Capítulo 3

Usando GUIDE

3.1 Acerca de GUIDE

Las interfaces gráficas de usuario (GUI - Graphical User Interface en inglés), es la forma en que el usuario interactúa con el programa o el sistema operativo de una computadora. Una GUI contiene diferentes elementos gráficos tales como : botones, campos de texto, menús, gráficos, etc.

Existen diferentes leguajes de programación que permiten crear una GUI tales como: C, Visual Basic, TK, etc. solo por mencionar algunos. Todos ellos permiten usar diferentes controles y maneras de programarlos. MatLab nos permite realizar GUIs de una manera muy sencilla usando GUIDE (Graphical User Interface Development Enviroment).

3.1.1 Creando una GUI en MatLab

Una de las tantas herramientas con la que cuenta MatLab, es la creación de GUIs. La forma de implementar las GUI con MatLab es crear los objetos y definir las acciones que cada uno va a realizar. Al usar GUIDE obtendremos dos archivos:

- Un archivo FIG – Contiene la descripción de los componentes que contiene la interfase.
- Un archivo M – Contiene las funciones y los controles del GUI así como el callback

Un callback se define como la acción que llevará a cabo un objeto de la GUI cuando el usuario lo active. Para ejemplificarlo, suponga que en una ventana existe un botón el cual al presionarlo ejecutará una serie de acciones, a eso se le conoce como la función del callback.

3.2 Iniciando GUIDE

Para crear una GUI en MatLab usamos GUIDE, ya sea que tecleemos guide en la ventana de comandos de MatLab o lo ejecutemos del menú principal File –> New -> GUI (Fig 3.2.1).

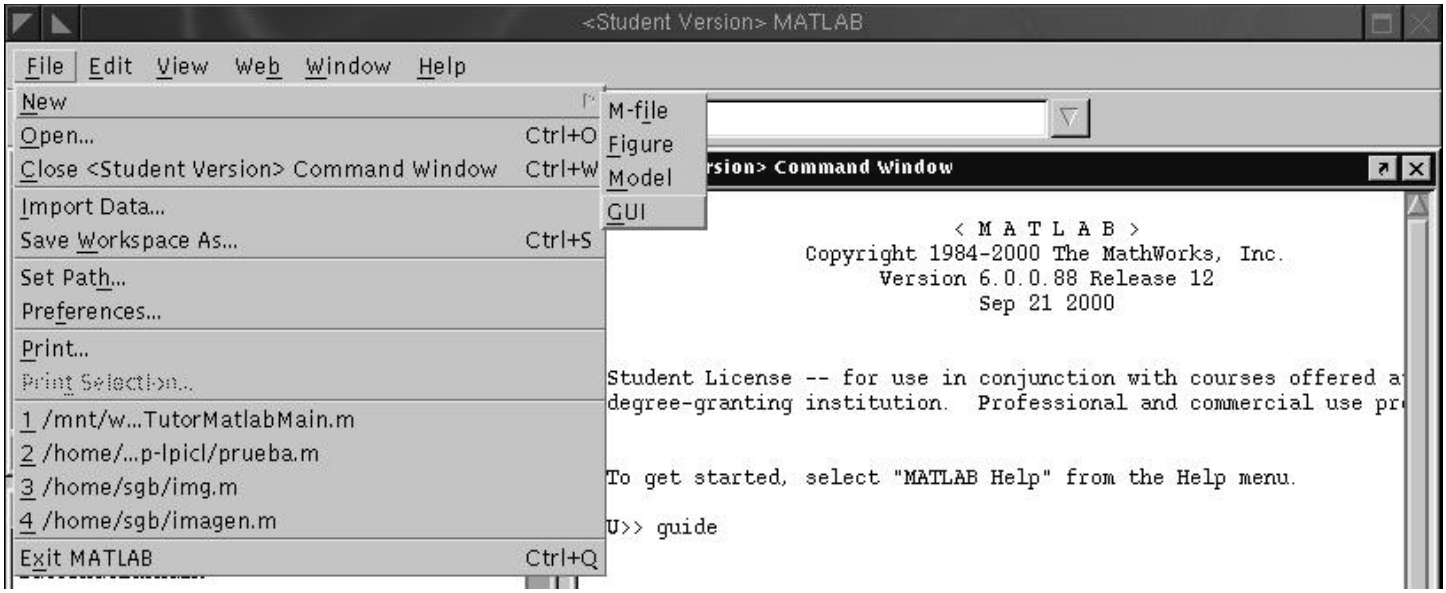

Fig. 3.2.1 – Ejecución de GUIDE usando el menú principal o la ventana de comandos.

Una vez hecho lo anterior MatLab nos mostrará una área de diseño similar a la de la figura 3.2.2. En la parte superior se encuentran los menús y opciones de GUIDE, en la parte izquierda se aprecian los diferentes controles y en la parte central el área de diseño donde pondremos los controles a usar.

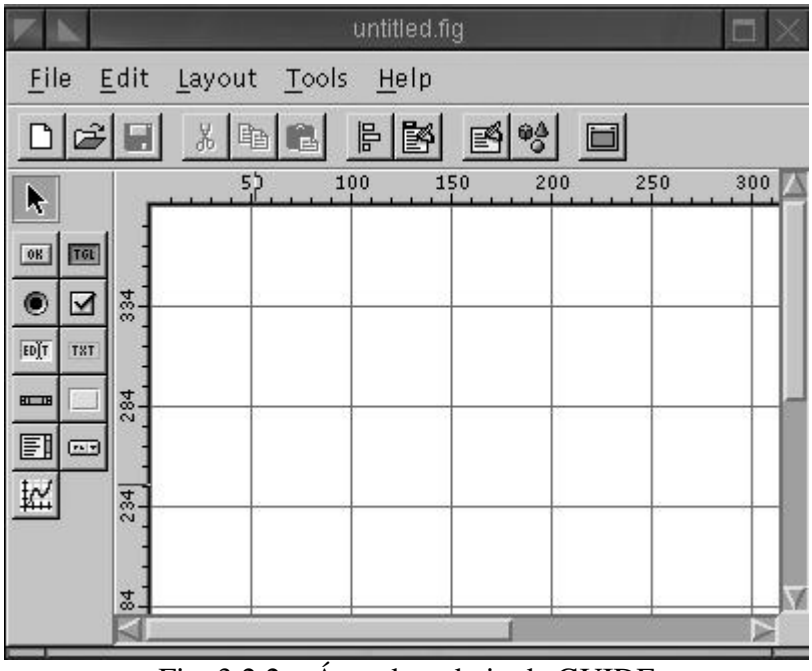

Fig. 3.2.2 – Área de trabajo de GUIDE

3.2.1 Partes de GUIDE

De todo lo anterior mencionaremos las partes más importantes de GUIDE:

 Inspector de propiedades – Cada control cuenta con diferentes propiedades y es aquí donde podremos cambiar: el color, el nombre, el tag, el valor, el callback entre otros.

 Activar Figura – Una vez que hayamos terminado de diseñar presionamos este botón para activar la figura y poder probar nuestra GUI.

 Push Button – Crea un botón

 Radio Button – Crea un botón circular

 Edit Text – Crea una campo de texto

 Axes – Crea una área para gráficas.

 Frame – Crea un marco que puede contener otros controles

 Static Text – Crea un letrero

3.2.2 Propiedades de los controles

Para entender las propiedades de un control primero crearemos un botón y luego activamos el inspector de propiedades (Fig. 3.2.2.1)

Como se ha mencionado las propiedades varían dependiendo del control a usar, a continuación se explican las más comunes:

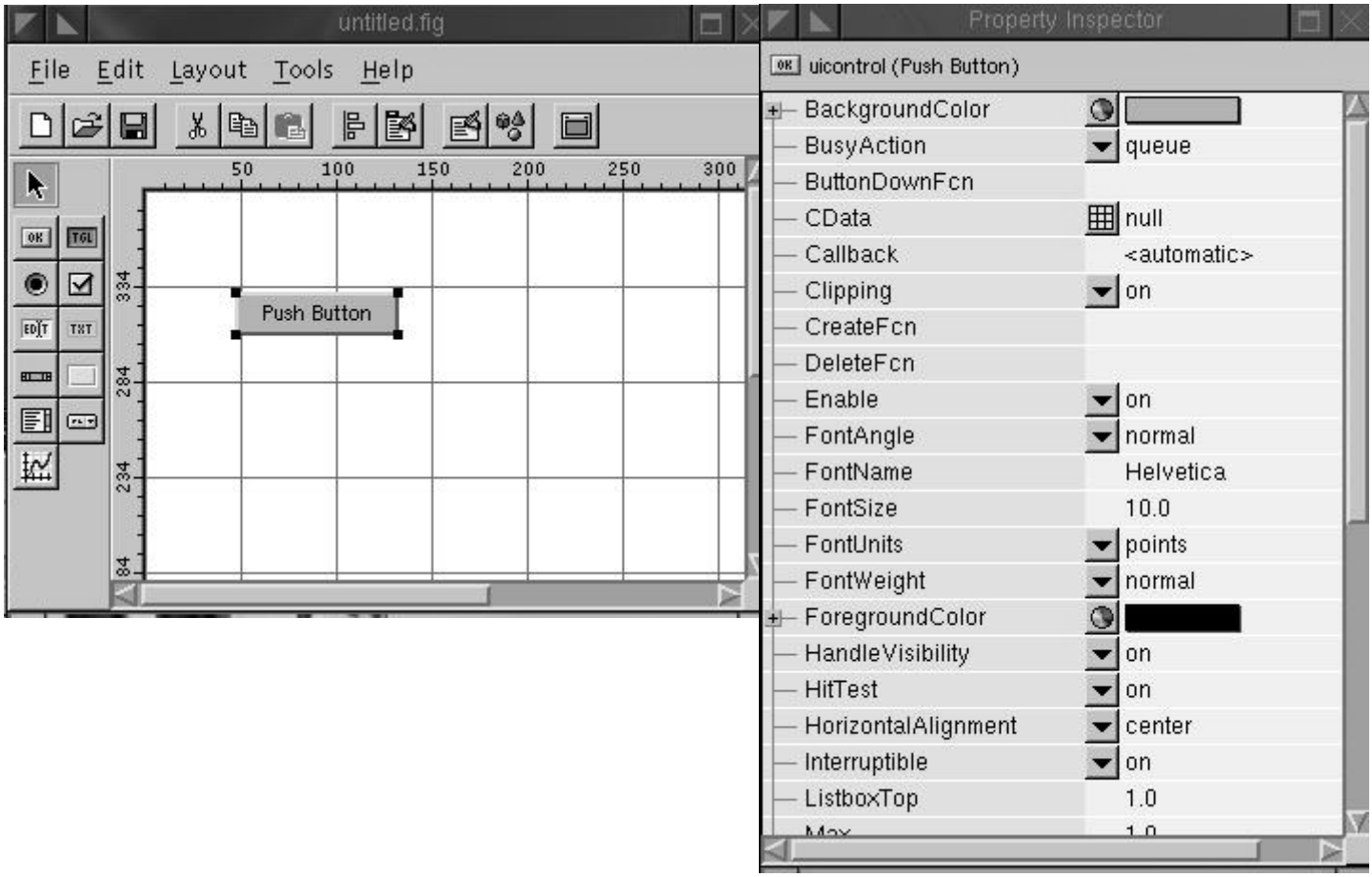

Fig. 3.2.2.1 – Propiedades del control botón

Background Color – Cambia el color del fondo del control

CallBack – La propiedad más importante del control, ya que le dice al control que hacer cuando este se active

Enable – Activa o desactiva un control

String – En el caso de botones, cajas de texto, texto estático; es el texto que muestra el control

Tag – Otra de las propiedades más importantes ya que con este es posible regresar datos o identificar al control

3.2.3 Hola Mundo en GUIDE

Quizá uno de los ejemplos más simples es el de Hola Mundo en cualquier lenguaje de programación. Para ello creamos una nueva figura y agregamos un Static Text y un Push Button (Fig. 3.2.3.1) y lo guardamos como HolaMundo.Fig

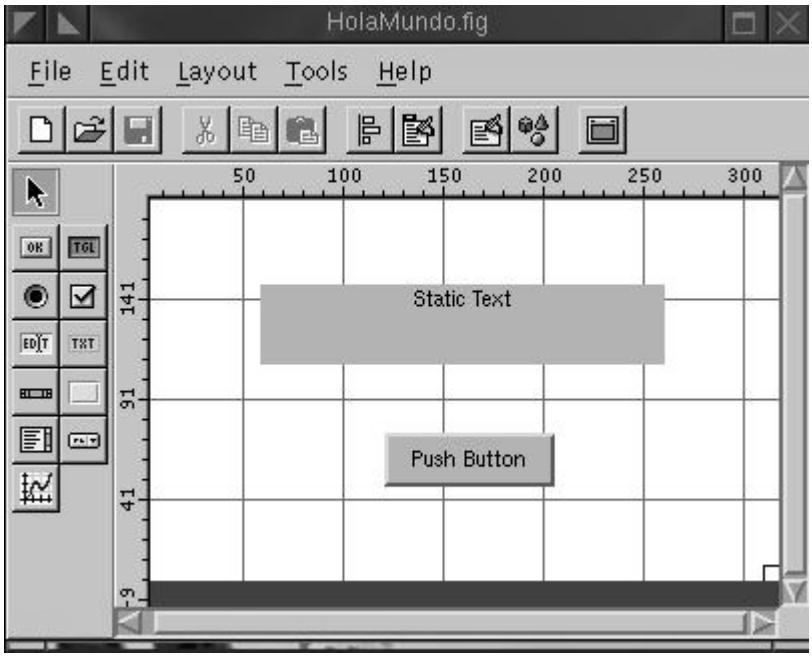

Fig. 3.2.3.1 – Ejemplo de HolaMundo

GUIDE automáticamente nos mostrará una ventana nueva con el archivo HolaMundo.m y con un código que igualmente es generado por GUIDE. Si se observa el código notará que hay líneas con % , lo cual indica que se trata de un comentario en el script.

Ahora seleccione el Static Text y active el Inspector de Propiedades y modifique la propiedad String por: Hola Mundo GUIDE (Fig. 3.2.3.2) Repita la misma operación para el Push Button pero en String ponga: Salir (Fig. 3.2.3.3)

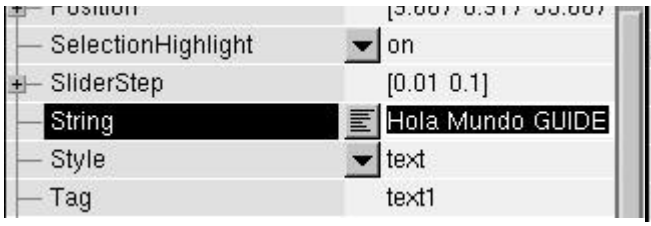

Fig. 3.2.3.2 – Propiedad modificada en Static Text

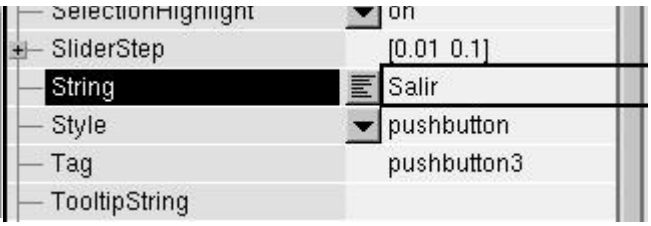

Fig. 3.2.3.3 – Propiedad modificada en Push Button

Como siguiente paso vamos a editar el callback de nuestro botón Salir para que al ser activado cierre la ventana. Para ello seleccionamos el control con el botón derecho del ratón y escogemos Edit CallBack (Fig. 3.2.3.4)

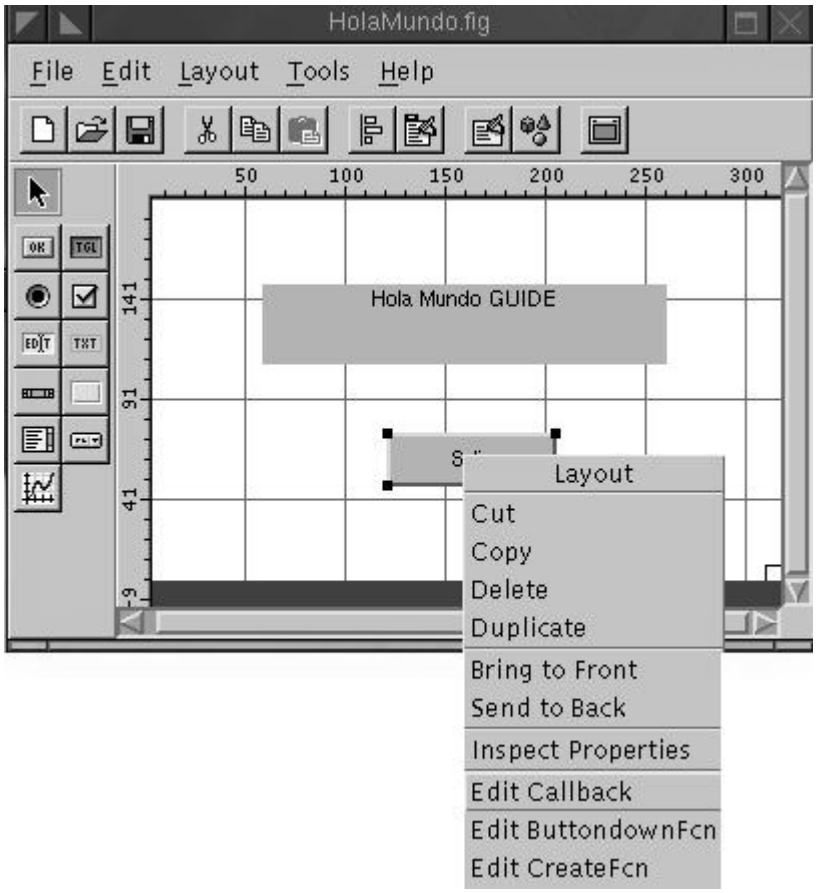

Fig. 3.2.3.4 – Editando el CallBack del botón Salir

GUIDE nos mostrará el editor de funciones del archivo HolaMundo.m en donde escribimos nuestras instrucciones, para cerrar una figura en MatLab de la forma más sencilla es agregar :

close(gcbf)

Lo cual indica que cerraremos una figura de tipo grafico. La manera de agregarlo en nuestra función se aprecia en la figura 3.2.3.5, una vez guardado el archivo HolaMundo.m activamos la figura y tendremos una salida como se muestra en la figura 3.2.3.6.

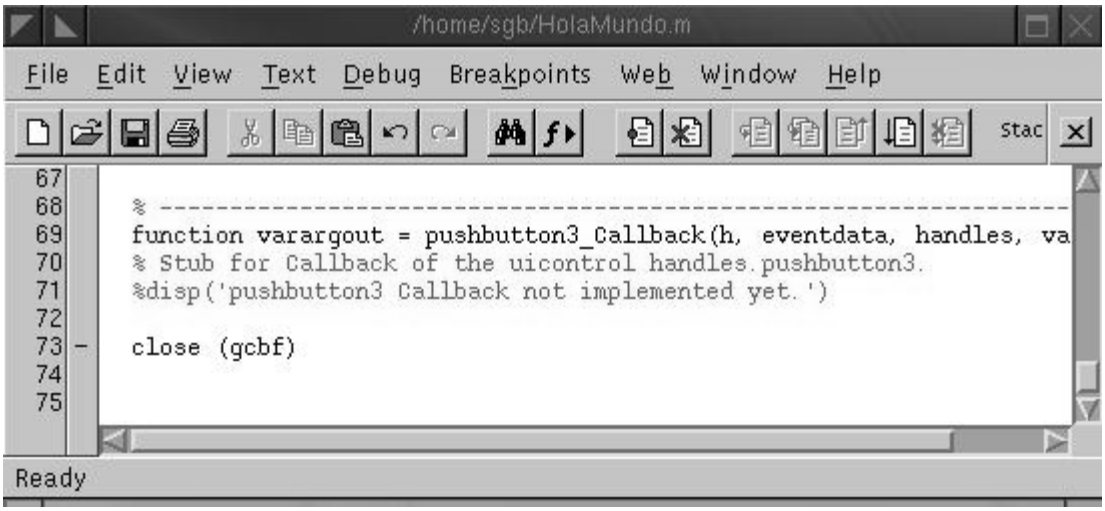

Fig. 3.2.3.5 – Agregando código a la función del botón Salir

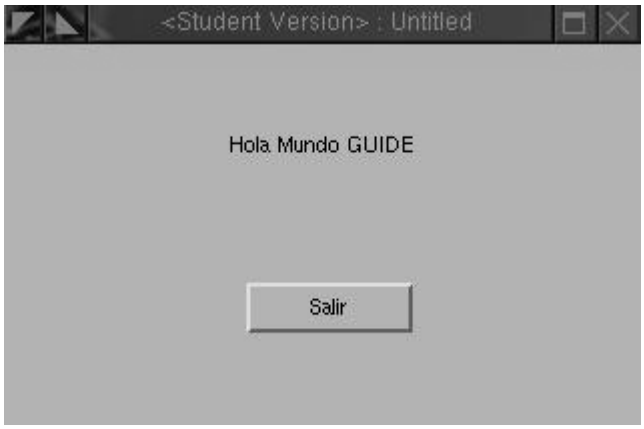

Fig. 3.2.3.6 - Salida de Hola Mundo

Hay que mencionar que en este momento contamos con dos archivos: HolaMundo.fig y HolaMundo.m Ambos archivos son necesarios para poder ejecutar nuestro programa, otra manera de ejecutarlo es usando la ventana de comandos y poniendo:

EDU>>HolaMundo

Para listar el contenido podemos usar los comandos ls o dir, para saber donde estamos podemos usar el comando pwd (validos en sistemas UNIX)(Fig. 3.2.3.7)

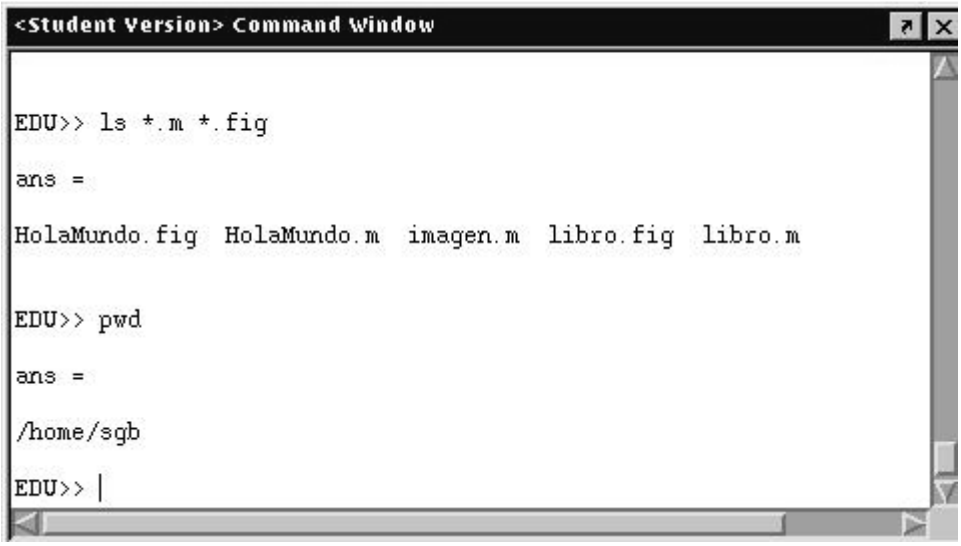

Fig. 3.2.3.7 – Uso de comandos en la ventana de comandos de MatLab

3.3 Calculadora de derivadas/integrales/fourier (Calculador DEINFU)

El siguiente proyecto nos permitirá explicar como usar los Text Box para pedir información y Static Text para desplegar resultados, así como algunas otras propiedades importantes. Para empezar creamos una nueva figura y agregamos un Static Box a un lado ponemos un Text Box y al lado del Text Box ponemos un Push Button, debajo de estos controles ponemos 6 Static Text y finalmente agregamos dos Push Buttons; y lo guardamos como: CalcDEINFU.fig, al finalizar tendremos una interface como la que se muestra en la figura 3.3.1.

En el primer Static Text cambiamos sus propiedades con el Inspector de Propiedades y modificamos el String a: Función: De ahí el Text Box cambiamos la propiedad de Strig a: x , al primer Push Button cambiamos el String a: Calcula.

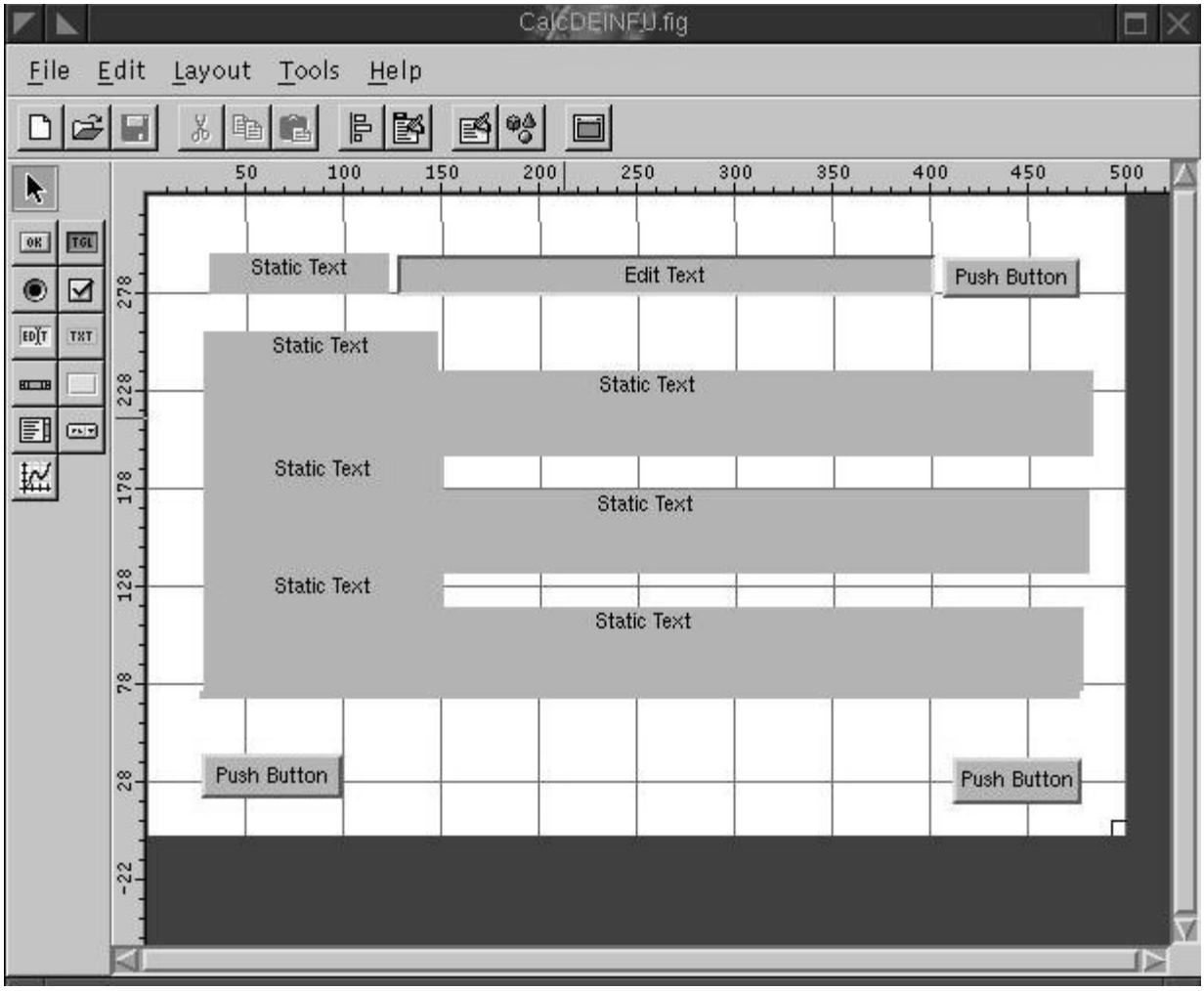

Fig. 3.3.1 – Interfase del Proyecto CalcDEINFUL

En el siguiente orden modificamos las propiedades de String de cada Static Text:

String: Derivada
String:
String: Integral
String:
String: Fourier
String:

Finalmente para el Push Button del lado izquierdo modificamos la propiedad String a: Limpia y al Push Button del lado derecho String a: Salir. Finalmente obtendremos los controles como en la figura 3.3.2.

Ahora vamos a editar el callback del botón salir y agregamos el código necesario para cerrar la ventana, luego editamos el callback del botón Limpia, cuya acción es limpiar la caja de texto de la función y los resultados de cada operación.

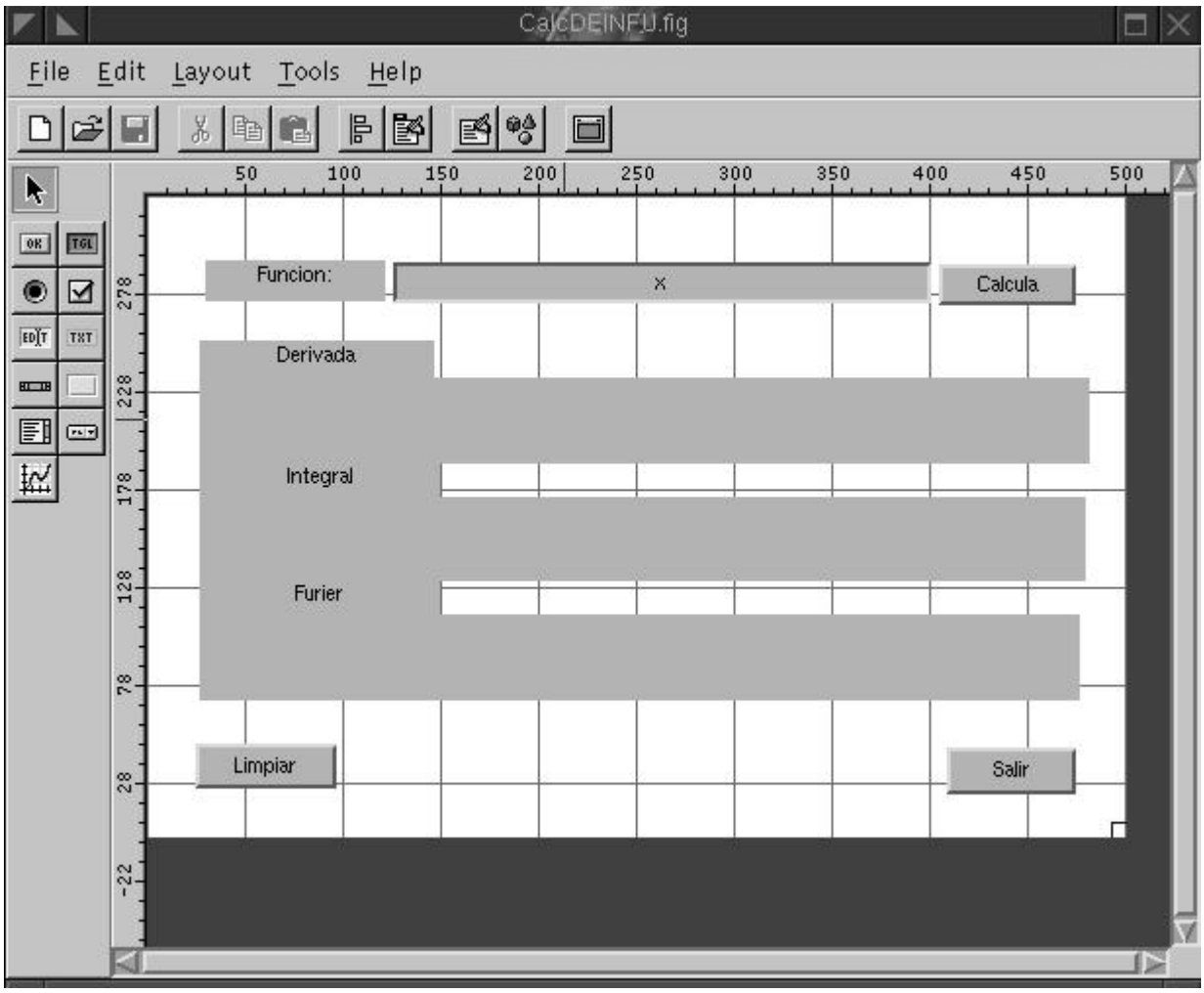

Fig. 3.3.2 – Propiedades String modificadas en cada control.

Para hacer ello hay que considerar agregar un identificador a cada Tag de cada control, esto es, poner un nombre a cada control para luego llamarlo e identificarlo. Para ello seleccionamos Tag en el Inspector de propiedades. Primero editamos el Tag de nuestro control de la caja de texto por: la_ecuacion (Fig. 3.3.3) Y para los Static Text que no tienen texto modificamos sus Tag en el siguiente orden por:

res_deriva

res_integra

res_furier

Fig. 3.3.3 – Editando Tag con Inspector de Propiedades

Ahora es posible saber que va a pasar con cada Static Text al presionar el botón Limpia. El algoritmo en general para limpiar sería:

1 – Definir variables

2 - Limpiar Variables

3 - Buscar los controles con el Tag

4 – Definir limpia

5 - Limpiar controles

Aquí hay que introducir el uso de variables globales y locales. Las globales se refieren a que pueden ser compartidas en cualquier momento por otro script. Las locales son las que se definen en un mismo momento y pueden ser usadas por el mismo script. Existen un comando de importancia para ver las variables locales y globales el cual es who, en la figura 3.3.4 se observa como usar who para ambos tipos de variables.

La sintaxis en general para activar un control es:

```
Variable =  findobj ( gcbf, 'Tag', 'Nombre-de-Tag' ) ;
```

Lo cual nos indica que primero crea una variable local la cual contiene las propiedades del objeto cuyo Tag es: Nombre-de-Tag, con esto ya hemos activado nuestro control, ahora hay que regresar la acción; para ello la sintaxis es:

```
set ( Variable , ' Propiedad' , Operación ) ;
```

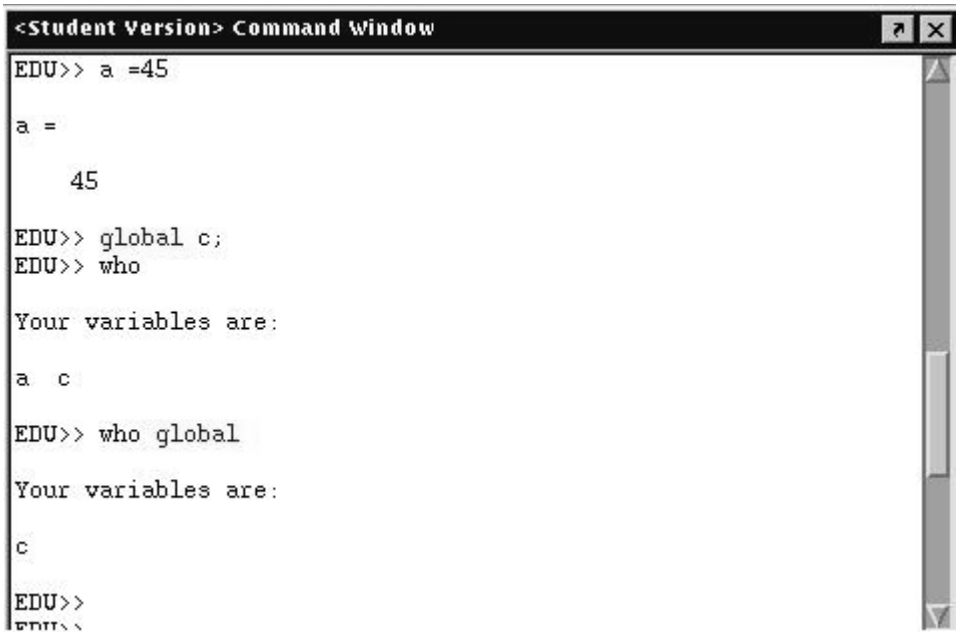

Fig. 3.3.4 – Obteniendo variables locales y globales

Está instrucción nos dice que va a usar la variable local que contiene las propiedades del objeto y va a modificar otra propiedad (String, Enable, BackColor, etc.) con un determinado valor. Para ejemplificar esto vamos a editar el CallBack del botón Limpia.

Limpio variables locales y globales

```
clear;
clear global;
```

Obtengo el control de la derivada

```
para_derivada = findobj (gcbf, 'Tag', 'res_deriva' );
```

Limpio el control

```
limpia = ' ';
set (para_derivada, 'String', limpia );
```

Y realizamos lo mismo para los demás controles, el listado completo se observa en la figura 3.3.5.

Hasta el momento no hemos pedido datos o información, el algoritmo para obtener la ecuación es:

1 – Defino variables

2 – Obtengo datos (Ecuación)

Para lograr ello se usa la siguiente instrucción:

Variable = eval (get (gcbo, 'String'));

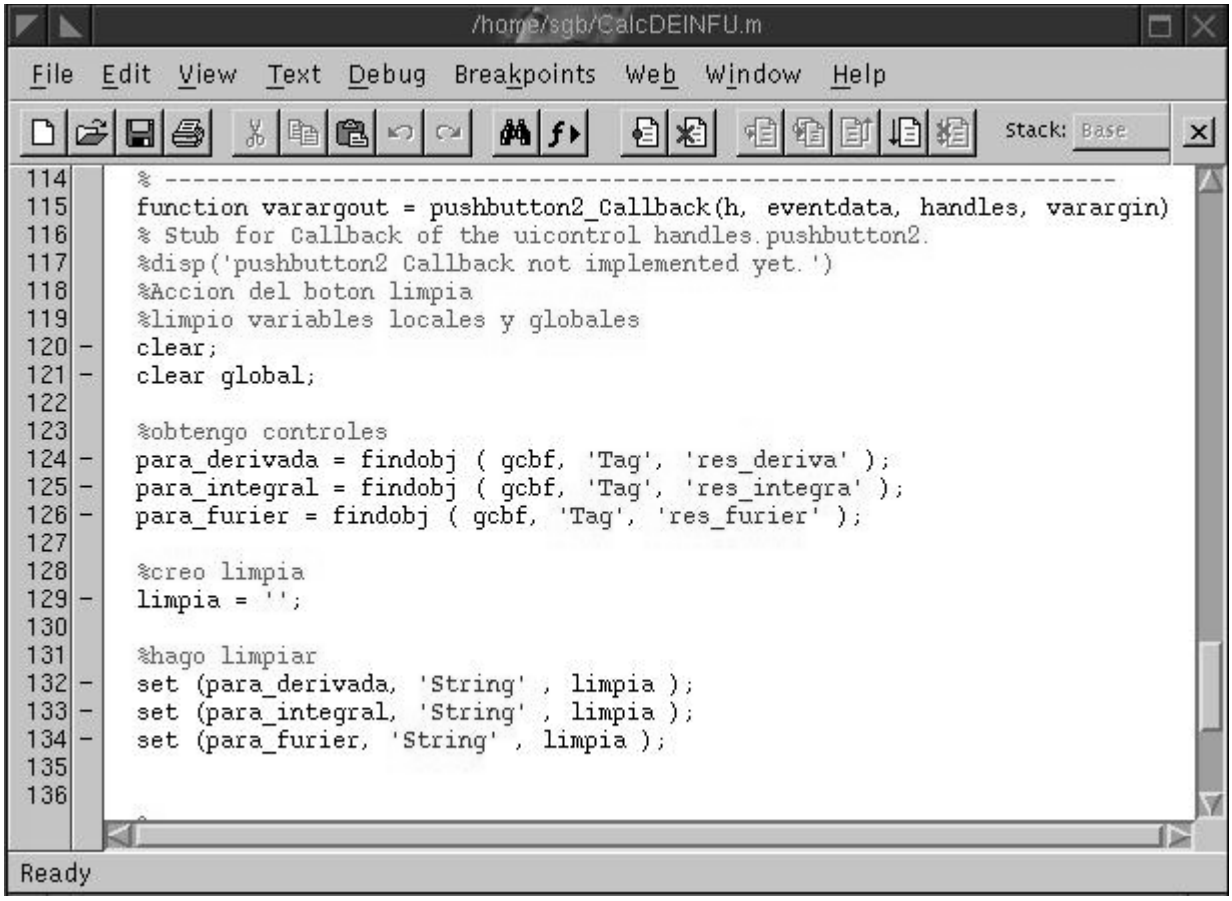

Fig. 3.3.5 – Código completo del botón Limpia

Lo que significa que creamos una variable donde vamos a almacenar la información que le llegue a la caja de texto para luego utilizarla. Hay que recordar que aquí es necesario definirla como una variable de tipo global.

El código completo de la caja de texto se observa en la figura 3.3.6.

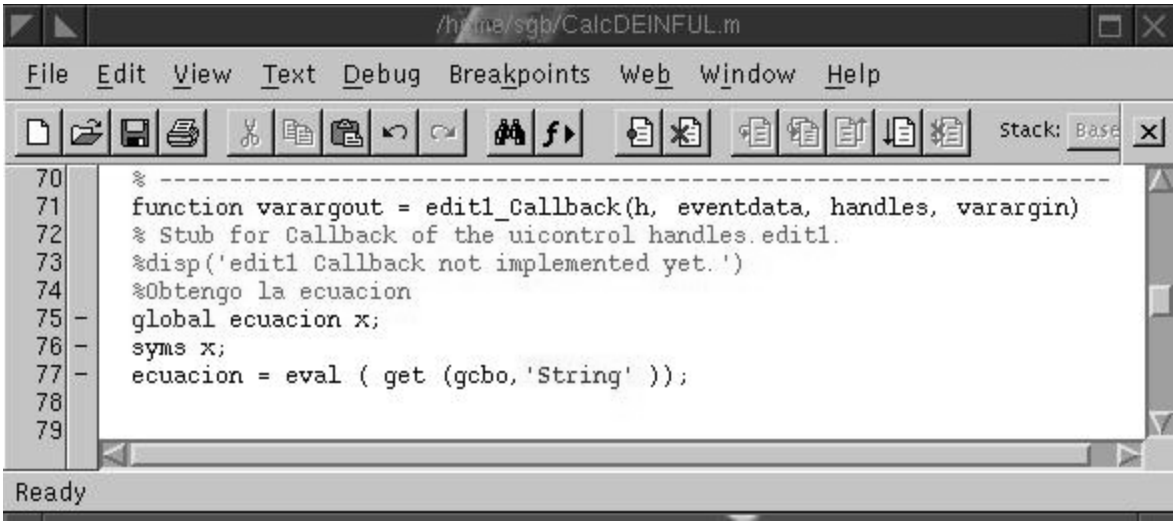

Fig. 3.3.6 – Código de la caja de texto

Finalmente llevaremos a cabo las operaciones correspondientes, las que se ejecutan al presionar el botón Calcula. Para ello editamos el callback según el siguiente algoritmo:

1 - Obtengo variables de la ecuación

2 – Busco controles con el Tag

3 – Hago cálculos

4 – Presento resultados

Usando las instrucciones:

Preparamos las variables a usar:

```
global ecuacion x;
syms x;
```

Obtengo el control de la derivada

```
para_derivada = findobj ( gcbf, 'Tag', 'res_deriva' );
```

Realizo el calculo de la derivada:

```
derivada1 = diff (ecuacion);
```

Ahora preparo el resultado para poderlo desplegar en forma de carácter usando la instrucción char()

```
derivada2 = char (derivada1);
```

Finalmente despliego el resultado de la derivada:

```
set (para_derivada, 'String', derivada2 );
```

Y realizo la misma operación para los otros dos cálculos. En la figura 3.3.7 se aprecia el código completo del botón Calcula.

Para probar nuestra aplicación activamos la figura y tendremos una salida como en la figura 3.3.8

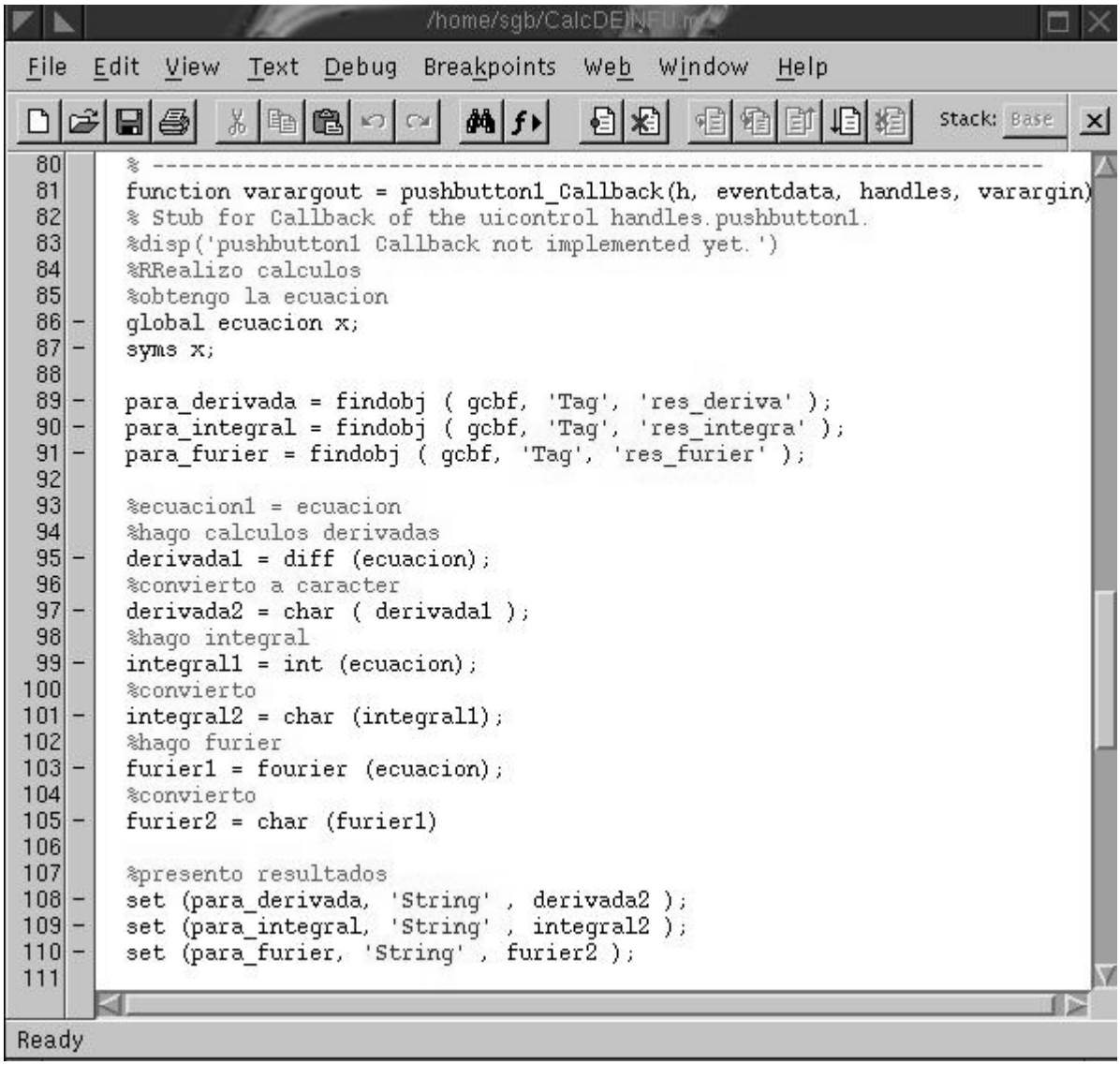

Fig. 3.3.7 – Código completo del botón Calcula

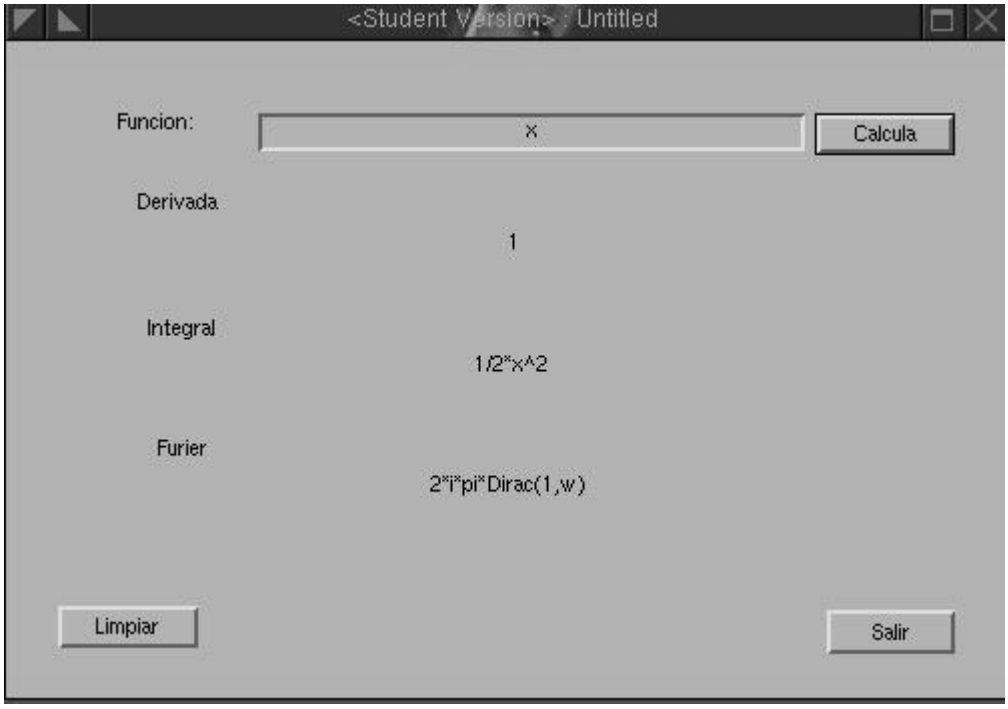

Fig. 3.3.8 – Prueba de la aplicación