ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD

"DISEÑO E IMPLEMENTACION DE UNA INTERFAZ GRAFICA UTILIZADA POR EL USUARIO PARA ACCIONAR UNA GRUA PROTOTIPO DE MANDO COMPUTARIZADO"

TESIS DE GRADO

Previa a la Obtención del Título de:

INGENIERO EN COMPUTACION

Presentada por:

GIGLIA BEATRIZ TOBALINA DITO

GUAYAQUIL-ECUADOR

1.993

A G R A D E C I M I E N T O

AL ING. SIXTO GARCIA A.

D E D I C A T O R I A

A DIOS
A MIS PADRES
A MI HIJA

D E C L A R A C I O N E X P R E S A

" La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tésis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Titulos Profesionales de la ESPOL)

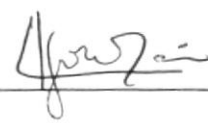
GIGLIA BEATRIZ TOBALINA DITO

MIEMBROS DEL TRIBUNAL DE GRADO

ING. ARMANDO ALTAMIRANO
SUB-DECANO

ING. SIXTO GARCIA
DIRECTOR DE TESIS

ING. JAIME PUENTE
MIEMBRO PRINCIPAL



ING. CARLOS JORDAN
MIEMBRO SUPLENTE

RESUMEN

El presente trabajo consiste en implementar una interfaz gráfica amigable al usuario para accionar una grúa prototipo a escala, de mando computarizado.

Inicialmente se analizan los conceptos de Programación Orientada a Objetos y el Lenguaje Borland C++ con utilitario Windows. A continuación se describen los recursos de hardware: puerto serial, ratón y joystick, necesarios para el manejo de la grúa.

La Interfáz Gráfica utiliza el mouse en un ambiente de ventanas para seleccionar los movimientos de la grúa, de igual manera con el joystick se determinarán dichos movimientos. La grúa prototipo realiza los siguientes movimientos: subir o bajar el gancho, abrir o cerrar el brazo y rotación del cuerpo a la derecha o a la izquierda.

El control se realiza desde cualquier computador personal IBM-PC, el cuál se comunica con la grúa a través de una interfaz digital.

INDICE GENERAL

Pág.

Pág.

Pág.

INDICE DE FIGURAS

Pág.

INDICE DE TABLAS

Pág.

INTRODUCCION

La Interfaz Gráfica para Accionar la Grúa Prototipo, nos permite manejar la grúa de una manera sencilla y agradable.

El objetivo principal es utilizar los conceptos de Programación Orientada a Objetos en una aplicación que trabaja en un ambiente de ventanas, y en la que se manejan los puertos del computador, como son: puerto serial, puerto paralelo y puerto de juegos.

Programar en un Lenguaje Orientado al Objeto, supone crear nuevos tipos de datos llamados clases, y enseñar a estas clases como enviar mensajes. Una clase envía un mensaje mediante un método. Se definen variables de este nuevo tipo de dato y se denominan objetos o instancias. Es a estos objetos que se envían los mensajes. Para el desarrollo de la interfaz gráfica se defininieron las clases, sus métodos y los objetos mecesarios.

Se utilizó también una Interfaz Digital para controlar los movimientos de la grúa. La Interfaz Digital se configura enviando una señal desde el puerto paralelo del computador.

De acuerdo a la señal que la interfaz digital recibe, la grúa prototipo realiza un movimiento diferente.

El control de la grúa se realiza con el ratón, en un ambiente de ventanas, y con el joystick. De esta forma se implemento una interfáz gráfica amigable para accionar la grúa.

CAPITULO I

DESCRIPCION DE LOS RECURSOS DE SOFTWARE

1.1 PROGRAMACION ORIENTADA A OBJETOS

1.1.1 GENERALIDADES

La Programación Orientada a Objetos es una nueva forma de pensar cómo resolver algunos de los problemas planteados en el mundo de la informática.
En lugar de tratar de adaptar el problema a algún aspecto familiar al computador, el computador se adapta al problema.

En la programación orientada al objeto, se examina el problema en " entidades " independientes que se relacionan con otras partes del problema. Estas entidades no se eligen por su computerización, sino porque tienen algún límite físico o conceptual que los separa del resto del problema. Las entidades son representadas como objetos en el programa

para computadora.

El objetivo es tener una correspondencia, una a una, entre entidades en el problema físico y objetos del programa.

1.1.2 EVOLUCION

La Programación Orientada a Objetos surgió para dar una mejor solución a los grandes problemas.

A medida que se van desarrollando los lenguajes, se va desarrollando también la posibilidad de resolver problemas cada vez más complejos. En la evolución de cada lenguaje, llega un momento en que comienzan a existir dificultades a la hora de manejar programas que sean de un cierto tamaño y sofisticación.

Los primeros lenguajes de programación, se diseñaron para minimizar los errores creados en la traducción de un concepto de programación a una representación máquina. Estos lenguajes utilizan palabras similares al inglés, y permiten la sustitución de posiciones de

memoria con nombres elegidos por el programador.

LIMITES DE LOS LENGUAJES FUNCIONALES

La complejidad aumenta a medida que se dispone de más recursos. Los primeros programadores pronto se dieron cuenta de la dificultad de crear programas más grandes y elaborados que funcionaran correctamente.

Los lenguajes de procedimientos fueron la siguiente etapa en la evolución de lenguajes. El modelo utilizado en estos lenguajes es la **caja negra**. Estas " cajas " son funciones (en Pascal, procedimientos) y en el nivel superior, el programa es un conjunto de llamadas de función.

El concepto de caja negra no modifica los datos fuera de sus límites, es decir no genera **efectos laterales**, en la práctica este imperativo no es demasiado severo, lo que provoca una desventaja ya que los datos no están completamente bajo el control de la caja.

A finales de los años 60 y principios de los 70 los profesionales vieron que el límite de los lenguajes de procedimientos estaba en su complejidad de manejo.

DESARROLLO ESTRUCTURADO

Los programas, realizados con lenguajes procedurales tradicionales, pueden ser difíciles de modificar y mantener. La reacción a este problema fué forzar la estructura de los programas a través de una metodología denominada **desarrollo estructurado**. Las técnicas salvaron al lenguaje de procedimientos para su empleo en grandes proyectos. Sin embargo, requiere un alto grado de prudencia y planificación para que el proyecto se ensamble rápida y correctamente, que quede sin errores y sea fácil de mantener.

El desarrollo estructurado no funciona tan bien como parece, ya que es un intento de salvar a un grupo concreto de lenguajes sin ajustarse

realmente a las verdaderas necesidades de los programadores.

Aunque se pudiera aprovechar algunas técnicas útiles del lenguaje estructurado, no fué una "solución mágica". Las técnicas estructuradas fueron parte de la revolución científica. Cada vez que el sistema tradicional se iba destruyendo se intentaba por todos los medios salvarle.

La siguiente etapa, tenía que ser, por tanto, la creación de un nuevo sistema que aceptara las limitaciones que el viejo sistema intentaba negar, surge así la **programación orientada a objetos** la cuál tiende a enfatizar el diseño incluso más que las técnicas estructuradas, pudiéndose pensar en un "desarrollo estructurado mejor".

1.1.3 SOPORTE PARA EXPERIMENTOS Y ESTRUCTURAS

Al utilizar el lenguaje ensamblador se podía dejar de decodificar números y en su lugar tratar con palabras. Los lenguajes de procedimientos ocultaron la complejidad de las

operaciones en los datos. A su vez, los lenguajes orientados a objetos encubren la complejidad del programa en sí mismo, así como la ocultación de los datos.

Un lenguaje orientado a objetos, enfatiza los tipos de datos y las operaciones intrínsecas que pueden desarrollarse en aquellos tipos de datos.

En la programación orientada a objetos, los datos no fluyen libremente por el sistema, ya que están protegidos de alguna modificación accidental. Son los **mensajes,** más que los datos, los que se pueden mover en el sistema.
Un lenguaje orientado a objetos apoya la experimentación de dos formas:

- Los objetos son paquetes compactos que no se rompen simplemente porque se añadan otros objetos. Se pueden introducir nuevos objetos y codificarlos, sabiendo que los errores del nuevo código serán aislados del resto del sistema.

- Los tipos de objetos nuevos, pueden derivarse de los viejos.

1.1.4 PROCESO DE TRADUCCION DEL LENGUAJE

Todos los lenguajes de programación, traducen un lenguaje fácil de comprender por los hombres a otro que pueda ser ejecutado por la computadora, denominado **instrucciones máquina**. Tradicionalmente los traductores se dividen en dos clases: **intérpretes** y **compiladores**.

Los Intérpretes traducen código fuente, escrito en el lenguaje de programación, a actividades que pueden estar compuestas por grupos de instrucciones máquina que inmediatamente pueden ejecutarse. Los intérpretes tienen muchas ventajas, el paso del código escrito al código ejecutable es casi inmediato, y el código fuente está siempre disponible para que el intérprete pueda ser mucho más específico cuando ocurra un error. Sin embargo, cuando se crean grandes proyectos tienen importantes limitaciones ya que el intérprete, o una versión simplificada, debe estar siempre en memoria para ejecutar el código lo que puede

introducir restricciones de rapidez inaceptables.

Un Compilador traduce código fuente en instrucciones máquina directamente. Es un proceso complicado que normalmente, sigue una serie de etapas. Dependiendo de el tipo de compilador, los programas generados tienden a requerir mucho menos espacio para ejecutarse, y lo hacen más rapidamente. Cabe destacar que se puede realizar compilación por módulos independientes.

1.1.5 POR QUE SE NECESITAN OBJETOS

Los objetos son necesarios para dar solución a problemas relacionados con los lenguajes de programación tradicionales, estos son la creación de entidades múltiples y la abstracción de datos.

CREAR ENTIDADES MULTIPLES

Cuando se utilizan lenguajes de procedimientos tradicionales, las funciones no se adaptan fácilmente a nuevas situaciones. Para manejar entidades múltiples debe declararse una estructura de datos capaz de soportar tada la información necesaria para cada entidad.

Si se escribe el código utilizando las características orientadas a objetos, la creación de entidades múltiples es bastante sencilla: se declara una estructura de datos para cada entidad, y se llama a las funciones con la dirección de una estructura como parámetro, de esta forma es fácil crear una entidad como varias entidades.

ABSTRACCION DE DATOS

Cuando se crea una estructura de datos en un lenguaje procedimental, se crean normalemente funciones para manipular la estructura, que pueden utilizarse unicamente en esa estructura. Otras funciones no sabrían como manipular la

estructura, ya que ésta no es un tipo de dato incorporado.

La programación orientada a objetos soporta entidades múltiples y bibliotecas mediante la **abstracción de datos**. La abstracción de datos significa que se puede combinar la estructura de datos y las operaciones de esa estructura en un nuevo tipo de dato abstracto, el cuál se comporta como los tipos de datos que son incorporados al lenguaje, con la única excepción de que no son parte de la definición de lenguaje, sino de algo que ha sido creado por el programador. Este nuevo tipo de dato se denomina clase.

1.1.6 ORGANIZACION DEL CODIGO

Los programas orientados a objetos tienen dos conceptos separados: **descripción de la interfaz** y la **implementación de una clase**. El interfaz describe qué hace la clase, y la implementación define como funciona ésta.

COMO ENCONTRAR OBJETOS EN UN SISTEMA

Un objeto en el programa de la computadora, debería corresponder, lo más exactamente posible, a un objeto en el problema real, de esta manera los programas son más faciles de reparar y modificar y más baratos de mantener.

1.1.7 VENTAJAS DE LA HERENCIA

En un lenguaje orientado al objeto, se puede **heredar** las características de un tipo definido por el usuario (clase) en otro. La herencia es útil por dos motivos: soporte para codificar y creación de programas extensibles.

SOPORTE

La herencia da soporte a la codificación ya que ayuda a reutilizar el código en la clase. Si tiene una clase operacional que ya ha sido escrita, no necesita recurrir al código fuente y entender la implementación; únicamente se

hacen cambios donde se considera necesario y se reutiliza el código viejo.

CREACION DE PROGRAMAS EXTENSIBLES

La creación de programas extensibles se refiere al concepto de **subclase** orientado al objeto que ayuda a crear soluciones más fáciles de mantener y ampliar.

Cuando una clase derivada hereda de una clase base, los objetos de la clase derivada retienen, todavía, el número de miembros en la clase base. Mediante la derivación de muchas clases a partir de la misma clase base, se puede crear un grupo de clases que tengan el mismo interfaz, pero diferentes implementaciones.

La utilización de interfaces idénticos con diferentes implementaciones se denomina **Polimorfismo**. Para ampliar un programa polimórfico, simplemente derive una nueva subclase a partir de la misma clase base que heredaron los otros objetos genéricos. La nueva

subclase puede manejarse por el mismo programa sin modificación.

1.1.8 EMPLEO DE SOBRECARGA

El concepto de un **Objeto Inteligente** que decida que hacer con un mensaje tiene una característica adicional en la programación orientada a objetos.

La **Sobrecarga** significa que un nombre de una función puede utilizarse de diferentes formas, dependiendo de la lista de argumentos. Sobrecargar un operador implica que se le dá al operador un significado diferente, de acuerdo al nuevo tipo de dato con que se emplea.

1.1.9 DEFINICION DE UN LENGUAJE ORIENTADO AL OBJETO

" Programar en un lenguaje orientado a objetos supone crear nuevos tipos de datos llamados **clases** y enseñar a estos tipos de datos como manejar **mensajes**. Mediante un **método** se le dice a una clase qué hacer con un mensaje. Se

definen variables de este tipo de dato y se denominan **objetos o instances,** es a estos objetos que se envían los mensajes. "

- Un lenguaje orientado al objeto o a objetos, soporta tres características básicas: escritura de datos abstractos, herencia y polimorfísmo.
- Las clases pueden crearse por medio de la herencia y pueden tener características polimórficas.

- La abstracción y ocultación de datos reduce la dependencia entre los módulos. Esto significa que la modificación del módulo de una sección no afectará al código en otra sección, debido a que los cambios no repercuten en todo el sistema.

- La herencia ayuda a la reutilización del código. Cuando se utiliza con el polimorfísmo, la herencia es útil para reutilizar diseños del programa y crear programas extensibles. Esto no es automático, por lo que debe incorporarlo al sistema.

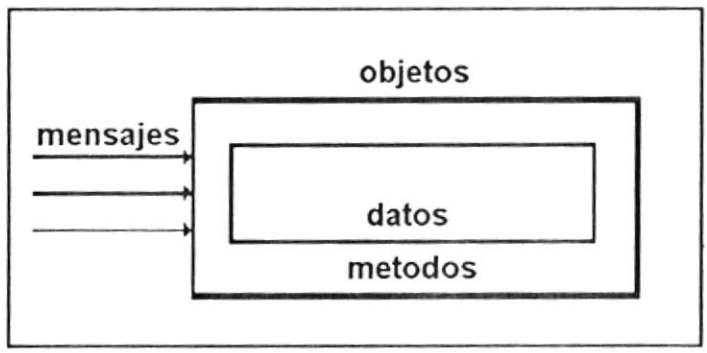


Fig No 1.- Esquema de la Estructura de un Objeto

1.2 LENGUAJE C++ Y UTILITARIO WINDOWS

1.2.1 HISTORIA DEL LENGUAJE C++

El lenguaje C++ nació en los laboratorios Bell de AT&T y ha sido estrechamente asociado con el sistema operativo UNIX, ya que su desarrollo se realizó en este sistema y debido a que tanto UNIX como el propio compilador C y la casi totalidad de los programas y herramientas de UNIX fueron escritos en C. Su eficiencia y claridad han hecho que el lenguaje ensamblador apenas haya sido utilizado en UNIX.

Este lenguaje está inspirado en el lenguaje B escrito por **Ken Thompson** en 1970 con intención de recodificar el UNIX, que en la fase de

arranque estaba escrito en ensamblador, en vista a su transportabilidad a otras máquinas. B era un lenguaje evolucionado e independiente de la máquina, inspirado en el lenguaje BCPL concebido por **Martin Richard** en 1967.

En 1972, **Dennis Ritchie,** toma el relevo y modifica el lenguaje B, creando el lenguaje C y reescribiendo el UNIX en dicho lenguaje. La novedad que proporcionó el lenguaje C sobre el B fue el diseño de tipos y estructuras de datos.

Los tipos básicos de datos eran **char** (carácter), **int** (entero), **float** (reales en simple precisión) y **double** (reales en doble precisión). Posteriormente se añadieron los tipos **short** (enteros con longitud menor o igual de un entero), **long** (enteros de longitud mayor o igual a la longitud de un entero), **unsigned** (enteros sin signo) y **enumeraciones**. Los tipos estructurados básicos de C son las **estructuras**, las **uniones** y los **arrays**. Estos permiten la definición y declaración de tipos derivados de mayor complejidad.

Las instrucciones de control de flujo de C son las habituales de la programación estructurada: **if**, **for**, **while**, **switch-case** todas incluidas en su predecesor BCPL.

C incluye también punteros y funciones. Los argumentos de las funciones se pasan por valor, esto es copiando su valor, lo cual hace que no se modifiquen los valores de los argumentos en la llamada. Cuando se desea modificar los argumentos en la llamada, éstos se pasan por referencia, es decir, se pasan las direcciones de los argumentos. Por otra parte, cualquier función puede ser llamada recursivamente.

Una de las peculiaridades de C es su riqueza de operadores. Puede decirse que prácticamente dispone de un operador para cada una de las posibles operaciones en código máquina.

Hay toda una serie de operaciones que pueden hacerse con el lenguaje C, que realmente no están incluidas en el compilador propiamente dicho, sino que las realiza un **preprocesador** justo antes de cada compilación. Las dos más importantes son **# define** (directriz de

sustitución simbólica o de definición) e **#include** (directriz de inclusión en el fichero fuente).

Finalmente, C, que ha sido pensado para ser altamente transportable y para programar lo improgramable, igual que otros lenguajes tiene sus inconvenientes. Carece de instrucciones de entrada/salida, de instrucciones para manejo de cadenas de caracteres, con lo que este trabajo queda para la librería de rutinas, con la consiguiente pérdida de transportabilidad. La excesiva libertad en la escritura de los programas puede llevar a errores en la programación que, por ser correctos sintácticamente no se detectan a simple vista.

Por otra parte las precedencias de los operadores convierten a veces las expresiones en pequeños rompecabezas. A pesar de todo, C ha demostrado ser un lenguaje extremadamente eficaz y expresivo.

Este lenguaje a evolucionado paralelamente a Unix. Así, en 1980, se añaden al lenguaje C, características como: **clases**, chequeo y

conversión de los tipos de argumentos de función, etc.; el resultado fué el lenguaje denominado **"C con Clases"**.

En 1983/84 , "C con Clases" fue rediseñado, extendido y nuevamente implementado. El resultado se denominó **"Lenguaje C++"** . Las extensiones principales fueron: funciones virtuales y operadores con varias funciones.

Después de algún otro refinamiento más, **C++** queda disponible en 1985. Este lenguaje fue inventado por Bjarne Stroustrup (AT&T Bell Laboratories) y documentado por varios libros suyos.

El nombre de C++ se debe a Rick Mascitti, significando "el carácter evolutivo de las transformaciones de C" ("++" es el operador de incremento de C).

1.2.2 CARACTERISTICAS DE C++

- Permite definir nuevos tipos con sus propias caractísticas.

- La clase es el concepto clave de C++.
- Abstracción de datos.
- Permite la programación orientada a objetos.
- Sustitución de funciones en línea.
- Chequeo y conversión de tipos de función.
- Varias funciones pueden compartir el mismo nombre.
- Permite definir una función como miembro de una estructura
- Incluye operadores para reservar y liberar memoria.

1.2.3 C++ COMO LENGUAJE ORIENTADO A OBJETOS

La terminología utilizada en la programación orientada a objetos varía en función del lenguaje utilizado.

TABLA N° I

TERMINOLOGIA DEL LENGUAJE C++

CONCEPTO	C++
Objeto	Objeto
Clase	Clase
Metodo	Funcion Miembro
Variables Asociadas	Datos Miembro
Mensaje	Llamada a Funcion

C++ soporta la escritura de datos abstractos con el constructor **class**. Los elemetos de la clase (miembros) pueden incorporarse en tipos de datos, objetos o funciones. Los miembros pueden ser **public** (disponibles para todos), **private** (unicamente disponibles para algunos miembros) o **protected** (disponibles para los miembros de la clase y de las clases heredadas). La combinación de, por ejemplo, funciones y datos se denomina **encapsulación**. La posibilidad de impedir que los datos sean vistos por personas no autorizadas se denomina **ocultación de los datos**.

C++ soporta la herencia. Se puede derivar un nuevo tipo definido por el usuario a partir de uno ya existente, y hacer cambios únicamente cuando se necesiten. La herencia contribuye a una reutilización del código más fácil y es fundamental para implementar el polimorfismo.

C++ soporta el polimorfismo con la palabra clave **virtual**. Cuando se crea una función virtual en una clase base, puede redefinirse en una clase derivada.

La sobrecarga de función proporciona el soporte para el concepto general de "enviar un mensaje a un objeto e indicarle qué hacer con él".

C++ soporta la programación orientada al objeto sin perder la eficacia de C.

1.2.4 UTILITARIO WINDOWS

C++ permite crear proyectos con utilitario windows, lo cuál nos proporciona un ambiente de trabajo sencillo y divertido.

En windows, todas las operaciones se realizan dentro de los límites de la ventana de la aplicación. También permite crear un icono de la aplicación, el cuál es un pequeño gráfico que representa la aplicación.

Los comandos de windows se presentan en menús. Cada aplicación tiene sus propios menús situados en el extremo superior de cada ventana de aplicación. Asimismo, cada aplicación posee un menú Control que se abre desde un pequeño cuadro situado en la esquina superior izquierda de cada ventana.

CAPITULO II

DESCRIPCION DE LOS RECURSOS DE HARDWARE

2.1 PUERTO SERIAL

La interfaz en serie RS232 está, a nivel teórico, definida con toda precisión según la normativa de la Asociación de Industrias Eléctricas. La norma especifica el tipo de conector a utilizar, un "miniatura D" de 25 patillas, las señales destinadas a cada patilla y los niveles de señales. De las muchas patillas que presenta la interfaz estándar, normalmente solo se utilizan 2, para transmitir los datos en serie desde el ordenador del periférico; la 3, para recibir los datos transmitidos desde éste, y la 7, la señal a tierra. Tanto los dispositivos de transmisión como los de recepción se han de ajustar para que la velocidad de transmisión de los datos y el formato de los datos transmitidos sean los mismos.

La RS232 es una interfaz en serie "bidireccional", con una patilla (terminal) para transmitir datos desde el ordenador y otro terminal destinado a recibirlos. Los datos se envían de un bit cada vez por vía del terminal de "datos transmitidos" y se

reciben a través del terminal de "datos recibidos". El flujo de bits puede asumir diversos formatos estandarizados, lo importante es que los dispositivos de recepción y de transmisión empleen el mismo.

Dado que cada byte de información se envía hacia afuera como un flujo de bits en serie, el software que controla la interfaz indica cuando comienza el primer bit del dato y cuando ha terminado el último bit. El procedimiento utilizado tiene un único "bit de inicio" (0, en lógica de Boole), seguido de ocho bits de datos, a continuación se encuentra el "bit de parada" (un 1 lógico).

Es necesario especificar inicialmente la velocidad a la cuál se transmiten los datos, ya que, de lo contrario, se podría malinterpretar el patrón de impulsos que representa en forma de ceros y unos a los ocho bits de datos. Esta velocidad de transmisión de datos se denomina **velocidad baudio**, en honor del francés Baudot, que la inventara en el siglo XIX. Las velocidades baudio oscilan entre 75 y 9.600 baudios. Estas cifras corresponden a los 75 y 9.600 bits por segundo que se transmiten, y como normalmente hay 10 bits (incluyendo los bits de inicio y de parada) para cada carácter, la velocidad de transmisión de

caracteres equivale a la décima parte de la velocidad baudio.

La figura a continuación detalla los pines del puerto serial.

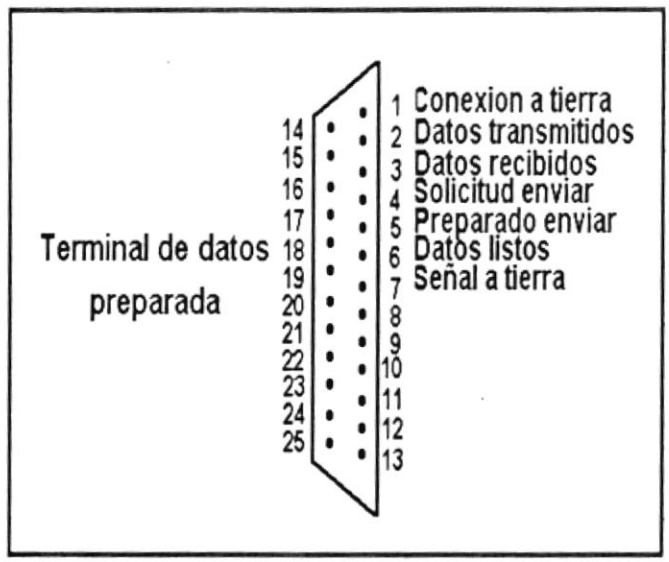


Fig No 2.- Funciones estandarizadas para las patillas de la interfaz RS232.

2.2 RATON

El primer ratón se patentó en el Stanford Research Institute de California en 1.970.

La figura siguiente ilustra un ratón.

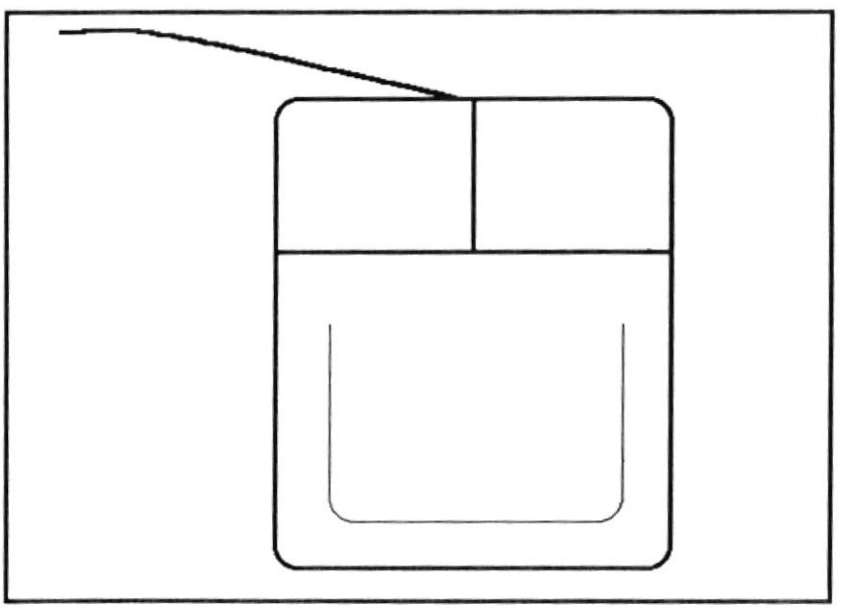


Fig No 3.- Ratón Estándar

El ratón funciona detectando su movimiento a través de cualquier superficie plana en las direcciones arriba- abajo e izquierda-derecha, así como en las combinaciones de ambas. Estos movimientos se convierten directamente en desplazamientos del cursor

(o señalador) en la pantalla. Existen dos métodos para generar las señales eléctricas del movimiento del ratón. En ambos procedimientos, en la cara inferior del ratón hay una bola grande que se apoya en la superficie sobre la cuál se está moviendo.

La rotación del punto de apoyo de la bola del ratón se transfiere a unos cojinetes cilíndricos internos.

En uno de los sistemas, los extremos de estos cilindros están provistos de ruedas de código que poseen pistas alternas de material conductor y no conductor. Los impulsos recibidos los cuenta el software operativo del ratón y le permiten dar una lectura de la posición del cursor en la pantalla.

En el otro sistema, los cojinetes llevan acoplados dos discos acanalados. A los discos se les dirige continuamente una luz y, al otro lado de ellos, una célula fotoeléctrica detecta ópticamente el haz. Los impulsos de luz que pasan a través de las ranuras se convierten luego en señales eléctricas, que se tratan de la misma manera que en el sistema mecánico.
Existen, asímismo, otros sistemas. Hay uno, por ejemplo, en que el ratón se utiliza en conjunción con un relleno especial cubierto por un patrón de puntos.

Una luz en el interior del cuerpo del ratón ilumina la superficie de relleno cubierta por el ratón y este patrón lo detecta un chip procesador óptico especial. Cualquier movimiento del ratón hará que cambie el patrón que detecta el chip, que puede calcular al instante hasta dónde se ha movido el dispositivo y en qué dirección. Este sistema ofrece la ventaja de que no posee partes móviles, pero es mucho más caros que los otros.

Una vez que se ha desplazado el cursor hasta el lugar de la pantalla requerido, se puede dar entrada pulsando una de los botones del ratón. El número de botones de que dispone el ratón varía de un fabricante a otro.

La ventaja principal de todos los ratones, y del software que se ha producido para complementarlos, es que los pueden utilizar todas aquellas personas que no poseen experiencia con teclado. En vez de tener que digitar el nombre de un programa o de pulsar ciertas letras o números para seleccionar una función, el usuario simplemente mueve el ratón de modo que el cursor de la pantalla señale la aplicación o el cursor de acción que se requiera, y pulsa el botón para activarlo. La mayoría de los ratones utiliza su propia interfaz especial, pero

ésta se puede enchufar en cualquier conexión RS232, usando el conector estándar de 25 canales.

2.3 JOYSTICK

El joystick transfiere el control de la aplicación a las manos del usuario. Internamente está formado por dos potenciómetros, los que son utilizados frecuentemente en dispositivos electrónicos en los que se tenga que variar el voltaje. Los potenciómetros tienen una serie de resistencias eléctricas y un contacto deslizante. El valor de la resistencia en el circuito cambia al desplazar el contacto. El computador mide el cambio de resistencia y traduce esta información en un dato procesable. Un potenciómetro controla el movimiento vertical y otro el horizontal.

La palanca de mando del joystick está sostenida por dos apoyos basculantes, montados perpendicularmente entre sí y están conectados a los potenciómetros. Cuando se mueve la palanca, los contactos de los potenciómetros deslizan y cambian la resistencia eléctrica.

A continuación se presenta el gráfico del joystick y el circuito potenciométrico del mismo.

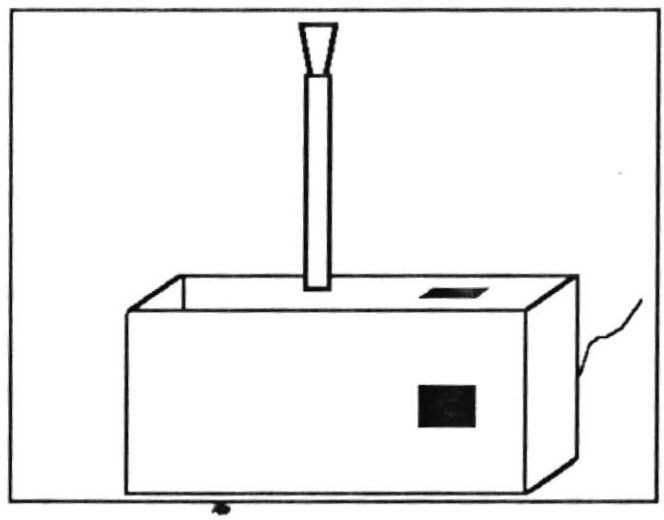


Fig No 4.- Joystick.

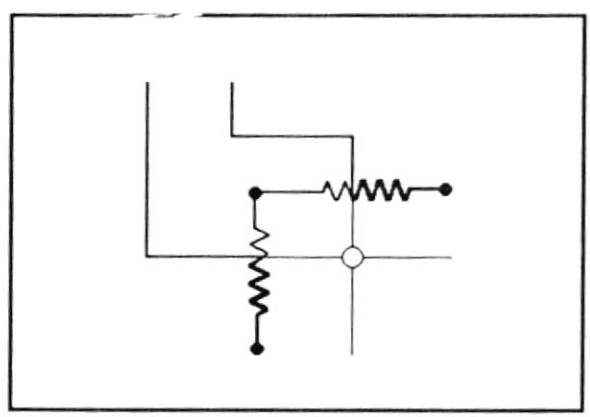


Fig No 5.- Circuito Potenciométrico del Joystick.

CAPITULO III

IMPLEMENTACION DE LA INTERFAZ GRAFICA PARA USUARIO

3.1 PLANTEAMIENTO DEL PROBLEMA

La interfaz gráfica a implementarse permitirá el manejo amigable de la grúa prototipo. El control de la grúa se realizará con el ratón en un ambiente de ventanas o con el joystick, con lo cuál el usuario encontrará divertido y fácil accionar la grúa.

El objetivo es simplificar la selección de los movimientos de la grúa y obtener una respuesta del elemento mecánico en tiempo real.

Los movimientos de la grúa prototipo son los siguientes:

- Subir el gancho,
- Bajar el gancho,
- Abrir el brazo,
- Cerrar el brazo,
- Rotación del cuerpo a la derecha, y
- Rotación del cuerpo a la izquierda.

Se diseñará una interfaz gráfica en un ambiente windows con menues pull-down, la primera opción del menú permitirá seleccionar cualquiera de los seis movimientos de la grúa. La segunda opción del menú permitirá detener el movimiento. En la misma interfaz se podrá realizar el movimiento de la grúa con el joystick, para lo cuál previamente se deberá seleccionar la opción de manejo con joystick y determinar con que dispositivo de la grúa se desea trabajar; los dispositivos son:

- Gancho,
- Brazo, y
- Cuerpo.

3.2 ESTRATEGIA DE SOLUCION

Para desarrollar la interfaz gráfica se utilizó los conceptos de Programación Orientada a Objetos, eligiendo para la implementación el lenguaje Borland C++, versión 3.1. Se escogió trabajar en Borland C++, versión 3.1 debido a que permite crear interfaces con ambiente windows.

3.2.1 PUERTO PARALELO

La interfaz en paralelo transmite la información de un byte cada vez, pero además de las ocho "líneas de datos" necesita también proporcionar otras señales para que el computador y el periférico sepan cuándo es posible transmitir datos y cuándo no. Las señales proporcionadas por la interfaz paralela se detallan en la siguiente tabla.

TABLA Nº II

SEÑALES DE LA INTERFAZ PARALELA

DE DATO 0 A DATO 7	Ocho cables para transportar los ocho bits del byte que se está transmitiendo.
ADK	Una señal de entrada para el computador que indica que el dispositivo receptor está preparado para aceptar datos.
GND	El cable "a tierra" que proporciona una referencia común de 0 voltios tanto al computador como al dispositivo peri-
BUSY	Una señal desde el dispositivo periférico hasta el computador que indica que el periférico no puede aceptar información
STROBE	Una señal de salida desde el computador que le indica al periférico que la información está preparada y debe leerla.

3.2.2 PUERTO DE JUEGOS

No hay ninguna interfaz estándar para el manejo del joystick, aunque muchos fabricantes han adoptado el modelo de Atari. La mayoría de dichas interfaces poseen simplemente cinco líneas activas (una desde cada interrruptor de las cuatro modalidades del movimiento de la palanca de mando, y la quinta para el botón de disparo).

3.2.3 INTERFAZ DIGITAL DE CONTROL DE LA GRUA PROTOTIPO

La interfaz digital utilizada para controlar la grúa prototipo, tiene un puerto de control de entrada y un puerto de control de salida.

El puerto de control de entrada se conecta al puerto paralelo de la computadora. El puerto de control de salida se conecta al panel de motores de la grúa prototipo; tal como se muestra en la figura a continuación:

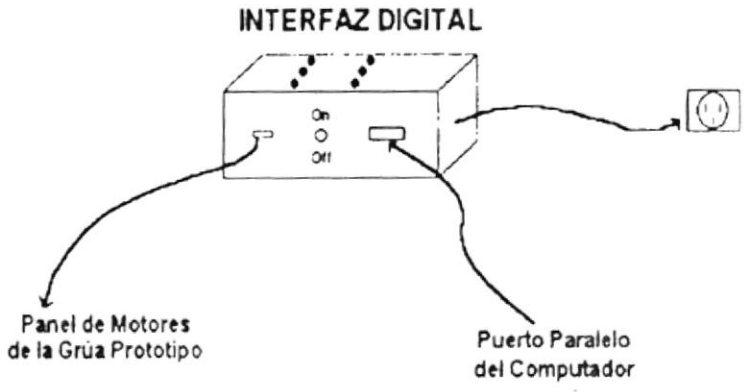


Fig No 6.- Interfaz Digital de Control de la grúa prototipo.

Dependiendo del movimiento que se desea obtener, se envía la configuración correspondiente a la interfaz digital a través del puerto paralelo de la computadora. A continuación la interfaz digital envía la señal eléctrica para accionar el motor de la grúa y de esta forma se realiza el movimiento seleccionado.

Los tipos de configuración de la interfaz digital y los movimientos asociados a estos, se muestran en la tabla siguiente:

TABLA N^{O} III

CONFIGURACION DE LA INTERFAZ DIGITAL Y

MOVIMIENTOS DE LA GRUA ASOCIADOS

CONFIGURACION DE LA TARJETA DIGITAL	MOVIMIENTOS DE LA GRUA ASOCIADOS
11111101b	Subir el Gancho
11111110b	Bajar el Gancho
11111011b	Abrir el Brazo
11110111b	Cerrar el Brazo
11011111b	Rotación Derecha
11101111b	Rotación Izquierda
11111111b	Detener el Movimiento

3.2.4 MANEJO DEL JOYSTICK

El Joystick se conecta al puerto de juegos del computador, el cuál tiene un tipo de comunicación serial.

Para controlar la grúa con el joystick, se debe realizar una lectura de la señal recibida desde el puerto de juegos al accionar los

conmutadores o mover la palanca de mando del joystick.

El código utilizado para obtener la lectura del puerto de juegos, y las rutinas de prueba realizadas, se encuentran en el Apendice A.

De iguál manera que con la selección de opciones desde el menú pull-down, al activar alguna señal del joystick, se configura la interfaz digital para mover la grúa prototipo.

3.2.5 ESQUEMA DE CONEXIONES

Adicionalmente a la conexión con la interfaz digital, se deben conectar el ratón y el joystick. El ratón se conecta a un puerto serial y el joystick al puerto de juegos. De esta forma el esquema completo de conexiones se detalla en la figura siguiente:

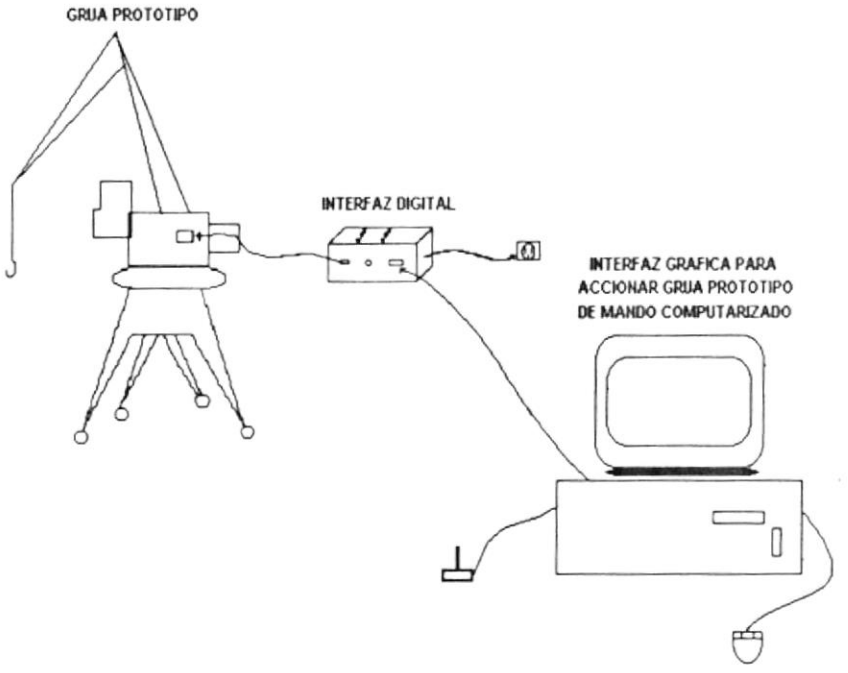


Fig No 7.- Diagrama Completo de Conexiones

El ratón se emplea para seleccionar el movimiento de la grúa en la interfaz gráfica, en un ambiente de ventanas. Con el joystick también se podrá controlar dichos movimientos.

3.2.6 ESTRUCTURA DEL PROYECTO

El proyecto se desarrolló basado en los conceptos de Programación Orientada a Objetos, de esta forma se generaron los siguientes archivos:

- **rec_grua.rc**, contiene los recursos como iconos del menú, bitmaps, y el menú pull-down.
- **grua.cpp**, corresponde al programa fuente del proyecto.
- **helpwind.cpp**, genera la ayuda del proyecto.
- **owl.def**, archivo de definiciones de memoria.
- **bwc.lib**, corresponde a las librerías.

Se crearon dos clases:

- **TGruaApp**,
- **TMyWindow**,

El objeto principal del proyecto es **GruaApp**.

Se definió también la variable global **Joy**, para determinar el dispositivo de la grúa que se va a controlar con el joystick.

3.3 ALGORITMOS

3.3.1 CONTROL DEL MOTOR

El control del motor para cada uno de los movimientos de la grúa, lo realiza la interfaz digital.

Se especificará con **SLPT1** la señal enviada por el puerto paralelo del computador al puerto de control de entrada de la interfaz digital. De acuerdo a la señal de enviada, que puede ser:

11111111b,

11111101b,

11111110b,

11111011b,

11110111b,

11011111b, o

11101111b.

La interfaz digital, a través de su puerto de control de salida, envía la señal **SSID** al panel de control de los motores de la grúa.

Como se mencionó previamente, la grúa prototipo tiene tres motores, cada uno con dos sentidos de rotación:

- **M1A**, Motor 1 en sentido de rotación A.
- **M1B**, Motor 1 en sentido de rotación B.
- **M2A**, Motor 2 en sentido de rotación A.
- **M2B**, Motor 2 en sentido de rotación B.
- **M3A**, Motor 3 en sentido de rotación A.
- **M3B**, Motor 3 en sentido de rotación B.

Cada motor controla un dispositivo de la grúa, así M1 maneja el gancho, M2 el brazo y M3 el cuerpo. Dependiendo del sentido de rotación del motor, se realizarán seis movimientos:

- **M1A**, Subir el Gancho.
- **M1B**, Bajar el Gancho.
- **M2A**, Abrir el Brazo.
- **M2B**, Cerrar el Brazo.
- **M3A**, Rotación a la Derecha.
- **M3B**, Rotación a la Izquierda.

En la tabla siguiente se muestra el control de los motores de la grúa prototipo.

TABLA N^{O} IV

CONTROL DE MOTORES DE LA GRUA PROTOTIPO

SLPT1	SSID
11111101b	M1A
11111110b	M1B
11111011b	M2A
11110111b	M2B
11011111b	M3A
11101111b	M3B
11111111b	Encera el Puerto

El Algoritmo de Control del Motor se describe como sigue:

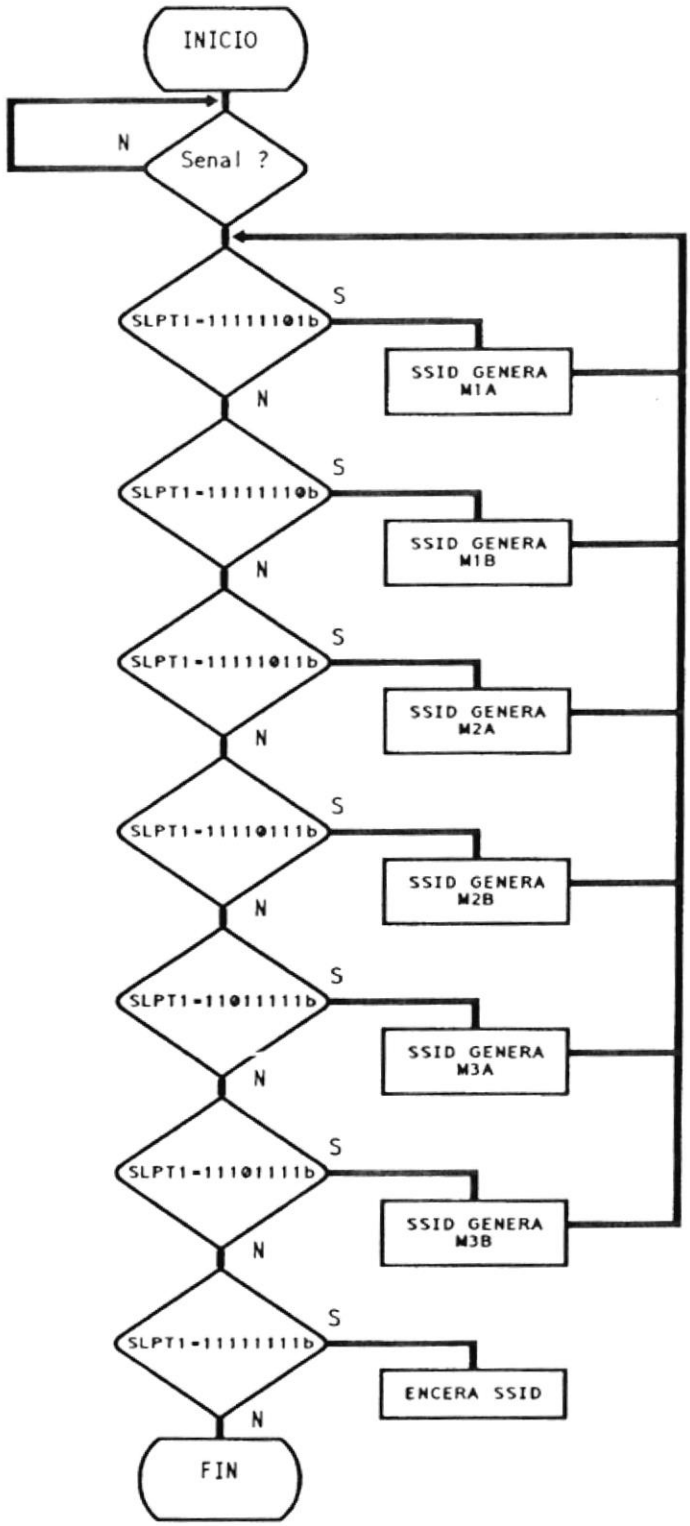


Fig No 8.- Algoritmo de Control del Motor

3.3.2 INTERFAZ GRAFICA

La interfaz gráfica se implementó en Borland C++, utilizando los criterios de programación orientada a objetos.

Inicialmente se definieron las entidades del proyecto y se determinaron las clases necesarias para la implementación del mismo.

La figura a continuación presenta un esquema de todas las clases que intervienen en la interfaz y la relación que existe entre ellas.

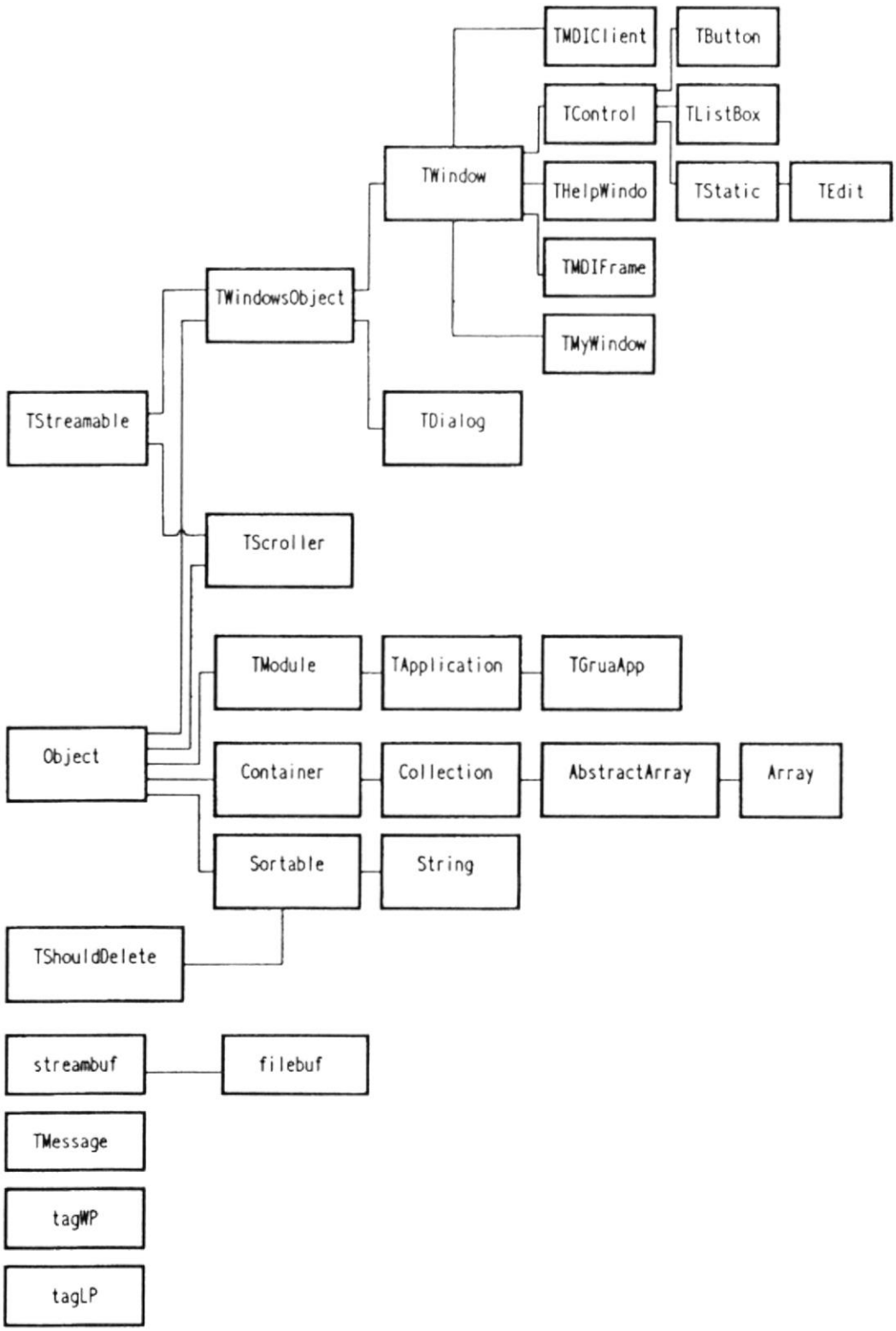
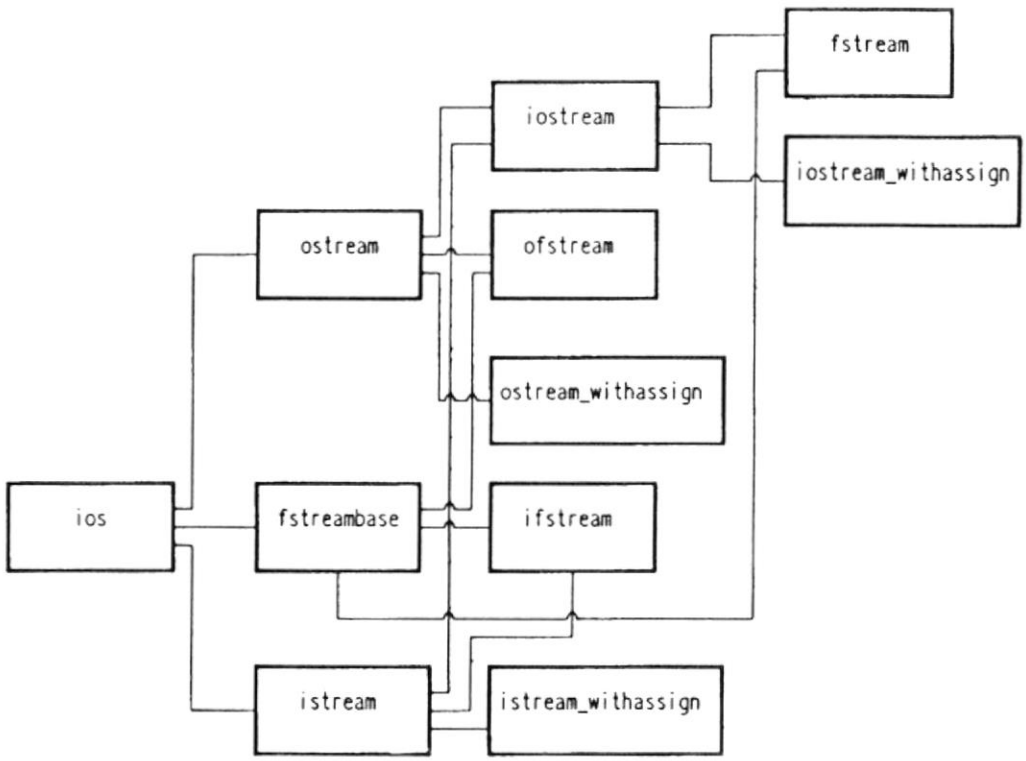
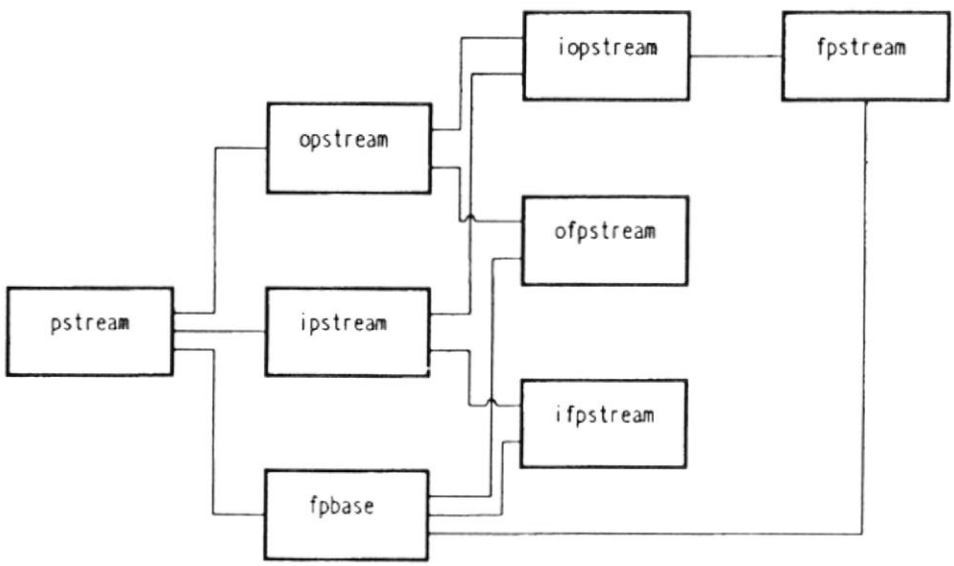


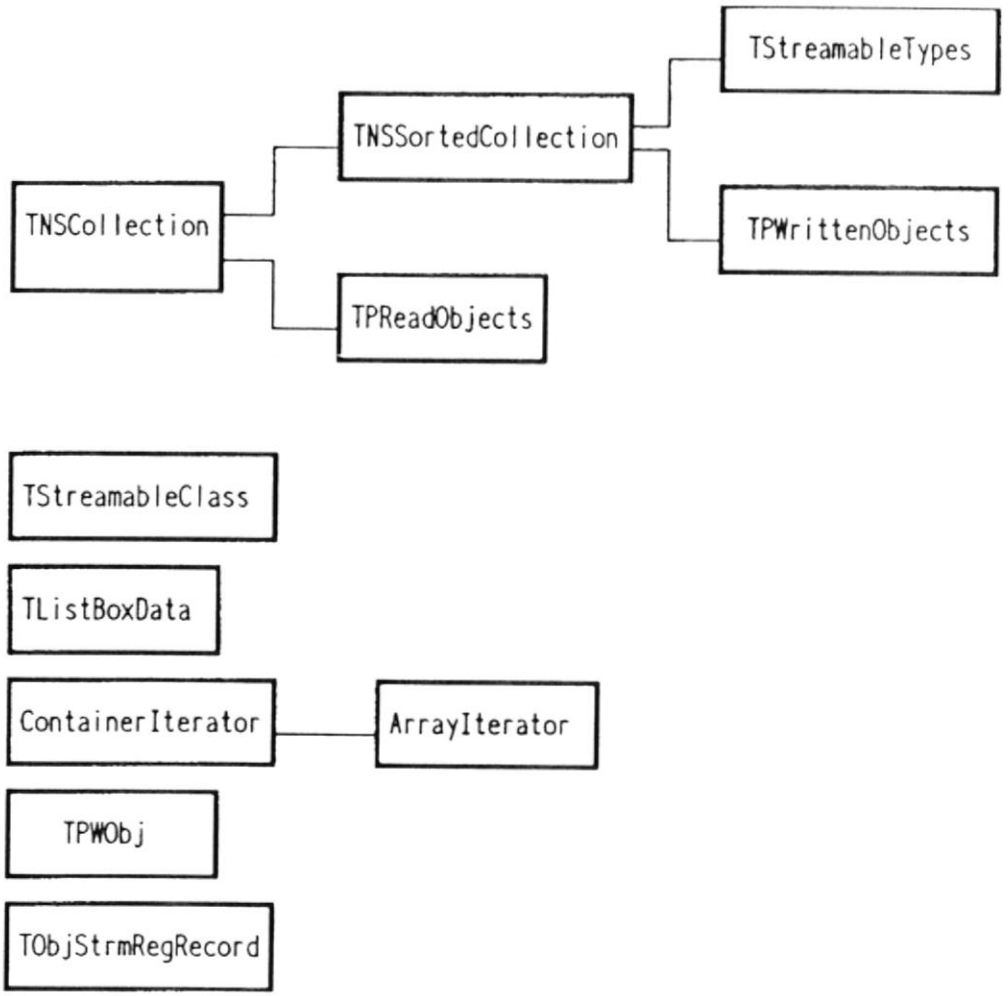
Fig No 9.- Interfaz Gráfica

...CONTINUACIÓN Fig. No 9





...CONTINUACIÓN Fig. No 9



3.4 PRUEBAS DEL PROGRAMA

Al ejecutar el programa, inicialmente se realiza la presentación del proyecto como se muestra a continuación:

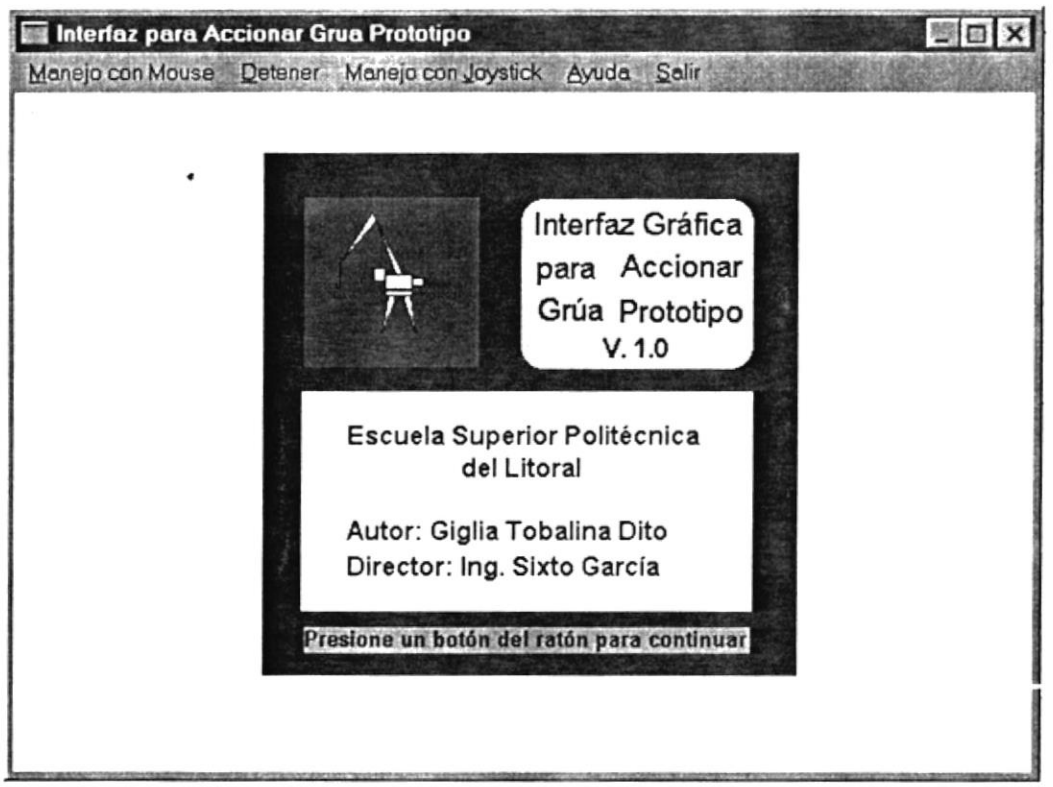


Fig No 10.- Presentación de la Interfaz Gráfica

A continuación presenta una pantalla que tiene el formato de las aplicaciones windows. El título es **Interfaz para Accionar Grua Prototipo.**

Las opciones que muestra el menú principal se pueden seleccionar con el ratón o presionando Alt y la tecla subrrayada. Dichas opciones son:

Manejo con Mouse, Detener, Manejo con Joystick, Ayuda y Salir.

La figura 11 ilustra la pantalla.

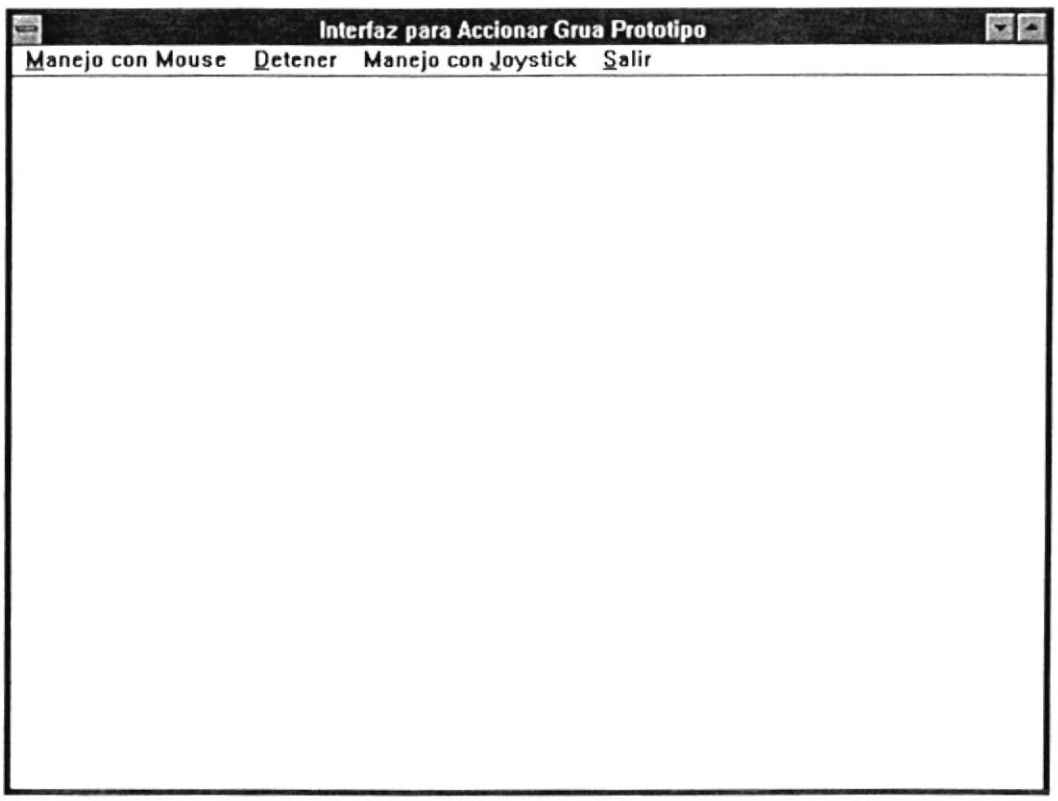


Fig No 11 .- Pantalla Inicial de la Interfaz

Si selecciona Manejo con Mouse, el menú le mostrará las siguientes opciones:

- Subir Gancho,
- Bajar Gancho,
- Abrir Brazo,
- Cerrar Brazo,
- Rotación derecha, y
- Rotación izquierda.

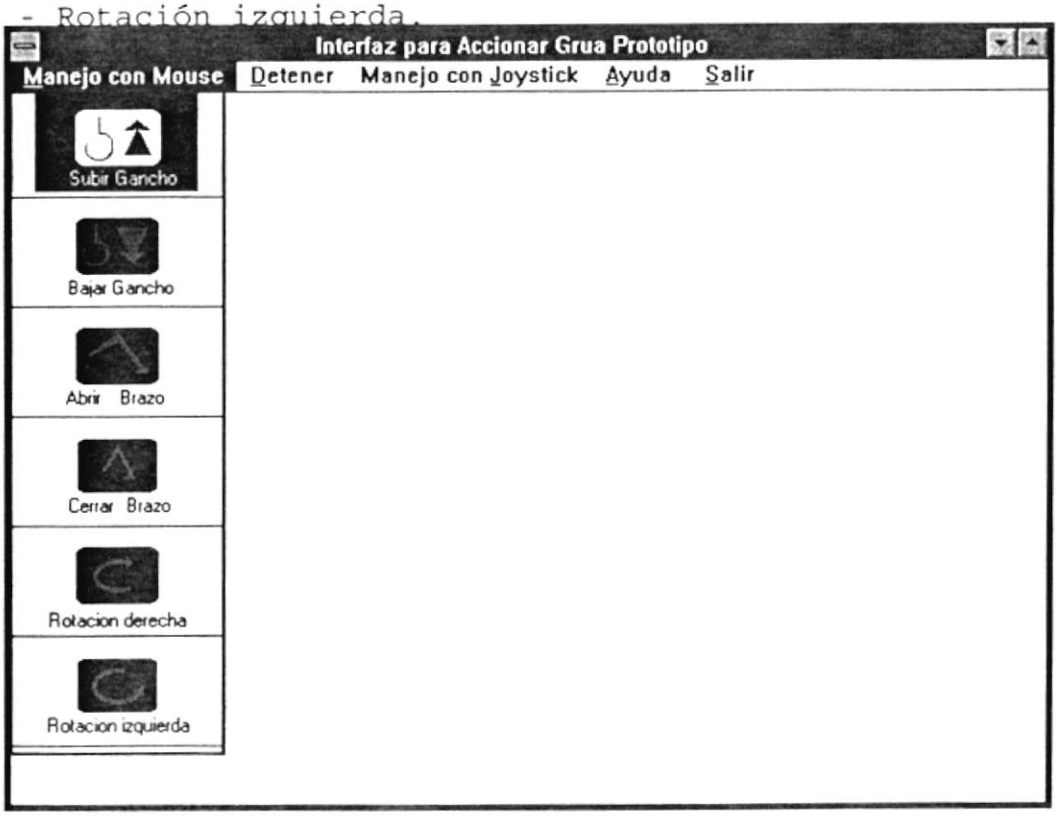


Fig No 12 .- Manejo con Mouse

Cuando se escoge el movimiento, se envía la señal correspondiente al Puerto Paralelo, lo cuál configura la interfaz digital y el elemento mecánico responde inmediatamente.

La segunda opción del menú principal es Detener, con la que se muestra la opción STOP y se envía la señal que detiene el movimiento de la grúa.

A continuación se muestra la pantalla.

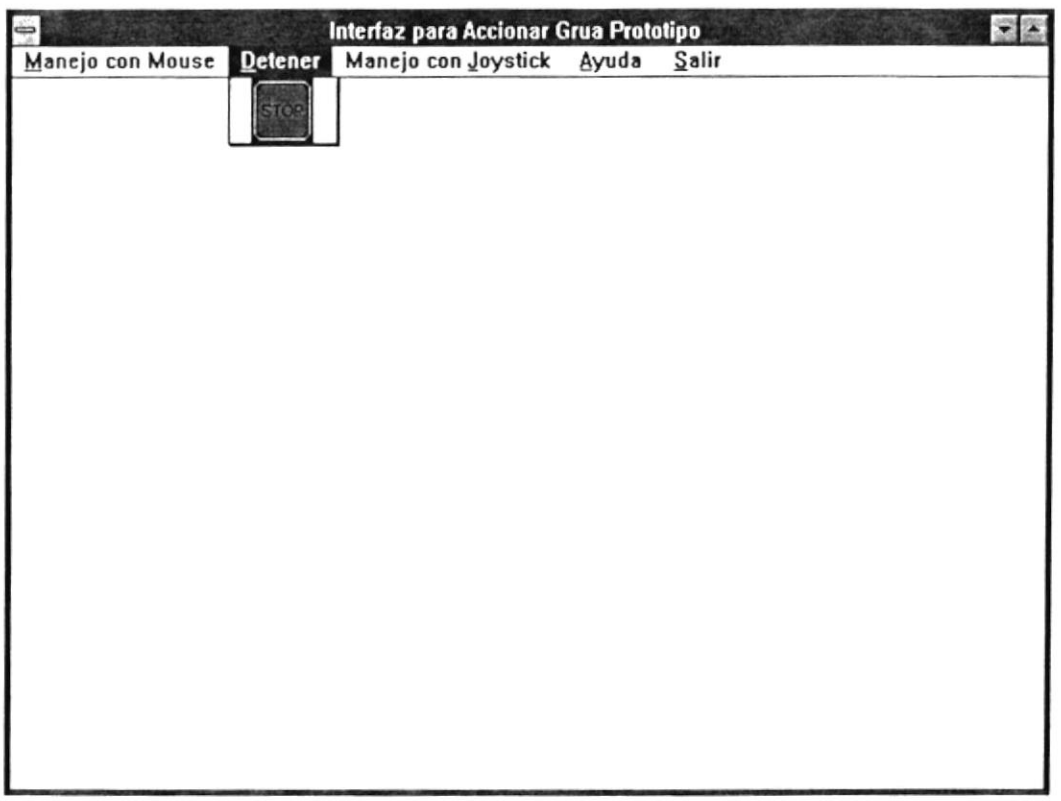


Fig No 13 .- Detener el Movimiento

La tercera opción del menú es la de Manejo con Joystick, en la que se muestran los dispositivos con los que se puede controlar la grúa. Luego que se elige el dispositivo (Gancho, Cuerpo o Brazo), el

control de la interfaz es transferido al Joystick y se procede a manejar el elemento mecánico.

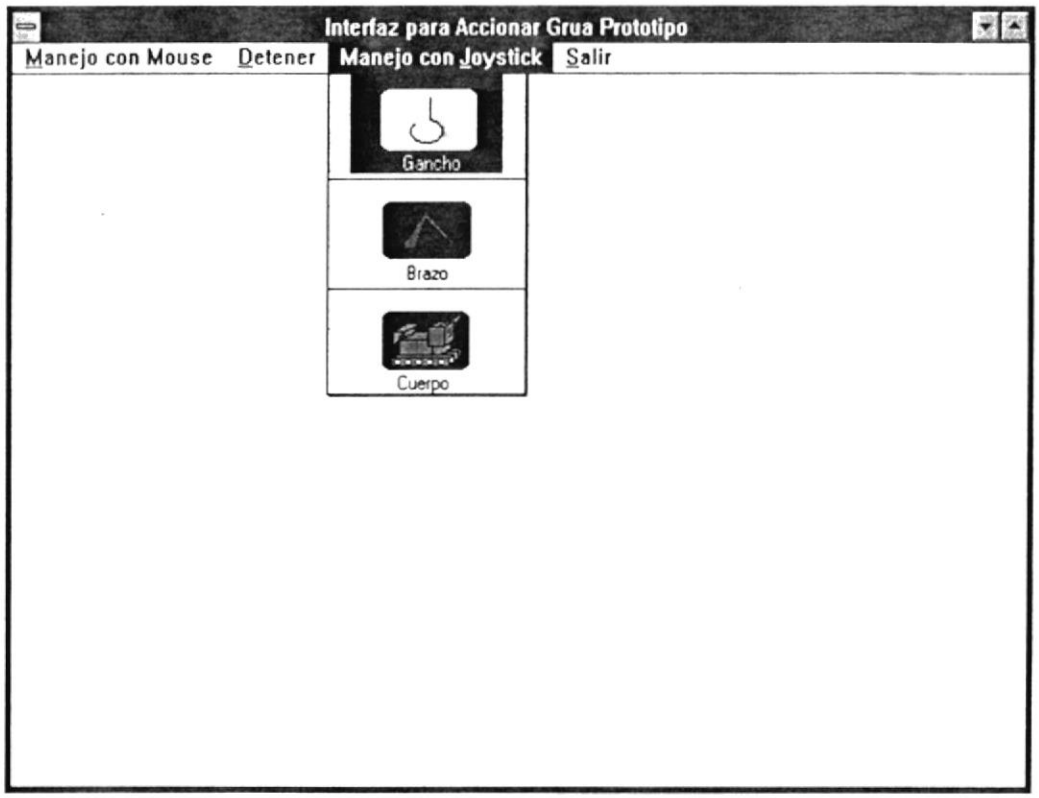


Fig No 14.- Manejo con Joystick

La opción Ayuda presenta una explicación del manejo de la interfaz, así puede seleccionar ayuda de los siguientes temas: conexiones, interfaz digital, manejo con mouse o manejo con joystick.

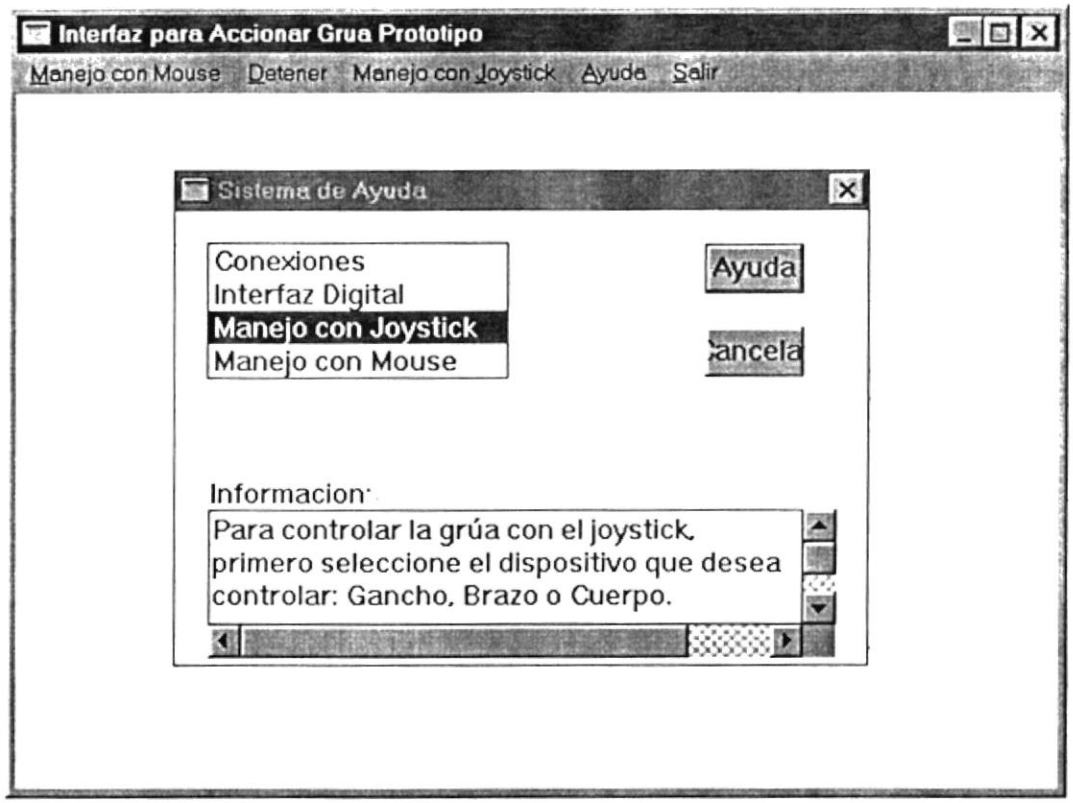


Fig No 15.- Ayuda sobre la Interfaz

Finalmente si selecciona Salir, termina de ejecutar la interfaz. A continuación se muestra la opción en la pantalla.

Fig No 16.- Salir de la Interfaz

La interfaz presenta adicionalmente todas las ventajas de una aplicación windows, es decir si selecciona con el ratón el cuadrado superior izquierdo de la pantalla, se le presentan las opciones:

- Restore,
- More,
- Size,
- Minimize,

- Maximize,
- Close (Alt + F4), y
- Switch To... (Ctrl + ESC).

También podrá cambiar el tamaño de la pantalla y desplazar su ubicación.

CAPITULO IV

USO DE LA INTERFAZ GRAFICA

4.1 DESCRIPCION DEL MANEJO DE LA INTERFAZ

Para manejar la interfaz, inicialmente debe realizar correctamente todas las conexiones de hardware.

Primero coloque el cable del puerto paralelo del computador al puerto de control de entrada de la interfaz digital, luego el cable desde el puerto de control de salida de la interfaz digital al panel de motores de la grúa prototipo. Adicionalmente conecte el joystick al puerto de juegos de su computador y el ratón al puerto serial o al puerto del ratón.

La interfaz para accionar la grúa prototipo es una aplicación de windows, por lo cuál necesita tener instalado windows 3.1 en su computador.

Al ejecutar la interfaz gráfica incialmente se muestra una presentación del proyecto, la cuál incluye el nombre de la aplicación, versión, autor,

director del proyecto y deberá dar un click del ratón para continuar.

El menú principal muestra las siguientes opciones: Manejo con Mouse, Detener, Manejo con Joystick, Ayuda y Salir.

Dichas opciones se pueden seleccionar con el ratón o presionando Alt y la tecla subrrayada.

Al seleccionar Manejo con Mouse, el menú le permitirá: Subir Gancho, Bajar Gancho, Abrir Brazo, Cerrar Brazo, Rotación derecha, y Rotación izquierda.

Cuando se inicia alguno de los movimientos de la grúa, seleccionando STOP dentro de la opción Detener, detenemos el movimiento.

Si desea controlar la grúa prototipo con el joystick, seleccione en el menú Manejo con Joystick, y escoja el dispositivo con que trabajará: Gancho, Cuerpo o Brazo, de esta forma el control de la grúa se realiza con el joystick de la siguiente manera:

- Gancho: Presione el botón rojo para subir el gancho, presione los botones negro y rojo al mismo tiempo para bajar el gancho. Presione el botón negro para detener el movimiento.

- Brazo: Presione el botón rojo para abrir el brazo, presione los botones negro y rojo al mismo tiempo para cerrar el brazo. Presione el botón negro para detener el movimiento.

- Cuerpo: Presione el botón rojo para rotar a la derecha; presione los botones negro y rojo al mismo tiempo para rotar a la izquierda. Presione el botón negro para detener el movimiento.

Puede seleccionar Ayuda para obtener una explicación del manejo de la interfaz.

Finalmente si selecciona Salir, termina de ejecutar la interfaz.

4.2 APLICACIONES PRACTICAS

La aplicación más importante de la interfaz gráfica está en el campo de la medicina, ya que por su fácil manejo puede ser utilizada por minusválidos.

Recordemos que la grúa prototipo puede ser reemplazada por otro elemento mecánico de tres motores.

En las industrias, sirve para automatizar procesos, ya que el proyecto desarrollado para equipos reales, permitiría controlar máquinas desde un cuarto de control central.

También puede ser utilizada como un juego para niños.

CONCLUSIONES Y RECOMENDACIONES

- Se utilizaron los conceptos de programación orientada a objetos en la implementación de la interfaz gráfica, la programación orientada a objetos es una nueva forma de pensar cómo resolver algunos de los problemas planteados en el mundo de la informática. En lugar de tratar de adaptar el problema a algún aspecto familiar al computador, el computador se adapta al problema. Por esta razón la programación orientada a objetos facilita la solución de los grandes problemas y nos permite obtener aplicaciones de fácil manejo y portabilidad.

- Para implementar la interfaz gráfica se establecieron las clases, sus métodos, y los objetos de estos nuevos tipos de datos, con lo cuál se aplicó la programación orientada a objetos.

- El control de la interfaz con el ratón y con el joystick, se desarrolló con un ambiente windows, lo cuál permite accionar la gúa de una manera amigable al usuario. Con esta aplicación se manejaron los puertos

del computador, tanto para enviar como para recibir señales.

- Existieron ciertas dificultades en la lectura de información del puerto de juegos ya que no existe mucha documentación del mismo.

- Se presentó el inconveniente de que el joystick que se utiliza en el proyecto, únicamente envía tres señales de control. Por esta razón en el manejo de la grúa con joystick se debe seleccionar previamente en el menú, el dispositivo que desea controlar. Cabe destacar que la interfáz gráfica permite que se adapte cualquier joystick, ya que realiza una lectura del puerto y en base a esa señal mueve el elemento mecánico.

- La comunicación entre el computador y la interfaz digital es rápida, por lo que la respuesta del movimiento de la grúa, se realiza instantáneamente cuando se ha seleccionado algún movimiento, lo que implica que el tiempo de respuesta a las señales es inmediato.
- Es necesario que el computador tenga un puerto de juegos, para el manejo de la grúa con el joystick
- Por ser una aplicación para windows, necesita ejecutar la interfaz gráfica bajo ambiente windows 3.1.

De acuerdo a la experiencia realizada, se pueden establecer las siguientes recomendaciones:

- Verifique que todas las conexiones de hardware se encuentren correctamente realizadas.

- En caso de cambiar el joystick por uno con mayor número de señales, se debe eliminar la opción de manejo con joystick y controlar la grúa únicamente moviendo el joystick o seleccionando los movimientos en el menú con el ratón.

APENDICE A

LISTADO DE PROGRAMAS

```
/* GRUA.CPP: INTERFAZ GRAFICA PARA ACCIONAR GRUA
PROTOTIPO DE MANDO COMPUTARIZADO.*/


// Archivos de cabecera

#include <dos.h>
#include <mem.h>
#include <stdio.h>
#include <string.h>
#include <owl.h>
#include <conio.h>

#include "helpwind.h"
#include "grua.h"

int joy;
BOOL timeover,bandera;

// Class para definir la aplicacion grua

class TGruaApp : public TApplication
{
   public:
        TGruaApp(LPSTR ApName, HINSTANCE Inst, HINSTANCE
PrevInst
          , LPSTR CmdLine, int CmdShow) :
TApplication(ApName
          , Inst, PrevInst,CmdLine,CmdShow){};
          virtual void InitMainWindow();
          virtual void IdleAction();
};


// Class para definir las propiedades de la pantalla

class TMyWindow : public TWindow
{
  public:
        HBITMAP Grua[10];

        TMyWindow(PTWindowsObject Aparent, LPSTR
ATitle);
        virtual void SetupWindow();
        virtual void CloseWindow();
        virtual void Subir(RTMessage) = [CM_FIRST +
101];
        virtual void Bajar(RTMessage) = [CM_FIRST +
102];
        virtual void Derecha(RTMessage) = [CM_FIRST +
105];
```

```
        virtual void Izquierd(RTMessage) = [CM_FIRST +
106];
        virtual void Abrir(RTMessage) = [CM_FIRST +
103];
        virtual void Cerrar(RTMessage) = [CM_FIRST +
104];
        virtual void Detener(RTMessage) = [CM_FIRST +
107];
        virtual void Salir(RTMessage) = [CM_FIRST +
111];
        virtual void Gancho(RTMessage) = [CM_FIRST +
108];
        virtual void Brazo (RTMessage) = [CM_FIRST +
109];
        virtual void Cuerpo (RTMessage) = [CM_FIRST +
110];
        virtual void Ayuda (RTMessage) = [CM_FIRST +
998];
        virtual void WMLButtonDown(RTMessage) =[WM_FIRST
+ WM_LBUTTONDOWN];
        virtual void WMRButtonDown(RTMessage) =[WM_FIRST
+ WM_RBUTTONDOWN];
        virtual void WMDestroy (TMessage& Message ) =
[WM_FIRST + WM_DESTROY ];
        void DrawBMP(HDC PaintDC,int x,int y,HBITMAP
BitMap );
        virtual void Paint(HDC PaintDC,PAINTSTRUCT&
PaintInfo);
        private:
          HDC PaintDC;
          int Row, Col;
       HBITMAP GRUA;

 };


TMyWindow::TMyWindow(PTWindowsObject Aparent, LPSTR
ATitle)
      : TWindow(Aparent, ATitle)
{
   AssignMenu("MENU_GRUA");

   //Medidas de la pantalla

      Attr.X = 0;
      Attr.Y = 0;
      Attr.W = 640;
      Attr.H = 480;

}
```

```
void TMyWindow::Ayuda(RTMessage)
{
  PTWindow HelpWindow;

  HelpWindow = new THelpWindow(this);
  HelpWindow->Attr.Style |= WS_POPUPWINDOW |
WS_CAPTION;
  HelpWindow->Attr.X = 100;
  HelpWindow->Attr.Y = 100;
  HelpWindow->Attr.W = 420;
  HelpWindow->Attr.H = 300;
  GetApplication()->MakeWindow(HelpWindow);
}


void TMyWindow::Gancho(RTMessage)
{
 joy=1;
}

void TMyWindow::Brazo(RTMessage)
{
 joy=2;
}

void TMyWindow::Cuerpo(RTMessage)
{
 joy=3;
}


// Constructor de Pantalla

void TMyWindow::SetupWindow()
{
  int i;
  HMENU MnHand;
  char TmpStr[20];

  MnHand = GetMenu(HWindow);
  GRUA= LoadBitmap( GetApplication()->hInstance,
"GRUA");
  joy=1;
  for(i=0; i<10; ++i)
  {
    GetMenuString(MnHand, MID_SUBIR +i,
TmpStr,20,MF_BYCOMMAND);
    Grua[i] = LoadBitmap (GetApplicationObject()-
>hInstance, TmpStr);
    ModifyMenu(MnHand, MID_SUBIR+i,
MF_BYCOMMAND|MF_BITMAP
```

```
                , MID_SUBIR+i, (LPSTR) MAKELONG (Grua[i],
0));
    }
     TWindow::SetupWindow();
}

 void TMyWindow::WMDestroy (TMessage& Message )
 {
     DeleteObject (GRUA);
        TWindow::WMDestroy (Message);
 }


// Destructor de Pantalla

  void TMyWindow::CloseWindow()
  {
     int i;

     for(i=0; i<10; ++i)
     DeleteObject(Grua[i]);
     TWindow::CloseWindow();
   }


// BITMAP EN LA PANTALLA:

// Manejo de la pantalla

   void TMyWindow::Paint(HDC PaintDC,PAINTSTRUCT&)
 {
 if (timeover)
      {
       DrawBMP(PaintDC,40,10,GRUA);
      }
      else
       bandera=FALSE;
 }
 // Botón izquierdo del mouse
 void TMyWindow::WMLButtonDown(RTMessage)
 {
   timeover=FALSE;
   if (bandera)
   InvalidateRect(HWindow,NULL,TRUE);
 }
 // Botón derecho del mouse

 void TMyWindow::WMRButtonDown(RTMessage)
 {
   timeover=FALSE;
   if (bandera)
   InvalidateRect(HWindow,NULL,TRUE);
```

```
 }

 // Dibujo del bitmap

 void TMyWindow::DrawBMP(HDC PaintDC,int x, int y,
HBITMAP BitMap )
{
    HDC MemDC;
    BITMAP bm;
    BOOL MadeDC;

    if (PaintDC == 0)
    {
     PaintDC = GetDC(HWindow);
     MadeDC = TRUE;
    }
    else
     MadeDC = FALSE;
     MemDC = CreateCompatibleDC (PaintDC);
     SelectObject (MemDC,BitMap);
     GetObject (GRUA,sizeof (bm),(LPSTR) &bm );
     BitBlt( PaintDC, x, y, bm.bmWidth,bm.bmHeight,
MemDC, 0, 0, SRCCOPY );
     DeleteDC (MemDC );
     if (MadeDC )
            ReleaseDC (HWindow, PaintDC );
 }


// MOVIMIENTOS DE LA GRUA:


  // Subir el gancho

 void TMyWindow::Subir(RTMessage)
  {
     asm {
     mov dx,3bch;
        mov al,11111101b;
     out dx,al;
     }
  }


  // Bajar el gancho

  void TMyWindow::Bajar(RTMessage)
  {
     asm {
     mov dx,3bch;
        mov al,11111110b;
        out dx,al;
```

```
    }
}

// Rotacion del cuerpo a la derecha

 void TMyWindow::Derecha(RTMessage)
{
    asm {
    mov dx,3bch;
       mov al,11011111b;
       out dx,al;
    }
}

// Rotacion del cuerpo a la izquierda

void TMyWindow::Izquierd(RTMessage)
{
    asm {
    mov dx,3bch;
    mov al,11101111b;
    out dx,al;
    }
}

// Abrir el brazo

void TMyWindow::Abrir(RTMessage)
{
    asm {
    mov dx,3bch;
    mov al,11111011b;
       out dx,al;
    }
}

// Cerrar el brazo

void TMyWindow::Cerrar(RTMessage)
{
    asm {
    mov dx,3bch;
    mov al,11110111b;
    out dx,al;
    }
}
```

```
// Detener el movimiento

void TMyWindow::Detener(RTMessage)
{
   asm {
   mov dx,3bch;
   mov al,11111111b;
   out dx,al;
   }
}


// Salir de Interfaz

void TMyWindow::Salir(RTMessage)
{
   CloseWindow();
}



/* Manejo del joystick*/


void TGruaApp::IdleAction()
{
// Chequeo del puerto

    int port = 0x201;

if (joy==1)
 {
// MANEJO DEL GANCHO

    if(inport(port)==0xFFE0)
    {

    // Subir el gancho
       asm {
         mov dx,3bch;
            mov al,11111101b;
         out dx,al;
         }
     }

 if(inport(port)==0xFFC0)
 {
    // Bajar el Gancho
    asm {
    mov dx,3bch;
       mov al,11111110b;
       out dx,al;
```

```
    }
 }

 if(inport(port)==0xFFD0)
       {
         asm {
           mov dx,3bch;
           mov al,11111111b;
           out dx,al;
         }
     }

   if(inport(port)==0xFFFF)
       {
         asm {
           mov dx,3bch;
           mov al,11111111b;
           out dx,al;
         }
     }
  }
  if (joy==2)
 {

// MANEJO DEL BRAZO

     if(inport(port)==0xFFE0)
     {

     // Abrir el Brazo
        asm {
          mov dx,3bch;
             mov al,11111011b;
          out dx,al;
          }
      }

 if(inport(port)==0xFFC0)
 {

    //Cerrar el Brazo
          asm {
          mov dx,3bch;
             mov al,11110111b;
             out dx,al;
    }
 }

 if(inport(port)==0xFFD0)
       {
       asm {
         mov dx,3bch;
```

```
          mov al,11111111b;
          out dx,al;
      }
 }

   if(inport(port)==0xFFFF)
       {
         asm {
           mov dx,3bch;
           mov al,11111111b;
           out dx,al;
         }
       }
  }


   if (joy==3)
 {

// MANEJO DEL CUERPO

    if(inport(port)==0xFFE0)
    {

    // Rotacion Derecha
       asm {
         mov dx,3bch;
            mov al,11011111b;
         out dx,al;
         }
     }

 if(inport(port)==0xFFC0)
 {
    // Rotacion Izquierda
    asm {
    mov dx,3bch;
       mov al,11101111b;
       out dx,al;
    }
 }

 if(inport(port)==0xFFD0)
       {
         asm {
           mov dx,3bch;
           mov al,11111111b;
           out dx,al;
         }
       }

   if(inport(port)==0xFFFF)
```

```
        {
          asm {
            mov dx,3bch;
            mov al,11111111b;
            out dx,al;
          }
        }
    }



}

// PROGRAMA PRINCIPAL


void TGruaApp::InitMainWindow()
{
     MainWindow = new TMyWindow(NULL,"Interfaz para
Accionar Grua Prototipo" );
}



int PASCAL WinMain(HINSTANCE Inst, HINSTANCE PrevInst,
LPSTR CmdLine
      , int CmdShow)
{
    joy=1;
    timeover=bandera=TRUE;
    TGruaApp GruaApp("Interfaz para Accionar Grua
Prototipo", Inst
                     , PrevInst, CmdLine, CmdShow);
    GruaApp.Run();
    return GruaApp.Status;
}
```

```
//  Recursos de la Interfaz


MENU_GRUA MENU
BEGIN
     POPUP "&Manejo con Mouse"
     BEGIN
          MENUITEM "SUBIR", 101
          MENUITEM SEPARATOR
          MENUITEM "BAJAR", 102
          MENUITEM SEPARATOR
          MENUITEM "ABRIR", 103
          MENUITEM SEPARATOR
          MENUITEM "CERRAR", 104
          MENUITEM SEPARATOR
          MENUITEM "DERECHA", 105
          MENUITEM SEPARATOR
          MENUITEM "IZQUIERD", 106
          MENUITEM SEPARATOR
     END

     POPUP "&Detener"
     BEGIN
          MENUITEM "DETENER", 107
     END

     POPUP "Manejo con &Joystick"
     BEGIN
          MENUITEM "GANCHO", 108
          MENUITEM SEPARATOR
          MENUITEM "BRAZO", 109
          MENUITEM SEPARATOR
          MENUITEM "CUERPO", 110
     END

     POPUP "&Ayuda"
     BEGIN
          MENUITEM "&Ayuda Interfaz", IDHELP
     END

     POPUP "&Salir"
     BEGIN
          MENUITEM "&Salir Interfaz", 111
     END

END

SUBIR BITMAP "subir.bmp"
BAJAR BITMAP "bajar.bmp"
DERECHA BITMAP "rotacion.bmp"
IZQUIERD BITMAP "izquierd.bmp"
ABRIR BITMAP "abrir.bmp"
```

```
CERRAR BITMAP "cerrar.bmp"
GANCHO BITMAP "gancho.bmp"
BRAZO BITMAP "brazo.bmp"
CUERPO BITMAP "cuerpo.bmp"
DETENER BITMAP "detener.bmp"
GRUA BITMAP "grua.bmp"

ICON_1 ICON "grua.ico"
```

```
// Opción de Ayudad de la Interfaz


#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <owl.h>
#include <static.h>
#include <edit.h>
#include <listbox.h>
#include <button.h>
#include "helpwind.h"

THelpWindow::THelpWindow(PTWindowsObject AParent) :
   TWindow(AParent, "Sistema de Ayuda")
{
  DisableAutoCreate();
  Attr.Style |= WS_POPUPWINDOW | WS_CAPTION;
  Attr.X = 100;
  Attr.Y = 100;
  Attr.W = 420;
  Attr.H = 300;
  ListBox = new TListBox(this, ID_LISTBOX, 20, 20, 180,
80);
  new TButton(this, ID_BUTTON1, "Ayuda",320, 20, 60,
30, TRUE);
  new TButton(this, ID_BUTTON2, "Cancelar",320, 70, 60,
30, FALSE);
  Edit = new TEdit(this, ID_EDIT, "", 20, 180, 380, 90,
40, TRUE);
  new TStatic(this, -1, "Informacion:", 20, 160, 160,
20, 0);
}

void THelpWindow::SetupWindow()
{
  TWindow::SetupWindow();
  ListBox->AddString("Interfaz Digital");
  ListBox->AddString("Conexiones");
  ListBox->AddString("Manejo con Mouse");
  ListBox->AddString("Manejo con Joystick");
  ListBox->SetSelIndex(0);
};

void THelpWindow::HandleListBoxMsg(RTMessage Msg)
{
  char SelString[25];

  if ( Msg.LP.Hi == LBN_DBLCLK )
  {
```

```
    ListBox->GetSelString(SelString,
sizeof(SelString));
    FillEdit(SelString);
  }
}

void THelpWindow::HandleButton1Msg(RTMessage)
{
  char SelString[25];

  ListBox->GetSelString(SelString, sizeof(SelString));
  FillEdit(SelString);
}

void THelpWindow::HandleButton2Msg(RTMessage)
{
  CloseWindow();
}

void THelpWindow::FillEdit(Pchar SelString)
{
  Pchar HelpStr;

  if ( strcmp(SelString, "Interfaz Digital") == 0 )
  {  HelpStr =
       "La interfaz digital permite controlar los \r\n"
       "movimientos de la grúa prototipo.\r\n"
       "A través del puerto serial del computador se
envía\r\n"
       "la configuración correspondiente al puerto
de\r\n"
       "control de entrada de la interfaz digital. De
esta\r\n"
       "manera se realizan los diferentes movimientos
de la grúa.";
  };
  if ( strcmp(SelString, "Conexiones") == 0 )
  {  HelpStr =
       "Debe verificar que las conexiones físicas
estén\r\n"
       "correctas.\r\n"
       "Estas conexiones son:\r\n"
       "1.- El cable del puerto paralelo del
computador\r\n"
       "    debe conectarse al puerto de control de
\r\n"
       "    entrada de la interfaz digital.\r\n"
       "2.- El puerto de control de salida de la
interfaz\r\n"
       "    digital debe conectarse al panel de
control\r\n"
       "    de los motores de la grúa.\r\n"
```

```
        "3.- El joystick se conecta al puerto de
juegos\r\n"
        "    del computador.\r\n"
        "4.- El ratón se conecta al puerto serial o
al\r\n"
        "    puerto para ratón del computador";
  }
  if ( strcmp(SelString, "Manejo con Mouse") == 0 )
  {  HelpStr =
        "Al seleccionar manejo con mouse, deberá
escoger\r\n"
        "el movimiento de la grúa que desea
realizar.\r\n"
        "Para detener el movimiento, seleccione\r\n"
        "Detener en el menú principal y escoja la
opción\r\n"
        "STOP.";
  };
  if ( strcmp(SelString, "Manejo con Joystick") == 0 )
  {  HelpStr =
        "Para controlar la grúa con el joystick,\r\n"
        "primero seleccine el dispositivo que desea\r\n"
        "controlar: Gancho, Brazo o Cuerpo. \r\n"
        "Si su selección fue Gancho:\r\n"
        "- Presione el botón rojo para subir el
gancho\r\n"
        "- Presione el botón negro y el rojo para \r\n"
        "  bajar el gancho\r\n"
        "- Presione el botón negro para detener el
movimiento\r\n"
        "Si su selección fue Brazo:\r\n"
        "- Presione el botón rojo para abrir el
brazo\r\n"
        "- Presione el botón negro y el rojo para \r\n"
        "  cerrar el brazo\r\n"
        "- Presione el botón negro para detener el
movimiento\r\n"
        "Si su selección fue Cuerpo:\r\n"
        "- Presione el botón rojo para rotar a la
derecha\r\n"
        "- Presione el botón negro y el rojo para \r\n"
        "  rotar a la izquierda\r\n"
        "- Presione el botón negro para detener el
movimiento\r\n"
        "";
  };
  Edit->SetText(HelpStr);
}
```

```
// Manejo del Joystick, rutina de prueba
// con C++


#include <stdio.h>
#include <dos.h>
#include <conio.h>

int main(void)
{
  unsigned char result;
  int port = 0x201;
  clrscr();
  for(int i=1;i<10;i++)
  {
    result=inportb(port);
    gotoxy(20,20);
    printf("Byte leido del puerto %d =
0x%X\n",port,result);
    sleep(2);
  };
  return 0;
}
```

```
// Manejo del Joystick, rutina de prueba
// con C++


#include <stdio.h>
#include <dos.h>
#include <conio.h>

int main(void)
{
  unsigned result;
  unsigned port = 0x201;
  clrscr();
  for(int i=1;i<10;i++)
  {
    result=inpw(port);
    gotoxy(20,20);
    printf("Palabra leida del puerto %d =
0x%X\n",port,result);
    sleep(2);
  };
  return 0;
}
```

```
// Manejo del Joystick, rutina de prueba
// con lenguaje ensamblador

#include <dos.h>
#include <stdio.h>

main(void)
{
for(int i=0;i<20;i++)
{

asm{
    mov dx,0201h;
    in ax,dx;
    mov cx,ax;
    add cl,60;
    mov dl,cl;
    mov ah,02;
    INT 21H;
    add ch,60;
    mov dl,ch;
    mov ah,02;
    INT 21H
    }
 sleep(2);
 };
 return 0;
}
```

BIBLIOGRAFIA

1. Peter Coad, Edward Yourdon, "Object-Oriented Analysis", 1.990.
2. Al Stevens, "Teach Yourself C++", 1.990.
3. Bruce Eckel, "Aplique C++", 1.990.
4. Manuales de Borland C++ para Windows y DOS, versión 3.1, 1.993.
5. Revistas Mi Computer, 1.990.