UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POSTGRADO

"SIMULADOR PARA FRESADORA DE CONTROL NUMERICO COMPUTARIZADO F3-CNC"

POR:

ING. FERNANDO MONTEMAYOR IBARRA

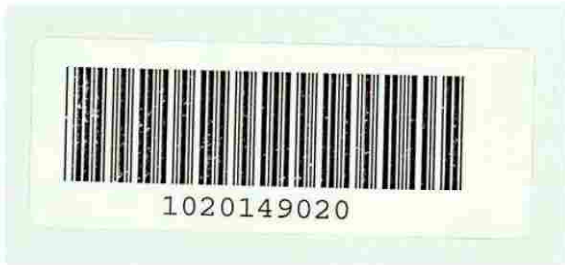
T E S I S

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
DE LA INGENIERIA EN MANUFACTURA
CON ESPECIALIDAD EN AUTOMATIZACION

SAN NICOLAS DE LOS GARZA, N. L., JUNIO DE 2002

2002

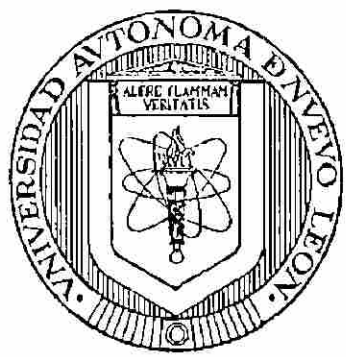
"SIMULADOR PARA FRESADORA DE CONTROL NUMERICO COMPUTARIZADO F3-CNC"



UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO



"SIMULADOR PARA FRESADORA DE CONTROL NUMÉRICO COMPUTARIZADO F3-CNC"

POR

ING. FERNANDO MONTEMAYOR IBARRA

TESIS

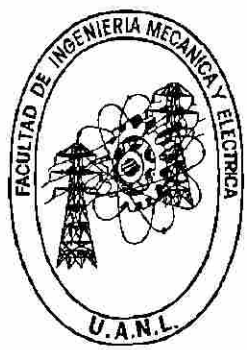
EN OPCIÓN AL GRADO DE MAESTRO EN CIENCIAS DE LA INGENIERÍA EN MANUFACTURA CON ESPECIALIDAD EN AUTOMATIZACIÓN

SAN NICOLÁS DE LOS GARZA, N.L. JUNIO DEL 2002

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO



"SIMULADOR PARA FRESADORA DE CONTROL NUMÉRICO COMPUTARIZADO F3-CNC "

POR

ING. FERNANDO MONTEMAYOR IBARRA

TESIS

EN OPCIÓN AL GRADO DE MAESTRO EN CIENCIAS DE LA INGENIERÍA EN MANUFACTURA CON ESPECIALIDAD EN AUTOMATIZACIÓN

SAN NICOLÁS DE LOS GARZA, N.L. JUNIO DEL 2002

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO

Los miembros del comité de tesis recomendamos que la tesis SIMULADOR PARA FRESADORA DE CONTROL NUMÉRICO COMPUTARIZADO F3-CNC, realizada por el ingeniero Fernando Montemayor Ibarra, matrícula 681283 sea aceptada para su defensa como opción al grado de Maestro en Ciencias de la Ingeniería en Manufactura con especialidad en Automatización.

El Comité de Tesis

M.C. Roberto Mireles Palomares
Asesor

Dra. Patricia del Carmen Zambrano Robledo
Coasesor

M.C. David Oliva Álvarez
Coasesor

Vo. Bo.

Dr. Guadalupe Alán Castillo Rodríguez
División de Estudios de Posgrado

PRÓLOGO

Este trabajo representa el esfuerzo por proveerle a los alumnos que se preparan en el uso de máquinas de CNC (Control Numérico Computarizado), de una herramienta útil para el aprendizaje. Permitiéndoles conocer las trayectorias de las herramientas desde una vista tridimensional.

La implementación de este simulador quedará bajo la autorización de los maestros correspondientes, permitiendo un ahorro considerable en la adquisición de equipo de enseñanza. Aún dista de las características de simuladores profesionales, pero es ya un intento por desarrollar un simulador acorde a las máquinas de CNC con que cuenta la Universidad Autónoma de Nuevo León.

ÍNDICE

CAPÍTULO		PÁGINA
	Síntesis	1
1.	**Introducción**	2
1.1.	Descripción del problema	2
1.2.	Objetivo de la tesis	3
1.3.	Hipótesis	3
1.4.	Límites del estudio	3
1.5.	Justificación del trabajo	3
1.6.	Metodología	4
1.7.	Revisión bibliográfica	5
2.	**Características de la fresadora F3-CNC**	7
2.1.	Descripción de las partes de la máquina	7
2.2.	Características técnicas de la máquina	9
2.3.	Características del control	10
2.4.	Dimensiones para valores de entrada	11
2.5.	Sistema de coordenadas	11
2.6.	La estructura de un programa	12
2.7.	Programación	13
2.8.	El estado inicial del control EMCOTRONIC M1	15
2.9.	Distribución de códigos en grupos	15
3.	**Descripción de los Códigos**	18
3.1.	Códigos sin sintaxis	18
3.2.	Códigos con sintaxis	25

4. Codificación 35

4.1. Introducción 35

4.2. Builder C++ 5.0 35

4.3. OpenGL 36

5. Uso del simulador 40

5.1. Introducción 40

5.2. Interfase del usuario 40

5.3. Códigos aceptados por el simulador 47

6. Ejemplos de simulación 48

6.1. Ejemplos 48

7. Conclusiones y recomendaciones 54

7.1. Conclusiones 54

7.2. Recomendaciones 55

Bibliografía 56

Lista de Tablas 57

Lista de Figuras 58

Apéndice A Listado de los programas 60

Glosario 101

Autobiografía 102

SÍNTESIS

La presente tesis incluye las características técnicas de la máquina fresadora, como lo son los datos de las dimensiones, valores máximos permitidos, los códigos de programación y la sintaxis de aquellos que lo requieran.

Puesto que algunos códigos requieren sintaxis y otros no, fueron separados para una mejor comprensión. Además, de los códigos que permiten el movimiento de la herramienta se incluyó la trayectoria que describe el centro de la herramienta.

Se explica el uso del simulador, desde los botones en la barra de herramientas hasta las ventanas de diálogo que aparecen, requiriendo determinados valores.

Los ejemplos incluidos muestran la similitud entre el recorrido de la herramienta en el simulador y el recorrido que tendría la herramienta en la máquina fresadora.

Se incluyó el listado completo del simulador, realizado en lenguaje C, para ambiente Windows.

1 INTRODUCCIÓN

1.1 DESCRIPCIÓN DEL PROBLEMA

En el laboratorio de máquinas y herramientas II de la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León se realizan prácticas en una fresadora de control numérico, para realizar dichas prácticas sólo se cuenta con cuatro tableros para simulaciones.

Los tableros son para preparar al alumno antes de que llegue a utilizar la máquina, el problema es que sólo son cuatro tableros para brigadas de hasta 12 alumnos, por lo cual éstos no se preparan adecuadamente, ya que por cada tablero hay sólo un alumno que puede utilizarlo.

Otro problema que se ha presentado es que los tableros empiezan a presentar fallas electrónicas debido a que el equipo ya tiene muchos años de uso, y sustituirlos es demasiado costoso.

1.2 OBJETIVO DE LA TESIS

El objetivo es realizar un programa para simular el recorrido de la herramienta partiendo de una codificación para la máquina fresadora F3-CNC, que sea capaz de ser utilizado en cualquier computadora con ambiente Windows, sin necesidad de algún aditamento especial.

Además que el maestro o instructor de laboratorio tendrá una herramienta más para aprovechar en la enseñanza.

1.3 HIPÓTESIS

Se parte del supuesto que para un mejor aprovechamiento de las máquinas y mejor entrenamiento para los alumnos, es necesario un simulador para control numérico computarizado.

1.4 LÍMITES DEL ESTUDIO

El programa sólo simulará el recorrido de la herramienta con la codificación propia de la fresadora de control numérico computarizado marca EMCO, modelo F3-CNC con un tablero de control EMCOTRONIC M1. Por lo tanto, no se incluyen en el programa del simulador la totalidad de los códigos de la máquina fresadora.

1.5 JUSTIFICACIÓN DEL TRABAJO DE TESIS

El desarrollo de este simulador beneficiará a los alumnos que cursen el laboratorio de Máquinas Herramientas II, incluyendo también la clase, permitiéndoles trabajar y entrenarse en una computadora antes de utilizar la fresadora de control numérico computarizado. Ya que de una manera virtual el alumno podrá ver las trayectorias de la

herramienta antes de llevarlas a cabo físicamente, contribuyendo a evitar el desperdicio de recursos y posibles daños en la máquina fresadora. Asimismo contribuirá a un ahorro bastante considerable de recursos económicos, al no adquirir costosos equipos para reemplazar a los ya existentes ó programas de simulación que no se adaptan al tipo de programación de la máquina en cuestión.

1.6 METODOLOGÍA

Etapa 1 Obtener códigos de la máquina, manual y clase de Máquinas-Herramientas II.

Hacer la comparación de códigos , descartar y adecuar aquellos que lo requieran

Etapa 2 Prepara el formato (sintaxis) de cada código que lo requiera. Obtener el recorrido que realiza en la máquina, para que el simulador siga la misma trayectoria.

Etapa 3 Respecto a la trayectoria de cada código de la máquina fresadora, encontrar el equivalente en el lenguaje de programación C, utilizando la librería gráfica OpenGL.

Etapa 4 Realizar la parte del programa que leerá e interpretará el archivo que contendrá las instrucciones para la máquina fresadora.

Etapa 5 Preparar la pantalla de presentación del simulador.

Etapa 6 Realizar la parte del programa que graficará los códigos ya interpretados.

Etapa 7 Realizar pruebas de las trayectorias entre la máquina fresadora de control numérico y el simulador utilizando la misma codificación.

Etapa 8 Conclusiones

Etapa 9 Propuestas

1.7 REVISIÓN BIBLIOGRÁFICA

EMCO MAIER & CO, "Programming Instruction EMCOTRONIC M1", 1986

Este es el manual de la fresadora, y sirvió para conocer todos los detalles de la máquina, desde sus características técnicas, hasta la programación de la misma. Describe todos los códigos, desde su sintaxis hasta el movimiento que tiene la herramienta. Incluye algunas ayudas útiles para programar.

Plastock Roy A., Gordon Kalley, "Gráficas por computadora", McGraw-Hill, noviembre de 1987

Este libro es una ayuda rápida para la introducción a las gráficas por computadora, sirvió para conocer los aspectos técnicos de los dibujos en 2D y 3D.
Aquí se explica las representaciones gráficas (vistas) que puede tener un objeto de tres dimensiones.

Charte Ojeda Francisco, "C++ Builder 4 – Guía Práctica para usuarios", Anaya Multimedia, 1999

Este libro se utilizó como la introducción sobre la programación en Builder. En el se aprende a utilizar el entorno gráfico para la generación de interfaces gráficas en Windows.

Charte Ojeda Francisco, "Programación con C++ Builder 5", Anaya Multimedia, 2000

Este libro se utilizó para desarrollar aplicaciones más avanzadas de la interfaz gráfica, utilizando ejemplos ahí descritos para la programación del simulador. Como lo son las tablas de cadenas, los iconos, y el uso de lectura y escritura de archivos de texto.

Jimenez Ruiz Manuel, "Desarrollo de herramientas multimedia para el apoyo del aprendizaje de la librería gráfica OpenGL", mayo del 2000

Esta información se utilizó para realizar la parte gráfica del simulador, tomando los ejemplos descritos para la programación de la librería gráfica.

2 CARACTERÍSTICAS DE LA FRESADORA F3-CNC

2.1 DESCRIPCIÓN DE LAS PARTES DE LA MÁQUINA

A continuación se describirán las características técnicas de la fresadora de control numérico computarizado F3-CNC, con ello se podrá conocer cuales son las bondades, limitaciones y rangos de valores en los que puede realizar los trabajos [3].

Se listan las partes más importantes de la máquina en la figura 2.1, aunque no en el orden de importancia.

1.- Puerta de protección contra virutas

2.- Cabezal de fresar

3.- Husillo

4.- Mesa de trabajo

5.- Manguera de refrigerante

6.- Marcas de referencia

7.- Unidad de control EMCOTRONIC M1

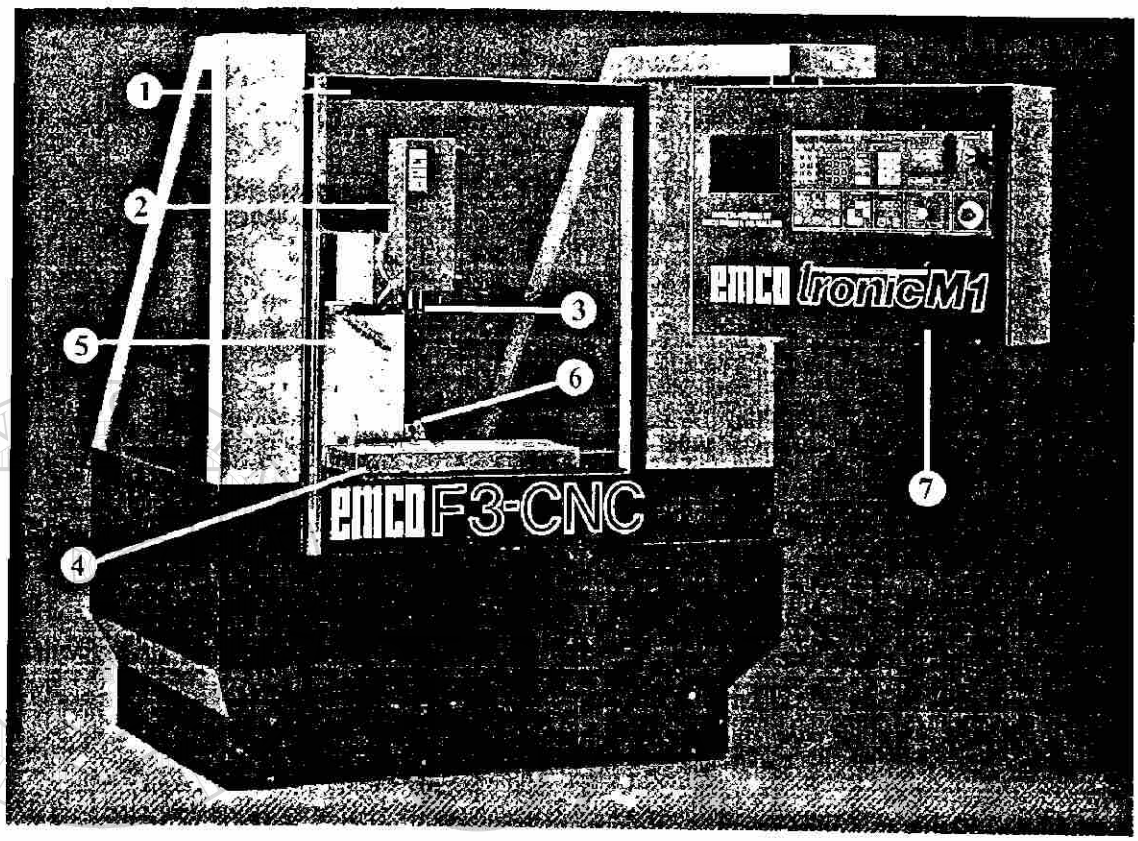


Figura 2.1 Partes de la máquina fresadora

2.2 CARACTERÍSTICAS TÉCNICAS DE LA MÁQUINA

La tabla 2.1 muestra las capacidades de la máquina, tanto en dimensiones para los recorridos de la herramienta, así como las velocidades máximas de la misma. Se incluye también las dimensiones de la mesa.

Área de trabajo		
Recorrido longitudinal (X)	[mm]	300
Recorrido transversal (Y)	[mm]	200
Recorrido vertical (Z)	[mm]	350
Superficie de sujeción vertical:	[mm]	600 x 180
2 ranuras en T	[mm]	12 x 21
Distancia entre ranuras	[mm]	90
Distancia del husillo de fresar a mesa de fresar	[mm]	35 - 385
Velocidades de husillo:	[rpm]	150 - 3000
Accionamiento		
Motor de corriente continua, rendimiento 100% de tiempo de conexión	[kW]	2,5
Accionamiento del avance:		
Velocidad de marcha acelerada	[mm/min]	3000
Avance de trabajo	[mm/min]	1 - 3000
Resolución	[mm]	0.0025
Superficie de instalación		
(Largo x Ancho x Altura)	[mm]	1700 x 1700 x 1800
Peso		
Peso	[kg]	aprox. 900

Tabla 2.1 Características de la máquina

2.3 CARACTERÍSTICAS DEL CONTROL

La tabla 2.2 muestra la lista de los códigos que el control es capaz de procesar. En capítulos posteriores se explicará la sintaxis de cada uno de ellos.

Control EMCOTRONIC M1			
Precisión de entrada		[mm]	0.001 (0.0001")
Variación sobre el avance		[%]	0 - 140
Variación sobre la velocidad del husillo		[%]	50 - 120
Margen de interpolación		[mm]	+ - 9999.999
Memoria para corrección de herramientas		[herramientas]	10
Formato de programa			
Estructura según DIN 66025			
Entrada de punto decimal			
O	Número de programa (00 - 99)		
N	Número de registro (0000 - 9999)		
G	Funciones de recorrido (00 - 99)		
	G00 = Movimiento rápido		
	G01 = Interpolación lineal		
	G02 = Interpolación circular a favor de las manecillas del reloj		
	G03 = Interpolación circular en contra de las manecillas del reloj		
	G04 = Retardo		
	G17 - G22 = Conmutación de ejes		
	G40 - G42 = Compensación de radio de herramienta		
	G53 - G59 = Registro de desplazamiento de posición		
	G70 = Programación en pulgadas		
	G71 = Programación métrica		
	G81 = Ciclo de taladrado		
	G82 = Ciclo de taladrado con retardo		
	G83 = Ciclo de taladrado profundo con retroceso		
	G84 = Ciclo de machueleado		
	G86 = Ciclo de taladrado profundo con rompevirutas		
	G87 = Ciclo de cajas rectangulares		
	G92 = N.d.r. máx. (fijar registro)		
	G94 = Velocidad de avance en mm/min.		
	G95 = Avance en µm /rev.		
	G98 = Regreso al plano de inicio		
	G99 = Regreso al plano de retroceso		
X, Y, Z	Coordenadas absolutas		
U, V, W	Coordenadas relativas		
P0...P7	Parámetros auxiliares		

Tabla 2.2 Características del control

D0...D7	Sobremedidas, profundidad de acercamiento, etc. en 1/1000 mm
F	Avance en mm/min., m/rev.
S	Velocidad del husillo
T	Llamada de herramienta, elección de corrección de herramienta (de cuatro dígitos)
L	Número de subprograma/Repeticiones (de cuatro dígitos)
M	Funciones (00 - 99)
	M00 = Parada intermedia programable
	M03 = Husillo en marcha a derecha
	M04 = Husillo en marcha a izquierda
	M05 = Parada de husillo
	M08 = Refrigerante encendido
	M09 = Refrigerante apagado
	M17 = Fin de subprograma
	M30 = Fin de programa con retorno al comienzo del programa
	M38 = Parada de precisión encendida
	M39 = Parada de precisión apagada

Tabla 2.2 Características del control (continuación)

2.4 DIMENSIONES PARA VALORES DE ENTRADA

La tabla 2.3 muestra los valores numéricos permitidos por el control.

Direcciones	Métrico	
	Valores	Unidades
X, Y, Z, U, W, I, J, K	0 a ±8000.000	[mm]
Avance por minuto	0 a 2200	[mm/min]
Avance por revolución	0 a 2000	[µm/rev] 1/1000 mm/rev
Radio máximo en arcos	65.000	[mm]
G, M, O	0 a 99	
N, L, T	0 a 9999	
S	Cuatro dígitos	

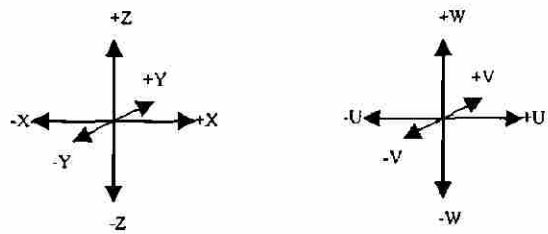
Tabla 2.3 Valores de entrada

2.5 SISTEMA DE COORDENADAS

La máquina es capaz de desplazarse en tres ejes simultáneamente. Una vez que se establece el origen, el sistema de ejes coordenados en forma absoluta queda determinado

como se muestra en la figura 2.2a y en la figura 2.2b se muestra para coordenadas incrementales o relativas.

Las coordenadas absolutas están referidas a un sólo punto fijo, llamado origen. Mientras que en las coordenadas relativas cada coordenada localizada se convierte en el origen para la próxima coordenada a localizar.



a) b)

Figura 2.2 Sistema de coordenadas

2.6 LA ESTRUCTURA DE UN PROGRAMA

Un programa de CNC contiene todas las instrucciones e informaciones que se necesitan para maquinar una pieza de trabajo.

A cada programa se le asigna un número, que se encuentra comprendido de 00 a 99 y se le antepone la letra "O".

Las líneas de un programa son llamadas Bloques.

Cada Bloque está compuesto de Palabras.

Cada Palabra está compuesta de una letra seguida de un valor numérico.

Éstas letras son conocidas como las Direcciones.

A continuación se enlistan las direcciones que utiliza este control, su significado y entre paréntesis se indica el rango de valores:

O número de programa principal (00 a 99)

L número de subrutina (cuatro dígitos)

N número de bloque (0000 a 9999)

G	función de recorrido (00 a 99)
M	función miscelánea (00 a 99)

X, Y, Z	coordenadas absolutas
U, V, W	coordenadas incrementales
I, J, K	parámetros de arco

P0 . . . P7	parámetros auxiliares
D0 . . . D7	parámetros auxiliares

F	avance
S	velocidad del husillo (cuatro dígitos)
T	llamada de herramienta (cuatro dígitos)

Los valores para las direcciones X, Y, Z, U, V, W, P0, P1, P3, P4, I, J ,K se tienen que programar con punto decimal. Sin el punto decimal, los valores se asumen como μm (micras).

Este control permite la programación modal, significa que los valores de las direcciones quedan activos hasta que no se dé un nuevo valor para la misma. Esto permite una simplificación en la programación.

2.7 PROGRAMACIÓN

Los programas constan de tres partes claramente visibles:

1.- Encabezado (Se definen valores iniciales)

2.- Cuerpo del programa (Instrucciones para el movimiento de la herramienta, generalmente comienza con G00)

3.- Fin de programa (Se desactivan los valores iniciales)

No existen regulaciones generales para el modo de programación. Cada programador confeccionará el programa a su propio parecer, de una manera sencilla y clara.
Sin embargo, se pueden cumplir ciertas directrices en cuanto a encabezado y fin de programa.

El encabezado deberá incluir:

1.- El sistema de unidades
2.- Unidades del avance
3.- Llamada a la herramienta y su compensación
4.- Velocidad de giro del husillo
5.- Sentido de giro del husillo
6.- Paro exacto conectado o desconectado
7.- Activación del cero de pieza

El fin de programa deberá incluir:

1.- La descompensación de la herramienta
2.- La desactivación del cero de pieza
3.- El desplazamiento de la herramienta a la posición de casa
4.- La instrucción para fin de programa

NOTA:

Aunque no en este orden, pero si debe incluirse esta información en el programa.

2.8 EL ESTADO INICIAL DEL CONTROL EMCOTRONIC M1

El estado inicial es determinado por el fabricante del control. Y la razón es una operación práctica y segura en la utilización de la máquina.

Los siguientes códigos se activan desde el momento que se enciende la máquina y no hay necesidad de programarlos.

Códigos G

G40 Cancelación de corrección del radio de herramienta
G71 Medidas en milímetros
G53 Cancelación de origen 1 y 2
G56 Cancelación de origen 3, 4 y 5
G94 Velocidad de avance en mm/min ó 1/100 in/min
G98 Regreso al plano de inicio
G17 Selección del plano X-Y

Códigos M

M05 Husillo principal desconectado
M09 Refrigerante desconectado
M39 Cancelación de modo de parada exacta

2.9 DISTRIBUCIÓN DE CÓDIGOS EN GRUPOS

Las siguientes dos tablas muestran la distribución de todos los códigos empleados por la máquina. Es posible programar varios códigos G y/o M en un mismo bloque, la única condición es que pertenezcan a distintos grupos, en caso de que no lo sean, sólo se

ejecutará el último de ellos. Esto aplica tanto para códigos G como para códigos M. En la tabla 2.4 se listan los códigos G y en la tabla 2.5 los códigos M agrupados.

Lista de códigos G		
Grupo 0	**G00**	Movimiento rápido
	G01	Interpolación lineal
	G02	Interpolación circular a favor de las manecillas del reloj
	G03	Interpolación circular en contra de las manecillas del reloj
	G04	Retardo en tiempo
	G81	Ciclo de taladrado
	G82	Ciclo de taladrado con retardo
	G83	Ciclo de taladrado profundo con retroceso
	G84	Ciclo de machueleado
	G86	Ciclo de taladrado profundo con rompeviruta
	G87	Ciclo de cajas rectangulares
Grupo 2	**G94**	Velocidad de avance en mm/min.
	G95	Avance en µm/rev.
Grupo 3	**G53**	Cancelación de origen 1 y 2
	G54	Activar origen 1
	G55	Activar origen 2
Grupo 4	**G92**	Fijar registro
Grupo 5	**G56**	Cancelación de origen 3, 4 y 5
	G57	Activar origen 3
	G58	Activar origen 4
	G59	Activar origen 5
Grupo 6	**G25**	Llamada a subrutina
	G27	Salto incondicional
Grupo 7	**G70**	Medidas en pulgadas
	G71	Medidas en milímetros
Grupo 8	**G40**	Cancelación de la corrección del radio de la herramienta
	G41	Corrección del radio de la herramienta a la izquierda
	G42	Corrección del radio de la herramienta a la derecha
Grupo 9	**G17**	Cambio de ejes
	G18	Cambio de ejes
	G19	Cambio de ejes
	G20	Cambio de ejes
	G21	Cambio de ejes
	G22	Cambio de ejes
Grupo 11	**G98**	Regreso al plano de inicio
	G99	Regreso al plano de retroceso

Tabla 2.4 Lista de códigos G agrupados

NOTA:

El grupo 1 y 10 no existen en esta máquina.

Lista de códigos M		
Grupo 0	**M03**	Giro del husillo a favor del reloj
	M04	Giro del husillo en contra del reloj
	M05	Paro del husillo
Grupo 1	**M38**	Parada de precisión encendida
	M39	Parada de precisión apagada
Grupo 2	**M00**	Parada programada
	M17	Fin de subrutina
	M30	Fin de programa y regreso al inicio
Grupo 3	**M08**	Refrigerante encendido
	M09	Refrigerante apagado

Tabla 2.5 Lista de códigos M agrupados

3 DESCRIPCIÓN DE LOS CÓDIGOS

3.1 CÓDIGOS SIN SINTAXIS

Los códigos aquí listados no necesitan ningún otro valor o parámetro para poder funcionar en la máquina. La sola programación provoca su funcionamiento.

M00 PARADA PROGRAMADA

Este comando produce una parada en la ejecución de un programa de piezas.

Se desconectan el husillo de fresado, los avances y el refrigerante.

La puerta de protección contra virutas puede abrirse sin que se active la alarma.

La ejecución del programa puede continuar con CYCLE START (Arranque de ciclo).

Se emplea cuando dentro del proceso de maquinado se deban realizar medidas, cambio de herramienta, cambio de sujeción de la pieza, etc.

M03 GIRO DEL HUSILLO A FAVOR DEL RELOJ

El husillo se activa siempre que se haya programado ciertas revoluciones, la puerta de protección contra virutas esté cerrada y haya una pieza de trabajo debidamente amarrada.

M03 ha de utilizarse para todas las herramientas de corte a la derecha. Y el sentido se determina en una vista superior de la máquina. Véase la figura 3.1.

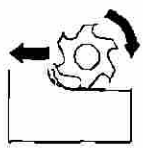


Figura 3.1 M03 Giro del husillo a favor del reloj

M04 GIRO DEL HUSILLO EN CONTRA DEL RELOJ

Las mismas condiciones que en M03.

M04 debe emplearse para todas las herramientas de corte a la izquierda. La figura 3.2 muestra la vista superior de la herramienta girando en contra de las manecillas del reloj.

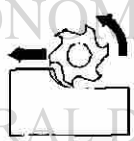


Figura 3.2 M04 Giro del husillo en contra del reloj

M05 PARO DEL HUSILLO

Se frena eléctricamente el motor principal.

Al final del programa el husillo se apaga al programar el código M30.

M08 REFRIGERANTE ENCENDIDO

La bomba de refrigerante se conecta.

El flujo del refrigerante se controla por medio de la válvula de flujo instalada en la manguera del refrigerante.

M09 REFRIGERANTE APAGADO

La bomba de refrigerante se desconecta.

El flujo del refrigerante se detiene.

M17 FIN DE SUBRUTINA

M17 se escribe en el último bloque de una subrutina. Regresa el control al programa anterior o a un nivel superior.

Puede estar sólo en este bloque o con otras funciones.

La llamada a una subrutina y M17 no pueden estar en el mismo bloque.

M30 FIN DE PROGRAMA PRINCIPAL

Determina el fin del programa. Todos los programas deben terminar con este código. Regresa al inicio del programa, esperando una nueva ejecución.

Adicionalmente apaga el refrigerante y el husillo.

M38 PARADA DE PRECISIÓN ENCENDIDA

Se utiliza cuando se quiera una transición más pronunciada. Los movimientos axiales se paran por completo en la coordenada programada y tan sólo entonces es cuando se realiza el movimiento siguiente.

Una parada completa de los carros en la coordenada cuesta tiempo.

M39 PARADA DE PRECISIÓN APAGADA

El control EMCOTRONIC M1 está diseñado de manera tal que se acelera el eje Y antes de alcanzar el punto objetivo en dirección X. Se alcanza así un movimiento uniforme en caso de transiciones de contornos.

G17 CAMBIO AL PLANO X-Y

Cambia el sistema de ejes al plano X-Y

G18 CAMBIO AL PLANO Y-Z

Cambia el sistema de ejes al plano Y-Z

G19 CAMBIO AL PLANO X-Z

Cambia el sistema de ejes al plano X-Z

G40 CANCELACIÓN DE LA CORRECCIÓN DEL RADIO DE LA HERRAMIENTA

La descompensación del radio de la herramienta se realiza con G40.

Desactiva G41 o G42, según se encuentren activados.

G40 puede programarse en el mismo bloque que G00 o G01, o en el bloque anterior.

G40 se programa generalmente en el bloque de retirada al punto de cambio de herramienta.

G41 CORRECCIÓN DEL RADIO DE LA HERRAMIENTA A LA IZQUIERDA

Con la corrección del radio de la herramienta aciva, el control calcula automáticamente una trayectoria paralela al contorno, por lo que el radio de la herramienta se compensa.

Para poder calcular una trayectoria que tome en cuenta el radio de la herramienta, tiene que haber una compensación de herramienta activada en la tabla de compensaciones de la máquina. La figura 3.3 muestra el recorrido que seguiría el centro de la herramienta al estar compensado o descompensado el radio de la herramienta.

Figura 3.3 Compensación y descompensación del radio de la herramienta

Si la herramienta (vista en la direción de maquinado) está a la izquierda del contorno a mecanizar, hay que seleccionar G41, para una mejor apreciación observe la figura 3.4.

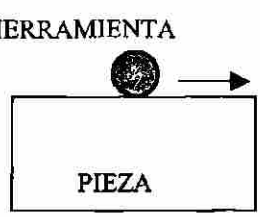


Figura 3.4 Corrección a la izquierda

G42 CORRECCIÓN DEL RADIO DE LA HERRAMIENTA A LA DERECHA

Si la herramienta (vista en la direción de mecanizado) está a la derecha del contorno a maquinar, hay que seleccionar la corrección del radio con G42, y esto puede apreciarse en la figura 3.5.

Figura 3.5 Corrección a la derecha

G70 MEDIDAS EN PULGADAS

Todos los valores en X, Y, Z, F, U, V, W y todos los parámetros D y P expresan sus unidades en pulgadas.

G71 MEDIDAS EN MILÍMETROS

Todos los valores en X, Y, Z, F, U, V, W y todos los parámetros D y P expresan sus unidades en milímetros.

G94 VELOCIDAD DE AVANCE EN MM/MIN

Las unidades del avance (F) se expresan en mm/min.

G95 AVANCE EN μM/REV

Las unidades del avance (F) se expresan en 1/1000 mm/rev.

G98 REGRESO AL PLANO DE INICIO

G99 REGRESO AL PLANO DE RETROCESO

Estos códigos se utilizan en combinación con alguno de los ciclos como G81, G82, G83, G86 y G87, y su finalidad es controlar la salida de la herramienta una vez que ha realizado el maquinado. La herramienta puede subir hasta el plano de inicio o sólo al plano de retroceso, según el código programado.

El fabricante de la máquina pone como condición que se programe G98 o G99 antes del parámetro P3 o P4. La figura 3.6 muestra la altura que alcanza la herramienta una vez finalizado el ciclo.

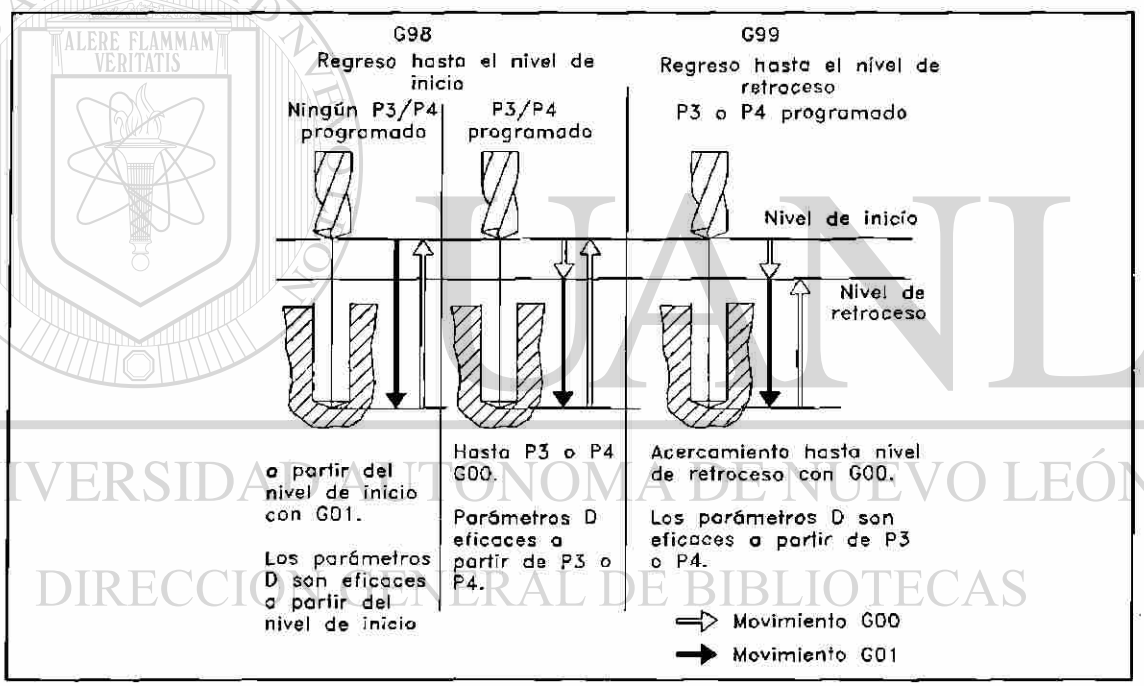


Figura 3.6 G98/G99, regreso de la herramienta

3.2 CÓDIGOS CON SINTAXIS

Ahora los códigos listados aquí, necesitan de ciertos valores o parámetros para poder funcionar, y la mayoría utiliza coordenadas para desplazar la herramienta de un lugar a otro. Los códigos que incorporan una F en su sintaxis, se utilizan para corte de material por medio de la herramienta.

G00 MOVIMIENTO RÁPIDO

G00 X/U... Y/V… Z/W…

X,Y,Z	Coordenadas del punto final
U,V,W	Distancia del punto inicial al punto final

La instrucción tiene como finalidad desplazar la herramienta en línea recta desde el lugar en que se encuentra hasta los valores indicados en las coordenadas (figura 3.7), para acercar o alejar la herramienta del material, pero con la condición de que nunca haga contacto con el material.

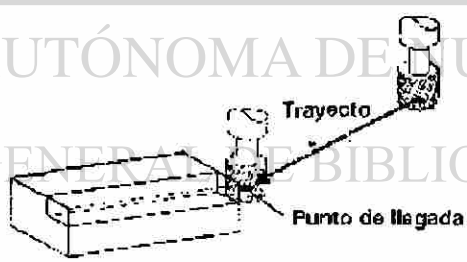


Figura 3.7 G00, código de desplazamiento rápido

NOTA: El centro de la herramienta siempre alcanzará el valor de las coordenadas.

G01 INTERPOLACIÓN LINEAL

G01 X/U... Y/V... Z/W... F...

X,Y,Z	Coordenadas del punto final
U,V,W	Distancia del punto inicial al punto final
F	Avance

Su finalidad es cortar el material en línea recta (figura 3.8), desde donde se encuentre la herramienta hasta las coordenadas indicadas.

La velocidad con que se corta el material es controlado por F.

También se usa esta instrucción para hacer contacto entre la herramienta y el material.

G01 X0.0 Y0.0 Z0.0 F100

Figura 3.8 G01, código de interpolación lineal

G02 INTERPOLACIÓN CIRCULAR A FAVOR DE LAS MANECILLAS DEL RELOJ

G02 X/U... Y/V... Z/W... I... J... K... F...

X,Y,Z	Coordenadas del punto final
U,V,W	Distancia del punto inicial al punto final
I, J, K	Distancia de la herramienta al centro del arco

F Avance

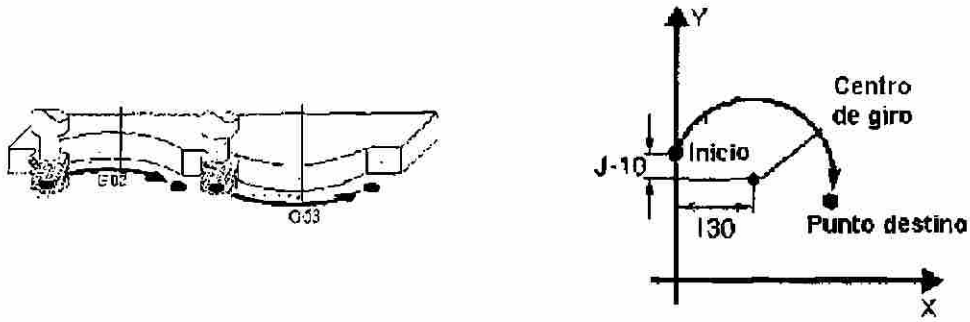


Figura 3.9 G02/G03, códigos de interpolación circular

Su finalidad es de producir cortes con una trayectoria circular en dirección de las manecillas del reloj, sólo se le indica el punto final, y las coordenadas del centro de la trayectoria que se quiere describir (figura 3.9). El radio es descompuesto en dos vectores conocidos como I, J y es la distancia del centro de la herramienta en el punto de inicio hasta el centro de la trayectoria circular.

G03 INTERPOLACIÓN CIRCULAR EN CONTRA DE LAS MANECILLAS DEL RELOJ

G03 X/U... Y/V... Z/W... I... J... K... F...

X,Y,Z	Coordenadas del punto final
U,V,W	Distancia del punto inicial al punto final
I, J, K	Distancia de la herramienta al centro del arco
F	Avance

Su fin es idéntico al de G02 sólo que la trayectoria sigue una dirección en contra de las manecillas del reloj (figura 3.9).

G04 RETARDO EN TIEMPO

G04 D4...

D4 Retardo en tiempo [1/10 seg.]

Permite un retardo en la ejecución del programa, los valores aceptables son de 1 a 10000 (1/10 – 1000 seg.). Y entra en función hasta el final del bloque en el que se programa.

G81 CICLO DE TALADRADO

G81 X/U... Y/V... Z/W... P3/P4... F...

X,Y	Coordenadas del centro del agujero
Z	Profundidad del agujero
P3/P4	Establece el nivel de retroceso
F	Avance

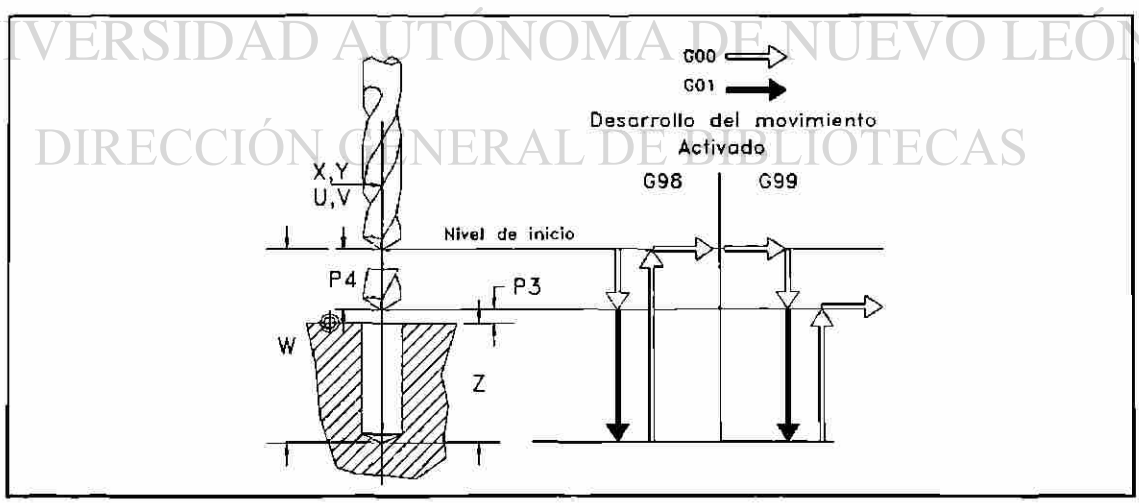


Figura 3.10 G81, ciclo de taladrado

Su finalidad es de realizar taladrados en el lugar indicado. Es posible controlar la salida de la herramienta por medio de los códigos G98 o G99 ya descritos en el punto anterior.

G82 CICLO DE TALADRADO CON RETARDO

G82 X/U... Y/V... Z/W... P3/P4... D4... F...

X,Y	Coordenadas del centro del agujero
Z	Profundidad del agujero
P3/P4	Establece el plano de retroceso
D4	Retardo en el fondo del agujero [1/10 seg.]
F	Avance

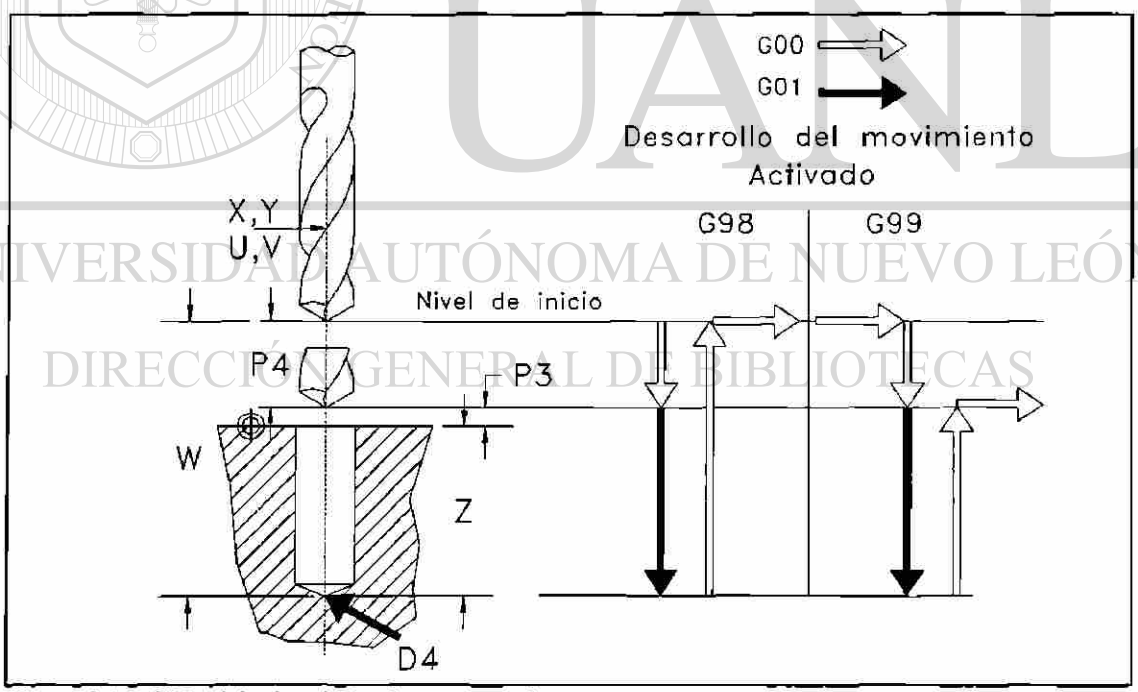


Figura 3.11 G82, ciclo de taladrado con retardo

Su finalidad es idéntica a G81, sólo se agrega un retardo en la ejecución del ciclo cuando la herramienta llega a la profundidad indicada.

G83 CICLO DE TALADRADO PROFUNDO CON RETROCESO

G83 X/U... Y/V... Z/W... P3/P4... D3... D5... D6... F...

X,Y	Coordenadas del centro del agujero
Z	Profundidad del agujero
P3/P4	Establece el plano de retroceso
D3	Profundidad del primer taladrado [1/1000 mm]
D5	Porcentaje de reducción [%]
D6	Profundidad mínima de taladrado [1/1000 mm]
F	Avance

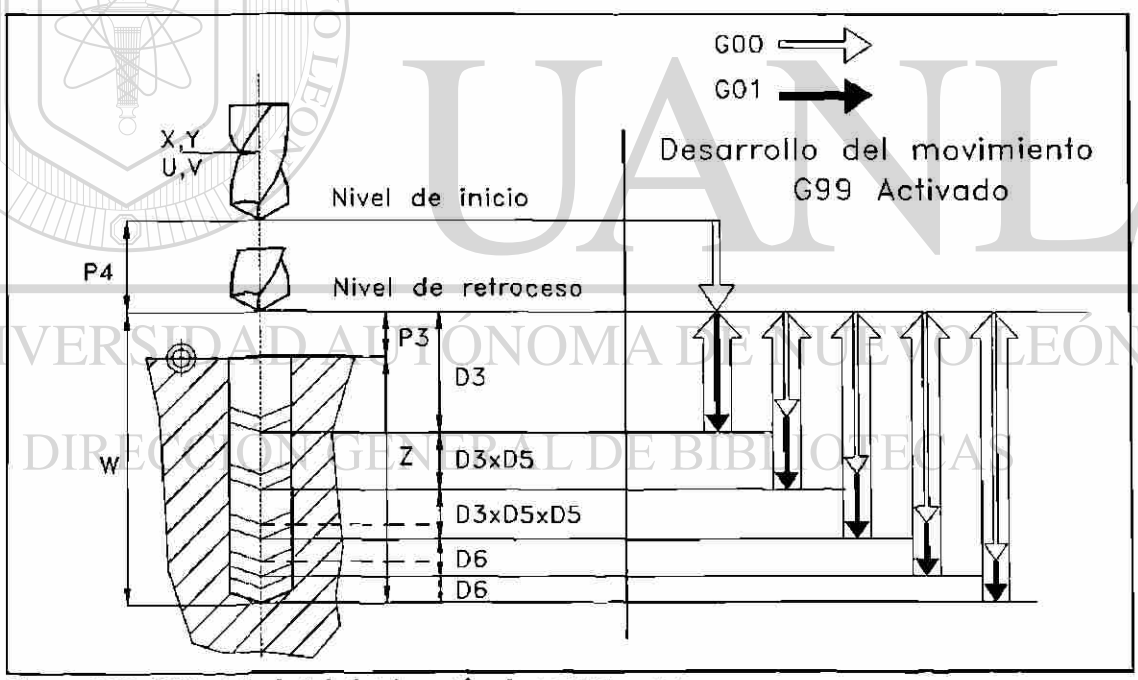


Figura 3.12 G83, ciclo de taladrado profundo con retroceso

El ciclo trabaja produciendo un taladrado, pero la herramienta sale completamente del material como se indica en la figura, de manera que el taladrado lo realiza por pasos, retirándose la herramienta completamente cada vez que esta corta material.

El uso de G83 tiene que ver con el taladrado de agujeros profundos. Se considera un agujero profundo cuando este tiene cinco o más veces el diámetro de la herramienta.

La utilidad de este ciclo es alargar la vida de la herramienta. Y se le debe emplear cuando se trabaja con materiales que producen una viruta muy pequeña. Esta es retirada cada vez que la herramienta sale del material para luego volverse a introducir y cortar más material, repitiéndose el ciclo hasta llegar a la profundidad indicada.

Este procedimiento se realiza de la siguiente manera:

La herramienta baja cortando hasta un profundidad indicada por D3, luego sale completamente del material, para volverse a introducir cortando hasta una profundidad de D3*D5, de manera que si D5 tiene algún valor, la profundidad será cada vez más pequeña debido a que D5 es un porcentaje de reducción. Cuando la profundidad es más pequeña que el valor asignado a D6, desde ese momento la herramienta tendra una profundidad constante, dada por D6.

G86 CICLO DE TALADRADO PROFUNDO CON ROMPEVIRUTA

G86 X/U... Y/V... Z/W... P3/P4... D3... D5... D6... F...

X,Y	Coordenadas del centro del agujero
Z	Profundidad del agujero
P3/P4	Establece el plano de retroceso
D3	Profundidad del primer taladrado [1/1000 mm]
D5	Porcentaje de reducción [%]
D6	Profundidad mínima de taladrado [1/1000 mm]
F	Avance

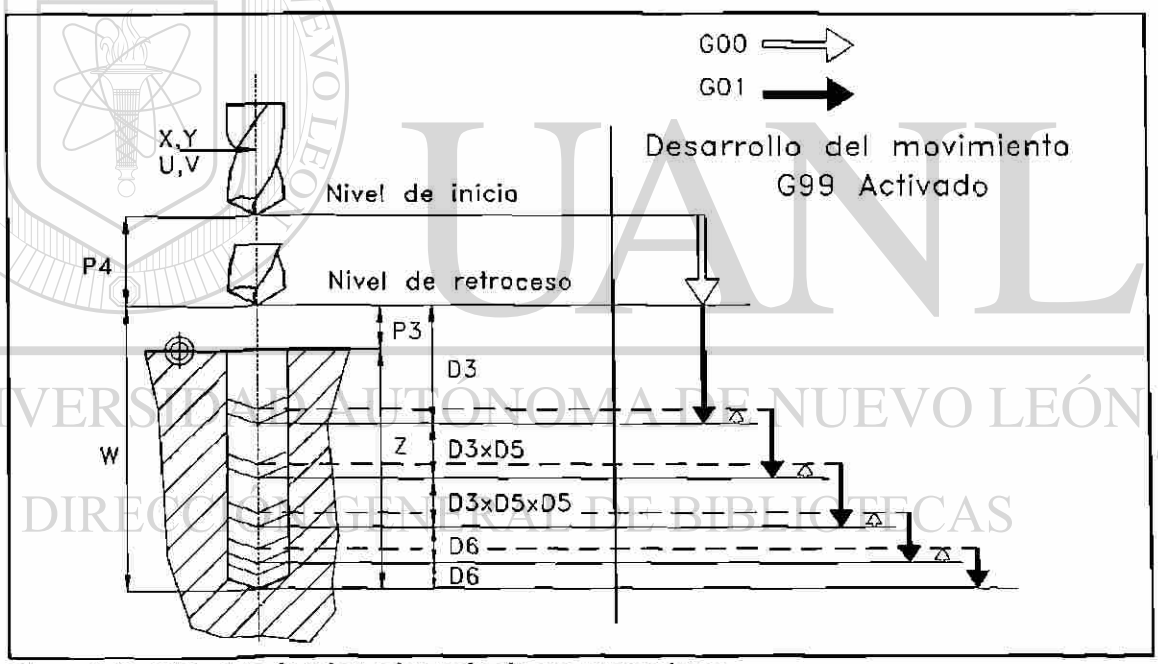


Figura 3.13 G86, ciclo de taladrado profundo con rompeviruta

El ciclo G86 (véase la figura 3.13) es similar a G83, sólo que en este la herramienta no sale completamente del material, pero si realiza un pequeño retroceso en el corte de manera que el material cortado no produzca un viruta continua.

Así que este ciclo sólo debe emplearse en materiales que al ser maquinados tienden a producir una viruta larga y continua, provocando que esta quede atorada en las estrías de la herramienta ocasionando su ruptura.

G87 CICLO DE CAJAS RECTANGULARES

G87 X/U... Y/V... Z/W... P3/P4... P0... P1... D3... D5... D7... F...

X,Y	Coordenadas del centro de la caja rectangular
Z	Profundidad de la caja
P3/P4	Establece el plano de retroceso
D3	Profundidad de cada corte [1/1000 mm]
D5	Sentido de corte
	D5=02 Fresado a favor de las manecillas del reloj
	D5=03 Fresado en contra de las manecillas del reloj
D7	Avance en profundidad
	D7=0 Avance rápido
	D7=1 Avance a la mitad del valor programado
F	Avance

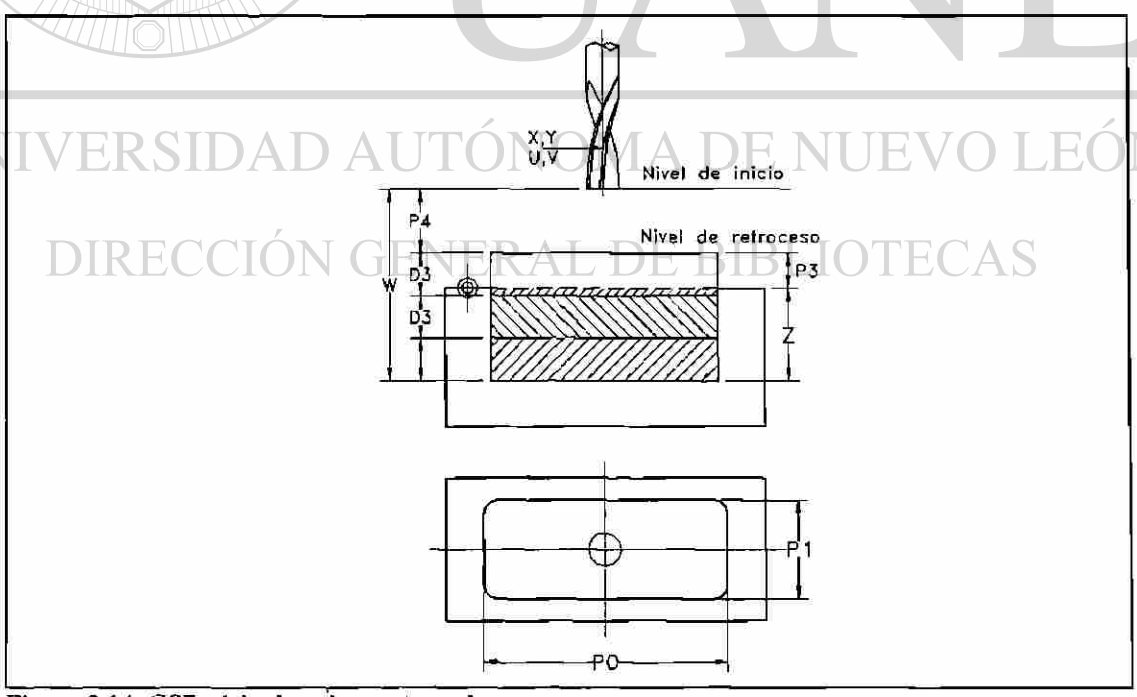


Figura 3.14 G87, ciclo de cajas rectangulares

Este ciclo se utiliza para realizar operaciones de careado y/o cajeado. (Figura 3.14).

La herramienta describe una trayectoria de manera que produce una caja rectangular con las dimensiones expresadas en P0 y P1. La herramienta parte desde el centro de la caja formando una espiral cuadrada hasta llegar a la orilla. El ciclo toma en cuenta el diámetro de la herramienta de manera que se produzca una caja realmente con las dimensiones expresadas. La herramienta puede producir esta espiral a favor o en contra de las manecillas del reloj según sea indicado por el parámetro D5.

La caja es hecha en capas, y cada capa tiene la misma profundidad. La profundidad máxima de cada capa es expresada por el parámetro D3.

4 CODIFICACIÓN

4.1 INTRODUCCIÓN

El presente capítulo tiene como objetivo presentar la manera en que se realizó el simulador.

Incluyendo las operaciones matemáticas para la realización de los cálculos, como lo son para la distancia recorrida por la herramienta y el tiempo empleado por la misma para realizar todo el recorrido propuesto en el programa.

En el apéndice puede observarse la codificación completa del simulador.

4.2 BUILDER C++ 5.0

Para el desarrollo de esta tesis se utilizó el programa Builder C++ de Borland, en su versión 5. El entorno de desarrollo de C++ Builder es simple, flexible y potente, contando con un gran número de componentes prefabricados que facilitan la creación de cualquier aplicación [8].

Utiliza como plataforma el Windows, por lo que los programas generados presentan una interfaz gráfica. Los programas se desarrollan en un entorno visual, simplificando la

programación de la interfaz como lo son los iconos, los botones, etc., propios del sistema operativo Windows.

4.3 OPENGL

OpenGL de Silicon Graphics, es una librería gráfica para el desarrollo de objetos en 3D. Está disponible para varias plataformas como Mac, Linux, Windows, y estaciones de trabajo de Silicon Graphics. Sólo necesita tres funciones como mínimo para operar, cada una de ellas con algunos comandos propios de OpenGL [4][5].

La primera de ellas se ejecuta para crear la ventana y cada vez que se cambia el tamaño de la ventana.

La segunda función se ejecuta al menos una vez, y se encarga de la generación del dibujo.

Y la tercera función es la principal de la aplicación, se encarga de generar todo el proceso, utilizando las dos funciones anteriores.

Se utilizó OpenGL por su facilidad de programación y su potencia para la generación de gráficos en 3D.

OpenGL sólo está compuesto de 110 comandos. Y la mayoría de estos comandos utilizan el prefijo gl.

En la tabla 4.1 se incluye sólo la lista de los comandos utilizados en el simulador.

COMANDO	DESCRIPCIÓN
glBegin	Inicio de la lista de vértices
glClearColor	Limpia el área del dibujo
glColor3f	Establece el color
glColor3ub	Establece el color en formato RGB
glDisable	Desactivar parámetros
GLdouble	Variable de coma flotante, 64 bits
glEnable	Activar parámetros
glEnd	Fin de la lista de vértices
glLineStipple	Establece el tipo de línea punteada
glLineWidth	Establece el ancho de las líneas
glLoadIdentity	Abre la matriz actual en la matriz modelador
glMatrixMode	Especifica la matriz del modelador
glNormal3f	Establece la normal a una superficie
glPopMatrix	Extrae la matriz de la pila
glPushMatrix	Guarda la matriz en la pila
glRotatef	Rota el dibujo
glScalef	Escala el dibujo
glTranslated	Traslada el dibujo
gluDeleteQuadric	Borra toda la información guardada de una cuádrica
gluDisk	Genera un disco
gluNewQuadric	Establece parámetros para una cuádrica
gluPartialDisk	Genera un disco parcial
gluPerspective	Proyección en perspectiva
glVertex3f	Establece un vértice
glViewport	Área visible de la representación del dibujo

Tabla 4.1 Lista de comandos GL utilizados en el simulador

Para generar las líneas rectas que indican la trayectoria de la herramienta se utilizó el siguiente formato:

```
glBegin (GL_LINES);
    glVertex3f (X1, Y1, Z1);
    glVertex3f (X2,Y2, Z2);
glEnd();
```

Donde las instrucciones "glBegin" y "glEnd" delimitan entre ellos el trazado de alguna figura geométrica. Esta figura geométrica es indicada entre paréntesis en la primer línea, y en este caso se trata de una línea.

La instrucción "glVertex3f" es para localizar vértices en el espacio, en este caso se utilizan dos instrucciones, puesto que para una línea es necesario indicar sus dos extremos.

Las coordenadas X_1, Y_1 y Z_1 corresponden al primer punto o extremo de la línea y las coordenadas X_2,Y_2 y Z_2 al segundo punto o extremo de la línea.

A cada código G se le asignó un color específico, para una mejor visualización en el recorrido de la herramienta. Véase la tabla 4.2.

La instrucción "glColor3ub" es responsable del color de las líneas, y para generar cada color se parte de tres componentes, como son: rojo, verde y azul, con valores comprendidos entre 0 y 255.

Código	Color	Valores
G00	Amarillo	255,255,0
G01	Verde	0,255,0
G81	Rojo	255,0,0
G82	Morado	255,0,255
G83	Blanco	250,250,250
G86	Gris	150,150,150
G87	Marrón	80,80,10

Tabla 4.2 Colores de las líneas de las trayectorias en códigos G

Para las trayectorias cuando la herramienta tiene la máxima velocidad de avance, como es el caso del código G00 o desplazamiento rápido, se utilizó una línea punteada. Las líneas continuas corresponden a movimientos con un avance programado, como lo son todos los códigos G que representan movimientos de corte en el material.

Para obtener las líneas punteadas se programaron las siguientes instrucciones:

```
glLineStipple(4,0xAAAA);
glEnable(GL_LINE_STIPPLE);
```

Donde la primera instrucción configura el espaciado entre los segmentos de la línea punteada, y la segunda instrucción la activa para todas las líneas ha ser dibujadas.

Para volver a dibujar líneas continuas, se desactiva la instrucción anterior con:

```
glDisable(GL_LINE_STIPPLE);
```

5 USO DEL SIMULADOR

5.1 INTRODUCCIÓN

El simulador proveerá de una mejor perspectiva del recorrido de la herramienta.

Aún cuando sólo se muestra la trayectoria del centro de la herramienta. El presente capítulo pretende dar a conocer el uso y la interpretación correcta del simulador, incluyendo sus limitaciones.

5.2 INTERFASE DEL USUARIO

El simulador fue desarrollado en una interfase amigable, del tipo Windows. La figura 5.1 muestra la pantalla principal (interfase gráfica) donde puede apreciarse la barra de menús, la barra de herramientas, el área gráfica, la barra vertical de información y la barra de estado.

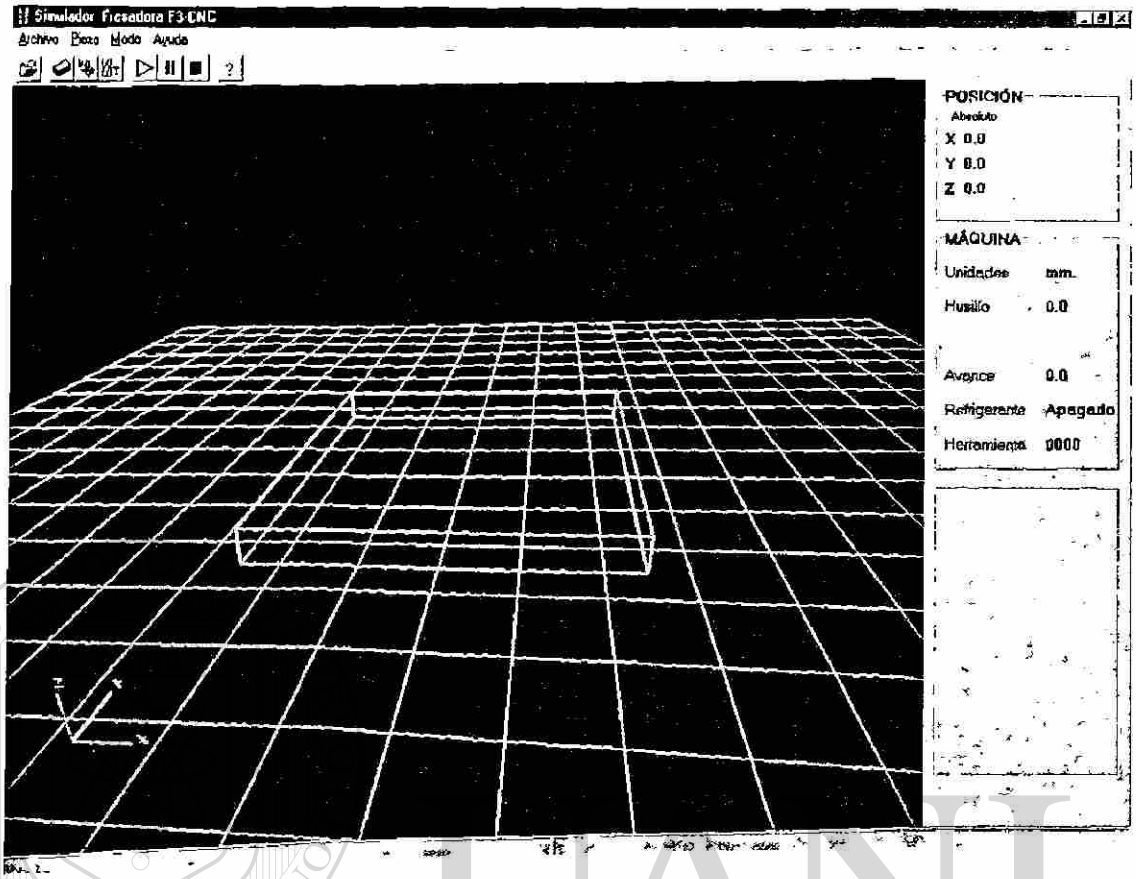


Figura 5.1 Interfase del usuario

La barra de menús

Está situada en la parte superior, ofrece una serie de menús desplegables que contiene la mayoría de los procesos para el uso del simulador.

La barra de herramientas

Está situada justo debajo de la barra de menús, e incluye una serie de botones con los procesos generales del simulador, como son abrir archivo, modificación del entorno de trabajo, modo de operación, etc.

El área gráfica

Ocupa la mayor parte de la pantalla y es donde se muestran los recorridos de la herramienta. Contiene la pieza y los ejes coordenados también son mostrados.

La barra vertical de información

Ocupa la parte derecha de la pantalla, y contiene la información acerca de los recorridos de la herramienta, unidades y velocidades.

La barra de estado

Muestra la codificación del programa, y sólo se muestra la línea que se encuentre en ejecución dentro del programa.

EL TECLADO

Se utiliza el teclado para introducir valores necesarios, además puede accesarse a las opciones de la barra de menú, como son:

ALT-A	Archivo
ALT-P	Pieza
ALT-M	Modo
ALT-U	Ayuda

EL RATÓN

El ratón o mouse es el dispositivo apuntador básico en Windows. En principio funciona según tres botones.

El botón izquierdo sirve para señalar o seleccionar alguna opción como lo son los botones, además manteniéndolo pulsado en el área gráfica permite rotar la figura.

El botón central manteniéndolo pulsado en el área gráfica permite acercar o alejar la figura.

El botón derecho manteniéndolo pulsado en el área gráfica permite mover la figura en el plano XY.

ABRIR ARCHIVOS

Para abrir algún archivo, es necesario oprimir el botón de la figura 5.2 o bien, oprimir ALT-A para que aparezca la ventana de diálogo de la figura 5.3

Figura 5.2 Botón para abrir archivo

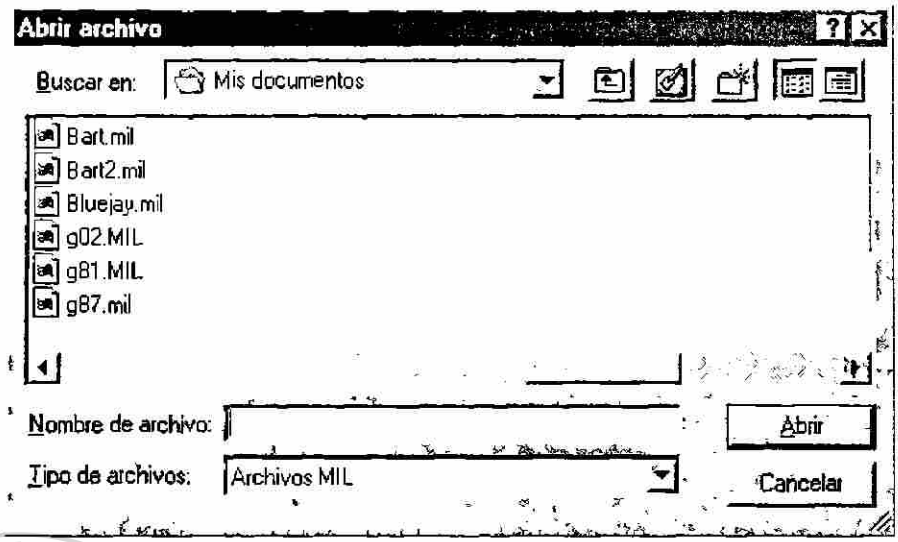


Figura 5.3 Ventana de diálogo par abrir archivo

El simulador sólo podrá abrir archivos que tengan la extensión .MIL
Estos archivos deberán tener un formato de texto, que contenga las instrucciones propias de la Fresadora F3.

DIMENSIONES DE LA PIEZA

Es posible modificar las dimensiones de la pieza para adecuarla a las dimensiones necesarias. Para realizar esto es necesario oprimir el botón que aparece en la figura 5.4 o bien, oprimir ALT-P.

Figura 5.4 Botón para modificar las dimensiones de la pieza

Hecho lo anterior, aparecerá la ventana de diálogo que se muestra en la figura 5.5

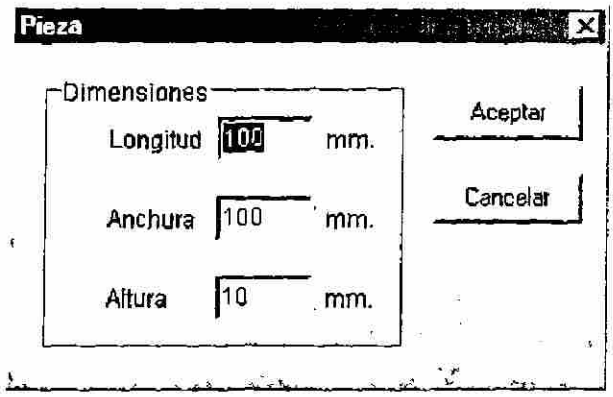


Figura 5.5 Ventana de diálogo para dimensiones de la pieza

Los valores para las dimensiones de la pieza deben ser expresados en milímetros. Al momento de realizar cualquier cambio en estas dimensiones, automáticamente se actualizará el dibujo de la pieza en el área de dibujo.

CERO DE PIEZA

El cero de la pieza define el origen para el sistema de coordenadas. Este origen se coloca en algún lugar de la pieza para facilitar la programación. Normalmente se encontrará situado en la esquina superior izquierda de la pieza, de manera que la misma quede en el primer cuadrante, por lo que los valores de X e Y serán normalmente positivos.

El simulador, así como la máquina, cuenta con cinco puntos para almacenar los valores de X, Y y Z que es la distancia desde el cero de la máquina hasta el cero de la pieza.

Para acceder a la tabla que contiene esta información es necesario oprimir el botón que aparece en la figura 5.6


Figura 5.6 Botón para establecer cero de pieza

Una vez hecho lo anterior aparecerá la ventana de diálogo que se muestra en la figura 5.7, donde en la primer casilla vertical se tiene el número de lista, y se encuentra numerada del uno al cinco, para cada número hay una X, Y y Z.

Los valores aquí contenidos representan la distancia entre el cero de máquina y el cero de pieza. Tanto el simulador como la máquina utilizan estos valores para trasladar el origen a los nuevos valores contenidos en la tabla.

Los valores introducidos se graban en un archivo llamado PSO.MIL, de manera que no necesitan teclearse cada vez que se utilice el simulador. Estos valores pueden ser enteros positivos o negativos.

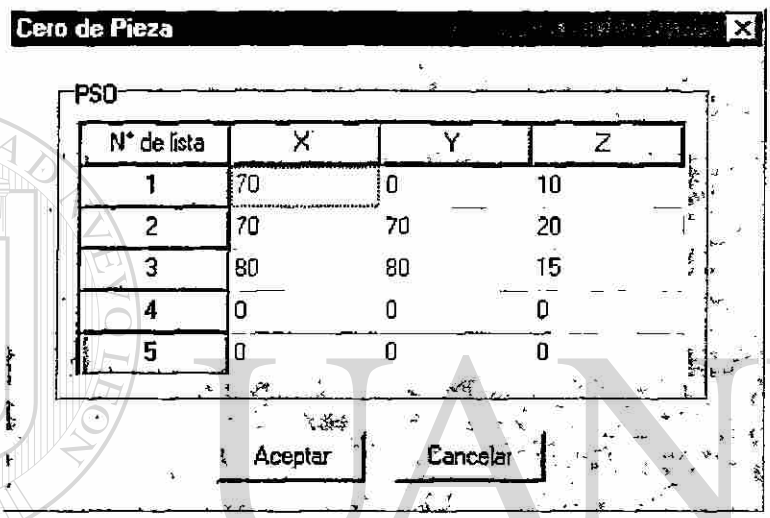


Figura 5.7 Ventana de diálogo para establecer el cero de pieza.

CERO DE HERRAMIENTA

En la máquina el cero de herramienta servirá para calibrar cada una de las herramientas utilizadas, de manera que la unidad de control tenga información como la longitud y radio de las herramientas. Dicha información se graba en una tabla en la memoria de la unidad de control. El simulador contiene una tabla similar, aunque solo registra los radios de las herramientas empleadas.

Es necesario para utilizar el ciclo G87, haber introducido el radio de la herramienta que se utilizará en dicho ciclo.

Para acceder a esta tabla en el simulador, hay que oprimir el botón que se muestra en la figura 5.8.


Figura 5.8 Botón para indicar los radios de las herramientas

Hecho lo anterior aparecerá la ventana de diálogo de la figura 5.9 donde para cada herramienta hay que indicar su radio.

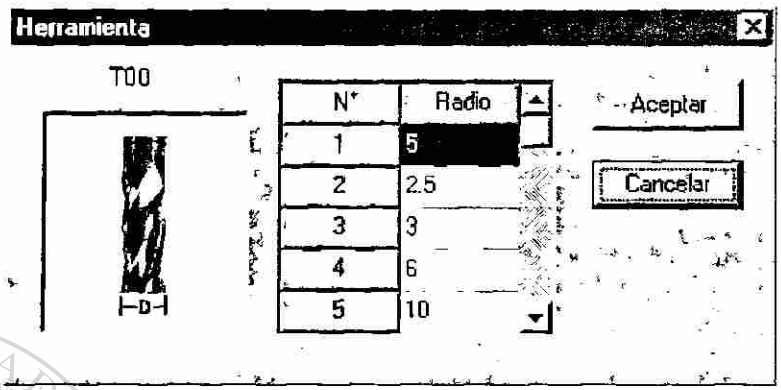


Figura 5.9 Ventana de diálogo para indicar el radio de las herramientas

El radio deberá introducirse en forma de milímetros; la tabla tiene capacidad para almacenar hasta nueve radios para herramientas. Toda la información de la tabla se graba en un archivo llamado TOOLS.MIL, de manera que si se utiliza la misma herramienta, no tenga que introducirse la información cada vez que se utiliza el simulador.

EJECUCIÓN DEL PROGRAMA

Para ejecutar el programa pueden utilizarse los botones de correr, pausar y detener.

La figura 5.10 muestra los tres botones respectivamente.

Figura 5.10 Botones para ejecución de los programas

Conforme el programa se ejecuta, en la barra de estado se muestra cada línea del programa que se está ejecutando en ese momento. En la barra de información se muestran los valores de las coordenadas, así como información de las unidades de los valores, velocidades y herramientas utilizadas.

5.3 CÓDIGOS ACEPTADOS POR EL SIMULADOR

A continuación se listan los códigos G para trayectorias de la herramienta que son aceptados por el simulador.

G00 X... Y... Z...

G01 X... Y... Z... F...

G81 X... Y... Z... P3=... F...

G82 X... Y... Z... P3=... F...

G83 X... Y... Z... P3=... D3=... D5=... D6=... F...

G86 X... Y... Z... P3=... D3=... D5=... D6=... F...

G87 X... Y... Z... P0=... P1=... P3=... D3=... D5=... F...

Cada una de las sintaxis de los códigos es idéntica en su significado a la presentada por el manual de la máquina como se presenta en el capítulo 3.

Algunos de los códigos listados anteriormente no incluyen la sintaxis completa como lo señala el manual de la máquina fresadora, debido a que el objetivo del simulador es describir la trayectoria de la herramienta. Por lo cual se omitieron algunos parámetros de la sintaxis de los códigos.

6 EJEMPLOS DE SIMULACIÓN

6.1 EJEMPLOS

A continuación se presentan algunos ejemplos realizados en el simulador; cada ejemplo contiene el dibujo de la pieza como se obtendría esta en la máquina fresadora, así como el dibujo de la trayectoria de la herramienta en el simulador.

Las líneas punteadas corresponden a movimientos rápidos y las líneas continuas corresponden a movimientos de corte. Las figuras de la izquierda, para cada ejemplo, pertenecen a pantallas grabadas del simulador.

Téngase en cuenta que las líneas trazadas, sólo indican el recorrido del centro de la herramienta.

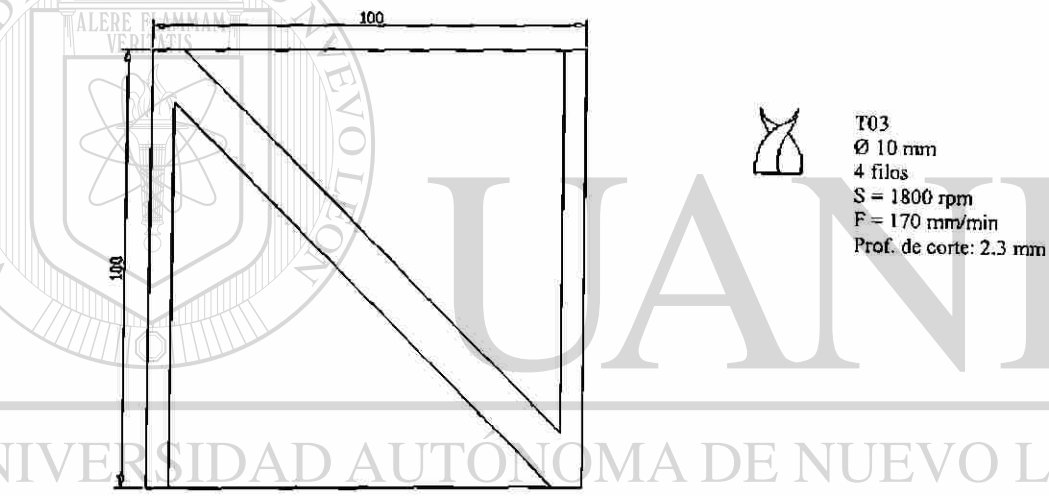
La figura de cada ejemplo, se encuentra dividida de la siguiente manera:
(a) vista superior de la pieza requerida, incluyendo información técnica,
(b) vista isométrica de la pieza requerida, generada en AutoCAD, y
(c) vista de la pieza y el recorrido del centro de la herramienta, obtenida del simulador de CNC.

EJEMPLO 1

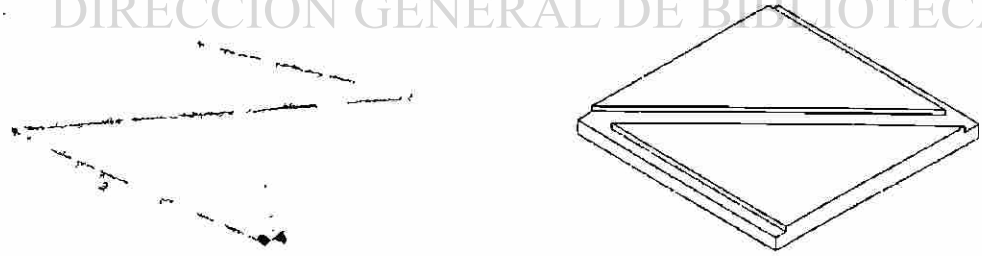
(Figura 6.1)

```
N010  G71  G53  G56
N020  T0303  G94  G59
N030  S1800  M03
N040  G00  X0.0  Y0.0  Z1.0
N050  G01  Z-2.3  F170
N060  Y100.0
N070  X100.0  Y0.0
N080  Y100.0
N090  G00  Z1.0
N100  T0300  G53  G56
N110  G00  X147.267  Y217.623  Z378.400
N120  M30
```

(a)



(b)



(c)

Figura 6.1 Ejemplo 1

EJEMPLO 2

(Figura 6.2)

```
N010 G71 G53 G56
N020 T0303 G94 G59
N030 S1800 M03
N040 G00 X45.0 Y0.0 Z1.0
N050 G01 Z-2.3 F170
N060 Y100.0
N070 X55.0
N080 Y0.0
N090 G00 Z1.0
N100 T0300 G53 G56
N110 G00 X147.267 Y217.622 Z378.4
N120 M30
```

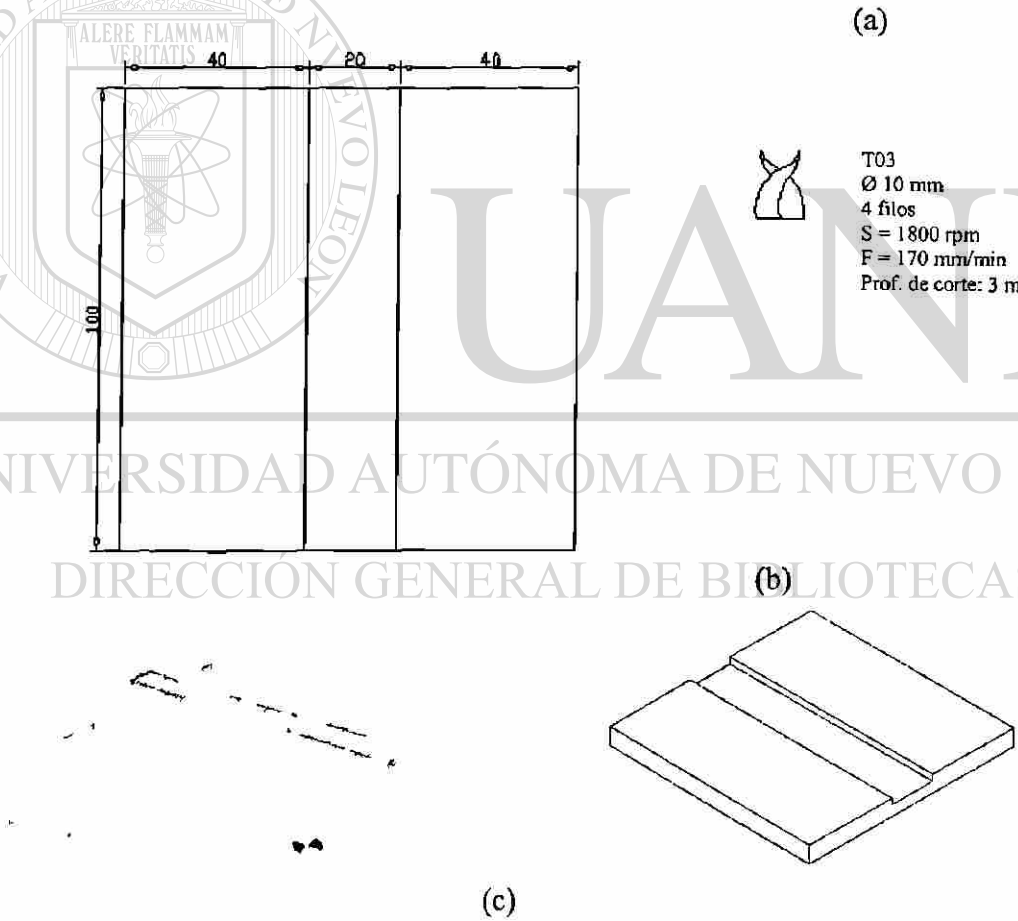


Figura 6.2 Ejemplo 2

EJEMPLO 3

(Figura 6.3)

```
N010  G71  G53  G94
N020  G00  X147.267  Y217.622  Z378.4
N030  T0505  G94  G54
N040  S1592  M03
N050  G00  X50.0  Y0.0  Z1.0
N060  G01  Z-10.0  F318
N070  Y100.0  Z0.0
N080  G00  Z1.0
N090  T0500  G53
N100  G00  X147.267  Y217.622  Z378.4
N110  M30
```

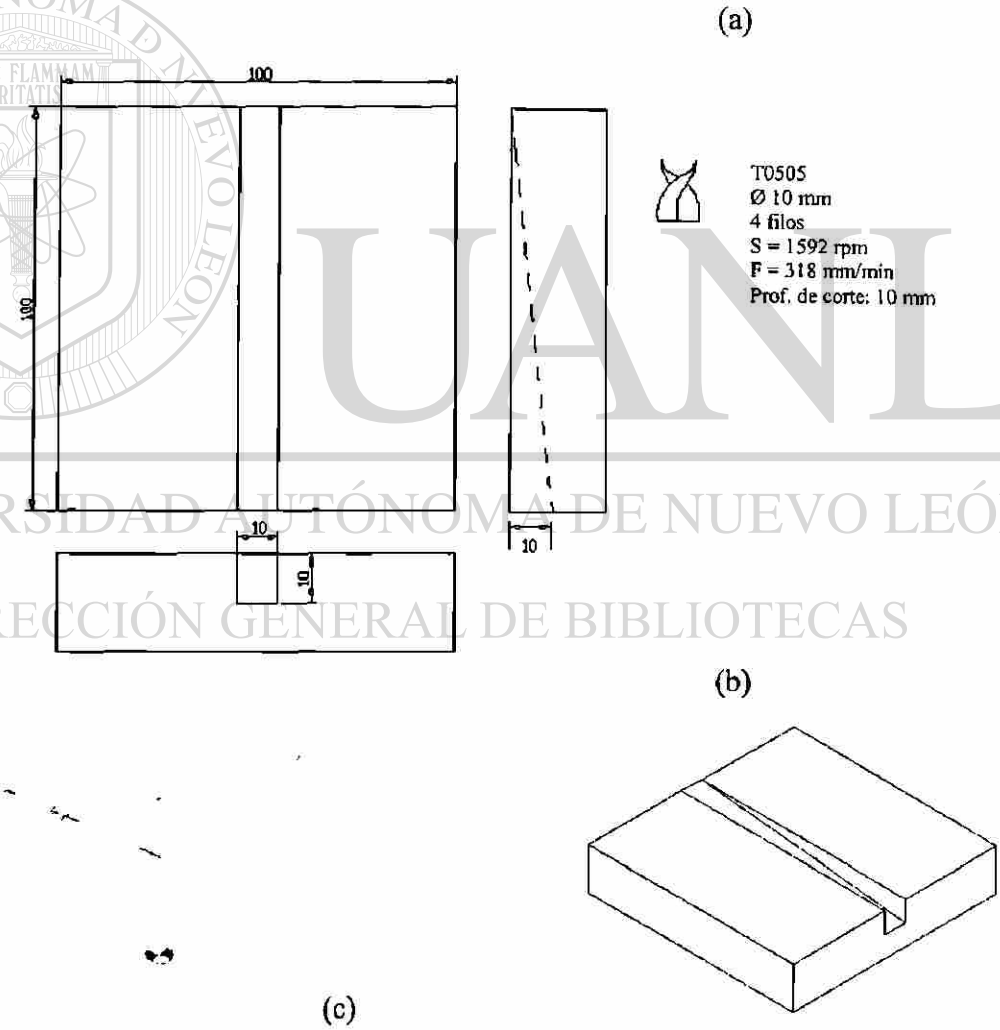


Figura 6.3 Ejemplo 3

EJEMPLO 4

(Figura 6.4)

```
N010  G71  G53  G56
N020  T0505  G94  G54
N030  S1592  M03
N040  G00  X5.0  Y100.0  Z1.0
N050  G01  Z0.0  F318
N060  Y0.0  Z-10.0
N070  X95.0
N080  Y100.0  Z0.0
N090  G00  Z1.0
N100  T0500  G53
N110  G00  X147.267  Y217.622  Z378.4
N120  M30
```

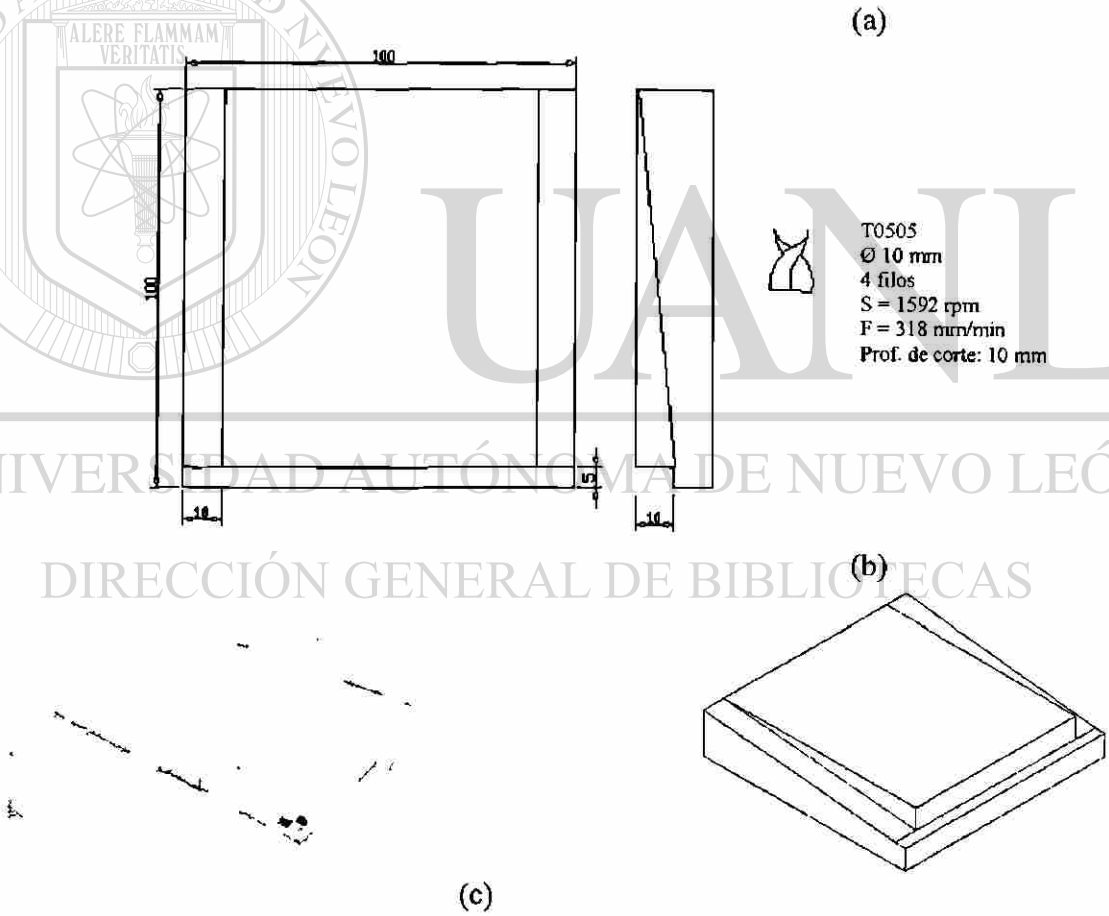


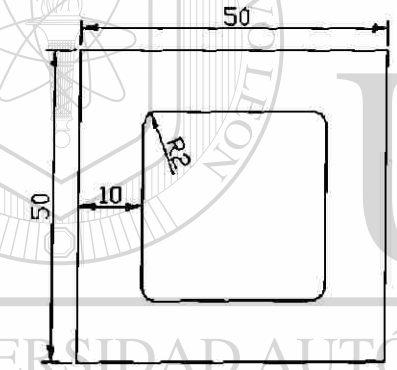
Figura 6.4 Ejemplo 4

EJEMPLO 5

(Figura 6.5)

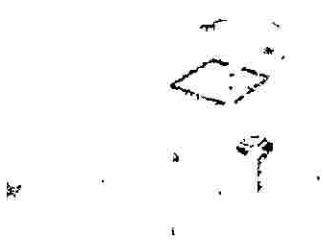
```
N010  G71  G53  G56
N020  T0101  G55  G94  G54
N030  S530  M03
N040  G00  X17.5  Y17.5  Z2.0
N050  G01  Z-5.0  F53
N060  Y32.5
N070  X32.5
N080  Y17.5
N090  X17.5
N100  G00  Z5.0
N110  T0100  G53
N120  G00  X147.267  Y217.622  Z378.4
N130  M30
```

(a)



T01
Ø 15 mm
4 filos
S = 530 rpm
F = 53 mm/min
Prof. de corte: 5 mm

(b)



(c)

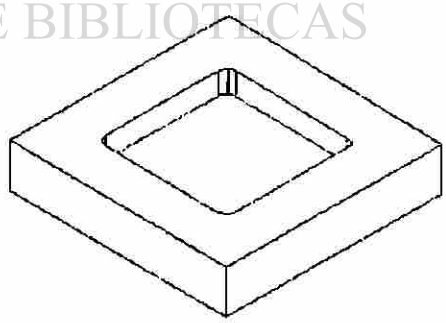


Figura 6.5 Ejemplo 5

7 CONCLUSIONES Y RECOMENDACIONES

7.1 CONCLUSIONES

Con esta introducción a la programación de gráficos en 3D aplicada a simuladores de CNC, se nos permite visualizar de una mejor manera el recorrido de la herramienta, permitiendo una mejor comprensión y aprendizaje de la programación de las máquinas de CNC.

El simulador no necesita muchos requerimientos para su utilización, por lo tanto el tiempo de aprendizaje para su utilización es muy corto. Aún cuando en algunos casos se requiere de un poco de imaginación, la visualización resulta atractiva para el estudiante, resultándole en un mejor aprovechamiento, ya que podrá utilizarlo para su clase y sus sesiones de laboratorio.

7.2 RECOMENDACIONES

El simulador no incorpora todos los códigos de programación de la máquina fresadora, unos por no ser relevantes en la trayectoria de la herramienta, y algunos otros, como G02/G03, por la carencia en el tiempo para desarrollarlos.

Este simulador es un primer intento, por lo que es susceptible a modificarse y corregir posibles errores que pudieran aparecer al tenerlo ya implementado en el laboratorio de Máquinas – Herramientas II.

Se propone como motivo para otro desarrollo, utilizar como base la programación de este simulador para aplicarlo a controles como SINUMERIK 810 que también se encuentra en las máquinas fresadoras del laboratorio.

BIBLIOGRAFÍA

[1] Charte Ojeda Francisco, "C++ Builder 4 – Guía Práctica para usuarios", Anaya Multimedia, 1999

[2] Charte Ojeda Francisco, "Programación con C++ Builder 5", Anaya Multimedia, 2000

[3] EMCO MAIER & CO, "Programming Instruction EMCOTRONIC M1", 1986

[4] Jimenez Ruiz Manuel, "Desarrollo de herramientas multimedia para el apoyo del aprendizaje de la librería gráfica OpenGL", http://odra.dcs.fi.uva.es/area2/opengl/index2.html, mayo del 2000

[5] OpenGL Programming Guide, Addison-Wesley, http://ask.ii.uib.no/ebt-bin/nph-dweb/dynaweb/SGI_Developer/OpenGL_PG/

[6] Plastock Roy A., Gordon Kalley, "Gráficas por computadora", McGraw-Hill, noviembre de 1987

[7] Schildt Herbert, "Turbo C/C++ - Manual de referencia", McGraw-Hill, 1992

[8] Software "Borland C++ Builder Standard Versión 5", Inprise Corporation, 2000

LISTA DE TABLAS

TABLAS	PÁGINA
Tabla 2.1 Características de la máquina	9
Tabla 2.2 Características del control	10
Tabla 2.3 Valores de entrada	11
Tabla 2.4 Lista de códigos G agrupados	16
Tabla 2.5 Lista de códigos M agrupados	17
Tabla 4.1 Lista de comandos GL utilizados en el simulador	37
Tabla 4.2 Colores de las líneas de las trayectorias en códigos G	39

LISTA DE FIGURAS

FIGURAS **PÁGINA**

Figura 2.1 Partes de la máquina fresadora 8
Figura 2.2 Sistema de coordenadas 12
Figura 3.1 M03 Giro del husillo a favor del reloj. 19
Figura 3.2 M04 Giro del husillo en contra del reloj 19
Figura 3.3 Compensación y descompensación del radio de la herramienta 22
Figura 3.4 Corrección a la izquierda 22
Figura 3.5 Corrección a la derecha 23
Figura 3.6 G98/G99, regreso de la herramienta 24
Figura 3.7 G00, código de desplazamiento rápido 25
Figura 3.8 G01, código de interpolación lineal 26
Figura 3.9 G02/G03, códigos de interpolación circular 27
Figura 3.10 G81, ciclo de taladrado 28
Figura 3.11 G82, ciclo de taladrado con retardo 29
Figura 3.12 G83, ciclo de taladrado profundo con retroceso 30
Figura 3.13 G86, ciclo de taladrado profundo con rompeviruta 32
Figura 3.14 G87, ciclo de cajas rectangulares 33
Figura 5.1 Interfase del usuario 41
Figura 5.2 Botón para abrir archivo 43
Figura 5.3 Ventana de diálogo par abrir archivo 43
Figura 5.4 Botón para modificar las dimensiones de la pieza 43
Figura 5.5 Ventana de diálogo para dimensiones de la pieza 44
Figura 5.6 Botón para establecer cero de pieza 44
Figura 5.7 Ventana de diálogo para establecer el cero de pieza 45
Figura 5.8 Botón para indicar los radios de las herramientas 45
Figura 5.9 Ventana de diálogo para indicar el radio de las herramientas 46

Figura 5.10 Botones para ejecución de los programas 46
Figura 6.1 Ejemplo 1 49
Figura 6.2 Ejemplo 2 50
Figura 6.3 Ejemplo 3 51
Figura 6.4 Ejemplo 4 52
Figura 6.5 Ejemplo 5 53

APÉNDICE A
LISTADO DE LOS PROGRAMAS

A continuación se listan todos los programas que conforman el simulador.

Las palabras en negrita corresponde al nombre del archivo.

DEMOPROJ.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("DemoProj.res");
USEFORM("Demo1.cpp", Form1);
USEFORM("Material.cpp", MaterialBox);
USEFORM("about.cpp", AboutBox);
USEFORM("Herramienta.cpp", ToolBox);
USEFORM("CeroW.cpp", CeroPiezaBox);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try
  {
    Application->Initialize();
    Application->CreateForm(__classid(TForm1), &Form1);
         Application->CreateForm(__classid(TMaterialBox), &MaterialBox);
         Application->CreateForm(__classid(TToolBox), &ToolBox);
         Application->CreateForm(__classid(TCeroPiezaBox), &CeroPiezaBox);
         Application->Run();
```

```
    }
    catch (Exception &exception)
    {
      Application->ShowException(&exception);
    }
    return 0;
}
//---------------------------------------------------------------------------
```

DEMO1.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Demo1.h"
#include "material.h"
#include "about.h"
#include "herramienta.h"
#include "cerow.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "OpenGLPanel"
#pragma resource "cursors.RES"
//#pragma link "Coorde"
//#pragma link "Unit2"
#pragma resource "*.dfm"
//#pragma resource "cursors.RES"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
  MouseDown = false;

  //ZoomZ=(float)TrackBar1->Position;
  ZoomZ=200.0;
  Escala[0] = 50.0;
  Escala[1] = 50.0;
  Escala[2] = 5.0;
  Desplaza[0]=0.0;
  Desplaza[1]=0.0;
  Ejecutar = 0;
  Feed =0;
  Ejecuta[0].XX = 147.1;
  Ejecuta[0].YY = 378.400;
  Ejecuta[0].ZZ = 117.1;
  Label7->Caption = "0.0";
  Label8->Caption = "0.0";
  Label9->Caption = "0.0";
  Label10->Caption = "0.0";
  Label11->Caption = "0.0";
  Label12->Caption = "0.0";
  Label22->Caption = "mm.";
```

```
  Label23->Caption = "0.0";
  Label25->Caption = "0.0";
  Label26->Caption = "Apagado";
  Label27->Caption = "0000";
}

//---------------------------------------------------------------------------

void __fastcall TForm1::OpenGLPanel1Init(TObject *Sender)
{
        glViewport(0,0,(GLsizei)OpenGLPanel1->Width,(GLsizei)OpenGLPanel1->Height);
        glMatrixMode(GL_PROJECTION);

        glLoadIdentity();
  if ( OpenGLPanel1->Height==0)

        gluPerspective(45, (GLdouble)OpenGLPanel1->Width, 1.0, 3000.0);
  else
    gluPerspective(60, (GLdouble)OpenGLPanel1->Width/
                (GLdouble)OpenGLPanel1->Height,1.0, 3000.0);
        glMatrixMode(GL_MODELVIEW);

        glLoadIdentity();
        glEnable(GL_DEPTH_TEST);
    glEnable(GL_LINE_SMOOTH);
    glClearColor(0.0,0.0,0.0,1.0);


}
//---------------------------------------------------------------------------
void __fastcall TForm1::OpenGLPanel1Resize(TObject *Sender)
{
    glViewport(0,0,(GLsizei)OpenGLPanel1->Width,(GLsizei)OpenGLPanel1->Height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if ( OpenGLPanel1->Height==0)
        gluPerspective(45, (GLdouble)OpenGLPanel1->Width, 1.0, 3000.0);
    else
      gluPerspective(60, (GLdouble)OpenGLPanel1->Width/
        (GLdouble)OpenGLPanel1->Height, 1.0, 3000.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

}
//---------------------------------------------------------------------------

void __fastcall TForm1::OpenGLPanel1Paint(TObject *Sender)
{
int ii;
GLUquadricObj* q=gluNewQuadric();

  // EJECUTA Ejecuta[1000];
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();

    // Dibuja los ejes X, Y, Z
```

```
glTranslated(-3.0, -2.0, -5.0);
glRotatef(RotX, 1.0, 0.0, 0.0);
glRotatef(RotY, 0.0, 1.0, 0.0);

glPushMatrix();
    glTranslated(0.0, 0.55, 0.0);
    glScalef(0.13,0.13,0.13);
    glColor3f(0.0,0.0,1.0);
    OpenGLPanel1->Draw3DText("Z");
glPopMatrix();
glPushMatrix();
    glTranslated(0.55, 0.0, 0.0);
    glScalef(0.13,0.13,0.13);
    glColor3f(1.0,0.0,0.0);
    OpenGLPanel1->Draw3DText("X");
glPopMatrix();
glPushMatrix();
    glTranslated(0.0, 0.0, -0.55);
    glScalef(0.13,0.13,0.13);
    glColor3f(0.0,1.0,0.0);
    OpenGLPanel1->Draw3DText("Y");
glPopMatrix();

glLineWidth(2.0);
glBegin(GL_LINE_LOOP);
    glColor3f(1.0,0.0,0.0);
    glVertex3f(0.0,0.0,0.0);
    glVertex3f(0.5,0.0,0.0);
glEnd();
glBegin(GL_LINE_LOOP);
    glColor3f(0.0,1.0,0.0);
    glVertex3f(0.0,0.0,0.0);
    glVertex3f(0.0,0.0,-0.5);
glEnd();
    glBegin(GL_LINE_LOOP);
    glColor3f(0.0,0.0,1.0);
    glVertex3f(0.0,0.0,0.0);
    glVertex3f(0.0,0.5,0.0);
glEnd();

glPopMatrix();


glPushMatrix();

glTranslated(0.0, 0.0, -ZoomZ);
glRotatef(RotX, 1.0, 0.0, 0.0);
glRotatef(RotY, 0.0, 1.0, 0.0);
glTranslated(Desplaza[0], 0.0,Desplaza[1]);

glLineWidth(1.0);

Rejilla();
glScalef(Escala[0],Escala[2],Escala[1]);

glBegin(GL_LINE_LOOP);
```

```
  glColor3f(1.0,1.0,1.0);
  glVertex3f( 1.0, 1.0, 1.0);
  glVertex3f( -1.0, 1.0, 1.0);
  glVertex3f( -1.0, -1.0, 1.0);
  glVertex3f( 1.0, -1.0, 1.0);
glEnd();

glBegin(GL_LINE_LOOP);
  glVertex3f( -1.0, -1.0, 1.0);
  glVertex3f( -1.0, 1.0, 1.0);
  glVertex3f( -1.0, 1.0, -1.0);
  glVertex3f( -1.0, -1.0, -1.0);
glEnd();

glBegin(GL_LINE_LOOP);
  glNormal3f( 0.0, -1.0, 0.0);
  glVertex3f( -1.0, -1.0, 1.0);
  glVertex3f( 1.0, -1.0, 1.0);
  glVertex3f( 1.0, -1.0, -1.0);
  glVertex3f( -1.0, -1.0, -1.0);
glEnd();

glBegin(GL_LINE_LOOP);
  glVertex3f(-1.0, 1.0,-1.0);
  glVertex3f( 1.0, 1.0,-1.0);
  glVertex3f( 1.0,-1.0,-1.0);
  glVertex3f(-1.0,-1.0,-1.0);
glEnd();

glBegin(GL_LINE_LOOP);
  glVertex3f( 1.0, 1.0, 1.0);
  glVertex3f( 1.0,-1.0, 1.0);
  glVertex3f( 1.0,-1.0,-1.0);
  glVertex3f( 1.0, 1.0,-1.0);
glEnd();

glBegin(GL_LINE_LOOP);
  glVertex3f( 1.0, 1.0, 1.0);
  glVertex3f( -1.0, 1.0, 1.0);
  glVertex3f( -1.0, 1.0, -1.0);
  glVertex3f( 1.0, 1.0, -1.0);
glEnd();

glPopMatrix();

//if (Ejecutar==1)
//{
//    glPushMatrix();
//  glTranslated(-1.0, 1.0, 1.0);      // PSO
glPushMatrix();

    glTranslated(0.0, 0.0, -ZoomZ);
    glRotatef(RotX, 1.0, 0.0, 0.0);
    glRotatef(RotY, 0.0, 1.0, 0.0);
    glTranslated(Desplaza[0], 0.0, Desplaza[1]);
    glTranslated(-120.0, -5.0, 120.0);
```

```
    glTranslated(PSOX[Origen], PSOZ[Origen], -PSOY[Origen]);
    W();

if(Play==0)
{
    glBegin(GL_POINTS);
     glVertex3f(    Ejecuta[0].XX-PSOX[Origen],    Ejecuta[0].ZZ-PSOZ[Origen],    -Ejecuta[0].YY-
PSOY[Origen]);
    glEnd();
   glEnable(GL_LINE_STIPPLE);
   for(ii=1; ii <=Ejecutar; ii++)
    {
    switch(Ejecuta[ii].tipo)
     {
         case 0: glLineStipple(4,0xAAAA);
              glEnable(GL_LINE_STIPPLE);
              break;
         case 1: glDisable(GL_LINE_STIPPLE);
              break;
     }
    switch(Ejecuta[ii].color)
     {
         case 0: glColor3ub(255,255,0);     //AMARILLO
              break;
         case 1: glColor3ub(0,255,0);       //VERDE
              break;
         case 2: glColor3ub(0,0,255);       //AZUL
              break;
         case 3: glColor3ub(255,0,0);       //ROJO
              break;
         case 4: glColor3ub(255,0,255);     //MORADO
              break;
         case 5: glColor3ub(250,250,250);       //MORADO
              break;
         case 6: glColor3ub(150,150,150);       //GRIS
              break;
         case 7: glColor3ub(80,80,10);      //GRIS
              break;
     }
    glBegin(GL_LINES);
       glVertex3f( Ejecuta[ii-1].XX, Ejecuta[ii-1].ZZ, -Ejecuta[ii-1].YY);
       glVertex3f( Ejecuta[ii].XX, Ejecuta[ii].ZZ, -Ejecuta[ii].YY);
    glEnd();

    glDisable(GL_LINE_STIPPLE);
    }
    glDisable(GL_LINE_STIPPLE);
}
  glPopMatrix();
}

//---------------------------------------------------------------------------

void __fastcall TForm1::OpenGLPanel1MouseDown(TObject *Sender,
   TMouseButton Button, TShiftState Shift, int X, int Y)
{
```

```
    if (Button == mbLeft) Boton = 1;
   //  OpenGLPanel1->Cursor = (TCursor)ROTAR;}
    if (Button == mbMiddle) Boton = 2;
    if (Button == mbRight) Boton = 3;

    MouseDown=true;

    LastX=X;
    LastY=Y;
}
//---------------------------------------------------------------------

void __fastcall TForm1::OpenGLPanel1MouseUp(TObject *Sender,
    TMouseButton Button, TShiftState Shift, int X, int Y)
{
 MouseDown=false;
 Boton = 0;
 //OpenGLPanel1->Cursor = crDefault;
}
//---------------------------------------------------------------------

void __fastcall TForm1::OpenGLPanel1MouseMove(TObject *Sender,
    TShiftState Shift, int X, int Y)
{
 if(MouseDown)
        {
     if(Boton == 1 )
     {
         RotX += (float)(X - LastX)/5.0f;
         RotY += (float)(Y - LastY)/5.0f;
         LastX=X;
         LastY=Y;
        // OpenGLPanel1->Cursor = (TCursor)ROTAR;
     }
     if(Boton == 2)
     {
         ZoomZ += (float)(X - LastX)/5.0f;
         ZoomZ -= (float)(Y - LastY)/5.0f;
         LastX=X;
         LastY=Y;
         //OpenGLPanel1->Cursor = crAcercar;
     }
     if(Boton == 3)
     {
//          OpenGLPanel1->Cursor = LoadCursor(HInstance, "Mover");
         Desplaza[0] += (float)(X - LastX)/5.0f;
         Desplaza[1] -= (float)(LastY - Y)/5.0f;
         LastX=X;
         LastY=Y;
         //OpenGLPanel1->Cursor = crSizeAll;
     }
  OpenGLPanel1->Repaint();

  }
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
//    int i;
        Button3 -> Caption = IntToStr(i);
        StatusBar1 -> Panels ->Items[0]->Text = Memo1->Lines->Strings[i];
        Parser(Memo1->Lines->Strings[i].c_str());
        //Ejecutar = 1;
        PostParser();
        OpenGLPanel1->Repaint();
        //PostParser();
    //}
    i++;
    if (i>lineas) i=0;

/*      while(!feof(pcnc))
    {
     if(fgets(cad, 100, pcnc))
     {
      StatusPanel1->Text = cad;
     }
    }
*/
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
 //Escala[0] = 100.0;
 //Escala[1] = 100.0;
 //Escala[2] = 10.0;

}
//---------------------------------------------------------------------------
void TForm1::W()
{
GLUquadricObj* q=gluNewQuadric();

    glPushMatrix();
        glRotatef(90.0, 1.0, 0.0, 0.0);
        glColor3ub(183,104,62);
        gluPartialDisk(q,.5,3,30,1,0,90);
        gluPartialDisk(q,.5,3,30,1,180,90);
        gluDisk(q,3,3.2,30,1);

        glColor3f(1.0,1.0,0.0);
        gluPartialDisk(q,.5,3,30,1,90,90);
        gluPartialDisk(q,.5,3,30,1,270,90);
    glPopMatrix();
    gluDeleteQuadric(q);
}
//---------------------------------------------------------------------------
void TForm1::Rejilla()
{
```

```
GLfloat zExtent, xExtent, xLocal, zLocal;
int loopX, loopZ;

glPushMatrix();
glTranslated(0.0, -Escala[2], 0.0);
glColor3f(0.0,0.7,0.7);
glBegin(GL_LINES);
    zExtent = 20*10;
    for(loopX = -10; loopX <= 10; loopX++)
    {
        xLocal = 20*loopX;
        glVertex3f(xLocal, 0.0, -zExtent);
        glVertex3f(xLocal, 0.0,  zExtent);
    }
    xExtent = 20*10;
    for(loopZ = -10; loopZ <= 10; loopZ++)
    {
        zLocal = 20*loopZ;
        glVertex3f(-xExtent, 0.0, zLocal);
        glVertex3f( xExtent, 0.0, zLocal);
    }
glEnd();
glPopMatrix();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FileOpenExecute(TObject *Sender)
{
    int e;
    if (OpenDialog1->Execute())
    {

      FileName = OpenDialog1->FileName;
      StatusBar1 -> Panels ->Items[0]->Text = FileName;
      Memo1->Lines->LoadFromFile(FileName);

    lineas = Memo1->Lines->Count;
    StatusBar1 -> Panels ->Items[1]->Text = IntToStr(lineas) + " lineas";
    i=0;
    for (e=0;e <= 999; e++)
    {
      Ejecuta[e].XX = 0;
      Ejecuta[e].YY = 0;
      Ejecuta[e].ZZ = 0;
    }
    //i=0;
    }

}
//---------------------------------------------------------------------------

void __fastcall TForm1::FileExitExecute(TObject *Sender)
{
    Close();
}
```

```
//---------------------------------------------------------------------

void __fastcall TForm1::HelpAboutExecute(TObject *Sender)
{
    ShowAboutBox();
}
//---------------------------------------------------------------------

void __fastcall TForm1::DimensionExecute(TObject *Sender)
{
    MaterialBox->ShowModal();
    OpenGLPanel1->Repaint();
}
//---------------------------------------------------------------------
void __fastcall TForm1::ToolRadiusClick(TObject *Sender)
{
    ToolBox->ShowModal();
}
//---------------------------------------------------------------------
void __fastcall TForm1::CeroPiezaExecute(TObject *Sender)
{
    CeroPiezaBox->ShowModal();
    OpenGLPanel1->Repaint();
}
//---------------------------------------------------------------------

//---------------------------------------------------------------------

char *MID(char nome[100], int desde, int longi)
{
  static char nom2[100];
  char fin;
  int n;

    n=0;
    strcpy(nom2,"");
    while(n<longi)
    {
        nom2[n]=nome[desde-1+n];
        n++;
    }
    nom2[n]='\0';
  return(nom2);
}
//---------------------------------------------------------------------
int TForm1::Parser(char ls[100])
{
  double p;
  int letter;
  char *ptry;
  char sp[100];
  char *Q;

  for(letter=1; letter< strlen(ls)+1; letter++)
  {
    Q=MID(ls,letter,1);
```

```
switch(*Q)
{
    case 'N':
      Q=MID(ls,letter+1,10);
            //printf("%04d\n",atoi(Q));
      Post[0] = atoi(Q);
      //Label10->Caption = Post[0];
            break;
    case 'G':
            Q=MID(ls,letter+1,10);
      p=atoi(Q);
      Post[1]=p;
      //Label11->Caption = Post[1];
      if(p==71) Label22 -> Caption = "mm.";
      if(p==70) Label22 -> Caption = "plg.";
      if(p==54) Origen = 0;
      if(p==55) Origen = 1;
      if(p==57) Origen = 2;
      if(p==58) Origen = 3;
      if(p==59) Origen = 4;
      if(p==94) Feed = 0;
      if(p==95) Feed = 1;
      Label22 -> Caption = Origen;
      Label21 -> Caption = PSOX[Origen];
      Label29 -> Caption = PSOY[Origen];
      Label31 -> Caption = PSOZ[Origen];
            //printf("%02d\n",atoi(Q));
            break;
    case 'X':
            Q=MID(ls,letter+1,10);
      p= strtold(Q,&ptry);
      Xold = Post[2];
      //Label10->Caption = Xold;
      Post[2]=p;

      Label7 -> Caption =p; //FloatToStr(p);
      //printf("%0.4f\n",p);
            break;
    case 'Y':
            Q=MID(ls,letter+1,10);
      p=_strtold(Q,&ptry);
      Yold = Post[3];
      Post[3]=p;

      Label8 -> Caption =p; //FloatToStr(p);
            //printf("%0.4f\n",p);
            break;
    case 'Z':
            Q=MID(ls,letter+1,10);
      p=_strtold(Q,&ptry);
      Zold = Post[4];
      Post[4]=p;

      Label9 -> Caption =p; //FloatToStr(p);
            //printf("%0.4f\n",p);
            break;
```

```
case 'I':
            Q=MID(ls,letter+1,10);
      p=_strtold(Q,&ptry);
      Post[5]=p;
      //Label7 -> Caption =FloatToStr(p);
      //printf("%0.4f\n",p);
            break;
    case 'J':
            Q=MID(ls,letter+1,10);
      p=_strtold(Q,&ptry);
      Post[6]=p;
      //Label8 -> Caption =FloatToStr(p);
            //printf("%0.4f\n",p);
            break;
     case 'K':
            Q=MID(ls,letter+1,10);
      p=_strtold(Q,&ptry);
      Post[7]=p;
      //Label9 -> Caption =FloatToStr(p);
            //printf("%0.4f\n",p);
            break;
     case 'F':
            Q=MID(ls,letter+1,10);
      p=atoi(Q);
      Post[8]=p;
      if(Feed == 1) Post[8] = p * Post[19];
      if(Feed == 0) Post[8] = p;
      Label25 -> Caption =p;
            //printf("%0.2f\n",p);
            break;
     case 'T':
            Q=MID(ls,letter+1,10);
      p=atoi(Q);
      Label27 -> Caption =p;
      Q=MID(ls,letter+3,2);
      T=atoi(Q);
            //printf("%04d\n",atoi(Q));
            break;
     case 'M':
            Q=MID(ls,letter+1,10);
      p = atoi(Q);
      if(p==30)
      {
          i=0;
          //Ejecutar=0;
      }
            //printf("%02d\n", atoi(Q));
            break;
    case 'S':
            Q=MID(ls,letter+1,10);
      p=atoi(Q);
      Post[19] = p;
      Label23 -> Caption =p;
            //printf("%0.2f\n",p);
            break;
  case 'P':
```

```
        Q=MID(ls,letter+1,1);
    p=atoi(Q);
    if(p== 0)
    {
        Q=MID(ls,letter+3,10);
        p = strtold(Q,&ptry);
        Post[17]=p;       //P0
    }
    if(p==1)
    {
        Q=MID(ls,letter+3,10);
        p= strtold(Q,&ptry);
        Post[18]=p;       //P1
    }
    if(p==3)
    {
        Q=MID(ls,letter+3,10);
        p= strtold(Q,&ptry);
        Post[11]=p;       //P3
    }
            break;
case 'D':
            Q=MID(ls,letter+1,1);
    p=atoi(Q);
    if(p==3)
    {
        Q=MID(ls,letter+3,10);
        p=atoi(Q);
        Post[12]=p/1000;       //D3
    }
    if(p==4)
    {
        Q=MID(ls,letter+3,10);
        p=atoi(Q);
        Post[13]=p/10;       //D4
    }
    if(p==5)
    {
        Q=MID(ls,letter+3,10);
        p=atoi(Q);
        Post[14]=p/100;       //D5
    }
    if(p==6)
    {
        Q=MID(ls,letter+3,10);
        p= strtold(Q,&ptry);
        Post[15]=p/1000;       //D6
    }
    if(p==7)
    {
        Q=MID(ls,letter+3,10);
        p=atoi(Q);
        Post[16]=p;          //D7
    }
            break;
```

```
        default: break;

    }
  }
  return 0;
}

//---------------------------------------------------------------------------

void __fastcall TForm1::PorBloque1Click(TObject *Sender)
{
((TMenuItem*)Sender)->Checked = ! ((TMenuItem*)Sender)->Checked;
    // PorBloque1 ->Checked = true;

}
//---------------------------------------------------------------------------

void __fastcall TForm1::ToolButton5Click(TObject *Sender)
{
    int ii = 0;
    Play =0;

    if (PorBloque1->Checked == true)
    {
        if(i==0) Ejecutar = 0;
        Button3 -> Caption = IntToStr(i);
        StatusBar1 -> Panels ->Items[0]->Text = Memo1->Lines->Strings[i];
        Parser(Memo1->Lines->Strings[i].c_str());
        PostParser();
        OpenGLPanel1->Repaint();
        i++;

        if (i>=lineas) {
                i=0;
                Play = 0;
                //Ejecutar = 0;

                OpenGLPanel1->Repaint();
                }
    }
    else
    {
        Play = 1;
        Ejecutar = 0;
        Play = 0;
        for(ii=0; ii<=lineas; ii++)
        {
            StatusBar1 -> Panels ->Items[0]->Text = Memo1->Lines->Strings[ii];
            Parser(Memo1->Lines->Strings[ii].c_str());
            PostParser();
            OpenGLPanel1->Repaint();
        }

    }
    //OpenGLPanel1->Repaint();
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::ToolButton7Click(TObject *Sender)
{
     i=0;
     Ejecutar = 0;
     Label7->Caption    "0.0";
     Label8->Caption = "0.0";
     Label9->Caption    "0.0";
     Label22->Caption    "mm.";
     Label23->Caption = "0.0";
     Label25->Caption    "0.0";
     Label26->Caption = "Apagado";
     Label27->Caption = "0000";
}
//---------------------------------------------------------------------------
void TForm1::PostParser()
{
     float x,y,z;
     float Prof;
     float ProfCapas;
     float Pasadas,Capas;
     x=Post[2];
     y=Post[3];
     z=Post[4];
     float D3;
     float P0,P1,P3;
     float Radio;
     float I,J,xx, yy;
     float Xc,Yc;
     float Xi, Yi, Xf, Yf;
     float Ai, Af;
     float Angle;
     float i;
     int Qx, Qy;
     float Rh;
     div_t n;
     float A,B,Aa,X,Y;
     if(Post[1] == 0)
     {
        Ejecuta[Ejecutar].XX = x;
        Ejecuta[Ejecutar].YY = y;
        Ejecuta[Ejecutar].ZZ = z;
        Ejecuta[Ejecutar].FF = 3000;
        Ejecuta[Ejecutar].tipo = 0;
        Ejecuta[Ejecutar].color = 0;
        //Label10->Caption= Ejecutar;

        Ejecutar ++;
     }
     if(Post[1] == 1)
     {
        Ejecuta[Ejecutar].XX = x;
        Ejecuta[Ejecutar].YY = y;
        Ejecuta[Ejecutar].ZZ = z;
```

```
    Ejecuta[Ejecutar].FF   Post[8];
    Ejecuta[Ejecutar].tipo   1;
    Ejecuta[Ejecutar].color   1;
    //Label10->Caption   Ejecutar;

    Ejecutar ++;
  }
/*  if(Post[1] == 2)
  {
    I = Post[5];
    J = Post[6];
    Radio = sqrt((I*I)+(J*J));
    / Label10->Caption   Radio;
    Xc = Xold + I;
    Yc = Yold + J;

    Xi = abs(I);
    Yi = abs(J);

    if(x >= Xc) { Xf = x - Xc;  Qx = -1;}
    if(x <= Xc) { Xf   Xc - x;  Qx =  1;}
    if(y >= Yc) { Yf   y - Yc;  Qy = -1;}
    if(y <  Yc) { Yf   Yc - y;  Qy =  1;}

    Label21->Caption   Xc;
    Label28->Caption= Yc;
    Label29->Caption= Xi;
    Label30->Caption= Yi;
    Label31->Caption= Xf;
    Label32->Caption= Yf;

    if(Xi == 0)
    {
      if(Yi > 0) Ai = 90;
      if(Yi < 0) Ai = 270;
    }
    if(Yi == 0)
    {
      if(Xi > 0) Ai = 0;
      if(Xi < 0) Ai = 180;
    }
    if((Xi != 0)&& (Yi != 0))
    {
      Ai = atan(Yi/Xi);
      if( I < 0 && J < 0) Ai +=0;
      if( I > 0 && J < 0) Ai +=90;
      if( I > 0 && J > 0) Ai +=180;
      if( I < 0 && J > 0) Ai +=270;
    }

    if(Xf == 0)
    {
      if(Yf > 0) Af = 90;
      if(Yf < 0) Af = 270;
    }
    if(Yf == 0)
```

```
{
  if(Xf > 0) Af   0;
  if(Xf < 0) Af   180;
}
if((Xf !  0)&& (Yf !  0))
{
  Af   atan(Yf Xf);
  if( Qx < 0 && Qy < 0) Af +=0;
  if( Qx > 0 && Qy < 0) Af +=90;
  if( Qx > 0 && Qy > 0) Af +=180;
  if( Qx < 0 && Qy > 0) Af +  270;
}

//At -= .1f;
//Af += .1f;
Label33->Caption   Af;
//Label28->Caption   Yc;
for(Angle - At; Angle >- Af; Angle -=0.1f)
{
   xx - Radio * cos(Angle);
   yy - Radio * sin(Angle);

   if(xx >- Xc) { Ejecuta[Ejecutar].XX = Xc - xx;}
   if(xx <  Xc) { Ejecuta[Ejecutar].XX = Xc + xx;}
   / if(y >= Yc) { Yf   y - Yc;  Qy - -1;}
    if(y <- Yc) { Yf = Yc - y;  Qy =  1;}
   /   Ejecuta[Ejecutar].XX = Xc + xx;
   Ejecuta[Ejecutar].YY = Yc - yy;
   Ejecuta[Ejecutar].ZZ = z;
   Ejecuta[Ejecutar].tipo = 1;
   Ejecuta[Ejecutar].color   6;
   Label10->Caption   Ejecutar;

   Ejecutar ++;
}

  Ejecuta[Ejecutar].XX = x;
  Ejecuta[Ejecutar].YY - y;
  Ejecuta[Ejecutar].ZZ = z;
  Ejecuta[Ejecutar].tipo   1;
  Ejecuta[Ejecutar].color - 6;

  Ejecutar ++;
}

if(Post[1] — 3)
{
  I = Post[5];
  J = Post[6];
  Radio = sqrt((I*I)+(J*J));
  Xc = Xold + I;
  Yc = Yold + J;
  Xi = abs(I);
  Yi = abs(J);
  if(x > Xc) { Xf = x - Xc;  Qx = -1;}
  if(x < Xc) { Xf = Xc - x;  Qx =  1;}
```

```
if(y > Yc) { Yf   y - Yc; Qy   -1;}
if(y < Yc) { Yf   Yc - y; Qy    1;}

//II   Xc + x;
 JJ   Yc + y;

Ai   atan(Yi/Xi);
Af   atan(Yf/Xf);

if( I < 0 && J < 0) Ai +  0;
if( I > 0 && J < 0) Ai +-90;
if( I > 0 && J > 0) Ai +  180;
if( I < 0 && J > 0) Ai +  270;

if( Qx < 0 && Qy < 0) Af +  0;
if( Qx > 0 && Qy < 0) Af +-90;
if( Qx > 0 && Qy > 0) Af +  180;
if( Qx < 0 && Qy > 0) Af +  270;

  Ai +- 1.0f;
  Af -- 1.0f;
  for(Angle   Af; Angle <   Ai; Angle -=1.0)
  {
    xx = Radio * cos(Angle);
    yy - Radio * sin(Angle);

     Ejecuta[Ejecutar].XX = Xc + xx;
     Ejecuta[Ejecutar].YY - Yc + yy;
     Ejecuta[Ejecutar].ZZ   Zold;
     Ejecuta[Ejecutar].tipo = 1;
     Ejecuta[Ejecutar].color = 1;
     Label10->Caption   Ejecutar;

     Ejecutar ++;
  }
  Ejecuta[Ejecutar].XX   x;
  Ejecuta[Ejecutar].YY   y;
  Ejecuta[Ejecutar].ZZ   z;
  Ejecuta[Ejecutar].tipo - 1;
  Ejecuta[Ejecutar].color   1;

}
*/
if(Post[1] == 81)
{
   Ejecuta[Ejecutar].XX - x;
   Ejecuta[Ejecutar].YY - y;
   Ejecuta[Ejecutar].ZZ   Zold;
   Ejecuta[Ejecutar].FF = 3000;
   Ejecuta[Ejecutar].tipo = 0;
   Ejecuta[Ejecutar].color = 2;
   Ejecutar ++;

   Ejecuta[Ejecutar].XX = x;
   Ejecuta[Ejecutar].YY - y;
```

```
Ejecuta[Ejecutar].ZZ    Post[11];
Ejecuta[Ejecutar].FF    3000;
Ejecuta[Ejecutar].tipo    0;
Ejecuta[Ejecutar].color    2;
Ejecutar ++;

Ejecuta[Ejecutar] XX    x;
Ejecuta[Ejecutar].YY    y;
Ejecuta[Ejecutar].ZZ    z;
Ejecuta[Ejecutar].FF    Post[8];
Ejecuta[Ejecutar].tipo    1;
Ejecuta[Ejecutar].color    2;
Ejecutar ++;

Ejecuta[Ejecutar].XX    x;
Ejecuta[Ejecutar].YY    y;
Ejecuta[Ejecutar].ZZ    Post[11];
Ejecuta[Ejecutar].FF    3000;
Ejecuta[Ejecutar].tipo = 0;
Ejecuta[Ejecutar].color    2;
Ejecutar ++;
Post[4]  Post[11];
}
if(Post[1] == 82)
{
Ejecuta[Ejecutar].XX    x;
Ejecuta[Ejecutar].YY    y;
Ejecuta[Ejecutar].ZZ    Zold;
Ejecuta[Ejecutar] FF    3000;
Ejecuta[Ejecutar].tipo    0;
Ejecuta[Ejecutar].color    3;
Ejecutar ++;

Ejecuta[Ejecutar].XX    x;
Ejecuta[Ejecutar].YY    y;
Ejecuta[Ejecutar].ZZ    Post[11];
Ejecuta[Ejecutar].FF    3000;
Ejecuta[Ejecutar].tipo = 0;
Ejecuta[Ejecutar].color = 3;
Ejecutar ++;

Ejecuta[Ejecutar].XX = x;
Ejecuta[Ejecutar].YY = y;
Ejecuta[Ejecutar].ZZ    z;
Ejecuta[Ejecutar].FF = Post[8];
Ejecuta[Ejecutar].tipo = 1;
Ejecuta[Ejecutar].color = 3;
Ejecutar ++;

Ejecuta[Ejecutar].XX = x;
Ejecuta[Ejecutar].YY = y;
Ejecuta[Ejecutar].ZZ    Post[11];
Ejecuta[Ejecutar].FF = 3000;
Ejecuta[Ejecutar].tipo = 0;
Ejecuta[Ejecutar].color = 3;
Ejecutar ++;
```

```
  Post[4]  Post[11];
}

if(Post[1]    83)
{
  Ejecuta[Ejecutar].XX   x;
  Ejecuta[Ejecutar].YY   y;
  Ejecuta[Ejecutar] ZZ   Zold;
  Ejecuta[Ejecutar].FF   3000;
  Ejecuta[Ejecutar].tipo   0;
  Ejecuta[Ejecutar].color   4;
  Ejecutar ++;

  Ejecuta[Ejecutar] XX   x;
  Ejecuta[Ejecutar].YY   y;
  Ejecuta[Ejecutar].ZZ   Post[11];
  Ejecuta[Ejecutar].FF   3000;
  Ejecuta[Ejecutar].tipo   0;
  Ejecuta[Ejecutar].color   4;
  Ejecutar ++;

  Profundidad   Post[11] - Post[12];
  D3 = Post[12];
  Prof  Post[11];     /P3

  while(Profundidad >- Post[4])
  {
    Ejecuta[Ejecutar].XX   x;
    Ejecuta[Ejecutar].YY   y;
    Ejecuta[Ejecutar].ZZ - Prof;
    Ejecuta[Ejecutar].FF   3000;
    Ejecuta[Ejecutar].tipo   0;
    Ejecuta[Ejecutar].color   4;
    Ejecutar ++;
    Prof - Profundidad + .5;  /User Monitor

   // x= x+ .1;

    Ejecuta[Ejecutar].XX   x;
    Ejecuta[Ejecutar].YY   y;
    Ejecuta[Ejecutar].ZZ   Profundidad;
    Ejecuta[Ejecutar].FF = Post[8];
    Ejecuta[Ejecutar].tipo = 1;
    Ejecuta[Ejecutar].color - 4;
    Ejecutar ++;
  //   x=x+.1;

    Ejecuta[Ejecutar].XX - x;
    Ejecuta[Ejecutar].YY   y;
    Ejecuta[Ejecutar].ZZ - Post[11];
    Ejecuta[Ejecutar].FF - 3000;
    Ejecuta[Ejecutar].tipo   0;
    Ejecuta[Ejecutar].color = 4;
    Ejecutar ++;
  // x=x+.1;
```

```
D3   D3 * Post[14];
if(D3 >= Post[15])
     {
     Profundidad   Profundidad - D3;
     }
else  {
     Profundidad   Profundidad - Post[15];
     }

}
Ejecuta[Ejecutar].XX   x;
Ejecuta[Ejecutar].YY   y;
Ejecuta[Ejecutar].ZZ   Post[11];
Ejecuta[Ejecutar].FF   3000;
Ejecuta[Ejecutar].tipo   0;
Ejecuta[Ejecutar].color   4;
Ejecutar ++;
Post[4]  Post[11];
}
if(Post[1] == 86)
{
  Ejecuta[Ejecutar].XX   x;
  Ejecuta[Ejecutar].YY   y;
  Ejecuta[Ejecutar].ZZ   Zold;
  Ejecuta[Ejecutar].FF   3000;
  Ejecuta[Ejecutar].tipo   0;
  Ejecuta[Ejecutar].color   5;
  Ejecutar ++;

  Ejecuta[Ejecutar].XX = x;
  Ejecuta[Ejecutar].YY = y;
  Ejecuta[Ejecutar].ZZ   Post[11];
  Ejecuta[Ejecutar].FF   3000;
  Ejecuta[Ejecutar].tipo   0;
  Ejecuta[Ejecutar].color = 5;
  Ejecutar ++;

  Profundidad   Post[11] - Post[12];
  D3 = Post[12];

  while(Profundidad >   Post[4])
  {
    Ejecuta[Ejecutar].XX = x;
    Ejecuta[Ejecutar].YY   y;
    Ejecuta[Ejecutar].ZZ = Profundidad;
    Ejecuta[Ejecutar].FF = Post[8];
    Ejecuta[Ejecutar].tipo = 1;
    Ejecuta[Ejecutar].color = 5;
    Ejecutar ++;
    // x=x+.1;

     Ejecuta[Ejecutar].XX   x;
     Ejecuta[Ejecutar].YY = y;
     Ejecuta[Ejecutar].ZZ = Profundidad + .5;
```

```
Ejecuta[Ejecutar].FF = 3000;
Ejecuta[Ejecutar].tipo = 0;
Ejecuta[Ejecutar].color = 5;
Ejecutar ++;
//x=x+.1;

D3 = D3 * Post[14];
if(D3 >= Post[15])
      {
      Profundidad = Profundidad -D3;
      }
 else  {
      Profundidad = Profundidad - Post[15];
      }
}
Ejecuta[Ejecutar].XX = x;
Ejecuta[Ejecutar].YY = y;
Ejecuta[Ejecutar].ZZ = Post[11];
Ejecuta[Ejecutar].FF = 3000;
Ejecuta[Ejecutar].tipo = 0;
Ejecuta[Ejecutar].color = 5;
Ejecutar ++;
Post[4]=Post[11];
}
if(Post[1] == 87)
{

P0 = Post[17];
P1 = Post[18];
P3 = Post[11];
Rh = Tools[T-1];
A = P0/2;
B = P1/2;

if(P0 >= P1) n= div(B,Rh);
if(P1 >  P0) n= div(A,Rh);

if( n.rem > 0) Pasadas = n.quot + 1;
else Pasadas = n.quot;

//n = div(A,Rh);
// if( n.rem > 0) Pasadas = n.quot + 1;
//else Pasadas = n.quot;

D3 = Post[12];

Ejecuta[Ejecutar].XX = x;
Ejecuta[Ejecutar].YY = y;
Ejecuta[Ejecutar].ZZ = Zold;
Ejecuta[Ejecutar].FF = 3000;
Ejecuta[Ejecutar].tipo = 0;
Ejecuta[Ejecutar].color = 7;
Ejecutar ++;

Ejecuta[Ejecutar].XX = x;
Ejecuta[Ejecutar].YY = y;
```

```
Ejecuta[Ejecutar].ZZ = Post[11];
Ejecuta[Ejecutar].FF = 3000;
Ejecuta[Ejecutar].tipo = 0;
Ejecuta[Ejecutar].color = 7;
Ejecutar ++;

if((P0 < Rh) && (P1< Rh))
   {
   ShowMessage("Imposible realizar la caja rectangular\nRevise  P0  y/o  P1");
    }
else
    {
    Prof = P3 + abs(z);
    n = div (Prof,D3);
    if( n.rem > 0) Capas = n.quot + 1;
    else Capas = n.quot;

    ProfCapas = Prof/Capas;
    Profundidad = P3 - ProfCapas;

     if(P0 >= P1) X = A - (Pasadas*Rh)+Rh;
     if(P1 > P0)  Y = B - (Pasadas*Rh)+Rh;

    Label33->Caption = Y;
    Label32->Caption = X;
    if (P1>P0)
    {
    while(Profundidad >= Post[4])
    {
          Ejecuta[Ejecutar].XX = x;
          Ejecuta[Ejecutar].YY = y;
          Ejecuta[Ejecutar].ZZ = Profundidad;
          Ejecuta[Ejecutar].FF = Post[8];
          Ejecuta[Ejecutar].tipo = 1;
          Ejecuta[Ejecutar].color = 7;
          Ejecutar ++;

          Ejecuta[Ejecutar].XX = x;
          Ejecuta[Ejecutar].YY = y-Y+Rh;
          Ejecuta[Ejecutar].ZZ = Profundidad;
          Ejecuta[Ejecutar].FF = Post[8];
          Ejecuta[Ejecutar].tipo = 1;
          Ejecuta[Ejecutar].color = 7;
          Ejecutar ++;

          Ejecuta[Ejecutar].XX = x;
          Ejecuta[Ejecutar].YY = y+Y-Rh;
          Ejecuta[Ejecutar].ZZ = Profundidad;
          Ejecuta[Ejecutar].FF = Post[8];
          Ejecuta[Ejecutar].tipo = 1;
          Ejecuta[Ejecutar].color = 7;
          Ejecutar ++;

          for(i=1; i<=Pasadas; i++)
          {
               X += Rh;
```

```
        if( X >= (A-Rh)) X=A-Rh;

        Ejecuta[Ejecutar].XX = x;
        Ejecuta[Ejecutar].YY = y+Y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = 3000;
        Ejecuta[Ejecutar].tipo = 0;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Ejecuta[Ejecutar].XX = x-X;
        Ejecuta[Ejecutar].YY = y+Y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Ejecuta[Ejecutar].XX = x-X;
        Ejecuta[Ejecutar].YY = y-(Y);
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Ejecuta[Ejecutar].XX = x+X;
        Ejecuta[Ejecutar].YY = y-(Y);
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Ejecuta[Ejecutar].XX = x+X;
        Ejecuta[Ejecutar].YY = y+Y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Ejecuta[Ejecutar].XX = x;
        Ejecuta[Ejecutar].YY = y+Y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Y+=Rh;
        if( Y > (B-Rh)) Y = B - Rh;

}
X = 0;
Y = B - (Pasadas*Rh)+Rh;
```

```
        Ejecuta[Ejecutar].XX = x;
        Ejecuta[Ejecutar].YY = y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = 3000;
        Ejecuta[Ejecutar].tipo = 0;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;
        Profundidad -= ProfCapas;
}
}
if (P0>=P1)
 {
 while(Profundidad >= Post[4])
 {
        Ejecuta[Ejecutar].XX = x;
        Ejecuta[Ejecutar].YY = y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Ejecuta[Ejecutar].XX = x-X+Rh;
        Ejecuta[Ejecutar].YY = y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;

        Ejecuta[Ejecutar].XX = x+X-Rh;
        Ejecuta[Ejecutar].YY = y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = Post[8];
        Ejecuta[Ejecutar].tipo = 1;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;
        for(i=1; i<=Pasadas; i++)
        {
                Y += Rh;
                if( Y >= (B-Rh)) Y=B-Rh;

                Ejecuta[Ejecutar].XX = x+X;
                Ejecuta[Ejecutar].YY = y;
                Ejecuta[Ejecutar].ZZ = Profundidad;
                Ejecuta[Ejecutar].FF = 3000;
                Ejecuta[Ejecutar].tipo = 0;
                Ejecuta[Ejecutar].color = 7;
                Ejecutar ++;

                Ejecuta[Ejecutar].XX = x+X;
                Ejecuta[Ejecutar].YY = y+Y;
                Ejecuta[Ejecutar].ZZ = Profundidad;
                Ejecuta[Ejecutar].FF = Post[8];
```

```
            Ejecuta[Ejecutar].tipo = 1;
            Ejecuta[Ejecutar].color = 7;
            Ejecutar ++;

            Ejecuta[Ejecutar].XX = x-X;
            Ejecuta[Ejecutar].YY = y+(Y);
            Ejecuta[Ejecutar].ZZ = Profundidad;
            Ejecuta[Ejecutar].FF = Post[8];
            Ejecuta[Ejecutar].tipo = 1;
            Ejecuta[Ejecutar].color = 7;
            Ejecutar ++;

            Ejecuta[Ejecutar].XX = x-X;
            Ejecuta[Ejecutar].YY = y-(Y);
            Ejecuta[Ejecutar].ZZ = Profundidad;
            Ejecuta[Ejecutar].FF = Post[8];
            Ejecuta[Ejecutar].tipo = 1;
            Ejecuta[Ejecutar].color = 7;
            Ejecutar ++;

            Ejecuta[Ejecutar].XX = x+X;
            Ejecuta[Ejecutar].YY = y-Y;
            Ejecuta[Ejecutar].ZZ = Profundidad;
            Ejecuta[Ejecutar].FF = Post[8];
            Ejecuta[Ejecutar].tipo = 1;
            Ejecuta[Ejecutar].color = 7;
            Ejecutar ++;

            Ejecuta[Ejecutar].XX = x+X;
            Ejecuta[Ejecutar].YY = y;
            Ejecuta[Ejecutar].ZZ = Profundidad;
            Ejecuta[Ejecutar].FF = Post[8];
            Ejecuta[Ejecutar].tipo = 1;
            Ejecuta[Ejecutar].color = 7;
            Ejecutar ++;

            X+=Rh;
            if( X > (A-Rh)) X = A - Rh;
        }
        Y = 0;
        X = A - (Pasadas*Rh)+Rh;

        Ejecuta[Ejecutar].XX = x;
        Ejecuta[Ejecutar].YY = y;
        Ejecuta[Ejecutar].ZZ = Profundidad;
        Ejecuta[Ejecutar].FF = 3000;
        Ejecuta[Ejecutar].tipo = 0;
        Ejecuta[Ejecutar].color = 7;
        Ejecutar ++;
        Profundidad -= ProfCapas;
}
}
Ejecuta[Ejecutar].XX = x;
Ejecuta[Ejecutar].YY = y;
Ejecuta[Ejecutar].ZZ = Post[11];
Ejecuta[Ejecutar].FF = 3000;
```

```
            Ejecuta[Ejecutar].tipo = 0;
            Ejecuta[Ejecutar].color = 7;
            Ejecutar ++;
            Post[4]=Post[11];
            }

    }

}

//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
      int ii;
      float D;
      float Dist;
      float Dx, Dy, Dz;

      Dist = 0;
        for(ii=1; ii <=Ejecutar; ii++)
            {
            Dx = Ejecuta[ii-1].XX - Ejecuta[ii].XX;
            Dy = Ejecuta[ii-1].YY - Ejecuta[ii].YY;
            Dz = Ejecuta[ii-1].ZZ - Ejecuta[ii].ZZ;

            D = sqrt((Dx*Dx) + (Dy*Dy) + (Dz*Dz));
            Dist = Dist + D;
            }
       Label28->Caption = Dist;
            Label30->Caption = Ejecuta[0].YY;
                  Label32->Caption = Ejecuta[0].ZZ;

}
//---------------------------------------------------------------------------
```


DEMO1.H

```
//---------------------------------------------------------------------------
#ifndef Demo1H
#define Demo1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "OpenGLPanel.h"
#include <ComCtrls.hpp>
#include <Dialogs.hpp>
#include <ExtCtrls.hpp>
#include <ToolWin.hpp>
#include <ActnList.hpp>
#include <ImgList.hpp>
#include <Menus.hpp>
```

```
#include <math.h>

#include "commondefs.h"

//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
   TOpenGLPanel *OpenGLPanel1;
      TImageList *ImageList1;
      TActionList *ActionList1;
      TAction *FileOpen;
      TAction *HelpAbout;
      TToolBar *ToolBar1;
      TToolButton *ToolButton1;
      TToolButton *ToolButton2;
      TToolButton *ToolButton3;
      TMainMenu *MainMenu1;
      TMenuItem *Archivo1;
      TMenuItem *Abrir1;
      TAction *FileExit;
      TMenuItem *Salir1;
      TMenuItem *Pieza1;
      TMenuItem *Ayuda1;
      TMenuItem *Ayuda2;
      TOpenDialog *OpenDialog1;
      TToolButton *ToolButton4;
      TToolButton *ToolButton5;
      TAction *Dimension;
      TMenuItem *Dimensiones1;
      TStatusBar *StatusBar1;
      TMemo *Memo1;
      TPanel *Panel1;
      TButton *Button3;
      TGroupBox *GroupBox1;
      TLabel *Label1;
      TLabel *Label2;
      TLabel *Label3;
      TLabel *Label14;
      TLabel *Label15;
      TLabel *Label4;
      TLabel *Label5;
      TLabel *Label6;
      TLabel *Label7;
      TLabel *Label9;
      TLabel *Label10;
      TLabel *Label11;
      TLabel *Label12;
      TGroupBox *GroupBox2;
      TLabel *Label13;
      TLabel *Label16;
      TLabel *Label17;
      TLabel *Label18;
      TLabel *Label19;
      TLabel *Label22;
      TLabel *Label23;
```

```
    TLabel *Label24;
    TLabel *Label25;
    TLabel *Label26;
    TToolButton *ToolButton6;
    TToolButton *ToolButton7;
    TToolButton *ToolButton8;
    TToolButton *ToolButton9;
    TMenuItem *Modo1;
    TMenuItem *Automtico1;
    TMenuItem *PorBloque1;
    TMenuItem *Pausa1;
    TMenuItem *Detener1;
    TLabel *Label8;
    TLabel *Label20;
    TLabel *Label27;
    TToolButton *ToolCeroPieza;
    TToolButton *ToolRadius;
    TLabel *Label21;
    TLabel *Label28;
    TLabel *Label29;
    TLabel *Label30;
    TLabel *Label31;
    TLabel *Label32;
    TLabel *Label33;
    TMenuItem *CerodePieza1;
    TAction *CeroPieza;
    TButton *Button1;

  void __fastcall OpenGLPanel1Init(TObject *Sender);
  void __fastcall OpenGLPanel1Resize(TObject *Sender);
  void __fastcall OpenGLPanel1Paint(TObject *Sender);

  void __fastcall OpenGLPanel1MouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y);
  void __fastcall OpenGLPanel1MouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y);
  void __fastcall OpenGLPanel1MouseMove(TObject *Sender, TShiftState Shift, int X, int Y);
    void __fastcall Button3Click(TObject *Sender);
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall FileOpenExecute(TObject *Sender);
    void __fastcall FileExitExecute(TObject *Sender);
    void __fastcall HelpAboutExecute(TObject *Sender);
    void __fastcall DimensionExecute(TObject *Sender);
    void __fastcall PorBloque1Click(TObject *Sender);
    void __fastcall ToolButton5Click(TObject *Sender);
    void __fastcall ToolButton7Click(TObject *Sender);
    void __fastcall ToolRadiusClick(TObject *Sender);
    void __fastcall CeroPiezaExecute(TObject *Sender);
    void __fastcall Button1Click(TObject *Sender);


private:  // User declarations
        int Escala[3];
        float Tools[9];
        float PSOX[5];
        float PSOY[5];
```

```
        float PSOZ[5];

public:                 // User declarations
    int Longitud, Altura, Anchura;
    float RotX, RotY,ZoomZ;
    Boolean MouseDown;
    int Boton;
    int LastX, LastY;
    int Desplaza[2];
    void Rejilla(void);
    void W(void);
    AnsiString FileName;
    int i;
    int lineas;
    float Post[19];
    float Xold, Yold,Zold,Profundidad;
    int Ejecutar;
    typedef struct {
            float XX;
            float YY;
            float ZZ;
            float FF;
            int tipo;
            int color;
            } EJECUTA;
    EJECUTA Ejecuta[1000];
    int Origen, Feed;
    int Play;
    int T;

    __fastcall TForm1(TComponent* Owner);
    friend class TMaterialBox;
    friend class TToolBox;
    friend class TCeroPiezaBox;
    int TForm1::Parser(char ls[100]);
    void PostParser(void);
};
//---------------------------------------------------------------------------
extern TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
const int crRotar   = 5;
const int crAcercar  = 6;
const int crMover   = 7;
```

ABOUT.CPP

```
//---------------------------------------------------------------------------
#include "about.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TAboutBox *AboutBox;

__fastcall TAboutBox::TAboutBox(TComponent* Owner) : TForm(Owner)
```

```
{
}

void __fastcall TAboutBox::FormCreate(TObject *Sender)
{
 Caption = Format("Acerca de %s",
           ARRAYOFCONST((Application->Title)));
 //ProgramIcon->Picture->Assign(Application->Icon);
 ProgramName->Caption = Application->Title;
}


void __fastcall ShowAboutBox()
{
 TAboutBox* a = new TAboutBox(Application);
 try{
  a->ShowModal();
 }
 catch(...){
  a->Free();
  throw;
 }
 a->Free();
}

//---------------------------------------------------------------------------
```

ABOUT.H

```
//Converting header file...
//---------------------------------------------------------------------------
#ifndef AboutHPP
#define AboutHPP
//---------------------------------------------------------------------------
#include <Windows.hpp>
#include <Classes.hpp>
#include <Graphics.hpp>
#include <Forms.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Buttons.hpp>
#include <ExtCtrls.hpp>
#include <SysUtils.hpp>

//-- type declarations -------------------------------------------------------
class TAboutBox : public TForm
{

__published:
        TButton *OKButton;
        TPanel *Panel1;
        TImage *ProgramIcon;
        TLabel *ProgramName;
        TLabel *Copyright;
    TLabel *Label1;
```

```
    TLabel *Label2;
        void __fastcall FormCreate(TObject *Sender);


public:
        __fastcall  virtual TAboutBox::TAboutBox(TComponent *Owner);
};

//-- var, const, procedure ---------------------------------------------
extern void __fastcall ShowAboutBox(void);
//-- end unit ----------------------------------------------------------
#endif   // About
```

CEROW.CPP

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "CeroW.h"
#include "Demo1.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TCeroPiezaBox *CeroPiezaBox;
//---------------------------------------------------------------------
__fastcall TCeroPiezaBox::TCeroPiezaBox(TComponent* Owner)
    : TForm(Owner)
{
char Valor[20];
    //FILE *en;
    Fila =1;
    Col =1;

    if((en = fopen("PSO.MIL", "rt")) == NULL)
    {
        ShowMessage("\nImposible abrir el archivo.\n");
    }
    while(!feof(en))
    {
        fgets(Valor, 20,en);
        StringGrid1->Cells[Col][Fila] = atof(Valor);
        Contenido[Col-1][Fila-1] = atof(Valor);
        Col++;
        if(Col>3)
        {
            Fila++;
            Col=1;
        }
    }
    StringGrid1->Cells[0][0] = "   N° de lista";
    StringGrid1->Cells[1][0] = "        X";
    StringGrid1->Cells[2][0] = "        Y";
```

```
    StringGrid1->Cells[3][0] = "        Z";
    StringGrid1->Cells[0][1] = "        1";
    StringGrid1->Cells[0][2] = "        2";
    StringGrid1->Cells[0][3] = "        3";
    StringGrid1->Cells[0][4] = "        4";
    StringGrid1->Cells[0][5] = "        5";
    fclose(en);
}
//---------------------------------------------------------------------------
void __fastcall TCeroPiezaBox::AceptarClick(TObject *Sender)
{
    int i,j;
    char* Val;
    AnsiString Valor;
    float ValorNumerico;

    if((en = fopen("PSO.MIL", "wt")) == NULL)
    {
        ShowMessage("\nImposible abrir el archivo.\n");
    }
    for(i=0;i<=5;i++)
    {
        Form1->PSOX[i] = Contenido[0][i];
        Form1->PSOY[i] = Contenido[1][i];
        Form1->PSOZ[i] = Contenido[2][i];
    }
    fseek(en,0,0);
    //ftell(en);
    for(i=0;i<=4;i++)
    {
      for(j=0;j<=2;j++)
      {
        Valor = AnsiString(Contenido[j][i]);
        while (Valor.Length() && Valor[Valor.Length()] == ' ')
        {
            Valor.SetLength(Valor.Length()-1);
        }
        fputs(Valor.c_str(),en);
        fputs("\n",en);
      }
    }
    fclose(en);
    Close();
}
//---------------------------------------------------------------------------

void __fastcall TCeroPiezaBox::StringGrid1KeyPress(TObject *Sender,
    char &Key)
{
    if ((Key < '0' || Key > '9') && (Key != 8) && (Key != '.') && (Key !='-'))
        Key = 0;
}
//---------------------------------------------------------------------------

void __fastcall TCeroPiezaBox::StringGrid1SetEditText(TObject *Sender,
    int ACol, int ARow, const AnsiString Value)
```

```
{
    AnsiString Valor;
    float ValorNumerico;

    Valor = Value;

    while (Valor.Length() && Valor[Valor.Length()] == ' ')
    {
        Valor.SetLength(Valor.Length()-1);
    }
    ValorNumerico = atof(Valor.c_str());
    Contenido[ACol-1][ARow-1] = ValorNumerico;
}
//---------------------------------------------------------------------------

void __fastcall TCeroPiezaBox::StringGrid1GetEditText(TObject *Sender,
    int ACol, int ARow, AnsiString &Value)
{
    Value = AnsiString(Contenido[ACol-1][ARow-1]);
}
//---------------------------------------------------------------------------

void __fastcall TCeroPiezaBox::StringGrid1SelectCell(TObject *Sender,
    int ACol, int ARow, bool &CanSelect)
{
    CanSelect = true;
}
//---------------------------------------------------------------------------

void __fastcall TCeroPiezaBox::StringGrid1DrawCell(TObject *Sender,
    int ACol, int ARow, TRect &Rect, TGridDrawState State)
{
    if(ACol>2)
    StringGrid1->Cells[ACol+1][ARow+1] = AnsiString(Contenido[ACol-1][ARow-1]);
}
//---------------------------------------------------------------------------

void __fastcall TCeroPiezaBox::StringGrid1GetEditMask(TObject *Sender,
    int ACol, int ARow, AnsiString &Value)
{
    //Value = "999.99";
}
//---------------------------------------------------------------------------

void __fastcall TCeroPiezaBox::CancelarClick(TObject *Sender)
{
    /*int i;
    for(i=0;i<=5;i++)
    {
        PSOx[i] = Contenido[0][i];
        PSOy[i] = Contenido[1][i];
        PSOz[i] = Contenido[2][i];
    }
    Label1->Caption = PSOx[1];
    Label2->Caption = PSOy[1];
    Label3->Caption = PSOz[1];
```

```
    */
    fclose(en);
    Close();
}
//---------------------------------------------------------------------------
```

CEROW.H

```
//---------------------------------------------------------------------------

#ifndef CeroWH
#define CeroWH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Grids.hpp>
#include <fstream.h>
//---------------------------------------------------------------------------
class TCeroPiezaBox : public TForm
{
__published:    // IDE-managed Components
    TGroupBox *GroupBox1;
    TStringGrid *StringGrid1;
    TButton *Aceptar;
    TButton *Cancelar;
    void __fastcall AceptarClick(TObject *Sender);
    void __fastcall StringGrid1KeyPress(TObject *Sender, char &Key);
    void __fastcall StringGrid1SetEditText(TObject *Sender, int ACol,
      int ARow, const AnsiString Value);
    void __fastcall StringGrid1GetEditText(TObject *Sender, int ACol,
      int ARow, AnsiString &Value);
    void __fastcall StringGrid1SelectCell(TObject *Sender, int ACol,
      int ARow, bool &CanSelect);
    void __fastcall StringGrid1DrawCell(TObject *Sender, int ACol,
      int ARow, TRect &Rect, TGridDrawState State);
    void __fastcall StringGrid1GetEditMask(TObject *Sender, int ACol,
      int ARow, AnsiString &Value);
    void __fastcall CancelarClick(TObject *Sender);

private:  // User declarations
public:         // User declarations
    FILE *en;
    float Contenido[3][5];
    float PSOx[5];
    float PSOy[5];
    float PSOz[5];
    //float Matriz[5][3];
    int Fila, Col;
    __fastcall TCeroPiezaBox(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TCeroPiezaBox *CeroPiezaBox;
```

```
//---------------------------------------------------------------------------
#endif
```

HERRAMIENTA.CPP

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Herramienta.h"
#include "Demo1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TToolBox *ToolBox;
//---------------------------------------------------------------------------
__fastcall TToolBox::TToolBox(TComponent* Owner)
    : TForm(Owner)
{
char Valor[20];
    Fila =1;
    Col =1;

    if((en = fopen("TOOLS.MIL", "rt")) == NULL)
    {
        ShowMessage("\nImposible abrir el archivo.\n");
    }
    while(!feof(en))
    {
        fgets(Valor, 20,en);
        StringGrid1->Cells[1][Fila] = atof(Valor);
        Contenido[0][Fila-1] = atof(Valor);
        Fila++;
    }
    StringGrid1->Cells[0][0] = "      N°";
    StringGrid1->Cells[1][0] = "   Radio";
    StringGrid1->Cells[0][1] = "      1";
    StringGrid1->Cells[0][2] = "      2";
    StringGrid1->Cells[0][3] = "      3";
    StringGrid1->Cells[0][4] = "      4";
    StringGrid1->Cells[0][5] = "      5";
    StringGrid1->Cells[0][6] = "      6";
    StringGrid1->Cells[0][7] = "      7";
    StringGrid1->Cells[0][8] = "      8";
    StringGrid1->Cells[0][9] = "      9";

}
//---------------------------------------------------------------------------
void __fastcall TToolBox::CancelarClick(TObject *Sender)
{
    Close();
}
//---------------------------------------------------------------------------
void __fastcall TToolBox::Edit1KeyPress(TObject *Sender, char &Key)
```

```
{
    if ((Key < '0' || Key > '9') && (Key != 8) && (Key != '.'))
        Key = 0;
}
//---------------------------------------------------------------------------
void __fastcall TToolBox::AceptarClick(TObject *Sender)
{
    int i;
    char* Val;
    AnsiString Valor;
    float ValorNumerico;

    if((en = fopen("TOOLS.MIL", "wt")) == NULL)
    {
        ShowMessage("\nImposible abrir el archivo.\n");
    }
    for(i=0;i<=9;i++)
    {
        Form1->Tools[i] = Contenido[0][i];
    }
    fseek(en,0,0);
    //ftell(en);
    for(i=0;i<=9;i++)
    {
        Valor = AnsiString(Contenido[0][i]);
        while (Valor.Length() && Valor[Valor.Length()] == ' ')
        {
            Valor.SetLength(Valor.Length()-1);
        }
        fputs(Valor.c_str(),en);
        fputs("\n",en);
    }
    fclose(en);
    Close();
}
//---------------------------------------------------------------------------

void __fastcall TToolBox::StringGrid1GetEditMask(TObject *Sender, int ACol,
    int ARow, AnsiString &Value)
{
    //Value = "####";
}
//---------------------------------------------------------------------------

void __fastcall TToolBox::StringGrid1SetEditText(TObject *Sender, int ACol,
    int ARow, const AnsiString Value)
{
    AnsiString Valor;
    float ValorNumerico;

    Valor = Value;

    while (Valor.Length() && Valor[Valor.Length()] == ' ')
    {
        Valor.SetLength(Valor.Length()-1);
    }
```

```
    ValorNumerico = atof(Valor.c_str());
    Contenido[ACol-1][ARow-1] = ValorNumerico;
}
//---------------------------------------------------------------------------

void __fastcall TToolBox::StringGrid1GetEditText(TObject *Sender, int ACol,
    int ARow, AnsiString &Value)
{
    Value = AnsiString(Contenido[ACol-1][ARow-1]);
}
//---------------------------------------------------------------------------

void __fastcall TToolBox::StringGrid1SelectCell(TObject *Sender, int ACol,
    int ARow, bool &CanSelect)
{
    CanSelect = true;
}
//---------------------------------------------------------------------------

void __fastcall TToolBox::StringGrid1DrawCell(TObject *Sender, int ACol,
    int ARow, TRect &Rect, TGridDrawState State)
{
    if(ACol>1)
    StringGrid1->Cells[ACol-1][ARow-1] = AnsiString(Contenido[ACol-1][ARow-1]);
}
//---------------------------------------------------------------------------
```

HERRAMIENTA.H

```
//---------------------------------------------------------------------------

#ifndef HerramientaH
#define HerramientaH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
#include <Grids.hpp>
#include <Mask.hpp>
#include <fstream.h>
//---------------------------------------------------------------------------
class TToolBox : public TForm
{
__published:    // IDE-managed Components
    TLabel *Label1;
    TButton *Aceptar;
    TButton *Cancelar;
    TPanel *Panel1;
    TImage *Image1;
    TStringGrid *StringGrid1;
    void __fastcall CancelarClick(TObject *Sender);
```

```
    void __fastcall Edit1KeyPress(TObject *Sender, char &Key);
    void __fastcall AceptarClick(TObject *Sender);
    void __fastcall StringGrid1GetEditMask(TObject *Sender, int ACol,
      int ARow, AnsiString &Value);
    void __fastcall StringGrid1SetEditText(TObject *Sender, int ACol,
      int ARow, const AnsiString Value);
    void __fastcall StringGrid1GetEditText(TObject *Sender, int ACol,
      int ARow, AnsiString &Value);
    void __fastcall StringGrid1SelectCell(TObject *Sender, int ACol,
      int ARow, bool &CanSelect);
    void __fastcall StringGrid1DrawCell(TObject *Sender, int ACol,
      int ARow, TRect &Rect, TGridDrawState State);
private:  // User declarations
public:         // User declarations
    FILE *en;
    float Contenido[1][9];
    float Tools[9];
    int Fila, Col;
    // float Tool[9];
    __fastcall TToolBox(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TToolBox *ToolBox;
//---------------------------------------------------------------------------
#endif
```

MATERIAL.CPP

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Material.h"
#include "Demo1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)

#pragma resource "*.dfm"
TMaterialBox *MaterialBox;

//---------------------------------------------------------------------------
__fastcall TMaterialBox::TMaterialBox(TComponent* Owner)
    : TForm(Owner)
{

}
//---------------------------------------------------------------------------
void __fastcall TMaterialBox::AceptarClick(TObject *Sender)
{
 char buff[10];

 Form1->Escala[0] = atoi(Edit1->Text.c_str())/2;
 Form1->Escala[1] = atoi(Edit2->Text.c_str())/2;
```

```
Form1->Escala[2] = atoi(Edit3->Text.c_str())/2;

Close();
}
//---------------------------------------------------------------------------

void __fastcall TMaterialBox::CancelarClick(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------

void __fastcall TMaterialBox::EditKeyPress(TObject *Sender, char &Key)
{
    if ((Key < '0' || Key > '9') && Key != 8)
      Key = 0;
}
//---------------------------------------------------------------------------
```

MATERIAL.H

```
//---------------------------------------------------------------------------

#ifndef MaterialH
#define MaterialH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <SysUtils.hpp>
#include <ComCtrls.hpp>
#include <Mask.hpp>
//---------------------------------------------------------------------------
class TMaterialBox : public TForm
{
__published:    // IDE-managed Components
    TGroupBox *GroupBox1;
    TLabel *Label1;
    TEdit *Edit1;
    TEdit *Edit2;
    TEdit *Edit3;
    TLabel *Label2;
    TLabel *Label3;
    TButton *Aceptar;
    TButton *Cancelar;
    TLabel *Label4;
    TLabel *Label5;
    TLabel *Label6;
    void __fastcall AceptarClick(TObject *Sender);
    void __fastcall CancelarClick(TObject *Sender);
    void __fastcall EditKeyPress(TObject *Sender, char &Key);
```

```
private:  // User declarations
public:         // User declarations
        float Escala[3];
        __fastcall TMaterialBox(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TMaterialBox *MaterialBox;
//---------------------------------------------------------------------------
#endif
```

GLOSARIO

2D	Nomenclatura para indicar gráficos en dos dimensiones.
3D	Nomenclatura para indicar gráficos en tres dimensiones.
Avance	Es la rapidez con la cual avanza la pieza de trabajo hacia la herramienta y se mide en mm./min.
Builder	Lenguaje de programación en C++ para ambiente Windows. Desarrollado por Inprise Co.
CNC	Control Numérico Computarizado. Sistema que aplicado a máquinas – herramientas automatiza y controla todas las acciones de la máquina por medio de un programa compuesto de letras y números
OpenGL	Librería gráfica para la generación de programas en 2D y 3D. Desarrollada por Silicon Graphics.
PSO	Position Shift Offset. Indica la distancia entre dos orígenes. Se aplica exclusivamente para establecer el cero de pieza.

AUTOBIOGRAFÍA

Fernando Montemayor Ibarra nació en Monclova, Coahuila el 14 de marzo de 1971, sus padres son el Sr. Fernando Montemayor Barrera y la Sra. María de los Angeles Ibarra Preciado.

Realizó sus estudios superiores en la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León, obteniendo el título de Ingeniero Mecánico Electricista en julio de 1994.

Su tesis para la obtención de la Maestría en Ciencias de la Manufactura con especialidad en Automatización, lleva por nombre: "Simulador de Fresadora de Control Numérico Computarizado F3-CNC".

Se ha desarrollado como catedrático de la Facultad de Ingeniería Mecánica y Eléctrica en el área de máquinas – herramientas de CNC, desde 1994 a la fecha.