Capítulo 3. Extensión de componentes

Te doy gracias vida, por tenerme aquí
Aunque tus senderos cuesten para mí
Y con mi trabajo poder demostrar
Que todo se puede, no hay que claudicar
[Ramírez, 2003]

Capítulo 3. Extensión de componentes

Una de las ventajas que ofrece Delphi es la de poder crear nuestros propios componentes o extender aquellos que ya existen, pues Delphi ofrece los códigos fuentes de sus componentes y pueden modificarse. Para cumplir con el objetivo de MexVox es necesario que extendamos algunos de los componentes y hacerlos que "hablen" para lo cual les asignaremos "voz" y algunas otras propiedades dependiendo del componente. Dentro de los componentes que se van a extender se encuentran: botón, memo, áreas de texto y listBox.

3.1 Crear un componente

Para poder crear un componente se necesita tener instalada la versión 6 o 7 de Delphi. Con el fin de que haya una mejor idea de lo que se debe hacer para extender un componente partiremos de una situación real. Vamos a extender el componente Edit para que "hable" cuando el usuario escriba en él. Para lograr nuestro propósito enumeraremos los requerimientos a cumplir:

1. Que pronuncie cada una de las letras que se van escribiendo
2. Si se borra una letra que pronuncie el nombre de la letra que fue borrada
3. Si el usuario se mueve con las flechas, de izquierda o derecha, se pronuncie la letra sobre la cual se encuentra posicionado el cursor.

Para ello necesitamos asignarle voz a nuestro componente y eso lo haremos por medio de una nueva propiedad. Una vez que conocemos las características de nuestro nuevo componente procederemos a su creación. Para ellos seguiremos los siguientes pasos:

1. Ejecutar el programa Delphi instalado
2. Dar clic en Component –> New Component
3. Aparecerá la siguiente ventana en la que debemos llenar cada uno de los siguientes campos: (ver figura 3.1)

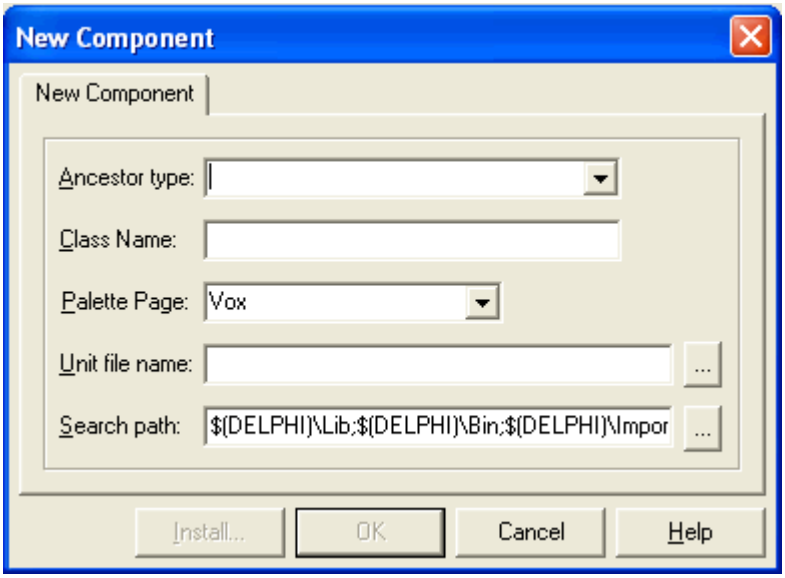

Figura 3.1 Nuevo Componente [Delphi, 2004]

- Ancestor type: Se debe elegir un componente que va a ser extendido
- Class Name: Nombre de la nueva clase que vamos a crear

- Palette page: Nombre de la barra de herramientas en la que el nuevo componente va a ser agregado
- Unit file name: nombre de la unidad y de la ruta en la que va a ser almacenado el código fuente de nuestro nuevo componente
- Path search: es la misma ruta donde fue guardado el componente
- Dar clic en OK

La figura 3.2 muestra como puede quedar el llenado de los campos si se hereda de la clase TEdit. También se puede determinar la barra de herramientas a la que se va a agregar el nuevo componente, en este caso es Vox. De igual manera se muestras la ruta de directorios de búsqueda de la clase que se va a crear y el directorio en donde se tiene almacenado el código fuente del componente recién creado.

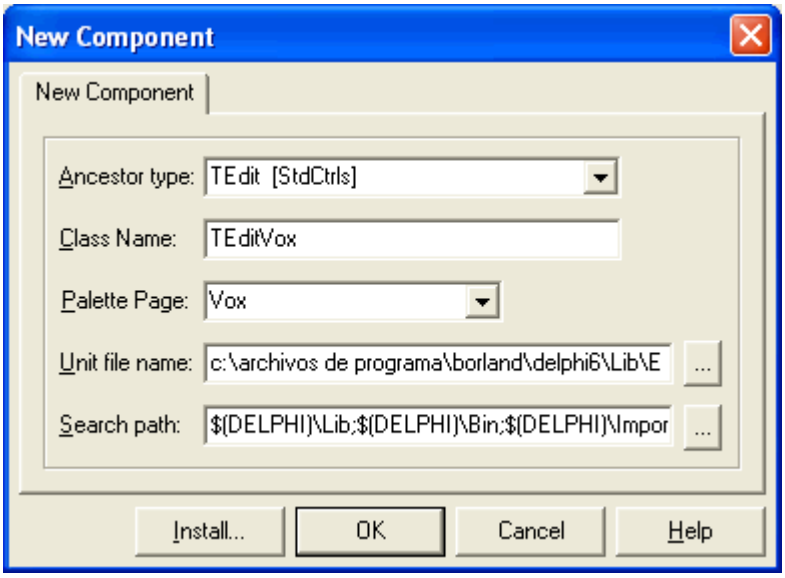

Figura 3.2 Ejemplo al crear TEditVox que hereda de TEdit [Delphi, 2004]

4. Posteriormente se presentará el código fuente de nuestro nuevo componente. En él podemos programar los procedimientos y funciones que deseemos, además de poder asignar nuevas propiedades, como es el caso de la "voz"

```
unit EditVox;
interface
uses
   Windows, Messages, SysUtils, Classes, Controls, StdCtrls;
type
  TEditVox = class(TEdit)
  private
    { Private declarations }
  protected
    { Protected declarations }
  public
    { Public declarations }
  published
    { Published declarations }
  end;
procedure Register;
implementation
procedure Register;
begin
  RegisterComponents('Vox', [TEditVox]);
end;
end.
```

5. Ahora vamos a dar una nueva propiedad a nuestro componente: "voz". Para ello haremos uso del componente Microsoft Direct Text-to-Speech (Versión 1.0) por lo que necesitamos incluir la librería ACTIVEVOICEPROJECTLib_TLB y declarar un objeto del tipo TDirectSS (Ver sección 3.4)

```
uses
  Windows, Messages, SysUtils, Classes, Controls, StdCtrls,
  MmSystem, ACTIVEVOICEPROJECTLib_TLB, Forms;
```

```
type
 TEditVox = class(TEdit)
  private
   { Private declarations }
     FSintetizador: TDirectSS;
```

6. El siguiente paso es declarar la nueva propiedad a la cual se le ha dado el nombre de Sintetizador

```
 published
{ Published declarations }
property Sintetizador: TDirectSS read FSintetizador write
FSintetizador;
```

7. El siguiente paso es hacer que pronuncie el sonido de la letra que fue escrita, borrada o en la que se encuentre actualmente el cursor. Para ello extenderemos el procedimiento *onKeyPress*. Este procedimiento es el encargado de "escribir" los caracteres que el usuario va tecleando. La idea es obtener el carácter que se tecleó y reproducir su sonido por medio del sintetizador.

```
protected
    { Protected declarations }
              procedure KeyPress(var Key: Char); override;

procedure TEditVox.KeyPress(var Key: Char);
begin
  inherited;
  if not (ord (key) in [$20..$7f]) then
      exit;
      sintetizador.Speak(key);
end;
```

8. Ahora reproduciremos el sonido de la letra en caso de ser borrada o si mueven el cursor por medio de las teclas de dirección

```
protected
    { Protected declarations }
    procedure KeyDown(var Key: Word; Shift: TShiftState); override;
var letra: char;
    posicion: integer;
procedure TEditVox.KeyDown(var Key: Word; Shift: TShiftState);
begin
  inherited;
  if  (key >= $20) and (key <= $7f) then
     begin
     case key of
        VK_RIGHT:  begin
              posicion := selStart + 1;
              letra := caption[posicion];
            sintetizador.Speak(letra);
              end;
        VK_LEFT:  begin
              posicion := selStart;
              letra := caption[posicion];
            sintetizador.Speak(letra);
              end;
        VK_DELETE:  begin
              posicion := selStart + 1;
              letra := caption[posicion];
            sintetizador.Speak(letra);   end;
         end;
      end
  else if key = VK_BACK  then
              if selStart > 0 then
                 begin
                  posicion := selStart;
                  letra := caption[posicion];
                  sintetizador.Speak(letra);
                 end
  else    exit;
end;
```

Para reproducir el sonido de las letras se puede seguir este método o hacerlo por medio de archivos de audio (.wav). Se detecta el carácter, se obtiene su valor ASCII se concatena con la extensión .wav y con la ruta del directorio en donde se encuentran almacenados los sonido y se reproducen por medio de playsound.

3.2 Instalar un componente

Después de haber programado y extendido nuestro componente debemos de instalarlo para que se pueda utilizar en el desarrollo de nuevas aplicaciones. Para ello llevaremos a cabo el siguiente procedimiento:

1. Ir al menú de la parte superior y dar clic en Component –> Install component (ver figura 3.3)

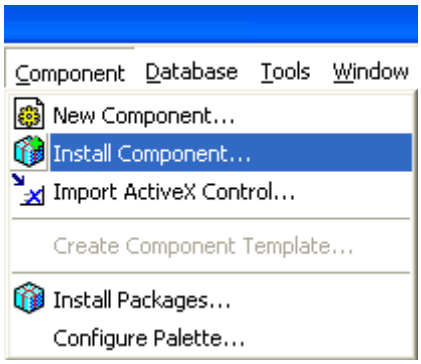

Figura 3.3 Instalar un componente [Delphi , 2004]

2. Posteriormente se mostrará una ventana (ver figura 3.4) en la que nos pide al dirección en donde se encuentra el componente que deseamos instalar.

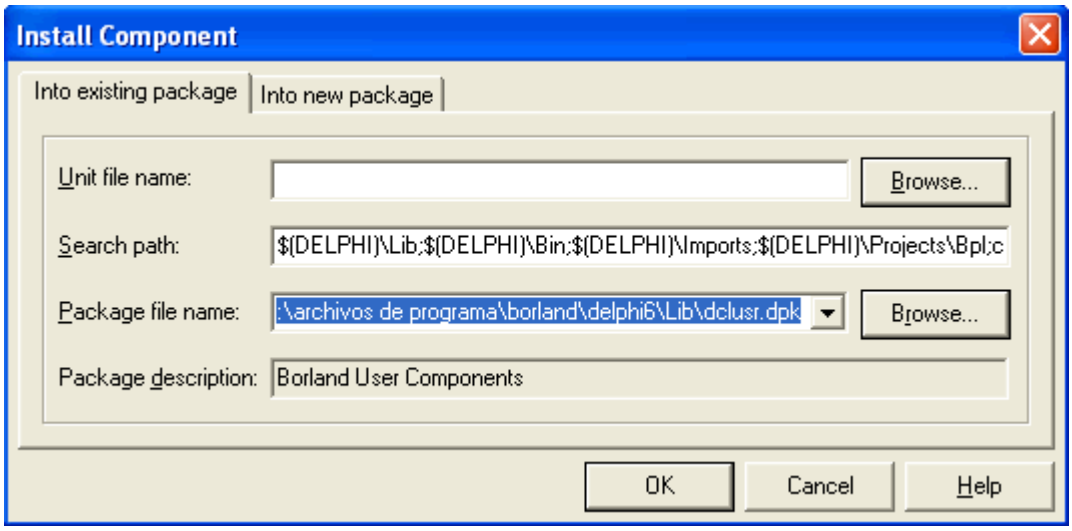

Figura 3.4 Nombre de la unidad del nuevo componente [Delphi, 2004]

3. Una vez que se hace clic en OK aparecerá la siguiente ventana (ver figura 3.5).

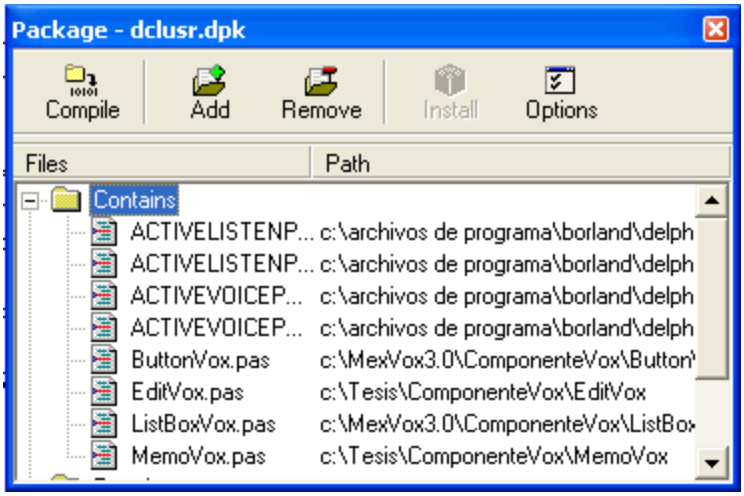

Figura 3.5 Ventana de compilación [Delphi, 2004]

Hacemos clic en el botón de Compile y nos indicará si el componente ha sido instalado o si hay algún problema.

4. Después de haber compilado sin ningún problema el componente aparecerá disponible en la barra de herramientas en la que haya sido incluido (ver figura 3.6). En el caso de nuestro ejemplo el componente EditVox se incluyó dentro de la barra de herramientas llamada Vox.

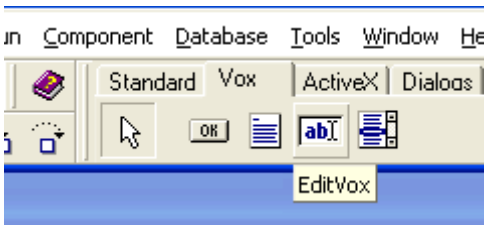

Figura 3.6 Barra de herramientas Vox con componentes creados [Delphi, 2004]

3.3 Utilizar el nuevo componente.

Haremos un pequeño ejemplo utilizando nuestro nuevo componente parlante. Vamos a crear una aplicación que únicamente va a tener un TEditVox en el cual podremos escribir y que, a su vez, pronunciará el sonido de los caracteres que se vayan tecleando.

1. Vamos al menú superior y hacemos clic en File –> New –> Application (ver figura 3.7)
2. Ya que la forma principal está disponible para empezar a trabajar vamos a buscar la barra de herramientas donde agregamos nuestro componente, en este caso nuestra barra se llama Vox. Una vez que ha sido seleccionada nos mostrará todos los componentes hemos agregado, por ejemplo, editVox. Damos clic en el icono de editVox, movemos el cursor sobre la forma principal y damos clic en el lugar en donde deseemos colocar el componente (ver figura 3.8).

3.

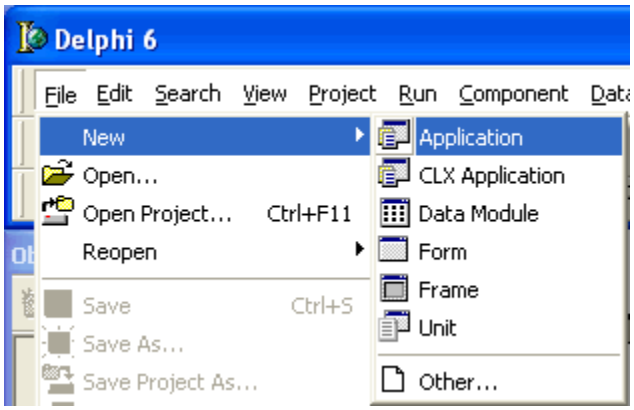

Figura 3.7 Crear una nueva aplicación [Delphi, 2004]

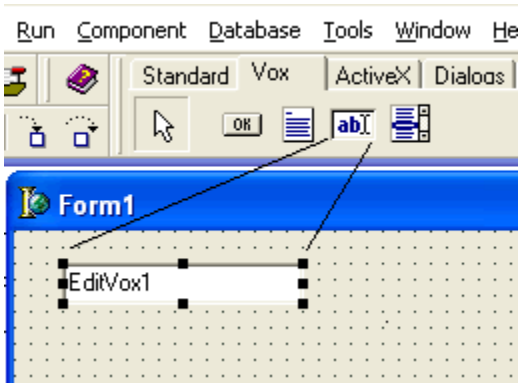

Figura 3.8 Utilizar el nuevo componente [Delphi, 2004]

4. En la parte inferior izquierda se encuentra una ventana con el nombre de *Object Inspector* y en ella se encuentran todas las propiedades y eventos relacionados con el objeto seleccionado en la forma principal. Es posible hacer cambios en cualquier propiedad que el programador desee. En las propiedades del componente editVox vamos a buscar la propiedad llamada *Name* y vamos a darle el nombre de: "miEditVox" (ver figura 3.9).

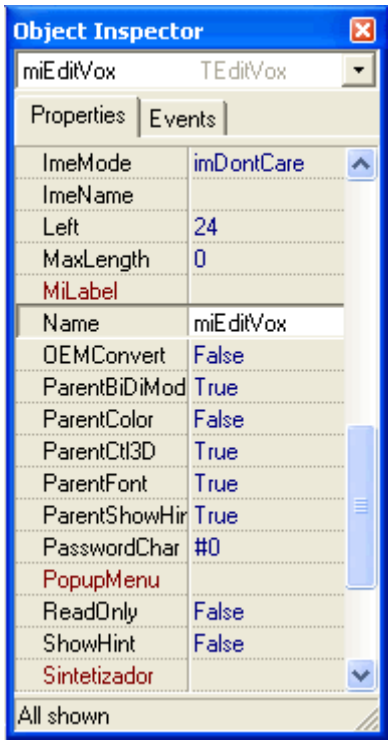

Figura 3.9 Ventana de propiedades y eventos de componentes [Delphi, 2004]

5. Ahora necesitamos agregar un objeto del tipo DirectSS, que funcionará como la voz de nuestro componente. DirectSS se encuentra en la barra de herramientas de nombre Active X y esta representado por una pequeña boca (ver figura 3.10). Se debe seguir el mismo procedimiento que en EditVox para agregarlo a la forma principal y le vamos a dar el nombre de: “primeraVoz”
6. Hacemos clic cobre el componente editVox. En object inspector vamos a buscar la propiedad que le asignamos a nuestro componente a la cual le dimos el nombre de sintetizador. Una vez que la encontramos hacemos clic sobre la flecha y nos mostrará los componentes DirectSS que hayamos agregado a la forma. En este caso únicamente hemos agregado un componente y le dimos el nombre de: “primeraVoz”. Seleccionamos nuestro sintetizador de voz (ver figura 3.11).

7.

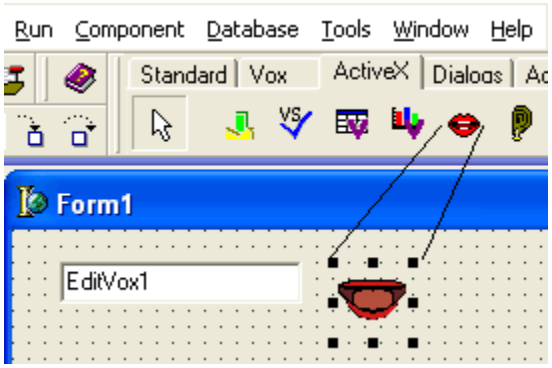

Figura 3.10 Agregar sintetizador de voz a la aplicación [Delhi, 2004]

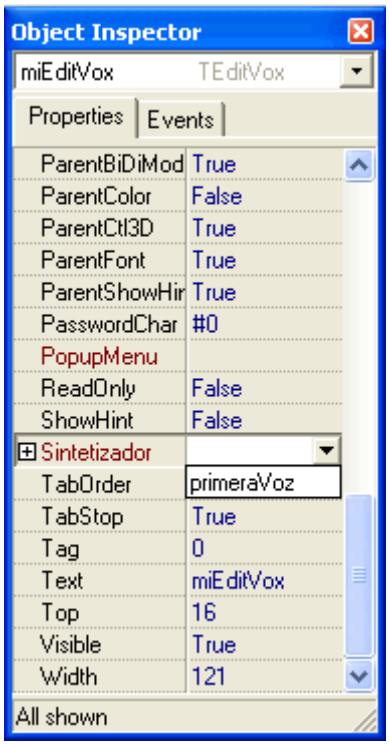

Figura 3.11 Asignar el sintetizador como una propiedad de TEditVox [Delhi, 2004]

8. Hacemos clic en Run –> Run y probamos nuestra aplicación. Debe de producir el sonido de los caracteres que tecleemos, al igual que si borramos uno de ellos o si movemos el cursor por medio de las flechas de selección.

Figura 3.12 Ejecución de la aplicación [Delphi, 2004]

En este ejemplo en específico la aplicación debe producir una señal audible que representa el sonido de la tecla "a", la cual, fue escrita dentro del área de texto (ver figura 3.12).

3.4 Instalación del componente DirectSS

DirectSS es un componente ActiveX que no viene incluido en las herramientas de trabajo en Delphi. Es necesario importar el componente para que pueda ser utilizado.

Procedimiento para instalar DirectSS:

1. Component –> Import ActiveX Control. (ver figura 3.13)
2. Se mostrará una ventana con el título: Import ActiveX y en ella una lita de todos los componente que pueden ser importados (ver figura 3.14).

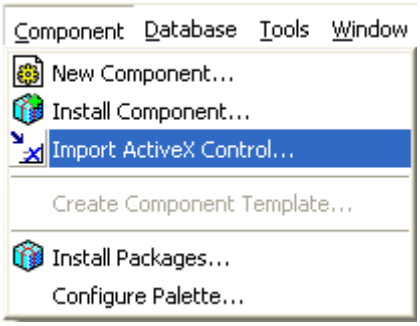

Figura 3.13 Importar componente Active X [Delphi, 2004]

3. De la lista se debe seleccionar el componente con el nombre: Microsoft Direct Text-to-Speech (Versión 1.0) (ver figura 3.14).

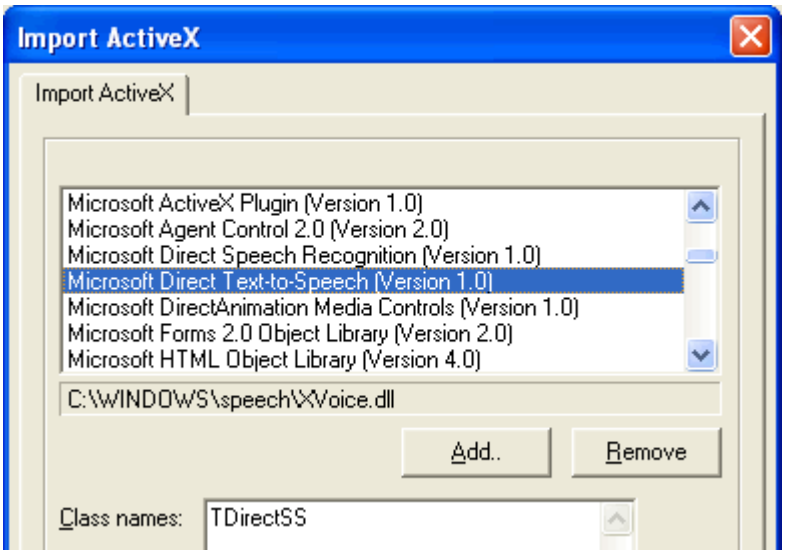

Figura 3.14 Importar componente Active X [Delphi, 2004]

4. Hacer clic en el botón de install en la parte inferior de la ventana
5. Hacer clic en las ventanas de confirmación de instalación.
6. Posteriormente se mostrará otra ventana con el nombre de: Package - dclusr.dpk (ver figura 3.15)

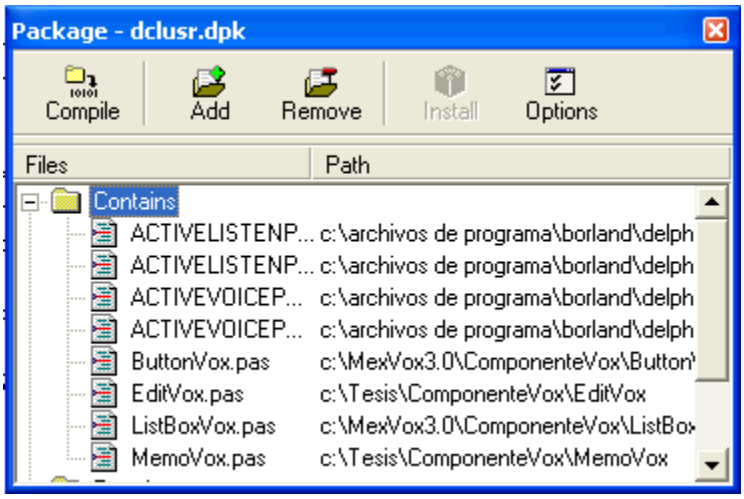

Figura 3.15 Ventana de compilación [Delphi, 2004]

7. Hacer clic en Compile
8. Una vez que se ha compilado el paquete el componente DirectSS debe ser agregado en la barra de herramientas ActiveX. Se puede identificar ya que esta representado por el icono de una boquita (ver figura 3.16).

Figura 3.16 Componente incluido en la barra de componente Active X [Delphi, 2004]

9. Una vez que ha sido instalado el componente DirectSS puede ser utilizado en la elaboración de cualquier aplicación.