List boxes (Part II)

By Jean-Yves Fock-Hoon, Quality Assurance Manager, 4D Inc.

TN 06-26

Overview

前後編で構成されるこのテクニカルノートは、デザインモードおよびプログラミング言語の両面からリストボックスとその使用方法を紹介することを目的としています。後編の今回は、リストボックスをプログラミング言語で制御する方法に焦点が絞られています。

Creating a list box from the language

最初のデモは、フォームに配置されたリストボックスの列のプロパティをコマンドで設定するというものです。メニューバーから Building a List box を選択すると、ダイアログで [Contacts];”Dial1”フォームが表示されます。Restart ボタンをクリックすると、リストボックスのプロパティが初期設定に戻ります。

はじめに Clear All ボタンをクリックしてみてください。ボタンのオブジェクトメソッドでは、Get number oflistbox columns 関数で列の数(表示/非表示列の数)が調べられます。次に DELETE LISTBOX COLUMN コマンドですべての列がクリアされ、リストボックスが空になります。

次は Insert Default ボタンをクリックしてみてください。クリアされた 4 列が再びリストボックスに挿入されます。正確には、列が挿入され、ヘッダのタイトルが設定されます。実行しているのは M_InsertColumn メソッドで、使用しているコマンドは INSERTLISTBOX COLUMN と BUTTON TEXT です。挿入された列には、デフォルトのフォント名、フォントサイズ、カラーが適用されます。

今度は More Columns ボタンをクリックしてみてください。同じメソッド M_InsertColumn がコールされ、4 列がさらに追加されます。

テキストの属性が設定できるのは、すでに存在する列に限られます。つまり、同じメソッドの中で列の追加と追加された列の設定を同時にすることはできません。
4 列がすべて追加された時点で Attributes ボタンを押してみてください。コールされるメソッド M_SetupColumn は、SET RBG COLORS、FONT、FONT SIZE および FONT STYLE コマンドを複合した汎用的なメソッドです。リストボックスの列ごとの背景色を設定するためなどに使用することができます。at_city 列の背景色は、意図的に他とは逆の設定にしています。

今度は Grid ボタンをクリックしてみてください。SET LISTBOX GRID COLOR および SHOW LISTBOX GRID コマンドによってリストボックスの罫線が表示やカラーが切り替わります。ボタンを何度もクリックすることによって 5 種類の設定を確認することができます。

Re-Order ボタンをクリックすると、列の並び替えが実行されます。列の順番を入れ替える操作に相当する単独のコマンドはありませんが、いくつかのコマンドを組み合わせれば、同じことが簡単に実現できます。次のようにすれば、列を並び替えることができます：

- 列の数を取得する
- それぞれの列の幅を調べる
- 列をすべてクリアする
- 並び替えたい順序で列を挿入する

列の幅については、Get listbox column width 関数で知ることができます。後に列を並び替えて挿入するために順序の配列も使用すると便利です。次に列をすべてクリアしてから挿入してゆくわけですが、その前に M_Setorder で挿入する順序を定める必要があります。列の順序は、オブジェクト名のローカル配列$al_newOrder で管理します。受け取ったオブジェクト名が列でなければ、メソッドは何もしません。列のオブジェクト名をメソッドに渡さなかった場合、その順序は元のままです。

列の順序が確定したならば INSERT LISTBOX COLUMN コマンドでループで実行し、列をリストボックスに挿入してゆくことができます。列の配列は、表示されようとしている順序でマルチソートを実行してから挿入します。すべての列が挿入された後、ヘッダのタイトルを設定し、列の幅を元に戻します。必要に応じ、ここでカラーやフォントを設定することができます。

ダイアログで提供されている最後のボタンは Best Object Size です。ここでは GET LISTBOX ARRAYS コマンドでヘッダや列のオブジェクト名を調べ、ループでオブジェクトサイズを評価しています。BEST OBJECT SIZE はヘッダおよび列について最良のオブジェクト幅を返すので、もっとも広い幅を必要とする列を特定し、その列の幅を優先的に大きくすることができます。

Moving Columns and Rows

次のデモは、リストボックスの下に合計を表示する変数オブジェクトを配置し、リストボックスの列の幅に合わせてそのオブジェクトをリサイズするというものです。このデモでは、[Contacts];”Dial2”フォームを使用します。

このデモでは、移動やリサイズによって変更された列の幅を取得する方法がポイントになっています。GET OBJECT RECT コマンドをリストボックスに対して使用すれば、。リストボックスの位置が分かります。列の数が分かっていれば、Get listbox column width 関数で幅を調べ

ることができ、その結果、リストボックスの座標と幅から列の水平位置を計算することができます。

合計行として使用するオブジェクトの高さはすでに分かっているかもしれません。分からなければ、GET OBJECT RECT コマンドで調べることができます。このオブジェクトを表示する位置は、リストボックスの真下であり、幅は列の幅の合計と一致している必要があります。

あとは、リストボックスの列の幅が変更されるたびにオブジェクトを MOVE OBJECT コマンドでリサイズするわけですが、その引き金となるイベントは On Column Moved と On Column resize に限定されます。関係するフォームイベントではリサイズされたオブジェクトを特定し、必要に応じて合計行のオブジェクトをリサイズします。

列が移動されたのであれば、すべての列の水平位置を再計算し、合計行のオブジェクトが正しく対応する列の下に表示されるように移動しなければなりません。移動された列に対するポインタは MOVED LISTBOX COLUMN NUMBER コマンドで調べることができ、それによって列の順序を並び替えて列の幅を計算し、合計行のオブジェクトを更新することができます。

列がリサイズされた場合、順序に変更はなく、幅が変更された 2 列についてのみ、情報を更新すればよいのですが、どの列がリサイズされたのかを知るためのコマンドがないため、やはりすべての列の幅を再計算し、合計行のオブジェクトを更新するのが無難です。

このダイアログは、垂直方向にリサイズすることができ、合計行のオブジェクトは、フォームリサイズによって垂直方向に移動するようになっています。水平方向の拡大リサイズは、マウスボタンがリリースされた時点で拡大した列の幅を取得し、合計行のオブジェクトを更新すれば問題ありませんが、水平方向の縮小リサイズは固定サイズである変数オブジェクトの阻まれて実行することができないため、水平方向のリサイズは無効に設定されています。このオブジェクトにリサイズプロパティを設定しても、ある一定のサイズよりは小さくリサイズできないため、その時点でフォームの縮小リサイズも阻まれてしまいます。このような制限があるため、サンプルではフォーム水平リサイズ自体を無効に設定しています。

Simple Drag and Drop

このデモは、同一のフォーム内に配置された複数のリストボックス間で単純なドラッグ＆ドロップを処理する方法の実践例です。[Contacts];"Dial3"フォームが使用されています。フォームには、リストボックス MyListBox および MyListBox2 が存在します。

MyListBox2 から MyListBox へ項目をドラッグ＆ドロップで挿入したいとしましょう。まず、MyListBox2 をドラッグ可に設定する必要があります。そうしないと、MyListBox オブジェクトがドラッグ＆ドロップイベントを受け付けないからです。さらに、MyListBox をドロップ可に設定する必要もあります。そのようにすれば、MyListBox オブジェクトは On Drag over および On Dropped イベントを評価できるようになります。On Drag over イベントでは、DRAG AND DROP PROPERTIES コマンドを実行し、何がどこからドロップされようとしているのかを調べることができます。ここでは MyListBox2 からのドロップだけを受け付けたいので、第 1 引数のポインタを調べ、それが MyListBox2 であるかを確認します。ここでは同じアイテムの連続ドロップは拒否したいので、デフォルトの動作としてすべてのドロップを拒否し、ドロップされようとしている MyListBox2 の項目要素がドロップを受け付ける MyListBox に存在しない場合、ドロップを許可しています。ドロップを許可するには$0 に 0 を代入します。ドロップを拒否するには$0 に 1 を代入します。ドロップが拒否されている場合、ドラッグ中にドロップ可を示す視覚効果は発生しません。

Drag and Drop between processes

このデモは、異なるプロセス間でリストボックス同士のドラッグ＆ドロップを処理する方法の実践例です。[Orders];"Input"および[Contacts];" DialProduct""フォームが使用されています。基本的には、前述のデモと同じことをしています。第 1 のプロセスは製品リストのプロセスであり、データファイルから取り出された製品データが簡単なリストボックスに展開されています。第 2 のプロセスは、リストボックスのドラッグ＆ドロップでデータを受け付け、ADD RECORD コマンドを実行するようになっています。追加すべきデータは、DRAG AND DROP PROPERTIES、RESOLVE POINTER、GET PROCESSVARIABLE で取得することができます。ここの部分の処理はとても単純です。

項目が挿入された後、直接その数量を更新することができます。0 を入力すれば、項目を取り除くことになります。0 以外の値を入力した場合、その行の合計値を再計算する必要があります。いずれの場合も、表全体を合計値を再計算する必要があります。

表全体を合計値を計算する際、更新されようとしている行を除く行の合計値をあらかじめ取得しておくと便利です。この値は、On Getting Focus イベントを調べることができます。行が取り除かれた場合、更新後の合計値は、その行を除く行の合計値です。行が更新された場合、更

新後の合計値は、その行を除く行の合計値に更新された行の新しい合計値を加えたものになります。この処理は、リストボックスの On Data Change イベントで実行することができます。

Editing a Cell

このデモは、リストボックスの行をスクロールしたり、行を更新したりするする方法の実践例です。[Contacts];"Dial5"フォームが使用されています。フォームには、表示列 2、非表示列 2 のリストボックスが存在し、データファイルから取り出された連絡先のリストが表示されます。フォームが表示された時点では、2 列しか表示されません。セルを更新しようとすると、残りの列も表示されます。これは On Getting Focus イベントで SET VISIBLE コマンドが使用されているためです。On getting Focus イベントは、フォームを最初に表示する際にも発生するため、そのタイミングでは SET VISIBLE コマンドを実行しないように vOnLoad 変数で管理しています。

セルが更新された後には、非表示列を隠すために SET VISIBLE コマンドを再び使用します。適しているフォームイベントは、On Data Change および On Losing Focus です。更新は、enter キーを押して更新を確定する(On Data Change)、あるいは何も変更しないで他のオブジェクトをクリックする(On Losing Focus)ことによって終了できるため、両方のイベントで非表示列を隠しています。

アルファベットのボタンがクリックされた場合、配列の並び替えを実行し、指定した文字から始まる最初のデータを探します。並び替えは MULTI SORT ARRAY コマンドで実行します。並び替えを実行した後、すぐに最初のデータを更新できるように、リストボックスをスクロールして最初のデータを表示するようにします。これには SCROLL LINES コマンドを使用します。項目の更新は EDIT ITEM コマンドで開始します。SCROLL LINES および EDIT ITEM は、いずれも行番号を必要とするので、Find in array 関数で特定した番号を渡し、項目の更新をするので、非表示列を表示します。

項目更新中に J ボタンをクリックすると、単純に更新が終了します。これはボタンをクリックしたことによってフォーカスが失われ、J から始まるデータがないために SCROLL LINES および EDIT ITEM が実行されなかったためです。

Form Events

このデモは、特にリストボックスのために導入された 6 種類の新しいフォームイベントの使用例です。[Contacts];"Dial6"フォームが使用されています。

- On Column moved フォームイベント：

このフォームイベントは、MOVED LISTBOX COLUMN NUMBER コマンドとペアで使用されるものです。どの列が、どこからどこへ移動されたのかを検出することができます。これらのフォームイベントとコマンドは、列の順序を特定するために使用できます。サンプルでは、列の順序を特定し、それぞれの列の幅を算出し、Amount 列の幅に合わせてリストボックスの下

に配置された合計行の SumAmount 変数オブジェクトをリサイズしています。

- On Column Resize フォームイベント：

列のリサイズを検出するフォームイベントです。列は同じ位置でリサイズすることもできるので、リサイズによっても幅が変動します。サンプルでは、リサイズ後に Amount 列の位置と幅に合わせてリストボックスの下に配置された合計行の SumAmount 変数オブジェクトをリサイズしています。

問題の変数オブジェクトのリサイズはいくらか厄介です。対応する Amount 列は、極端に小さくなったり、リストボックスの表示領域外に出たりするかもしれません。そのような場合、問題の変数オブジェクトは表示されるべきではありません。そのような理由から、サンプルでは GETOBJECT RECT および GET FORM PROPERTIES コマンドでリストボックスの座標とフォームの座標を取得しています。Amount 列の幅が、リストボックスの右端を超えているならば、リストボックスの表示領域外に列が出ていることを意味するので、変数オブジェクトはフォームの領域外に移動して表示されないようにします。フォームの幅はこの処理のために必要です。リストボックスに水平スクロールバーが存在する場合、それが使用されたことを検出するイベントはないので、列が表示領域内に現れたときに変数オブジェクトのサイズを更新することができません。そのようなわけでサンプルでは水平スクロールバーを非表示にしています。

回避策としては、Amount 列が極端に狭い場合や領域外にある場合は、変数オブジェクトを固定サイズで表示するという方法が考えられるかもしれません。いずれにしても、やはり Amount 列の下に変数オブジェクトが表示されるようにするためには、リストボックスの水平スクロールバーを非表示にする必要があります。

Note: サンプルコードでは、$x1 と $x2 に変数オブジェクトの最終的な座標を代入しています。この値は、列の幅とリストボックスのサイズに基づいて計算されています。値から 15 を減算しているのは、垂直スクロールバーの幅を考慮しているためです。

- On Header Click フォームイベント

ヘッダがクリックされたときに発生するイベントです。このイベントのコンテキストで、たとえばヘッダのテキスト属性を変更したりすることができます。カラムのプロパティで並び替えが有効にされている場合、ヘッダクリックで並び替えが実行されます。この自動的な並び替えを拒否するには、フォームイベントのコンテキストで $0 に -1 を代入します。

サンプルでは、Amount 列のヘッダをクリックするとヘッダのタイトルテキストが青色に変わるようになっています。なお、フォームイベントでは変数名を調べ、vAmount であれば $0:=1 を実行するようにもなっています。$0 に -1 が代入された場合、次の On After Sort フォームイベントは発生しないことになります。

- On After Sort フォームイベント：

ヘッダオブジェクトのクリックによる自動的な並び替えが実行された後に発生するイベントです。並び替え後に第 1 行を選択したり、冒頭の数行に基づいて特定の処理を実行したりするような場合に使用します。GET LISTBOX ARRAYS コマンドなどでヘッダの変数に対するポイン

タを取得していれば、その変数値を調べることによってヘッダの状態を知ることができます。値が 0 であれば、そのヘッダはいまだクリックされたことがないことを意味します。値が 1 または 2 であれば、昇順または降順の並び替えクリック操作があったことを意味します。サンプルでは、ヘッダの状態に基づいてタイトルをイタリック体またはボールド体に変えています。

- On Before Data Entry フォームイベント：

セルが更新されようとしているときに発生するイベントです。サンプルでは、更新されようとしている行の番号を調べ、それが最初の 3 行以内であれば POST KEY(13)でデータ入力を拒否しています。

- On Row Moved フォームイベント：

行が移動されたときに発生するイベントです。移動された行の番号は MOVED LISTBOX ROW NUMBER コマンドで調べることができます。

サンプルでは、移動した行の番号をウインドウタイトルに表示しています。

Multiple pages

このデモは、複数のリストボックスをフォームの複数ページに配置するという応用例です。[Contacts];"Dial7"フォームが使用されています。ページ 0、1、2、3 には、それぞれリストボックス MyListBox、MyListBoxP1、MyListBoxP2 および MyListBoxP3 が配置されています。常に表示されているページ 0 のリストボックスで適当な行を選択すると、その他のリストボックスはすべて表示内容が更新されます。ページはタブコントロールをクリックすれば切り替えることができます。このようにして同一のフォームに複数のリストボックスを配置することができるわけですが、忘れていけないのは、変数名が重複してはならないという点です。同じリストボックス、または同じページ(カレントページとページ 0)に配置された別のリストボックスで同じ変数を使用することはできません。同じ変数を同一ページ内の複数の箇所に表示させることは、現バージョンでは動作しているように思えるかもしませんが、サポートされている動作ではありません。

このサンプルでは、リストボックスに表示することができるすべての配列タイプをカバーし、ヘッダアイコンや選択リストの使用など、他のデモですでに紹介した機能も再利用しています。ページ 0 のリストボックスで行をクリックすると M_BestObjectSize メソッドがコールされ、表示中のページに応じてリストボックスの Mileage、Total または Margin 列の幅が最良サイズに再定義されるようになっています。

Puzzle

このデモは、リストボックスの少し変わった応用例です。[Contacts];"DialPuzzle"フォームが使用されています。15 枚のピクチャが表示され、reset ボタンをクリックすると、ピクチャがシャッフルされるようになっています。使用しているのは 8 列のリストボックスです。4 列はピクチャライブラリからロードされた画像を表示するピクチャ配列、4 列は対応する位置に表示

されているピクチャまたは空白を管理するための倍長整数配列です。ピクチャの移動コードは比較的簡単で、クリックを検出すると周囲に空白があるかを調べ、あれば配列の要素を入れ替えているだけです。リストボックス特有のコマンドを使用しているわけではなく、配列を表示するための入れ物としてリストボックスが使用されています。

Summary

リストボックスは、スクロールエリアよりも柔軟に制御でき、効果的に配列が表示できる強力なオブジェクトです。このテクニカルノートの前編では、デザインモードで設定するリストボックスのプロパティについて取り上げました。後編では、リストボックスをコマンドで操作すればどのようなことができるのか、いくつかの例を紹介しました。