Windowsアプリケーションの自動実演を用いたプレゼンテーション

大　西　孝　臣*

Presentation with the automated demonstration of Windows application software

Takaomi OHNISHI

Abstract

This article shows the technique to demonstrate Windows application software without any assistant. The author has applied the Rational Visual Test, which is essentially the testing tool for Windows application software, to the automated demonstration of Windows application software.

1．はじめに

コンピュータ関連、特にソフトウェア関連の学会や講演会などの会場のほとんどが、プレゼンテーション用の液晶プロジェクタを準備しており、講演者はOHPシートの代わりにノートPCを持参するスタイルが主流になりつつある。しかし、液晶プロジェクタを使用している講演者を見ていると、ドキュメントを提示する際は、Power Pointなどのプレゼンテーション用ソフトでの静止画によるものがほとんどであり、ソフトウェアの実演を行う際も、講演者の他にソフトウェアを操作する為の介助者を必要とするのが実状で、導入したノートPCの利便性を十分に生かしているとは言い難い。

そこで本稿では、日本ラショナルソフトウェア社製のRational Visual Test 6.0の機能を応用して、Windowsアプリケーションを随時コメントを表示させながら自動実行させる事で、人手を使わずにWindowsアプリケーションソフトの実演を含めたプレゼンテーションを行う手法を紹介する。

2．シナリオの編集

2．1　ユーザ操作の記録とファイル生成

Visual Testのユーティリティであるシナリオレコーダー[1]（図1）を使用すると、ユーザによるマウスやキーボード他への操作や、実行されたプログラムの状態(例えばウィンドゥのサイズ)などを「シナリオ」という形で記録し、BASICベースのTest言語[1]で書かれた「テストケースファイル」と呼ばれるソースファイルを生成できる。

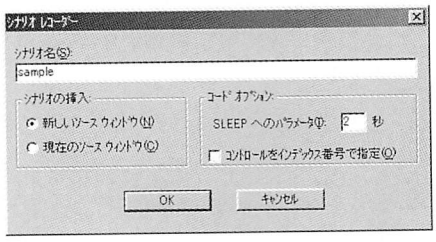

図1　シナリオレコーダーの実行開始画面

シナリオレコーダーによる、テストケースファイルのソースリストの概要を次に示す。

```
'$INCLUDE 'RECORDER.INC'
SetDefaultWaitTimeout(Timeout)
Scenario "sample"
        'Visual Test ｳｨﾝﾄﾞｳを最小化します。
        If GetHandle(GH_HWNDCLIENT) Then WMinWnd(GetHandle(GH_HWNDCLIENT))
        ' ｽｸﾘﾌﾟﾄを記録した時の解像度を選択します。
        CheckResolution (1024, 768)
... （ここでマウス・キーボードによる操作が「シナリオ」として記録される。） ...
End Scenario
```

テストケースファイルを“実行”する事で、シナリオに従ったシミュレーションが行われるが、Visual Testは本来、開発中のWindowsアプリケーションを高速テストする為のソフトウェアである為、シナリオレコーダーが生成したままでのテストケースファイルでは、人間の目に留まらない速さでシミュレーションが実行される。従って、テストケースファイルに対し、次節以降に示す適当な操作を施さなければならない。

2．2　マウス操作①（移動と項目選択）

図2に示す、「マウスカーソルの移動」→「メニューから項目[ファイル]を選択」→「マウスカー

＊　助　手　　情報工学科

ソルの移動」→「ポップアップメニューから項目[c:¥dsp¥Cos.c]を選択」の一連の操作をシナリオにする。

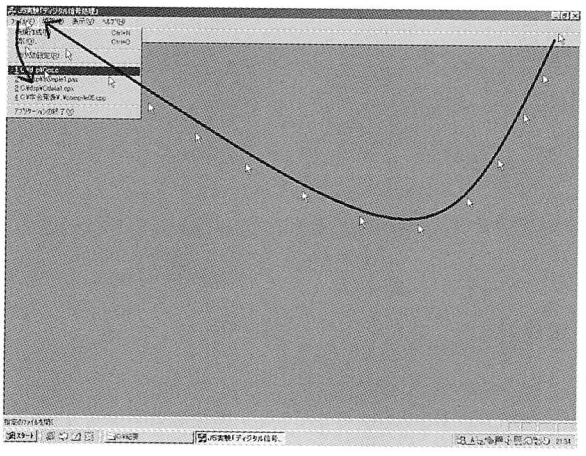

図2　マウスカーソルの移動と項目の選択

図2の操作を行うシナリオのソースファイルを次に示す。

```
'$INCLUDE 'RECORDER.INC'
SetDefaultWaitTimeout(Timeout)
Scenario "sample"
...
        dim i as single, j as single
        dim x(0 to 2) , y(0 to 2)
...
        ' マウスカーソルの移動①
        x(0) = 755
        y(0) = 35
        x(1) = 600
        y(1) = 300
        x(2) = 30
        y(2) = 25
        for i = 0.00 to 1.00 step 0.01
                j = 1.00 - i
                x_current = x(0)*j*(2.00*j-1.00) + x(1)*4.0*i*j + x(2)*i*(2.00*i-1.00)
                y_current = y(0)*j*(2.00*j-1.00) + y(1)*4.0*i*j + y(2)*i*(2.00*i-1.00)
                Play "{Moveto "+STR$(x_current)+", "+STR$(y_current)+"}"
                sleep(0.01)
        Next i
        Sleep(0.500)
        ' カレントウィンドウの設定
        CurrentWindow = MFndWndWait("Ｊ５実験「ディジタル信号処理」", FINDWINDOWFLAGS_IF, Timeout)
        Sleep(0.500)
        ' メニューからの項目選択
        WMenuSelect("ファイル(&F)")
        Sleep(2.000)
        ' マウスカーソルの移動②
        x(0) = 30
        y(0) = 25
        x(1) = 30
        y(1) = 100
        x(2) = 40
        y(2) = 112
        for i = 0.00 to 1.00 step 0.01
                j = 1.00 - i
                x_current = x(0)*j*(2.00*j-1.00) + x(1)*4.0*i*j + x(2)*i*(2.00*i-1.00)
                y_current = y(0)*j*(2.00*j-1.00) + y(1)*4.0*i*j + y(2)*i*(2.00*i-1.00)
                Play "{Moveto "+STR$(x_current)+", "+STR$(y_current)+"}"
                sleep(0.01)
        Next i
        Sleep(0.500)
        ' ポップアップメニューからの項目選択
        WMenuSelect("&1 C:¥¥dsp¥¥Cos.c")
        Sleep(2.000)
...
End Scenario
```

マウスカーソルの移動は、(x(0),y(0))、(x(1),y(1))、(x(2),y(2))の3点をそれぞれ始点、通過点、終点として、補間による座標計算を行い、Sleepステートメントによりディレイを掛けながら「Play "{Moveto …}"」ステートメントによりカーソルを移動させる。3点間の補間の手法は手頃なものであれば何でも良いと考える。上記リストではx、y各座標成分において単なる2次Lagrange補間を行ったが、数値計算法であるFEM（有限要素法）での技法である、自然座標系によるアイソパラメトリック要素での座標算出法[2)3)]にて記述した。

メニューやポップアップメニューからの項目選択はWMenuSelectステートメントを用いて行う。

2．3　マウス操作②（ドラッグ）

図3に示すWindowsアプリケーションの子ウィンドゥのサイズ変更などでの事例で必要な、マウスによるドラッグ操作をシナリオにする。

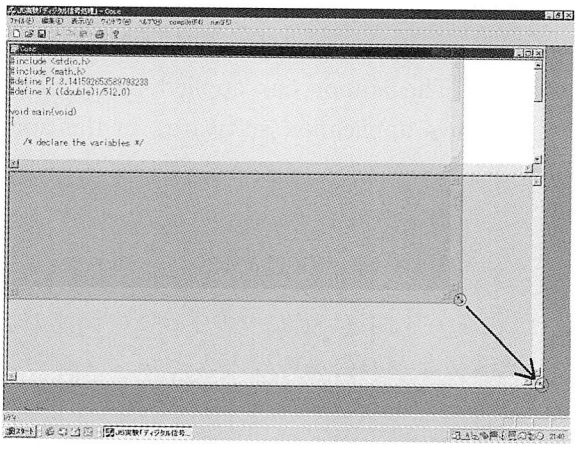

図3　ドラッグによる子ウィンドウのサイズ変更

図3の操作を行うシナリオのソースファイルを次に示す。

```
'$INCLUDE 'RECORDER.INC'
SetDefaultWaitTimeout(Timeout)
Scenario "sample2"
...
        ' マウスによるドラッグ
        x(0) = 810
        y(0) = 510
        x(1) = 900
        y(1) = 600
        x(2) = 1000
        y(2) = 700
        Play "{Btndown "+STR$(x(0))+", "+STR$(y(0))+", Left}"
        for i = 0.00 to 1.00 step 0.01
                j = 1.00 - i
                x_current = x(0)*j*(2.00*j-1.00) + x(1)*4.0*i*j + x(2)*i*(2.00*i-1.00)
                y_current = y(0)*j*(2.00*j-1.00) + y(1)*4.0*i*j + y(2)*i*(2.00*i-1.00)
                Play "{Moveto "+STR$(x_current)+", "+STR$(y_current)+"}"
                sleep(0.01)
        Next i
        Play "{Btnup "+STR$(x(2))+", "+STR$(y(2))+", Left}"
        Sleep(0.500)
...
End Scenario
```

マウスによるドラッグは、マウスカーソルの移動での始点と終点それぞれに対し、「Play "{Btndown …}"」と「Play "{Btnup …}"」の各ステートメントによりマウスのボタン操作をさせる事で実現する。

2．4　ステータスボックスによるコメント

プレゼンテーションを視聴する立場では、未知のWindowsアプリケーションを紹介される事になるので、アプリケーションをただ自動実行させ

て、口頭で説明するだけではなく、図4の様に実演の各要所にて、コメントを表示させるのが望ましい。

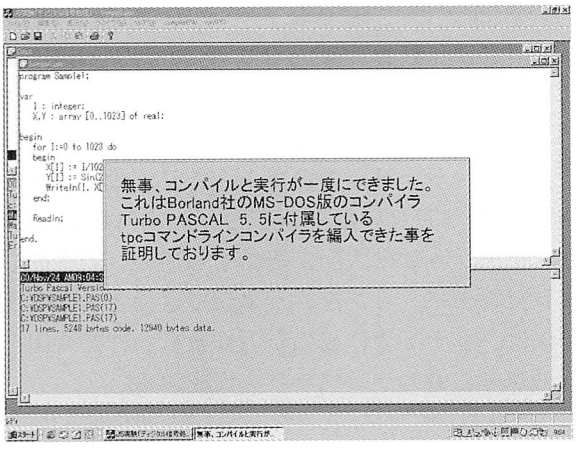

図4　ステータスボックスによるコメント

図4の操作を行うシナリオのソースファイルを次に示す。

```
'$INCLUDE 'RECORDER.INC'
SetDefaultWaitTimeout(Timeout)
Scenario "sample"
...
        ' ステータスボックスの表示
        StatusBox "無事、コンパイルと ･･･ 証明しております。",,660,230,TRUE,,"ＭＳゴシック",30
        Sleep(10,000)
        StatusBox Close
        Sleep(0.500)
...
End Scenario
```

コメントの表示は、StatusBoxステートメントを用いてステータスボックスを表示させて行う。

2．5　メッセージボックスによる待機

プレゼンテーションの最中には、一時的に講演者の口述の方に関心を惹きつけたい場合や、OHPなどの他の機器の操作に移りたい場合がある。

しかし、テストケースファイルを実行させて行うシミュレーションの最中は、マウスやキーボードなどの入力機器の操作を支配されてしまう為、シミュレーションの途中にて、シミュレーションの終了や一時停止／再開、あるいは条件によって複数のシナリオから選択させたい、等々の制御を、要求に即応する形で行う事ができない。

そこで、シミュレーション実行時での制御の方法として、図5に示す様に、シナリオにてメッセージボックスを表示させて待機させる手法が用意されている。

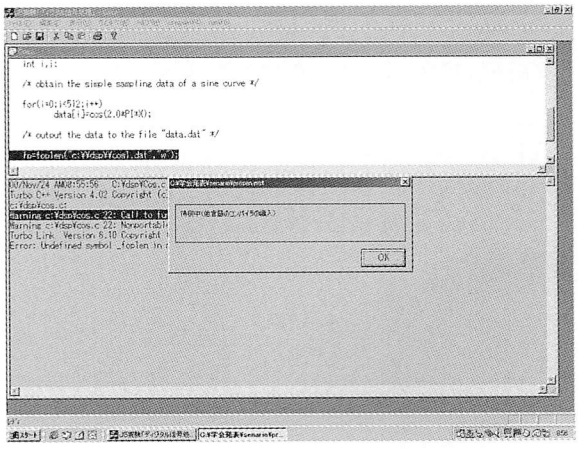

図5　メッセージボックスによる待機

図5の操作を行うシナリオのソースファイルを次に示す。

```
'$INCLUDE 'RECORDER.INC'
SetDefaultWaitTimeout(Timeout)
Scenario "sample"
...
        ' メッセージボックスによる待機
        MsgBox "待機中（他言語のコンパイラの編入）",MB_OK
        Sleep(3.000)
...
End Scenario
```

シミュレーションの待機は、MsgBoxステートメントを用いて行う。

図5の例では、ステータスボックスを終了させるボタンは「OK」のみであるが、他にも「YES」「NO」の様に2者選択を求める形式も用意されている。

3．シナリオによるシミュレーションの実行

作成したテストケースファイルを実行してシミュレーションを行うには、基本的にはVisual Testの実行画面を表示させて、メニューから辿って「実行」の項目を選択するか、ツールバーから「実行」を選択しなければならないが、Visual Testの起動時間が無駄時間になってしまう事と、メニューからの「実行」の項目への経路が深い（「ビルド」→「デバッグの開始」→「実行」）事が理由で、プレゼンテーションには向いていない。

そこで、作成したテストケースファイルのショートカットを、図6の様にデスクトップ画面にコピーして、ショートカットを右クリックする事で現れるポップアップメニューから「Run」の項

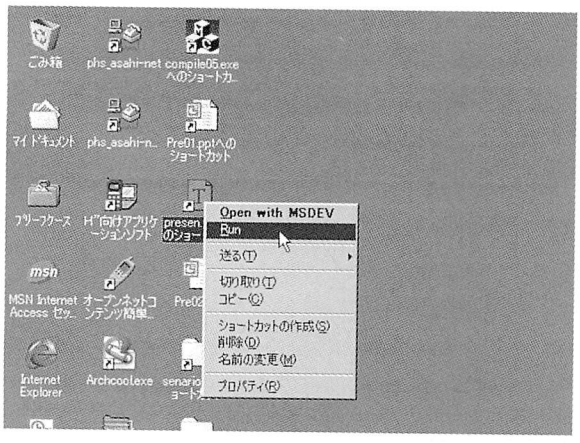

図6　デスクトップ画面からの実行

目を選択する事で、即座にシミュレーションを実行させる。

Power Pointで作成したプレゼンテーションファイルについても、ショートカットをデスクトップ画面にコピーして、右クリックさせるとポップアップメニューが現れ、「表示」の項目を選択する事により即座にスライドショーが開始されるので、段取りを決める際に、Visual Test、Power Pointそれぞれで作成したファイルのショートカットを並べる事で、デスクトップ画面上での一貫した操作によるプレゼンテーションが可能になり、講演者のストレスを軽減する事ができる。

4．シナリオ作成時に対応すべき問題点

シナリオを作成する際に、最も発生しやすい問題は、シミュレーションを行う毎にWindowsやWindowsアプリケーションの初期状態が異なる事に起因している。

例えば、シミュレーションの対象のWindowsアプリケーションがファイルを取り扱うという非常にありふれたケースにおいて、もし、1回のシミュレーションによって複数のファイルを開くならば、メニューの「ファイル」項目から開かれるポップアップメニューにある「最近使ったファイル」の順列がアプリケーションを実行する度に変わるという現象に対応しなければならない。

また、シナリオのテスト毎に、作成した分のシナリオのシミュレーションを行うだけの実時間の経過を必要とする為、ブレークポイントの使用や、シナリオにあるディレイの為のSleepステートメントをコメントアウトするなどの対応をしなければならない。

5．おわりに

Windowsアプリケーションの自動実演による、プレゼンテーションの手法を提案した。

著者は実際に研究発表の場[4]において、Power Pointでのドキュメントの提示と共に、Windowsアプリケーションの自動実演をOHPでのドキュメントの提示と同時進行させる形で試みた。結果、シナリオの形であらかじめ設定された時間管理の下で自動実演をしているWindowsアプリケーションの説明に終始する事で、効率的なプレゼンテーションを行う事ができたと考える。今後は、シナリオ作成の為の作業量、作業時間を低減させる為のライブラリを作成する。

参考文献

1）Visual Test ヘルプ、Rational Software Corporation、1996
2）K.H.Huebner著、山田　嘉昭訳、構造工学シリーズ③有限要素法、科学技術出版社、1978
3）小柴　正則、光・波動のための有限要素法の基礎、森北出版、1990
4）大西　孝臣・杉岡　一郎共著、Windowsアプリケーションにコマンドラインコンパイラを編入させた実習教材の開発、教育工学関連学協会連合第6回全国大会講演論文集、pp.821-822、2000

（平成12年11月27日受理）