辻林 涼子

株式会社ミガロ.

システム事業部 システム2課

Delphi/400で ドラッグ&ドロップを制御

ドロップ & ドラッグの操作により
Excel ファイルをシステムに取り込む方法を紹介する。

略歴
1987 年 3 月 3 日生まれ
2009 年近畿大学農学部卒
2009 年 04 月株式会社ミガロ . 入社
2009 年 04 月システム事業部配属

現在の仕事内容
入社 2 年目で、主に Delphi/400 を使用したシステムの受託開発を担当している。

1. はじめに

アプリケーションを開発する際に、設計者としては考えなければいけないことは多数あるが、必須条件として主に 2 つあると考える。

1 つは内部の設計をシンプルにし、保守性や拡張性を見越した設計にすることである。またもう 1 つは、ユーザーが満足する操作性を実現することだ。特にユーザーには、直感的な操作が分かりやすく使いやすい。

本レポートでは、直感的な操作として普段何気なく使われる Drag&Drop について、Windows での仕組みと Delphi に実装する方法を紹介したいと思う。

2. Drag&Dropとは何か

Drag&Drop とは、マウス操作の 1 つで、マウスでファイルやデータをクリックし選択したまま、移動（ドラッグ）させ別の場所でクリックしたボタンを離す（ドロップ）ことである。これは、Windows が提供している機能の 1 つである。【図 1】

この機能について大きく分類すると、以下の 3 つに分けられる。

●アイコン、フォルダへの Drag&Drop
ファイルを、別フォルダに移動やコピーできる。

●実行中のアプリケーションへの Drag &Drop
メモ帳にテキストファイルを Drag&Drop すると、ファイルを開く。

●アプリケーション内での Drag&Drop
Excel では、セルの移動ができる。

これらは Windows が、指定した Window、今回はドラッグ元のファイルとドラッグ先のファイルに対して、Drag&Drop の機能を呼び出しており、それぞれの処理が行われている。

また、受信側のアプリケーションによって、Drag&Drop を許可するものとしないものがある。これは、Delphi で実装する上でも当然必要な制御となる。Word、Excel、メモ帳などのファイルを開く機能のあるアプリケーションの多くは、Drag&Drop を制御する機能が実装されている。が､その他のアプリケーションでは、受け付けないものも見られる。

さらに、Drag&Drop を受け付けた場合でも、ファイルの種類により制限されることも多い。これは、PowerPoint のファイルをメモ帳、Excel に Drag&Drop してみると、そのことが理解できる。

メモ帳では、文字化けして読めない文字列が表示されるものの、メモ帳のテキストファイルを開くという処理としては正しい対応がなされる。【図 2-1】

これに対し Excel では、「ファイルが開けません」というエラーが出る。こちらは、Excel では PowerPoint のファイルを受け付けないよう指定されているからである。【図 2-2】

さて、今回は上記から、アイコンへの Drag&Drop、実行中のアプリケーション

への Drag&Drop、これらによる外部ファイルの取り込みについて解説していく。

3. Delphiで実現するためには

実際に Delphi で実現するためには、いくつか情報が必要である。必要となる情報とその取得方法について、アプリケーションのパターン別に説明しよう。

アイコンにドロップインしてアプリケーションを起動する場合

実際のところ、特に処理をしなくても Delphi で作成したプログラムアイコンは、ドロップインを受け付けるようになっている。しかし、受け入れ後の処理ロジックを作成していないため、何も起こらない。

何かを起こす処理ロジックを作成するには、受け入れたファイル名が必要である。コマンドライン引数を利用すれば、簡単にそのファイル名が分かる。コマンドライン引数とは、プログラム実行時に指定される引数のことで、Delphi では ParamCount と ParamStr を用いて取得できる。

コマンドプロンプトから実行する例を見ていただけると分かりやすい。【図 3】

コマンドプロンプトでは、以下の形式になっている。

アプリケーションのパス（半角スペース）開くファイルのパス

ParamCount では、このパラメータ（開くファイルのパス）の数を取得する。ParamStr では、指定した番号のパラメータを取得する。ただし、ParamStr の 0 番目はアプリケーションのパスが入っているため、番号を指定する場合には注意が必要だ。

実行中のアプリケーションにドロップインする場合

実行中アプリケーションでファイルを受け取る場合、Windows に、アプリケーションが Drag&Drop を受け入れることを伝えなければならない。この手続きを行うのが、Windows API の DragAcceptFiles 関数である。

API に WindowHandle と True（ドロップされたファイルの受け入れ許可指定）を渡すと、フォーム上で Drag&Drop の受け入れが可能となる。

WindowHandle とは、Message を送信する際に画面上のオブジェクトを識別するために、個々の要素に割り当てられる一意の番号のことである。フォームやアイコンなどの Drag&Drop の受け入れを許可するオブジェクトは、この WindowHandle で指定する。

DragAcceptFiles を許可すると、ファイルがドロップされた時、フォームに WM_DROPFILES という Windows Message が送られる。ただし、この Windows Message は、Drag&Drop が行われたことを伝えるだけで、実際のファイル名やパスは取得できない。

従って、DragQueryFile 関数を用いて、ファイルのパスを取得する必要がある。この API は、複数のファイルが渡された場合でも、1 つのファイルしか取得できないが、何番目のファイルを取得するかの指定は可能である。

また、パラメータのファイル番号部分に「$FFFFFFFF（= -1）」を指定した場合には、いくつのファイルがドロップされたのかを返す仕組みとなっている。

なお、必要なファイル名が取得できたら、最後に DragFinish 関数でメモリを開放する。これを行わなかった場合、使用可能なメモリが減っていき、Windows がフリーズする場合があるので必ず行うようにする。

4. Windows Messageとは

前章で出てきた「Windows Message」について、少し詳しく説明する。

Windows Message とは、Windows（OS）がそれぞれの Window を制御するために使っているもので、これらを Delphi で意図的に記述することにより、Windows が行っている動作を取得したり、逆に動作するようにしたりできる。

例えば、ユーザーがマウスやキーボードで入力した時、それらはまず Windows によって取得される。その後、Windows は、それぞれのアプリケーションに Windows Message として送信する。アプリケーションは、Windows Message を受け取ることによって、ユーザーがどのような入力を行ったのかを知ることができる。【図 4】

●”Hello!”の例

Delphi でイベントを記述する場合、例えば、Create イベントに”Hello!”というダイアログを表示する場合について考えてみよう。

このアプリケーションを実行した場合、まず Windows がフォームを作成し、Windows から Delphi へ WM_CREATE の Windows Message が送信される。

Delphi ではその Windows Message を受け取り、Delphi の中で Create イベントに書かれた内容を実行するよう、Windows 側に Windows Message を送信する。

すると、”Hello!”と書かれたダイアログが表示され、ダイアログ内のボタンを押すと、それがまた Windows Message で Delphi に送信され、「ダイアログを出す」という 1 文が終了する。

このように、Create イベントや Destroy イベント等の多くの Windows Message 取得は Delphi が自動で行うため、特に意識せず使用することができる。【図 5】

他方、Drag&Drop の Windows Message（WM_DROPFILES）は、対応している標準のイベントがないため、意図的に DragAcceptFiles を True に設定する必要がある。これによって、Drag&Drop の Windows Message（WM_DROPFILES）を取得することができるようになる。

5. 実装方法

実際に簡単なサンプルを作成する。

アイコンにドロップインしてアプリケーションを起動する場合

例として、ドロップしたファイルパスを表示するだけの簡単なプログラムを作成する。これは Form の OnCreate または OnShow のイベント時に、ParamStr を使用すればよい。単純に取得したパスを表示するだけのプログラム例である。

まずは、コンポーネントをフォームに追加する。

図1

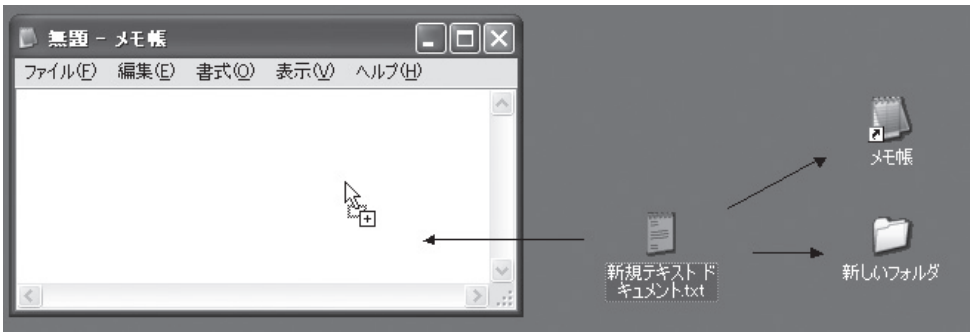

無題 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
メモ帳
新規テキスト ドキュメント.txt
新しいフォルダ

図2-1

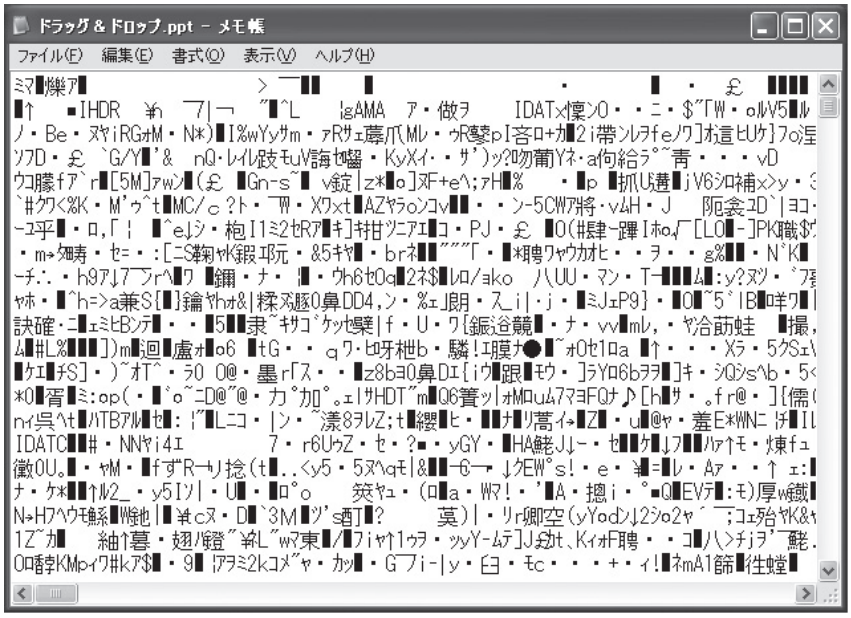
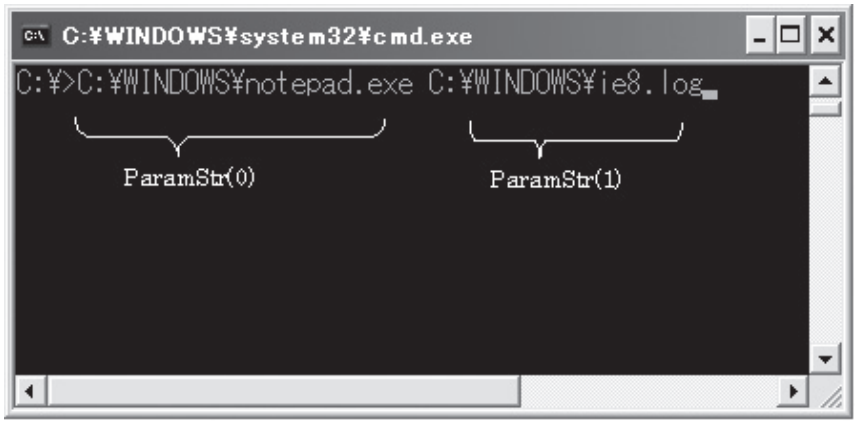

ドラッグ & ドロップ.ppt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

図2-2


Microsoft Excel
ドラッグ & ドロップ.ppt: ファイル形式が正しくありません。
OK

図3


C:¥WINDOWS¥system32¥cmd.exe
C:¥>C:¥WINDOWS¥notepad.exe C:¥WINDOWS¥ie8.log
ParamStr(0)
ParamStr(1)

【使用するコンポーネント】
・TPanel：ツールパレット（Standard）
・TMemo：ツールパレット（Standard）
・TBitBtn：ツールパレット（Additional）

次に、コンポーネントのプロパティを設定する。今回は TBitBtn の Kind を変更する程度で、他はデフォルトの設定でかまわない。

あとは、OnCreate または OnShow のイベントに、ソースを記述すれば完成だ。【ソース 1】

実行すると、画面のようになる。【図 6】

実行中のアプリケーションにドロップインする場合

例として、csv ファイルを Drag&Drop すると、先ほどのプログラムと同様に、ファイルパスを画面上の Memo に表示するプログラムを作成する。

こちらでも、まずはコンポーネントをフォームに追加する。

【使用するコンポーネント】
・TPanel：ツールパレット（Standard）
・TMemo：ツールパレット（Standard）
・TBitBtn：ツールパレット（Additional）

① uses 節に ShellAPI ユニットを追加する。
今回必要な DragAcceptFiles 関数、DragQueryFile 関数、DragFinish 関数は、すべて ShellAPI ユニットに格納されている。【ソース 2】
② Create 時に DragAcceptFiles 関数を呼び出す。【ソース 3】
Handle は自身のフォームの Handle、True は許可を意味している。
③ private 部に使用する手続きと Windows Message を追加する。【ソース 4】
④手続きの処理を記述する。【ソース 5】

処理は以上である。実際に動かすと、画面のようになる。【図 7】

アイコンとアプリケーションのどちらの場合も、一度に取得できるファイル名は 1 つだけなので、複数ある場合には繰り返し文を利用する。

また、アイコンへドロップインした場合、ParamStr で取得される 0 番目はアプリケーション自身のファイルである。一方、アプリケーションにドロップインした場合、DragQueryFiles で取得される 0 番目は、アプリケーション自身ではなくドロップインされたファイルである。これらにも注意が必要だ。

6. 応用

これまでの内容を用いて、少し実践的なプログラムを紹介する。今回は、実行中のアプリケーションにドロップインする場合を例とし、追加する形で紹介するが、アイコンにドロップインする場合にも、ほぼ同じ記述で使用できる。

次の処理では、取得するファイル数を 1 つに限定する。また、拡張子を csv 形式に限定する。取得できたファイルパスから csv ファイルのテキストを、TStringList を用いて StringGrid にセットするまでのロジックを追加していく。

【追加コンポーネント】
・TStringGrid：ツールパレット（Additional）

⑤コンポーネントを追加する。今回は sgList と命名している。
前章の手続きを、ソース例のように変更する。【ソース 6】

ソースについて説明していこう。

ExtractFileExt は、文字列から末尾に最も近いドット「.」を探し、それより後の部分を返す関数で、主に拡張子の取得に使用される。今回は csv に限定するために使用している。

StringGrid への入力は、まず変数 slCSV1（TStringList）にファイルのテキスト内容をすべて保持する。TStringList に入力された文字列は Count プロパティで個数が取得できるため、slCSV1 の Count プロパティから csv ファイルの明細行数を取得できる。

次に、取得した行数を StringGrid に設定し、テキストファイルを 1 行ずつ変数 slCSV2（TStringList）の CommmaText プロパティにセットする。

もともと 1 行であった文字列を CommaText プロパティにセットすると、カンマ「,」部分で区切った文字列のリストに変換される。

行数と同様に Count プロパティから列数を取得し、取得した行数・列数から StringGrid のセルに入力していく。

実行すると、以下のようになる。【図 8】

図 8-1：正常な場合
図 8-2：複数ファイルをドロップインした場合
図 8-3：csv 以外のファイルをドロップインした場合

これらをさらに応用すれば、簡単に csv から AS/400 へ、内容を確認・修正しながらファイル転送するプログラムなどの作成が可能である。同様に、OLE 等を用いて Excel ファイルの読み込みといったことも可能となる。

7. 最後に

外部ファイルの取り込みや Windows Message の処理と聞くと難しい印象を受けるかもしれないが、プログラミングしてみると実に 100 行にも満たない記述で実行できることが分かるだろう。

ユーザーの満足度が高いアプリケーションを開発するために、Drag&Drop など、こういった直感的な操作をアプリケーションに組み込むことを、今後も設計者として考慮していただければ幸いである。

M

図4

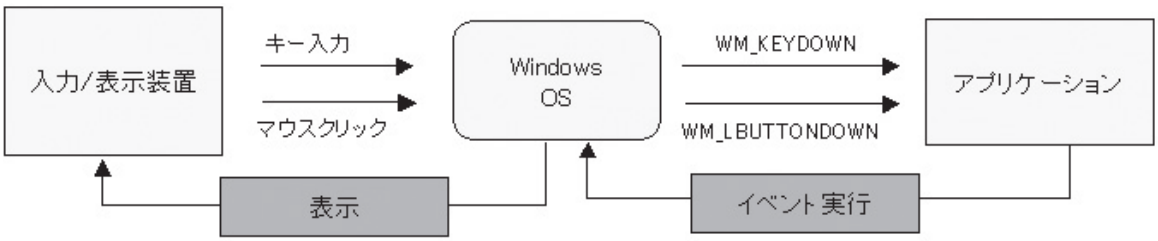


図5

Window Message	コード	意味	Delphi上でのイベント
WM_CREATE	0x0001	ウィンドウが作成された	OnCrete
WM_DESTROY	0x0002	ウィンドウが破棄された	OnDestroy
WM_PAINT	0x000F	ウィンドウを再描画する必要がある	OnPaint
WM_SIZE	0x0005	ウィンドウサイズが変更された	OnResize
WM_LBUTTONDOWN	0x0201	マウスの左ボタンが押された	OnMouseDown
WM_LBUTTONUP	0x0202	マウスの左ボタンが離された	OnMouseUp
WM_LBUTTONDBLCLK	0x0203	マウスの左ボタンがダブルクリックされた	OnDblClick
WM_KEYDOWN	0x0100	キーが押された	OnKeyDown
WM_KEYUP	0x0101	キーが離された	OnKeyUp

図6

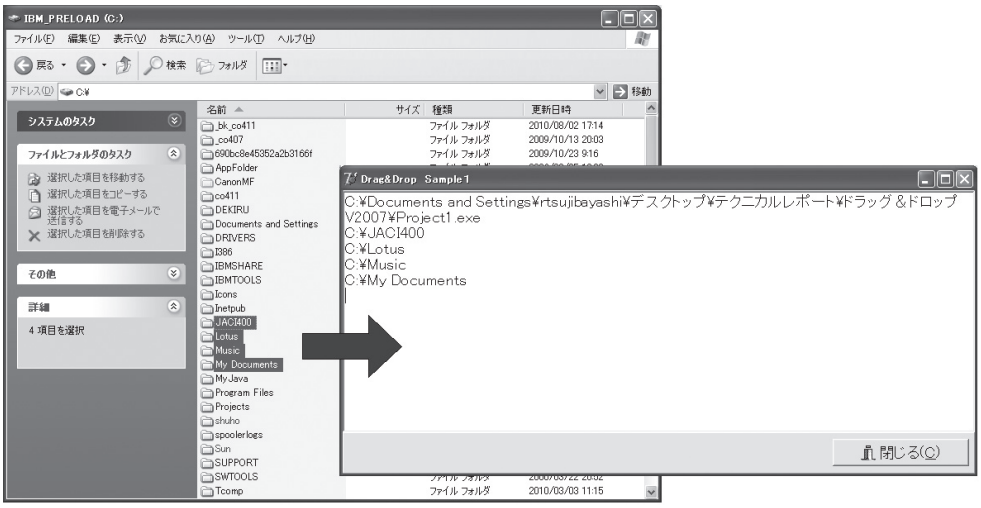


図7

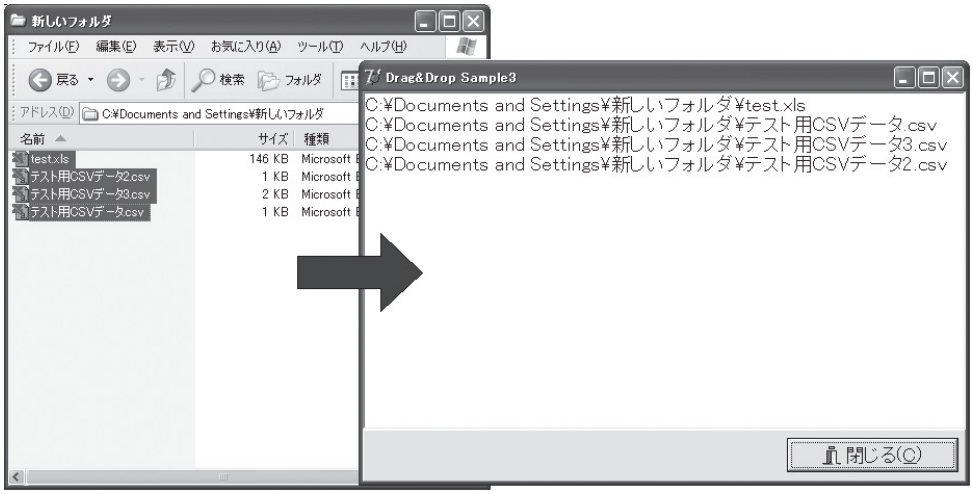


図8-1

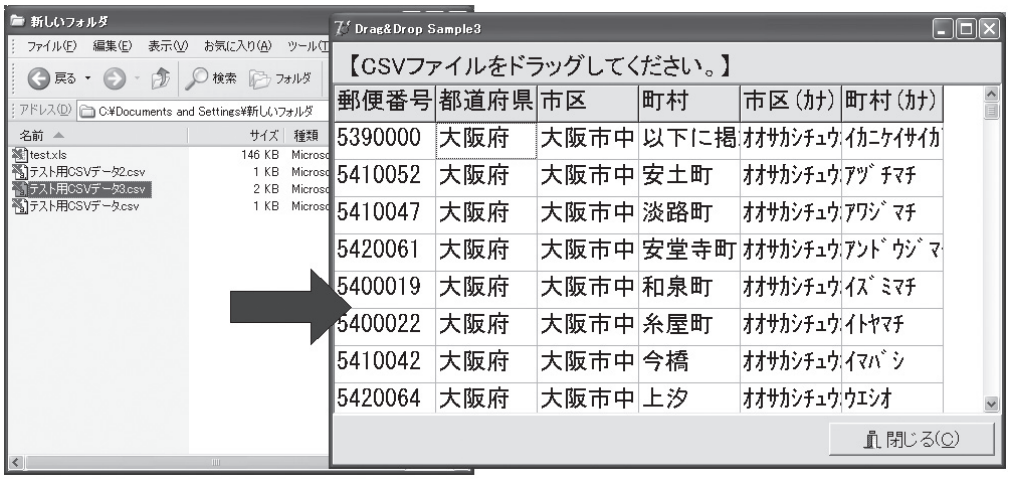

新しいフォルダ
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T)
戻る 検索 フォルダ
アドレス(D) C:¥Documents and Settings¥新しいフォルダ
名前 サイズ 種類
test.xls 146 KB Microsc
テスト用CSVデータ2.csv 1 KB Microsc
テスト用CSVデータ3.csv 2 KB Microsc
テスト用CSVデータ.csv 1 KB Microsc
Drag&Drop Sample3
【CSVファイルをドラッグしてください。】
郵便番号 都道府県 市区 町村 市区(カナ) 町村(カナ)
5390000 大阪府 大阪市中 以下に掲 ｵｵｻｶｼﾁｭｳ ｲｶﾆｹｲｻｲｶ
5410052 大阪府 大阪市中 安土町 ｵｵｻｶｼﾁｭｳ ｱﾂﾞﾁﾏﾁ
5410047 大阪府 大阪市中 淡路町 ｵｵｻｶｼﾁｭｳ ｱﾜｼﾞﾏﾁ
5420061 大阪府 大阪市中 安堂寺町 ｵｵｻｶｼﾁｭｳ ｱﾝﾄﾞｳｼﾞﾏ
5400019 大阪府 大阪市中 和泉町 ｵｵｻｶｼﾁｭｳ ｲｽﾞﾐﾏﾁ
5400022 大阪府 大阪市中 糸屋町 ｵｵｻｶｼﾁｭｳ ｲﾄﾔﾏﾁ
5410042 大阪府 大阪市中 今橋 ｵｵｻｶｼﾁｭｳ ｲﾏﾊﾞｼ
5420064 大阪府 大阪市中 上汐 ｵｵｻｶｼﾁｭｳ ｳｴｼｵ
閉じる(C)

図8-2

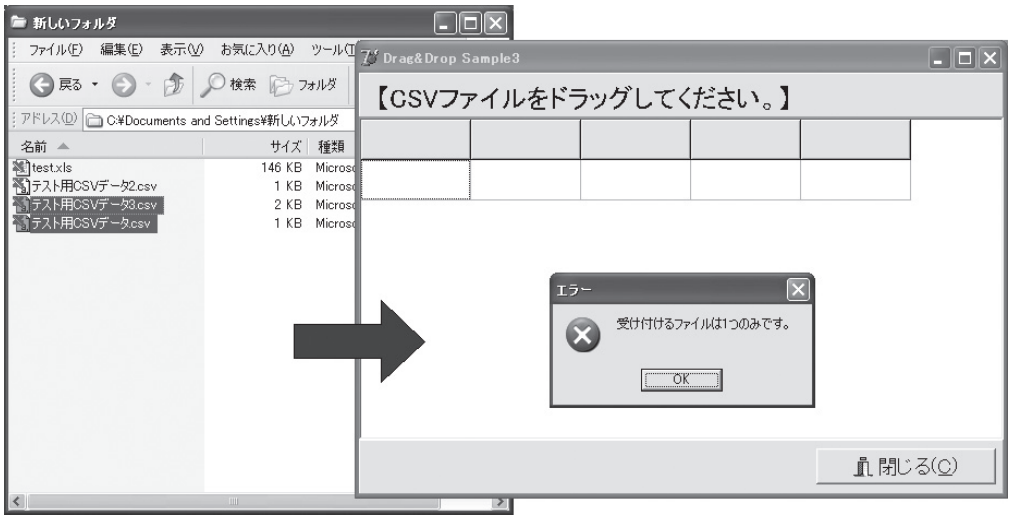
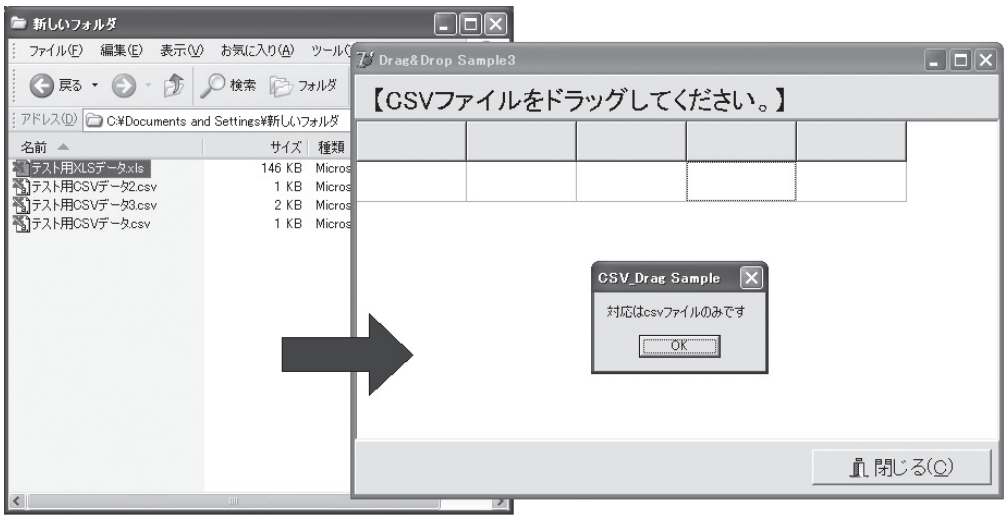

新しいフォルダ
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T)
戻る 検索 フォルダ
アドレス(D) C:¥Documents and Settings¥新しいフォルダ
名前 サイズ 種類
test.xls 146 KB Microsc
テスト用CSVデータ2.csv 1 KB Microsc
テスト用CSVデータ3.csv 2 KB Microsc
テスト用CSVデータ.csv 1 KB Microsc
Drag&Drop Sample3
【CSVファイルをドラッグしてください。】
エラー
受け付けるファイルは1つのみです。
OK
閉じる(C)

図8-3


新しいフォルダ
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T)
戻る 検索 フォルダ
アドレス(D) C:¥Documents and Settings¥新しいフォルダ
名前 サイズ 種類
テスト用XLSデータ.xls 146 KB Micros
テスト用CSVデータ2.csv 1 KB Micros
テスト用CSVデータ3.csv 2 KB Micros
テスト用CSVデータ.csv 1 KB Micros
Drag&Drop Sample3
【CSVファイルをドラッグしてください。】
CSV_Drag Sample
対応はcsvファイルのみです
OK
閉じる(C)

ソース1

```
procedure TForm1.FormCreate(Sender: TObject);
var
  i : Integer;
begin
  //1つ以上のファイルがDrag&Dropされた場合のみ行う
  if ParamCount > 0 then
  begin
    //パラメータの数だけ繰り返す
    for i := 0 to ParamCount do
    begin
      //メモにパラメータを表示する
      Memo1.Lines.Add(ParamStr(i));
    end;
  end;
end;
```

ソース2

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComObj, StdCtrls, Grids, Buttons, ExtCtrls, ComCtrls, AppEvnts,
  ShellAPI; // 追加する

type
  TForm1 = class(TForm)
```

ソース3

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  //ドラッグ&ドロップを受け付け許可
  DragAcceptFiles(Self.Handle, True);
end;
```

ソース4

```
private
  //Windowsからのメッセージ WM_DROPFILES を WMDropFiles で受け取るようにする
  procedure WMDropFiles(var msg: TWMDropFiles);
  message WM_DROPFILES;
```

ソース5

```
 try
   //ドロップされたファイル数を取得
   FileNum := DragQueryFile(msg.Drop, $FFFFFFFF, nil, 0);
   for i := 0 to FileNum - 1 do
   begin
     //ファイル名の取得
     DragQueryFile(msg.Drop, i, FileName, SizeOf(FileName));
     Memo1.Lines.Add(FileName);
   end;
 finally
   //ドラッグ完了時、メモリの解放
   DragFinish(Msg.Drop);
 end;
nd;
```

ソース6

```
procedure TForm1.WMDropFiles(var msg: TWMDropFiles);
var
  FileName : array[0..255] of Char;
  sFileName, sFileType : String;
  FileNum : Integer;
  slCSV1     : TStringList;
  slCSV2     : TStringList;
  i,j : Integer;
begin
  try
    FileNum := DragQueryFile(msg.Drop, $FFFFFFFF, nil, 0); //ファイル数の取得
    if FileNum <> 1 then // ファイル数を限定する
    begin
      MessageDlg('受け付けるファイルは1つのみです。', mtError, [mbOK], 0);
      Abort;
    end;
    DragQueryFile(msg.Drop, 0, FileName, SizeOf(FileName)); //ファイル名取得
    sFileName := FileName;                                  //文字列型にする
  finally
    //ドラッグ完了時、ハンドルの解放
    DragFinish(Msg.Drop);
  end;
  sFileType := ExtractFileExt(sFileName); //拡張子取得
  if sFiletype <> '.csv' then//エクセル以外の場合
  begin
    ShowMessage('対応はcsvファイルのみです');
    Abort;
  end;
  // Gridに入力
  slCSV1 := TStringList.Create;
  slCSV2 := TStringList.Create;
  //ファイルのオープン
  slCSV1.LoadFromFile(sFileName);
  sgList.RowCount := slCSV1.Count;  //行数取得
  for i := 0 to sgList.RowCount - 1 do
  begin
    slCSV2.CommaText := slCSV1[i];  //一行分情報
    sgList.ColCount := slCSV2.Count; //列数取得
    for j := 0 to slCSV2.Count -1 do
      sgList.Cells[j,i] := slCSV2[j];
  end;
  slCSV1.Free;
  slCSV2.Free;
  sgList.Col := 1;
  sgList.Row := 1;
end;
```