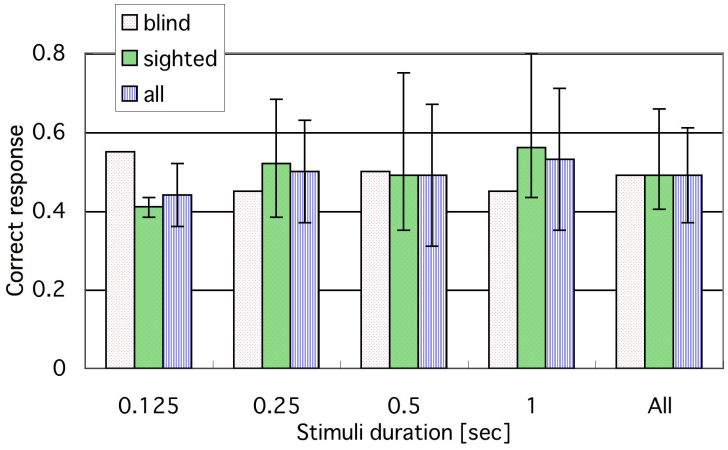


Fig. 2.7: Correct response of blind and sighted groups in each duration

示時間) が長いとき，被験者は鋭く定位する音像と広がった音像を交互に知覚する．これに対して，スイッチ間隔が短くなると，ただ一つの融合した音像が知覚されるようになり，スイッチ間隔が短くなればなるほど音像の広がりが増す．本実験結果に当てはめると，スピーカ提示時間が 0.125 sec のときは，音像が広がって聞こえてしまった可能性がある．今回の実験では，スピーカ提示時間は一定としていた．しかし，角の部分だけを長めに設定することで，全体として短い時間で提示する場合でも角のある図形として認識させられる可能性がある．

なお，スピーカ提示時間が 0.125 sec のとき，二被験者群間の正答率の差が最も大きくなった (Fig.2.7)．一般に視覚障碍者は聴覚提示された信号の時間分解能が晴眼者に比較して高いと報告されている [5, 6]．今回の実験では被験者数が少ないため，多くを述べることは出来ないが，時間分解能の高さの結果，スピーカ提示時間が短くなっても，鋭い音源定位，音像定位が出来たと推察出来る．

今回実験を行った範囲では，[刺激音: 8 times/sec, スピーカ提示時間: 0.25 もしくは 1.0 sec] のとき，最も正答率が高かった．しかし被験者によっては，スピーカ提示時間が 1.0 sec の場合，提示時間が長すぎて軌跡のはじめの部分を忘れてしまうとの内観を報告したものがあった．したがって，本実験での条件では [刺激音: 8 times/sec, スピーカ提示時間: 0.25 sec] の場合がスピーカマトリクスによる形状提示に最適であると結論出来る．

参考文献

[1] 伊福部 達, ”音の福祉工学”, コロナ社, pp:192-197, 1997.

[2] 伊福部 達, 吉本 千禎, ”聴覚の音源定位による文字認知”, ET78-13, 1979.

[3] 伊福部 達ら, ”感覚代行のための多音像定位機能の解析”, 日本音響学会 聴覚研究会資料, H-46-3, 1977.

[4] I. Pollack, ”Temporal switching between binaural information sources”, J Acoust Soc Am, 63(2):550-558, 1978.

[5] M. Bross and M. Borenstein, ”Temporal auditory acuity in blind and sighted subjects: a signal detection analysis”, Percept Mot Skills, 55(3 Pt 1):963–966, 1982 Dec.

[6] F. Gougoux, F. Lepore, M. Lassonde, P. Voss, R. J. Zatorre, and P. Belin. ”Pitch discrimination in the early blind”, Nature, 430(6997):309, 2004 Jul 15.

第3章　高音質スピーカマトリックスを用いた音源の移動によるアルファベット形状の提示

3.1　背景と目的

従来，文字や図形を音響刺激に変換する方法は種々考案されているが，Lexiphoneの研究に見られるように，その多くは，両耳ヘッドホンを介して，水平位置を左右の音量差で，垂直位置を音の高低変化で認知させるものである [1] ．この場合，水平位置は，左右の音量差による音の位置の変化で直感的に認識できるが，垂直位置は，周波数の高低変化で表すため，周波数情報を位置情報に解釈しなおす必要があり，形状を直感的に認識できない．また，伊福部らは，小型のスピーカを耳介を覆うように多数装着させ，文字認知実験を行い，一文字当りの提示時間と識別率との関係を調べた．提示時間が1秒以上では100 %近い識別率が得られたが，提示時間が短くなると，音の位置や移動方向の認知が難しくなるため，識別率は徐々に低下したと報告している [2] ．この実験では，聴覚の空間分解能が左右方向に比べて上下方向が著しく悪いことを考慮して，文字を90°回転して提示している．このため，聞き取った輪郭を90°回転し直して解釈する必要があり，図形提示方法として実用的ではない．また，小型のスピーカを耳介を覆うように多数装着させ音を提示しているため，受聴位置が狭く，一人ずつしか聴取できない．その後，この実験のように，多数のスピーカから音を逐次的に提示することは，正確な制御が難しいため，これまで伊福部ら以外，ほとんど試みられてこなかった．

石川らは，視覚障碍者の墨字学習を支援するために，点図ディスプレイと音声ガイドを用いた墨字の筆順学習システムを試作した．このシステムでは，点図ディスプレイに墨字を一文字丸ごと描画させたり，墨字を一筆ずつ描画させ，それを学習者が手でなぞることにより，触覚で墨字の筆順，偏や旁などを学ぶことができる．学習の際に，墨字の読み情報や操作方法などを音声情報として学習者に伝えることにより，視覚障碍者にとって使いやすいシステムを目指した [3] ．しかし，文字の各部分は認識できるものの，その文字の中での各部分の位置関係を把握することは困難であった．

そこで本論文では，音の移動による図形認識の可能性を検証するために，スピーカ・マトリクスを試作し，時系列的な音の発生 (本論文では“移動する音”とよぶ) による図形提示の可能性を検証する．近年，LEDのマトリクスを用い

た文字表示が多用されている [4] . これらは，１文字当たり，８行８列や，16 行 16 列や，24 行 24 列の LED マトリクス・ユニットが用いられている. そこで, 本実験では, アルファベット表示の可能な 8 行 8 列（計 64 個）に小型スピーカを配置した. 検証にあたっては，単なる，○，△，□といった単純な形状ではなく，縦線，横線，斜線，曲線を含み，それらの位置関係を認識して文字を判断する必要があるアルファベットの大文字を書き順に従って提示し，認識実験を行った.

3.2　アルファベット形状の判別実験

3.2.1　被験者

本実験の被験者は，15 名（20 代，男性 14 名，女性 1 名の晴眼者）である. いずれの被験者も，オージオメータによる 4 分法聴力検査により，正常な聴力を有することを確認した.

3.2.2　実験装置と刺激

本実験では，アルファベット大文字を，１文字ずつ書き順に沿って音を移動させることにより提示した．下記順に沿って移動する刺激音は，Fig.3.1 の各点にあるスピーカから逐次的に提示した.

P. L. Divenyi, I. J, Hirsh [5] や C. S. Watsonw [6] は，連続する 2 音を識別できるためには，各音の継続時間が少なくとも 20〜40 ms 必要であることを報告している. 本実験ではこの値を考慮して，かつ，迅速な図形提示の可能性を検証するために，各点を表現する音として，50 ms のホワイト・ノイズを 使用した.

一方，I. J. Hirsh [7] は，クリック音や種々の長さの帯域雑音，トーンパルスなど各種の音を用い，2 音の時間順序について識別閾を測定している．その結果，75 % の正答率を得るためには 2 音の立ち上がり時点を 17 ms 相隔てておかねばならないことを見出した．その後多くの研究者が種々の音を用いて同様な実験を行なっている [8]˜ [14] . その結果，2 音の時間順序の識別閾は，2ms から 150 ms にまで およんでおり，刺激パターンや被験者の習熟度や実験手続きなどによって値が変るようである．Hirsh らは逐次提示される 2 音を認識するには，50〜100 ms の無音時間が必要であると報告している [15] . 本実験では，この値を考慮して，かつ，迅速な図形提示の可能性を検証するために，角や線の切れ目など，書き順においてペンが止まる部分に 50 ms の無音を挿入した.

移動する音は，Fig.3.1 に示した 8 行 8 列に小型スピーカを配置したスピーカ・マトリクスにより提示した．各スピーカは，エンクロージャにはいれず，ボードに貼り付け，裏面からの音の放射を遮断した.

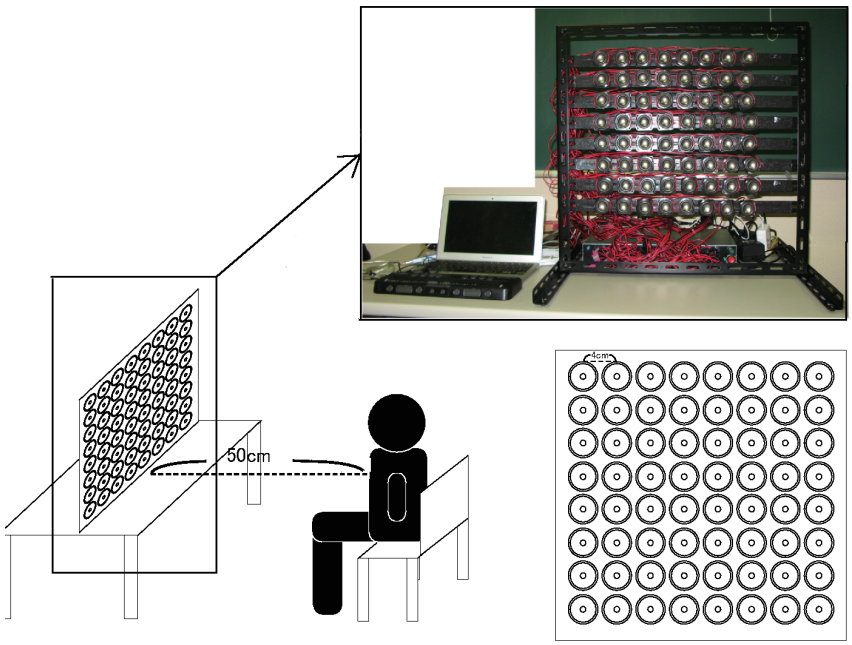


Fig. 3.1: The environment of the experiment.

音が動いたことを認識できる最小角度である最小可聴運動角 (minimum audible movement angle: MAMA) は正面方向で ± 3.6° であることが報告されている [16]˜ [18] . この基準を満たすように，スピーカ・マトリクスの各スピーカの中心間距離を 4 cm ，および，スピーカ・マトリクスと被験者との距離を 50 cm と定めた．この場合では，被験者の頭部中央を原点とし，隣り合うスピーカの成す角度は， 4.4° から 4.5° の範囲に収まる．

Fig.3.2 に，本装置のブロック図を示す．刺激提示装置は，パーソナル・コンピュータ (PC)，USB I/O モジュール（P&A Technology 製 : PA-S500)，スピーカ切り替え装置（Fig.3.2 switching device 参照），フルレンジ・スピーカ（AURA SOUND NSW1-205-8A 64 個，振動面の直径 : 2.54 cm, スピーカユニットのサイズ : 1 辺 3.5cm の正方形からなる．本実験では，2 章での結果を受け, 音源定位に十分なスペクトル情報を含むことのできるフルレンジ・スピーカを使用した．

PC に接続された USB I/O モジュールから TTL(トランジスタ・トランジスタ・ロジック）レベルの制御用ビット信号（5V トレラント）を出力し，スピーカ切り替え装置内部のフォトモス・リレーを ON/OFF した．スピーカ切り替え装置には，常に，刺激音（ホワイト・ノイズ）が入力されており，上記フォトモス・リレーの ON/OFF により，指定したスピーカから刺激音を出力できる．スピーカ切り替え装置には，64 個のスピーカが接続されている．このスピーカを，8 行 8 列のマトリクスになるように，近接して配置した．スピーカ・マトリクスは，各スピーカが被験者から見えないように，黒色のスピーカ用ネット (FOSTEX SFC83-08 m) で覆った．

森本・斉藤は，正中面全体について，音源定位を決定する周波数スペクトル

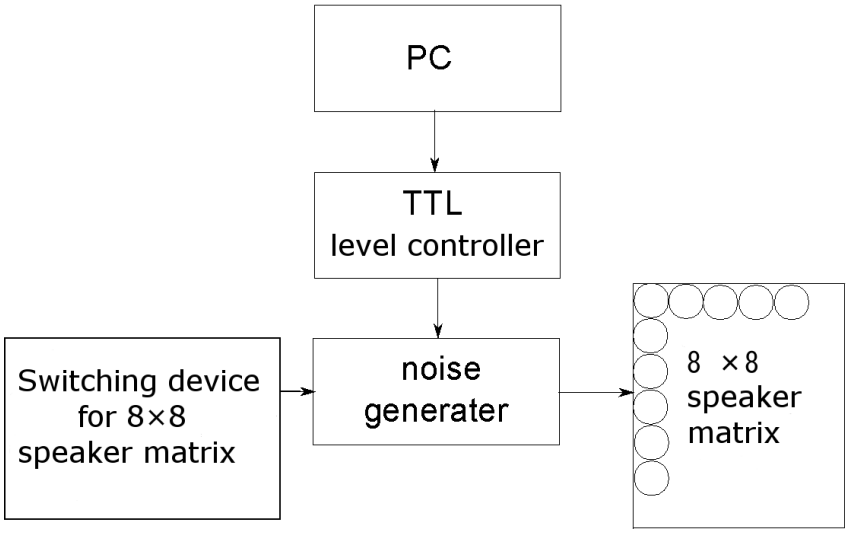


Fig. 3.2: Block diagram of the presentation system.

の要因を詳細に検討した．その結果，正中面全体にわたって正しい音源定位を得るためには，音源信号に 5 kHz〜10 kHz の周波数成分が含まれていなければならないことを示した [19] . Butler らは，正中面の前半分にある音源の仰角を定める手がかりは， 5 kHz から 11 kHz を中心周波数とする，1 オクターブ幅の谷の周波数であり，その谷は仰角が上昇するにしたがって高域に移動することを報告している [20] . 本実験で使用したスピーカ・マトリクスは，20 Hz から 10 kHz まで，ほぼフラットな周波数特性を持っており，提示した音の水平位置を認識する成分を充分含むように，刺激音として，20 Hz から 20 kHz のランダムノイズ（ホワイト・ノイズ）を用いた．すべてのスピーカから出る音は，実験室内の暗騒音レベルが 46 dB(A) であることから，おおむね 68 dB(A) になるように調整した．

3.2.3 実験手順

被験者を，Fig.3.1 に示すように，スピーカ・マトリクスの正面に座らせ，刺激音を聞かせた．被験者の耳の高さが, スピーカマトリクスの 4 行目と 5 行目の間，つまりスピーカマトリクスの垂直位置の中央に位置するように椅子の高さを調整した．実験中は，スピーカ・マトリクスを覆った黒色のネットの中央を見て，提示された音の動きを聞くように教示した．

まず，練習として，すべてのアルファベット刺激パタンを一度ずつ音提示した．この際，各刺激のアルファベット名称を教示した上で，音を聞かせた．

続いて，26 個のアルファベット大文字を，順番をランダムに提示し，それを 1 セットとして，全部で 5 セット聞かせ，合計 130 試行行った．各文字の刺激提示直後に，どのアルファベットとして認識したか，口頭で答えさせた．

刺激図形が 26 種類あるため，記憶効果を相殺する時間を設ける必要はないと考え，上記の練習と，それに続くアルファベットの書き順の音提示実験は休

憩を挟まずに行った．なお，実験の途中では，いつでも休憩できることを教示したが，休憩した被験者はいなかった．

全試行 (5セット）終了後，５回提示したうち３回以上間違った文字について，再び刺激音を聞かせた後，その刺激音がどのような軌跡を描いたかを図示させた．

3.2.4 書き順認識の要素

小学校学習指導要領解説算数編では,「図形の空間的要素」は,「形，大きさ，位置に関する視覚的，同時的 認識」と定義されていることが知られている．これに準じて，提示された音の移動により，図形や文字の形状を認識する要素として，以下の項目を挙げた．

a. 提示された線の向きがわかる．

b. 提示された線が直線か曲線かがわかる．

c. 提示された線の長さがわかる．

d. 提示された２線の位置関係がわかる．

後述する実験結果では，誤認識の要因を分析する要素として，上記項目を用いることにする．

3.2.5 結果

Tbl.3.1 に，各文字ごとの正答率と標準偏差を示す．正答率は，各被験者ごとに，各文字の５回の正答率を算出し，全被験者（15 名）の平均を求めたものである．

Tbl.3.2 より，各文字ごとの平均正答率 90%以上が 11 文字，80%以上が 20 文字，70%以上が 24 文字であることがわかる．

Tbl.3.1 から，正答率の悪い文字ほど，個人差が大きくなっており，識別しにくいことがわかる．

また，全試行終了後，5 回提示したうち，3 回以上誤答した文字を被験者に再び聞かせ，その音の軌跡がどのような図形に感じられたか図示させたところ，Fig.3.3 に示した結果が得られた．

Fig.3.3 から，誤答のパターンを以下のように分類した．

Tbl. 3.1: The average rate and the standard divition of correct answers in each character.

提示文字	平均正答率 (%)	標準偏差 (%)	提示文字	平均正答率 (%)	標準偏差 (%)
A	74.67	25.60	N	93.33	12.34
B	97.33	7.04	O	82.67	28.15
C	86.67	22.25	P	68.00	32.78
D	86.67	16.33	Q	93.33	9.76
E	94.67	11.87	R	93.33	16.33
F	89.33	12.80	S	61.33	34.20
G	92.00	21.11	T	77.33	29.15
H	78.67	20.66	U	93.33	16.33
I	98.67	5.16	V	92.00	16.56
J	89.33	12.80	W	81.33	20.66
K	76.00	22.93	X	96.00	8.28
L	93.33	12.34	Y	80.00	23.90
M	86.67	24.69	Z	84.00	18.82

Tbl. 3.2: Number of characters in correct answers (average).

60%以上 70%未満	2 文字
70%以上 80%未満	4 文字
80%以上 90%未満	9 文字
90%以上 100%以下	11 文字

3.2.5.1 提示された線の向きの誤認識

提示された線の向きを誤認識した代表的な例を Fig.3.4 に示す.

Fig.3.4 より，提示された線の水平成分，垂直成分を誤認識していることがわかる. 逐次提示される 2 音を認識するには，マスキング効果を相殺するために，50〜100 ms の無音時間が必要であることが報告されている [21]˜ [26] . 本実験では，この値を考慮して，かつ，迅速な図形提示の可能性を検証するために，角や線の切れ目など，書き順においてペンが止まる部分に 50 ms の無音を挿入した. しかし，向きを誤認識した線は，その前または後に提示された線の向きに影響されている. 線と線の間の無音を，少なくとも 150 ms にすることで，提示した線の向きの認識精度を向上させることができる. また，2 音源の音圧レベル差が 10 dB 以上あると，レベルの高い側に音像が定位されると報告されている [27] [28] . 今後の改善点において，前または後に提示された線

Pattern of alphabets for stimulus in this experiment	The figures of characteres which the subjects answered incorrectly	Pattern of alphabets for stimulus in this experiment	The figures of characteres which the subjects answered incorrectly
A	A A A A A	P	P P P 5
C	C c	Q	Q
D	D	R	R
F	F F	S	S S S 6 S S 6 S S
G	G	T	T T T T T
H	N	U	U U
J	J	V	V V V
K	K K K K K K K	W	W W M W
L	L L	X	X
M	M M	Y	Y Y Y Y Y
N	N	Z	Z Z Z Z Z
O	O O C C		

Fig. 3.3: The figures of characters which the subjects answered incorrectly.

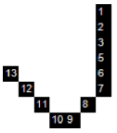
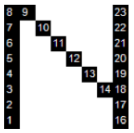
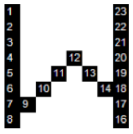
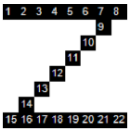
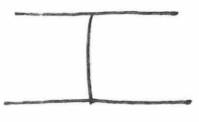
Pattern of alphabets for stimulus in this experoment	The examples of false recognition
	
	
	
	

Fig. 3.4: False recognition of direction of lines.

の向きに影響しないように，2 線を提示する音の音圧レベル差を 10 dB 以上にすることで，各線の向きを正しく認識できるようになると考えられる．

3.2.5.2　提示された線が直線か曲線かの誤認識

提示された線が直線か曲線か誤認識した代表的な例を Fig.3.5 に示す.

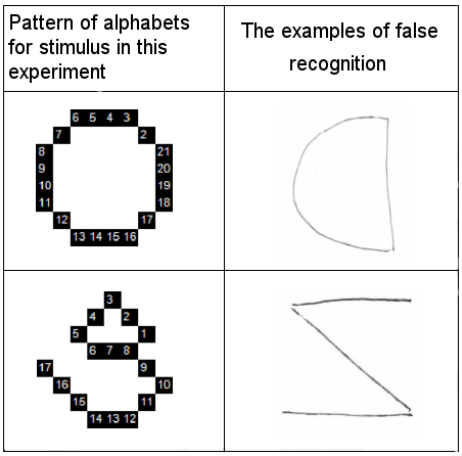


Fig. 3.5: False recognition of a straight line and a curved line.

Fig.3.5 より，提示された線が曲線のみにもかかわらず，一部を直線として認識していることがわかる．主な要因として，8 行 8 列では充分に曲線を提示できていなかったことが考えられる．図形情報を表現するスピーカ・マトリクスの密度を増やすことで，より詳細な曲線を表現できるようになると考えられる．

3.2.5.3　提示された線の長さの誤認識

提示された線の長さを誤認識した代表的な例を Fig.3.6 に示す.

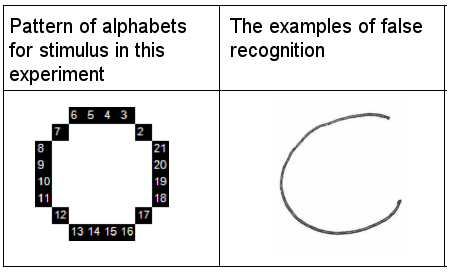


Fig. 3.6: False recognition of the length of a line.

Fig.3.6 より，一周した円として認識されていないことがわかる．主な要因として，継時マスキングによって，線の長さが実際より短く認識されたことが考えられる．

なお，聴覚の空間分解能は左右方向に比べて上下方向が著しく悪いことが知られているが，本実験では，被験者の正面から縦，横 40cm の範囲から音を提示したため，上下方向につぶれたようには認識されず，円に感じられた．

3.2.5.4　提示された２線の位置関係の誤認識

提示された２線の位置関係を誤認識した代表的な例を Fig.3.7 に示す.

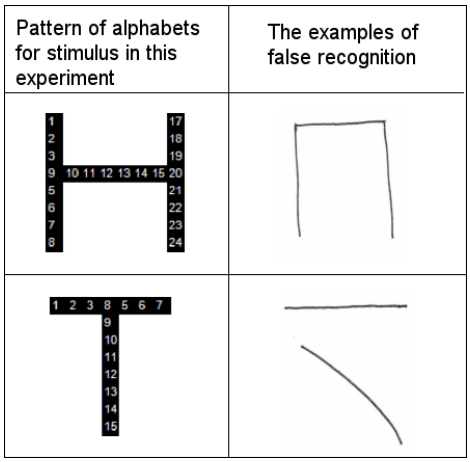


Fig. 3.7: False recognition of the relationship of two lines.

Fig.3.7 より，正中面付近に提示された線の高さが正しく認識されていないことがわかる. 主な要因として，正中面の上下の定位があいまいに知覚されたことが考えられる.

3.2.5.5　その他

X: 一人が存在しない曲線を， X の右側に認識した.

S: 六人は正しく図示したが，一人が，Z を左右反転したような図形として認識した．また，二人が，数字の 6 のように認識した.

W: 二人は正しく図示したが，一人が，M として認識した.

P: 三人は正しく図示した．一人が，数字の 5 のような図形として認識した.

R: 一人は正しく図示したが，一人が，かなり崩れて認識した.

3.2.5.6　正しく図示

C，D，G，Q，U，V，Y，は，アルファベットの書き順認識では， 5 回のうち 3 回以上間違ったにもかかわらず，正しく図示した.

3.2.6 考察

Tbl.3.2 より，平均正答率 90%以上が 11 文字，80%以上が 20 文字，70%以上が 24 文字であることがわかる．また，もっとも平均正答率の低い文字である S でも 61.33%，もっとも平均正答率の高い文字である I で 98.67% であることがわかる．すべての文字の平均正答率は，86.15%であった．この結果より，垂直平面上を移動する音により，縦線，横線，斜線，曲線を認識し，図形を高い正答率で認識できることが明らかになった．

誤認識の要因を 3.2.5.1 から 3.2.5.4 に分けて，下記の主要因を明らかにした．

1. 継時マスキング効果の影響
2. 曲線を表現するために充分なスピーカ・マトリクスの密度
3. 先行音効果の影響
4. 正中面定位（上下）のあいまい差

しかし，これらの要因は必ずしも一義的なものではなく，複合的に作用し合っている．したがって，提示スピードや音圧などを検討することで，より鮮明な図形提示ができる可能性がある．

「3.2.5.5 その他」の誤認識については，音の移動が複雑だったため，正しい形を認識できなかったものと考えられる．Chipman は，12 個の正方形を様々に並べてつくったパターンがどれくらい複雑に見えるかを判断基準として定め，複雑さ評定値を 7 種類の物理的変数で重回帰予測した．その結果，角の数や周の長さなどの量的変数には負の相関があり，対称性や反復性などの構造的変数には正の相関があることを報告している．Chipman はこの式から，量的変数が知覚的複雑さの上限を決定し，構造変数はその上限から複雑さを減ずることを提唱した [29] ．本実験の被験者の内観に,「早い音があちこちに飛んで，追うことが難しかった」という意見があった．複雑な形状を提示する場合, 利用者が認識しやすい提示スピードに調整できるようにするなどの対策が考えられる．

「3.2.5.6 正しく図示」に示した文字は，音の軌跡は正しく図示できていたにもかかわらず，アルファベット名称を正しく回答できなかった．この理由として，被験者の普段の書き順と，本実験での書き順が異なっていたことが考えられる．内観報告によると,「文字全体の形はわかったが，書き順が自分のものと違い混乱した」と報告している．この問題を回避することは，文字の書き順を習熟した成人を対象とした実験の場合，困難である．しかし，移動する音の軌跡によって，文字の全体的な形状は正しく認識できていた．

なお，全盲の視覚障碍者として，まず第 2 著者（守井清吾）を対象に，同じ条件で実験を行った．その結果,「D」を 1 度だけ未回答であった以外，他 129 試行はすべて正解した．「D」に関しては,「D か O か判断しかねた」との内観報告があった．この被験者は，アルファベットの形は知っているが，日常的に書いていないため,「D を縦長の閉じた図形」と認識し，O と迷ったようであ

る．さらに，第１著者（鈴木淳也）では，同じ条件で実験を行った結果，すべて正解した．今回は，二人のみの視覚障碍者被験者であるが，日常的に視覚に頼らない生活を行っているため，音源移動に関して認知感覚が優れ，晴眼者に比べほぼ100%と正解率が大きくなったと思われる．この結果からも，本方式が，視覚障碍者にも充分活用しうる手法であることが確認できた．

特に，筆者等は，視覚障碍者（全盲）でもあり，図形やグラフの迅速な認識の強いニーズを持っている．その観点で厳しく評価し，本システムの開発を重ねてきた．その結果，本実験で使用したスピーカ・マトリクスにより，さまざまな文字を迅速に表現できることを確認できた．加えて，提示スピードや提示音など，先鋭化するためのポイントも明らかにすることができた．

参考文献

[1] Beddos, M. P. and Suen, C. Y. : “Evaluation and a method of presentation of the sound output from the Lexiphone-A reading machine for the blind”,IEEE Trans., BME-18, pp.85–91,(1971)

[2] 伊福部達，吉本千禎:“聴覚の多音源定位機能を利用した視覚代行法について”, 第 3 回感覚代行シンポジウム論文集，製品科学研究所，pp.63–66 (1977)

[3] 湯瀬 裕昭, 姚 肇清, 石川 准:“点図ディスプレイと音声ガイドを用いた墨字の筆順学習システムの試作”，電信学技報. ET, 教育工学 102(697), pp73–78 (2003-02-28)

[4] 赤色ドットマトリクスＬＥＤ　８ｘ８ドット　ＯＳＬ６４１５０１－ＡＲＡ: LED(発光ダイオード) http://akizukidenshi.com/catalog/g/gI-05163/

[5] P.L.Divenyi and I.J.Hirsh, “The effect of blanking on the identification of temporal order in three-tone sequence”,erception and Psychophysics 17, pp.246–252 (1975)

[6] C.S.Watson, H.W.Wroton, W.J.Kelly and C.A.Benbassat:“Factors in the discrimination of tonal patterns. 1.Component frequency, temporal position, and silent intervals”,J.Acoust. Soc.Am.57, pp.1175–1185 (1975)

[7] I.J.Hirish, “Auditory perception of temporal order”,J.Acoust.Soc. Am. 31, pp.759–767 (1959)

[8] W.H.Fay, “Temporal Sequence in the Perception of Speech”, Mouton, the Hague, (1966)

[9] A.M.Lieberman, K.S.Harris, A.Jo and H.Lane:“The discrimination of relative onset-time of components of certain speech and non-speech”,J.Exp.Psychol.61, pp.379–388 (1961)

[10] J.H.Patterson and D.M.Green:“Discrimination of transient signals having identical energy apectra”,J.Acoust.Soc.Am.48, pp.894–905 (1970)

[11] R.Efron:“Conservation of temporal information by perceptual systems”,Perception and Psychophysics 14, pp.518–530 (1973)

[12] P.L.Divenyi:“Identification of temporal order in three-tone sequence”,J.Acoust.Soc.Am.55, pp.144–151 (1974)

[13] C.S.Watson, H.W.Wroton, W.J.Kelly and C.A.Benbassat:“Factors in the discrimination of tonal patterns. 1.Component frequency, temporal position, and silent intervals”,J.Acoust. Soc.Am.57, pp.1175–1185 (1975)

[14] D.E.Broadbent and P.Ladefoged:“Auditory perception of temporal order”,J.Acoust. Soc. Am.31, pp.1539 (1959)

[15] P.L.Divenyi and I.J.Hirsh:“The effect of blanking on the identification of temporal order in three-tone sequence”,Perception and Psychophysics 17, pp.246–252 (1975)

[16] イェンス フラウエルト, 森本 政之, 後藤 敏幸:“空間音響”, pp.54–56 (1986)

[17] R. Preibisch-Effenberger:“Die Schallokalisationsf,”ahigkeit desMenschen und ihre audiometrische Verwendung zur klinischen Diagnostik(The human faculty of sound localization and its audiometric application to clinical diagnostics)”, Dissertation, Technische Universit,”at, Dresden (1966)

[18] B.G. Haustein, W. Schirmer:“Messeinrichtung zur Untersuchung des Richtungslokalisationsverm ogens (A measuring apparatus for the investigation of the faculty of directional localization)”, Hochfrequenztech. u. Elektroakustik, 79, pp.96–101 (1970)

[19] 森本政之, 斉藤明博:“音の正中面定位について；刺激の周波数範囲と強さの影響について”, 日本音響学聴覚研資料, H-40-1 (1977)

[20] R.A.Butler and K.Belendiuk:“Spectral cues utilized in the localization of sound in the median sagittal plane”,J.Acoust.Soc.Am.61, pp.1264–1269 (1977)

[21] 比企:“連続音声中の各種の区分の持続時間の性質”，電通学誌，pp.50,1465 (1967)

[22] D. H. Raab:“Forward and backward masking between acoustic clicks”, J. Acoust. S0C. Am., 33, pp.137 (1961)

[23] L. L. Elliot:“Backward and forward masking of probe tones of different frequencies”, J. Acoust. SOC. Am., 34, pp.1116 (1962)

[24] L. L. Elliot:“Backward masking:Monotonic and dichotic conditions”, J, Acoust. SOC. Am., 34, pp.1108 (1962)

[25] H.N. Wright:“remporal summation and backward masking”, J. Acoust. soc. Am., 36, pp.927 (1964)

[26] H.Fastl:"Temporal masking effects: I. Broad band noise masker", Acustica, 35, pp.287 (1976)

[27] R.Y.Litovsky, B.GShinn-Cunningham:"Investigation of the relationship among three common measures of precedence: Fusion, localization dominance and discrimination suppression", J. Acoust. Soc. Am., I09(1), pp.346–58 (2001)

[28] K.Saberi, J.V.Antonio:"Precedence-effect thresholds for a population of untrained listeners as a function of stimulus intensity and interclick interval",J. Acoust. Soc. Am., 114(1), pp.420-9 (2003)

[29] S.R Chipman:"Complexity and Structure m Visua1 Pattems",J Exp Psych01 Gen, v01.106,n0.3,pp.269301 (1977)

第4章　触れた位置の線の方向提示による図形輪郭認識（インタラクションと聴覚フィードバックの融合）

4.1　背景と目的

弱視の人は，ルーペで拡大して，文字や図形を認識する．この場合，紙面全体をルーペに収めることはできないため，紙面のごく狭い範囲を拡大し，手でルーペを動かすことで，全体を認識する．また，オプタコンや点図ディスプレイを用いた触覚での観察においても，指先で局部を触り，紙面全体をなぞることで，全体を認識する．

このように、知覚と運動は別々に存在するのではなく，きわめて複雑に相互に作用しあっている．外界から取得された情報だけでなく，自己の運動イメージ生成過程と強い相互作用があり，その相互作用により認知が成立すると考えられている [1] .

そこで, 指先に触れた部分の線の方向を音の移動で提示できれば，オプタコンや点図ディスプレイのように輪郭をたどり全体像を認識できるばかりでなく，音圧や周波数を変化させることで, 触れた部分の色やコントラストを表現できる二値以外のパラメタがあり，２色以上の情報も表現できると考えた．本章ではまず、形状認識に焦点を絞って，実験を行った．

ペンタブレット上の接触位置に対応する線の形状を，スピーカマトリクスで逐次提示する装置を試作した．その上で，ユーザにこの装置を使用してもらい，図形の全体像を認識可能かを予備的に調べた．このシステムを「確認位置形状の音逐次提示式図形認識装置」と命名する（以下「音提示図形認識装置」と略す）．しかし，音での図形提示に当たっては，音の種類や提示時間など，考慮すべき条件が多い．

本稿では，まず線の方向を音で提示するための音の提示時間・提示方向について，予備的かつ重点的に検討する．用いる音は，第 2 章の実験結果よりホワイトノイズとした．

まず, 線の形を，音の配列で逐次提示する際の音として直観的に認識できるパラメタを確認した．(4.2 節）その上で，２節で設定した音で図形の認識実験を行い，本システムの有効性を検証した．(4.4 節）

4.2 線の形を音提示するためのスピーカ距離・提示時間の諸検討

線の方向を，音の配列で逐次提示する際の音として直観的に認識させられるスピーカ距離および提示時間を調べた．次に，ここで決定したスピーカ距離および提示時間を用いて，線分方向を音提示した際の知覚について確認を行った．

4.2.1 被験者

以下の実験 A から C の被験者は，晴眼者 1 名 (女性，35 歳)，視覚障碍者 1 名 (男性，42 歳) である．視覚障碍者は，後天性であり，全盲である．いずれの被験者も，オージオメータ (Rion AA-75) による純音聴力検査の結果，4 分法で 20dB HL 以下であった．また，実験の際は目隠しをさせ，視覚情報を遮断した．

4.2.2 実験手順

Fig.4.1 のように，4 個の圧電スピーカ (村田製作所:PKM34EWH1201C, 直径 3.5cm) を 2 行 2 列に配置した．この配置は，縦，横，斜めの方向を，2 音の逐次提示によって表現しうる最小単位である．

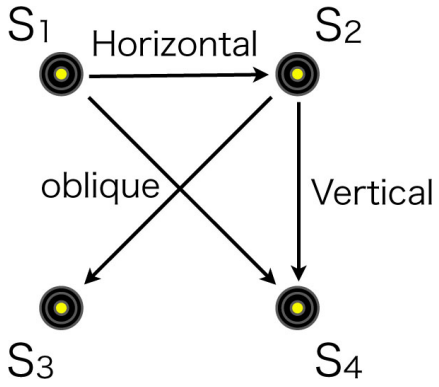


Fig. 4.1: Loudspeaker matrix setup composed by four piezoelectric loudspeakers for the experiment of alignment decision. Sound presentation of directions are also shown here.

縦，横，斜め方向の音提示に当たっては，Fig.4.1 に示すとおり，それぞれ $s_2 \rightarrow$ s4, s1 $\rightarrow$ s2, s1 $\rightarrow$ s4 もしくは s2 $\rightarrow$ s3 と逐次的に遷移する音を用いた．

刺激音は，ホワイトノイズ (周波数帯域：20Hz-20kHz) を用いた．スピーカからスピーカへの遷移時間 (以下，スピーカ遷移時間) は 0ms とした．全てのスピーカから提示する音圧は一定値 (概ね 60 dB) に揃えた．一つのスピーカにおける提示時間 (以下，スピーカ提示時間) は 40ms から 100ms まで 20ms 刻みとし，2 スピーカの中心間距離は 5 cm から 20cm まで 5cm 刻みとした．

被験者には，スピーカ提示時間／スピーカ間距離をランダムに設定した際の音を聞かせ，移動を感じられたかを応答させた．練習としてスピーカの中心間距離が 5cm（もっとも狭い場合）と２０ｃｍ（もっとも広い場合），１スピーカの提示時間が 40ms（もっとも短い場合），130ms（もっとも長い場合）のものを各 1 回ずつ聞かせた．続いて，本番として 4 (スピーカ提示時間) × 2 (方向) × 4 (スピーカ間距離) × 3 (繰り返し回数) = 144 回行った．

4.2.3 結果・考察

Fig.4.2 に，スピーカ提示時間／スピーカ間距離ごとの移動を知覚された率を示す．

大まかな傾向として，縦方向・横方向ともに，スピーカ間距離が大きくなるほど，提示時間が長くなるほど，移動を感じたと答える率が増加した．

横方向・縦方向ともに，提示時間が 40 ms のときは，スピーカ間距離が 15cm 以下の場合，移動の知覚は 100%にはならなかった．スピーカ間距離が 5cm の場合，横方向では 60 ms 程度から移動をある程度知覚出来るのに対し，縦方向

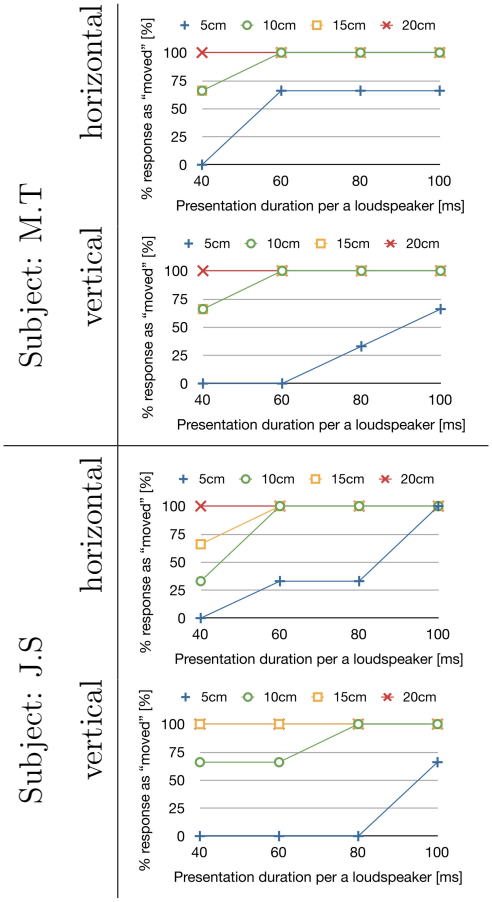


Fig. 4.2: Percentage of sound moving perception (horizontal and vertical directions) in relation to presentation speaker duration and inter-speaker distance in each subject

では，80 ms〜100 ms 程度必要であった．一方で，スピーカ間距離が 10cm 以上については，縦方向・横方向ともに移動感知覚において似た傾向を示した．

このことから，実用上，横方向よりも縦方向の方でより広めのスピーカ間距離が必要であるといえる．この結果は，縦方向・横方向における音源定位の分解能に関する結果と一致する．

また，各被験者の内観報告によると，M.T は，縦，横とも 10cm がもっともわかりやすく，逆に 20cm になると音を追わなければならないので若干わかりにくくなると報告している．一方，J.S は，横は 10cm，縦は 20cm でわかりやすいと応答している．縦，横を明確に知覚できるスピーカ間距離は，若干の個人差があることが考えられる．

両被験者ともに，縦，横を 100%認識できたのは，２スピーカ距離が 20cm，１スピーカ提示時間が 60ms のときであった (Fig.4.3)．このため，次節以降で用いるスピーカマトリクスでは，この結果を元に縦・横間のスピーカ間距離を 20 cm，スピーカ提示時間を 60 ms と設定した．

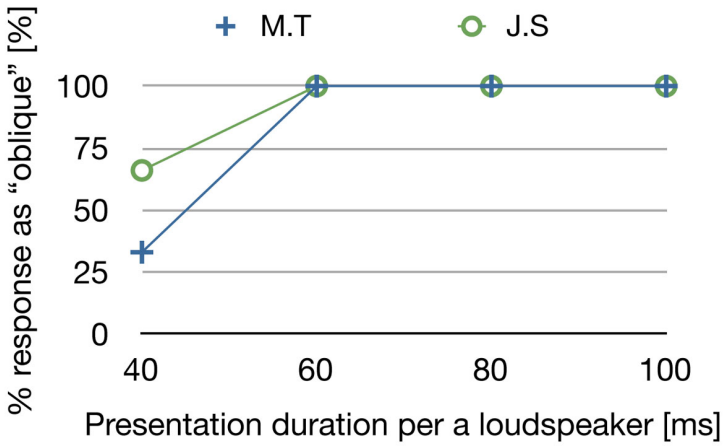


Fig. 4.3: Percentage of oblique direction perception in relation to presentation speaker duration

縦，横に加え，斜め方向の移動も確実に感じられる１スピーカ提示時間は，60ms であった．このことは，先行音効果の観点からも説明できる．先行音効果は，スピーカ２つから発せられる音の時間差が 30ms 以下の場合に発生する．先行音効果とは，２つのスピーカから連続的に音を提示する際の時間差がある閾値より短い場合に，後から出た音が先に音の出た場所から聞こえる効果のことである．このことからも，２音が別々の音として聞こえるためには，１スピーカあたりの提示時間を 50ms 以上とするのがよいことになる．先行音効果を考慮した場合，40ms はエコースレッショルドなので，人によっては先行音効果の影響で，移動感が感じにくくなる可能性がある．

斜線の方向提示

斜めの音の移動の知覚に関して，スピーカ間距離を 20cm にした上で確認した．実験手続きは同様であり，繰り返し回数は 12 sets = 4(speaker duration) × 1(direction: oblique) × 1(inter-speaker distance) × 3(repeats) である．

両被験者とも，40 ms の場合は，斜め方向の認識が曖昧になり，60 ms 以上で移動感を確実に知覚出来た．

そこで，両被験者が縦，横，斜めの音の移動を確実に感じられた値である２スピーカ距離を 20cm，１スピーカの提示時間を 60ms に定め，以下の実験 B, C を行った．

4.3　２線の角度の知覚

4.3.1　実験方法

刺激音は，ホワイトノイズ (周波数帯域：20Hz-20kHz) を用いた．スピーカからスピーカへの遷移時間 (以下，スピーカ遷移時間) は 0ms とした．全てのスピーカから提示する音圧は一定値 (概ね 60 dB) に揃えた．

この刺激音を用いて，Fig.4.1 のように，縦，横，斜め（右上から左下，左上から右下の２種類），また，Fig.4.4 のように角（左上，右上，左下，右下の４種類）の方向パタンをランダムに３回ずつ提示した．角を提示する際に，斜め方向→横方向の音源移動→斜め方向→…と繰り返す場合，提示された角の位置が明確でない．そのため，角を明確に提示するために，斜め方向の音源移動→横方向の音源移動→ 60 ms の無音→斜め…と提示した．

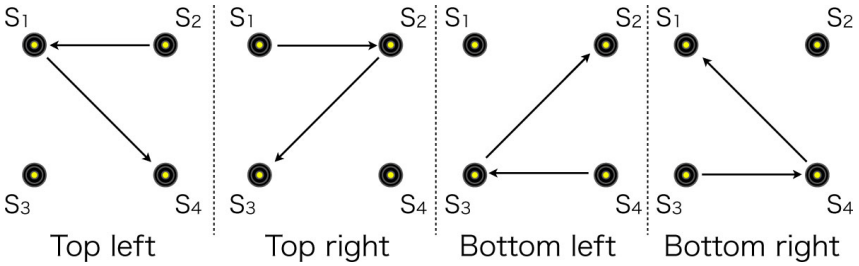


Fig. 4.4: Presentation method of diagram angle by loudspeaker matrix: the combinations of horizontal and oblique sound movings.

4.3.2　結果

いずれの方向パタンとも１回の提示で移動方向がわかった．これより，「２．２．２　実験方法」で設定した条件で，縦，横，斜めを確実に提示できることを確認できた．

上記実験より，２点または３点で線の方向を示すことで，図形の輪郭をたどれることがわかった．しかし，３点の音で角度を繰り返し提示した場合，三角形として認識できるが，どの角が提示したいものかがわからないとの内観を得た．そこで，３点で角度を繰り返して提示する際，提示の終点と，次の提示の始点の間に，１スピーカ提示分の無音を挿入した．この結果，２辺挟む角を表

現できるようになった．また，同様に，２点の音で線分を繰り返し提示する際も，提示の終点と，次の提示の始点の間に，１スピーカ提示分の無音を挟むことで，始点，終点を明確にでき，線分の傾きだけでなく，方向も示すことができることがわかった．

4.4 図形探索実験

4.1 節で紹介したオプタコンや点図ディスプレイは，探索位置の線の方向を触覚提示する装置である．そこで, 探索位置の線の方向を音の移動で提示したとき, 線をなぞって図形の輪郭を認識できるか確認するために, 触れた位置にある線の形を，音の配列で逐次提示する装置「音提示図形認識装置」を試作した．

4.2, 4.3, 4.4 節で用いたパラメータを利用して，スピーカマトリクスとペンタブレットを組み合わせたシステムを試作した．その上で，スクリーン上に提示された図形の全体像を認識可能か調べた．

4.4.1 音提示図形認識装置

Fig.4.5 に「音提示図形認識装置」のコンセプトを示す．

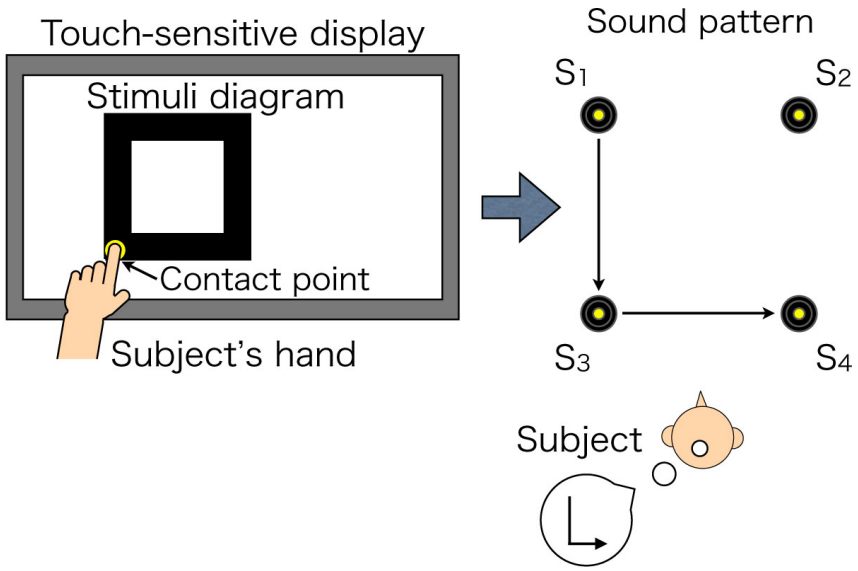


Fig. 4.5: Presentation method of 2-D diagram utilizing touch-sensitive visual display and loudspeaker matrix: aurally presentation of the corresponding line direction to contact position

本装置は，ノートパソコン，ペンタブレット，オーディオインタフェース，４個のスピーカからなる．

被験者は，ペンタブレットを用いて画面上の確認したい位置を移動する．確認している画面上の絶対位置は，ペンタブレット上のペン（手）の位置により

知ることができる．ペンタブレットと画面の四隅は一致しているため，被験者は，ペンタブレット上でペンを動かすことで，画面上の確認したい位置を動かせる．ユーザは自分の触れている PC のディスプレイ上の位置を，ペンタブレットの位置から知ることが出来る．これは，ペンタブレットの 4 隅と PC の 4 隅が一致するようにしているためである．被験者が触れた位置に対応する画面上の位置に線があれば，オーディオ・インタフェースを経由してその線の向きを，移動音を 4 個のスピーカから提示する．このときのペン（手）の動きにより，図形の輪郭を認識させるものである．

4.4.2 刺激

今回は，基礎検討として，水平，垂直，斜め (45°，135°）の線からなる図形を定時し，これらの移動方向を示す音を用いて輪郭認識実験を行った．

移動方向は，２スピーカ距離を 20cm，１スピーカの提示時間を 60ms の音で提示した．

また，触った位置の周囲に線分がある際，ガイド音が鳴るように設定した．ガイド音は 500Hz の純音とした．触った位置の上下左右に線分があった場合，線がある方向を 4 個のスピーカ (s-l,s-r,s-u,s-d) で提示した (Fig. 4.6)．ガイド音が鳴る範囲は，触った位置の周囲であり，半径 10pixel から 200pixel まで 10pixel 刻みで，被験者が変更できるようにした．本実験において全ての被験者は，ガイド音が鳴る範囲を 100pixel と設定した．

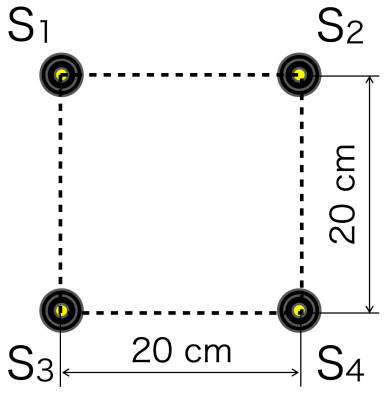


Fig. 4.6: Loudspeaker matrix which displays direction of a line segment by sound presentation

提示図形の種類は直角二等辺三角形（左下を直角とする二等辺三角形，右下を直角とする二等辺三角形），正方形の 3 種類である．各提示図形は，縦 300pixel，横 300pixel の正方形（１辺が約 7.5cm) に内接する大きさとした．提示図形は，45pixel（約 1cm) の太さの線で描画した．

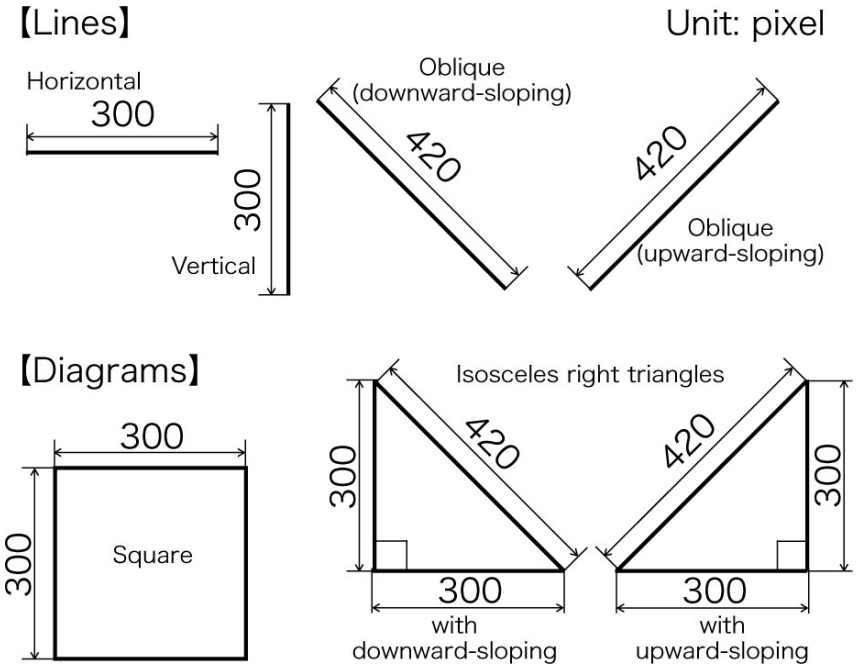


Fig. 4.7: Line and diagram stimuli

4.4.3 実験手順

実験を行うに当たり，まず被験者をスピーカマトリクスの前に座らせ，ペンタブレットを持たせた．実験を始める前に，被験者にはこれらの図形が提示されることを教示した．

次に，Fig.4.7 に示すような図形のいずれかをディスプレイ上に表示した．また提示エリアの左下隅に対応するペンタブレットの位置を凸点で教示した上で，輪郭探索を行わせた．

その上で，被験者がたどったペンの軌跡を記録した．

被験者には，ペンタブレットをペンでなぞるよう教示し，なぞりながら音を聞くことで，知覚された図形を答えさせた．その上で，１図形ごと，正答するまで形状知覚を続けさせ，被験者のペンの軌跡を記録した．

よって本実験は，練習５回，本番：３ (図形数) × 3 (繰り返し回数) = ９回で構成された．

4.4.4 結果・考察

実験者が探索を指示してから，被験者が輪郭をなぞり，いずれの図形であるか応答するまでのペンの軌跡を Fig.4.8 に示す．白の太い線は提示図形，細い線は被験者のペンの軌跡を表している．

いずれの被験者も，直線部分については，線の方向を正しく認識した上で，ペンを動かしていることがわかる．また，角周辺では，ペンを細かく左右上下に動かして，次に進む線を探している．

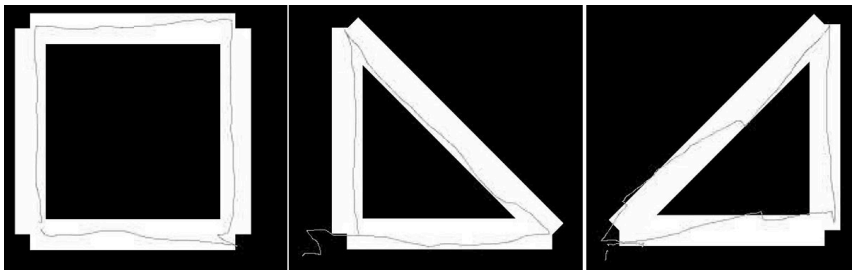


Fig. 4.8: Loci examples of subject's trace on diagram stimuli. Black, while and grey parts represent vacant, line, trace areas, respectively

また, ガイド音があると，ディスプレイ上に表示された図形の場所を発見するまで時間が短くなった (Fig.4.9).

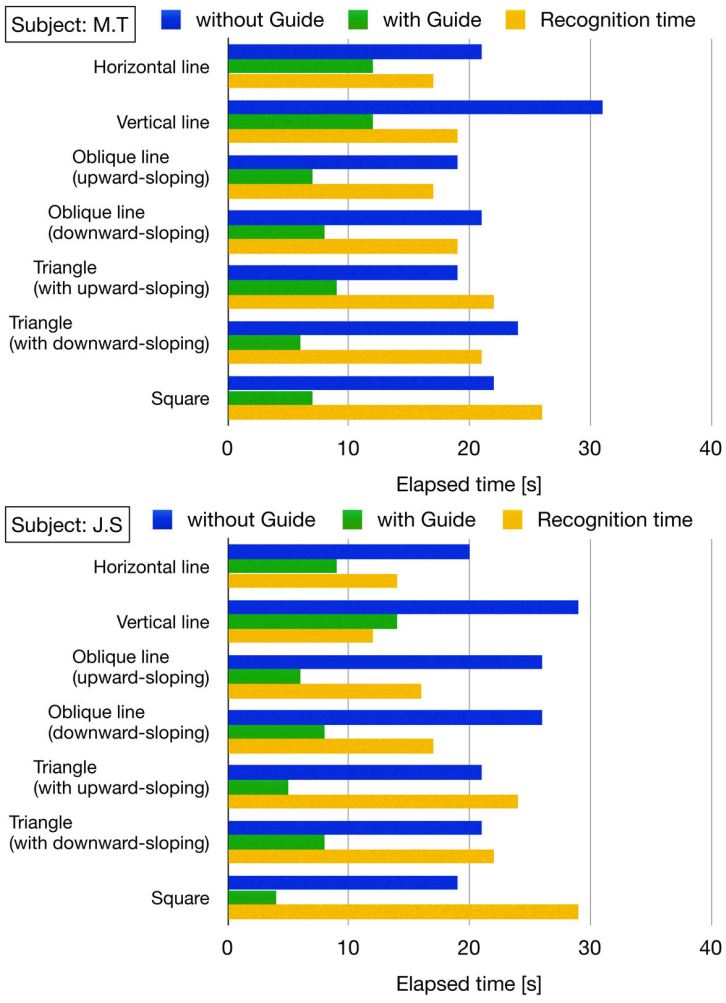


Fig. 4.9: Discovery time of the stimuli locations (legend: "without Guide" and "with Guide") and recognition time which shape is presented (legend:"Recognition time").

認識時間は輪郭線の長さに比例していた (Fig.4.10).

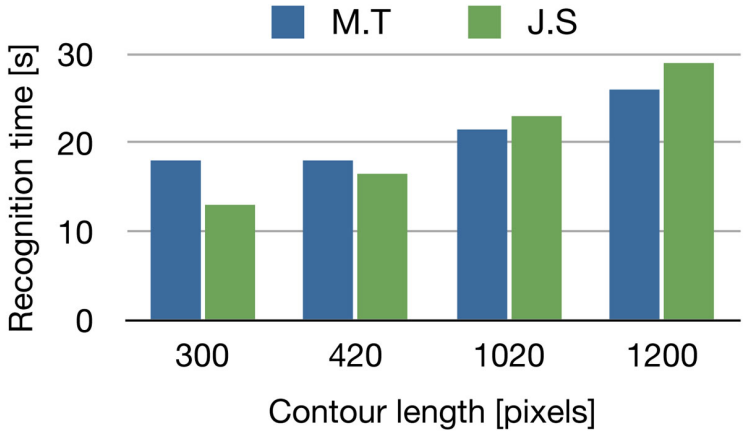


Fig. 4.10: Recognition time regarding contour length in each subject

4.5 曲線提示のためのパラメタ検討

4.5.1 はじめに

本稿では，操作に応じた音提示をスピーカマトリクスで行わせ，二次元パタンを知覚させる手法について検討する．操作方法に関しては，弱視の視覚障碍者がルーペや拡大読書機 [19] を用いて図形を認識する際の行動に着目した．具体的に彼らは，映し出される位置を能動的に変えたり拡大したりすることで，表示された対象物全体を認識している．

筆者は，視覚障碍者を想定した上で，触れた位置にある図形の線分方向を音で提示するシステムを試作し，評価した [20]．試作したシステムを Fig.4.11 に示す．タッチセンサ付ディスプレイ上に表示された図形の輪郭をなぞる際，触った位置の線分方向に応じた移動音が，スピーカマトリクスから提示されるというものである．このシステムによって，ユーザが線分で構成された図形を認識可能であることを確かめた．しかし，曲線で構成される図形に関する評価は行っておらず，検討が必要である．

そこで本稿では，曲線を音で提示する手法について基礎的な検討を行った．本研究でのアプローチの有効性を大まかに調べるために，少数の被験者での簡易的な検討を行った．まず，曲線として知覚されうる音の逐次提示方法について調べた (4.6 節)．その上で，直線と曲線の混在した図形を認識可能か調べ，任意の位置・大きさの曲線について認識可能か検討した (4.7 節)．

4.6 曲線提示に関する諸検討

曲線上に配置した複数のスピーカから音を逐次的に提示した時，直線ではなく曲線として知覚されるスピーカ個数および提示時間を調べた．その上で，次節で用いるスピーカマトリクスにおけるスピーカ個数，提示時間を決定した．

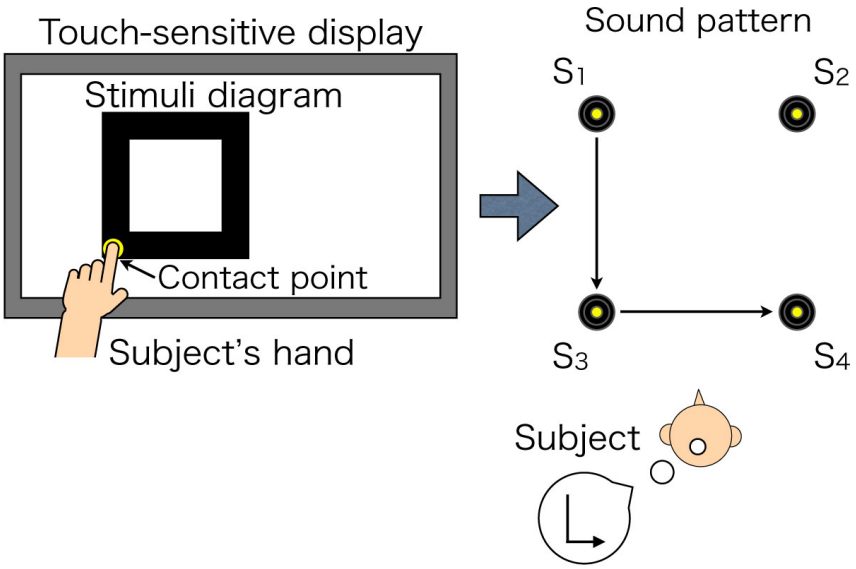


Fig. 4.11: Presentation concept of 2-D diagram utilizing touch-sensitive visual display and loudspeaker matrix: aurally presentation of the corresponding line direction to contact position.

4.6.1 実験装置

Fig.4.12 のように，圧電スピーカ (村田製作所: PKM34 EWH1201C, 直径 3.5cm) を，半径 12cm の扇形の弧に沿って配置した．この図のような装置によって提示される曲線上を移動する音を以下では刺激図形と呼ぶ．3〜6 個のスピーカを曲線上に等角度ごとに配置し，s_1 から番号順 (上側から順) に逐次的に音提示できるようにした (刺激図形は 4 種類)．これら圧電スピーカは，オーディオインタフェース (Presonus FP10) を経由して計算機上の Visual C#プログラムで制御した．

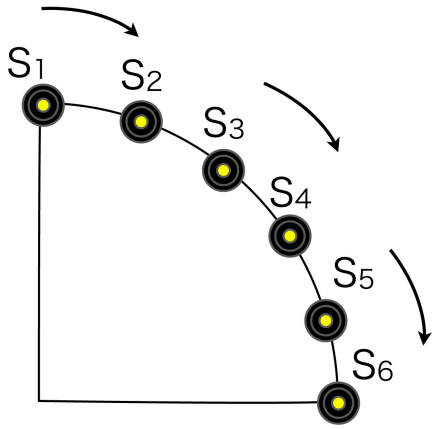


Fig. 4.12: Example of diagram stimuli. 3 ∼ 6 loudspeakers are arranged at regular angle intervals along a circular arc.

スピーカから提示された刺激音は，ホワイトノイズ (周波数帯域 : 20 Hz∼20

kHz，標本化周波数: 44.1 kHz) であった．スピーカからスピーカへの遷移時間 (以下，スピーカ遷移時間) は 0 ms とした．全てのスピーカから提示する音圧は一定値 (概ね 60 dB) に揃えた．一つのスピーカにおける提示時間 (以下，スピーカ提示時間) は 40 ms から 120 ms まで 20 ms 刻みとした．

4.6.2 実験手順

実験は静かな部屋 (幅:6 m × 奥行き:6 m× 高さ:2.5 m, 暗騒音: 30 dB) で行われた．壁から 0.5m の所に被験者用の椅子を配置した．また壁面には，被験者の頭の中心高さと同じ高さで刺激図形を設置できるようにした．

まず，実験者が壁面に 4 種類の刺激図形のうち一つを設置する．次に，刺激図形から音を提示する．この際，スピーカ提示時間はランダムである．その上で，被験者に音が鳴った場所の軌跡が曲線と感じられたか，確信度 (以下，曲線確信度) で応答させた．この確信度は，80 ～ 100%曲線と感じた場合は 5, 60 ～ 80%曲線と感じた場合は 4...，0 ～ 20%曲線 (80 ～ 100%直線) と感じた場合は 1 のように，20%刻みで 5 段階評価とした．

練習としてスピーカ個数が 6 個（最も多い場合）と 3 個（最も少ない場合）, スピーカ提示時間が 40 ms（最も短い場合）, 120 ms（最も長い場合）のものを各 1 回ずつ行わせた (計 4 回)．続いて，本番として 5(スピーカ提示時間)×4(スピーカ個数)×3(繰り返し回数) = 60 回行わせた．

4.6.3 結果

Fig.4.13 に，各被験者における，スピーカ提示時間・スピーカ個数ごとの曲線確信度を示す．大まかな傾向として，スピーカ個数が多くなればなるほど，提示時間が長くなればなるほど，曲線確信度が増加した．

スピーカ提示時間に着目すると，60 ms 以下の場合はスピーカ数が多くなっても，曲線として知覚されにくかった．一方で 100 ms 以上の場合では，スピーカ数が少なくても，曲線として知覚されやすかった (特に被験者 JS)．

またスピーカ個数に関しては，数が多いほど，曲線として知覚されやすい傾向が確認できた．今回の実験では，特にスピーカ個数が 6 のとき，スピーカ提示時間に依らず，曲線として知覚される場合が多かった．

以上の 2 点から，次節以降で用いる刺激図形は，スピーカ提示時間: 100 ms, スピーカ個数: 6 個とした．

4.7 図形認識実験

前節で作成した曲線 (s_1, s_4) の提示系に加え，s_1, s_4 を対角線とする正方形を構成する s_2, s_3 からなるスピーカマトリクスと，ペンタブレットのペンでの探索によって，提示された図形の形状を認識可能か調べた．

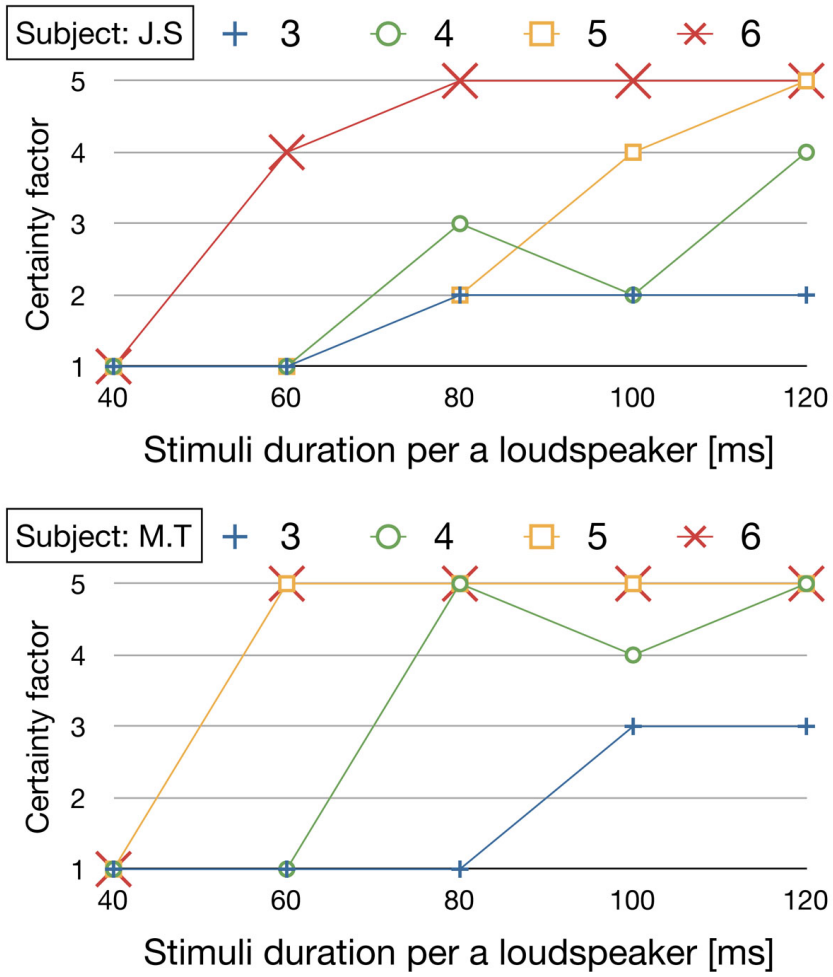


Fig. 4.13: Certainty factor regarding perception of sound trajectory as curve, in relation to stimuli duration and the number of loudspeakers.

4.7.1 図形提示方法

Fig.4.14 に図形提示コンセプトについて示す．タッチセンサ付ディスプレイ上に表示された図形の輪郭をなぞる際，触った位置の直線方向に応じた移動音が，スピーカマトリクスから提示されるというものである．

この際の操作方法は，弱視の視覚障碍者がルーペや拡大読書機 [19] を用いて図形を認識する際の行動に似ている．彼らは，映し出される位置を能動的に変えたり拡大したりすることで，表示された対象物全体を認識している．筆者は，この能動的な認識行動に着目した上で，表示された図形をユーザに探索させつつ，触った場所に応じて手がかりになる音を聞かせることによって，図形を認識させることを狙った．例えば，Fig.4.5 のスピーカマトリクスにおいて，直線は $s_1 \sim s_4$ のスピーカを用いて提示され，曲線は $s_1, s_4 \sim s_8$ を用いて提示されるようにした．

なお，この実験ではタッチセンサ付ディスプレイの代わりに，便宜上ペンタブレット (WACOM CTH-460 (横 147.2 mm，縦 92 mm)) と，コンピュータの

ディスプレイを用いた．被験者に視覚情報は提示されないため，タッチ情報を取得できるデバイスであれば，被験者へ与えられる情報は等価であることに依る．また，実験者側だけが提示図形の形状を分かれば良いためである．

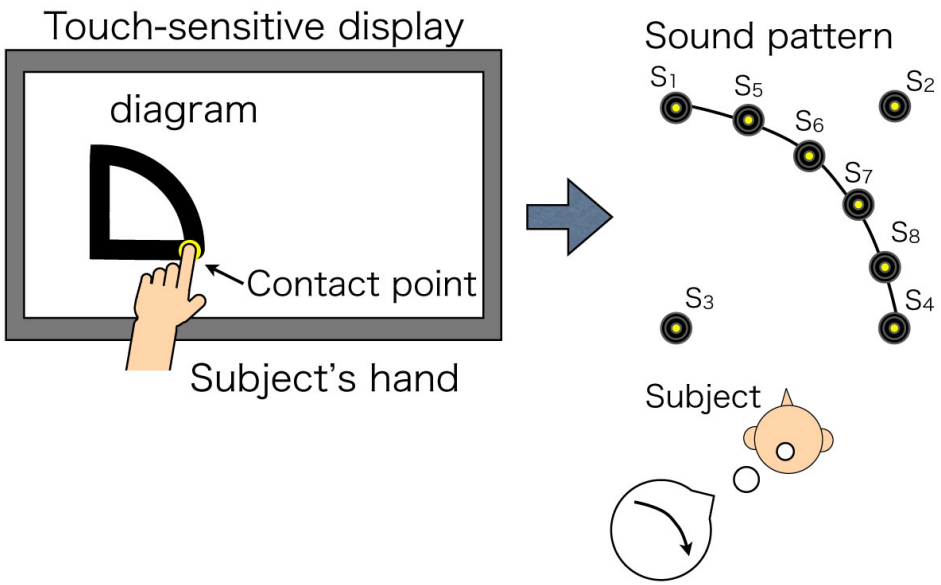


Fig. 4.14: Presentation concept of 2-D diagram utilizing touch-sensitive visual display and loudspeaker matrix in the experiment of section 4.7.

4.7.2 直線／曲線を含む図形の認識

4.7.2.1 刺激図形

刺激図形の種類は三角形（左下を直角とする二等辺三角形）および扇形（左下を直角とする扇形）の 2 種類である (Fig.4.15．各図形は，縦 300 pixels，横 300 pixels の正方形（一辺が約 7.5 cm) に内接する大きさとした．

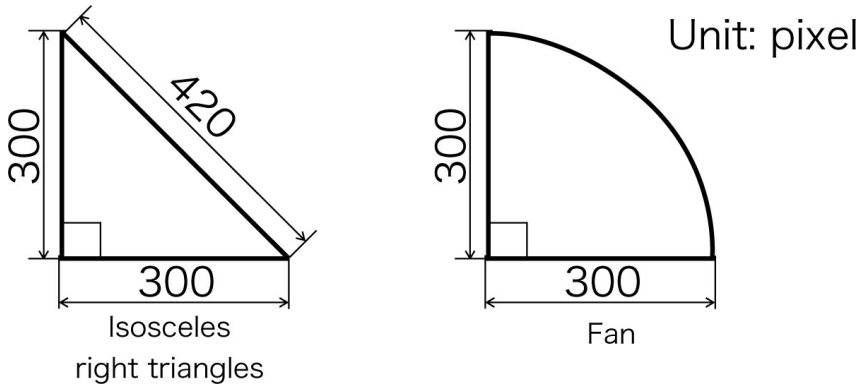


Fig. 4.15: Diagram stimuli

いずれの刺激図形の種類においても，実験を始める前に，被験者には提示される全刺激図形について教示した．また，刺激図形が任意の座標に提示されることも教示した．

4.7.2.2　実験手続

実験は 4.2, 4.3, 4.4, 4.6 節と同一の環境で行われた．まず被験者をスピーカマトリクスの前に座らせ，ペンタブレットを被験者に持たせた．次に，ペンタブレットが接続されたコンピュータのディスプレイ上に，Fig.4.15 に示すような図形のいずれかが表示されているのを実験者が確認した上で，被験者にペンタブレットで図形をなぞらせた．ペンがタブレットに触れると，その位置にある線の形状を被験者の正面に置かれたスピーカ列から逐次提示されるようにした．被験者には，ペンタブレット上をを専用ペンでなぞるよう，形状が分かったら図形の種類を答えるよう教示した．被験者には，正答するまで同一の図形が表示され続けることも教示した．

取得データは，正答するまでにかかった時間 (認識時間) である．

試行回数は，2(刺激図形の種類)×3(繰り返し回数) = 6 回行わせた．

4.7.3　結果

Fig.4.16 に各被験者ごとの図形（三角形，扇形）の認識時間を示す．いずれ

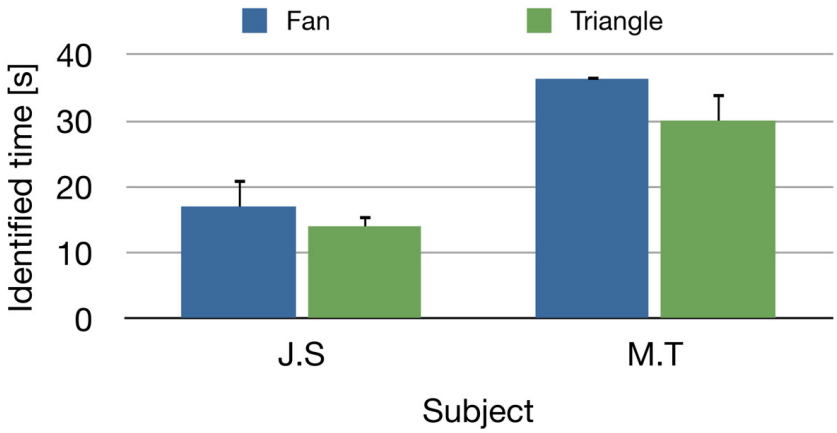


Fig. 4.16: Recognition time of diagram stimuli

の刺激図形が提示されたかについては，両被験者ともに一度で正答できた．概ね，扇形が提示された時の方が，認識時間がかかっていた．

4.8　曲線の形状認識

次に，曲線を探索させたとき，その形状を正しく認知気できているか確認した．

4.8.1 刺激図形

上下左右のいずれかに凸の半円 4 種類を，大きさ 2 種類 (半径 75, 150 pixels) で用意した (8 種類).

4.8.2 実験手続

実験のセットアップは 4.6 節と同様である．計測項目は，半円の凸の方向がどのように知覚されたかである．回数については，8(刺激図形の種類)×2(繰り返し回数) = 16 回行わせた.

4.8.3 結果

Fig.4.17 に認識された曲線を示す．それぞれのグラフの横軸は提示された図形で，縦軸は知覚された図形である．◯で書いた箇所で，提示図形に対して知覚された図形が何か分かるよう，グラフを作成した．なお，ある横軸の値に対して，縦軸に何も書かれていない場合は，被験者が何を提示されたのか分からなかった場合を示している.

J.S は，8 種類のうち 7 種類を正答できた．内観報告によると，線の形状を正しくなぞる事は出来なかったので，指を上下左右に動かして，大体の形状を判断していたとあった.

M.T は，8 種類のうち 2 種類を正答できた．正答できなかった形状については，線をなぞることができず，凸方向の判別もできなかった．M.T の内観報告によると，線の場所は発見できたが，その線の伸びている方向が分からず，探索を継続できなかったとあった.

4.8.4 考察

曲線と知覚され始めたスピーカ提示時間は 60 ms 以上であった．このことは，先行音効果 [21, 22, 23] の観点からも説明できる．先行音効果は，スピーカ 2 つから発せられる音の時間差が 30 ms 以下の場合に発生する．先行音効果とは，２つのスピーカから連続的に音を提示する際の時間差がある閾値より短い場合に，後から出た音が先に音の出た場所から聞こえる効果のことである．このことからも，2 音が別々の音として聞こえるためには，１スピーカあたりの提示時間を 50 ms 以上とするのがよいことになる．先行音効果を考慮した場合，40 ms はエコースレッショルドなので，人によっては先行音効果の影響で，移動感が感じにくくなる可能性がある．今回実験に参加した被験者では，この影響が見られる.

本研究に先立って，筆者は [20] にて直線の提示について検討した．この際，直線を提示する場合は，その直線の始点と終点のみを音で逐次提示するだけで良いことが分かった．本研究より曲線の提示の場合は，直線の提示と異なり，

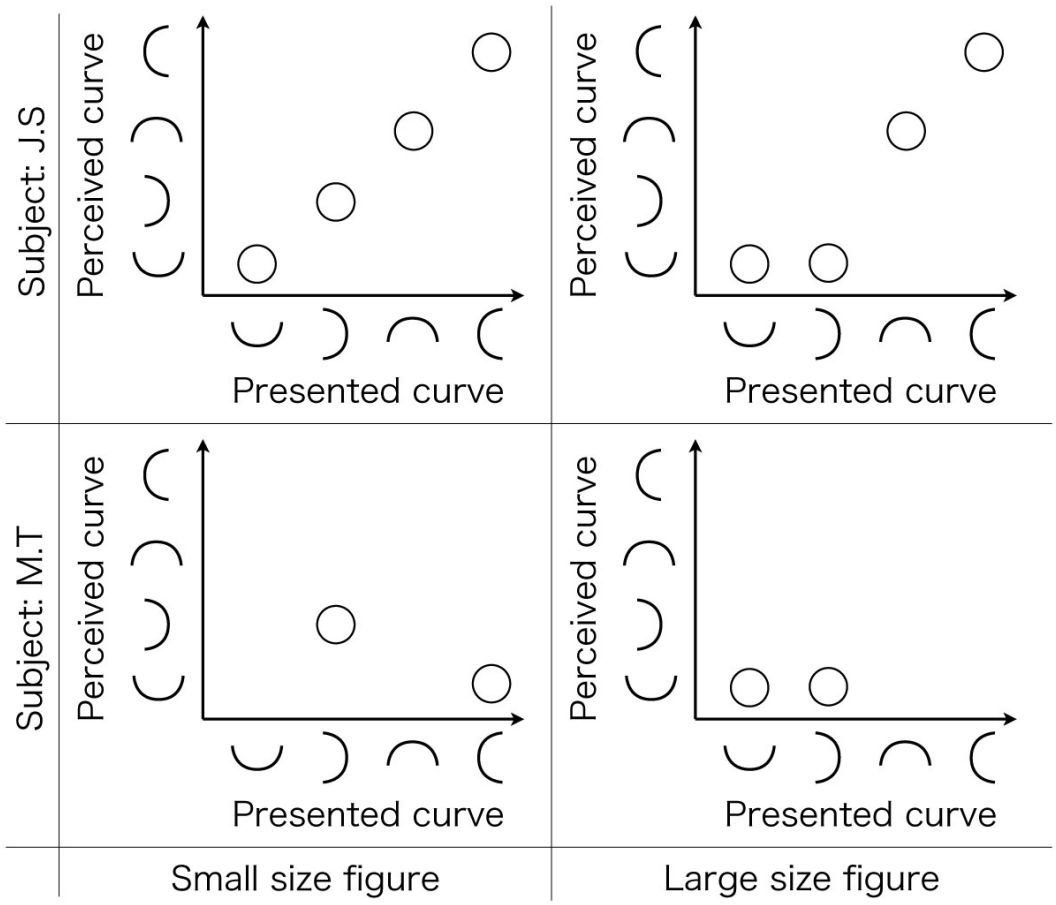


Fig. 4.17: Recognized curve in relation to presented curve.

スピーカの個数がある程度の数が必要であった．ただ，曲線の形状を厳密に表現するためには無限個のスピーカが必要であるはずである．4.7 節で用いたスピーカマトリクスは，多角形を表現したに過ぎないが，6 つのスピーカで曲線として知覚させることが出来た．曲線として知覚されるか，直線で構成された多角形として知覚されるかに関するスピーカ配置を提案するに当たっては，音源定位研究の知見と合わせて比較考察し，改めて検討を行う必要がある．

本研究において，曲線と直線の区別は 4.7 節の実験より可能であることが示唆された．しかしながら，どのような曲線が提示されたかを認識する際に困難が生じた．内観によると，線がどのような広がりを持って描かれているかを上手く提示できていないためだと考えられる．よって，どのようになぞれば図形の認識が出来るか，インタフェース側で支援する仕組みが必要である．

また，今後は，色やコントラストを，どのような音にマッピングすることで，複数の色やコントラストを含む画像を認識できるか検討していく．

参考文献

[1] 乾敏郎, 知覚と運動．乾敏郎（編），認知心理学 1 知覚と運動，東京大学出版会，pp.1-13　（1995）．

[2] J. Linvill and J. Bliss, "A direct translation reading aid for the visually impaired," *Proc, IEEE*, 54(1), pp:40–51, 1966.

[3] Dot view DV-2,
`http://www.kgs-jpn.co.jp/b_dv2.html`

[4] G. Legge, J. Mansfield, and S. Chung, "Psychophysics of reading:: Xx. linking letter recognition to reading speed in central and peripheral vision," *Vision Res.*, 41(6), pp:725–743, 2001.

[5] G Révész, "Psychology and art of the blind," London, Longmans, Green and Co., 1950.

[6] J. Gibson. "Observations on active touch," *Psychological review*, 69(6):477–491, 1962.

[7] B.C.J. Moore. "An introduction to the psychology of hearing," *Emerald Group Pub Ltd*, 2003.

[8] Y. Seki, "Acoustical design of city for the visually handicapped," *J. Acoust. Soc. Jpn.*, 54(5):387-392, 1998.

[9] T. Miura, T. Muraoka, and T. Ifukube. "Comparison of obstacle sense ability between the visually impaired and the sighted: A basic psychophysical study for designs of acoustic assistive devices," *Acoust. Sci. Tech.*, 31(2):137–147, 2010.

[10] N. Lessard, M. Pare, F. Lepore, and M. Lassonde. "Early-the visually impaired human subjects localize sound sources better than sighted subjects," *Nature*, 395(6699):278–280, 1998.

[11] J. Suzuki, T. Miura, M. Tsuchiya, K. Ueda, T. Ifukube, "Design of a loudspeaker-matrix which presents two-dimensional patterns," SICE SI2009, 2009. Dec.

[12] J. Suzuki, T. Miura, M. Tsuchiya, K. Ueda, H. Shinohara, T. Ifukube, "A display method of 2-D images using a loudspeaker matrix and its perceptional characteristics," (submitted)

[13] J. C. Makous and J. C. Middlebrooks. "Two-dimensional sound localization by human listeners," *J. Acoust. Soc. Am.*, 87(5), pp:2188–2200, 1990.

[14] H.Wallach, E.B. Newman, M.R. Rosenzweig, "The precedence effect in sound localization," *Am J Psychol*, 62(3), pp:315–36, 1949.

[15] R.Y. Litovsky, H.S. Colburn, W.A. Yost, S.J. Guzman, "The precedence effect," *J Acoust Soc Am*, 106(4 Pt 1), pp:1633–54, 1999.

[16] 鈴木 淳也, 三浦 貴大, 土屋 繭美, 上田 一貴, 伊福部 達, "二次元パタンを提示するためのスピーカマトリクスの設計," SICE SI2009, 2009.

[17] 鈴木 淳也, 三浦 貴大, 土屋 繭美, 上田 一貴, 篠原 寛明, 伊福部 達, "スピーカマトリクスによる二次元画像の提示法とその知覚特性" (submitted)

[18] J. Linvill and J. Bliss, "A direct translation reading aid for the blind," *Proc, IEEE*, 54(1), pp:40–51, 1966.

[19] G. Legge, J. Mansfield, and S. Chung, "Psychophysics of reading:: Xx. linking letter recognition to reading speed in central and peripheral vision," *Vision Res.*, 41(6), pp:725–743, 2001.

[20] J. Suzuki, T. Miura, M. Tsuchiya, H. Shinohara, T.Ifukube, "Presentation technique of 2-D diagram utilizing touch-sensitive visual display and loudspeaker matrix: aurally presentation of the corresponding line direction to contact position" (submitted)

[21] H.Wallach, E.B. Newman, M.R. Rosenzweig, "The precedence effect in sound localization," *Am J Psychol*, 62(3), pp:315–36, 1949.

[22] R.Y. Litovsky, H.S. Colburn, W.A. Yost, S.J. Guzman, "The precedence effect," *J Acoust Soc Am*, 106(4 Pt 1), pp:1633–54, 1999.

[23] 伊福部 達, "音の福祉工学", コロナ社, pp:192-197, 1997.

第5章　結章 -　本研究のまとめと今後の展望　-

視覚障碍者への画像情報の提示手段として音に着目し，第2章では，スピーカマトリクスを用いた図形提示手法の可能性について検討した．図形パタンの判断実験の結果，今回の実験で用いた 40 cm×40 cm の圧電スピーカマトリクスでは毎秒８回のインパルス列が最も効果的に図形提示が行えると結論付けた．

第3章では，複数の人が同時に聴取可能で，かつ書き順提示の可能な視覚障碍者用の音響式図形提示装置の開発を目的に，８行８列のスピーカ・マトリクスを試作し，アルファベットの大文字の書き順提示の認識実験を行った．その結果，アルファベットの大文字を，書き順に沿って移動する音で提示することで，平均正答率 90%以上が 11 文字，80%以上が 20 文字，70%以上が 24 文字であった．本実験の結果，直線，斜線，曲線を含む図形を，移動する音で提示することで認識できることを明らかにし，この手法の応用により，任意の図形を，移動する音で提示できる可能性が示された．加えて，適切な提示スピードを検討することで，より正しく図形を認識させられることがわかった．

第4章では，視覚障碍者への図形提示手段として，触れた位置にある図形の直線方向を音で提示するシステムを試作し，評価した．タッチセンサ付ディスプレイ上に表示された図形の輪郭をなぞる際，触った位置の線分方向に応じた移動音が，スピーカマトリクスから提示されるというものである．本稿で提案する「音提示図形認識装置」を用いて，直線部分については，線の方向を正しく認識した上で，輪郭線をたどり，図形の外形を把握できることが確認できた．また，角周辺では，手を細かく左右上下に動かして，次に進む線を探すことが確認された．また，本稿で提案する「音提示図形認識装置」において，２点の音の逐次提示により，移動方向を提示できることを確認した．また，３点の音の逐次提示により，90°，45°を提示できることを確認した．さらに，音でベクトルや任意の２辺挟角を繰り返し提示する際，提示の終点と，次の提示の始点の間に，無音を挟むことで，始点，終点を明確にでき，線分の傾きだけでなく，方向も示すことができることがわかった．

今後は，スピーカ数を増やすなどして，オプタコンや点図ディスプレイと同等の範囲の情報を音で提示できる手段を検討していく．また，音での輪郭提示のメリット，限界を見定めた上で，言語など他の聴覚要素での提示方法も検討していく．

その上で，音圧，周波数の音の弁別閾を活用し，さまざまな色に対応しうる

画像提示装置を検討していく．また，音での輪郭提示のメリット，限界を見定めた上で，言語など他の聴覚要素での提示方法も検討していく．

これまでの研究により,「視覚の空間分解能」を「聴覚の時間分解能」に変換することで，図形を認識できることを明らかにした．

これにより，触覚提示ではなしえない「聴覚ならではの図形提示」への道を開いた．たとえば，以下のようなユースケースが考えられる．

- 公共施設などでの，他人数への非難誘導表示
- 視覚特別支援学校などでの黒板的利用

今後は，本方式を，様々な図形やグラフの表示に活用できるように，提示スピード，刺激音などのパラメタを検討していく．また,16 行 16 列のスピーカ・マトリクスを試作し，より詳細な角度，曲線の提示をした場合，その情報量をどの程度認識できるか検討していく．

さらに, 空間音響の知見を生かし, 立体図形の表現にも挑戦していく．また, 色やコントラストを音で伝えることを目標に, 色やコントラストを音にマッピングする検討をしていく．

第6章　業績一覧

査読ありフルペーパー　鈴木淳也，守井清吾，篠原寛明, 広林茂樹, 移動する音源によるアルファベット形状の提示, 映像情報メディア学会誌, 論文特集ヒューマンインフォメーション〜情報メディアに対する人間特性の理解と応用〜, Vol. 67, No. 12 (２０１３年１２月号) p. J441-J447

国際学会プレシーディング（査読あり）　Junya Suzuki, Takahiro Miura, Mayumi Tsuchiya, Hiroaki Shinohara and Tohru Ifukube, Presentation technique of 2-D diagram utilizing touch-sensitive visual display and loudspeaker matrix: aurally presentation of the corresponding line direction to contact position, SICE SII2010, 2010.12, Sendai, Japan.

国内学会プレシーディング（査読なし）

1. 鈴木淳也, 三浦貴大, 土屋繭美, 上田一貴, 伊福部達, 二次元パタンを提示するためのスピーカマトリクスの設計, 第10回計測自動制御学会システムインテグレーション部門講演会論文集,2009.12, 芝浦工業大学豊洲キャンパス.(優秀講演賞)

2. 鈴木淳也, 三浦貴大, 土屋繭美, 篠原寛明, 伊福部達, 多音源定位による図形認識-探索部分の拡大提示法- 第11回計測自動制御学会システムインテグレーション部門講演会論文集,2010.12, 東北大学川内キャンパス.

その他業績　R. Nakada, Y. Hasegawa, S. Hirobayashi, T. Yoshizawa, T. Misawa, J. Suzuki , Wide-Area Sound-Control System for Reducing Reverberation Using Power Envelope Inverse Filtering, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E96-A, no.7, pp.1509-1517, 2013

受賞　[優秀講演賞] 鈴木淳也, 三浦貴大, 土屋繭美, 上田一貴, 伊福部達, 二次元パタンを提示するためのスピーカマトリクスの設計, 第10回計測自動制御学会システムインテグレーション部門講演会論文集,2009.12, 芝浦工業大学豊洲キャンパス.

第7章　謝辞

本研究は，2008 年から 5 年にわたり積み重ねてきました．研究を進めていく上で、多くの先生方のご指導をいただきました。また、伊福部研究室（東京大学)、廣林研究室（富山大学）の皆様とたくさんの議論を深めることができました。その他にも、さまざまな分野の方々にたくさんのご協力をいただきました。ここに、深く感謝の気持ちを記したいと思います。

以下に、本博士論文を執筆する上で、特に、深くご指導、ご協力いただいた方のお名前を記載し、感謝の意を示します。

本研究の所期の段階（2008 年）から、東京大学・先端化学技術センターの伊福部達先生に,「感覚代行」の考え方をご指導いただき,「図形（視覚情報)」を「音の動き（聴覚情報)」に変換する研究に着手することができました。また、同研究室の三浦貴大氏，土屋繭美氏，上田一貴氏には，実験の立案，図の作成をはじめ、研究のさまざまな側面でご協力いただきました．同研究室の藪謙一郎氏には，特に、ロジック回路についてご指導，ご協力をいただき，スピーカ切り替え装置を実現し，制御可能なスピーカの数を格段に増やすことができるようになりました。

富山大学の津田正明先生には, 博士取得への道を開いていただき、多くの励ましをいただきました。また、篠原寛明先生には、研究のフレームワークの立案方法についてご指導いただき、揺るぎのない研究を進めることができました。広林茂樹先生に，実験方法，分析方法、また論文執筆にいたるまで、ご指導いただきました．特に、「研究の訴求方法」についてご示唆をいただき、論文執筆のポイントを学ばせていただきました。また、西条寿夫先生には，解剖学的視点で多くのご助言をいただきました。また、安藤彰男先生には、空間音響のご専門のお立場から、詳細なご指導をいただきました。特に、用語の適切な解釈方法を学ばせていただきました。また、守井清吾氏とは、「視覚障碍者に図形情報を提示する」という同じ研究目的の元、深い議論をし、切磋琢磨してくることができました。また, 小澤 恭平君, 高木　裕久君に, 論文の図の作成をご協力いただきました。

なお，本研究の一部は，JST A-STEP(探索タイプ AS242Z00714K) の助成による成果である．

上記のとおり、ほんとうに多くの皆様のおかげで研究を進めることができ、この博士論文をまとめることができました。皆様に深く感謝するとともに、研究を発展させることこそが、皆様への恩返しだと思います。これからも、一生懸命、研究を進めていきますので、引き続き、ご指導、ご協力のほど、よろしくお願いいたします。

2014 年 3 月
鈴木淳也

付録 A　　スピーカ切り替え装置の原理および回路図

第 3 章の実験用に試作したスピーカ切り替え装置の回路図を記す.

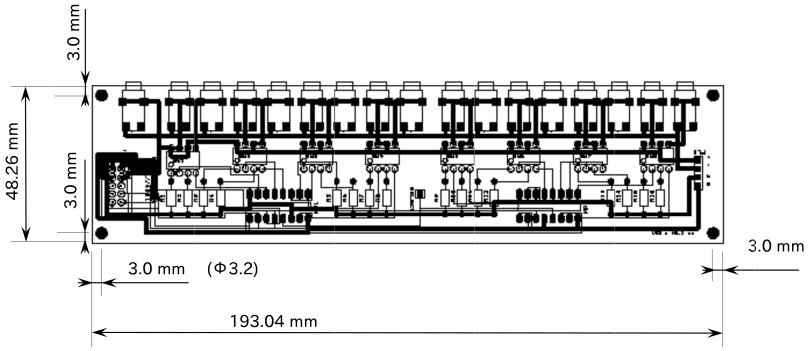


Fig. 1: 基板の外形図

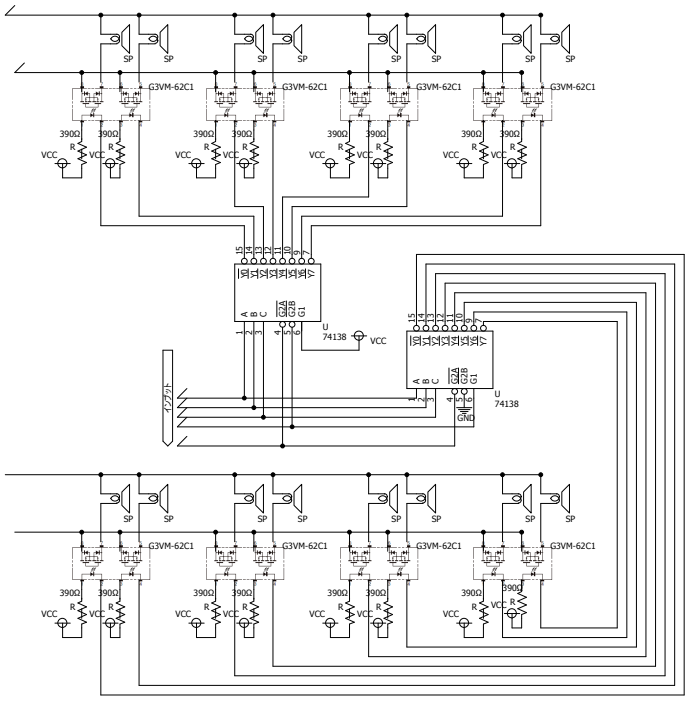


Fig. 2: スピーカ切り替え回路

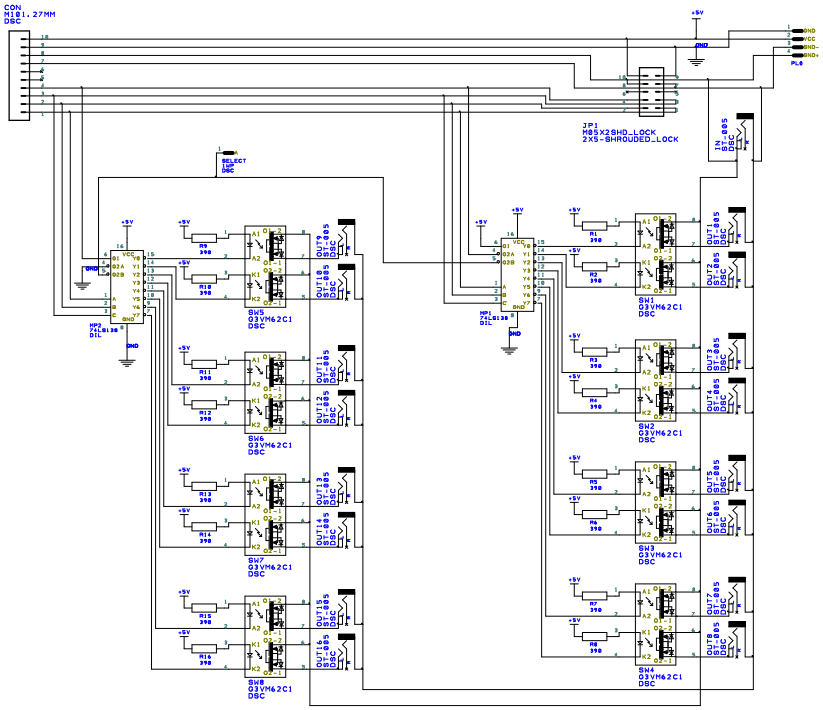


Fig. 3: 基板の回路図

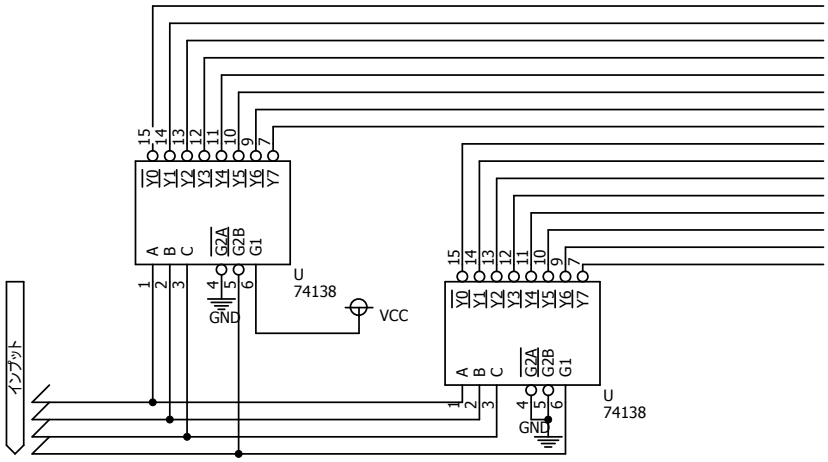


Fig. 4: 4 ビットで振り分ける部分

付録B　　第3章の実験用プログラム・ソース

リスト 1: sample source

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Speech;
using System.Speech.Synthesis;
using Microsoft.DirectX;
using Microsoft.DirectX.DirectSound;

namespace sound_blackboard
{
    public partial class Form1 : Form
    {
        private Pas500w Pasw;
        short wLogSlot; // デバイスの論理スロット番号を格納
        byte cbDat;
        string szBuf;
        SpeechSynthesizer syn = new SpeechSynthesizer();
        int interval_val=50;
        int mute_dur=1;
        int[] array1=null;
        int[] array2=null;
        int[] array3=null;
        int[] array4=null;
        int[] array5=null;
        int[] array6=null;
        int[] array7=null;
        int[] array8=null;
        int[] array9=null;
        int[] array10=null;
        int[] array=null;
        int[] rest=null;
        int _sp64=64;
        int pen=0;
        byte mute=254; // 無音用ビット const

        private Device _device = null;
        private SecondaryBuffer _buffer = null;
        private Guid[] id = new Guid[20];



        private void use_count(int[] array, int[]unuse)
        {
             for (int k=0; k<array.Length; k++)
             {
                  for (int l=1; l<=64; l++)
                  {
                       if (array[k]==l)
                       {
                            unuse[l-1]++;
                       }
                  }
             }
        }

        public Form1()
        {
            InitializeComponent();
        }
```

```
66
67          private void Form1_Load(object sender, EventArgs e)
68          {
69              // ----- PA-S500接続 -----
70              Pas500w.Resource Ri;     // リソース情報
71              int dwSwVal = 0;         // ロータリースイッチの値
72              int dwDirections = 0;    // ディレクション
73              bool bResult;            // 処理結果
74              int i;
75
76              // 生成
77              Pasw = new Pas500w();
78              Ri = new Pas500w.Resource();
79              Ri.dwMemBase = new int[Pas500w.MAX_MEM];
80              Ri.dwMemLength = new int[Pas500w.MAX_MEM];
81              Ri.dwMemAttrib = new int[Pas500w.MAX_MEM];
82              Ri.dwIOPortBase = new int[Pas500w.MAX_IO];
83              Ri.dwIOPortLength = new int[Pas500w.MAX_IO];
84              Ri.dwIRQRegisters = new int[Pas500w.MAX_IRQ];
85              Ri.dwIRQAttrib = new int[Pas500w.MAX_IRQ];
86              Ri.dwDMALst = new int[Pas500w.MAX_DMA];
87              Ri.dwDMAAttrib = new int[Pas500w.MAX_DMA];
88              Ri.dwReserved1 = new int[Pas500w.MAX_RSV];
89
90              // 初期化
91              Ri.dwNumMemWindows = 0;
92              for (i = 0; i < Pas500w.MAX_MEM; i++)
93              {
94                  Ri.dwMemBase[i] = 0;
95                  Ri.dwMemLength[i] = 0;
96                  Ri.dwMemAttrib[i] = 0;
97              }
98              Ri.dwNumIOPorts = 0;
99              for (i = 0; i < Pas500w.MAX_IO; i++)
100             {
101                 Ri.dwIOPortBase[i] = 0;
102                 Ri.dwIOPortLength[i] = 0;
103             }
104             Ri.dwNumIRQs = 0;
105             for (i = 0; i < Pas500w.MAX_IRQ; i++)
106             {
107                 Ri.dwIRQRegisters[i] = 0;
108                 Ri.dwIRQAttrib[i] = 0;
109             }
110             Ri.dwNumDMAs = 0;
111             for (i = 0; i < Pas500w.MAX_DMA; i++)
112             {
113                 Ri.dwDMALst[i] = 0;
114                 Ri.dwDMAAttrib[i] = 0;
115             }
116             for (i = 0; i < Pas500w.MAX_RSV; i++)
117             {
118                 Ri.dwReserved1[i] = 0;
119             }
120
121             //-------------------------------------------------
122             // PA-S500の使用を宣言
123             //-------------------------------------------------
124             wLogSlot = Pas500w.SLOT_AUTO;           // 未使用のデバイスを探します。
125             bResult = Pasw.Create(ref wLogSlot);    // 成功した場合、その時のデバイスの論理スロット番号が格納されます。
126             if (bResult == false)                   // デバイスが見つかりません。
127             {
128                 ErrorMessage(Pas500w.SLOT_AUTO);
129                 //this.Close();
130                 return;
131             }
132             szBuf = string.Concat("PA-S500 Created", Environment.NewLine,
                    Environment.NewLine, string.Format("LogSlot: {0}", wLogSlot));
```

```
133             ResultMessage(szBuf);
134             //-----------------------------------------------
135             // リソース情報とロータリースイッチの値を取得
136             //-----------------------------------------------
137             IntPtr pri = Marshal.AllocHGlobal(Marshal.SizeOf(Ri));  // アンマ
                    ネージメモリ確保
138             Marshal.StructureToPtr(Ri, pri, false);                 // マーシ
                    ャリング(マネージメモリ->アンマネージメモリ)
139             bResult = Pasw.GetResource(wLogSlot, pri);
140             Ri = (Pas500w.Resource)(Marshal.PtrToStructure(pri, typeof(Pas500w
                    .Resource))); // マーシャリング(アンマネージメモリ->マネージメ
                    モリ)
141             Marshal.FreeHGlobal(pri);                               // メモリ
                    を解放
142             if (bResult == false)
143             {
144                 ErrorMessage(wLogSlot);
145                 //this.Close();
146                 return;
147             }
148
149             if (Pasw.GetSwitchValue(wLogSlot, ref dwSwVal) == false)
150             {
151                 ErrorMessage(wLogSlot);
152                 //this.Close();
153                 return;
154             }
155             szBuf = string.Concat("Board Name: PA-S500", Environment.NewLine,
                    string.Format("IO Address  : {0:x4}h - {1:x4}h", Ri.
                    dwIOPortBase[0], Ri.dwIOPortBase[0] + Ri.dwIOPortLength[0] -
                    1), Environment.NewLine, string.Format("Switch        : {0:x}h
                    ", dwSwVal));
156             ResultMessage(szBuf);
157             //-----------------------------------------------
158             // すべてのPortのディレクションを出力に設定する
159             //-----------------------------------------------
160             if (Pasw.GetDirections(wLogSlot, ref dwDirections) == false)
161             {
162                 ErrorMessage(wLogSlot);
163                 //this.Close();
164                 return;
165             }
166             dwDirections = ~Pas500w.DIR_A_INPUT & dwDirections;   // A Port の
                    ビットを出力に変更します。
167             if (Pasw.SetDirections(wLogSlot, dwDirections) == false)
168             {
169                 ErrorMessage(wLogSlot);
170                 //this.Close();
171                 return;
172             }
173             ResultMessage("Port A のディレクションを出力に変更");
174
175             if (Pasw.GetDirections(wLogSlot, ref dwDirections) == false)
176             {
177                 ErrorMessage(wLogSlot);
178                 //this.Close();
179                 return;
180             }
181             dwDirections = ~Pas500w.DIR_B_INPUT & dwDirections;   // B Port の
                    ビットを出力に変更します。
182             if (Pasw.SetDirections(wLogSlot, dwDirections) == false)
183             {
184                 ErrorMessage(wLogSlot);
185                 //this.Close();
186                 return;
187             }
188             ResultMessage("Port B のディレクションを出力に変更");
189
190             if (Pasw.GetDirections(wLogSlot, ref dwDirections) == false)
191             {
192                 ErrorMessage(wLogSlot);
```

```
193                 //this.Close();
194                 return;
195             }
196             dwDirections = ~Pas500w.DIR_C_INPUT & dwDirections;   // C Port の
                    ビットを出力に変更します。
197             if (Pasw.SetDirections(wLogSlot, dwDirections) == false)
198             {
199                 ErrorMessage(wLogSlot);
200                 //this.Close();
201                 return;
202             }
203             ResultMessage("Port C のディレクションを出力に変更");
204
205             if (Pasw.GetDirections(wLogSlot, ref dwDirections) == false)
206             {
207                 ErrorMessage(wLogSlot);
208                 //this.Close();
209                 return;
210             }
211             dwDirections = ~Pas500w.DIR_D_INPUT & dwDirections;   // D Port の
                    ビットを出力に変更します。
212             if (Pasw.SetDirections(wLogSlot, dwDirections) == false)
213             {
214                 ErrorMessage(wLogSlot);
215                 //this.Close();
216                 return;
217             }
218             ResultMessage("Port D のディレクションを出力に変更");
219
220             if (Pasw.GetDirections(wLogSlot, ref dwDirections) == false)
221             {
222                 ErrorMessage(wLogSlot);
223                 //this.Close();
224                 return;
225             }
226             dwDirections = ~Pas500w.DIR_E_INPUT & dwDirections;   // E Port の
                    ビットを出力に変更します。
227             if (Pasw.SetDirections(wLogSlot, dwDirections) == false)
228             {
229                 ErrorMessage(wLogSlot);
230                 //this.Close();
231                 return;
232             }
233             ResultMessage("Port E のディレクションを出力に変更");
234
235             if (Pasw.GetDirections(wLogSlot, ref dwDirections) == false)
236             {
237                 ErrorMessage(wLogSlot);
238                 //this.Close();
239                 return;
240             }
241             dwDirections = ~Pas500w.DIR_F_INPUT & dwDirections;   // F Port の
                    ビットを出力に変更します。
242             if (Pasw.SetDirections(wLogSlot, dwDirections) == false)
243             {
244                 ErrorMessage(wLogSlot);
245                 //this.Close();
246                 return;
247             }
248             ResultMessage("Port F のディレクションを出力に変更");
249
250              cbDat = mute;
251              if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
252              {
253                      ErrorMessage(wLogSlot);
254                      //this.Close();
255                      return;
256              }
257              //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                     &H" + Convert.ToString(cbDat, 16).ToUpper();
258              //ResultMessage(szBuf);
```

```
259
260                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_B, cbDat) == false)
261                 {
262                         ErrorMessage(wLogSlot);
263                         //this.Close();
264                         return;
265                 }
266                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
267                 //ResultMessage(szBuf);
268
269                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_C, cbDat) == false)
270                 {
271                         ErrorMessage(wLogSlot);
272                         //this.Close();
273                         return;
274                 }
275                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
276                 //ResultMessage(szBuf);
277
278                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_D, cbDat) == false)
279                 {
280                         ErrorMessage(wLogSlot);
281                         //this.Close();
282                         return;
283                 }
284                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
285                 //ResultMessage(szBuf);
286
287                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_E, cbDat) == false)
288                 {
289                         ErrorMessage(wLogSlot);
290                         //this.Close();
291                         return;
292                 }
293                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
294                 //ResultMessage(szBuf);
295
296                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_F, cbDat) == false)
297                 {
298                         ErrorMessage(wLogSlot);
299                         //this.Close();
300                         return;
301                 }
302                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
303                 //ResultMessage(szBuf);
304
305
306               // サウンドデバイスの取得
307               DevicesCollection devices = new DevicesCollection();
308               int j = 0;
309               foreach (DeviceInformation devInfo in devices)
310               {
311                   id[j] = devInfo.DriverGuid;
312                   //devName[i]=devInfo.Description;
313                   j++;
314             }
315
316             try
317             {
318                 //this._device = new Device(id[2]);
319                 this._device = new Device();
320             }
321             catch (Exception )
322             {
323                 //MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK
                        , MessageBoxIcon.Error);
```

```
324             }
325
326             this._device.SetCooperativeLevel(this, CooperativeLevel.Normal);
327
328             BufferDescription BufferDescription = new BufferDescription();
329             BufferDescription.ControlVolume = true;
330             BufferDescription.ControlPan = true;
331             BufferDescription.ControlFrequency = true;
332             BufferDescription.GlobalFocus = false;
333
334             this._buffer = new SecondaryBuffer("tones/White.wav", this._device
                    );
335
336             this._buffer.Pan=0;
337
338
339
340               syn.Speak("準備できました。");
341         }
342
343         //=========================================
344         //    処理結果を表示
345         //=========================================
346         private void ResultMessage(string lpszbuf)
347         {
348             MessageBox.Show(lpszbuf, this.Text, MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
349         }
350         //=========================================
351         //    エラーメッセージを表示
352         //=========================================
353         private void ErrorMessage(short wLogSlot)
354         {
355             long dwRes = 0;
356             string szBuf = null;
357
358             dwRes = Pasw.GetLastError(wLogSlot);
359             switch (dwRes)
360             {
361                 case Pas500w.ERR_SYSTEM:
362                     dwRes = System.Runtime.InteropServices.Marshal.
                            GetLastWin32Error();
363                     szBuf = string.Concat("System error", Environment.NewLine,
                             Environment.NewLine, string.Format("Error code: &H{0:
                            X8}", dwRes));
364                     break;
365                 default:
366                     szBuf = string.Concat("PA-S500 error", Environment.NewLine
                            , Environment.NewLine, string.Format("Error code: &H
                            {0:X8}", dwRes));
367                     break;
368             }
369             MessageBox.Show(szBuf, this.Text, MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
370         }
371
372         private void Form1_FormClosed(object sender, FormClosedEventArgs e)
373         {
374             // PA-S500の使用を開放
375             //-------------------------------------------------
376             if (Pasw.Close(wLogSlot) == false)
377             {
378                 ErrorMessage(wLogSlot);
379                 //this.Close();
380                 return;
381             }
382             ResultMessage("PA-S500 Closed");
383             //this.Close();
384
385         }
386
```

```
387         private void sw_on(byte cbDat)
388         {
389             if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
390             {
391                 ErrorMessage(wLogSlot);
392                 //this.Close();
393                 return;
394             }
395         }
396 
397         int noise_OnOff=0;
398         int[] unuse=new int[64];
399         private void Form1_KeyDown(object sender, KeyEventArgs e)
400         {
401              // ----- 停止・無音
402              if( e.KeyCode == Keys.Escape  )
403              {
404                  cbDat = mute;
405                  if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
406                  {
407                          ErrorMessage(wLogSlot);
408                          //this.Close();
409                          return;
410                  }
411                  //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                         &H" + Convert.ToString(cbDat, 16).ToUpper();
412                  //ResultMessage(szBuf);
413                   this.Text=cbDat.ToString();
414                   syn.Speak(cbDat.ToString());
415              }
416 
417              // ----- スピーカ 1 から出力
418              if( e.KeyCode == Keys.F1  )
419              {
420                  cbDat = 0;
421                  if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
422                  {
423                          ErrorMessage(wLogSlot);
424                          //this.Close();
425                          return;
426                  }
427                  //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                         &H" + Convert.ToString(cbDat, 16).ToUpper();
428                  //ResultMessage(szBuf);
429                   this.Text=cbDat.ToString();
430                   syn.Speak(cbDat.ToString());
431              }
432 
433              // ----- 出力スピーカをデクリメント
434              if( e.KeyCode == Keys.F2  )
435              {
436                  if (cbDat > 0)
437                  {
438                       cbDat--;
439                  }
440                  if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
441                  {
442                          ErrorMessage(wLogSlot);
443                          //this.Close();
444                          return;
445                  }
446                  //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                         &H" + Convert.ToString(cbDat, 16).ToUpper();
447                  //ResultMessage(szBuf);
448                   this.Text=cbDat.ToString();
449                   syn.Speak(cbDat.ToString());
450              }
451 
452              // ----- 出力スピーカをインクリメント
453              if( e.KeyCode == Keys.F3  )
454              {
```

```
455                 if (cbDat < 255)
456                 {
457                      cbDat++;
458                 }
459                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
460                 {
461                         ErrorMessage(wLogSlot);
462                         //this.Close();
463                         return;
464                 }
465                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
466                 //ResultMessage(szBuf);
467                  this.Text=cbDat.ToString();
468                  syn.Speak(cbDat.ToString());
469             }
470
471             // ----- スピーカ63から出力
472             if( e.KeyCode == Keys.F4  )
473             {
474                 cbDat = 63;
475                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
476                 {
477                         ErrorMessage(wLogSlot);
478                         //this.Close();
479                         return;
480                 }
481                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
482                 //ResultMessage(szBuf);
483                  this.Text=cbDat.ToString();
484                  syn.Speak(cbDat.ToString());
485             }
486
487             // ----- スピーカ8から出力
488             if( e.KeyCode == Keys.F5  )
489             {
490                 cbDat = 8;
491                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
492                 {
493                         ErrorMessage(wLogSlot);
494                         //this.Close();
495                         return;
496                 }
497                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
498                 //ResultMessage(szBuf);
499                  this.Text=cbDat.ToString();
500                  syn.Speak(cbDat.ToString());
501             }
502
503             // ----- スピーカ16から出力
504             if( e.KeyCode == Keys.F6  )
505             {
506                 cbDat = 16;
507                 if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
508                 {
509                         ErrorMessage(wLogSlot);
510                         //this.Close();
511                         return;
512                 }
513                 //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                        &H" + Convert.ToString(cbDat, 16).ToUpper();
514                 //ResultMessage(szBuf);
515                  this.Text=cbDat.ToString();
516                  syn.Speak(cbDat.ToString());
517             }
518
519             // ----- スピーカ24から出力
520             if( e.KeyCode == Keys.F7  )
521             {
```

```
522                     cbDat = 24;
523                     if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
524                     {
525                             ErrorMessage(wLogSlot);
526                             //this.Close();
527                             return;
528                     }
529                     //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                            &H" + Convert.ToString(cbDat, 16).ToUpper();
530                     //ResultMessage(szBuf);
531                      this.Text=cbDat.ToString();
532                      syn.Speak(cbDat.ToString());
533             }
534
535             // ----- スピーカ32から出力
536             if( e.KeyCode == Keys.F8  )
537             {
538                     cbDat = 32;
539                     if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
540                     {
541                             ErrorMessage(wLogSlot);
542                             //this.Close();
543                             return;
544                     }
545                     //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                            &H" + Convert.ToString(cbDat, 16).ToUpper();
546                     //ResultMessage(szBuf);
547                      this.Text=cbDat.ToString();
548                      syn.Speak(cbDat.ToString());
549             }
550
551             // ----- スピーカ40から出力
552             if( e.KeyCode == Keys.F9  )
553             {
554                     cbDat = 40;
555                     if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
556                     {
557                             ErrorMessage(wLogSlot);
558                             //this.Close();
559                             return;
560                     }
561                     //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                            &H" + Convert.ToString(cbDat, 16).ToUpper();
562                     //ResultMessage(szBuf);
563                      this.Text=cbDat.ToString();
564                      syn.Speak(cbDat.ToString());
565             }
566
567             // ----- スピーカ48から出力
568             if( e.KeyCode == Keys.F11  )
569             {
570                     cbDat = 48;
571                     if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
572                     {
573                             ErrorMessage(wLogSlot);
574                             //this.Close();
575                             return;
576                     }
577                     //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                            &H" + Convert.ToString(cbDat, 16).ToUpper();
578                     //ResultMessage(szBuf);
579                      this.Text=cbDat.ToString();
580                      syn.Speak(cbDat.ToString());
581             }
582
583             // ----- スピーカ56から出力
584             if( e.KeyCode == Keys.F12  )
585             {
586                     cbDat = 56;
587                     if (Pasw.OutPort(wLogSlot, Pas500w.PORT_A, cbDat) == false)
588                     {
```

```
589                         ErrorMessage(wLogSlot);
590                         //this.Close();
591                         return;
592                     }
593                     //szBuf = "Port A へデータを出力" + "\n" + "Port A Out Data:
                            &H" + Convert.ToString(cbDat, 16).ToUpper();
594                     //ResultMessage(szBuf);
595                      this.Text=cbDat.ToString();
596                      syn.Speak(cbDat.ToString());
597                 }
598
599                 if( e.KeyCode == Keys.Space )
600                 {
601                      this._buffer.Stop();
602                    this._buffer = new SecondaryBuffer("tones/White.wav", this.
                           _device);
603                    this._buffer.Pan=0;
604
605                    if (noise_OnOff==0)
606                    {
607                         this._buffer.Play(0, BufferPlayFlags.Looping);
608                         noise_OnOff=1;
609                    } else {
610                         this._buffer.Stop();
611                         noise_OnOff=0;
612                    }
613
614                 }
615
616
617
618 // ----- 刺激提示用トリガー
619                 if( e.KeyCode == Keys.A  )
620                 {
621                       array1=new int[]{mute, 8, 15, 22, 28, 36, 43, 50, 57}; // -
                              p8
622                       array2=new int[]{8, 16, 24, 32, 40, 48, 56, _sp64}; // -v8
623                       array3=new int[]{36, 37, 38, 39, 40, mute, -1}; // h5
624                       array=new int[100];
625                       rest=new int[mute_dur];
626                       for (int i=0; i<mute_dur; i++)
627                       {
628                            rest[i]=mute;
629                       }
630
631                       array = array1.Concat(rest).ToArray();
632                       array = array.Concat(array2).ToArray();
633                       array = array.Concat(rest).ToArray();
634                       array = array.Concat(array3).ToArray();
635                       use_count(array, unuse);
636
637                       timer1.Interval=interval_val;
638                       timer1.Enabled=true;
639                 }
640
641                 if( e.KeyCode == Keys.Tab  )
642                 {
643                                 System.IO.StreamWriter sw = new System.IO.
                                        StreamWriter(
644                                     @"test.txt",
645                                     false,
646                                     System.Text.Encoding.GetEncoding("shift_jis"
                                            ));
647                       for (int m=0; m<64; m++)
648                       {
649                            if (unuse[m]==2)
650                            {
651                                  sw.WriteLine("2times:"+(m+1).ToString()+"\n");
652                            }
653                            if (unuse[m]==1)
654                            {
```

```
655                               sw.WriteLine("a times:"+(m+1).ToString()+"\n");
656                         }
657                         if (unuse[m]==0)
658                         {
659                               sw.WriteLine("no times:"+(m+1).ToString()+"\n");
660                         }
661                   }
662
663                               sw.Close();
664
665                   syn.Speak("Done");
666             }
667
668             if( e.KeyCode == Keys.B  )
669             {
670                   array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                          v8
671                   array2=new int[]{1, 2, 3, 4, 5, 6,   15, 23,    30, 29, 28,
                           27, 26, 25}; // h6-v2-h6
672                   array3=new int[]{25, 26, 27, 28, 29, 30,     39, 48,
                          55, 62,      61, 60, 59, 58, 57, mute, -1}; // h6-m2-p2
                          -h5
673                   array=new int[100];
674                   rest=new int[mute_dur];
675                   for (int i=0; i<mute_dur; i++)
676                   {
677                         rest[i]=mute;
678                   }
679
680                   array = array1.Concat(rest).ToArray();
681                   array = array.Concat(array2).ToArray();
682                   array = array.Concat(rest).ToArray();
683                   array = array.Concat(array3).ToArray();
684                   use_count(array, unuse);
685
686                   timer1.Interval=interval_val;
687                   timer1.Enabled=true;
688
689             }
690
691             if( e.KeyCode == Keys.C  )
692             {
693                   array=new int[]{mute, 23, 14, 5,     4, 11, 18, 25,     33,
                           42, 51, 60,    61, 54, 47, mute, -1}; // m3-p4-m4p3
694                   use_count(array, unuse);
695
696                   timer1.Interval=interval_val;
697                   timer1.Enabled=true;
698             }
699
700             if( e.KeyCode == Keys.D  )
701             {
702                   array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                          v8
703                   array2=new int[]{1, 2, 3, 4, 5,     14, 23, 32,     40, 47,
                           54, 61,     60, 59, 58, 57, mute, -1}; // h5-m3-p4-h4
704
705                   array=new int[100];
706                   rest=new int[mute_dur];
707                   for (int i=0; i<mute_dur; i++)
708                   {
709                         rest[i]=mute;
710                   }
711
712                   array = array1.Concat(rest).ToArray();
713                   array = array.Concat(array2).ToArray();
714                   use_count(array, unuse);
715
716                   timer1.Interval=interval_val;
717                   timer1.Enabled=true;
718
```

```
719
720
721                 }
722
723                 if( e.KeyCode == Keys.E  )
724                 {
725                       array1=new int[]{mute, 1, 2, 3, 4, 5, 6, 7, 8}; // -h8
726                       array2=new int[]{1, 9, 17, 25, 33, 41, 49, 57}; // -v8
727                       array3=new int[]{25, 26, 27, 28, 29, 30}; // h6
728                       array4=new int[]{57, 58, 59, 60, 61, 62, 63, _sp64, mute,
                              -1}; // h8
729
730                       array=new int[100];
731                       rest=new int[mute_dur];
732                       for (int i=0; i<mute_dur; i++)
733                       {
734                            rest[i]=mute;
735                       }
736
737                       array = array1.Concat(rest).ToArray();
738                       array = array.Concat(array2).ToArray();
739                       array = array.Concat(rest).ToArray();
740                       array = array.Concat(array3).ToArray();
741                       array = array.Concat(rest).ToArray();
742                       array = array.Concat(array4).ToArray();
743                       use_count(array, unuse);
744
745                       timer1.Interval=interval_val;
746                       timer1.Enabled=true;
747
748
749
750                 }
751
752                 if( e.KeyCode == Keys.F  )
753                 {
754                       array1=new int[]{mute, 1, 2, 3, 4, 5, 6, 7, 8}; // h8
755                       array2=new int[]{1, 9, 17, 25, 33, 41, 49, 57}; // -v8
756                       array3=new int[]{25, 26, 27, 28, 29, 30, mute, -1}; // h6
757
758                       array=new int[100];
759                       rest=new int[mute_dur];
760                       for (int i=0; i<mute_dur; i++)
761                       {
762                            rest[i]=mute;
763                       }
764
765                       array = array1.Concat(rest).ToArray();
766                       array = array.Concat(array2).ToArray();
767                       array = array.Concat(rest).ToArray();
768                       array = array.Concat(array3).ToArray();
769                       use_count(array, unuse);
770
771                       timer1.Interval=interval_val;
772                       timer1.Enabled=true;
773                 }
774
775                 if( e.KeyCode == Keys.G  )
776                 {
777                       //array1=new int[]{0, 24, 15, 6, 5, 4, 3, 10, 17, 25, 33,
                              41, 50, 59, 60, 61, 62, 55, 48};
778                       array1=new int[]{mute, 23, 14, 5,     4, 11, 18, 25,
                              33, 42, 51, 60,     61, 54, 47}; // m3-p4-m4p3
779                       array2=new int[]{45, 46, 47, 48}; // h4
780                       array3=new int[]{48, 56, _sp64, mute, -1}; // -v3
781
782                       array=new int[100];
783                       rest=new int[mute_dur];
784                       for (int i=0; i<mute_dur; i++)
785                       {
786                            rest[i]=mute;
```

```
787                     }
788
789                     array = array1.Concat(rest).ToArray();
790                     array = array.Concat(array2).ToArray();
791                     array = array.Concat(rest).ToArray();
792                     array = array.Concat(array3).ToArray();
793                     use_count(array, unuse);
794
795                     timer1.Interval=interval_val;
796                     timer1.Enabled=true;
797                 }
798
799
800                 if( e.KeyCode == Keys.H  )
801                 {
802                     array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                            v8
803                     array2=new int[]{25, 26, 27, 28, 29, 30, 31, 32}; // h8
804                     array3=new int[]{8, 16, 24, 32, 40, 48, 56, _sp64, mute,
                            -1}; // -v8
805
806                     array=new int[100];
807                     rest=new int[mute_dur];
808                     for (int i=0; i<mute_dur; i++)
809                     {
810                          rest[i]=mute;
811                     }
812
813                     array = array1.Concat(rest).ToArray();
814                     array = array.Concat(array2).ToArray();
815                     array = array.Concat(rest).ToArray();
816                     array = array.Concat(array3).ToArray();
817                     use_count(array, unuse);
818
819                     timer1.Interval=interval_val;
820                     timer1.Enabled=true;
821                 }
822
823                 if( e.KeyCode == Keys.I  )
824                 {
825                     array=new int[]{mute, 4, 12, 20, 28, 36, 44, 52, 60, mute,
                            -1}; // -v8
826                     use_count(array, unuse);
827
828                     timer1.Interval=interval_val;
829                     timer1.Enabled=true;
830                 }
831
832                 if( e.KeyCode == Keys.J  )
833                 {
834                     array=new int[]{mute, 7, 15, 23, 31, 31, 39, 47,     54,
                            61,     60, 51, 42, 33, mute, -1};
835                     use_count(array, unuse);
836
837                     timer1.Interval=interval_val;
838                     timer1.Enabled=true;
839                 }
840
841                 if( e.KeyCode == Keys.K  )
842                 {
843                     array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                            v8
844                     array2=new int[]{5, 12, 19, 26}; // -p4 // この行と次の行が
                            「く」になるようにすること
845                     array3=new int[]{26, 35, 44, 53, 62, mute, -1}; // -m5
846
847                     array=new int[100];
848                     rest=new int[mute_dur];
849                     for (int i=0; i<mute_dur; i++)
850                     {
851                          rest[i]=mute;
```

```
852                     }
853
854                     array = array1.Concat(rest).ToArray();
855                     array = array.Concat(array2).ToArray();
856                     array = array.Concat(rest).ToArray();
857                     array = array.Concat(array3).ToArray();
858                     use_count(array, unuse);
859
860                     timer1.Interval=interval_val;
861                     timer1.Enabled=true;
862             }
863
864             if( e.KeyCode == Keys.L  )
865             {
866                     array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                            v8
867                     array2=new int[]{57, 58, 59, 60, 61, 62, 63, _sp64, mute,
                            -1}; // h8
868
869                     array=new int[100];
870                     rest=new int[mute_dur];
871                     for (int i=0; i<mute_dur; i++)
872                     {
873                          rest[i]=mute;
874                     }
875
876                     array = array1.Concat(rest).ToArray();
877                     array = array.Concat(array2).ToArray();
878                     use_count(array, unuse);
879
880                     timer1.Interval=interval_val;
881                     timer1.Enabled=true;
882             }
883
884 // M:斜辺部が1段下がっている。確認すること。
885             if( e.KeyCode == Keys.M  )
886             {
887                     array1=new int[]{mute, 57, 49, 41, 33, 25, 17, 9, 1}; // v8
888                     array2=new int[]{2, 11, 20, 29}; // p4  // 一段下がっている
                            のでは・確認すること。
889                     array3=new int[]{29, 22, 15, 8}; // -m4 // ここも一段下がっ
                            ているのでは？
890                     array4=new int[]{8, 16, 24, 32, 40, 48, 56, _sp64, mute,
                            -1}; // -v8
891
892                     array=new int[100];
893                     rest=new int[mute_dur];
894                     for (int i=0; i<mute_dur; i++)
895                     {
896                          rest[i]=mute;
897                     }
898
899                     array = array1.Concat(rest).ToArray();
900                     array = array.Concat(array2).ToArray();
901                     array = array.Concat(rest).ToArray();
902                     array = array.Concat(array3).ToArray();
903                     array = array.Concat(rest).ToArray();
904                     array = array.Concat(array4).ToArray();
905                     use_count(array, unuse);
906
907                     timer1.Interval=interval_val;
908                     timer1.Enabled=true;
909             }
910
911             if( e.KeyCode == Keys.N  )
912             {
913                     array1=new int[]{mute, 57, 49, 41, 33, 25, 17, 9, 1}; // v8
914                     array2=new int[]{2, 11, 20, 29, 38, 47, 56}; // -m7
915                     array3=new int[]{_sp64, 56, 48, 40, 32, 24, 16, 8, mute,
                            -1}; // v8
916
```

```
917                     array=new int[100];
918                     rest=new int[mute_dur];
919                     for (int i=0; i<mute_dur; i++)
920                     {
921                          rest[i]=mute;
922                     }
923 
924                     array = array1.Concat(rest).ToArray();
925                     array = array.Concat(array2).ToArray();
926                     array = array.Concat(rest).ToArray();
927                     array = array.Concat(array3).ToArray();
928                     use_count(array, unuse);
929 
930                     timer1.Interval=interval_val;
931                     timer1.Enabled=true;
932                 }
933 
934                 if( e.KeyCode == Keys.O  )
935                 {
936                     array=new int[]{mute, 24, 15, 6,    5, 4, 3,     10, 17,
                                25, 33, 41,     50, 59,     60, 61, 62,     55, 48,
                                 40, 32, 24, mute, -1}; // m3-h3-p2-v3-m2h3p2v3
937                     use_count(array, unuse);
938 
939                     timer1.Interval=interval_val;
940                     timer1.Enabled=true;
941                 }
942 
943                 if( e.KeyCode == Keys.P  )
944                 {
945                     array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                            v8
946                     array2=new int[]{1, 2, 3, 4, 5, 6, 7,     16, 24,     31,
                            30, 29, 28, 27, 26, 25, mute, -1}; // h7v2-h7
947 
948                     array=new int[100];
949                     rest=new int[mute_dur];
950                     for (int i=0; i<mute_dur; i++)
951                     {
952                          rest[i]=mute;
953                     }
954 
955                     array = array1.Concat(rest).ToArray();
956                     array = array.Concat(array2).ToArray();
957                     use_count(array, unuse);
958 
959                     timer1.Interval=interval_val;
960                     timer1.Enabled=true;
961                 }
962 
963 // Q:中の直線が長すぎるかも。
964                 if( e.KeyCode == Keys.Q  )
965                 {
966                     array1=new int[]{mute, 24, 15, 6,     5, 4, 3,     10, 17,
                            25,     33, 41,     50, 59,     60, 61, 62,     55, 48,
                                 40, 32, 24}; // m3-h3-p3-v2-m2-m3p2v3
967                     array2=new int[]{35, 36, 37,      46, 55, _sp64, mute, -1};
                             // h3-m3
968 
969                     array=new int[100];
970                     rest=new int[mute_dur];
971                     for (int i=0; i<mute_dur; i++)
972                     {
973                          rest[i]=mute;
974                     }
975 
976                     array = array1.Concat(rest).ToArray();
977                     array = array.Concat(array2).ToArray();
978                     use_count(array, unuse);
979 
980                     timer1.Interval=interval_val;
```

```
981                     timer1.Enabled=true;
982                 }
983
984                 if( e.KeyCode == Keys.R  )
985                 {
986                     array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                        v8
987                     array2=new int[]{2, 3, 4, 5, 6, 7,     16, 24,     31, 30,
                        29, 28, 27, 26, 25}; // h6v2-h7
988                     array3=new int[]{28, 37, 46, 55, _sp64, mute, -1}; // -m5
989
990                     array=new int[100];
991                     rest=new int[mute_dur];
992                     for (int i=0; i<mute_dur; i++)
993                     {
994                          rest[i]=mute;
995                     }
996
997                     array = array1.Concat(rest).ToArray();
998                     array = array.Concat(array2).ToArray();
999                     array = array.Concat(rest).ToArray();
1000                    array = array.Concat(array3).ToArray();
1001                    use_count(array, unuse);
1002
1003                    timer1.Interval=interval_val;
1004                    timer1.Enabled=true;
1005                }
1006
1007                if( e.KeyCode == Keys.S  )
1008                {
1009                    //array=new int[]{0, 14, 5,     4, 11,     19, 28,     29,
                        30,     39, 48,     55, 62,     61, 60, 59,     50, 41,
                             33, 0, -1}; // m2-p2-m2h2-m2-p2-h3m2v1 // 鈴木
1010                    array=new int[]{mute, 23, 14, 5, 12, 19, 28, 29, 30, 39,
                        48, 55, 61, 60, 51, 42, 33, mute, -1}; // 高木
1011                    use_count(array, unuse);
1012
1013                    timer1.Interval=interval_val;
1014                    timer1.Enabled=true;
1015                }
1016
1017                if( e.KeyCode == Keys.T  )
1018                {
1019                    array1=new int[]{mute, 1, 2, 3, 4, 5, 6, 7}; // h7
1020                    array2=new int[]{4, 12, 20, 28, 36, 44, 52, 60, mute, -1};
                        // -v8
1021
1022                    array=new int[100];
1023                    rest=new int[mute_dur];
1024                    for (int i=0; i<mute_dur; i++)
1025                    {
1026                         rest[i]=mute;
1027                    }
1028
1029                    array = array1.Concat(rest).ToArray();
1030                    array = array.Concat(array2).ToArray();
1031                    use_count(array, unuse);
1032
1033                    timer1.Interval=interval_val;
1034                    timer1.Enabled=true;
1035                }
1036
1037                if( e.KeyCode == Keys.U  )
1038                {
1039                    array=new int[]{mute, 1, 9, 17, 25, 33,     42, 51, 60,
                            61, 54, 47, 40,     32, 24, 16, 8, mute, -1}; // -
                        v5-m3p4v4
1040                    use_count(array, unuse);
1041
1042                    timer1.Interval=interval_val;
1043                    timer1.Enabled=true;
```

```
1044                }
1045
1046                if( e.KeyCode == Keys.V  )
1047                {
1048                      array1=new int[]{mute, 1, 10, 19, 28, 37, 46, 55, _sp64};
                          // -m8
1049                      array2=new int[]{56, 48, 40, 32, 24, 16, 8, mute, -1}; //
                          v7
1050
1051                      array=new int[100];
1052                      rest=new int[mute_dur];
1053                      for (int i=0; i<mute_dur; i++)
1054                      {
1055                           rest[i]=mute;
1056                      }
1057
1058                      array = array1.Concat(rest).ToArray();
1059                      array = array.Concat(array2).ToArray();
1060                      use_count(array, unuse);
1061
1062                      timer1.Interval=interval_val;
1063                      timer1.Enabled=true;
1064                }
1065
1066 // M:斜辺部が1段下がっている。確認すること。
1067                if( e.KeyCode == Keys.W  )
1068                {
1069                      array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57}; // -
                          v8
1070                      array2=new int[]{50, 43, 36, 20}; // p4
1071                      array3=new int[]{38, 47, 56}; // -m3
1072                      array4=new int[]{_sp64, 56, 48, 40, 32, 24, 16, 8, mute,
                          -1}; // v8
1073
1074                      array=new int[100];
1075                      rest=new int[mute_dur];
1076                      for (int i=0; i<mute_dur; i++)
1077                      {
1078                           rest[i]=mute;
1079                      }
1080
1081                      array = array1.Concat(rest).ToArray();
1082                      array = array.Concat(array2).ToArray();
1083                      array = array.Concat(rest).ToArray();
1084                      array = array.Concat(array3).ToArray();
1085                      array = array.Concat(rest).ToArray();
1086                      array = array.Concat(array4).ToArray();
1087                      use_count(array, unuse);
1088
1089                      timer1.Interval=interval_val;
1090                      timer1.Enabled=true;
1091                }
1092
1093                if( e.KeyCode == Keys.X  )
1094                {
1095                      array1=new int[]{mute, 1, 10, 19, 28, 37, 46, 55, _sp64};
                          // -m8
1096                      array2=new int[]{8, 15, 22, 29, 36, 43, 50, 57, mute, -1};
                          // -p8
1097
1098                      array=new int[100];
1099                      rest=new int[mute_dur];
1100                      for (int i=0; i<mute_dur; i++)
1101                      {
1102                           rest[i]=mute;
1103                      }
1104
1105                      array = array1.Concat(rest).ToArray();
1106                      array = array.Concat(array2).ToArray();
1107                      use_count(array, unuse);
1108
```

```
1109                    timer1.Interval=interval_val;
1110                    timer1.Enabled=true;
1111                }
1112
1113                if( e.KeyCode == Keys.Y  )
1114                {
1115                    array1=new int[]{mute, 1, 10, 19, 28}; // -m4
1116                    array2=new int[]{8, 15, 22, 29}; // -p4
1117                    array3=new int[]{37, 45, 53, 61, mute, -1}; // -v4
1118
1119                    array=new int[100];
1120                    rest=new int[mute_dur];
1121                    for (int i=0; i<mute_dur; i++)
1122                    {
1123                        rest[i]=mute;
1124                    }
1125
1126                    array = array1.Concat(rest).ToArray();
1127                    array = array.Concat(array2).ToArray();
1128                    array = array.Concat(rest).ToArray();
1129                    array = array.Concat(array3).ToArray();
1130                    use_count(array, unuse);
1131
1132                    timer1.Interval=interval_val;
1133                    timer1.Enabled=true;
1134                }
1135
1136                if( e.KeyCode == Keys.Z  )
1137                {
1138                    array1=new int[]{mute, 1, 2, 3, 4, 5, 6, 7, 8}; // h8
1139                    array2=new int[]{15, 22, 29, 36, 43, 50}; // -p6
1140                    array3=new int[]{57, 58, 59, 60, 61, 62, 63, _sp64, mute,
                            -1}; // h8
1141
1142                    array=new int[100];
1143                    rest=new int[mute_dur];
1144                    for (int i=0; i<mute_dur; i++)
1145                    {
1146                        rest[i]=mute;
1147                    }
1148
1149                    array = array1.Concat(rest).ToArray();
1150                    array = array.Concat(array2).ToArray();
1151                    array = array.Concat(rest).ToArray();
1152                    array = array.Concat(array3).ToArray();
1153                    use_count(array, unuse);
1154
1155                    timer1.Interval=interval_val;
1156                    timer1.Enabled=true;
1157                }
1158
1159                // 最外角の四角形
1160                if( e.KeyCode == Keys.D1  )
1161                {
1162                    array1=new int[]{mute, 1, 2, 3, 4, 5, 6, 7, 8};
1163                    array2=new int[]{8, 16, 24, 32, 40, 48, 56, _sp64};
1164                    array3=new int[]{_sp64, 63, 62, 61, 60, 59, 58, 57};
1165                    array4=new int[]{57, 49, 41, 33, 25, 17, 9, 1, mute, -1};
1166
1167                    array=new int[100];
1168                    rest=new int[mute_dur];
1169                    for (int i=0; i<mute_dur; i++)
1170                    {
1171                        rest[i]=mute;
1172                    }
1173
1174                    array = array1.Concat(rest).ToArray();
1175                    array = array.Concat(array2).ToArray();
1176                    array = array.Concat(rest).ToArray();
1177                    array = array.Concat(array3).ToArray();
1178                    array = array.Concat(rest).ToArray();
```

```
1179                    array = array.Concat(array4).ToArray();
1180
1181                    timer1.Interval=interval_val;
1182                    timer1.Enabled=true;
1183              }
1184
1185              // 横線
1186              if( e.KeyCode == Keys.D2  )
1187              {
1188                    array1=new int[]{mute, 1, 2, 3, 4, 5, 6, 7, 8};
1189                    array2=new int[]{9, 10, 11, 12, 13, 14, 15, 16};
1190                    array3=new int[]{17, 18, 19, 20, 21, 22, 23, 24};
1191                    array4=new int[]{25, 26, 27, 28, 29, 30, 31, 32};
1192                    array5=new int[]{33, 34, 35, 36, 37, 38, 39, 40};
1193                    array6=new int[]{41, 42, 43, 44, 45, 46, 47, 48};
1194                    array7=new int[]{49, 50, 51, 52, 53, 54, 55, 56};
1195                    array8=new int[]{57, 58, 59, 60, 61, 62, 63, mute, -1};
1196
1197                    array=new int[100];
1198                    rest=new int[mute_dur];
1199                    for (int i=0; i<mute_dur; i++)
1200                    {
1201                         rest[i]=mute;
1202                    }
1203
1204                    array = array1.Concat(rest).ToArray();
1205                    array = array.Concat(array2).ToArray();
1206                    array = array.Concat(rest).ToArray();
1207                    array = array.Concat(array3).ToArray();
1208                    array = array.Concat(rest).ToArray();
1209                    array = array.Concat(array4).ToArray();
1210                    array = array.Concat(rest).ToArray();
1211                    array = array.Concat(array5).ToArray();
1212                    array = array.Concat(rest).ToArray();
1213                    array = array.Concat(array6).ToArray();
1214                    array = array.Concat(rest).ToArray();
1215                    array = array.Concat(array7).ToArray();
1216                    array = array.Concat(rest).ToArray();
1217                    array = array.Concat(array8).ToArray();
1218
1219                    timer1.Interval=interval_val;
1220                    timer1.Enabled=true;
1221              }
1222
1223              // 縦線
1224              if( e.KeyCode == Keys.D3  )
1225              {
1226                    array1=new int[]{mute, 1, 9, 17, 25, 33, 41, 49, 57};
1227                    array2=new int[]{2, 10, 18, 26, 34, 42, 50, 58};
1228                    array3=new int[]{3, 11, 19, 27, 35, 43, 51, 59};
1229                    array4=new int[]{4, 12, 20, 28, 36, 44, 52, 60};
1230                    array5=new int[]{5, 13, 21, 29, 37, 45, 53, 61};
1231                    array6=new int[]{6, 14, 22, 30, 38, 46, 54, 62};
1232                    array7=new int[]{7, 15, 23, 31, 39, 47, 55, 63};
1233                    array8=new int[]{8, 16, 24, 32, 40, 48, 56, mute, -1};
1234
1235                    array=new int[100];
1236                    rest=new int[mute_dur];
1237                    for (int i=0; i<mute_dur; i++)
1238                    {
1239                         rest[i]=mute;
1240                    }
1241
1242                    array = array1.Concat(rest).ToArray();
1243                    array = array.Concat(array2).ToArray();
1244                    array = array.Concat(rest).ToArray();
1245                    array = array.Concat(array3).ToArray();
1246                    array = array.Concat(rest).ToArray();
1247                    array = array.Concat(array4).ToArray();
1248                    array = array.Concat(rest).ToArray();
1249                    array = array.Concat(array5).ToArray();
```

```
1250                    array = array.Concat(rest).ToArray();
1251                    array = array.Concat(array6).ToArray();
1252                    array = array.Concat(rest).ToArray();
1253                    array = array.Concat(array7).ToArray();
1254                    array = array.Concat(rest).ToArray();
1255                    array = array.Concat(array8).ToArray();
1256
1257                    timer1.Interval=interval_val;
1258                    timer1.Enabled=true;
1259            }
1260
1261            if( e.KeyCode == Keys.D0  )
1262            {
1263                 if (pen==0)
1264                 {
1265                      timer2.Interval=30;
1266                      timer2.Enabled=true;
1267                      syn.Speak("pen on");
1268                      pen=1;
1269                 } else {
1270                      timer2.Enabled=false;
1271                      sw_on((byte)0);
1272                      pen=0;
1273                      syn.Speak("pen off");
1274                 }
1275            }
1276
1277 // ----- interval変数の変更
1278            if( e.KeyCode == Keys.Left  )
1279            {
1280                 if (interval_val>50)
1281                 {
1282                      interval_val-=50;
1283                      syn.Speak(interval_val.ToString());
1284                 }
1285            }
1286
1287            if( e.KeyCode == Keys.Right  )
1288            {
1289                 if (interval_val<500)
1290                 {
1291                      interval_val+=50;
1292                      syn.Speak(interval_val.ToString());
1293                 }
1294            }
1295
1296            if( e.KeyCode == Keys.Down  )
1297            {
1298                 if (mute_dur>0)
1299                                {
1300                      mute_dur--;
1301                      syn.Speak((interval_val*mute_dur).ToString());
1302                 }
1303            }
1304
1305            if( e.KeyCode == Keys.Up  )
1306            {
1307                 if (mute_dur<10)
1308                 {
1309                      mute_dur++;
1310                      syn.Speak((interval_val*mute_dur).ToString());
1311                 }
1312            }
1313
1314            //if( e.KeyCode == Keys.Space  )
1315            //{
1316                 //syn.Speak("1Sp="+interval_val.ToString()+"ms, "+"Mute="+(
                         interval_val*mute_dur).ToString());
1317            //}
1318
1319       }
```

```
1320
1321          int tm=0;
1322          private void timer1_Tick(object sender, EventArgs e)
1323          {
1324                 if (array[tm]!=-1)
1325               {
1326                    this.Text=array[tm].ToString();
1327                    sw_on((byte)(array[tm]-1));
1328               } else {
1329                    timer1.Enabled=false;
1330                    tm=0;
1331                    syn.Speak("end");
1332               }
1333
1334               tm++;
1335          }
1336
1337          private void timer2_Tick(object sender, EventArgs e)
1338          {
1339              int x=Cursor.Position.X;
1340              int y=Cursor.Position.Y;
1341              if (y>0 && y<=100)
1342              {
1343                   if (x>0 && x<=170)
1344                   {
1345                        sw_on((byte)0);
1346                   } else if (x>170 && x<=340)
1347                   {
1348                        sw_on((byte)1);
1349                   } else if (x>340 && x<=510)
1350                   {
1351                        sw_on((byte)2);
1352                   } else if (x>510 && x<=680)
1353                   {
1354                        sw_on((byte)3);
1355                   } else if (x>680 && x<=850)
1356                   {
1357                        sw_on((byte)4);
1358                   } else if (x>850 && x<=1020)
1359                   {
1360                        sw_on((byte)5);
1361                   } else if (x>1020 && x<=1190)
1362                   {
1363                        sw_on((byte)6);
1364                   } else if (x>1190 && x<=1378)
1365                   {
1366                        sw_on((byte)7);
1367                   } else {
1368                        sw_on((byte)100);
1369                   }
1370              } else if (y>100 && y<=200)
1371              {
1372                   if (x>0 && x<=170)
1373                   {
1374                        sw_on((byte)8);
1375                   } else if (x>170 && x<=340)
1376                   {
1377                        sw_on((byte)9);
1378                   } else if (x>340 && x<=510)
1379                   {
1380                        sw_on((byte)10);
1381                   } else if (x>510 && x<=680)
1382                   {
1383                        sw_on((byte)11);
1384                   } else if (x>680 && x<=850)
1385                   {
1386                        sw_on((byte)12);
1387                   } else if (x>850 && x<=1020)
1388                   {
1389                        sw_on((byte)13);
1390                   } else if (x>1020 && x<=1190)
```

```
1391                    {
1392                         sw_on((byte)14);
1393                    } else if (x>1190 && x<=1378)
1394                    {
1395                         sw_on((byte)15);
1396                    } else {
1397                         sw_on((byte)100);
1398                    }
1399
1400               } else if (y>200 && y<=300)
1401               {
1402                    if (x>0 && x<=170)
1403                    {
1404                         sw_on((byte)16);
1405                    } else if (x>170 && x<=340)
1406                    {
1407                         sw_on((byte)17);
1408                    } else if (x>340 && x<=510)
1409                    {
1410                         sw_on((byte)18);
1411                    } else if (x>510 && x<=680)
1412                    {
1413                         sw_on((byte)19);
1414                    } else if (x>680 && x<=850)
1415                    {
1416                         sw_on((byte)20);
1417                    } else if (x>850 && x<=1020)
1418                    {
1419                         sw_on((byte)21);
1420                    } else if (x>1020 && x<=1190)
1421                    {
1422                         sw_on((byte)22);
1423                    } else if (x>1190 && x<=1378)
1424                    {
1425                         sw_on((byte)23);
1426                    } else {
1427                         sw_on((byte)100);
1428                    }
1429
1430               } else if (y>300 && y<=400)
1431               {
1432                    if (x>0 && x<=170)
1433                    {
1434                         sw_on((byte)24);
1435                    } else if (x>170 && x<=340)
1436                    {
1437                         sw_on((byte)25);
1438                    } else if (x>340 && x<=510)
1439                    {
1440                         sw_on((byte)26);
1441                    } else if (x>510 && x<=680)
1442                    {
1443                         sw_on((byte)27);
1444                    } else if (x>680 && x<=850)
1445                    {
1446                         sw_on((byte)28);
1447                    } else if (x>850 && x<=1020)
1448                    {
1449                         sw_on((byte)29);
1450                    } else if (x>1020 && x<=1190)
1451                    {
1452                         sw_on((byte)30);
1453                    } else if (x>1190 && x<=1378)
1454                    {
1455                         sw_on((byte)31);
1456                    } else {
1457                         sw_on((byte)100);
1458                    }
1459
1460               } else if (y>400 && y<=500)
1461               {
```

```
1462                    if (x>0 && x<=170)
1463                    {
1464                         sw_on((byte)32);
1465                    } else if (x>170 && x<=340)
1466                    {
1467                         sw_on((byte)33);
1468                    } else if (x>340 && x<=510)
1469                    {
1470                         sw_on((byte)34);
1471                    } else if (x>510 && x<=680)
1472                    {
1473                         sw_on((byte)35);
1474                    } else if (x>680 && x<=850)
1475                    {
1476                         sw_on((byte)36);
1477                    } else if (x>850 && x<=1020)
1478                    {
1479                         sw_on((byte)37);
1480                    } else if (x>1020 && x<=1190)
1481                    {
1482                         sw_on((byte)38);
1483                    } else if (x>1190 && x<=1378)
1484                    {
1485                         sw_on((byte)39);
1486                    } else {
1487                         sw_on((byte)100);
1488                    }
1489
1490               } else if (y>500 && y<=600)
1491               {
1492                    if (x>0 && x<=170)
1493                    {
1494                         sw_on((byte)40);
1495                    } else if (x>170 && x<=340)
1496                    {
1497                         sw_on((byte)41);
1498                    } else if (x>340 && x<=510)
1499                    {
1500                         sw_on((byte)42);
1501                    } else if (x>510 && x<=680)
1502                    {
1503                         sw_on((byte)43);
1504                    } else if (x>680 && x<=850)
1505                    {
1506                         sw_on((byte)44);
1507                    } else if (x>850 && x<=1020)
1508                    {
1509                         sw_on((byte)45);
1510                    } else if (x>1020 && x<=1190)
1511                    {
1512                         sw_on((byte)46);
1513                    } else if (x>1190 && x<=1378)
1514                    {
1515                         sw_on((byte)47);
1516                    } else {
1517                         sw_on((byte)100);
1518                    }
1519
1520               } else if (y>600 && y<=700)
1521               {
1522                    if (x>0 && x<=170)
1523                    {
1524                         sw_on((byte)48);
1525                    } else if (x>170 && x<=340)
1526                    {
1527                         sw_on((byte)49);
1528                    } else if (x>340 && x<=510)
1529                    {
1530                         sw_on((byte)50);
1531                    } else if (x>510 && x<=680)
1532                    {
```

```
                    sw_on((byte)51);
                } else if (x>680 && x<=850)
                {
                    sw_on((byte)52);
                } else if (x>850 && x<=1020)
                {
                    sw_on((byte)53);
                } else if (x>1020 && x<=1190)
                {
                    sw_on((byte)54);
                } else if (x>1190 && x<=1378)
                {
                    sw_on((byte)55);
                } else {
                    sw_on((byte)100);
                }

            } else if (y>700 && y<=779)
            {
                if (x>0 && x<=170)
                {
                    sw_on((byte)56);
                } else if (x>170 && x<=340)
                {
                    sw_on((byte)57);
                } else if (x>340 && x<=510)
                {
                    sw_on((byte)58);
                } else if (x>510 && x<=680)
                {
                    sw_on((byte)59);
                } else if (x>680 && x<=850)
                {
                    sw_on((byte)60);
                } else if (x>850 && x<=1020)
                {
                    sw_on((byte)61);
                } else if (x>1020 && x<=1190)
                {
                    sw_on((byte)62);
                } else if (x>1190 && x<=1378)
                {
                    sw_on((byte)63);
                } else {
                    sw_on((byte)100);
                }

            }
        }

        private void timer3_Tick(object sender, EventArgs e)
        {

        }

        private void timer4_Tick(object sender, EventArgs e)
        {

        }

        private void timer5_Tick(object sender, EventArgs e)
        {

        }

        private void timer6_Tick(object sender, EventArgs e)
        {

        }

        private void timer7_Tick(object sender, EventArgs e)
```

```
1604            {
1605
1606            }
1607
1608            private void timer8_Tick(object sender, EventArgs e)
1609            {
1610
1611            }
1612
1613            private void timer9_Tick(object sender, EventArgs e)
1614            {
1615
1616            }
1617
1618            private void timer10_Tick(object sender, EventArgs e)
1619            {
1620
1621            }
1622
1623            private void timer11_Tick(object sender, EventArgs e)
1624            {
1625
1626            }
1627
1628            private void timer12_Tick(object sender, EventArgs e)
1629            {
1630
1631            }
1632
1633            private void timer13_Tick(object sender, EventArgs e)
1634            {
1635
1636            }
1637
1638            private void timer14_Tick(object sender, EventArgs e)
1639            {
1640
1641            }
1642
1643            private void timer15_Tick(object sender, EventArgs e)
1644            {
1645
1646            }
1647
1648            private void timer16_Tick(object sender, EventArgs e)
1649            {
1650
1651            }
1652
1653            private void timer17_Tick(object sender, EventArgs e)
1654            {
1655
1656            }
1657
1658            private void timer18_Tick(object sender, EventArgs e)
1659            {
1660
1661            }
1662
1663            private void timer19_Tick(object sender, EventArgs e)
1664            {
1665
1666            }
1667
1668            private void timer20_Tick(object sender, EventArgs e)
1669            {
1670
1671            }
1672
1673            private void timer21_Tick(object sender, EventArgs e)
1674            {
```

```
1675
1676            }
1677
1678            private void timer22_Tick(object sender, EventArgs e)
1679            {
1680
1681            }
1682
1683            private void timer23_Tick(object sender, EventArgs e)
1684            {
1685
1686            }
1687
1688            private void timer24_Tick(object sender, EventArgs e)
1689            {
1690
1691            }
1692
1693            private void timer25_Tick(object sender, EventArgs e)
1694            {
1695
1696            }
1697
1698            private void timer26_Tick(object sender, EventArgs e)
1699            {
1700
1701            }
1702
1703
1704        }
1705    }
```

付録C　　第4章の実験用プログラム・ソース

リスト 2: sample source

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using Microsoft.DirectX;
using Microsoft.DirectX.DirectSound;
using System.Speech.Synthesis;

namespace GetColorFromScreen
{
    public partial class Form1 : Form
    {
        int space=0;
        int guide=1;
        int temp_l=0;
        int temp_r=0;
        int temp_u=0;
        int temp_d=0;
        int temp_lu=0;
        int temp_ld=0;
        int temp_ru=0;
        int temp_rd=0;
        int thick=45;
        int area=100;
        int pattern=1;
        int wmode=0;

        private Device _device0 = null;
        private Device _device1 = null;
        private Device _device2 = null;
        private Device _device3 = null;
        private Device _device4 = null;

        private SecondaryBuffer _buffer1 = null;
        private SecondaryBuffer _buffer2 = null;

        private SecondaryBuffer _buffer3 = null;
        private SecondaryBuffer _buffer4 = null;

        private SecondaryBuffer _buffer5 = null;
        private SecondaryBuffer _buffer6 = null;

        private SecondaryBuffer _buffer7 = null;
        private SecondaryBuffer _buffer8 = null;

        private SecondaryBuffer _buffer9 = null;
        private SecondaryBuffer _buffer10 = null;

        private SecondaryBuffer _buffer11 = null;
        private SecondaryBuffer _buffer12 = null;

        private SecondaryBuffer _buffer101 = null;
        private SecondaryBuffer _buffer102 = null;
        private SecondaryBuffer _buffer103 = null;
        private SecondaryBuffer _buffer104 = null;


        private Guid[] id = new Guid[20];


```

```
66          Graphics g ;  // Graphics オブジェクト
67          Image bmp;
68          int start = 0 ;  // 1 = 描画中
69          int startX ;     // Line X 起点
70          int startY ;     // Line Y 起点
71
72
73           [DllImport("user32.dll")]
74           static extern IntPtr GetDC(IntPtr hwnd);
75
76           [DllImport("user32.dll")]
77           static extern Int32 ReleaseDC(IntPtr hwnd, IntPtr hdc);
78
79           [DllImport("gdi32.dll")]
80           static extern uint GetPixel(IntPtr hdc, int nXPos, int nYPos);
81
82           static public System.Drawing.Color GetPixelColor(int x, int y)
83           {
84                IntPtr hdc = GetDC(IntPtr.Zero);
85                uint pixel = GetPixel(hdc, x, y);
86                ReleaseDC(IntPtr.Zero, hdc);
87                Color color = Color.FromArgb((int)(pixel & 0x000000FF),
88                    (int)(pixel & 0x0000FF00) >> 8,
89                    (int)(pixel & 0x00FF0000) >> 16);
90                return color;
91           }
92
93           Color c;
94
95           static void say(string s)
96           {
97                SpeechSynthesizer syn = new SpeechSynthesizer();
98                syn.Speak(s);
99           }
100
101
102          public Form1()
103          {
104              InitializeComponent();
105          }
106
107          private void Form1_Load(object sender, EventArgs e)
108          {
109               // Graphics オブジェクトの取得
110               pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1.
                      Height);
111               Image bmp = pictureBox1.Image;
112               Graphics g = Graphics.FromImage(bmp);
113
114              // サウンドデバイスの取得
115              DevicesCollection devices = new DevicesCollection();
116              int i = 0;
117              foreach (DeviceInformation devInfo in devices)
118              {
119                  id[i++] = devInfo.DriverGuid;
120              }
121
122              try
123              {
124                  this._device0 = new Device(id[0]);
125                  this._device1 = new Device(id[2]);
126                  this._device2 = new Device(id[3]);
127                  this._device3 = new Device(id[4]);
128                  this._device4 = new Device(id[5]);
129              }
130              catch (Exception )
131              {
132                  //MessageBox.Show(ex.ToString(), "Error", MessageBoxButtons.OK
                        , MessageBoxIcon.Error);
133              }
134              this._device0.SetCooperativeLevel(this, CooperativeLevel.Normal);
```

```
135             this._device1.SetCooperativeLevel(this, CooperativeLevel.Normal);
136             this._device2.SetCooperativeLevel(this, CooperativeLevel.Normal);
137             this._device3.SetCooperativeLevel(this, CooperativeLevel.Normal);
138             this._device4.SetCooperativeLevel(this, CooperativeLevel.Normal);
139
140             BufferDescription BufferDescription = new BufferDescription();
141             BufferDescription.ControlVolume = true;
142             BufferDescription.ControlPan = true;
143             BufferDescription.ControlFrequency = true;
144             BufferDescription.GlobalFocus = false;
145
146             this._buffer1 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device1);
147             this._buffer2 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device1);
148
149             this._buffer3 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device1);
150             this._buffer4 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device2);
151
152             this._buffer5 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device2);
153             this._buffer6 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device2);
154
155             this._buffer7 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device2);
156             this._buffer8 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device1);
157
158             this._buffer9 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device1);
159             this._buffer10 = new SecondaryBuffer("tones/WhiteNoise.wav", this.
                    _device2);
160
161             this._buffer11 = new SecondaryBuffer("tones/500.wav", this.
                    _device1);
162             this._buffer12 = new SecondaryBuffer("tones/500.wav", this.
                    _device2);
163
164             this._buffer101 = new SecondaryBuffer("tones/WhiteNoise.wav", this
                    ._device3);
165             this._buffer102 = new SecondaryBuffer("tones/WhiteNoise.wav", this
                    ._device3);
166             this._buffer103 = new SecondaryBuffer("tones/WhiteNoise.wav", this
                    ._device4);
167             this._buffer104 = new SecondaryBuffer("tones/WhiteNoise.wav", this
                    ._device4);
168
169              // パンの設定
170              this._buffer1.Pan=-10000;
171              this._buffer2.Pan=10000;
172
173              this._buffer3.Pan=10000;
174              this._buffer4.Pan=10000;
175
176              this._buffer5.Pan=10000;
177              this._buffer6.Pan=-10000;
178
179              this._buffer7.Pan=-10000;
180              this._buffer8.Pan=-10000;
181
182              this._buffer9.Pan=10000;
183              this._buffer10.Pan=-10000;
184
185              this._buffer11.Pan=-10000;
186              this._buffer12.Pan=10000;
187
188              this._buffer101.Pan=-10000;
189              this._buffer102.Pan=10000;
```

```
190
191             this._buffer103.Pan=-10000;
192             this._buffer104.Pan=10000;
193
194
195             this._buffer1.Volume=-10000;
196             this._buffer2.Volume=-10000;
197             this._buffer3.Volume=-10000;
198             this._buffer4.Volume=-10000;
199             this._buffer5.Volume=-10000;
200             this._buffer6.Volume=-10000;
201             this._buffer7.Volume=-10000;
202             this._buffer8.Volume=-10000;
203             this._buffer9.Volume=-10000;
204             this._buffer10.Volume=-10000;
205             this._buffer11.Volume=-10000;
206             this._buffer12.Volume=-10000;
207             this._buffer101.Volume=-10000;
208             this._buffer102.Volume=-10000;
209             this._buffer103.Volume=-10000;
210             this._buffer104.Volume=-10000;
211
212
213             this._buffer1.Play(0, BufferPlayFlags.Looping);
214             this._buffer2.Play(0, BufferPlayFlags.Looping);
215             this._buffer3.Play(0, BufferPlayFlags.Looping);
216             this._buffer4.Play(0, BufferPlayFlags.Looping);
217             this._buffer5.Play(0, BufferPlayFlags.Looping);
218             this._buffer6.Play(0, BufferPlayFlags.Looping);
219             this._buffer7.Play(0, BufferPlayFlags.Looping);
220             this._buffer8.Play(0, BufferPlayFlags.Looping);
221             this._buffer9.Play(0, BufferPlayFlags.Looping);
222             this._buffer10.Play(0, BufferPlayFlags.Looping);
223             this._buffer11.Play(0, BufferPlayFlags.Looping);
224             this._buffer12.Play(0, BufferPlayFlags.Looping);
225             this._buffer101.Play(0, BufferPlayFlags.Looping);
226             this._buffer102.Play(0, BufferPlayFlags.Looping);
227             this._buffer103.Play(0, BufferPlayFlags.Looping);
228             this._buffer104.Play(0, BufferPlayFlags.Looping);
229
230             say("起動しました");
231         }
232
233         int tm1=0;
234         private void timer1_Tick(object sender, EventArgs e)
235         {
236             tm1++;
237
238             if (pattern==1)
239             {
240                 if (tm1==1)
241                 {
242                     _buffer1.Volume=0;
243                 }
244                 if (tm1==2)
245                 {
246                     _buffer1.Volume=-10000;
247                     _buffer2.Volume=0;
248                 }
249                 if (tm1==3)
250                 {
251                     _buffer2.Volume=-10000;
252                 }
253                 if (tm1==4) // ここの値を増やすと、繰り返しの区切り目を広げる
                        ことができる。
254                 {
255                     timer1.Enabled=false;
256                     tm1=0;
257                 }
258             }
259
```

```
260             if (pattern==2)
261             {
262                if (tm1==1)
263                {
264                     _buffer3.Volume=0;
265                }
266                if (tm1==2)
267                {
268                     _buffer3.Volume=-10000;
269                     _buffer4.Volume=0;
270                }
271                if (tm1==3)
272                {
273                     _buffer4.Volume=-10000;
274                }
275                if (tm1==4)
276                {
277                     timer1.Enabled=false;
278                     tm1=0;
279                }
280             }
281
282             if (pattern==3)
283             {
284                if (tm1==1)
285                {
286                     _buffer9.Volume=0;
287                }
288                if (tm1==2)
289                {
290                     _buffer9.Volume=-10000;
291                     _buffer10.Volume=0;
292                }
293                if (tm1==3)
294                {
295                     _buffer10.Volume=-10000;
296                }
297                if (tm1==4)
298                {
299                     timer1.Enabled=false;
300                     tm1=0;
301                }
302             }
303
304             if (pattern==4)
305             {
306                if (tm1==1)
307                {
308                     _buffer11.Volume=0;
309                }
310                if (tm1==2)
311                {
312                     _buffer11.Volume=-10000;
313                     _buffer12.Volume=0;
314                }
315                if (tm1==3)
316                {
317                     _buffer12.Volume=-10000;
318                }
319                if (tm1==4)
320                {
321                     timer1.Enabled=false;
322                     tm1=0;
323                }
324             }
325
326             if (pattern==5)
327             {
328                if (tm1==1)
329                {
330                   _buffer5.Volume=0;
```

```
331                 }
332                 if (tm1==2)
333                 {
334                     _buffer5.Volume=-10000;
335                     _buffer6.Volume=0;
336                 }
337                 if (tm1==3)
338                 {
339                     _buffer6.Volume=-10000;
340                     _buffer7.Volume=0;
341                 }
342                 if (tm1==4)
343                 {
344                     _buffer7.Volume=-10000;
345                     _buffer8.Volume=0;
346                 }
347                 if (tm1==5)
348                 {
349                     _buffer8.Volume=-10000;
350                 }
351                 if (tm1==7)
352                 {
353                     timer1.Enabled=false;
354                     tm1=0;
355                 }
356             }
357 
358              if (pattern==6)
359              {
360                 if (tm1==1)
361                 {
362                     _buffer7.Volume=0;
363                 }
364                 if (tm1==2)
365                 {
366                     _buffer7.Volume=-10000;
367                     _buffer8.Volume=0;
368                 }
369                 if (tm1==3)
370                 {
371                     _buffer8.Volume=-10000;
372                     _buffer1.Volume=0;
373                 }
374                 if (tm1==4)
375                 {
376                     _buffer1.Volume=-10000;
377                     _buffer2.Volume=0;
378                 }
379                 if (tm1==5)
380                 {
381                     _buffer2.Volume=-10000;
382                 }
383                 if (tm1==7)
384                 {
385                     timer1.Enabled=false;
386                     tm1=0;
387                 }
388             }
389 
390              if (pattern==7)
391              {
392                 if (tm1==1)
393                 {
394                     _buffer1.Volume=0;
395                 }
396                 if (tm1==2)
397                 {
398                     _buffer1.Volume=-10000;
399                     _buffer2.Volume=0;
400                 }
401                 if (tm1==3)
```

```
                {
                    _buffer2.Volume=-10000;
                    _buffer3.Volume=0;
                }
                if (tm1==4)
                {
                    _buffer3.Volume=-10000;
                    _buffer4.Volume=0;
                }
                if (tm1==5)
                {
                    _buffer4.Volume=-10000;
                }
                if (tm1==7)
                {
                    timer1.Enabled=false;
                    tm1=0;
                }
            }

             if (pattern==8)
             {
                if (tm1==1)
                {
                    _buffer3.Volume=0;
                }
                if (tm1==2)
                {
                    _buffer3.Volume=-10000;
                    _buffer4.Volume=0;
                }
                if (tm1==3)
                {
                    _buffer4.Volume=-10000;
                    _buffer5.Volume=0;
                }
                if (tm1==4)
                {
                    _buffer5.Volume=-10000;
                    _buffer6.Volume=0;
                }
                if (tm1==5)
                {
                    _buffer6.Volume=-10000;
                }
                if (tm1==7)
                {
                    timer1.Enabled=false;
                    tm1=0;
                }
            }

             if (pattern==9)
             {
                if (tm1==1)
                {
                    _buffer7.Volume=0;
                }
                if (tm1==2)
                {
                    _buffer7.Volume=-10000;
                    _buffer8.Volume=0;
                }
                if (tm1==3)
                {
                    _buffer8.Volume=-10000;
                    _buffer11.Volume=0;
                }
                if (tm1==4)
                {
                    _buffer11.Volume=-10000;
```

```
473                     _buffer12.Volume=0;
474                 }
475                 if (tm1==5)
476                 {
477                     _buffer12.Volume=-10000;
478                 }
479                 if (tm1==7)
480                 {
481                     timer1.Enabled=false;
482                     tm1=0;
483                 }
484             }
485 
486              if (pattern==10)
487              {
488                 if (tm1==1)
489                 {
490                     _buffer4.Volume=0;
491                 }
492                 if (tm1==2)
493                 {
494                     _buffer4.Volume=-10000;
495                     _buffer3.Volume=0;
496                 }
497                 if (tm1==3)
498                 {
499                     _buffer3.Volume=-10000;
500                     _buffer9.Volume=0;
501                 }
502                 if (tm1==4)
503                 {
504                     _buffer9.Volume=-10000;
505                     _buffer10.Volume=0;
506                 }
507                 if (tm1==5)
508                 {
509                     _buffer10.Volume=-10000;
510                 }
511                 if (tm1==7)
512                 {
513                     timer1.Enabled=false;
514                     tm1=0;
515                 }
516             }
517 
518              if (pattern==11)
519              {
520                 if (tm1==1)
521                 {
522                     _buffer11.Volume=0;
523                 }
524                 if (tm1==2)
525                 {
526                     _buffer11.Volume=-10000;
527                     _buffer12.Volume=0;
528                 }
529                 if (tm1==3)
530                 {
531                     _buffer12.Volume=-10000;
532                     _buffer5.Volume=0;
533                 }
534                 if (tm1==4)
535                 {
536                     _buffer5.Volume=-10000;
537                     _buffer6.Volume=0;
538                 }
539                 if (tm1==5)
540                 {
541                     _buffer6.Volume=-10000;
542                 }
543                 if (tm1==7)
```

```
544                 {
545                     timer1.Enabled=false;
546                     tm1=0;
547                 }
548             }
549 
550              if (pattern==12)
551              {
552                 if (tm1==1)
553                 {
554                     _buffer9.Volume=0;
555                 }
556                 if (tm1==2)
557                 {
558                     _buffer9.Volume=-10000;
559                     _buffer10.Volume=0;
560                 }
561                 if (tm1==3)
562                 {
563                     _buffer10.Volume=-10000;
564                     _buffer6.Volume=0;
565                 }
566                 if (tm1==4)
567                 {
568                     _buffer6.Volume=-10000;
569                     _buffer5.Volume=0;
570                 }
571                 if (tm1==5)
572                 {
573                     _buffer5.Volume=-10000;
574                 }
575                 if (tm1==7)
576                 {
577                     timer1.Enabled=false;
578                     tm1=0;
579                 }
580             }
581 
582         }
583 
584          int near_l;
585          int near_r;
586          int near_u;
587          int near_d;
588 
589          private void timer2_Tick(object sender, EventArgs e)
590          {
591              int x=System.Windows.Forms.Cursor.Position.X;
592              int y=System.Windows.Forms.Cursor.Position.Y;
593 
594              //this.Text=x.ToString()+","+y.ToString()+" "+GetPixelColor(x, y
                     ).Name.ToString();
595 
596 
597 
598              // 右に何か点があるか否かの判定
599              if ( GetPixelColor(x+area, y).Name.ToString()=="ffffffff"
600                   && GetPixelColor(x, y).Name.ToString()=="ff000000"
601                   && (x<1023-area) )
602              {
603                   if(near_r==0)
604                   {
605                        temp_r=x; // 初めて見つけた位置を保持
606                        near_r=1;
607                   }
608              }
609 
610              if ( (x>temp_r && x<temp_r+area) )
611              {
612                  this.Text=x.ToString()+","+y.ToString()+".....→....";
613                  if (guide==1)
```

```
614                     {
615                          _buffer102.Volume=(x-(temp_r+area))*45;
616                     }
617                     near_r=1;
618                 }
619
620                 // そのエリアを外れたとき
621                 if (x<temp_r || x>temp_r+area)
622                 {
623                       _buffer102.Volume=-10000;
624                       this.Text=x.ToString()+","+y.ToString();
625                       near_r=0;
626                 }
627
628
629                 // 左に何か点があるか否かの判定
630                 if ( GetPixelColor(x-area, y).Name.ToString()=="ffffffff"
631                      && GetPixelColor(x, y).Name.ToString()=="ff000000"
632                      && (x<1023-area) )
633                 {
634                      if (near_l==0)
635                      {
636                           temp_l=x; // 初めて見つけた位置を保持
637                           near_l=1;
638                      }
639                 }
640
641                 if ( (x>temp_l-area && x<temp_l) )
642                 {
643                     this.Text=x.ToString()+","+y.ToString()+".....←....";
644                     if (guide==1)
645                     {
646                          _buffer101.Volume=((temp_l-area)-x)*45;
647                     }
648                     near_l=1;
649                 }
650
651                 // そのエリアを外れたとき
652                 if (x>temp_l || x<temp_l-area)
653                 {
654                     _buffer101.Volume=-10000;
655                     this.Text=x.ToString()+","+y.ToString();
656                     near_l=0;
657                 }
658
659
660                 // 上に何か点があるか否かの判定
661                 if ( GetPixelColor(x, y-area).Name.ToString()=="ffffffff"
662                      && GetPixelColor(x, y).Name.ToString()=="ff000000"
663                      && (y>area) )
664                 {
665                      if (near_u==0)
666                      {
667                           temp_u=y; // 初めて見つけた位置を保持
668                           near_u=1;
669                      }
670                 }
671
672                 if ( near_u==1 && (y>temp_u-area && y<temp_u) )
673                 {
674                     this.Text=x.ToString()+","+y.ToString()+".....↑....";
675                     _buffer103.Volume=((temp_u-area)-y)*45;
676                     near_u=1;
677                 }
678
679                 // そのエリアを外れたとき
680                 if (y>temp_u || y<temp_u-area)
681                 {
682                      _buffer103.Volume=-10000;
683                      this.Text=x.ToString()+","+y.ToString();
684                      near_u=0;
```

```
685                 }
686
687
688                 // 下に何か点があるか否かの判定
689                 if ( GetPixelColor(x, y+area).Name.ToString()=="ffffffff"
690                      && GetPixelColor(x, y).Name.ToString()=="ff000000"
691                      && (y<730-area) )
692                 {
693                      if (near_d==0)
694                      {
695                           temp_d=y; // 初めて見つけた位置を保持
696                           near_d=1;
697                      }
698                 }
699
700                 if ( near_d==1 && (y>temp_d && y<temp_d+area) )
701                 {
702                     this.Text=x.ToString()+","+y.ToString()+".....↓....";
703                     if (guide==1)
704                     {
705                           _buffer104.Volume=(y-(temp_d+area))*45;
706                     }
707                     near_d=1;
708                 }
709
710                 // そのエリアを外れたとき
711                 if (y<temp_d || y>temp_d+area)
712                 {
713                      _buffer104.Volume=-10000;
714                      near_d=0;
715                      this.Text=x.ToString()+","+y.ToString();
716                      near_d=0;
717                 }
718
719
720
721                 // 黒ならば
722                 if (GetPixelColor(x, y).Name.ToString()=="ff000000")
723                 {
724                     this.Text=x.ToString()+","+y.ToString();
725                     if (tm1==0)
726                     {
727                         _buffer1.Volume=-10000;
728                         _buffer2.Volume=-10000;
729                         _buffer3.Volume=-10000;
730                         _buffer4.Volume=-10000;
731                         _buffer5.Volume=-10000;
732                         _buffer6.Volume=-10000;
733                         _buffer7.Volume=-10000;
734                         _buffer8.Volume=-10000;
735                         _buffer9.Volume=-10000;
736                         _buffer10.Volume=-10000;
737                         _buffer11.Volume=-10000;
738                         _buffer12.Volume=-10000;
739                     }
740
741                 }
742
743
744
745
746                 // 横線の判定
747                 /*if (
748                      (GetPixelColor(x, y).Name.ToString()=="ffffffff"
749                      && GetPixelColor(x-thick, y).Name.ToString()=="ffffffff"
750                      && GetPixelColor(x-2*thick, y).Name.ToString()=="ffffffff")
751                      || (GetPixelColor(x, y).Name.ToString()=="ffffffff"
752                      && GetPixelColor(x+thick, y).Name.ToString()=="ffffffff"
753                      && GetPixelColor(x+2*thick, y).Name.ToString()=="ffffffff")
754                      && (x>200 && x<1000)
755                      && (y>100 && y<700)  )*/
```

```
756                 if (GetPixelColor(x, y).Name.ToString()=="ffffffff")
757                 {
758                     this.Text=x.ToString()+","+y.ToString()+"----";
759 
760                     if (tm1==0)
761                     {
762                           pattern=1;
763                           timer1.Enabled=true;
764                     }
765                 }
766 
767                 // 縦線の判定
768                 /*if (
769                      (GetPixelColor(x, y).Name.ToString()=="ffffffff"
770                      && GetPixelColor(x, y+thick).Name.ToString()=="ffffffff"
771                      && GetPixelColor(x, y+2*thick).Name.ToString()=="ffffffff")
772                      || (GetPixelColor(x, y).Name.ToString()=="ffffffff"
773                      && GetPixelColor(x, y-thick).Name.ToString()=="ffffffff"
774                      && GetPixelColor(x, y-2*thick).Name.ToString()=="ffffffff")
775                      && (x>200 && x<1000)
776                      && (y>100 && y<700)  )*/
777                 if (GetPixelColor(x, y).Name.ToString()=="fffffffe")
778                 {
779                      this.Text=x.ToString()+","+y.ToString()+"||||";
780 
781                     if (tm1==0)
782                     {
783                          pattern=2;
784                          timer1.Enabled=true;
785                     }
786                 }
787 
788                 // 斜線（プラス）の判定
789                 /*if (
790                      (GetPixelColor(x, y).Name.ToString()=="ffffffff"
791                      && GetPixelColor(x-thick, y+thick).Name.ToString()==
                             "ffffffff"
792                      && GetPixelColor(x-2*thick, y+2*thick).Name.ToString()==
                             "ffffffff" )
793                      || (GetPixelColor(x, y).Name.ToString()=="ffffffff"
794                      && GetPixelColor(x+thick, y-thick).Name.ToString()==
                             "ffffffff"
795                      && GetPixelColor(x+2*thick, y-2*thick).Name.ToString()==
                             "ffffffff" )
796                      && (x>200 && x<1000)
797                      && (y>100 && y<700)  )*/
798                 if (GetPixelColor(x, y).Name.ToString()=="fffffeff")
799                 {
800                      this.Text=x.ToString()+","+y.ToString()+"////";
801 
802                     if (tm1==0)
803                     {
804                          pattern=3;
805                          timer1.Enabled=true;
806                     }
807                 }
808 
809                 // 斜線（マイナス）の判定
810                 /*if (
811                      (GetPixelColor(x, y).Name.ToString()=="ffffffff"
812                      && GetPixelColor(x-thick, y-thick).Name.ToString()==
                             "ffffffff"
813                      && GetPixelColor(x-2*thick, y-2*thick).Name.ToString()==
                             "ffffffff" )
814                      || (GetPixelColor(x, y).Name.ToString()=="ffffffff"
815                      && GetPixelColor(x+thick, y+thick).Name.ToString()==
                             "ffffffff"
816                      && GetPixelColor(x+2*thick, y+2*thick).Name.ToString()==
                             "ffffffff" )
817                      && (x>200 && x<1000)
818                      && (y>100 && y<700)  )*/
```

```
819                 if (GetPixelColor(x, y).Name.ToString()=="fffeffff")
820                 {
821                      this.Text=x.ToString()+","+y.ToString()+"＼＼＼＼";
822 
823                     if (tm1==0)
824                     {
825                          pattern=4;
826                     timer1.Enabled=true;
827                     }
828                 }
829 
830                 // 角（左下）の判定
831                 if (
832                      GetPixelColor(x, y-thick).Name.ToString()=="fffffffe"
833                      && GetPixelColor(x+thick, y).Name.ToString()=="ffffffff")
834                 {
835                      this.Text=x.ToString()+","+y.ToString()+"LLLL";
836 
837                     if (tm1==0)
838                     {
839                          pattern=5;
840                     timer1.Enabled=true;
841                     }
842                 }
843 
844                 //  角（左上）の判定
845                 if (
846                      GetPixelColor(x+thick, y).Name.ToString()=="ffffffff"
847                      && GetPixelColor(x, y+thick).Name.ToString()=="fffffffe")
848                 {
849                      this.Text=x.ToString()+","+y.ToString()+"「「「「";
850 
851                     if (tm1==0)
852                     {
853                          pattern=6;
854                     timer1.Enabled=true;
855                     }
856                 }
857 
858                 //  角（右上）の判定
859                 if (
860                      GetPixelColor(x-thick, y).Name.ToString()=="ffffffff"
861                      && GetPixelColor(x, y+thick).Name.ToString()=="fffffffe")
862                 {
863                      this.Text=x.ToString()+","+y.ToString()+"----||||";
864 
865                     if (tm1==0)
866                     {
867                          pattern=7;
868                     timer1.Enabled=true;
869                     }
870                 }
871 
872                 //  角（右下）の判定
873                 if (
874                      GetPixelColor(x-thick, y).Name.ToString()=="ffffffff"
875                      && GetPixelColor(x, y-thick).Name.ToString()=="fffffffe")
876                 {
877                      this.Text=x.ToString()+","+y.ToString()+"」」」」";
878 
879                     if (tm1==0)
880                     {
881                          pattern=8;
882                     timer1.Enabled=true;
883                     }
884                 }
885 
886                 //  角（左猫耳）の判定
887                 if (
888                      GetPixelColor(x, y+thick).Name.ToString()=="fffffffe"
```

```
889                  && GetPixelColor(x+thick, y+thick).Name.ToString()=="
                         fffeffff")
890              {
891                   this.Text=x.ToString()+","+y.ToString()+"||||＼＼＼＼";
892
893                  if (tm1==0)
894                  {
895                       pattern=9;
896                  timer1.Enabled=true;
897                  }
898              }
899
900              //  角（右猫耳）の判定
901              if (
902                   GetPixelColor(x, y+thick).Name.ToString()=="fffffffe"
903                   && GetPixelColor(x-thick, y+thick).Name.ToString()=="
                         fffffeff")
904              {
905                   this.Text=x.ToString()+","+y.ToString()+"////||||";
906
907                  if (tm1==0)
908                  {
909                       pattern=10;
910                  timer1.Enabled=true;
911                  }
912              }
913
914              //  角（右向き）の判定
915              if (
916                   GetPixelColor(x-thick, y).Name.ToString()=="ffffffff"
917                   && GetPixelColor(x-thick, y-thick).Name.ToString()=="
                         fffeffff")
918              {
919                   this.Text=x.ToString()+","+y.ToString()+"→";
920
921                  if (tm1==0)
922                  {
923                       pattern=11;
924                  timer1.Enabled=true;
925                  }
926              }
927
928              //  角（左向き）の判定
929              if (
930                   GetPixelColor(x+thick, y).Name.ToString()=="ffffffff"
931                   && GetPixelColor(x+thick, y-thick).Name.ToString()=="
                         fffffeff")
932              {
933                   this.Text=x.ToString()+","+y.ToString()+"←";
934
935                  if (tm1==0)
936                  {
937                       pattern=12;
938                  timer1.Enabled=true;
939                  }
940              }
941
942
943          }
944
945
946                       //Stopwatchオブジェクトを作成する
947                  System.Diagnostics.Stopwatch sw = new System.Diagnostics.
                         Stopwatch();
948
949          private void Form1_KeyDown(object sender, KeyEventArgs e)
950          {
951              Pen hpen=new Pen(Color.FromArgb(0xff,0xff,0xff,0xff),thick);
952              Pen vpen=new Pen(Color.FromArgb(0xff,0xff,0xff,0xfe),thick);
953              Pen spen=new Pen(Color.FromArgb(0xff,0xff,0xfe,0xff),thick);
954              Pen bpen=new Pen(Color.FromArgb(0xff,0xfe,0xff,0xff),thick);
```

```
955             Pen cpen=new Pen(Color.FromArgb(0xff,0xff,0xff,0xfd),thick);
956
957              // 刺激図形の描画
958              if( e.KeyCode == Keys.A )
959              {
960                 Font newfont=new Font("MS明朝",40);
961                 pictureBox1.CreateGraphics().DrawString("NH_2", newfont,
        Brushes.Gray,100,200);
962                 pictureBox1.CreateGraphics().DrawLine(hpen,280,220,380,220);
963                 pictureBox1.CreateGraphics().DrawString("C", newfont,Brushes.
        Gray,400,200);
964                 pictureBox1.CreateGraphics().DrawLine(hpen,460,220,560,220);
965                 pictureBox1.CreateGraphics().DrawString("COOH", newfont,
        Brushes.Gray,580,200);
966                 pictureBox1.CreateGraphics().DrawString("R", newfont,Brushes.
        Gray,400,20);
967                 pictureBox1.CreateGraphics().DrawString("H", newfont,Brushes.
        Gray,400,400);
968                 pictureBox1.CreateGraphics().DrawLine(vpen,420,180,420,80);
969                 pictureBox1.CreateGraphics().DrawLine(vpen,420,260,420,380);
970
971                  say("アミノサン");
972              }
973
974              if( e.KeyCode == Keys.C )
975              {
976                  //直径100ピクセルの白色の円を描画する
977                  //pictureBox1.Image = new Bitmap(pictureBox1.Width,
                         pictureBox1.Height);
978                  Image bmp = pictureBox1.Image;
979                  Graphics g =                Graphics.FromImage(bmp);
980                  g.DrawEllipse(cpen,200,100,300,300);
981                  pictureBox1.Image = bmp;
982                  say("表示しました");
983              }
984
985              if( e.KeyCode == Keys.R )
986              {
987                  //白色破線の長方形を描画する
988                  //pictureBox1.Image = new Bitmap(pictureBox1.Width,
                         pictureBox1.Height);
989                  Image bmp = pictureBox1.Image;
990                  Graphics g =                Graphics.FromImage(bmp);
991                  g.DrawLine(hpen,200,100,500,100);
992                  g.DrawLine(vpen,500,100,500,400);
993                  g.DrawLine(hpen,500,400,200,400);
994                  g.DrawLine(vpen,200,400,200,100);
995                  pictureBox1.Image = bmp;
996                  say("表示しました");
997              }
998
999              if( e.KeyCode == Keys.H )
1000             {
1001                 //横線
1002                 pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1
                         .Height);
1003                 Image bmp = pictureBox1.Image;
1004                 Graphics g =                Graphics.FromImage(bmp);
1005                 g.DrawLine(hpen,200,100,500,100);
1006                 pictureBox1.Image = bmp;
1007                 say("表示しました");
1008             }
1009
1010             if( e.KeyCode == Keys.V )
1011             {
1012                 //縦線
1013                 pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1
                         .Height);
1014                 Image bmp = pictureBox1.Image;
1015                 Graphics g =                Graphics.FromImage(bmp);
1016                 g.DrawLine(vpen,200,100,200,400);
```

```
1017                    pictureBox1.Image = bmp;
1018                    say("表示しました");
1019                }
1020
1021                if( e.KeyCode == Keys.P )
1022                {
1023                    //傾きがプラスの斜線
1024                    pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1
                            .Height);
1025                    Image bmp = pictureBox1.Image;
1026                    Graphics g =                Graphics.FromImage(bmp);
1027                    g.DrawLine(spen,500,100,200,400);
1028                    pictureBox1.Image = bmp;
1029                    say("表示しました");
1030                }
1031
1032                if( e.KeyCode == Keys.M )
1033                {
1034                    //傾きがマイナスの斜線
1035                    pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1
                            .Height);
1036                    Image bmp = pictureBox1.Image;
1037                    Graphics g =                Graphics.FromImage(bmp);
1038                    g.DrawLine(bpen,200,100,500,400);
1039                    pictureBox1.Image = bmp;
1040                    say("表示しました");
1041                }
1042
1043                if( e.KeyCode == Keys.T )
1044                {
1045                    //3角形
1046                    pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1
                            .Height);
1047                    Image bmp = pictureBox1.Image;
1048                    Graphics g =                Graphics.FromImage(bmp);
1049                    g.DrawLine(vpen,200,100,200,400);
1050                    g.DrawLine(hpen,200,400,500,400);
1051                    g.DrawLine(bpen,500,400,200,100);
1052                    pictureBox1.Image = bmp;
1053                    say("表示しました");
1054                }
1055
1056                if( e.KeyCode == Keys.Y )
1057                {
1058                    //3角形
1059                    pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1
                            .Height);
1060                    Image bmp = pictureBox1.Image;
1061                    Graphics g =                Graphics.FromImage(bmp);
1062                    g.DrawLine(vpen,500,100,500,400);
1063                    g.DrawLine(hpen,500,400,200,400);
1064                    g.DrawLine(spen,200,400,500,100);
1065                    pictureBox1.Image = bmp;
1066                    say("表示しました");
1067                }
1068
1069
1070                if( e.KeyCode == Keys.F1 )
1071                {
1072                     this._buffer1 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device1);
1073                     this._buffer2 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device1);
1074
1075                     this._buffer3 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device1);
1076                     this._buffer4 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device2);
1077
1078                     this._buffer5 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device2);
```

```
1079                    this._buffer6 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device2);
1080
1081                    this._buffer7 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device2);
1082                    this._buffer8 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device1);
1083
1084                    this._buffer9 = new SecondaryBuffer("tones/WhiteNoise.wav",
                            this._device1);
1085                    this._buffer10 = new SecondaryBuffer("tones/WhiteNoise.wav",
                             this._device2);
1086
1087                    this._buffer11 = new SecondaryBuffer("tones/WhiteNoise.wav",
                             this._device1);
1088                    this._buffer12 = new SecondaryBuffer("tones/WhiteNoise.wav",
                             this._device2);
1089
1090                    this._buffer101 = new SecondaryBuffer("tones/500.wav", this.
                            _device3);
1091                    this._buffer102 = new SecondaryBuffer("tones/500.wav", this.
                            _device3);
1092                    this._buffer103 = new SecondaryBuffer("tones/500.wav", this.
                            _device4);
1093                    this._buffer104 = new SecondaryBuffer("tones/500.wav", this.
                            _device4);
1094
1095                    // パンの設定
1096                    this._buffer1.Pan=-10000;
1097                    this._buffer2.Pan=10000;
1098
1099                    this._buffer3.Pan=10000;
1100                    this._buffer4.Pan=10000;
1101
1102                    this._buffer5.Pan=10000;
1103                    this._buffer6.Pan=-10000;
1104
1105                    this._buffer7.Pan=-10000;
1106                    this._buffer8.Pan=-10000;
1107
1108                    this._buffer9.Pan=10000;
1109                    this._buffer10.Pan=-10000;
1110                    this._buffer11.Pan=-10000;
1111                    this._buffer12.Pan=10000;
1112
1113                    this._buffer101.Pan=-10000;
1114                    this._buffer102.Pan=10000;
1115                    this._buffer103.Pan=-10000;
1116                    this._buffer104.Pan=10000;
1117
1118                    this._buffer1.Volume=-10000;
1119                    this._buffer2.Volume=-10000;
1120                    this._buffer3.Volume=-10000;
1121                    this._buffer4.Volume=-10000;
1122                   this._buffer5.Volume=-10000;
1123                    this._buffer6.Volume=-10000;
1124                    this._buffer7.Volume=-10000;
1125                    this._buffer8.Volume=-10000;
1126                    this._buffer9.Volume=-10000;
1127                    this._buffer10.Volume=-10000;
1128                    this._buffer11.Volume=-10000;
1129                    this._buffer12.Volume=-10000;
1130                    this._buffer101.Volume=-10000;
1131                    this._buffer102.Volume=-10000;
1132                    this._buffer103.Volume=-10000;
1133                    this._buffer104.Volume=-10000;
1134
1135
1136
1137                    this._buffer1.Play(0, BufferPlayFlags.Looping);
1138                    this._buffer2.Play(0, BufferPlayFlags.Looping);
```

```
1139                    this._buffer3.Play(0, BufferPlayFlags.Looping);
1140                    this._buffer4.Play(0, BufferPlayFlags.Looping);
1141                    this._buffer5.Play(0, BufferPlayFlags.Looping);
1142                    this._buffer6.Play(0, BufferPlayFlags.Looping);
1143                    this._buffer7.Play(0, BufferPlayFlags.Looping);
1144                    this._buffer8.Play(0, BufferPlayFlags.Looping);
1145                    this._buffer9.Play(0, BufferPlayFlags.Looping);
1146                    this._buffer10.Play(0, BufferPlayFlags.Looping);
1147                    this._buffer11.Play(0, BufferPlayFlags.Looping);
1148                    this._buffer12.Play(0, BufferPlayFlags.Looping);
1149
1150                    this._buffer101.Play(0, BufferPlayFlags.Looping);
1151                    this._buffer102.Play(0, BufferPlayFlags.Looping);
1152                    this._buffer103.Play(0, BufferPlayFlags.Looping);
1153                    this._buffer104.Play(0, BufferPlayFlags.Looping);
1154
1155
1156                    say("縦、横、ホワイトノイズにしました");
1157                }
1158
1159                if( e.KeyCode == Keys.F2 )
1160                {
1161                    this._buffer1 = new SecondaryBuffer("tones/400.wav", this.
                        _device1);
1162                    this._buffer2 = new SecondaryBuffer("tones/400.wav", this.
                        _device1);
1163
1164                    this._buffer3 = new SecondaryBuffer("tones/500.wav", this.
                        _device1);
1165                    this._buffer4 = new SecondaryBuffer("tones/500.wav", this.
                        _device2);
1166
1167                    this._buffer5 = new SecondaryBuffer("tones/400.wav", this.
                        _device2);
1168                    this._buffer6 = new SecondaryBuffer("tones/400.wav", this.
                        _device2);
1169
1170                    this._buffer7 = new SecondaryBuffer("tones/500.wav", this.
                        _device2);
1171                    this._buffer8 = new SecondaryBuffer("tones/500.wav", this.
                        _device1);
1172
1173                    this._buffer9 = new SecondaryBuffer("tones/pulse_30.wav",
                        this._device1);
1174                    this._buffer10 = new SecondaryBuffer("tones/pulse_30.wav",
                        this._device2);
1175
1176                    this._buffer11 = new SecondaryBuffer("tones/pulse_30.wav",
                        this._device1);
1177                    this._buffer12 = new SecondaryBuffer("tones/pulse_30.wav",
                        this._device2);
1178                    this._buffer101 = new SecondaryBuffer("tones/800.wav", this.
                        _device3);
1179                    this._buffer102 = new SecondaryBuffer("tones/800.wav", this.
                        _device3);
1180                    this._buffer103 = new SecondaryBuffer("tones/800.wav", this.
                        _device4);
1181                    this._buffer104 = new SecondaryBuffer("tones/800.wav", this.
                        _device4);
1182
1183                    // パンの設定
1184                    this._buffer1.Pan=-10000;
1185                    this._buffer2.Pan=10000;
1186
1187                    this._buffer3.Pan=10000;
1188                    this._buffer4.Pan=10000;
1189
1190                    this._buffer5.Pan=10000;
1191                    this._buffer6.Pan=-10000;
1192
1193                    this._buffer7.Pan=-10000;
```

```
1194                    this._buffer8.Pan=-10000;
1195
1196                    this._buffer9.Pan=10000;
1197                    this._buffer10.Pan=-10000;
1198                    this._buffer11.Pan=-10000;
1199                    this._buffer12.Pan=10000;
1200                    this._buffer101.Pan=-10000;
1201                    this._buffer102.Pan=10000;
1202                    this._buffer103.Pan=-10000;
1203                    this._buffer104.Pan=10000;
1204
1205                    this._buffer1.Volume=-10000;
1206                    this._buffer2.Volume=-10000;
1207                    this._buffer3.Volume=-10000;
1208                    this._buffer4.Volume=-10000;
1209                   this._buffer5.Volume=-10000;
1210                    this._buffer6.Volume=-10000;
1211                    this._buffer7.Volume=-10000;
1212                    this._buffer8.Volume=-10000;
1213                    this._buffer9.Volume=-10000;
1214                    this._buffer10.Volume=-10000;
1215                    this._buffer11.Volume=-10000;
1216                    this._buffer12.Volume=-10000;
1217                    this._buffer101.Volume=-10000;
1218                    this._buffer102.Volume=-10000;
1219                    this._buffer103.Volume=-10000;
1220                    this._buffer104.Volume=-10000;
1221
1222
1223                    this._buffer1.Play(0, BufferPlayFlags.Looping);
1224                    this._buffer2.Play(0, BufferPlayFlags.Looping);
1225                    this._buffer3.Play(0, BufferPlayFlags.Looping);
1226                    this._buffer4.Play(0, BufferPlayFlags.Looping);
1227                    this._buffer5.Play(0, BufferPlayFlags.Looping);
1228                    this._buffer6.Play(0, BufferPlayFlags.Looping);
1229                    this._buffer7.Play(0, BufferPlayFlags.Looping);
1230                    this._buffer8.Play(0, BufferPlayFlags.Looping);
1231                    this._buffer9.Play(0, BufferPlayFlags.Looping);
1232                    this._buffer10.Play(0, BufferPlayFlags.Looping);
1233                    this._buffer11.Play(0, BufferPlayFlags.Looping);
1234                    this._buffer12.Play(0, BufferPlayFlags.Looping);
1235                    this._buffer101.Play(0, BufferPlayFlags.Looping);
1236                    this._buffer102.Play(0, BufferPlayFlags.Looping);
1237                    this._buffer103.Play(0, BufferPlayFlags.Looping);
1238                    this._buffer104.Play(0, BufferPlayFlags.Looping);
1239
1240                    say("横をホワイトノイズ、、縦をピンクノイズにしました");
1241                }
1242
1243                if( e.KeyCode == Keys.F9 )
1244                {
1245                    if (guide==1)
1246                    {
1247                        guide=0;
1248                        _buffer101.Stop();
1249                        _buffer102.Stop();
1250                        _buffer103.Stop();
1251                        _buffer104.Stop();
1252                        say("ガイド音 Off");
1253                    } else if (guide==0)
1254                    {
1255                        guide=1;
1256                        this._buffer101.Play(0, BufferPlayFlags.Looping);
1257                        this._buffer102.Play(0, BufferPlayFlags.Looping);
1258                        this._buffer103.Play(0, BufferPlayFlags.Looping);
1259                        this._buffer104.Play(0, BufferPlayFlags.Looping);
1260                        say("ガイド音 On");
1261                    }
1262
1263                }
1264
```

```
1265                if( e.KeyCode == Keys.F11 )
1266                {
1267                     timer2.Enabled=false;
1268                     say("パターンテストモードにしました");
1269                }
1270
1271                if( e.KeyCode == Keys.F12 )
1272                {
1273                     timer2.Interval=20;
1274                     timer2.Enabled=true;
1275                     say("画面スキャンモードにしました");
1276                }
1277
1278                if( e.KeyCode == Keys.Enter )
1279                {
1280                     if (wmode==0)
1281                     {
1282                          /*sw.Reset();
1283                          //ストップウォッチを開始する
1284                          sw.Start();*/
1285
1286                          wmode=1;
1287                          say("軌跡表示開始");
1288                     } else {
1289                          //sw.Stop();
1290                          wmode=0;
1291                          pictureBox1.Image.Save(@"data.jpg", System.Drawing.
                                 Imaging.ImageFormat.Jpeg);
1292                          say("軌跡画像を保存しました");
1293                     }
1294                }
1295
1296                if( e.KeyCode == Keys.Space )
1297                {
1298                     say(sw.Elapsed.ToString());
1299                }
1300
1301                if( e.KeyCode == Keys.Up )
1302                {
1303                    if (area<200)
1304                    {
1305                         area+=10;
1306                         say(area.ToString());
1307                    }
1308                }
1309
1310                if( e.KeyCode == Keys.Down )
1311                {
1312                    if (area>10)
1313                    {
1314                         area-=10;
1315                         say(area.ToString());
1316                    }
1317                }
1318
1319                if( e.KeyCode == Keys.PageUp )
1320                {
1321                    if (thick<=90)
1322                    {
1323                         thick+=5;
1324                    }
1325                    say(thick.ToString());
1326                }
1327
1328                if( e.KeyCode == Keys.PageDown )
1329                {
1330                    if (thick>=20)
1331                    {
1332                         thick-=5;
1333                    }
1334                    say(thick.ToString());
```

```
1335
1336                }
1337
1338                if( e.KeyCode == Keys.D1 )
1339                {
1340                    timer1.Interval=40;
1341                    say("40ミリ秒");
1342                }
1343
1344                if( e.KeyCode == Keys.D2 )
1345                {
1346                    timer1.Interval=50;
1347                    say("50ミリ秒");
1348                }
1349
1350                if( e.KeyCode == Keys.D3 )
1351                {
1352                    timer1.Interval=60;
1353                    say("60ミリ秒");
1354                }
1355
1356                if( e.KeyCode == Keys.D4 )
1357                {
1358                    timer1.Interval=70;
1359                    say("70ミリ秒");
1360                }
1361
1362                if( e.KeyCode == Keys.D5 )
1363                {
1364                    timer1.Interval=80;
1365                    say("80ミリ秒");
1366                }
1367
1368                if( e.KeyCode == Keys.D6 )
1369                {
1370                    timer1.Interval=90;
1371                    say("90ミリ秒");
1372                }
1373
1374                if( e.KeyCode == Keys.D7 )
1375                {
1376                    timer1.Interval=100;
1377                    say("100ミリ秒");
1378                }
1379
1380                if( e.KeyCode == Keys.D8 )
1381                {
1382                    timer1.Interval=110;
1383                    say("110ミリ秒");
1384                }
1385
1386                if( e.KeyCode == Keys.D9 )
1387                {
1388                    timer1.Interval=120;
1389                    say("120ミリ秒");
1390                }
1391
1392                if( e.KeyCode == Keys.D0 )
1393                {
1394                    timer1.Interval=130;
1395                    say("130ミリ秒");
1396                }
1397
1398
1399
1400                if( e.KeyCode == Keys.Subtract )
1401                {
1402                    tm1=0;
1403                    pattern=1;
1404                    timer1.Enabled=true;
1405                }
```

```
1406
1407                if( e.KeyCode == Keys.Add )
1408                {
1409                    tm1=0;
1410                    pattern=2;
1411                    timer1.Enabled=true;
1412                }
1413
1414                if( e.KeyCode == Keys.Divide )
1415                {
1416                    tm1=0;
1417                    pattern=3;
1418                    timer1.Enabled=true;
1419                }
1420
1421                if( e.KeyCode == Keys.Multiply )
1422                {
1423                    tm1=0;
1424                    pattern=4;
1425                    timer1.Enabled=true;
1426                }
1427
1428                if( e.KeyCode == Keys.NumPad1 )
1429                {
1430                    tm1=0;
1431                    pattern=5;
1432                    timer1.Enabled=true;
1433                }
1434
1435                if( e.KeyCode == Keys.NumPad7 )
1436                {
1437                    tm1=0;
1438                    pattern=6;
1439                    timer1.Enabled=true;
1440                }
1441
1442                if( e.KeyCode == Keys.NumPad9 )
1443                {
1444                    pattern=7;
1445                    timer1.Enabled=true;
1446                }
1447
1448                if( e.KeyCode == Keys.NumPad3 )
1449                {
1450                    pattern=8;
1451                    timer1.Enabled=true;
1452                }
1453
1454                if( e.KeyCode == Keys.Escape )
1455                {
1456                     this.pictureBox1.Image = null;
1457                    // Graphics オブジェクトの取得
1458                    pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1
                        .Height);
1459                    Image bmp = pictureBox1.Image;
1460                    Graphics g = Graphics.FromImage(bmp);
1461
1462                    _buffer1.Volume=-10000;
1463                    _buffer2.Volume=-10000;
1464                    _buffer3.Volume=-10000;
1465                    _buffer4.Volume=-10000;
1466                    _buffer5.Volume=-10000;
1467                    _buffer6.Volume=-10000;
1468                    _buffer7.Volume=-10000;
1469                    _buffer8.Volume=-10000;
1470                    _buffer9.Volume=-10000;
1471                     _buffer10.Volume=-10000;
1472                    _buffer11.Volume=-10000;
1473                    _buffer12.Volume=-10000;
1474                    _buffer101.Volume=-10000;
1475                    _buffer102.Volume=-10000;
```

```
1476                    _buffer103.Volume=-10000;
1477                    _buffer104.Volume=-10000;
1478
1479                     say("クリア");
1480                }
1481
1482
1483            }
1484
1485             private void pictureBox1_MouseDown_1(object sender, MouseEventArgs e)
1486             {
1487                 if (wmode==1)
1488                 {
1489                      start = 1 ;
1490                      startX = e.X ;
1491                      startY = e.Y ;
1492                 }
1493             }
1494
1495             private void pictureBox1_MouseUp_1(object sender, MouseEventArgs e)
1496             {
1497                 if (wmode==1)
1498                 {
1499                     start = 0 ;
1500                 }
1501             }
1502
1503             private void pictureBox1_MouseMove_1(object sender, MouseEventArgs e)
1504             {
1505                  if (wmode==1)
1506                  {
1507                      if (start == 0) return ;
1508                     //pictureBox1.Image = new Bitmap(pictureBox1.Width,
                            pictureBox1.Height);
1509                     Image bmp = pictureBox1.Image;
1510                     Graphics g =                Graphics.FromImage(bmp);
1511                      g.DrawLine(Pens.Yellow, startX-1, startY-1, e.X-1, e.Y-1) ;
1512                      startX = e.X ;
1513                      startY = e.Y;
1514                     //pictureBox1.Image = bmp;
1515                     //pictureBox1.Refresh();
1516                  }
1517             }
1518
1519             private void pictureBox1_Click(object sender, EventArgs e)
1520             {
1521
1522             }
1523
1524             int tm3=0;
1525             private void timer3_Tick(object sender, EventArgs e)
1526             {
1527                  tm3++;
1528
1529                  if ( near_l==0 )
1530                  {
1531                     //_buffer100.Volume=-10000;
1532                     //timer3.Enabled=false;
1533                     //tm3=0;
1534                  }
1535             }
1536
1537             int tm4=0;
1538             private void timer4_Tick(object sender, EventArgs e)
1539             {
1540                  tm4++;
1541
1542                  if (near_r==0 )
1543                  {
1544                     //_buffer100.Volume=-10000;
1545                     //timer4.Enabled=false;
```

```
1546                        //tm4=0;
1547                    }
1548
1549                }
1550
1551        }
1552    }
```