 从一个程序拖动数据到另一个程序（典型的情况是拖动文本）已经不是什么新鲜事了，很多共享软件都支持这个功能（比如说著名的flashget、netants等的浮动窗口功能）。作者一直想在自己的软件中实现这个功能，经过一段时间的资料搜索，有了部分的了解，但这些文档大多数使用C 描述。于是，好东西（也算不上好好的吧J）不敢独享，经过整理我将自己用delphi的实现方法写出来，并简单的讲解一下OLE Drag and Drop机制。 所谓OLE Drag and Drop不用翻译大家一看就能知道它的意思了，它使不同的程序（或同一个程序）通过相互拖动数据来进行交互成为可能。在这方面windows为我们在后面做了很复杂的工作，幸运的是我们不用担心它的复杂性，windows已经为我们提供了两个相当关键的接口：IDropSource、IDropTarget，我们只用实现这两个接口便可以方便的实现OLE Drag and Drop，前者由允许拖放自己数据的数据源程序实现，后者由允许接收拖放数据的数据目标程序所实现。在本文中，我们只讨论后者，因为我们只希望接收来自其它程序拖放过来的数据，而前者已经被大多数程序实现了（如IE、windows帮助系统等，如果想了解更多关于IDropSource的实现请参看win32 sdk帮助文件）。 接下来我们简单的了解一下windows是怎样在后面实现数据拖放的，然后我们实现 IDropTarget的一个例子程序（关于程序中的api和格式会在出现的时候给予说明）。Windows在后台调用了一个重要的DoDragDrop 函数来

检测接口和调用有我们实现的接口方法，下面是这个函数工作时大概的步骤：当我们开始向可以接收数据的窗体拖动数据时，DoDragDrop首先检查鼠标下的窗体是否被注册为可以接收的窗体（通过RegisterDragDrop api来注册，该函数有两个参数，第一个为要注册的窗体的句柄，第2个为指向我们实现IDropTarget的类的一个对象指针，在我们的窗体不需要再接收任何拖动过来的数据时使用RevokeDragDrop来解除注册，它只有一个参数，就是欲解除的窗体句柄，另外重要的一点是要成功的调用这些函数，我们必须在程序开始时使用OleInitialize(nil)在结束时调用OleUninitialize以便初始化OLE library。） 如果窗体可以接收拖动，DoDragDrop便调用IDropTarget接口的DragEnter方法，该方法通过一个引用参数返回一个拖动的效果 dwEffect，它可以有不同的取值（通过检查IDataObject来决定），你可以在帮助中找到这些值，其中有表示复制、剪切等的操作（指对于实现 IDropSource的程序），具体的你还会在下文的代码中看到。然后DoDragDrop通过调用 IDropSource::GiveFeedback来将dwEffect传递给IDropSource。 接下来 DoDragDrop根据鼠标的状态调用诸如IDropTarget接口的DragOver、DragLeave方法，整个过程是在循环中不断的检测鼠标的状态来实现的，如果这时你改变了拖动目标它会再次检测新的目标并重复上面的过程，如果你在键盘上同时按住了其它的键，它会调用IDropSource::QueryContinueDrag并在改变了键盘状态码（你可以通过DragEnter、DragOvert中的 grfKeyState参数来检测改值，并根据它做相应的工作）后继续重复上面的过程。 当我们最后松开鼠标后DoDragDrop将调用IDropTarget的Drop方法

，它最后一次返回dwEffect，最后根据dwEffect我们可以在这个方法中得到IDataObject中的数据，一次完整的拖放操作就完成了。下面的图说明一次拖放操作的过程： 上面说了这么多，其实都是windows在后台为我们所做的工作，我们只是大概的了解一下这个过程，下面我们通过一个例子来实现一个可以接受文本的 memo，窗体中只有一个Tmemo，请注意代码中的注释。我们先来看看在程序主窗口创建和撤消时需要做的一些必要的初始化和结束操作： constructor TForm1.Create(AOwner: TComponent). begin inherited Create(AOWner). OleInitialize(nil). DragAndDropOLE:=TDragAndDropOLE.Create. // TDragAndDropOLE便是我们要实现IDropTarget接口的类 end. destructor TForm1.Destroy. begin DragAndDropOLE.Free. OleUninitialize. inherited. end. 下面我们来看看关键的TDragAndDropOLE的实现，首先他应该实现IunKnown接口，这是一个基本的接口用来实现引用计数，熟悉COM的朋友应该都知道这个接口及其实现方法，下面只给出实现它的代码不做详细说明，主要要注意的是IDropTarget的实现方法： 首先是TDragAndDropOLE的声明部分： type TDragAndDropOLE=Class(TObject,IUnknown,IDropTarget) private CanDrop:HResult. fe:TFormatEtc.//数据的格式，在实现部分给出详细说明 FRefCount:integer.//引用计数 protected { Iunkown } function _AddRef:integer.stdcall. function _Release:integer.stdcall. function QueryInterface(const IID:TGUID.out Obj):HResult.stdcall. { I0dropTarget } function DragEnter(const dataObj: IDataObject. grfKeyState: Longint. pt:

TPoint. var dwEffect: Longint): HResult.stdcall. function DragOver(grfKeyState: Longint. pt: TPoint.var dwEffect: Longint):HResult.stdcall. function DragLeave: HResult.stdcall. function Drop(const dataObj: IDataObject. grfKeyState: Longint. pt: TPoint. var dwEffect: Longint): HResult. stdcall. 