Windows による画像の短時間提示：SharpDX と Direct3D 9 の利用[1)]

星　野　祐　司

Displaying Images Tachistoscopically Using Windows Computers: An Application of SharpDX and Direct3D 9

Yuji Hoshino

Abstract

A C# program for tachistoscopic presentation was tested. It was constructed by using SharpDX（Mutel, 2014）and includes Direct3D 9 managed codes for the processes to display images on a monitor connected with a personal computer. Results of the tests showed that the processes of the program can be synchronized with vertical blanking intervals of raster scanning. Functional differences between the full screen mode and the windowed mode were discussed in terms of psychological experiments.

パーソナル・コンピュータ（以下、パソコン）と周辺機器を用いれば、心理学実験で用いられる刺激（文字列や画像）を提示することや、実験参加者の反応を記録することが可能である。そのため、パソコンが普及して以来、実験刺激の提示や実験参加者の反応の取得を自動的に行う装置としてパソコンを利用する方法が追及されてきた。たとえば、パソコンを使って基礎的な心理学実験を行う方法については、最近のものであれば、久本・関口（2011）、兵藤・須藤（2012）、北村・坂本（2004）、水野（2004）、中鹿・佐伯・桑原（2011）、酒井・松下・松本（2007）などがある。

この論文では Microsoft Windows をオペレーティング・システム（OS）とするパソコンを使って画像の短時間提示を行うプログラムを作成し、心理学実験を行うパソコン環境について検討を行った。パソコンが普及し始めたころは MS-DOS のように比較的単純な機能の OS が利用され、Basic や C あるいはアセンブリなどのコンピュータ言語を使って機種固有の機能を利用する心理学実験プログラムが作成されていた。現在では、CPU の処理速度や記憶装置の容量が向上していると同時に、パソコンを制御する OS や利用できる周辺機器やコンピュータ言語が大きく変化し、複雑になっている。今日では、コンピュータ言語に習熟していなくても心理学の実験プログラムを作成できるアプリケーション・ソフトウェアが充実している。たとえば、Windows 環境であれば、DirectRT（Empirisoft, 2014）、E-Prime（Psychology Software, 2014）、Inquisit（Millisecond Software、n.d.）、Presentation（Neurobehavioral Systems, 2014）、PsychoPy（Peirce, 2014）、SuperLab（Cedrus Corporation, 2014）などが存在する（ソフトウェアの比較については、Stahl, 2006）。しかし、Visual Basic

や C# などのコンピュータ言語を用いて心理学実験用のプログラムを作成することは現在も行われている。

パソコンに接続しているモニターを心理学実験の刺激提示装置として利用する場合には、パソコンの画像データをモニターに転送する処理について考慮する必要がある。パソコン内で処理された画像データは、ラスタースキャン処理よってモニター画面上に表示される。ラスタースキャンでは水平走査と垂直走査を組み合わせて 1 コマの画像情報が転送される。1 秒間にモニター上の画像が更新される回数はリフレッシュレートと呼ばれ、市販されている液晶ディスプレイでは、ほとんどの場合、60 Hz に設定されている。コンピュータ・ゲーム用に開発された液晶ディスプレイでは 120 Hz 以上のリフレッシュレートで表示できる。ラスタースキャンの走査線が画面最下部の水平線を描画してから画面最上部の水平線を描画し始めるまでを垂直帰線期間と呼ぶ。この期間にはモニター画面上への描画が行われない。画像を短時間提示する心理学実験では垂直帰線期間と画像提示プログラムの進行を同期させることが必要になることがある。同期が不十分であれば、プログラムの進行と画面更新の同期が不十分であると実験刺激の一部あるいは全体が表示されない可能性や、刺激の提示時間についての設定、あるいは刺激提示から実験参加者の反応までの経過時間の計測が不正確になる可能性がある。Microsoft Windows が搭載されているパソコンでは、垂直同期を保ちながら瞬間提示を行うプログラムを作成するために DirectX（Microsoft, 2014a）が利用されてきた（Forster, 2014; Forster & Forster, 2003; 星野, 2001; Xie, Yang, & He, 2005）。

DirectX は Windows 用ソフトウェア作成のために Microsoft が開発したライブラリー集であり、コンピュータ・ゲームなどで利用される 3 次元データを含んだ画像情報の高速処理や入力装置の制御などを行うためのプログラム集である。この DirectX を利用すれば、垂直同期を保ちながら画像を表示するプログラムを作成することが可能である。最近の Windows（Windows Aero 以降）では画像処理機能を強化するために、パソコン側のグラフィックカードを制御するためのソフトウェアであるディスプレイ・ドライバーの仕様（WDDM: Windows Display Driver Model）が更新されている（Microsoft, 2006）。DirectX を利用したプログラムでは Windows のディスプレイ・ドライバーを経由して画像処理が行われるので、ディスプレイ・ドライバーの機能強化は心理学実験用プログラム作成にとってよい影響を及ぼすことが期待できる。

Microsoft Windows に限らず最近の OS では、いくつかのアプリケーションを異なるウィンドウに表示させて複数の作業を効率的に行えるように設計されている。しかし、パソコンの処理を 1 つのアプリケーションに集中させることにはいくつかの利点がある。たとえば、画面全体を使ってコンピュータ・ゲームの表示が行われていれば、操作者が臨場感を持つことが期待できる。また、コンピュータの資源を 1 つのアプリケーションに集中させることでより高速な処理が可能になる。このような特性は心理学実験にとっても望ましい。DirectX にはモニターの全画面を利用する表示モードがある。DirectX による全画面モードでは 1 つのアプリケーションが 1 つのモニターに関する処理をほぼ占有できるので、処理の安定化と高速化が期待できる。また、DirectX による全画面モードを使用すれば、垂直同期を保ちながら画像表示を行うことが可能である。最近、Microsoft Windows に新しいディスプレイ・ドライバーの仕様が取り入れられ、バックバッファーを利用したウィンドウズ・デスクトップの描画が採用されている（Microsoft, 2006）。DirectX によるウィンドウモードでの画像提示がどの程度確実に行われるのかについて本論文で検証した。

Microsoft DirectX を用いたソフトウェア開発では、DirectX SDK（software development kit）あ

るいは Windows SDK と Microsoft Visual Studio、そしてプログラミング言語の C++ を組み合わせて利用することが推奨されている。しかし、C++ の習得は必ずしも容易ではなく、より簡便な方法が求められる。Windows 環境でのソフトウェア開発では、習得が容易とされるプログラミング言語である Visual Basic あるいは C# がしばしば利用されている。Visual Basic あるいは C# から DirectX を利用できるように、DirectX のライブラリーに修正を加えた Managed DirectX が Microsoft あるいは有志らによって開発されている（たとえば、DirectX と Visual Basic を用いたプログラム作成については、大槻, 2007; 山崎･exilis, 1999）。SlimDX（SlimDX Group, 2009）と SharpDX（Mutel, 2014）は、C# から Managed DirectX を利用するために有志が開発したライブラリーである。本論文ではこれらの中で、最近開発された SharpDX を利用してプログラムを作成した。

DirectX は 1995 年に公表された後に改訂が行われ 2009 年には Microsoft から DirectX 11 が公開されているが、本論文では 2002 年に公開された DirectX 9 を用いてプログラムを開発した。画像をモニター上に提示するプログラムの作成が本論文の目的であったため、最新の機能を必要としなかった。DirectX 9 に関する情報はインターネット上（たとえば、Microsoft, 2014b）、あるいは著作（たとえば、Miller, 2004）などで入手可能であり、画像表示を行うプログラムの作成が比較的容易である。本論文では、DirectX 9 を用いて画像提示プログラムを作成し、プログラムがモニター画面の垂直同期を監視しながら画像を正確なタイミングで確実に提示できるかどうかを検討した。

画像提示プログラムの作成

Microsoft が無料で提供しているプログラム開発用のソフトウェアである Visual Studio Express 2013 for Windows Desktop を用いて画像提示プログラムを作成した。作成したプログラムのソースコードは http://www.psy.ritsumei.ac.jp/hoshino/dex/ からダウンロード可能である。使用したコンピュータ言語は C# である。プログラム開発では .NET Framework 2.0 と SharpDX 2.6 を参照した。SharpDX は、DirectX を利用したプログラムを C# を用いて作成するためのライブラリーであり、無料で配布されている。作成したプログラムは 1 つのモニター画面を占有して画像表示を行うため、2 つのモニターが接続しているパソコンを用いてプログラム開発を行った。

プログラムの基本構成

作成したプログラムは Windows のデスクトップ環境で動作する。プログラムは 3 つの画像を、1 つずつ、位置をずらしながら連続的にモニター画面に表示する。このプログラムによる処理は 3 つの段階に分けられる。第 1 の段階では各種の設定を行うための Windows フォームが表示される（図 1）。このフォームを用いて、画像の提示時間や、プログラムの経過時間を保存するためのファイル名などを設定する。次の段階ではもう 1 つのフォームが表示される。この段階で DirectX による画像表示処理が準備される。最後の段階で DirectX による画像表示が行われる。

プログラムを起動すると、最初に、図 1 に示すフォームが表示される。このフォームを使って、プログラムによる画像提示に関するいくつかの設定を行う。マウスを使用するか、それともキーボードのスペースキーを使うか。DirectX による全画面モードを用いるか、それともウィンドウモードにするか。プログラムの画像提示処理をモニターの垂直帰線期間に同期させるか。プログラムの進

行とモニター画面の更新を同期させる場合、各画像の提示時間はモニター画面の垂直帰線期間を何回待つ設定にするのか。3 つの画像を提示する処理の経過時間を保存するか。保存する場合には、ファイル名の入力が求められる。OK ボタンをクリックすると、次のフォームが表示される。

図 1　各種設定を行うフォーム。

2 番目のフォームの表示とともに、DirectX による画像表示を準備する処理が実行される。このフォーム上に OK ボタンがあり、このボタンをマウスでクリックするか、スペースキーを押すことで DirectX による画像表示が始まる。設定のためのフォームが表示されたモニターを用いて画像が表示される。画像表示が終了すると、再び 2 番目のフォームが表示される。

Direct3D による画像表示

Direct3D は DirectX に含まれる画像処理のためのライブラリーである。Direct3D には 3 次元画像を描画する機能（たとえば、物体の回転をシミュレーションする機能）が数多く含まれているが、作成したプログラムでは画像データを画像用のメモリーに転送する機能を利用した。Direct3D による画像表示では、モニターに表示する画像情報を保持するために 2 つ以上のバッファーを確保できる。1 つのフロントバッファーと 1 つのバックバッファーを持つ構成がもっとも単純である。フロントバッファーは必ず 1 つで、このバッファーに登録されている情報がモニター画面上に表示されている。Direct3D は、まず、バックバッファーに画像情報を描画する。次にフロントバッファーとモニターの関係を切り離し、バックバッファーが新たなフロントバッファーになる。そして、これまでのフロントバッファーはバックバッファーになり、新たな情報を書き込むことが可能になる。このようにフロントバッファーとバックバッファーを切り替えることで、モニター画面の表示が更新される。

作成したプログラムでは DirectX 9 に含まれる Direct3D 9 を利用した。プログラムが Direct3D 9 による画像処理を行うためには、まず、基本的な設定を行うための初期化処理が必要である。Direct3D による画像表示に関する諸特性を設定するために、PresentParameters クラスの変数に値を代入する必要がある。クラスはプログラミング用語であり、一種の型である。ここでは変数名を pParams とする。

```
PresentParameters pParams = new PresentParameters {
    Windowed = false,
    BackBufferCount = 1,
    BackBufferFormat = current.Format,
    BackBufferWidth = current.Width,
    BackBufferHeight = current.Height,
    FullScreenRefreshRateInHz = current.RefreshRate,
    PresentationInterval = PresentInterval.One,
    SwapEffect = SwapEffect.Discard };                                    (1)
```

文 1 は C# によるプログラムの記述例である。なお、本論文で示されるプログラムの記述例はすべて C# を用いた。文 1 に示されている Windowed プロパティは、Direct3D が Windows フォームを利用するならば true に、全画面モードを使うならば false に設定する。したがって、文 1 は全画面モードを行うための設定例である。続いて、画像メモリーに設定される描画用のバックバッファーの数、描画書式、画像の幅と高さ、リフレッシュレートなどが設定される。文 1 に含まれる current 変数には Direct3D が使用するモニターの情報が登録されている。たとえば、current.RefreshRate にはモニター画面のリフレッシュレートが代入されている。PresentationInterval は、Direct3D による画像情報の更新でモニター画面の垂直同期を待つかどうかを設定するための値が代入される。文 1 では、モニターの垂直帰線期間を 1 回待ってから、画面に表示される画像情報を更新するように設定している。SwapEffect はフロントバッファーを切り替えるときの処理を指定するために用いる。文 1 ではもっとも単純な処理（SwapEffect.Discard）を設定している。

```
Device d3d = new Device (new Direct3D (), iD, DeviceType.Hardware,
    form1.Handle, CreateFlags.SoftwareVertexProcessing, pParams);         (2)
```

文 2 では pParams を使って Direct3D が初期化される。d3d は Device クラスの変数名である。Device クラスは Direct3D の諸機能と表示装置を結びつけるためのクラスである。iD はモニターを示す変数名であり、モニターが 2 つパソコンに接続されていれば、0 か 1 のどちらかが代入されている。

Direct3D 9 には、2 次元の画像データを処理する機能を持つ Sprite クラスが存在する。Sprite クラスの Draw 機能を使って画像データをバックバッファーへ転送させることができる。DirectX を用いた表示画面の更新は、2 枚の画用紙を交互に見せる動作にたとえることができる。一方の画用紙を見せている間にもう一枚の画用紙を手元に置き絵を描く。絵を描き終わったらその画用紙を見せ、今まで見せていた画用紙を手元に置き、新たな絵を描く。バックバッファーへの描画が完了すると、モニター画面に表示する画像メモリーの領域を切り替え、今までフロントバッファーであった領域がバックバッファーになる。また、今までバックバッファーだった領域がフロントバッファーになる。Direct3D 9 を用いた場合、バックバッファーを更新し、フロントバッファーとバックバッファーを切り替える一連の処理は Device クラスの Present 機能によって行われる。Present 機能の動作は文 1 に示した変数 pParams によって設定されている。

瞬間的な画像提示を行う実験では、画像を欠けることなく、設定した提示時間どおりに確実に表示することが必要である。画像情報の更新時に垂直帰線期間を待たない設定では、画像処理とラスタースキャンの進行が同期していないため、画像が表示されない、あるいは一部しか表示されない可能性がある。また、モニター画面の垂直同期が 1 周期終了する間に 2 回以上の画像情報の更新が

行われる可能性がある。

心理学実験用プログラムでは垂直同期についてもう 1 点検討すべき点がある。Present 機能が垂直同期を保ちながら画像情報を更新していても、モニター画面の更新と実験用プログラムの進行が同期するとは限らない点である。なぜならば、Present 機能による画像情報の更新と Present 機能を呼び出したプログラムは並列的に処理されるからである。心理学実験用プログラムでは、プログラムの処理をモニター画面の更新に合わせながら進行させることが求められる状況がある。たとえば、画面を提示した直後から実験参加者による反応までの経過時間を測定する場合、あるいは、提示後に行う処理までの経過時間を設定する場合である。

モニターの更新に合わせながらプログラムを進行させることは、ラスタースキャンの状態を監視すれば可能である。

```
do {
    RasterStatus rs = d3d.GetRasterStatus(0);
} while(!rs.InVBlank);                                    (3)
d3d.Present();                                            (4)
do {
    RasterStatus rs = d3d.GetRasterStatus(0);
} while(rs.InVBlank);                                     (5)
```

文 3 では垂直帰線期間になるまで do ループを繰り返えし、文 4 では present 機能を用いて画面が更新される。文 5 では、垂直帰線期間であれば do ループを繰り返えし、プログラムを遅延させている。このようにしてプログラムの進行とモニター画面の更新を同期させることができる。一方、文 3 と 5 を省き、文 4 の Present 機能のみを実行すれば、モニター画面の更新を待たずにプログラムの処理が進むことになる。

経過時間の測定

経過時間の測定には .NET Framework の StopWatch クラスを用いた。StopWatch クラスの ElapsedMilliseconds 機能を用いれば、1 ms 単位で処理間の経過時間を計測した。たとえば、文 3 の直前に計測開始の文を挿入し、文 5 の直後に経過時間を計測するための文を挿入すれば、文 3 の処理が開始され、文 5 の処理が終了するまでの経過時間が計測できる。画像提示にかかわる処理の経過時間を知ることで、モニター画面の更新とプログラムの進行が同期されているかどうかを推測することができる。たとえば、モニター画面の更新に要する時間はリフレッシュレートに依存するため、画面更新に要する時間よりプログラムの進行に要した時間が短ければプログラムの進行と画面更新は同期されていないことになる。

キーボードとマウスの入力検出

心理学実験では、実験参加者による反応時間を記録するためにキーボードやマウスの操作をできるだけ即時的に検出することが必要になる場合がある。DirectX には DirectInput と呼ばれるキーボードやマウスからの入力を処理するライブラリーが含まれている。本論文で作成したプログラムでは Windows API（application programming interface）に含まれる GetAsynkKeyState 関数を利用した。Windows API は Microsoft Windows の機能をアプリケーション・ソフトウェアが利用するた

めに用意されている。GetAsynkKeyState 関数を利用すれば、キーボードとマウスからの入力を即時的に検出することが可能である。

while（GetAsyncKeyState（Keys.Space）>= 0）; （6）

while（GetAsyncKeyState（Keys.LButton）>= 0）; （7）

文 6 はスペースキーの押下を、文 7 はマウスの左ボタンがクリックされたことを検出する。作成したプログラムでは DirectX を使って画像を表示する直前にメッセージを画面に表示し、スペースキーが押されるか、またはマウスがクリックされると画像の提示を開始した。

原因がはっきりしないのであるが、作成したプログラムではマウスの入力待ち状態がしばらく続くと問題が発生した。DirectX によるウィンドウモードで、マウスのクリックを待っている状態が一定時間続くと（おそらく、6 秒以上）、画面の更新が 100 ms ほど遅れるか、更新されても画像が表示されないことがあった。マウス待ちのために文 8 の while 文の繰り返しが続くと、プログラムが“応答なし”の状態になってしまうようであった。全画面モードではそのような状態にならなかった。ウィンドウモードでの処理では、マウスのクリックを待つ while ループを別スレッドにすることで問題は解決した。

画像

作成したプログラムが提示した画像を図 2 に示す。縦と横が、それぞれ、1,000 ピクセルと 333 ピクセルの 1 つの画像を、図 2 に示した 2 本の縦線にそって 3 分割した画像を用いた。各画像は横が 111 ピクセルであった。プログラムは 3 つの画像を 1 つずつ連続的に表示した。その際、2 番目と 3 番目の画像の表示位置をそれぞれ 111 ピクセルずつ横にずらした。すなわち、2 番目の画像は 1 番目の画像の右隣に、重ならないように、かつ隙間がないように表示された。同様に、3 番目の画像は 2 番目の右横に表示された。

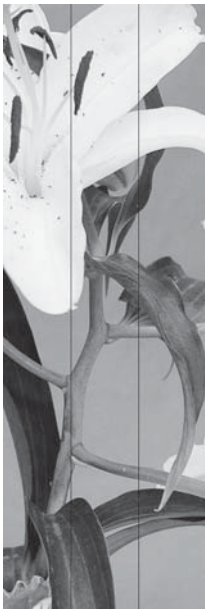

図 2　モニター画面に表示した画像。1 つの画像を縦に 3 分割して、画像を 3 つ作成した。分割箇所を示すために画像中に縦線を挿入した。

装置

作成したプログラムを検証するために 2 台のパソコンを使用した。1 台目は Microsoft Windows 8.1 Pro をインストールした Dell OptiPlex 9010 である（パソコン 1）。CPU は Intel Core i5-3570 3.4GHz、グラフィックカードは AMD Radeon HD 7470 であった。接続されている液晶ディスプレイは Dell 1707FP と Dell 170S であった。もう 1 台は Windows 7 SP1 Professional をインストールした Dell OptiPlex 990 である（パソコン 2）。Microsoft DirectX end-user runtimes（June 2010）をインストールして使用した。CPU は Intel Core i7-2600 3.4 GHz であった。グラフィックカードは ATI Radeon HD 5450 が 2 台挿入されていて、それぞれに液晶ディスプレイ Dell P2314H と BenQ XL2411Z）を接続した。液晶ディスプレイとパソコンは DVI ケーブルを使って接続した。どちらのパソコンにもキーボードとマウスが USB 接続されていた。

プログラムの処理と画像表示の時間的ずれを確認するために、プログラムによる画像提示処理の完了と同時に外部の LED を点灯させた。外部 LED はマイクロコントローラ（Arduino Uno R3）を使用して制御した（Arduino については、D'Ausilio, 2012; 牧野, 2012）。Arduino とパソコンは USB ケーブルを介して接続し、パソコンから Arduino に命令を送信した。LED を接続した Arduino にあらかじめ制御プログラムを読み込ませ、パソコンからの命令によって LED が点灯するようにした。

モニター画面の更新状況や LED の発光を確認するために、1 インチ型の CMOS センサーを搭載するデジタルカメラ（Nikon 1 V2）を使用した。Nikon 1 V2 はスローモーション動画を撮影できるので、1 秒間に 400 コマ（400 fps）の設定でモニター画面を動画撮影した。また、1 秒間あたり 60 コマの設定で静止画を連写することも可能なので、画像を表示しているモニター画面を連続撮影した。

瞬間提示の検証

全画面モード

全画面モードを使用して、プログラムが画像を提示する処理に要した時間を計測した。作成したプログラムは 3 つの画像を連続的に 1 つずつ表示する。Direct3D の Present 機能による画像情報の更新は、文 1 に示す変数 pParams の値によってその動作が設定される。文 1 に示すように、モニターの垂直同期に合わせて画像データを更新する設定でプログラムを実行した。プログラムが各画像を提示する期間は垂直同期 1 周期分の長さ（1 フレーム）に設定した。1 つの画像提示が終了すると、空白画面を挿入せずに次の画像が提示された。検証で用いた液晶ディスプレイのリフレッシュレートが 60 Hz であったので、1 周期は 16.7 ms である。計測は、パソコンがネットワークや周辺機器に接続されたままの状態で行った。また、モニター画面をデジタルカメラで撮影して、画像が設定どおりに表示されるかどうかを確認した。

モニターの画面更新に合わせてプログラムが進行するように設定した場合（文 3、4、5 が順に処理される場合）、ひとつの画像についての処理を完了してから次の画像処理を完了するまでの経過時間は、各画像を 1 フレームごとに提示する設定であれば、16.7 ms（1/60 秒）になるはずである。Windows 8.1 を OS とするパソコン 1 を用いて、各画像の処理に要する時間を 30 回計測した結果を表 1 に示す。プログラムの進行をモニターの画面更新に合わせるように設定して計測した結果は、予想（1/60 秒）と一致した。また、最小値と最大値の差が最大 1 ms であることからわかるように処理の経過時

間は安定していた。この計測結果は、設定どおりに、プログラムの進行がモニター画面の更新と同期していたことを示している。

表１　パソコン１による全画面モードでの計測結果

画像	平均	最小	最大	標準偏差
プログラムの処理を画面の更新と同期させる場合				
1番目	16.9	16	17	0.3
2番目	16.0	16	16	0.0
3番目	17.0	17	17	0.0
1・2・3	49.9	49	50	0.3
プログラムの処理を画面の更新と同期させない場合				
1番目	0.5	0	1	0.5
2番目	1.5	1	2	0.5
3番目	6.7	0	16	5.4
1・2・3	8.8	2	18	5.4

注：各画像の処理に要する時間（ms）を 30 回計測した。“1・2・3” はすべての画像を処理するために要する時間を示す。各画像を 1 フレームごとに表示する設定であった。

プログラムの進行と画面の更新の同期を行わない設定にして（文 3 と 5 を省略し、文 4 のみを実行する）、3 つの画像のそれぞれの処理に要する時間を計測した結果を表 1 の下段に示す。各画像を 1 フレーム提示する設定であったが、各画像の処理に要する時間は 0 ms から 16 ms まで変動した。1 回の画面更新に要する時間は 16.7 ms なので、多くの場合、画像表示が完了する前にプログラムが次の画像に関する処理を行っていることがわかる。この結果は、モニター画面を更新する処理と、プログラムによる画像提示の処理が並列的に行われていることを示している。

表２　パソコン２による全画面モードでの計測結果

画像	平均	最小	最大	標準偏差
プログラムの処理を画面の更新と同期させる場合				
1番目	17.0	17	17	0.0
2番目	16.4	16	17	0.5
3番目	16.7	17	17	0.5
1・2・3	50.4	50	51	0.3
プログラムの処理を画面の更新と同期させない場合				
1番目	2.5	1	4	0.6
2番目	0.9	0	2	0.4
3番目	10.5	1	24	6.8
1・2・3	13.9	4	28	6.8

注：各画像の処理に要する時間（ms）を 30 回計測した。“1・2・3” はすべての画像を処理するために要する時間を示す。各画像を 1 フレームごとに表示する設定であった。

表 2 には、Windows 7 SP1 を組み込んだパソコン 2 による計測結果を示す。モニターの垂直同期に合わせてプログラムが進行するように設定した場合には表 1 に示したパソコン 1 の測定結果と大きな違いは見られなかった。プログラムの進行が画面の更新と同期しない設定にした場合、パソコン 1 の処理と比べて全体的にパソコン 2 の処理が遅くなっていた。

画像のすべてが欠けることなく設定どおりに表示されることを確認するために、デジタルカメラを用いて 400 fps の動画撮影を行った。モニター画面のリフレッシュレートが 60 Hz であったので、400 fps の動画撮影を行うと、モニター画面の 1 コマは動画の 6.7 コマに相当する。撮影した動画の各コマを、ビデオ編集ソフトを使用して表示させてモニター画面の変化を確認した。その結果、プログラムの進行をモニター画面の垂直同期に合わせるか、合わせないかにかかわらずプログラムが提示した画像は 1 フレームごとに画像の最上部から最下部までモニター画面に表示されることがパソコン 1 と 2 で確認できた。プログラムの処理は文 3 と 5 を用いることでモニターの画面更新に合わせた進行になるが、画面に表示する画像データの更新は Direct3D の Present 機能（文 4）に依存する。Present 機能の画像処理と作成したプログラムの処理は並列的に実行されるため、プログラムの進行をモニターの垂直同期に合わせるかどうかにかかわらず、画像は画面上で 1 フレームごとに更新されて表示される。

モニター画面をデジタルカメラで連続撮影した画像を図 3 に示す。図 3 には時間的に連続した 6 画像が示されている。撮影ではコンピュータ 2 に接続した液晶ディスプレイ（BenQ XL2411Z）を用いた。BenQ XL2411Z の設定は、リフレッシュレートが 60 Hz、Blur Reduction がオフ、AMA がオフ、Instant Mode がオンであった。デジタルカメラによる撮影では、1 秒間に 60 コマの静止画を連写するように設定した。また、各画像の露光時間を 1/250 秒に設定した。プログラムによる画像提示では、2 番目の画像が 1 番目の画像の、モニター画面に向かって右側に描画され（図 3C）、3 番目の画像が 2 番目の画像の右側に描画される（図 3D）というように提示が右方向に推移する。また、モニター画面の更新では、ラスタースキャンが画面の一番下に達すると画面の最上部へ移動して画像データを描画する。カメラのシャッター機能によって撮影範囲が制限されている場合、撮影範囲がモニター画面の上部から下部、あるいはモニターに向かって画面の右側から左側に移動すると更新された画面の一部が撮影されない可能性がある。なぜならば、カメラの撮影範囲が移動した後の画面更新は撮影されないからである。そのためカメラのグリップが上になるように縦置きすることで、カメラの撮影範囲がモニター画面の左側から右側に動くようにした。図 3B、C、D では画像の端が水平にならず、斜めになっている。これはカメラのシャッターによる撮影範囲が画像の右側に移動するため、画像の右側部分は左側部分と比べてより最近の状態（垂直走査線が下に移動した状態）が撮影されるためである。

図 3B では 1 番目の画像が表示され始め、図 3C では 1 番目の画像の下半分と 2 番目の画像の上半分が、図 3D で 3 番目の画像が途中まで表示されている。さらに、図 3E では 3 番目の画像の下半分が表示されて、図 3F では画像提示が完了し何も表示されていない。したがって、設定どおり、各画像は 1 フレーム（1/60 秒）ごとに表示されていることがわかる。図 3B では、1 番目の画像と画面に表示した教示（“ スペースキーを押してください。”）のほぼ左半分が重なって表示されている。また、図 3C では 2 番目の画像と 3 番目の画像が、図 3D では 3 番目の画像と 4 番目の画像が同時に表示されている部分がある。これらの現象は、液晶の応答の遅れによる影響と考えられる。

ウィンドウモード

文 1 で示した変数 pParams の Windowed 部分を true に設定すると、全画面モードではなくウィンドウモードを用いた表示を行う。つまり、Direct3D による画像表示がデスクトップ内の 1 区画に制限される。経過時間の測定ではプログラムが各画像を 1 フレームごとに提示するように設定した。

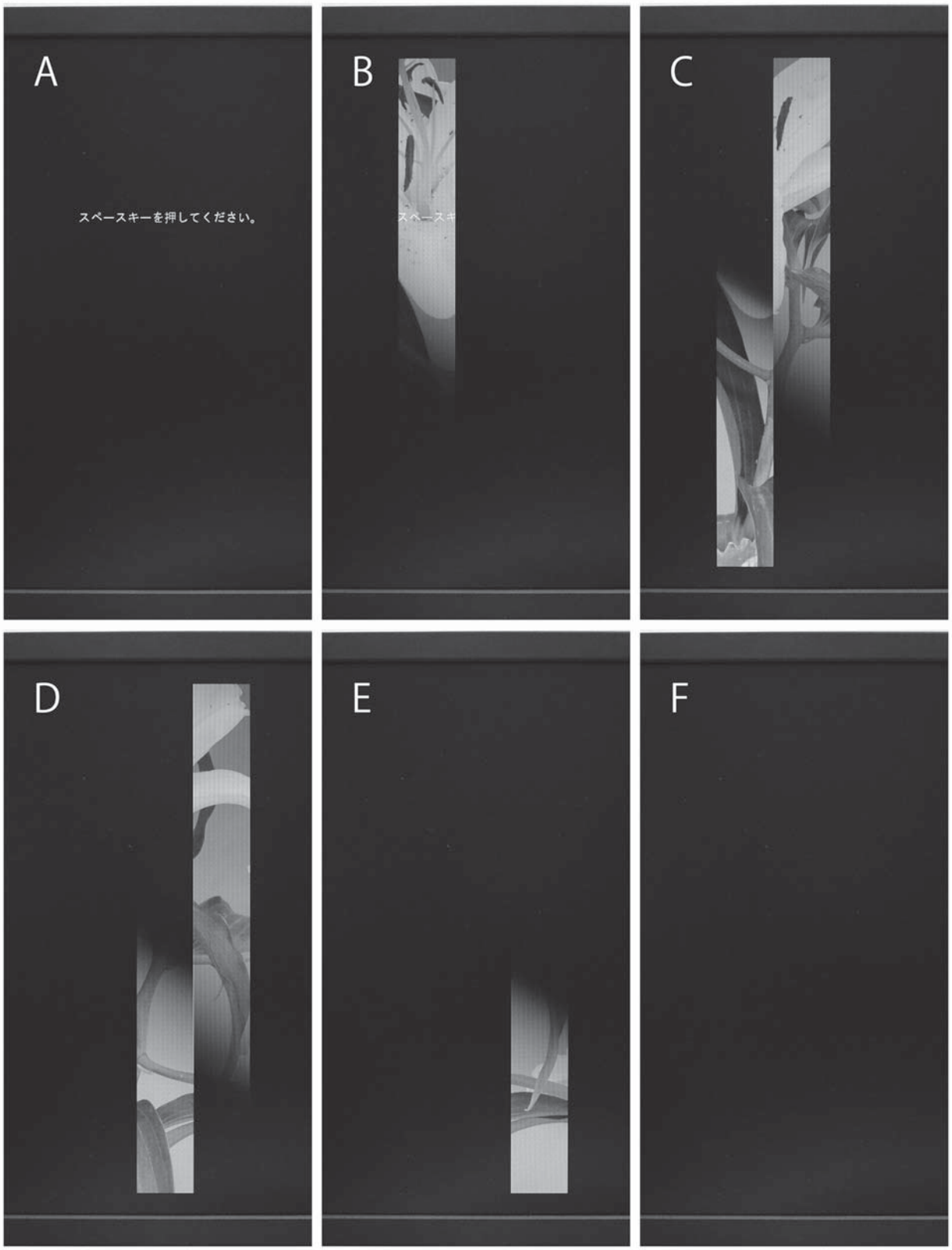

図 3　モニター画面に 3 枚の画像を連続表示する経過を、デジタルカメラを用いて、1 秒当たり 60 コマの設定で連写した。各画像を 1 フレームごとに表示する設定であった。モニター（BenQ XL2411Z）のリフレッシュレートは 60 Hz であった。カメラを縦置き（グリップ部分を上）にして撮影した。各画像の撮影時露光時間は 1/250 秒であった。

パソコン 1 を用いて、ウィンドウモードの設定で計測した結果を表 3 に示す。プログラムが画像提示に要する時間は、プログラムの進行をモニターの画面更新に合わせる設定では全画面モードとほぼ同じ結果であった。デジタルカメラを使って、モニター画面の更新を 400 fps の設定で動画撮影し、各画像が 1 フレームごとに表示されることを確認した。全画面モードと同じように、ウィンドウモードにおいてプログラムがラスタースキャンの情報を入手でき、設定どおりに画像が更新されるのは Windows で使用されているディスプレイ・ドライバーの仕様改訂が反映されているためと考えられる。

プログラムの進行がモニター画面の更新を待たない設定では、全画面モードでの画像提示（表 1）と比べて処理時間が長くなり、モニターの垂直同期にプログラムの進行を合わせる設定とほとんど変わらないことがわかる（表 3）。この結果から、全画面モードでの処理は Windows の他の処理より優先的に実行されていると考えられる。ウィンドウモードでは、Windows フォーム全般に関連する処理あるいはデスクトップ全体を描画する処理などが原因で遅延が発生するのであろう。

表 3　パソコン 1 におけるウィンドウモードでの計測結果

画像	平均	最小	最大	標準偏差
プログラムの処理を画面の更新と同期させる場合				
1 番目	16.7	16	17	0.5
2 番目	16.0	16	16	0.0
3 番目	17.0	17	17	0.0
1・2・3	49.7	49	50	0.5
プログラムの処理を画面の更新と同期させない場合				
1 番目	11.6	4	19	4.2
2 番目	16.7	16	17	0.5
3 番目	16.4	16	17	0.5
1・2・3	44.7	37	52	4.3

注：各画像の処理に要する時間（ms）を 30 回計測した。“1・2・3” はすべての画像を処理するために要する時間を示す。各画像を 1 フレームごとに表示する設定であった。

パソコン 2 を用いて計測した結果を表 4 に示す。プログラムの進行をモニター画面の垂直同期に合わせる場合であっても、合わせない場合であっても、3 つの画像を提示し終えるまでに 50 ms 以上の経過時間を示す場合があった。このことは画像の提示時間が垂直同期の 1 周期よりも長くなっている可能性を示す。3 つの画像が表示される経過を撮影した動画を分析した結果、画像を 1 フレームごとに提示する設定であるにもかかわらず、画像の提示が 2 フレームに達する場合があることを確認した。作成したプログラムの各処理に要する時間を検討したところ、Present 機能が遅延を引き起こしていることがわかった。すなわち、Present 機能が垂直帰線期間を 1 回余計に待っていることになる。Present 機能の内部処理による遅延であるので、作成したプログラムの変更で画面更新の遅延を修正することは困難である。

パソコン 2 の場合、ウィンドウモードでは 1 フレームごとの画像提示にしばしば失敗することが示された。画像の提示時間を垂直同期の 2 周期に設定して計測を行うと設定どおりに表示されていたが、画像の短時間提示や厳密な時間設定を必要とする心理学実験ではウィンドウモードを使用しないほうが適切であろう。

表 4　パソコン 2 におけるウィンドウモードでの計測結果

画像	平均	最小	最大	標準偏差
プログラムの処理を画面の更新と同期させる場合				
1 番目	24.0	17	38	10.1
2 番目	21.6	16	62	9.6
3 番目	18.2	16	33	3.2
1・2・3	63.8	50	133	21.5
プログラムの処理を画面の更新と同期させない場合				
1 番目	19.3	6	40	10.7
2 番目	19.2	16	30	4.5
3 番目	17.4	16	20	1.1
1・2・3	55.8	39	88	15.7

注：各画像の処理に要する時間（ms）を 30 回計測した。“1・2・3” はすべての画像を処理するために要する時間を示す。各画像を 1 フレームごとに表示する設定であった。

液晶ディスプレイ

図 3 の画像からわかるように、液晶ディスプレイでは次のフレームが描画されるまで直前に表示された画像の表示状態が維持される。図 3 の画像撮影で使用した液晶ディスプレイ（BenQ XL2411Z）はコンピュータ・ゲームでの利用を想定した機器であり、早い応答速度を持つ液晶が使用されている（液晶ディスプレイの応答性全般については、Simmons, 2014）。そのため、図 3B、C、D からわかるように、2 つの画像が同時に表示されている部分は比較的狭い範囲であった。

パソコン 2 に接続されていた Dell P2314H は液晶の応答が遅いタイプの液晶ディスプレイである。図 4 に、この液晶ディスプレイを用いて、3 番目の画像が描画されている状態を 1/250 秒の露光時間で撮影した画像を示す（図 3D に相当）。図 4 から、この液晶ディスプレイでは、3 番目の画像を表示する期間に 2 番目の画像が表示されたままになっていることがわかる。モニター画面上の同じ場所に異なる画像が連続提示されると、2 つの画像のそれぞれの明るさに依存して 2 つの画像が重なって表示されると考えられる。液晶ディスプレイは今日一般的に利用される表示装置であるが、表示を終了した直前の画像が現在の画像表示に影響を及ぼすため、1 フレームごとに異なる画像を提示させることは、厳密に考えると難しいといえよう。

ゲーム用の液晶ディスプレイ（たとえば、BenQ XL2411Z）のように液晶の応答が十分早ければ、画像を 1 フレーム提示した直後に 1 フレーム以上の空白画面の提示を行うことで、直前の画像の影響を受けずに画像を連続的に提示することは可能であると考えられる。BenQ XL2411Z の場合、グラフィックカードが対応可能であれば 144 Hz のリフレッシュレートで提示できる。したがって、画像の提示間に空白画面を挿入しても 13.9 ms ごとに画像を提示することが可能である。

液晶ディスプレイによっては、オーバードライブと呼ばれる処理によって液晶の応答性を高めている。この機能を用いると、画像の短時間提示では画像の明るさおよび色彩の再現性に問題が生じる可能性がある。短時間であるが、画像情報に忠実ではない輝度の諧調で画像が表示される可能性があるためである（詳しくは、林, 2005）。本論文の検証実験では、オーバードライブを無効にするために BenQ XL2411Z の AMA 機能をオフにして使用した。AMA 機能をオンに設定すると提示した画像のコントラストが強めに表示されることを、デジタルカメラでモニター画面を撮影して確認した。

図 4　モニター画面に 3 番目の画像が表示されている状態を撮影した。モニター（Dell P2314H）のリフレッシュレートは 60 Hz であった。画像撮影時の露光時間は 1/250 秒であった。

液晶ディスプレイ装置によっては、液晶のバックライトを制御してバックライトを消灯した期間を 1 フレーム内に挿入し、動画表示でのブレを軽減させようとしている。BenQ XL2411Z ではこの機能を Blur Reduction と呼んでいる。今回の検証実験では Blur Reduction をオフにして撮影した。

液晶ディスプレイ装置内の回路がパソコンから送られてきた画像データを処理するためには一定の時間が必要である。そのため、モニターに画像が表示されるまでに遅延が引き起こされる（EIZO, n.d.）。一部の液晶ディスプレイ装置には、液晶の応答を高速化するために、あるいは表示される画像の品質を改善するために高度な画像処理回路が組み込まれている。そのような場合、遅延が長くなり、心理学実験の実施に支障をきたす可能性がある。本論文の検証実験では、プログラムによる画像提示処理が完了した直後に外部の LED を点灯させ、モニター画面と LED をともに動画撮影することで表示の遅れを確認した。撮影ではパソコン 2 と BenQ XL2411Z を用いた。BenQ の液晶ディスプレイには入力信号と画面表示との遅延を最小にする機能（Instant Mode）があるので、これをオンに設定して撮影した。撮影された動画の分析から、ほぼ 1 フレーム（1/60 秒）に相当する遅れが確認できた。この遅れ原因が液晶ディスプレイ装置内の画像処理なのか、グラフィックカードの処理なのか、あるいは Windows のディスプレイ・ドレイバーによる処理なのかははっきりしない。

信号処理による遅延は音響処理に特殊な装置を備えているコンピュータにおいても発生する可能性がある。たとえば、Windows 8.1 がインストールされているノートパソコン（Sony Vaio Pro 11）では、3 番目の画像を提示する処理が完了した直後に鳴るように設定したビープ音（Console.Beep ()）が 200 ms ほど遅れていた。音は心理学実験で合図として利用されることがあるので、プログラムによる処理とスピーカから聞こえる音のタイミングが一致しなければ合図として機能しなくなる可能性がある。

Windows パソコンによる短時間画像提示

Windows 8.1 あるいは Windows 7 SP1 がインストールされているパソコンを用いて、画像の瞬間提示を行う心理学実験を想定したプログラムを作成し、設定どおりに画像が短時間表示されるかどうかを検証した。プログラムは Microsoft Visual Studio Express と SharpDX（Mutel, 2014）を用いて開発し、DirectX の画像処理機能を利用した。液晶ディスプレイを使用して画像を表示した。画像表示が更新される経過を確認するために、デジタルカメラを使用して液晶ディスプレイを撮影した。

DirectX による全画面モードを用いて画像提示を行った場合、プログラムの進行をモニターの画面更新に合わせることが可能であり、画像は設定どおりに表示されることを確認した。また、モニター画面の垂直同期に合わせて画面を更新する処理は高速で安定していることが確認できた。しかし、液晶ディスプレイの応答速度は十分でなく、提示時間を 1 周期（16.7 ms）に設定しても、1 周期以上の期間、画像が表示されていることが認められた。画像を連続的に提示する場合には、提示間に空白画面を挿入する方法を検討すべきであろう。

DirectX によるウィンドウモードでの画像提示では、全画面モードと同様に、モニター画面の垂直同期に合わせた画面更新が可能であった。また、全画面モードと同様に、ウィンドウモードであってもモニター画面のラスタースキャン情報を利用してプログラムの進行を制御できた。ウィンドウモードを用いれば、通常の Windows プログラミングによるフォームの処理と、DirectX による画像提示処理の切り替えが円滑になり、画像提示と実験参加者による判断の相互作用を含む心理学実験プログラムの作成が容易になる。ただし、Windows 7 SP1 をインストールしたパソコンでは、モニターの垂直同期に合わせて 1 フレームごとに画面を更新する設定での処理が安定せず、画像が更新されるまでに 2 フレームを要することがしばしばあった。Windows 8.1 をインストールしたパソコンでは、そのような現象を確認できなかった。今回検証した 2 台のパソコンは OS 以外にも異なる要素があるので、Windows のバージョンが影響するのかどうかは明確ではない。

注

1）本研究の一部は、文部科学省私立大学戦力的研究基盤形成支援事業「インクルーシブ社会に向けた支援の＜学＝実＞連環型研究」（代表者：稲葉光行、立命館大学人間科学研究所、2013 年 4 月～ 2016 年 3 月）の補助を受けた。

文 献

Cedurs Corporation（2014）. *SuperLab Home Page*. Retrieved from http://superlab.com/

D'Ausilio, A.（2012）. Arduino: A low-cost multipurpose lab equipment. *Behavior Research Methods*, **44**, 305-313.

Empirisoft（2014）. *MediaLab and DirectRT Psychology Software and DirectIN Response Hardware*. Retrieved from http://www.empirisoft.com/

EIZO（n.d.）. フレーム遅延とスルーモード　Retrieved from EIZO ライブラリー website: http://www.eizo.co.jp/eizolibrary/videos/throughmode/index.html

Forster, J. C.（2014）. DMDX Updates Page. Retrieved from Jonathan C. Forster's stuff website: http://www.u.arizona.edu/~jforster/dmdx.htm

Forster, K. I., & Forster, J. C.（2003）. DMDX: A Windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, & Computers*, **35**, 116-124.

林利明（2005）. スペック表記に潜む落とし穴：応答速度の虚像と実像 Retrieved from EIZO ライブラリー website: http://www.eizo.co.jp/eizolibrary/other/itmedia01_03/

星野祐司（2001）. DirectX による画面表示の制御と反応の取得：Microsoft Windows を用いた心理学実験プログラムの作成　立命館文学, **570**, 1-24.

久本博行・関口理久子（2011）. やさしい Excel で心理学実験　培風館

兵藤宗吉・須藤 智（編著）（2012）. 認知心理学基礎実験入門［改訂版］　八千代出版

北村英哉・坂本正浩（2004）. パーソナル・コンピュータによる心理学実験入門：誰でもすぐにできるコンピュータ実験　ナカニシヤ出版

牧野浩二（2012）. たのしくできる Arduino 電子工作　東京電機大学出版局

Microsoft、(2006). Windows Vista ディスプレイ ドライバ モデル　Retrieved from Microsoft Developer Network website: http://msdn.microsoft.com/ja-jp/library/aa480220.aspx

Microsoft, (2014a). *DirectX Graphics and Gaming*. Retrieved from Microsoft Developer Network website: http://msdn.microsoft.com/en-us/library/windows/desktop/ee663274%28v=vs.85%29.aspx

Microsoft, (2014b). *DirectX 9.0 for Managed Code*. Retrieved from Microsoft Developer Network website: http://msdn.microsoft.com/en-us/library/windows/desktop/bb318658%28v=vs.85%29.aspx

Miller, T. (2004). *Managed DirectX 9: Graphics and Game Programming*. Indianapolis: Sams Publishing.

Millisecond Software (n.d.). *Inquisit by Millisecond Software*. Retrieved from https://www.millisecond.com/

水野りか（2004）. Web を介してできる基礎・認知心理学実験演習　ナカニシヤ出版

Mutel, A. (2014). *SharpDX - Managed DirectX*. Retrieved from http://sharpdx.org/

中鹿直樹・佐伯大輔・桑原正修（2011）. はじめての行動分析学実験―Visual Basic でまなぶ実験プログラミング　ナカニシヤ出版

Neurobehavioral Systems (2014). *Presentation: Precise, powerful stimulus delivery*. Retrieved from http://www.neurobs.com/

大槻雄一郎（2007）. 15 歳からはじめる DirecX 9 3D ゲームプログラミング教室 Visual Basic 編　ラトルズ

Pierce, J. W. (2014). *PsychoPy: Psychology software in Python*. Retrieved from http://www.psychopy.org/

Psychology Software Tools (2014). *Psychology Software Tools: E-Prime application suite for psychology experiment design, implementation, and analysis*. Retrieved Psychology Software Tools website: http://www.pstnet.com/eprime.cfm

酒井浩二・松下正修・松本寛史（2007）. 今すぐ体験！パソコンで認知心理学　ナカニシヤ出版

Simmons, A. (2014). *Factors affecting PC monitor responsiveness*. Retrieved PC Monitors website: http://pcmonitors.info/articles/factors-affecting-pc-monitor-responsiveness/

SlimDX Group (2009). *SlimDX*. Retrieved from http://slimdx.org/

Stahl, C. (2006). Software for generating psychological experiments. *Experimental Psychology*, **53**, 218-232.

Xie, S., Yang, Z., & He, J. (2005). Millisecond-accurate synchronization of visual stimulus displays for cognitive research. *Behavior Research Methods*, **37**, 373-378.

山崎由喜憲・exilis（GWP）（1999）. Visual Basic DirectX プログラミング　ソフトバンク　パブリッシング

（本学文学部教授）