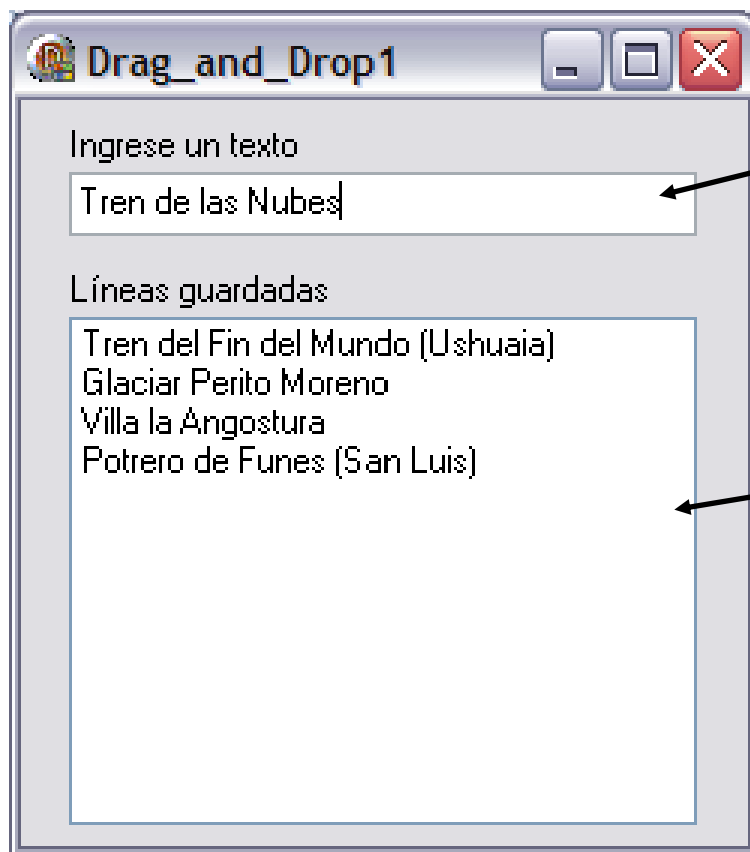




Drag & Drop (arrastrar y soltar)

- * Este proceso está definido entre:
 - ▣ La Componente donde comienza el arrastre.
 - ▣ Propiedad **DragMode := dmAutomatic;**
 - ▣ La Componente donde se "suelta".
 - ▣ Eventos:
 - **OnDragOver** : indicar aquí de qué componentes recibe.
 - **OnDragDrop** : qué se hace con la información.

Ejemplo : El texto del LabeledEdit se puede *arrastrar y soltar* en el ListBox



Propiedad

DragMode := dmAutomatic

Eventos

OnDragOver

OnDragDrop

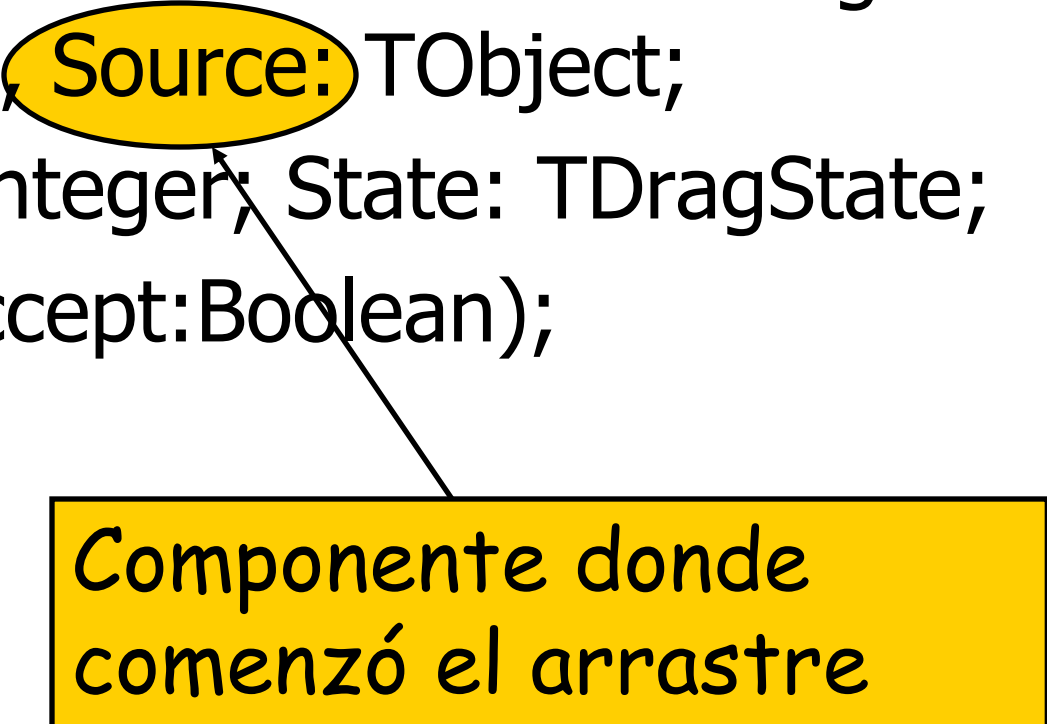
Drag_and_Drop1.dpr

OnDragOver del ListBox

Forma General

```
procedure TForm1.ListBox1DragOver(  
  Sender, Source: TObject;  
  X, Y: Integer; State: TDragState;  
  var Accept: Boolean);  
begin  
  
end;
```

Componente donde
comenzó el arrastre



OnDragOver del ListBox

Forma General

```
procedure TForm1.ListBox1DragOver(  
  Sender, Source: TObject;  
  X, Y: Integer; State: TDragState;  
  var Accept: Boolean);  
begin  
  
end;
```

Componente que recibe la información (donde se "suelta").

OnDragOver del ListBox

Forma General

```
procedure TForm1.ListBox1DragOver(  
    Sender, Source: TObject;  
    X, Y: Integer; State: TDragState;  
    var Accept: Boolean);
```

```
begin
```

```
end;
```

El evento debe analizar si la componente receptora está dispuesta a aceptar la información.



OnDragOver del Listbox (de quien recibe)

```
procedure TForm1.ListBox1DragOver(  
    Sender, Source: TObject;  
    X, Y: Integer; State: TDragState;  
    var Accept: Boolean);  
begin  
    Accept := Source is TLabelEdit;  
end;
```

Si Accept := TRUE aceptará de cualquier componente.

OnDragDrop

(qué se hace con la información)

```
procedure TForm1.ListBox1DragDrop(  
    Sender, Source: TObject; X, Y: Integer);  
Var Linea : TLabeledEdit;  
    Lista : TListBox;  
begin  
    Linea := Source as TLabeledEdit;  
    Lista := TListBox( Sender);  
    Lista.Items.add( Linea.Text );  
    Linea.Text := " ;  
end;
```

A yellow box labeled "Dónde comenzó el arrastre" (Where the drag started) has two arrows pointing to the "Source" parameter in the procedure signature and the "Source" variable in the assignment statement "Linea := Source as TLabeledEdit;".

OnDragDrop

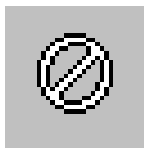
(qué se hace con la información)

```
procedure TForm1.ListBox1DragDrop(  
    Sender, Source: TObject; X, Y: Integer);  
Var Linea : TLabelEdit;  
    Lista : TListBox;  
begin  
    Linea := Source as TLabelEdit;  
    Lista := TListBox(Sender);  
    Lista.Items.add( Linea.Text );  
    Linea.Text := "";  
end;
```

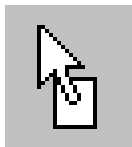
A diagram illustrating the flow of information. A pink oval highlights the `Sender` parameter in the procedure signature. A line connects this oval to a pink box labeled "Componente que recibe". Another line connects this box to the `Sender` parameter in the `TListBox(Sender)` constructor call within the procedure body.

Drag & Drop

- Verificar el funcionamiento de la aplicación.
- Puede ver que durante el arrastre aparecen dos tipos de cursor:



- Sobre las componentes no habilitadas.



- Sobre el ListBox (el único componente habilitado)

Modificación para borrar las líneas del ListBox



- Modificar el ejercicio anterior para permitir seleccionar una línea del ListBox que será borrada al ser arrastrada sobre el formulario.
- Ver que en el ListBox la propiedad `DragMode := dmAutomatic`



OnDragOver del Formulario (sólo recibe del ListBox)

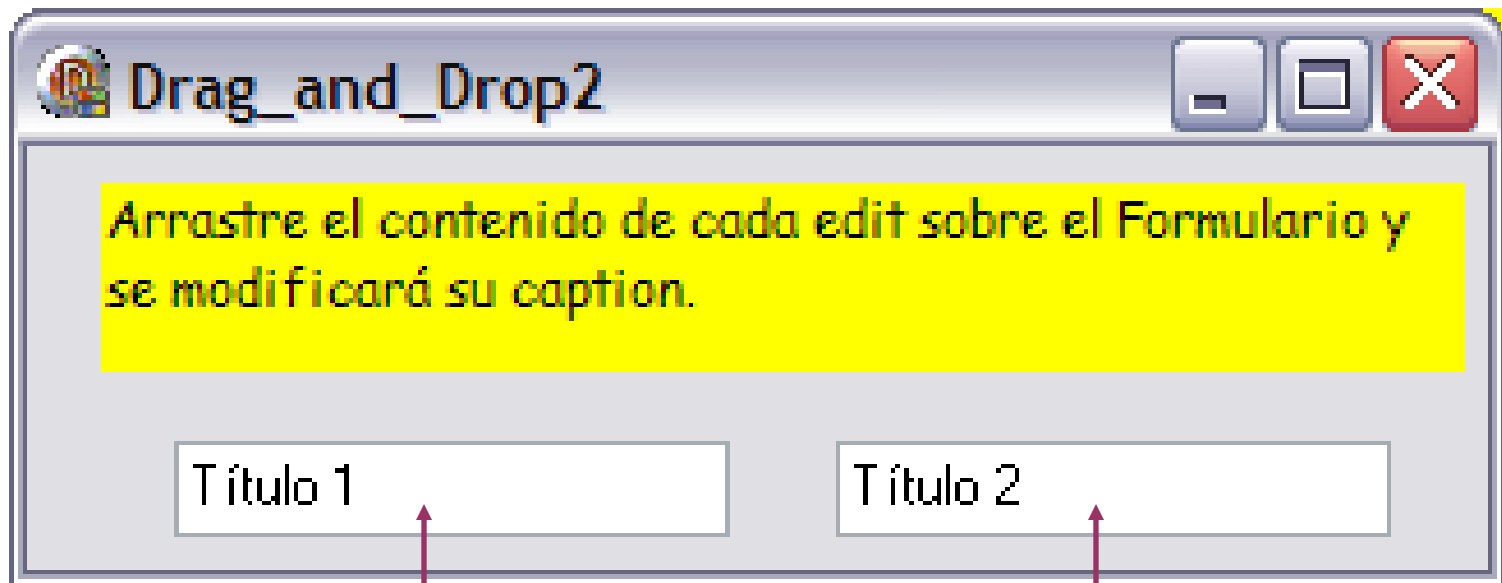
```
procedure TForm1.FormDragOver(  
    Sender, Source: TObject;  
    X, Y: Integer;  
    State: TDragState;  
    var Accept: Boolean);  
begin  
    Accept := Source is TListBox;  
end;
```



OnDragDrop del Formulario (borra la línea seleccionada)

```
procedure TForm1.FormDragDrop(  
    Sender, Source:TObject;  
    X,Y:Integer);  
  
begin  
    { borra la línea seleccionada }  
    (Source as TListBox).DeleteSelected;  
end;
```

Ejemplo : Drag_and_Drop2.dpr





Evento DragOver del Formulario

```
procedure TForm1.FormDragOver(Sender,  
    Source: TObject; X, Y: Integer;  
    State: TDragState; var Accept: Boolean);  
begin  
    //decir de quién recibe el formulario  
    Accept := Source is TEdit;  
end;
```



Evento DragDrop del Formulario

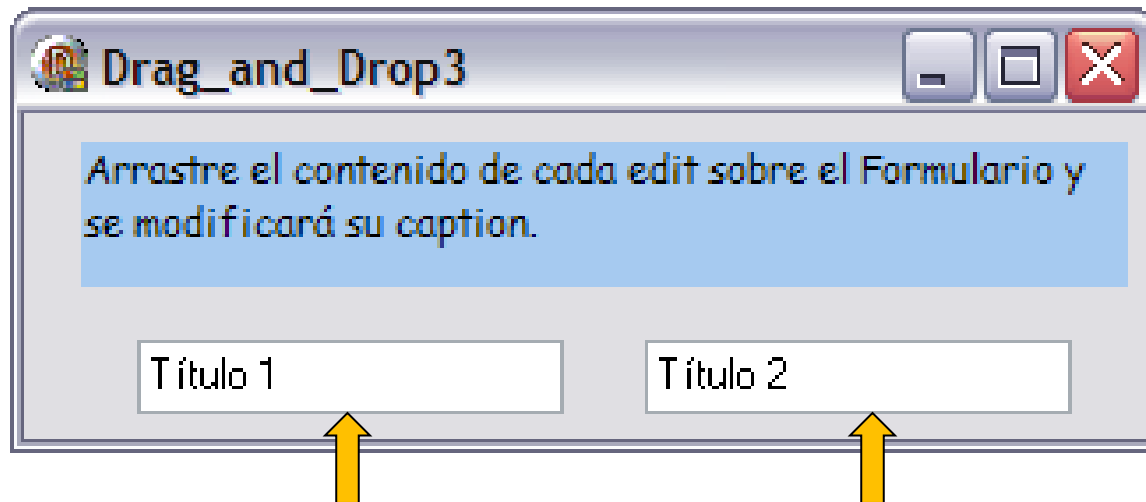
```
procedure TForm1.FormDragDrop(  
    Sender, Source: TObject; X, Y: Integer);  
begin  
    //cómo se hace el arrastre?  
    Form1.Caption:=(Source as TEdit).Text;  
end;
```



Inicio del Drag & Drop

- El uso de la propiedad DragMode para iniciar el arrastre puede interferir con el funcionamiento normal del mouse.
- Ej: en un Edit no se puede usar el mouse para pasarle el foco.
- Es conveniente dejar DragMode en dmManual y usar los eventos que responden al mouse para iniciar el arrastre con el método **BeginDrag**

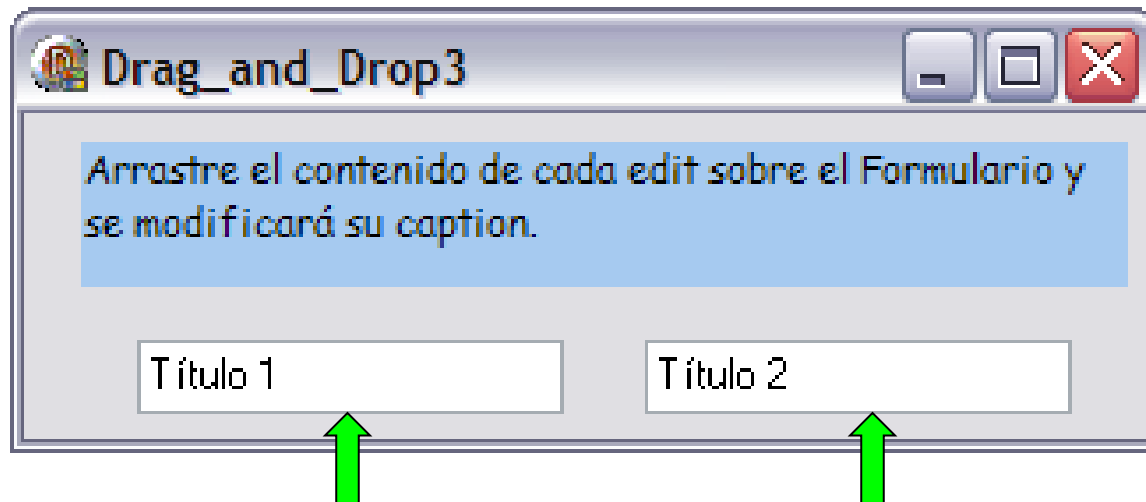
Ejemplo : Drag_and_Drop3.dpr



Propiedad

Dragmode = dmManual

Ejemplo : Drag_and_Drop3.dpr



Evento OnMouseDown
`Edit1.BeginDrag(False)`

Evento OnMouseDown
`Edit2.BeginDrag(False)`

Veamos los parámetros del método **BeginDrag**



Método BeginDrag

- Comienza el arrastre desde un control cuando la propiedad DragMode es dmManual.

```
procedure BeginDrag(Immediate: Boolean;  
                    Threshold: Integer = -1);
```

- Si Immediate = TRUE, el arrastre comienza inmediatamente.
- Si Immediate = FALSE, el arrastre comienza cuando el mouse se desplaza la cant.de pixels indicada por Threshold.



Método BeginDrag

■ Ejemplos

- `Edit1.BeginDrag(TRUE);`
- `Edit1.BeginDrag(FALSE);`
- `Edit1.BeginDrag(FALSE, -2);`
- `Edit1.BeginDrag(FALSE, 10);`

El arrastre se inicia inmediatamente



Si el 1er. parámetro es FALSE, el arrastre NO comienza inmediatamente



Método BeginDrag

■ Ejemplos

- `Edit1.BeginDrag(TRUE);`
- `Edit1.BeginDrag(FALSE);`
- `Edit1.BeginDrag(FALSE, -2);`
- `Edit1.BeginDrag(FALSE, 10);`

- El 2do.param indica la cantidad de pixels que debe desplazarse el mouse para comenzar el arrastre.
- Es opcional.
- Si no está o es <0 , se toma el valor por defecto.