Технология Drag and Drop. Часть 3. Теоретические основы

Цели:

- Закрепить навыки работы с механизмом *Drag&Drop*
- Приобрести практические навыки использования механизма *Drag&Drop* для переноса информации из внешнего приложения.

Технология *Drag&Drop* позволяет производить перенос информации не только между компонентами одного и того же приложения, но и между различными приложениями Windows.

В отличие от буксировки внутри одной программы, для работы с ‘внешней’ буксировкой необходимо обрабатывать сообщения, посылаемые ОС приложению.

Подготовка приложения к получению данных:

- **Этап 1. Инициализация.**

  В случае переноса данных из другого приложения (в данном примере – имени файла из Explorer) мы не можем контролировать начало переноса. В этом случае мы просто готовим приложение к самой возможности получения данных извне.

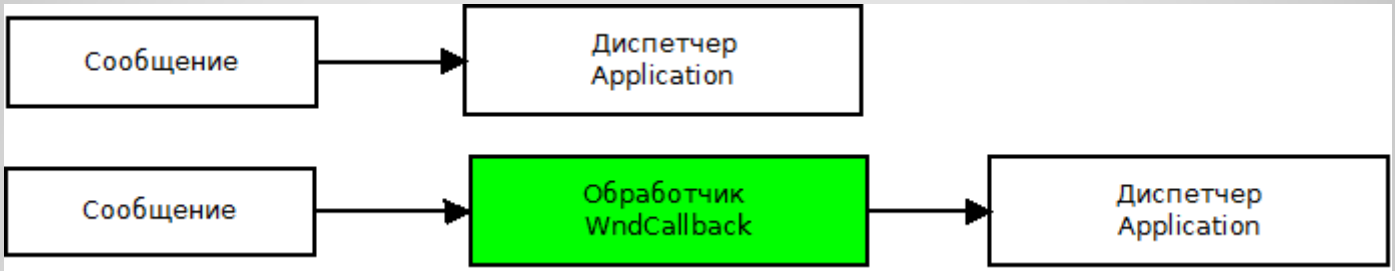
Подготовка приложения производится с помощью API функций Windows, и, для их использования в Lazarus, необходимо в раздел Uses добавить модуль Windows и вспомогательный модуль LCLProc.

Для активации получения информации, необходимо сделать вызов *DragAcceptFiles*.

- **Этап 2. Сообщение Windows.**

  При переносе данных извне, приложение получает сообщение **WM_DROPFILES**, при попытке ‘сброса’ над ним перетягиваемого из Explorer’а файлов. Чтобы ‘поймать’ это сообщение, необходим обработчик этого события.

  Для получения сообщений ОС Windows в чистом виде, необходимо встроить новый обработчик в цепь обработки событий (WndCallback). Из созданного обработчика необходимо вызвать старый диспетчер сообщений, иначе стандартные обработчики событий не будут корректно вызваны.



Для реализации своего обработчика необходимо запомнить старый и создать новую функцию обработчика.

В раздел **var** необходимо будет добавить описание:

```
PrevWndProc : WNDPROC;
```

Новый обработчик будет выглядеть так:

```
function WndCallback(Ahwnd: HWND; uMsg: UINT;
    wParam: WParam; lParam: LParam):LRESULT; stdcall;
```

В качестве аргументов ему будут переданы все параметры сообщения.

- **Этап 3. Вызов обработчика сообщения**.

  Теперь все сообщения будут проходить через новую функцию. При отпускании кнопки мыши над приложением или любым из его компонент, вырабатывается событие WM_DROPFILES, которое и должен обрабатывать WndCallback.

- **Этап 4. Получение данных в обработчике**.

  В обработчик сообщения, от ОС Windows, передаются параметры сообщения:

  hWnd - окно адресата

  uMsg - номер сообщения (WM_DROPFILES)

  lParam, wParam - параметры сообщения

  В параметрах храниться внешняя структура типа HDROP, с помощью которой можно получить имена файлов.

Данная структура передается функции ***DragQueryFile***.

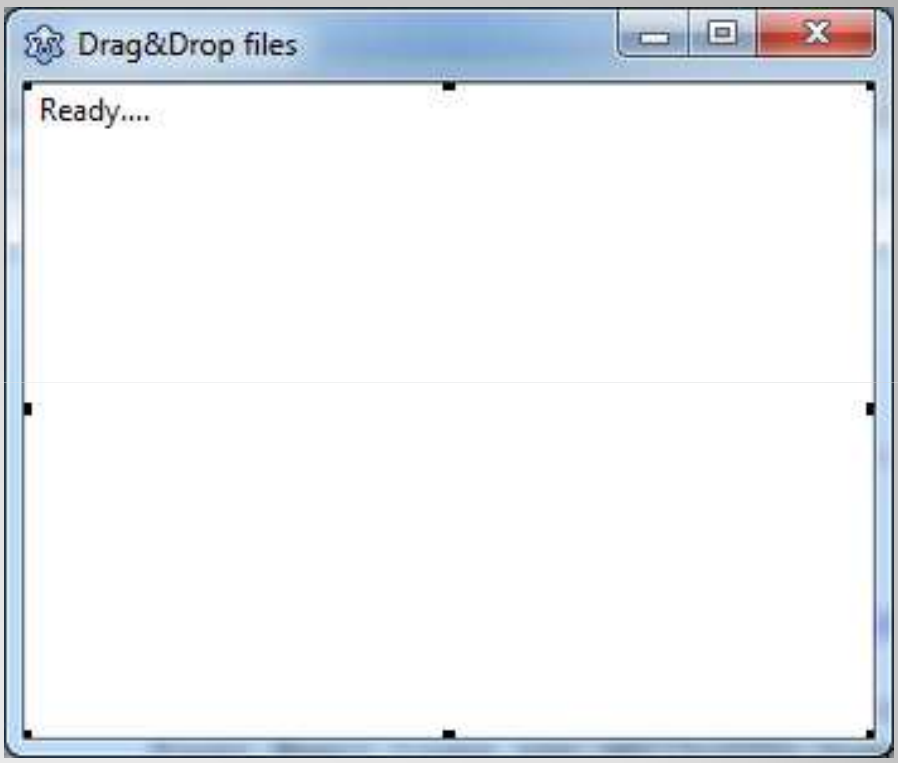
В качестве других аргументов, данной функции передается индекс файла в структуре (для получения нескольких имен файлов) и буфер для принятия одного имени файла в программу. Если в качестве индекса указано значение -1, то функция вернет количество передаваемых в сообщении файлов.

Для очистки внешней структуры, использованной при передаче фалов необходимо воспользоваться функцией ***DragFinish***.

Задание
Перенос *данных из Explorer Windows*

Создадим макет главной формы приложения. Для этого разместим на ней компонент ***TMemo*** и установим для него свойство ***Align*** в ***alClient*****.**

В этом компоненте мы будем отображать имена фалов, принятых извне.



Добавим модули **Windows** и **LCLProc** в соответствующий раздел программы

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,Dialogs, StdCtrls, Windows, LCLProc;
```

Добавим переменную для хранения старого обработчика сообщений Windows в раздел **var**.

```
var
  Form1: TForm1;
  PrevWndProc : WNDPROC;
```

Установим готовность приложения к приему информации и запомним старый диспетчер сообщений.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  PrevWndProc:=Windows.WNDPROC(
                  SetWindowLongPtr(Self.Handle,GWL_WNDPROC,
                  PtrInt(@WndCallback)));
  DragAcceptFiles(Handle, true);
end;
```

Добавим код обработчика сообщений Windows

```
function WndCallback(Ahwnd: HWND; uMsg: UINT;
         wParam: WParam; lParam: LParam):LRESULT; stdcall;
var
  Drop  : HDROP;
  N,L,i : integer;
  fname : string;
begin
    if uMsg=WM_DROPFILES then
    begin
      result:=0;
      Drop:=HDROP(wParam);
      N:=DragQueryFile(Drop,$FFFFFFFF,nil,0); // Число файлов
      if N<1 then exit;
```

Код обработчика сообщений Windows (***продолжение***)

```
for i:=0 to N-1 do
 begin
  L:=DragQueryFile(Drop, i, nil, 0); // получаем длину i-ого имени
  SetLength(fname,L);
  DragQueryFile(Drop,I,PChar(fname),L+1); // имя i-ого файла
  Form1.Memo1.Lines.Add(ANSIToUTF8(fname));
end;
DragFinish(Drop);
exit;
end;
    result:=CallWindowProc(PrevWndProc,Ahwnd, uMsg,
                                        WParam, LParam);
end;
```

В данном виде обработчик добавляет в Memo1 само имя файла (причем только одного). Для того, чтобы открыть переносимый файл, необходимо воспользоваться методом `Memo1.Lines.LoadFromFile(fname);`

Дополнительные задания

1. Создать проект, который позволяет открывать картинки (компонент ***TImage***). Картинки должны иметь расширениее (тип) ***.bmp***
2. Создать проект, который открывает только текстовые файлы (***.txt***). Для этого необходимо проверять расширение файлы (с помощью функции ***ExtractFileExt(имя_файла).***
3. Создать приложение, которое автоматически будет определять тип файла и загружать его как картинку или как текстовый документ.