高松大学紀要，41．41～73

心理学実験プログラムの開発と教育場面での活用（２）

西　浦　和　樹*・伊　藤　克　浩**・近　藤　武　夫***

A development of computer program designed for introductory psychological experiment and application to introductory psychological education（2）

Kazuki Nishiura, Katsuhiro Ito, Takeo Kondou

Abstract

The purpose of this study was to develop a computer program designed for psychological experiment “Visual illusory conjunction” and “Short term memory”, and to apply it to an introductory psychological education．We attempted to describe it by a programming language “Visual Basic 6” for Windows 2000，and to open the source code．

The features of the program are（1）realization of control of rapid display by DirectX7 SDK ；(2) use of DirectInput in order to conduct for reaction-time critical tasks；(3) programming for some experimental procedures：for example，reaction time．

The findings of the applications to class “introductory psychological experiment” suggest that we would have usefulness of reaction time task with DirectInput．

Key words：Visual Basic　DirectX　Reaction time task

１．はじめに

電子メール，WWWなどネットワークを介した情報技術の利用が進むにつれて，基本ソフトウェア（OS）の世代交代が進展しつつある。Windowsマシンの普及とともに，Windowsをベースとしたプログラム開発も比較的容易になってきている。

しかしながら，近年の日本心理学会における知覚・認知・記憶領域における学会発表資料を見る限りにおいて，AVタキストスコープ，パーソナルコンピュータを利用した心理学実験が実施されているにもかかわらず，依然として，Windows上での心理学実験はほと

＊　高松短期大学保育学科
＊＊　比治山大学現代文化学部
＊＊＊広島大学大学院教育学研究科

んど行われていない。この最大の理由として，学会員のほとんどがコンピュータ以外を専門とする心理学者であることが挙げられる。

西浦・伊藤（2001）は，Windows上で比較的容易に心理学実験が行えるように，心理学実験「注意の範囲（Sparling，1960）」を題材として，Visual BasicとDirectXを活用し，ミリ秒単位の画面制御を可能とする心理学実験プログラムを開発した。ここで必要とされた手続きは，凝視点が呈示され，続いてミリ秒単位で刺激が呈示されるというものであった。この実験プログラムを教育実践に活用し，教育場面における心理学実験プログラム活用の有用性を確認した。

しかし，依然として，西浦（1998，1999）で用いたようなRSVP法による高速な刺激の継時呈示が要求されるような心理学実験プログラムや，心理学実験で利用頻度の高い反応時間の計測が要求されるようなWindows上での心理学実験プログラムの開発は依然として行われていないのが現状である。

２．本研究の目的

本研究は，西浦・伊藤（2001）に引き続き，高速な画面制御や精度の高い反応時間の計測を必要とする心理学実験プログラムの開発を試み，教育場面で活用することを目的とする。ここでは，Direct Drawによる画面制御プログラミングおよび反応時間の計測プログラミングの開発指針を明らかにし，教育実践における実験データの活用方法を検討する。この取り組みにより，授業「心理学実験」改善のための一助となるであろうし，心理学教育カリキュラム改善のための基礎的資料とすることが期待できる。

３．心理学実験プログラム開発の指針

まず，心理学教育場面での利用を考えると，大学で心理学実験を受講するユーザはワープロ，メール，WWWなどの比較的利用頻度の高いアプリケーションを使うものの，毎日利用することが少ない初心者ユーザであることが想定される。このようなユーザが実験者や被験者として実験プログラムを利用することを考えると，実験プログラムはユーザビリティに配慮した設計でなければならない。

また，心理学実験では，高度な画面制御やキーボード等による入出力制御が要求されるので，Windows APIを介さずにDirectXを活用し，直接的にハードウェアを操作できるようにすることで，比較的容易にアプリケーションの開発が可能なように配慮しなければな

らない。

以下では，Visual　BasicとDirectXにより実験プログラム「視覚的錯誤」および「短期記憶」を開発し，心理学教育実践での活用を試みた。

４．実験プログラム「視覚的錯誤」の開発とその教育実践

「視覚的錯誤」実験の概要　「視覚的錯誤」実験は，刺激が継時的に呈示されると，ある時点の刺激が別の時点の刺激と見間違う見誤り現象を確認するための実験である（Broadbent & Broadbent，1986；McLean，Broadbent，& Broadbent，1982；西浦，1998）。

刺激が次々と連続して（継時的に）呈示される刺激呈示法は，高速継時呈示法と呼ばれている。このような状況で，被験者は次々と出現する黒文字系列の中から白文字を見つけ出し，その白文字が何であったのかを報告すること，さらにその白文字がどの程度よく見えたのかを報告することが求められる。

実験の結果は，白文字を正しく報告することもあれば，その前後の文字を白文字であったと誤って報告することもある。これら文字が報告される範囲は，白文字とその前後の文字までのおよそ３項目程度である。このことから，視覚的錯誤現象は，並列的に３項目程度の保持能力を持つ短期概念バッファからの文字の読み取りに困難が生じたために生起するものと考えられている。

実験プログラムの開発　本実験プログラムは，西浦・伊藤（2001）の「注意の範囲」実験プログラムを改変し，学生が比較的容易に視覚的錯誤現象を確認できるようにという意図を持って開発された。実験プログラムの流れは，DirectX宣言部，刺激セットのランダム化，被験者データの記録，DirectX初期化，刺激の呈示，呈示刺激の記録，DirectXの解放となっている。

以下では，実験プログラム作成時に注意した点を述べる。従来の実験プログラムは凝視点と刺激配列の２画面の制御で十分であったが，本実験では，凝視点，15項目のプレターゲット項目，ターゲット項目，８項目のポストターゲット項目の25画面の制御が必要である。さらに，西浦（1998，1999）では，文字の報告の仕方を自由記述としたが，本実験では，白文字とその前後６文字，さらに「（刺激系列に含まれない）ダミー文字」と「わからない」を含む８つの選択肢とした。この変更は，実験データの集計（平均報告率の算出）を容易にするためものである。また，「よく見えない」から「よく見える」の主観的明瞭度について５段階評定とした。実験データはデータファイル（RSVPExpData.txt）と

して保存され，実験終了後，データファイルは条件別に相対系列位置ごとの報告数の一覧となっており，表計算ソフト上で「平均報告率」「主観的明瞭度」の結果処理に利用可能な状態になっている。

教育実践　K大学教育学部の授業「心理学実験」の受講生は10名（うち留学生１名が参加）であり，実験の概要については，配布資料を用いて解説した。その中で，データ取得に伴う留意点として倫理規定の解説を行い，データの整理，レポート作成の手順を解説し，自分自身で実験実習が進行できるように配慮した。ここでは，実験計画と手続きを把握させるために，受講生自身が被験者となり実験に参加し，実験に行うに当たっての留意点として，倫理規定等の解説を同時に行った。その後，受講生が受講生以外の学生１名を被験者として実験を実施した。このときの実験条件は，西浦（1998）を参考にISI 75ms条件とし，視覚的錯誤現象が比較的安定して確認できることがわかっているためである。なお，本実験は，西浦（1998）の実験１の追試であり，文字種の違い（アルファベット，平仮名，漢字）が視覚的錯誤現象に及ぼす影響を検討することとした。

実験データの取得は，学生が自ら被験者の依頼を行い，実験室を共有する方法で行った。このため，データの取得には約２週間を費やすこととなった。このようにして得た実験データ（18名）に基づいて現象の確認を行うと同時に，実験データを多角的に検討することの有用性を理解させるために，３次元視覚情報化による解釈を試みた。

実験の結果は図１に示される。実験の結果に従い，文字種（3）×相対系列位置（3）の分散分析を行い，文字種（$F(2,34)=19.406$，$p<.01$），相対系列位置（$F(2,34)=76.424$，

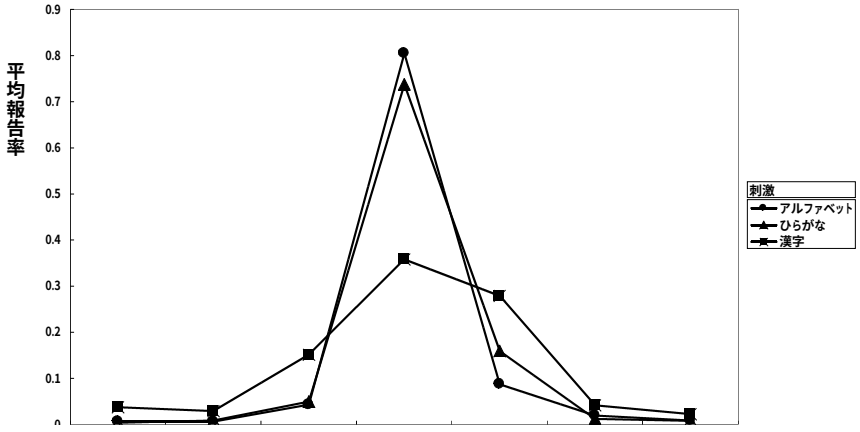


図１　視覚的錯誤実験の結果（相対系列位置における平均報告率）

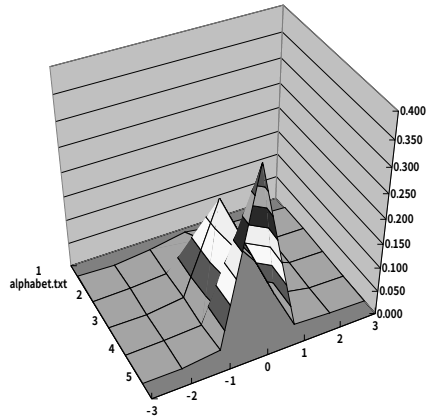


図２－１　相対系列位置及び主観的明瞭度に対する平均報告率（アルファベット条件）

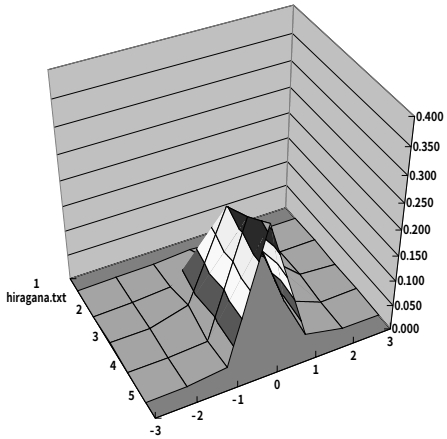


図２－２　相対系列位置及び主観的明瞭度に対する平均報告率（ひらがな条件）

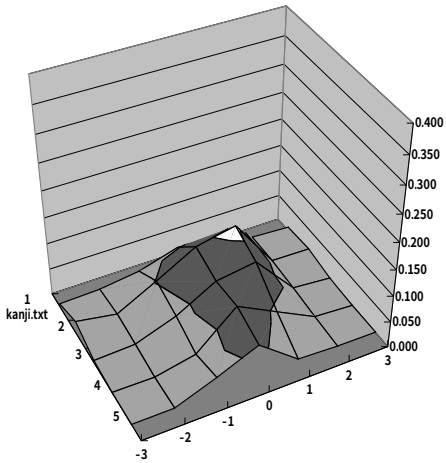


図２－３　相対系列位置及び主観的明瞭度に対する平均報告率（漢字条件）

p<.01）の主効果，文字種×相対系列位置（$F(4,68)=61.030$，p<.01）の交互作用が有意であることを確認した。下位検定として文字種についての多重比較（Ryan法による）を行った結果，アルファベット条件と平仮名条件は漢字条件より報告率が高かった。つまり，アルファベットと平仮名では漢字よりも標的である白文字の報告が容易であったと考えられる。また，交互作用の単純主効果（p<.01）が見られ，アルファベット条件の報告が比較的容易であり，アルファベット条件に比べ平仮名条件は相対系列位置１での報告が多く，また他２条件と比較して漢字条件は相対系列位置－１，１での報告が多かった。

これらの結果は，西浦（1998）の実験結果と一致し，高速に連続して文字が呈示される場合，同時に３文字程度の項目を保持する短期概念バッファの存在することを示すものであり，このことが確認できた。また，主観的明瞭度を加味した３次元グラフ（図２－１，図２－２，図２－３）によって，正しい文字を報告したとしても主観的な見えの査定は動的に変化する様子が見て取れた。

学生へのデータ返却を行い，レポート作成させると同時に，各自のデータをグラフ上にプロットさせることで，本実験への関心を高めさせることができた。今後の課題として，主観的明瞭度の動的変化について詳細な分析を行うと同時に，視覚的錯誤現象解明に向けた新たな実験計画の策定及びデータの蓄積が必要であると考えられる。

５．実験プログラム「短期記憶」の開発とその教育実践

「短期記憶」実験の概要　「短期記憶」実験は，短期記憶からの文字の検索がどのように行われているのかについて反応時間を手がかりにして検討するための実験である。被験者は，記憶セットの中から１～４個の記銘項目を覚えておき，数秒後に出現するテスト項目が記銘項目に含まれていたかどうかを判断することが求められる（図３）。実験の結果は，記銘項目が１～４項目と増えるにしたがって，反応時間が長くなることが見出されている。これらの結果から，短期記憶からの情報の検索が系列的・並列的探索が行われているのか，あるいは悉皆走査・途中打ち切り走査が行われているのか議論の余地が残されている。

実験プログラムの開発　本実験プログラムは上述の「視覚的錯誤」実験プログラムの一部を変更することで作成した。主たる変更箇所は，反応時間取得プログラムを組み込んだ点である。

以下では，実験プログラム作成時に注意した点を述べる。従来の実験プログラムは，被

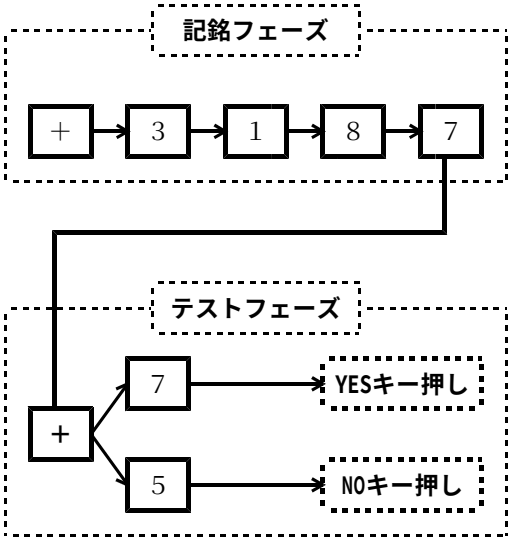


図３　短期記憶実験の刺激呈示例

実験開始時間，被験者名，年齢，性別，矯正視力，呈示時間，刺激間間隔，記憶セットの大きさ

```
開始時間2 nishirua      30 男性    裸眼        500    0    4
  4  3  6  2  0  9  7  5  8  1  3  1  1  591
  5  8  3  2  0  4  7  1  6  9  2  0  0  381
  4  8  6  1  5  7  3  2  0  9  1  0  0  421
  8  0  3  4  9  5  7  6  1  2  4  0  0  591
  1  2  9  7  3  6  8  5  0  4  1  0  0  601
  7  9  0  1  4  2  6  3  5  8  1  0  0  712
  2  0  7  9  3  4  8  5  6  1  3  0  0  442
  9  7  8  1  6  3  5  2  4  0  3  1  1  662
  0  3  7  9  6  1  4  8  5  2  1  0  0  752
  0  7  9  2  5  1  6  8  4  3  2  0  0  501
  6  4  3  2  1  9  8  0  5  7  3  0  0  471
  1  3  6  8  4  2  5  0  7  9  1  0  0  411
  2  3  0  6  9  5  8  1  7  4  4  1  1  692
  8  0  3  2  4  5  6  1  7  9  4  0  0  482
  6  7  1  2  4  0  9  3  5  8  2  0  0  361
  4  3  2  7  1  6  8  9  0  5  3  1  1  732
  2  1  3  8  5  4  7  9  6  0  2  0  0  431
```

刺激セット「0」～「9」までをランダム順に並び替えた数列（この中から記憶セットの大きさに応じて先頭の項目から順次呈示）

記憶セットの大きさ，記憶セットの中にテスト項目が含まれているかどうか（0：含まれる場合；1：含まれない場合），被験者の反応（0：Yes反応；1：No反応），反応時間

図４　短期記憶実験のデータファイルの例

験者のパフォーマンスを測定するため，キー押しによる正誤判断の確認を行えばよかった。しかし，本実験プログラムは，テスト刺激の呈示と同時にタイマ時間の計測を開始し，キー押しと同時にタイマ時間を取得する。このような実験手続きをWindows上での実験プログラムの記述に置き換えると，キー押しの状態を格納しているバッファ情報を取得し，

バッファ情報をクリアしながらリアルタイムにキー情報を得られるように記述しなければならない。もしこのように記述しなければ，リアルタイムに精確な反応時間を取得できないという問題が生じる。

本実験で得られた反応時間は，実験データファイル（MemoryScanningData.txt）として保存され，実験終了後，表計算ソフト上で結果処理に利用される。なお，実験データファイルの書式は，図４に示されるとおりである。各行の右から４項目が利用される。その中で，右から４番目の項目は記憶セットの大きさ，２，３番目の２項目は，記憶セット項目がテスト項目として呈示されたかどうか，及び被験者の反応，の順に記録されている。つまり，後の反応時間の集計では，Yes反応は（０，０），No反応は（１，１）の中央値が算出され，平均反応時間として図示される（図５）。

教育実践　K大学教育学部の授業「心理学実験」の受講生は８名であり，実験の概要については，配布資料を用いて解説した。その中で，データ取得，結果の整理，レポート作成の手順を解説し，心理学実験が自学自習できるように配慮した。先の実験と同様に，実験計画と手続きの把握のために，受講生自身が被験者となり，後に受講生以外の被験者２名に協力を要請する方法で実験を行った。実験計画は，記憶セットの大きさ（１～４項目）×テスト項目が記憶セットに含まれているかどうか（Yes／No）の２要因計画であった。24名の被験者が実験に参加し，先の実験と同様，実験室を共有する方法で，データ取得に２週間ほどの期間を要した。

実験結果の集計は，記憶項目数ごとにYes反応とNo反応についての反応時間の中央値を算出した（図５）。記憶セットの大きさ（4）×テスト項目が記憶セットに含まれているかどうか（2）の分散分析を行った結果，記憶セットの大きさ（$F(3,69)=56.097$，$p<.01$），テスト項目が記憶セットに含まれているかどうか（$F(1,23)=35.698$，$p<.01$）の主効果が有意であった。また，両者の交互作用についても有意であり（$F(3,69)=8.653$，$p<.01$），下位検定の結果，記憶セットの大きさ３（547.5ms）と４（560.6ms）を除くすべての項目間に有意差が見られた。実験の結果は，１項目の走査にYes反応時には38.2ms（切片395.8ms），No反応時には25.1ms（切片464.8ms）の時間を要することが示された。

本実験結果は，記憶項目数の増加に伴って反応時間が漸増することが示され，記憶セットとテスト項目を系列的に比較処理しているとするSternberg（1966）の実験結果と一致する。ところが，Sternberg（1966）の実験結果と比較し，Yes反応とNo反応で記憶走査に要する時間が異なることから，Yes反応では項目の比較が終了した時点で処理を打ち切る中

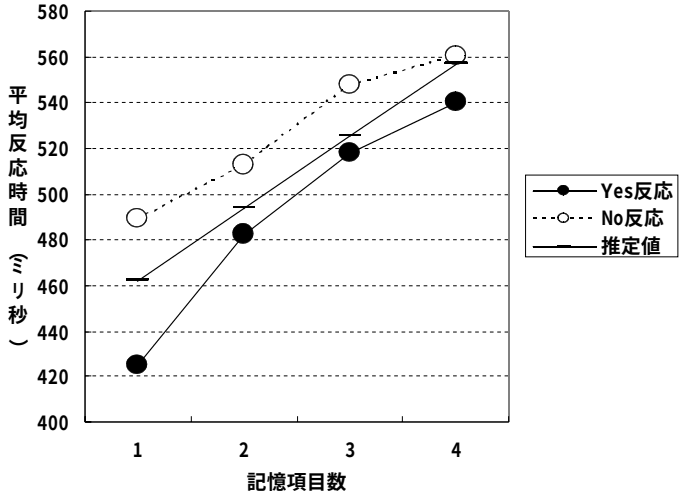


図５　短期記憶実験の結果（記憶項目数に対する平均反応時間）

途打ち切り走査方略を，No反応では最後まで項目を走査する悉皆探索方略を取りながら，記憶探索するものと考えられよう。ただし，これらの点は追試を行うなど，検討の余地が残されている。

６．結論と今後の課題

本研究では，２つの心理学実験プログラムの開発を行い，授業にて実験プログラムの活用を試みた。また，本研究では，西浦（1998）やSternberg（1966）の追試を行うことで，実験プログラムの動作確認を同時に行った。本教育実践で取得した実験データは，心理学実験に初めて参加する被験者も含まれており，これらの被験者は実験に不慣れであり，実験以外の剰余変数の影響を考慮する必要もあるだろうが，概ね先行研究の結果を確認することができた。したがって，心理学実験の概要説明などで，学生の理解を助けるために実験のデモンストレーションを行うなど，多くの教育場面での活用の可能性が見出された。

さらに，実験プログラムの開発の観点から，Direct Inputの利用による反応時間の計測が可能になることで，心理学実験で利用されることの多い行動指標を得ることが可能となった。この点は，新たな実験プログラムの開発が容易になったことからも意義深い。本実験プログラム「短期記憶」を本論文の補助資料（プログラムリスト１～４）として添付するので，各自の実験計画に応じて適宜修正するなど，心理学実験環境改善のために役立てられることを大いに期待する。

引用文献

McLean，J．P．，Broadbent，D．E．，& Broadbent，M．H．P．1982 Combining attributes in rapid serial visual presentation task．*Journal of Experimental Psychology：Human Perception and Performance*，**22**，332－341．

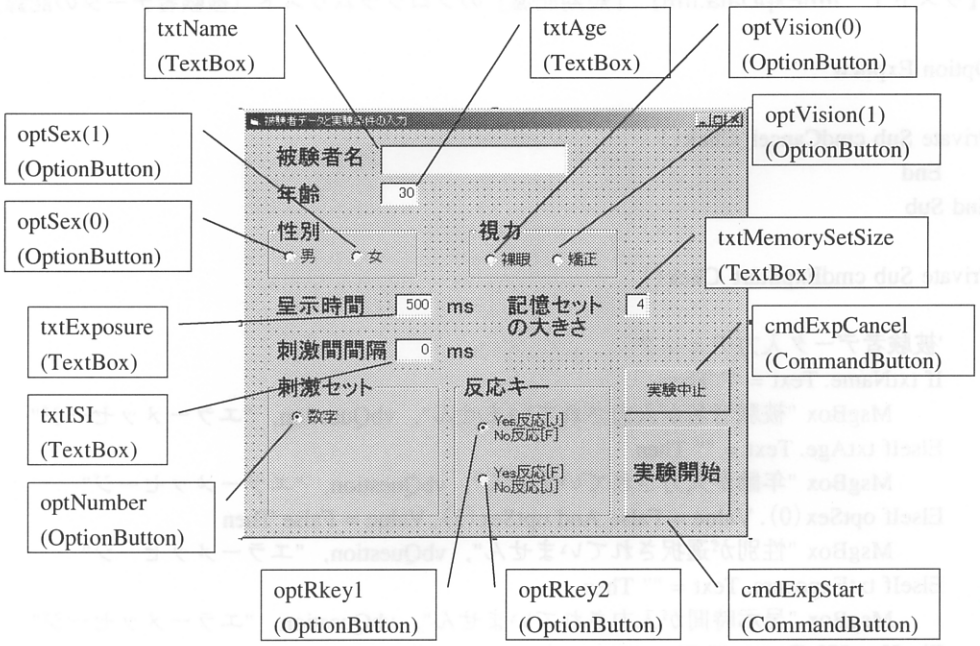
Broadbent，D．E．，& Broadbent，M．H．P．1986 Encoding speed of visual features and the occurrence of illusory conjunctions．*Perception*，**15**，515－524．

西浦和樹　1998　高速度継時呈示事態における注意資源の消費とモニタリングに関する研究　心理学研究，**69**，178－187．

西浦和樹・伊藤克浩　2001　Visual　Basicを活用した心理学実験プログラムの開発　日本教育工学会第17回全国大会講演論文集，527－528．

Sternberg，S．1966 High-speed scanning in human memory．Science，**153**，652－654．

利島保・生和秀敏（編著）　1993　心理学のための実験マニュアル：入門から基礎・発展へ　北大路書房



補助資料　被験者データフォームデザイン（frmExpData.frm）．リスト1のプログラムリストに対応する。デザインの説明は，オブジェクト名（オブジェクト）を指す。

【リスト１　frmExpData.frm】「短期記憶」のプログラムリスト（被験者データの記録）

```
Option Explicit

Private Sub cmdCancel_Click()
    End
End Sub

Private Sub cmdExpStart_Click()

    '被験者データ入力チェック
    If txtName.Text = "" Then
        MsgBox "被験者名が入力されていません", vbQuestion, "エラーメッセージ"
    ElseIf txtAge.Text = "" Then
        MsgBox "年齢が入力されていません", vbQuestion, "エラーメッセージ"
    ElseIf optSex(0).Value = False And optSex(1).Value = False Then
        MsgBox "性別が選択されていません", vbQuestion, "エラーメッセージ"
    ElseIf txtExposure.Text = "" Then
        MsgBox "呈示時間が入力されていません", vbQuestion, "エラーメッセージ"
    ElseIf txtISI.Text = "" Then
        MsgBox  "刺激間間隔が入力されていません", vbQuestion, "エラーメッセージ"
    ElseIf optVision(0).Value = False And optVision(1).Value = False Then
        MsgBox "視力が選択されていません", vbQuestion, "エラーメッセージ"
    ElseIf optNumber.Value = False Then
        MsgBox  "刺激セットが選択されていません", vbQuestion, "エラーメッセージ"
    ElseIf txtMemorySetSize.Text = "" Then
        MsgBox "記憶セットの大きさが入力されていません．", vbQuestion, "エラーメッセージ"
    Else
    '実験データをグローバル変数に格納
        Subject.strName = txtName.Text

        If optSex(0).Value = True Then
            Subject.strSex = "男性"
        Else
            Subject.strSex = "女性"
        End If

        Subject.intAge = Val(txtAge.Text)                '年齢
        Subject.intExposure = Val(txtExposure.Text)   '呈示時間
        Subject.intISI = Val(txtISI.Text)                '刺激間間隔
        Subject.intMemorySetSize = Val(txtMemorySetSize)  '記憶セットの大きさ

        If optVision(0).Value = True Then
            Subject.strVision = "裸眼"
        Else
```

```
            Subject.strVision = "矯正"
        End If

        If optNumber.Value = True Then
            Subject.strStimulusSet = "number.txt" '刺激セット（数字）
        End If

        If optRkey1.Value = True Then
            Subject.blnRkey = True
        ElseIf optRkey2.Value = True Then
            Subject.blnRkey = False
        End If

        Unload frmExpData
    End If

End Sub
```

【リスト２　frmFileInfo.frm】「短期記憶」のプログラムリスト（刺激の作成）

```
Option Explicit

Private Sub Form_Load()
    Dim strStim() As String '刺激セット数に対応させた動的配列

    Label1(1).Caption = Subject.strStimulusSet

    'ファイルをオープンする
    Call FileOpen

    '刺激セットを読み込む
    Dim strStimulus As String
    Do While Not EOF(1)
        Input #1, strStimulus
        List1.AddItem strStimulus
    Loop

    'リスト（刺激セット）数を調べ，配列に格納
    Dim i As Integer, j As Integer, k As Integer
    Dim intLcount As Integer 'リスト数

    intLcount = List1.ListCount
    ReDim strStim(List1.ListCount) As String

    For i = 1 To intLcount
        strStim(i) = List1.List(i - 1)
    Next i

    'ランダムに選出した10項目（最大10項目）を作成する
    Dim intRnd1 As Integer 'Rnd関数の戻り値１
    Dim intRnd2 As Integer 'Rnd関数の戻り値２
    Dim strStrtemp As String 'シャッフル中の文字を一時的に保持する格納用
    Dim intMemorySettemp As Integer '記憶セットを一時的に保持する格納用
    Dim intTestItemtemp As Integer 'テスト項目を一時的に保持する格納用
    Dim lngShuffleTime As Long 'シャッフル時間の設定
    Dim intUpper As Integer '刺激個数の上限
    Dim intLower As Integer '刺激個数の下限

    '時間の測定用
    Dim lLastTime As Long
    Dim lNowTime As Long
    Dim lPastTime As Long
```

```
lngShuffleTime = 5
intUpper = intLcount
intLower = 1

'96*intMemorySetSize×10文字(k)の刺激を作成する
For j = 1 To intTrial
    '刺激項目のシャッフル開始
    Randomize    '乱数ジェネレータの初期化

    'ループ前の時間を取得
    lLastTime = timeGetTime()
    lPastTime = 0

    Do While lPastTime < lngShuffleTime
        'ループ中の時間を取得
        lNowTime = timeGetTime()
        'ループ時間から経過時間を算出
        lPastTime = lNowTime - lLastTime

        intRnd1 = Int((intUpper - intLower + 1) * Rnd + intLower)
        intRnd2 = Int((intUpper - intLower + 1) * Rnd + intLower)
        strStrtemp = strStim(intRnd1)
        strStim(intRnd1) = strStim(intRnd2)
        strStim(intRnd2) = strStrtemp

        'イベント処理
        DoEvents
    Loop
    '1試行分（10項目）の刺激を格納
    For k = 1 To 10
        strS(j, k) = strStim(k)        'ここで，パブリック変数
    Next k
Next j

'96*intMemorySetSize(j)試行の記憶セットを割り当てる
For j = 1 To intTrial
    intMemorySet(j) = ((j + Subject.intMemorySetSize - 1) Mod Subject.intMemorySetSize) + 1
Next j

'テスト項目が記名項目と同じ0か否か1
For j = 1 To 48 * Subject.intMemorySetSize
    intTestItem(j) = 0
    intTestItem(j + 48 * Subject.intMemorySetSize) = 1
Next j
    '刺激項目のシャッフル開始
```

```
        Randomize    '乱数ジェネレータの初期化

        'ループ前の時間を取得
        lLastTime = timeGetTime()
        lPastTime = 0
        intUpper = intTrial '刺激の上限を変更
        Do While lPastTime < lngShuffleTime
            'ループ中の時間を取得
            lNowTime = timeGetTime()
            'ループ時間から経過時間を算出
            lPastTime = lNowTime - lLastTime

            intRnd1 = Int((intUpper - intLower + 1) * Rnd + intLower)
            intRnd2 = Int((intUpper - intLower + 1) * Rnd + intLower)
            '刺激セットのシャッフル
            intMemorySettemp = intMemorySet(intRnd1)
            intMemorySet(intRnd1) = intMemorySet(intRnd2)
            intMemorySet(intRnd2) = intMemorySettemp
            'テスト項目のシャッフル
            intTestItemtemp = intTestItem(intRnd1)
            intTestItem(intRnd1) = intTestItem(intRnd2)
            intTestItem(intRnd2) = intTestItemtemp
            'イベント処理
            DoEvents
        Loop

End Sub
```

【リスト３　DX.bas（標準モジュール）】「短期記憶」のプログラムリスト（DirectX関連）

```
'Option Explicit

' ═══════════════════════════════════════════════
'DirectXオブジェクト作成
' ═══════════════════════════════════════════════
Public objDX As New DirectX7

' -----------------------------------------------
'DirectDraw
' -----------------------------------------------

'DirectX7オブジェクトを作成
Public objDD As DirectDraw7

'プライマリサーフェス
Public objDDSPrimary As DirectDrawSurface7

'セカンダリサーフェス
Public objDDSSecondary As DirectDrawSurface7

'256色モードなので，正しい色で表示するためのパレット
Public objDDPalette As DirectDrawPalette

' -----------------------------------------------
'DirectInput
' -----------------------------------------------

'DirectInputオブジェクト
Public objDI As DirectInput

'DirectInputDeviceオブジェクト（キーボード）
Public objDIDeviceKeyboard As DirectInputDevice

'DirectInputDeviceオブジェクト（マウス）
Public objDIDeviceMouse As DirectInputDevice

'キーボードの状態を取得するためのユーザ定義型配列
Public keyboardState As DIKEYBOARDSTATE

'マウスの状態を取得するためのユーザ定義型配列
Public mouseState As DIMOUSESTATE
```

```
' ═══════════════════════════════════════
'DirectDraw初期化
' ═══════════════════════════════════════
Public Sub InitDirectDraw()

'DirectDrawオブジェクトを作成
    Set objDD = objDX.DirectDrawCreate("")

'DirectDraw協調レベルを設定
    Call objDD.SetCooperativeLevel(frmMain.hWnd, DDSCL_EXCLUSIVE Or DDSCL_FULLSCREEN)

'ディスプレイモードを変更（ここのリフレッシュレートを変更すること）
    Call objDD.SetDisplayMode(640, 480, 8, 0, DDSDM_DEFAULT)

'プライマリサーフェスを作成
    Dim ddsd As DDSURFACEDESC2

    'プライマリサーフェスを作成するためのサーフェス情報を設定
    With ddsd
        .lFlags = DDSD_CAPS Or DDSD_BACKBUFFERCOUNT
        .ddsCaps.lCaps = DDSCAPS_PRIMARYSURFACE Or DDSCAPS_FLIP Or DDSCAPS_COMPLEX
        .lBackBufferCount = 1
    End With

    'プライマリサーフェスを作成
    Set objDDSPrimary = objDD.CreateSurface(ddsd)

'セカンダリサーフェスを取得
    Dim ddcaps As DDSCAPS2

    'セカンダリサーフェスを作成するためのサーフェス情報を設定
    ddcaps.lCaps = DDSCAPS_BACKBUFFER

    'セカンダリサーフェスを取得
    Set objDDSSecondary = objDDSPrimary.GetAttachedSurface(ddcaps)

End Sub

' ═══════════════════════════════════════
'DirectInput初期化
' ═══════════════════════════════════════
Public Sub InitDirectInput()

'DirectInputオブジェクトの作成
    Set objDI = objDX.DirectInputCreate()
```

```
' ------------------------------------------------------
'キーボードデバイス
' ------------------------------------------------------

'キーボードデバイスを作成
    Set objDIDeviceKeyboard = objDI.CreateDevice("GUID_SysKeyboard")

'キーボードデバイスのデータフォーマットを設定
    Call objDIDeviceKeyboard.SetCommonDataFormat(DIFORMAT_KEYBOARD)

'キーボードデバイスの協調レベルを設定
    Call objDIDeviceKeyboard.SetCooperativeLevel(frmMain.hWnd, DISCL_BACKGROUND Or DISCL_NONEXCLUSIVE)

    'バッファのサイズを設定する
    Dim diProp As DIPROPLONG

    With diProp
        .lSize = Len(diProp)
        .lHow = DIPH_DEVICE
        .lData = 64
        .lObj = DIPROP_BUFFERSIZE
    End With

    Call objDIDeviceKeyboard.SetProperty("DIPROP_BUFFERSIZE", diProp)

'キーボードへのアクセス権を得る
    Call objDIDeviceKeyboard.Acquire

' ------------------------------------------------------
'マウスデバイス
' ------------------------------------------------------

'マウスデバイスを作成
    Set objDIDeviceMouse = objDI.CreateDevice("GUID_SysMouse")

'マウスデバイスのデータフォーマットを設定
    Call objDIDeviceMouse.SetCommonDataFormat(DIFORMAT_MOUSE)

'マウスデバイスの協調レベルを設定
    Call objDIDeviceMouse.SetCooperativeLevel(frmMain.hWnd, DISCL_BACKGROUND Or DISCL_NONEXCLUSIVE)

'マウスデバイスへのアクセス権を得る
    Call objDIDeviceMouse.Acquire

End Sub
```

```
' ═══════════════════════════════════════════════
'DirectX終了処理
' ═══════════════════════════════════════════════
Public Sub TerminateDX()
'ディスプレイモードを復元
    Call objDD.RestoreDisplayMode

'協調レベルを復元
    Call objDD.SetCooperativeLevel(frmMain.hWnd, DDSCL_NORMAL)

    DoEvents

' -----------------------------------------------
'DirectDrawオブジェクトの解放
' -----------------------------------------------
    Set objDDPalette = Nothing
    Set objDDSSecondary = Nothing
    Set objDDSPrimary = Nothing
    Set objDD = Nothing

' -----------------------------------------------
'デバイスのアクセス権を解放（DirectInput）
' -----------------------------------------------
    Call objDIDeviceKeyboard.Unacquire
    Call objDIDeviceMouse.Unacquire

    Set objDIDeviceKeyboard = Nothing
    Set objDIDeviceMouse = Nothing
    Set objDI = Nothing

'DirectXオブジェクトの解放
    Set objDX = Nothing

End Sub
```

【リスト４　MemoryScanning.bas（標準モジュール）】「短期記憶」のプログラムリスト（実験手続き）

```
Option Explicit

' ------------------------------------------------------
'Windows API
' ------------------------------------------------------

'マルチメディアタイマ
Public Declare Function timeGetTime Lib "winmm.dll" () As Long

'Beep音
Private Declare Function Beep Lib "kernel32" (ByVal dwFreq As Long, ByVal dwDuration As Long) As Long

'マウスカーソル
Private Declare Function ShowCursor Lib "user32" (ByVal bShow As Long) As Long

' ------------------------------------------------------
'グローバル変数
' ------------------------------------------------------

'被験者データ配列
Public Type SubjectData
    strName As String
    intAge As Integer
    strSex As String
    strVision As String
    intExposure As Integer
    intISI As Integer
    strStimulusSet As String
    intMemorySetSize As Integer
    blnRkey As Boolean
End Type
Public Subject As SubjectData

Public Const intTrial = 384 '96 * Subject.intMemorySetSize ' 記憶セット１あたり96試行＊記憶セット数
Public lngBGcolor As Long           '背景色

Public strS(intTrial, 10) As String    '刺激格納用（０－９）
Public intMemorySet(intTrial) As Integer '記憶セット格納用（1-intMemorySetSize）
Public intTestItem(intTrial) As Integer 'テスト項目格納用（０：Yes　１：No）
Public intTestItem2 As Integer 'テスト項目乱数格納用（１－記憶セット）
```

```
Public intKeys As Integer       'キー押し反応用
Public lReactionTime As Long '反応時間格納用

Public iTrial As Integer '試行ループ用

Public bExitLoopInst As Boolean          '教示用
Public bExitLoopPushkey As Boolean          'キー押し判定用

Sub Main()
    Dim i As Integer, j As Integer   'ループ用変数
    lngBGcolor = 255 '背景色
    '被験者データの入力画面表示
    frmExpData.Show 1

    '刺激の作成（frmFileInfoをロードすることでフォームを見せることなくプログラム実行
    Load frmFileInfo
    Unload frmFileInfo

    '各種条件の設定
        'DirectDraw初期化
        Call InitDirectDraw

        'DirectInput初期化
        Call InitDirectInput

        Load frmMain

    '試行数，休憩，実験刺激の保存用ファイルオープン
    '被験者データの記録
    Call SaveSubject


    '刺激系列の描画（i試行，j項目）
    For iTrial = 1 To intTrial
        '試行の開始のみ
        If iTrial = 1 Then
            'マウスカーソル消去
            ShowCursor 0

            '条件の表示と教示
            Call Instruction

        End If
        '正解のときの乱数決定
        intTestItem2 = PositionT(iTrial)
```

```
'マウスをクリックする（キーを押す）とスタート
    Call Blank
    Call Pushkey

'刺激系列の呈示
'凝視点の呈示（刺激系列の最初であれば表示）
    Call Fixation

'凝視点の呈示時間
    Wait（1000）

'ブランクの呈示
    Call Blank

'ブランクの呈示時間（要確認）
    Wait（Subject.intISI）

'記銘フェーズ
For j = 1 To intMemorySet(iTrial)

    '文字の描画
    Call Update(iTrial，j)

    '文字の呈示時間
    Wait（Subject.intExposure）

    'ブランクの呈示時間
    Call Blank

    'ブランクの呈示時間
    Wait（Subject.intISI）

Next j

'テストフェーズ
'凝視点の呈示
    Call Fixation

    Wait（1000）

    Call PhaseTests(iTrial)

'刺激系列，反応時間の保存
Call SaveItems(iTrial)
```

```
        '試行間間隔（データ保存）
        Wait (500)

    Next iTrial

    '実験終了時刻の記録とファイルのクローズ
    Call FileClose

    'マウスカーソル表示
    ShowCursor 1

    'DirectX終了処理
    Call TerminateDX
    Unload frmMain
    End
End Sub

' ═══════════════════════════════════════
'入出力用ファイルオープン
' ═══════════════════════════════════════

Public Sub FileOpen()

    '刺激セット読み込み用
    Dim fname1 As String
    If Right$(App.Path, 1) <> "¥" Then
        fname1 = App.Path & "¥" & Subject.strStimulusSet
    Else
        fname1 = App.Path & Subject.strStimulusSet
    End If
    Open fname1 For Input As #1

    '実験データ保存用
    Dim fname2 As String
    If Right$(App.Path, 1) <> "¥" Then
        fname2 = App.Path & "¥MemoryScanningData.txt"
    Else
        fname2 = App.Path & "MemoryScanningData.txt"
    End If
    Open fname2 For Append As #2

End Sub
```

```
' ═══════════════════════════════════════
'入出力用ファイルクローズ
' ═══════════════════════════════════════

Public Sub FileClose()
    Dim i As Integer

    '保存用ファイルに実験終了時刻を記録
    Write #2, Format(Now, "終了時間yyyy年m月d日hh時nn分ss秒")

    Close #1, #2

End Sub

' ═══════════════════════════════════════
'被験者データの記録
' ═══════════════════════════════════════

Public Sub SaveSubject()

    '実験開始時刻＆被験者データの記録
    Write  #2, Format(Now, "開始時間yyyy年m月d日hh時nn分ss秒"), Subject.strName,
    Subject.intAge, Subject.strSex, Subject.strVision, Subject.intExposure, Subject.intISI,
    Subject.intMemorySetSize

End Sub

' ═══════════════════════════════════════
'刺激系列の記録
' ═══════════════════════════════════════

Public Sub SaveItems(i As Integer)
    Dim j As Integer

    '被験者データの記録
     'Write  #2, Subject.strName, Subject.intAge, Subject.strSex, Subject.strVision,
    Subject.intExposure, Subject.intISI, Subject.strStimulusSet, "第" & i & "試行",

    '刺激系列の記録
    For j = 1 To 10 'intMemorySet(i)
        Write #2, strS(i, j),
    Next j

    Write #2, intMemorySet(i), intTestItem(i), intKeys, lReactionTime
```

```
End Sub

' ═══════════════════════════════════════
'凝視点の描画
' ═══════════════════════════════════════
Public Sub Fixation()
    Dim rcRect As RECT
    With rcRect
        .Top = 0
        .Left = 0
        .Right = 640
        .Bottom = 480
    End With

'セカンダリサーフェスを白で塗りつぶす
    Call objDDSSecondary.BltColorFill(rcRect, lngBGcolor)

'描画フォント
    objDDSSecondary.SetFont frmMain.Font
    objDDSSecondary.SetForeColor (RGB(0, 0, 0))

    objDDSSecondary.DrawText 290, 210, "＋", False

'サーフェスをフリップする
    Call objDDSPrimary.Flip(objDDSSecondary, DDFLIP_WAIT)

End Sub

' ═══════════════════════════════════════
'ブランクの描画
' ═══════════════════════════════════════
Public Sub Blank()
    Dim rcRect As RECT
    With rcRect
        .Top = 0
        .Left = 0
        .Right = 640
        .Bottom = 480
    End With

'セカンダリサーフェスを白で塗りつぶす
    Call objDDSSecondary.BltColorFill(rcRect, lngBGcolor)

'サーフェスをフリップする
    Call objDDSPrimary.Flip(objDDSSecondary, DDFLIP_WAIT)
```

```
End Sub

' ═══════════════════════════════════════
'文字の描画
' ═══════════════════════════════════════
Public Sub Update(i As Integer, j As Integer)
    Dim rcRect As RECT
    With rcRect
        .Top = 0
        .Left = 0
        .Right = 640
        .Bottom = 480
    End With

'セカンダリサーフェスを白で塗りつぶす
    Call objDDSSecondary.BltColorFill(rcRect, lngBGcolor)

'描画フォント
        objDDSSecondary.SetFont frmMain.Font
        objDDSSecondary.SetForeColor (RGB(0, 0, 0))

        objDDSSecondary.DrawText 290, 210, strS(i, j), False

        'サーフェスをフリップする
        Call objDDSPrimary.Flip(objDDSSecondary, DDFLIP_WAIT)

End Sub

' ═══════════════════════════════════════
'文字の描画
' ═══════════════════════════════════════
Public Sub Update2(i As Integer)
    Dim rcRect As RECT
    With rcRect
        .Top = 0
        .Left = 0
        .Right = 640
        .Bottom = 480
    End With

'セカンダリサーフェスを白で塗りつぶす
    Call objDDSSecondary.BltColorFill(rcRect, lngBGcolor)

'描画フォント
        objDDSSecondary.SetFont frmMain.Font
```

```
        objDDSSecondary.SetForeColor (RGB(0, 0, 0))

        If intTestItem(i) = 0 Then
            'Yesの場合
            objDDSSecondary.DrawText 290, 210, strS(i, intTestItem2), False
        Else
            'Noの場合
            objDDSSecondary.DrawText 290, 210, strS(i, 10), False
        End If
        'サーフェスをフリップする
        Call objDDSPrimary.Flip(objDDSSecondary, DDFLIP_WAIT)

End Sub

' ═══════════════════════════════════════
'テスト項目表示用（１－intMemorySet項目の範囲）の
'決定用関数
' ═══════════════════════════════════════
Public Function PositionT(i As Integer) As Integer
    '乱数系列の初期化
    Randomize

    '乱数の取得（１－intMemorySet項目）
    PositionT = Int(intMemorySet(i) * Rnd + 1)

End Function

' ═══════════════════════════════════════
'時間待ち用関数（引数は10ms以上で指定）
' ═══════════════════════════════════════
Public Function Wait(n As Long) As Long
    '垂直回帰の判定（リフレッシュレート100Hz固定，10ms以上５ms単位で使用）
    Dim VBS As Long
    VBS = 0
    'n = n / 10          '100Hzの場合
    n = n * 60 / 1000     '60Hzの場合

    '誤差を少なくするために，ループ条件を四捨五入
    Do While VBS <= CLng(Format(2 * n, "#0"))
        If objDD.GetVerticalBlankStatus() = 0 Then
            Do While objDD.GetVerticalBlankStatus() = 0
            Loop
        Else
            Do While objDD.GetVerticalBlankStatus() <> 0
            Loop
```

```
        End If
        VBS = VBS + 1
    DoEvents      'これを入れないと画面がちらつく
    Loop

End Function

' ═══════════════════════════════════════
'教示
' ═══════════════════════════════════════
Public Sub Instruction()

    Dim rcRect As RECT
    With rcRect
        .Top = 0
        .Left = 0
        .Right = 0
        .Bottom = 0
    End With

    Call objDDSSecondary.BltColorFill(rcRect, lngBGcolor)

    '描画フォント
    frmMain.FontSize = 20
    objDDSSecondary.SetFont frmMain.Font
    objDDSSecondary.SetForeColor (RGB(0, 0, 0))

    objDDSSecondary.DrawText 50, 200, "これから実験を開始します", False
    objDDSSecondary.DrawText 200, 250, "スペースを押してください.", False
    'サーフェスをフリップする
    Call objDDSPrimary.Flip(objDDSSecondary, DDFLIP_WAIT)

    'キーを押すと先に進む
    bExitLoopInst = False
    Do While Not bExitLoopInst

        '押されているキーの状態を取得
        Call objDIDeviceKeyboard.GetDeviceStateKeyboard(keyboardState)

        '押されているマウスの状態を取得
        Call objDIDeviceMouse.GetDeviceStateMouse(mouseState)

        'マウス右ボタン（またはスペースキー）が押された場合
        If mouseState.buttons(1) Or keyboardState.Key(DIK_SPACE) <> 0 Then
            'Beep音
```

```
            Beep 650，50
            Wait（500）
            'ループを抜ける
            bExitLoopInst = True

        ElseIf keyboardState.Key(DIK_ESCAPE) <> 0 Then

            'プログラム終了
            Call FileClose
            Call TerminateDX
            Unload frmMain
            End

        End If
    Loop
    frmMain.FontSize = 36

End Sub

' ═══════════════════════════════════════
'キー押し待ち
' ═══════════════════════════════════════
Public Sub Pushkey()

    Dim rcRect As RECT
    With rcRect
        .Top = 0
        .Left = 0
        .Right = 0
        .Bottom = 0
    End With

    Call objDDSSecondary.BltColorFill(rcRect，lngBGcolor)

    '描画フォント
    frmMain.FontSize = 20
    objDDSSecondary.SetFont frmMain.Font
    objDDSSecondary.SetForeColor（RGB(0，0，0)）

    objDDSSecondary.DrawText 100，200，"第" & iTrial & "試行" & "（" & intTrial & "）"，False
    objDDSSecondary.DrawText 100，250，"スペースキーを押してください．"，False
    'サーフェスをフリップする
    Call objDDSPrimary.Flip(objDDSSecondary，DDFLIP_WAIT)

    'キーを押すと先に進む
```

```
    bExitLoopPushkey = False
    Do While Not bExitLoopPushkey

        '押されているキーの状態を取得
        Call objDIDeviceKeyboard.GetDeviceStateKeyboard(keyboardState)

        '押されているマウスの状態を取得
        Call objDIDeviceMouse.GetDeviceStateMouse(mouseState)

        'マウス右ボタン（またはスペースキー）が押された場合
        If keyboardState.Key(DIK_SPACE) Or mouseState.buttons(1) <> 0 Then
            'Beep音
            Beep 650, 50
            Wait (500)
            'ループを抜ける
            bExitLoopPushkey = True

        ElseIf keyboardState.Key(DIK_ESCAPE) <> 0 Then

            'プログラム終了
            Call FileClose
            Call TerminateDX
            Unload frmMain
            End

        End If
    Loop
    frmMain.FontSize = 36

End Sub

' ========================================
'テストフェーズ
' ========================================
Public Sub PhaseTests(i As Integer)


    'テスト項目の呈示
    Call Update2(i)

    '反応時間の測定
    '時間測定用
    Dim lLastTime As Long
    Dim lPastTime As Long
```

```
lLastTime = timeGetTime()
'ループ開始
bExitLoopPushkey = False
Do While Not bExitLoopPushkey

    'キーボードの状態を格納しているバッファを取得し，バッファをクリアする
    Dim didod(64) As DIDEVICEOBJECTDATA
    Call objDIDeviceKeyboard.GetDeviceData(didod, DIGDD_DEFAULT)

    '配列の要素を調べ，押されている場合はリストに追加する
    Dim j As Integer
    For j = 0 To 64
        'Jキーが押されたとき
        If didod(j).lOfs = DIK_J Then
            If didod(j).lData = 128 Then
                '経過時間取得
                lReactionTime = didod(j).lTimeStamp - lLastTime
                '尚早反応対策
                If lReactionTime > 0 Then
                    '経過時間取得
                    lReactionTime = didod(j).lTimeStamp - lLastTime
                    'キー情報取得
                    If Subject.blnRkey = True Then
                        intKeys = 0 'Yes反応
                    Else
                        intKeys = 1 'No反応
                    End If
                    'ループを抜ける
                    bExitLoopPushkey = True
                End If
            End If
        End If
        'Fキーが押されたとき
        If didod(j).lOfs = DIK_F Then
            If didod(j).lData = 128 Then
                '経過時間取得
                lReactionTime = didod(j).lTimeStamp - lLastTime
                If lReactionTime > 0 Then
                    '経過時間取得
                    lReactionTime = didod(j).lTimeStamp - lLastTime
                    'キー情報取得
                    If Subject.blnRkey = True Then
                        intKeys = 1
                    Else
                        intKeys = 0
```

```
                        End If
                        'ループを抜ける
                        bExitLoopPushkey = True
                    End If
                End If
            End If
        Next j
        DoEvents
        'キーボードバッファを解放（これをしないとループ中のキー押しをリアルタイムに取得できない）
        Erase didod
    Loop

End Sub
```

高 松 大 学 紀 要

第 41 号

平成16年２月25日　印刷
平成16年２月28日　発行

編集発行

高　　松　　大　　学
高 松 短 期 大 学
〒761-0194　高松市春日町960番地
TEL（087）841－3255
FAX（087）841－3064