Uso básico de Project Builder & Interface Builder en OPENSTEP usando Objective-C

Alcántara Cabrera Moisés
moycc94@gmail.com

Escuela Superior de Cómputo I.P.N.
XX Verano de la Investigación Científica
Universidad Autónoma de Puebla
Departamento de Aplicación de Microcomputadoras
Puebla, Pue. México.

23 de agosto de 2010

Resumen

La finalidad del documento es presentar de una manera sencilla el uso básico de dos aplicaciones de programación, Project Builder e Interface Builder, utilizando el sistema operativo OPENSTEP. Empleando como lenguaje de programación Objective-C. El documento contiene una explicación del entorno de trabajo de estas aplicaciones, seguido del desarrollo de varios programas desde su creación hasta su ejecución.

1. Introducción

NeXTSTEP es un sistema operativo desarrollado por Steve Jobs y un grupo de desarrolladores que él mismo se llevó de Macintosh. Iniciaron su creación en 1985 y en 1988 fue liberado; fue un sistema destinado a venderse únicamente para la educación, sin embargo se pasó de la educación al desarrollo de aplicaciónes ejecutivas, pero fue hasta febrero 1993 que se detuvo la producción de hardware. Meses después se lanzó NeXTSTEP para procesadores Intel [7]. En 1995 se trabaja en OPENSTEP, pero rápidamente se pierde el interés en el desarrollo del sistema y se lanza otro entorno de desarrollo diferente, esta vez para servidores web llamado NeXT Computer mejor conocido como NeXTcube o simplemente "The Cube"[8].

NeXTSTEP es un entorno orientado a objetos, fue la combinación de los siguientes elementos[1]:

- Un sistema operativo Unix basado en el kernel Mach_kernel, más código fuente de la Universidad de California, UC Berkeley, proveniente de su sistema operativo BSD.
- Display PostScript y un motor gráfico.
- Incluyendo Objective-C como lenguje de programación.
- Kits de aplicación oritentados a objetos.
- Paquetes de C que permiten la comunicación entre procesos y aplicaciones de subprocesos múltiples.

Debido a esto los programadores en ese entonces y aún ahora han visto las facilidades del trabajo en NeXT debido a su naturaleza orientada a objetos, es por ello que aseguraron que se desarrollaba una aplicación de tres a diez veces más rápido que bajo Windows.

2. Herramientas básicas de desarrollo

Para programar en el sistema operativo OPENSTEP se sigue un proceso de desarrollo que generalmente se divide en 3 partes: diseño, codificación y depuración. Cada uno de estas partes se conjuntan en herramintas importantes de desarrollo que utilizaremos. Será una breve descripción y más adelante observaremos el entorno de cada una de ellas.

El diseño de una aplicación debe tener en cuenta: la fucionalidad, la estructura y la interfaz de usuario, se deben esbozar ideas de la aplicación antes de escribir cualquier línea de código. Una de las herramientas que es muy eficaz para el diseño interfaces en OPENSTEP es Interface Builder.

Interface Builder es una aplicación creada en 1988 por el desarrollador fránces Jean-Marie Hullot. Es una herramienta útil en el diseño de aplicaciones, lo que implica hacer interfaces más rápidas y gráficas. Permite unir componentes visuales, tales como son: ventanas, menus, botones, etc.; cada una de ellos tiene sus propios atributos los cuales pueden modificarse, así como crear conexiones entre objetos. Guarda los archivos con extensión ".nib"que significa ("NeXTSTEP Interface Builder")[2].

La codificación de la aplicación es también relativamente sencilla, existe en Interface Builder un proceso que ayuda al usuario a tener un esqueleto de las clases que están en el panel Inspector. Si deseamos primero crear el código en un archivo fuera del proyecto, Project Builder nos permite añadir nuevos archivos en el momento que lo desemos, éstos pueden ser código en C, Objective-C y C++; así como archivos pswrap que continen código PostScript en función de C.

Project Builder es una más de nuestras aplicaciones para el desarrollo en OPENSTEP, esta involucrado en cada una de las etapas del proceso de desarrollo, desde el inicio de un nuevo proyecto, hasta la instalación de la aplicación. Para que un archivo sea parte del proyecto debe recidir en el directorio del proyecto y debe tener una extensión PB.project el cual no es modificable, se puede añadir código fuente, cambiar nombre de proyecto o directorio de instalación ya que el archivo PB.project se actualiza [2].

Por último la depuración del programa, en donde la mejor elección es tomar el depurador de Project Builder; de lo contrario iniciar en una terminal con los comandos del shell mediante la directiva GDB.

El lenguaje de programación Objective-C es orientado a objetos implementado con el lenguaje C, es por ello que se puede meter código puro de C y compilar en Objective-C, fue creado en 1980 por Brand Cox y en 1988 lo hacen el lenguaje de programación de NeXTSTEP y años más tarde para OPENSTEP.

3. Ambiente de trabajo con OPENSTEP, Interface Builder y Project Builder

El uso de OPENSTEP requiere adaptarse a su ambiente gráfico nuevo, conocer su ventajas, y su manera de utilizarlas. En la Figura 1 se muestran algunos elementos principales de la pantalla principal de este sistema operativo.

Primeramente la barra de menús y submenús, que es una lista vertical de la que se pueden elegir diferentes opciones, estas pueden tener submenús, que son mostrados con un triángulo apuntando hacia la derecha y que se puede des-

prender de la barra de menus para ponerlo en cualquier lugar del espacio de trabajo. Si se tiene activado el botón derecho del mouse también podrá obtener el menú del programa activo presionándolo en cuaquier parte del área de trabajo libre.

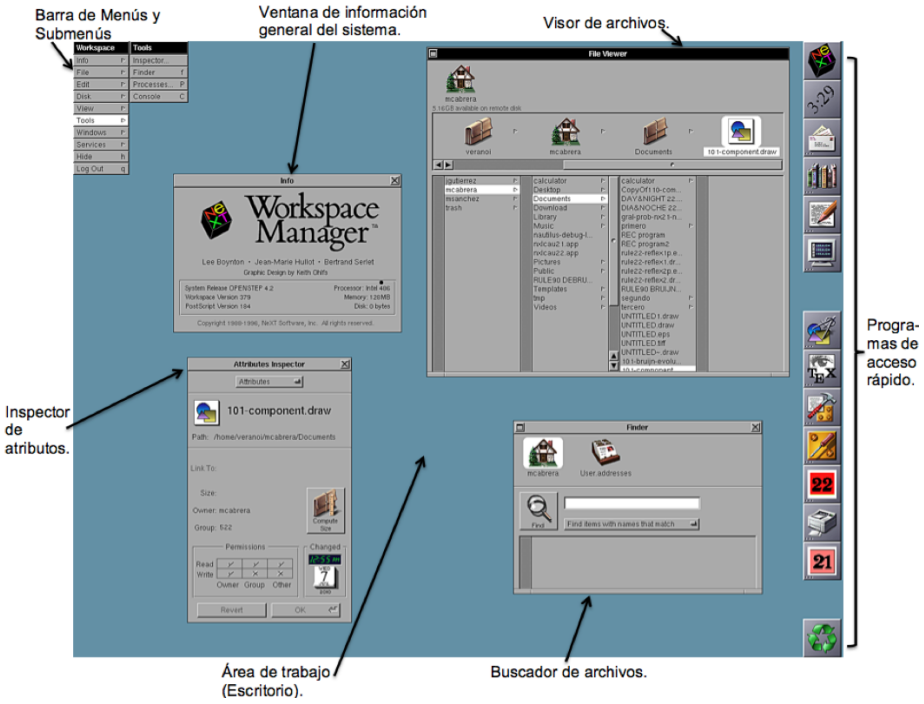

Figura 1: Entorno de trabajo en OPENSTEP.

Dentro del menú de nuestra área de trabajo y accesando al submenú Info podemos elegir la opción Info Panel, que nos muestra la ventana que precisamente tiene el nombre de "Info" y que nos despliega la información general del sistema.

El visor de archivos es un ventana que optimiza la organización de los archivos que existen en nuestra sesión. En esa área se despliegan los archivos que existen como usuario y su contenido, poniendo iconos para su mayor comprensión. La forma de visualizar esta ventana depende del usuario.

El inspector de atributos, por su parte, lo podemos elegir en el submenú Tools y en la opción Inspector, este nos mostrará una nueva ventana que contiene información importante sobre atributos, permisos, tipo de archivo, tamaño, ruta, así como su fecha de modificación, se pueden editar algunos atributos si así se desea.

El área de trabajo es nuestro escritorio en sí. Sobre este se abrirán las aplciaciones que se utilicen y sus herrameintas necesarias, por otro lado el buscador de archivos nos sirve comúnmente para buscar archivos de los cuales no sabemos la ruta en la que se encuentran.

Por último la derecha de la Figura 1 podemos ver iconos para acceso directo de programas que pueden estar o no ejecutandose; notese que el icono de NeXT no tiene tres puntos en la parte inferior izquiera, esto indica que se esta ejecutando.

Una vez que tenemos una noción del entorno gráfico de OPENSTEP y que podemos ubicar algunas de sus funciones, podemos definir el ambiente gráfico de las herramientas que usaremos para diseñar, codificar y depurar nuestras aplicaciones. Comenzaré por explicar rápidamente el entorno de desarrollo de Project Builder, más adelante, en los ejemplos explicaré como funciona cada uno de los elementos que utilizaremos para crear un proyecto, modificar, agregar código, etc.

Una vez ejecutado el programa mostrará una ventana como se muestra en la Figura 2.

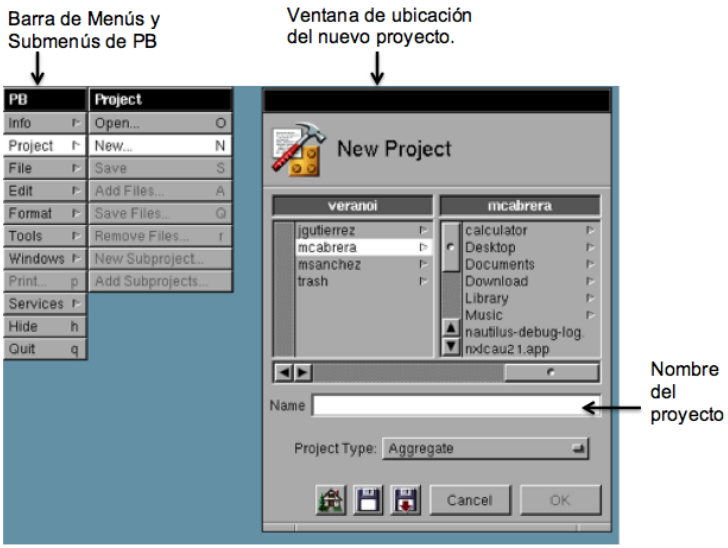

Figura 2: Ventana de Creación de proyecto en Project Builder.

En esta imagen se puede observar una nueva barra de menú específicamente para acciones a realizar por Project Builder; tomaremos la opción New del menú Project, la cual desplegará una ventana, en la cual elegiremos la ruta y el nombre del nuevo proyecto.

Una vez creado un nuevo proyecto nos muestra una ventana como se muestra en la Figura 3, que contendrá los archivos que se utilizarán de una forma organizada, como son las clases, los archivos de cabecera, las interfaces, etc., a esta ventana se le conoce como: gestor de archivos.

Podemos ver del lado dereho del gestor de archivos al inspector del proyecto, el cual contiene algunas especificaciones necesarias de la aplicación como es el nombre del proyecto, el idioma, y el nombre del achivo ".nib"que tendrá el código principal de ejecución.

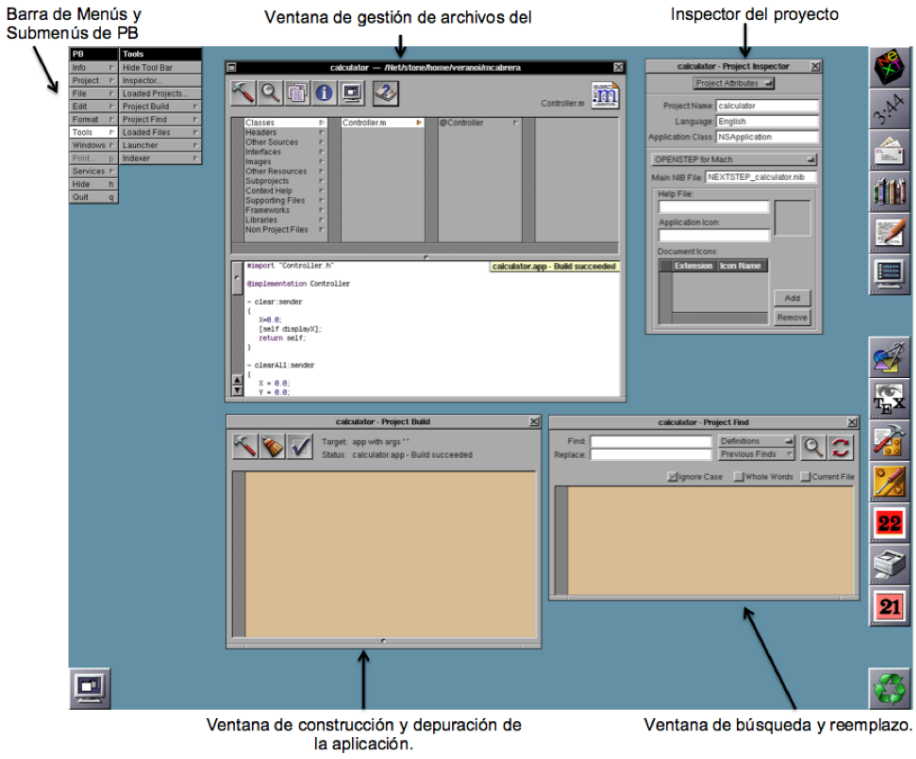

Figura 3: Entorno de trabajo de Project Builder.

Una de las ventanas importantes es la de contrucción y depuración de nuestro programa, la cual compila y crea el archivo ejecutable de nuestro proyecto. Por último podemos buscar algunas palabras dentro de todo nuestro proyecto, esta acción nos la facilita la ventana de búsqueda y reemplazo. Y como ya había mencionado, la barra de menús es diferente para cada programa.

Ahora describiremos rápidamente el entorno de trabajo de Interface Builder el cual nos ayudará como ya lo presente en la introducción, a diseñar una interfaz de usuario para nuestro programa.

En la Figura 4 se muetra Interface Builder en ejecuación; vemos primero una ventana que lleva como título "My Window" y con ella un menú que esta debajo de la barra de menú principal, estos dos elementos son nuestra aplicación por default; en la parte superior derecha se encuentra la ventana de controles y panles que podemos agregar a la ventana de nuestra aplciación, debajo de esta se encuentra el inspector de la aplciación que a su vez nos muestra los atributos de cada control ingresado a nuestra ventana de trabajo siempre y cuando este seleccioando.

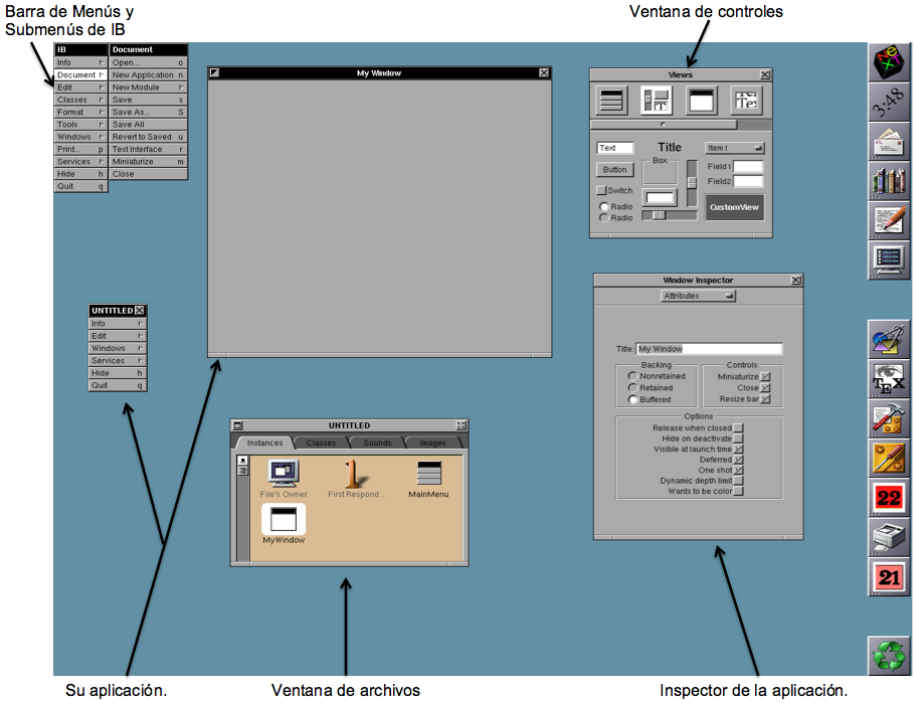

Figura 4: Entorno de trabajo de Interface Builder.

Por último tenemos debajo de la ventana de la aplicación la ventana de archivos, que se encarga de administrar de manera ordenada los archivos que generamos, así maneja el acceso a recursos como las clases, imagenes, sonidos.

Ya conocido el entorno de trabajo de OPENTSTEP, Project Builder e Interface Builder, será más fácil el uso de sus herramientas con ayuda de 3 ejemplos explicativos.

4. "Desarrollo de aplicaciones"

4.1. "Ejemplo 1: Hola Mundo (paso a paso)"

Uno de los ejemplos más sencillos que se pueden desarrollar con Project Builder e Interface Builder es un hola mundo. Enseguida se explican los pasos a seguir para crear esta simple aplicación.

1. Primero, abrir Project Builder, enseguida elegir de su menú principal la opción Project —>New Project, al desplegar la ventana de nuevo proyecto elegimos primeramente la ruta de ubicación, el nombre del proyecto ("holamundo" en este caso), y por último el tipo de aplicación que para este caso es una aplicación, al final presionamos el botón de OK.

2. Se muestra la ventana de gestor de archivos del proyecto, dentro de esta ventana elegimos el archivo NEXTSTEP_holamundo.nib que se encuentra en la carpeta de Interfaces, y presionamos doble click sobre este archivo para editar la interfaz de nuestro programa. Al abrir el ".nib"nos abre inmediatamente la aplicación Interface Builder mostrandonos el entorno de trabajo para nuestra aplicación.

3. Dentro del menú principal de Interface Builder elegimos Tools —>Inspector el cual nos servirá para modificar los atributos de nuestros componentes y ventanas. Ahora seleccionado la ventana de trabajo que tiene como título principal "My Window", se puede modificar en la ventana de inspector en donde se pone "Hola Mundo" al igual que en la barra de menús del programa.

4. Después de la paleta de controles se elige un campo de texto (NSTextField) y se dezplaza con un click sostenido hasta el área de la ventana de la aplicación en donde se desee. Al tener uno de los objetos colocados dentro de nuestra ventana podemos modificar algunos de sus atributos, Colores, Alineación, etc., y presionando dandole doble click a nuestro objeto podemos cambiar el título del mismo, en este caso será "Hola Mundo".

5. Una vez cambiado el texto de nuestro objeto se puede cambiar el tamaño y tipo de letra, para ello del menú principal de Interface Builder se elije Format —>Font —>FontPanel... el cual muestra una ventana de donde se puede elegir el tipo de letra, y tamaño que se quiera, al final presionamos el botón establecer, para que se realicen los cambios.

6. Al término de hacer las modificaciones en el tipo de letra, nos dirigimos a el menú principal Project —>Save, para guardar los cambios de nuestro proyecto.

7. Una vez que se ha guardado procedemos a compilar esta sencilla aplicación. Dentro de nuestro gestor de archivos podemos identificar en la parte

superior un icono con un martillo, el cual es una liga a nuestro panel de contrucción de nuestra aplicación. Presionamos un click sobre él.

8. En el panel que se presenta se identifica nuevamente el martillo el cual compila el programa al darle un click, y si no tiene errores crea un archivo con extensión ".app"que es un archivo ejecutable en OPENSTEP, de lo contrario genera una lista con los errores o advertencias que tiene tu programa, estos son ligas a las líneas en tu código que deben ser corregidas y así depurar nuestro programa.

9. Una vez que es correcta la compilación, se puede buscar el archivo ejecutable en la ruta en donde se guardo el proyecto, en este caso es llamado "holamundo.app" y al darle doble click se ejecuta. El resultado debe ser parecido a la Figura 5.

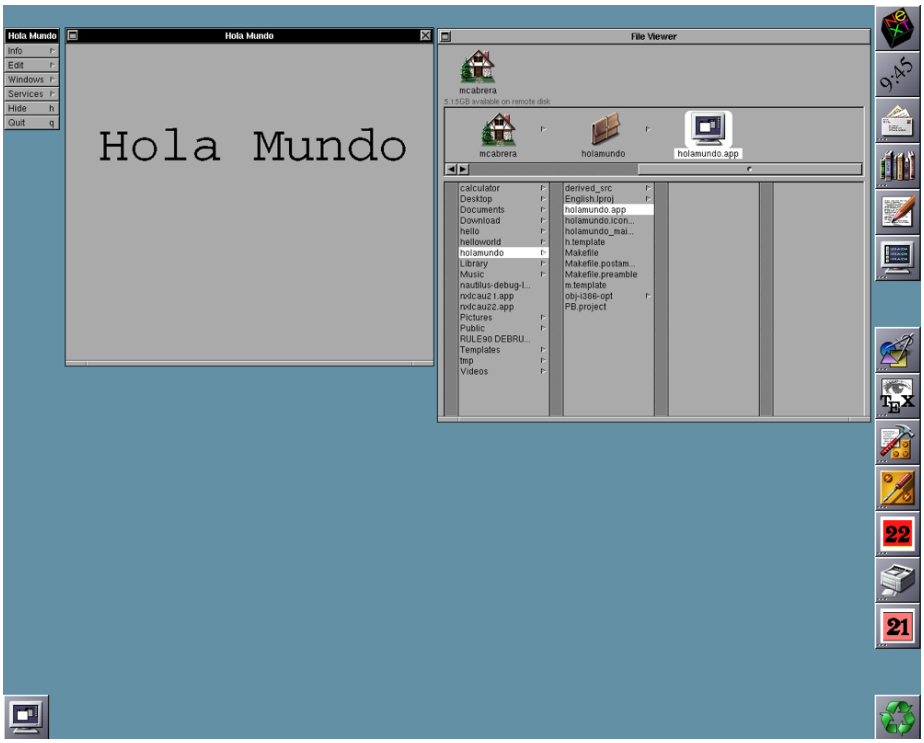

Figura 5: Ejecución Hola Mundo.

10. Una vez ejecutado el programa se puede ver que tiene un icono por omisión, podemos tambien darnos cuenta del título de la vantana, así como de la barra de menú, y las opciones que se elijieron para el título. Por otro lado podemos ver que en nuestro visor de archivos en la ubicación del proyecto se encuentra nuestro archivo ejecutable.

4.2. "Ejemplo 2: Conversor de temperatura (Empezado con código)"

Este ejemplo al igual que el anterior es sencillo, pero con la diferencia de la creación y el inicio de modificación de código en los archivos ".m", que contienen las implementaciones de las clases del programa y en los archivos ".h", donde declaran las variables, y las clases correspondietes. También se muestra como hacer las conexiones entre los objetos de la interfaz de la aplicación.

La aplicación es un conversor de grados Celsius a Fahrenheit, bastante sencilla pero cumple con los elementos necesarios para entender aun más como se usa Interface Builder y Project Builder.

Primeramente creamos en Project Builder un nuevo proyecto de tipo aplicación con el nombre de conversor. Una vez creado el proyecto, dentro de nuestro gestor de archivos, y en la carpeta interfaces, ubicamos al archivo llamado "conversor.nib", al cual abriremos con doble click para cargar una nueva interfaz.

Ya que tenemos abierta la nueva interfaz podemos añadir objetos a ella de tal manera que se vea como en la Figura 6. Para poder editar algunos de los atributos de los objetos insertados, tome en cuenta las siguientes recomendaciones:

Figura 6: Interfaz del conversor de grados.

- Para editar los atributos de los componentes de la aplicación abrimos del menú principal de Interface Builder elejimos la opción Tools —>Inspector y podrémos cambiar algunos de los atributos de los objetos de manera rápida.
- Para modificar la letra de los campos de texto debemos ir al menú principal de Interface Builder elegir Format —>Font —>Font Panel, se despliega una ventana como se muestra en la Figura 7, en donde se puede elegir el tipo, estilo y tamaño de la letra.

- Para cambiar el color de las letras debemos seleccionar el campo de texto que queremos editar, una vez que lo hemos seleccionado podemos darnos cuenta que en el inspector hay un atributo llamado color de texto el cual cambairemos con la paleta de colores ubicada en Tools —>Colors, teniendo esta paleta mostrada en la Figura 8, elegimos el de muestra preferencia y arrastramos hasta el atributo del inspector que cambia el color del texto.

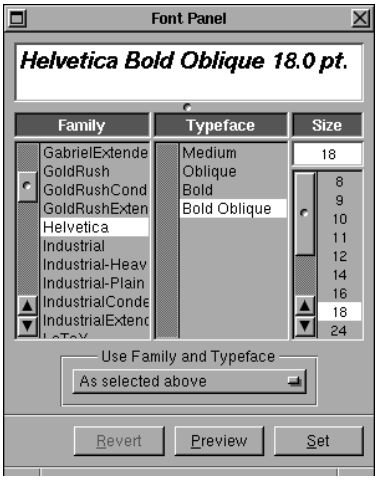

Figura 7: Panel de fuente.

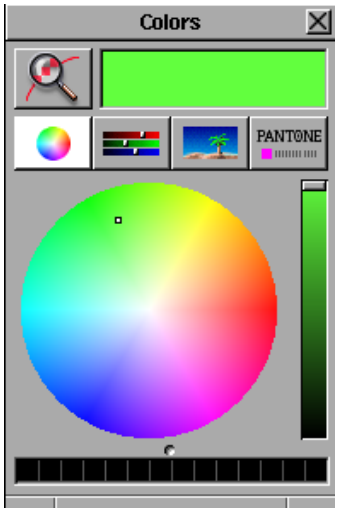

Figura 8: Paleta de colores.

- Para poner una imagen en un botón nos dirigimos al visor de archivos del proyecto y en la pestaña de imagenes como se muestra en la Figura 9, elegimos la que nos agrade y la arrastramos hasta el botón de la aplicación, dentro del inspector se puede modificar la posición de esta imagen, con respecto al texto. Debemos tomar en cuenta que si queremos agregar una imagen nueva, basta con agregarla arrastrandola a venta de la Figura 9, aceptando que se añadirá al proyecto.
- Debemos fijarnos que nuestra caja de texto de la opción de salida no sea editable, podemos modificar uno de sus atributos dentro de el inspector quitando la opción Editable.
- Por último debemos redimensionar la ventana para que se vea compacta como en la Figura 6.

Una vez creada la interfaz, debemos definir las salidas y los métodos que se utilizarán en este programa, para ello nos colocamos en el visor de archivos en la pestaña de Clases y creamos con el click secundario sobre nuestra clase NSObject una subclase a la cual llamaremos "Calcular" en ella agregaremos las salidas y los métodos a utilizar; para añadir éstas debemos dar click secundario sobre nuestra nueva subclase y elegir añadir salida, en ella pondremos las dos

Figura 9: Visor de archivos (Pestaña de imagenes).

salidas que usaremos, la primera será "entrada" y la segunda "salida" y para el método, se elige añadir accción, en la cual añadiremos "calcular", ya agregados debemos crear los archivos ".m" y ".h"para ello con el botón decho del ratón sobre la subclase "Calcular" presionamos Classes —>Crear archivos, esta acción presentará dos ventanas, la primera con la ruta de los archivos tanto .m como .h, y en la segunda aceptando que se añadiran al proyecto, se pondrá en ambas ventanas que si se acepta tanto la ubicación como la adición de los archivos al proyecto. Estos archivos son un esqueleto del código de nuestro programa.

Ahora podemos iniciar nuestra subclase presionando botón derecho sobre ella y en Classes —>Inicializar. Con ello creamos un objeto nuevo en la pestaña de Instancias de nuestro visor de archivos.

El siguiente paso es crear las conexiones necesarias para el funcionamiento del programa, estas realizan la interacción entre objetos y código de implementación.

Comenzaremos con nuestra caja de texto de entrada, y teniendo presionado el botón de Ctrl del teclado arrastramos la línea de conexión hasta el botón de Calcular, dentro de el inspector elegimos la acción perform click y presionamos doble click o Conectar, como se muestra en la Figura 10.

> ***Nota: Asegurese de que la conexión se establezca de manera correcta, ya que si se presiona un solo click o más de tres, la conexión no se realiza.***

De la misma forma hacemos la conexión desde el botón Calcular hasta el icono creado al inicializar la subclase, conectando con la acción calcular. Después para conectar nuestras salidas, debemos tomar nuestro objeto Calcular del visor de archivos y conectarlo con la caja de texto de entrada y la acción

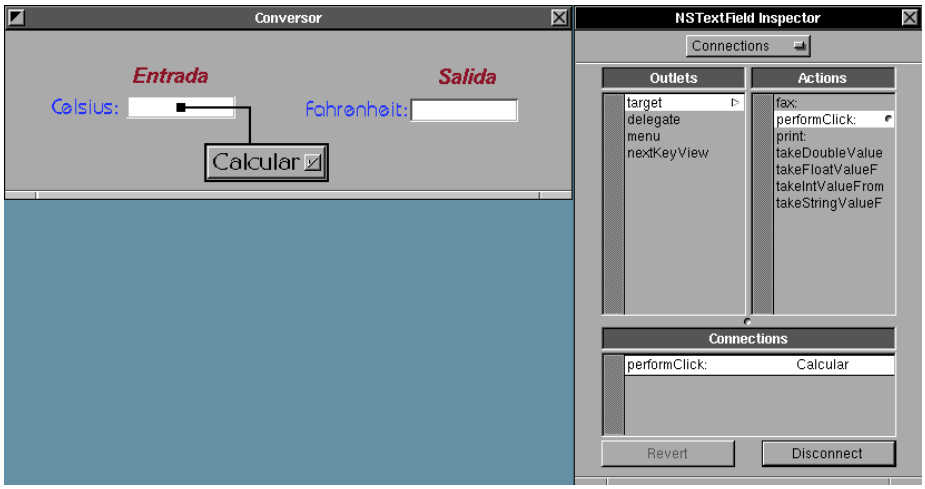

Figura 10: Creación de conexiones.

entrada, e igualmente con la caja de salida pero en este caso con la acción salida.

Terminadas las conexiones podemos editar nuestros archivos Calcular.h y Calcular.m que se encuentran en nuestro gestor de archivos. Empecemos por Calcular.h, el cual define las clases, los archivos de cabecera y las variables globales. El archivo debe contener el siguiente código:

```
#import <AppKit/AppKit.h>

@interface Calcular : NSObject
{
    id entrada;
    id salida;
}
- calcular:sender;
@end
```

Con este código declaramos nuestras cajas de texto de la aplicación de la cual se obtendrán y pondrán los datos, así como la declaración de la clase calcular, la cual se implementará en el archivo Calcular.m.

Al término guardamos los cambios y abrimos el archivo Calcular.m que es nuestra implementación de la clase Calcular, en este archivo agregamos las siguientes líneas:

```
#import "Calcular.h"

@implementation Calcular

- calcular:sender
{
  float gradosF;
  [entrada selectText:self];
  gradosF= ((9.0*[entrada floatValue])/5.0)+32.0;
  [salida setFloatValue:gradosF];
  return self;
}
@end
```

En este archivo declaramos una variable flotante local, la cual se encargara de guardar el resultado del cálculo realizado, obtnemos el texto de entrada, hacemos el cálculo, y por último mandamos la salida a nuestra caja de texto correspondiente.

Ya que hemos editado los archivos, guardamos los cambios tanto en Interface Builder como en Project Builder.

Ahora procedemos a compilar nuestro programa, esto lo podemos hacer con la herramienta que nos proporciona Project Builder, el cual su icono es un martillo que al presionarlo nos muestra un panel de compilación, como se muestra en la Figura 11, en este panel tenemos la opción de compilar, depurar, o meter argumentos a nuestro programa, utilizaremos el de compilación que es el primer icono, el cual nos crea un archivo ejecutable que en este caso será "conversor.app" siempre y cuando no existan errores dentro del programa, de lo contrario despliega una lista con los posibles errores.

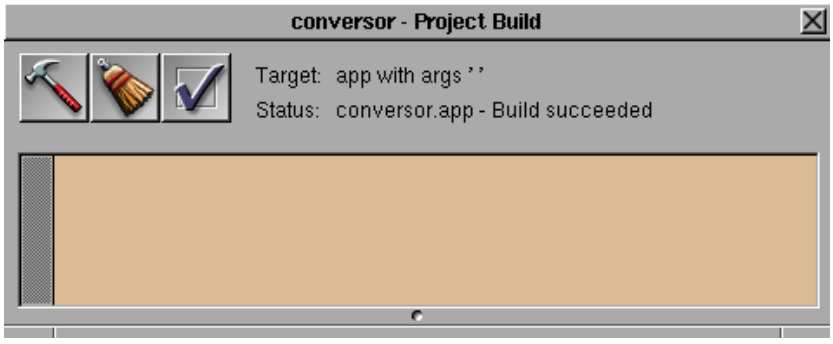

Figura 11: Panel de compilación.

Algunos de los errores más comunes son:

- La sintaxis en el códido (; - [] -).
- Falta de declaración de variables.

También se presentan algunas advertencias, de las más comunes son:

- Declaración de variables que no son usadas.
- Traspaso de información de diferentes tipos de datos.
- Mala o nula implementación de alguna clase ya declarada.

Al ejecutar el archivo conversor.app que se guarda dentro de la carpeta en donde se encuentra el proyecto, debe funcionar correctamente y tener una apariencia como se muestra en la Figura 12.

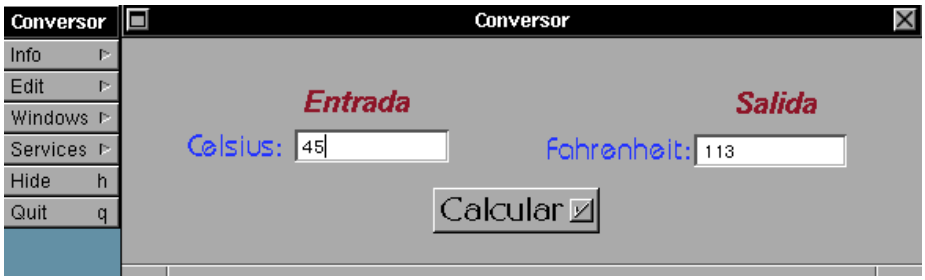

Figura 12: Ejecución del programa (conversor.app).

4.3. "Ejemplo 3: Calculadora"

En este ejemplo aprenderemos a usar matrices de objetos, hacer conexiones con la función *delegate*, insertar paneles o ventanas a nuestro programa, e insertar un icono para identificar nuestra aplicación al ejecutarse.

Empezaremos abriendo un nuevo proyecto de Project Builder, lo llamaremos calculadora y crearemos con nuestra ventana de controles una interfaz parecida a la de la Figura 13.

Para poder utilizar los botones de una matriz tanto de botones, cajas de elección, etc., debemos identificarlos de alguna forma; para eso se deben modificar los atributos de cada botón, pero no sólo el nombre sino también su etiqueta, la cual será la que identificará al objeto dentro del código.

Para llevar a cabo ésto debemos seguir las siguientes sugerencias:

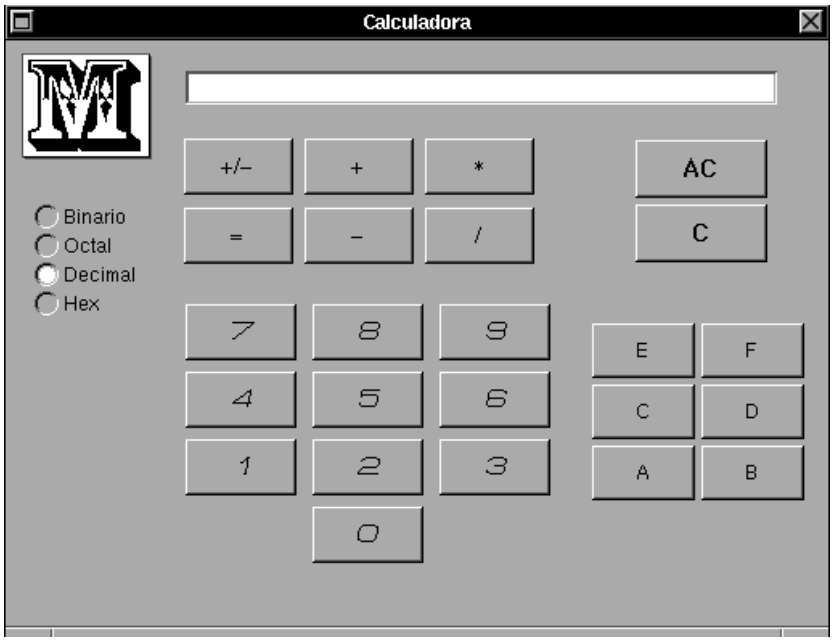

Figura 13: Interfaz gráfica de la calculadora.

- Editar los atributos de la caja de texto con la ventana de Inspector y modificar su alineación, la cual deberá ser hacia la derecha, quitar la selección de la casilla con la opción de editable, y eliminar el texto dentro del elemento.
- Para crear las matrices de los operadores, de los números, de las teclas de borrado y del teclado númerico, únicamente debe insertar un solo botón y con la tecla de Ctrl en una se sus esquinas arrastrarlo hasta tener el número de botones que sea necesario.
- Después debemos ponerle una etiqueta a cada botón, debemos tomar en cuenta que la etiqueta no es lo mismo que el título del botón. En la ventana Inspector existe una opción llamada *tag*, la cual será el identificador del botón dentro del código. Para observe el valor de cada uno en el Cuadro 1.
- En el caso de la númeración los botones al costado del cero se desabilitan y se ponen invisibles, estas opciones estan en el Inspector del objeto, el *tag* de estos bototnes será su mismo número, y en el caso del teclado alfabéticola letra A tendrá el valor de 10, la letra B el 11, y así sucesivamente.
- Por último los botones de limpiar, limpiar todo, y cambio de signo tienen su propia acción dentro del código, es por ello que no requieren un identificador.

Título	Etiqueta (*Tag*)	Título	Etiqueta (*Tag*)
+	1001	Binario	2
-	1002	Octal	8
*	1003	Decimal	10
/	1004	Hex	16
=	1005		

Cuadro 1: Etiquetas de matrices.

Una vez que se ha terminado de diseñar la interfaz, creamos una subclase de la clase NSObject, a la nueva subclase creada la nombraremos "Control", el cual contendrá las acciones y salidas que se muestran en el Cuadro 2.

Salidas	**Acciones**
lectura	limpiar
baseMatriz	limTodo
tecAlf	metedDig
	meterOp
	signo
	ponerBase

Cuadro 2: Etiquetas de matrices.

Ya que estan agregadas las acciones a nuestra subclase procedemos a crear los archivos Control.h y Control.m e inicializamos nuestra instancia de la clase Control.

Ahora se hacen las siguientes conexiones, las cuales son necesarias para el funcionamiento del programa.

1. Conectar la matriz del teclado númerico con la instancia Control y elegir la acción meterDig.
2. Dando doble click sobre el botón CA conectarlo con la instancia Control y la acción limTodo.
3. Igualmente con doble click pero ahora sobre el botón C y conectarlo con la instancia Control y la acción limpiar.
4. Ahora de la instancia Control hacia la caja de texto en donde obtendremos el resultado de nuestras operaciones, eligiendo la salida llamada lectura.
5. Desde la matriz de operadores creamos la conexión con la instancia Control y la acción meterOp.

6. Como se menciona arriba el botón de signo dentro de la matriz de operadores no tiene un identificador debido a que tiene una acción específica, en este caso la conectaremos con la acción signo.

7. Nuevamente de la instancia de la clase Control pero ahora hacia la matriz de botones de elección de base, eligiendo con en las opciones de salida la llamada baseMatriz.

8. Por último creamos la conexión de la instancia de la clase Control a el teclado alfabético, esta a su vez con la salida tecAlf.

Ya terminadas las conexiones, procederemos a editar el archivo Control.h; en él importamos las librerías que se ocupan, en este caso a demás de la librería principal, importamos también *string.h* pues la conversión a número binario ocupa la función *strcpy* que se encuentra en dicha librería. Por otro lado ocupamos listas al poner la base en la que se presentará el número y estas estan contenidas *objc/List.h*.

Declaramos una estructura llamada "enum" que contendrá los identificadores de la matriz de operadores, depués nuestras variables del programa dentro de la interfaz Control que es contenida en la clase NSObject depués de esto declarar los métodos que serán implementados. Y al último un *delegate* que se explicará más adelante.

```
#import <AppKit/AppKit.h>
#import <string.h>
#import <objc/List.h>

enum{
  SUMA = 1001,
  RESTA = 1002,
  MULTIPLICACION = 1003,
  DIVISION = 1004,
  IGUAL = 1005
};

@interface Control : NSObject
{
    id lectura;
    BOOL band1;
    BOOL band2;
    int operacion;
    float X;
    float Y;
    int base;
    id baseMatriz;
```

```
    id tecAlf;
}
- limpiar:sender;
- limTodo:sender;
- meterDig:sender;
- meterOp:sender;
- mostrarR;
- signo:sender;
- ponerBase:sender;
@end

@interface Control(ApplicationDelegate)

- inicio:sender;

@end
```

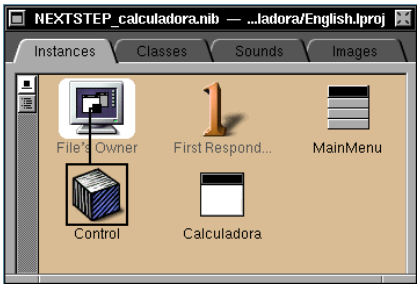

Figura 14: Conexión delegate.

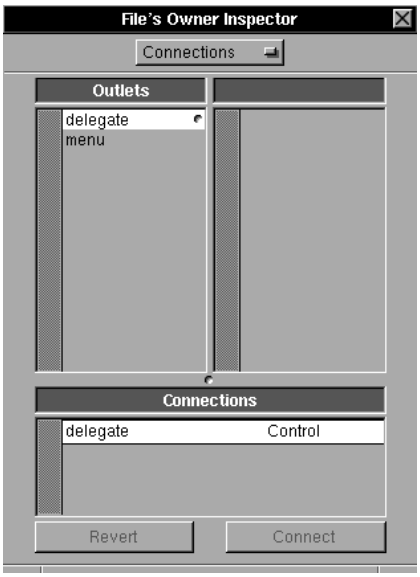

Figura 15: Ventana Inspector para delegate.

La implementación de estos métodos se realiza en el archivo Control.m, en este archivo se importa a la librería creada por nosotros Control.h; al final del archivo existe una implemetntación de *@implementation Control(ApplicationDelegate)* la cual nos sirve para invocar a un método automáticamente en respuesta a una acción del usuario, en NeXTSTEP *delegate* proporciona un sistema para modificar o controlar el comportamiento de un objeto, y en este caso para controlar

con un objeto otro objeto.

Para hacer este procedimiento basta con dirigirse al visor de archivos de nuestro proyecto en la pestaña de instancias y elegir nuestro archivo propietario y hacr una conexión hasta la instancia de la clase Control. Una vez hecho esto en la ventana Inspector elegir la salida *delegate* para hacer la conexión. Ver la Figura 14 y Figura 15 .

Entonces el código de implementación en el archivo Control.m quedaría de la siguiente forma:

```
#import "Control.h"

char *ltob(unsigned long val, char *buf)
{
  int i;
   for(i=0; i<32; i++){
     buf[i]= (val & (1<<(31-i)) ? '1' : '0');
   }
   buf[32]='\0';
   for(i=0; i<32; i++){
        if(buf[i]!= '0') return (&buf[i]);
      }
   return (&buf[31]);
}


@implementation Control

- limpiar:sender
{
   X=0.0;
   [self mostrarR];
   return self;
}

- limTodo:sender
{
   X = 0.0;
   Y = 0.0;
   band2 = NO;
   band1 = NO;
   [self mostrarR];
   return self;
}
```

```
- meterDig:sender
{
   if(band1){
   Y=X;
   X=0.0;
   band1 = NO;
   }
   X=(X*base)+[[sender selectedCell] tag];
   [self mostrarR];
   return self;
}

- meterOp:sender
{
   if(band2){
    switch(operacion){
     case SUMA:
      X=Y+X;
      break;
     case RESTA:
      X=Y-X;
      break;
     case MULTIPLICACION:
      X=Y*X;
      break;
     case DIVISION:
      X=Y/X;
      break;
    }
   }
   Y =X;
   band2 = YES;
   operacion = [[sender selectedCell]tag];
   band1 = YES;
   [self mostrarR];
   return self;
}

- signo:sender
{
  X=-X;
  [self mostrarR];
  return self;
}

- mostrarR
```

```
{
   char buf[256];
   char buff[256];
   switch(base){
     case 10:
      sprintf(buf, "%15.10g", X);
      break;
     case 16:
      sprintf(buf, "%x", (unsigned int)X);
      break;
     case 8:
      sprintf(buf, "%o", (unsigned int)X);
      break;
     case 2:
      strcpy(buf, ltob((int)X, buff));
      break;
    }
   [lectura setStringValue:buf];
   return self;
}

- ponerBase:sender
{
  int i;
  id cellList;
  base = [[sender selectedCell] tag];
  cellList = [tecAlf cellList];
  for(i=0; i<[cellList count]; i++){
    id cell = [cellList objectAt: i];
  [cell setEnabled: ([cell tag] < base)];
  }
  [self mostrarR];
  return self;
}
@end

@implementation Control(ApplicationDelegate)

- inicio:sender
{
 [self ponerBase:baseMatriz];
 [self limTodo:self];
 return self;
}
@end
```

Añadiremos ahora una imagen a nuestra aplicación para identificarla. Esto es muy sencillo unicamente habrá que seleccionar el proyecto principal y en atributos de la aplicación tiene la opción de agregar un icono para tu programa como se muestra en la Figura 16, únicamente arrastra hasta el espacio señalado la imagen y listo, saldrá una ventana que se agregará la imagen al proyecto aceptas y listo.

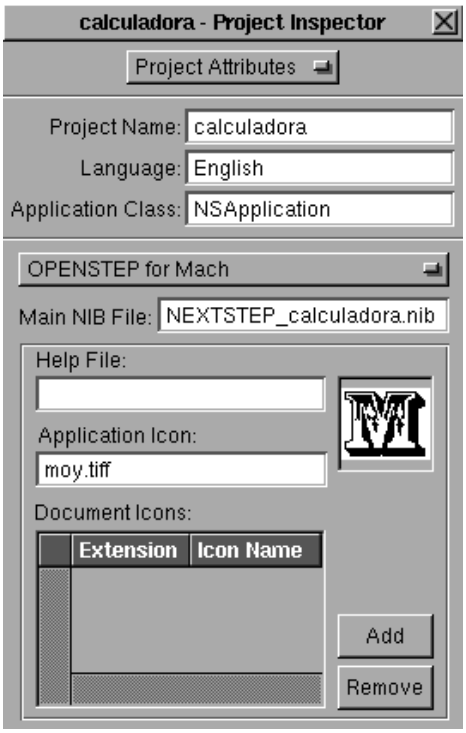

Figura 16: Agregar imagen a la aplicación.

El siguiente paso es agregar un panel de información sobre el programa, primero nos vamos al menú de nuestra aplicación y elegimos la opción Info —>InfoPanel y en atributos le quitamos la casilla desabilitada. Una vez habilitado esta opción elegimos de nuestra ventana de controles un panel y modificamos sus atributos de tal forma que se vea como la Figura 17

Para hacer que este panel salga cuando presionemos la opción en nuestro menú, debemos hacer una conexíon entre Info Panel del menú y el Panel y elegir de nuestra ventana de Inspector la acción *makeKeyAndOrderFront*.

Después de esto guardamos los cambios en nuestro proyecto, compilamos y ejecutamos. Debemos checar que cada uno de nuestros botones funcione perfec-

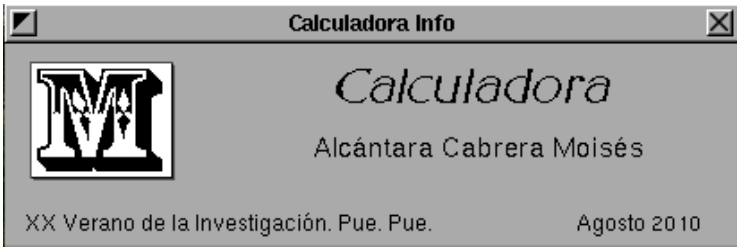

Figura 17: Panel de información.

tamente para ello probar con el ejemplo de la Figura 18 y Figura 19

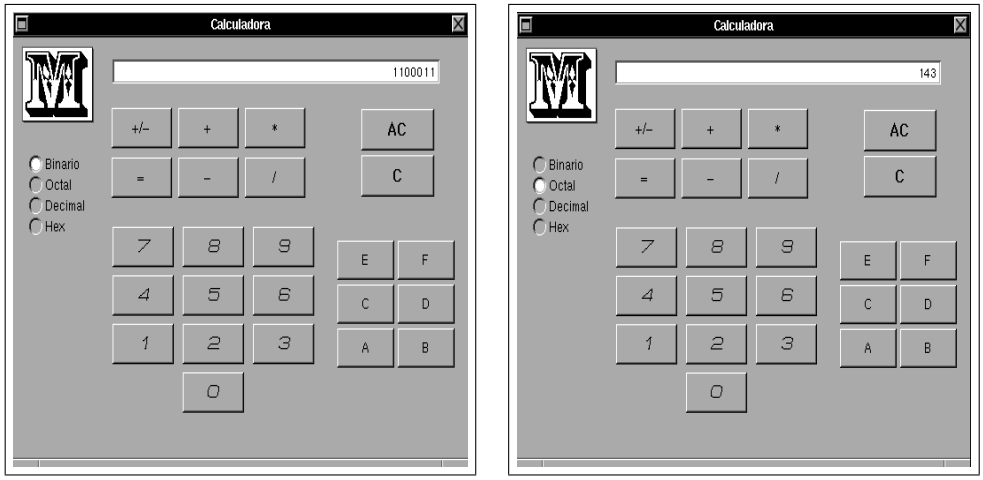

Figura 18: Conversión Binario y octal.

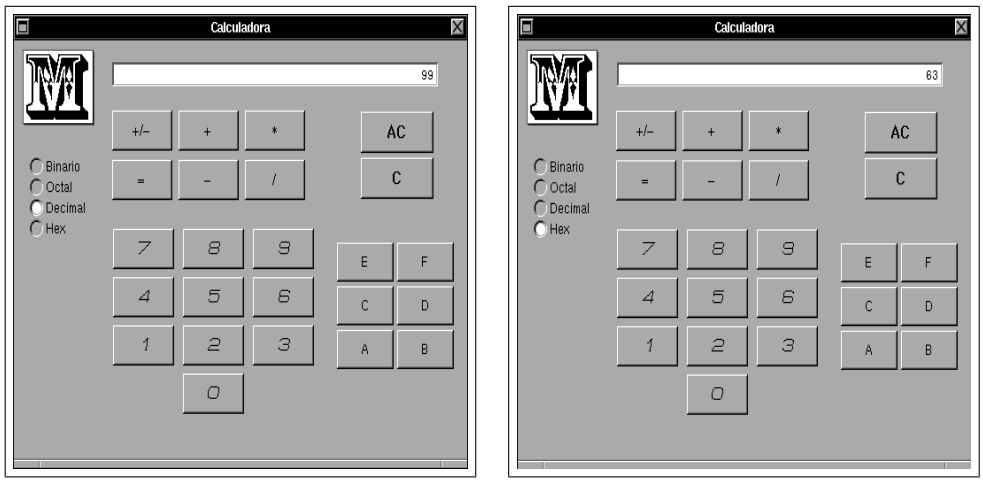

Figura 19: Conversión Décimal y Hexadécimal.

5. Conclusión

Con estos ejemplos observamos el uso básico de las herramientas de programaciónen Project Builder, Interface Builder y el código en Objective-C. Debemos tener en cuenta que desde sus inicios con NeXTSTEP hasta ahora estas herramientas han sufrido modificaciones en sus diferentes versiones. También recordar que Apple ha adoptado estas herramientas de programación bajo su entorno de trabajo llamado Xcode, cambiando la forma de uso de las aplicaciones, las librerías, etc., es por ello que los programas que aquí se presentan fueron compilados y ejecutados con el siguiente software:

- Versión del sistema OPENSTEP 4.2
- Project Builder Versión 4.2 (v300.2)
- Interface Builder Versión 4.2 (v840)

6. Referencias

[1] Garfinkel Simson L., *NeXTSTEP PROGRAMMING: step one object-oriented applications*, Springer-Verlag, New York, USA, 1993.

[2] NeXTSTEP Publications, *NeXTSTEP DEVELOPMENT TOOLS AND TECHNIQUES*, NeXTSTEP Publications, USA, 1992.

[3] NeXT Publications, *Cocoa programming for MAC OS X*, Ed. Addison-Wesley, USA, May 2008.

[4] Sun Microsystems, Inc., *OpenStep User Interface Guidelines*, Mountain View, CA, USA, 1995 NeXRT Computer, Inc.

[5] Sun Microsystems, Inc., *OpenStep Programming Reference*, Mountain View, CA, USA, 1995 NeXRT Computer, Inc.

[6] Sun Microsystems, Inc., *OpenStep Development Tools*, Mountain View, CA, USA, 1995 NeXRT Computer, Inc.

[7] Paul Lynch, NeXTSTEP Technical Review, *http://www.paullynch.org/NeXTSTEP/NeXTSTEP.TechReview.html*, Broadway, Manhattan, USA. Fecha de modificación: Febrero 1997.

[8] EcuRed, Servidores Web, *http://www.ecured.cu/index.php/Servidores_Web*, Cuba, Cuba. Fecha de modificación: Agosto 2010.