SIMULADOR DE MECANIZADOS EN CÓDIGO ISO-6983

PROYECTO FINAL DE CARRERA

ESCUELA TÉCNICA SUPERIOR DE INFORMÁTICA

UNIVERSIDAD POLITÉCNICA DE VALENCIA

AUTOR: JOSÉ MARTÍN CARBONELL

DIRECTOR: EDUARDO VENDRELL VIDAL

VALENCIA, SEPTIEMBRE DE 2011.

Título del proyecto:	Simulador de mecanizados en código ISO-6983
Titulación:	Ingeniería Superior Informática. (esp. informática industrial)
Autor:	José Martín Carbonell
Director:	Eduardo Vendrell Vidal

Dedicado.

A mi padre, por haberme hecho ser quien soy.

A mi madre, por enseñarme el camino hacia mis metas desde que era un niño.

Agradecimientos.

A mi hermana Sonia, mi fuente inagotable de inspiración, sabiduría y amor a la ciencia. Sin duda alguna, mi ejemplo a seguir.

A Eduardo Vendrell por su inestimable ayuda y confianza y por haberme brindado la oportunidad de desarrollar este proyecto final de carrera.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE FÓRMULAS

CAPÍTULO 1

INTRODUCCIÓN

1.1. Presentación del problema.

En la actualidad, los procesos de mecanizado se han convertido en una capacidad funcional indispensable en las cadenas de producción, siendo, por tanto, fundamental el conocimiento, análisis y perfeccionamiento de dichos procesos y los sistemas informáticos que los generan y controlan en el ámbito de la ingeniería informática.

Llevando los conceptos de máquina-herramienta y sistema informático a un único sistema, obtenemos las máquinas de *Control Numérico por Computador* (en adelante *CNC*), que asumen, actualmente, la práctica totalidad de los procesos de producción automatizado del planeta.

Debido al gran número de máquinas CNC existentes en el mercado, se consideró la necesidad de englobar la programación de dichas máquinas en un lenguaje de programación estándar: el código ISO-6983, más comúnmente conocido como *Código G.*

Este transcurso de sucesos nos lleva a una consideración: actualmente se produce en masa de forma automática, enlazando sistemas informáticos (que consumen recursos virtuales) y máquinas físicas (que consumen recursos reales), por tanto, y gracias a la capacidad de la tecnología actual, podemos hacer servir los sistemas informáticos para simular los resultados de los mecanizados reales, creando un paso previo que permite ahorrar recursos comprobando la adecuación del resultado a la necesidad de producción. Surgen de esta forma los simuladores de mecanizados.

Tras esta breve descripción del escenario, se puede considerar el planteamiento del problema según las siguientes motivaciones:

1. **Docencia:** considerando la importancia de los sistemas de mecanizado, es de vital importancia dotar a los futuros ingenieros de los conocimientos de programación ISO-6983 que, desde el bajo nivel, les permita llegar a diseñar y comprender el proceso de funcionamiento de soluciones avanzadas, tales como los paquetes CAD/CAM, o entender la interpretación del código CNC que realiza la práctica totalidad de los centros de mecanizado existentes.
2. **Multiplataforma:** el estado tecnológico actual nos hace ver la necesidad de considerar la ejecución de cualquier programa en el máximo de plataformas y sistemas operativos posibles, a fin de obtener un mayor alcance al usuario, sin restringir el entorno en el que éste se encuentre más cómodo.

3. **Tiempo real:** íntimamente ligado a la docencia, la ejecución y representación de instrucciones en tiempo real permite al usuario conocer, de forma directa y por separado, el resultado en el mundo real de cada una de ellas, facilitando la comprensión y permitiendo la edición directa en caso de no obtener el resultado deseado.

4. **Exportable a la máquina real:** considerando la unión existente entre el sistema y el proceso real, la simulación ha de poder convertirse en prototipo y producto, llevando a la práctica el trabajo realizado.

1.2. Objetivos del proyecto.

Considerando las motivaciones descritas en el apartado anterior, se acuerda la elaboración de un programa simulador de mecanizados, fuertemente ligado a la docencia, haciendo uso del conjunto de instrucciones del Código G.

A fin de poder concluir en su uso docente, se particulariza su funcionamiento a los procesos de mecanizado realizados con la máquina CNC *EMCO PC MILL 125*, con la cual se trabaja actualmente en los laboratorios de mecanizado del Departamento de Ingeniería de Sistemas y Automática (DISA) de la Universidad Politécnica de Valencia.

1.2.1. Objetivos finales.

1. Desarrollo de un programa docente, consistente en un simulador de mecanizados mediante lenguaje ISO-6983, donde el alumno pueda aprender a programar mecanizados en máquinas CNC, de forma interactiva y en tiempo real, mostrando el resultado del mecanizado en 3D y con capacidad de exportación a la máquina real, adaptando la funcionalidad del simulador a la máquina EMCO PC Mill 125.

2. Implantación del simulador como herramienta docente en los laboratorios de mecanizado del DISA y el resto de departamentos que se considere.

1.2.2. Objetivos intermedios.

1. El programa será **multiplataforma**, pudiendo ser ejecutado en múltiples tipos de arquitectura de 32 y 64 bits (Intel, AMD, PowerPC, SPARC), así como en múltiples sistemas operativos (Linux, MacOSX, Windows y Solaris).

2. La ejecución de instrucciones será en **tiempo real**, de tal forma que al insertar una determinada instrucción en Código G, se mostrará el resultado

inmediato de su efecto en la configuración de la máquina o en el mecanizado, siguiendo un código específico de colores.

3. Al tratarse de un lenguaje interpretado por el propio programa y en tiempo real, será necesaria **la creación de un analizador léxico-sintáctico de Código G**, capaz de asociar el conjunto de instrucciones correctas al conjunto de acciones adecuadas, y las incorrectas al manejador de errores.

4. Para simplificar la tarea docente, se creará una **interfaz gráfica de usuario fácil e intuitiva**, que motive al alumno a la interactuación y aprendizaje desde cero. Dicha interfaz permitirá comprobar, a simple vista, el estado de la máquina y el resultado en 3D del mecanizado.

5. El alumno ha de recibir una herramienta de auto-aprendizaje, con los contenidos necesarios accesibles desde el propio programa que le permitan aprender desde cero a programar mecanizados en código ISO-6983. Es por ello que se creará un **sistema de ayuda integrada interactivo**.

6. El programa será **multiventana**, pudiéndose ejecutar múltiples instancias independientes y simultáneas.

1.3. Descripción de la estructura de la memoria.

El presente documento se divide en 9 bloques diferenciados y 2 Anexos. Para facilitar la lectura y comprensión, a continuación se describe, de forma concisa, cada uno de los bloques y anexos contenidos en el documento.

1. **Introducción.**
   Breve descripción inicial de las causas que llevan al planteamiento del problema, objetivos finales e intermedios y beneficios del desarrollo del proyecto.

2. **Estado del arte.**
   Análisis de situación en materia de los procesos de mecanizado y análisis comparativo de distintas soluciones en relación al proyecto que nos ocupa.

3. **Las máquinas de control numérico.**
   Descripción de las máquinas de control numérico, su historia y la previsión de futuro, con un especial análisis de la máquina EMCO PC MILL 125.

4. **Descripción del programa.**
   Descripción detallada del desarrollo del proyecto: descripción del programa y sus características, planificación, herramientas utilizadas, estructura y diseño en alto y bajo nivel.

5. **Geometría analítica.**
   Descripción de los métodos matemáticos empleados para el cálculo de trayectorias y la representación gráfica de la simulación en el espacio tridimensional.

6. **Ejemplo de uso.**
   Desarrollo completo de un caso práctico de mecanizado haciendo uso del simulador.

7. **Conclusiones.**
   Análisis de resultados obtenidos en base a los objetivos y planificación establecida, problemas encontrados y proyección de futuro del proyecto.

8. **Referencias.**

9. **Bibliografía.**

Anexo A. Manual de usuario.
Manual detallado de las funcionalidades y usos del simulador.

Anexo B. El código ISO-6983 en mecaCNC.

CAPÍTULO 2

ESTADO DEL ARTE

En la actualidad, la mayor parte de los trabajos de producción profesionales en serie o a gran escala, son simulados en un paso previo a su fabricación, ya que la tecnología actual permite, además de comprobar el estado final de la pieza en base a un mecanizado, comprobar otros factores que podían modificar el resultado, tales como el material, el desgaste de la herramienta, la temperatura, las posibles colisiones, etc.

Las bondades de los sistemas simulados son evidentes: se observa previa a su fabricación cual será el resultado, pudiendo modificar parámetros en la programación o la herramienta, y ahorrando así los costes asociados a una producción con un resultado no deseado.

La aproximación conseguida por un simulador de mecanizados en la actualidad es increíblemente fiel a la realidad. Es por ello que, en el proceso de diseño de una nueva pieza, el simulador supone en la mayoría de los casos, la herramienta que controlará la máquina para obtener el propio resultado del mecanizado.

Son muchas las empresas que ofrecen este tipo de software, incluso las propias fabricantes de máquinas-herramienta y centros de mecanizado los ofrecen como parte del propio sistema, consiguiendo así una harmonía necesaria entre el diseño por computador, la simulación y fabricación asistida por computador.

Podemos concluir, por tanto, que los simuladores se han convertido en un elemento más, indispensable en el proceso de fabricación automatizado.

2.1. Sistemas CAD/CAM.

Previo al análisis del estado del arte en relación a los simuladores de mecanizados, cabe explicar el estado de la fabricación en base a sistemas CAD/CAM.

CAD *(Computer-Aided Design, Diseño Asistido por Computadora)* abarca el uso de un amplio rango de herramientas computacionales, orientadas principalmente a la ingeniería y la arquitectura, con el objetivo de diseñar elementos propios de sus actividades. Dichas herramientas permiten la especificación de elementos en 2D (mediante la generación de planos acotados en múltiples vistas) y 3D (permitiendo una visión tridimensional foto-realista del resultado).

CAM *(Computer-Aided Manufacture, Manufactura Asistida por Computador)* concibe el uso de computadores y tecnología avanzada en todas las fases de manufactura de un producto, reduciendo al mínimo la intervención del operario. Podemos establecer el ciclo de manufactura de un producto como aquellas fases que comprenden el marketing, la ingeniería, la gestión de la producción y la propia fabricación del producto.

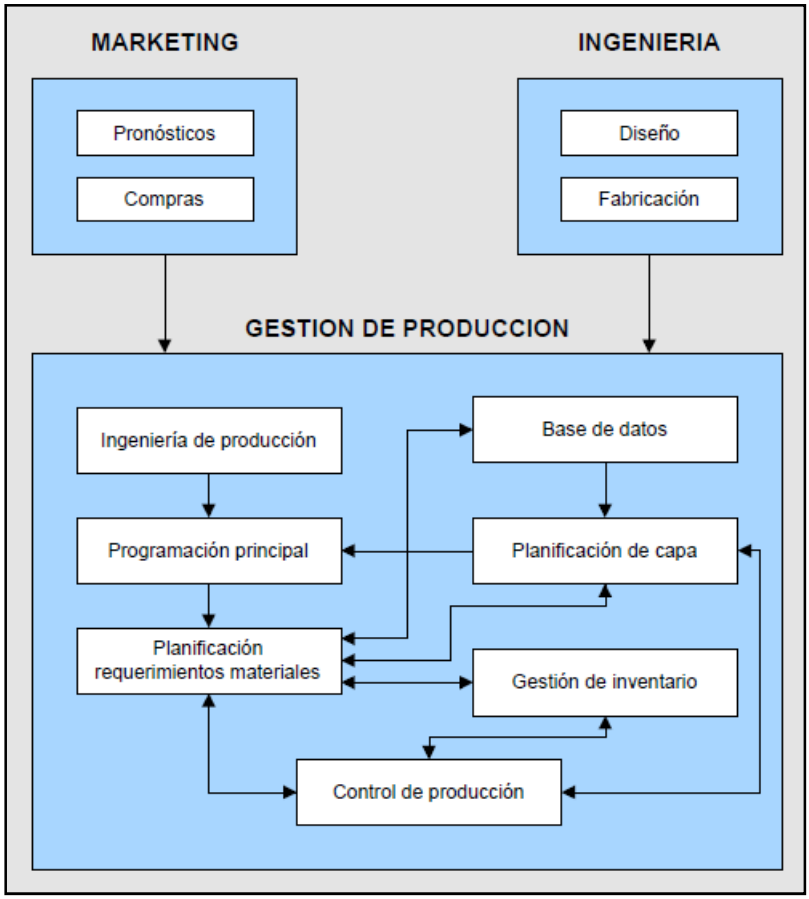


Figura 2.1: Diagrama del proceso de fabricación. [1]

En la actualidad, el proceso de fabricación une inexorablemente el CAD y el CAM, generándose un sistema común denominado CAD/CAM. Considerando estos conceptos iniciales, en el presente documento repararemos únicamente en el proceso de diseño y fabricación de una pieza, obviando el resto de elementos del proceso de manufactura.

La tendencia actual en el mercado de la producción enfatiza en la facilidad del proceso y la obtención de buenos resultados si la etapa de diseño se realiza haciendo uso de software de modelado 3D. Así pues, el proceso de diseño y fabricación de una pieza tiene, de manera generalista, a cumplir un proceso similar al siguiente:

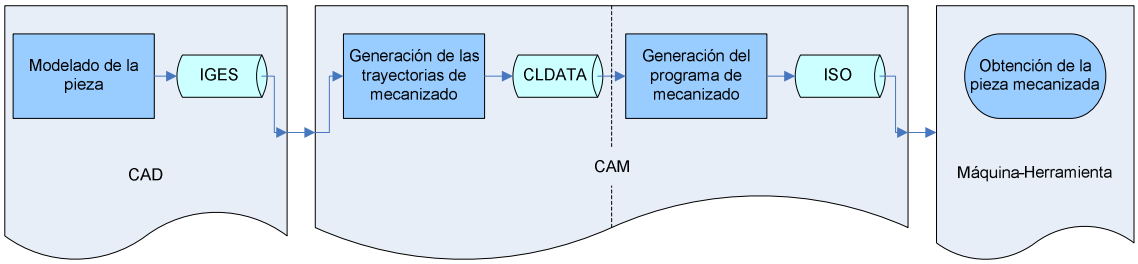


Figura 2.2: Ejemplo de proceso CAD/CAM. [1]

Como se puede comprobar en el gráfico de proceso de la *Figura 2.2*, a efectos de los sistemas de fabricación asistida por computador, se pueden identificar tres fases iniciales (además de la propia obtención de la pieza) que podrán ser llevadas a cabo mediante soluciones distintas para cada una de las fases o haciendo uso de una sola aplicación capaz de reproducir todo el proceso.

2.1.1. Modelado de la pieza.

Se diseña, en 2D o 3D (de mayor uso en el ámbito que nos ocupa), la pieza que se desea obtener. Las propiedades foto-realistas del software actual permitirán una representación fidedigna del resultado, pudiendo establecer los distintos materiales que compondrán la pieza previo al proceso de redenrización del modelo tridimensional.

Esta fase inicial del proceso es fundamental para la obtención de buenos resultados. La especificación de las herramientas a utilizar, el tipo de material y, evidentemente, las medidas que componen la forma han de estar bien definidas, tarea que se simplifica en su comprobación gracias a la simulación del modelo.

El final del proceso de modelado en el ejemplo anterior supone la obtención de la Especificación para Intercambio Inicial de Gráficos (IGES). Este tipo de datos definirán la geometría de la pieza, a partir de la cual se podrán realizar los análisis correspondientes en la siguiente etapa del proceso, apoyándose en la abstracción de la representación a favor de datos medibles y computables.

2.1.2. Generación de las trayectorias de mecanizado.

El análisis de los datos obtenidos de la etapa anterior permitirá definir las trayectorias de la máquina-herramienta que darán lugar al volumen previamente diseñado.

Durante esta parte del proceso, se pueden identificar ciertos problemas prematuros que den lugar a resultados no deseados, como es el caso de la detección de colisiones, donde el cálculo de la trayectoria para la creación de una determinada forma, lleva a un estado donde la geometría de la máquina y la pieza hacen imposible mecanizar.

Como resultado del proceso de generación de trayectorias en el ejemplo, se obtiene un fichero CLDATA (Cutter Location Data), que describe las ubicaciones o caminos de corte de la máquina.

2.1.3. Generación del programa de mecanizado.

Habiendo obtenido los datos de las trayectorias de corte de la fase anterior, el proceso se encuentra en disposición de traducir todos los datos obtenidos a secuencias de código interpretables por la máquina-herramienta o el centro de mecanizado.

Se producirá, por tanto, una conversión de los puntos que definen los caminos de corte a los parámetros de las instrucciones de interpolación del estándar ISO. El análisis y computación de todos los datos obtenidos hasta ahora en el proceso generarán, por tanto, la secuencia de instrucciones ISO que compondrán el programa de mecanizado que resulta en la pieza inicialmente diseñada.

En conclusión, para el proceso ejemplo se obtendrá el programa de mecanizado en código ISO-6983, y con ello podrán darse por concluidas las fases del proceso en las que ha sido necesario el uso de software adicional al propio centro de mecanizado, bien sea mediante el uso de un programa para cada fase o bien mediante el uso de una solución para todo el proceso.

2.1.4. Obtención de la pieza mecanizada.

Disponiendo del programa de mecanizado y habiendo podido simular su correcto resultado, únicamente restará el último paso: preparar la máquina herramienta para la realización del mecanizado.

Este último paso puede implicar en mayor medida la necesidad física del operario, ya que será en este momento cuando se deba emplazar y fijar el bruto, asignar las herramientas necesarias en los porta-brocas y realizar la carga del programa ISO obtenido, aunque este último, actualmente, también puede ser automatizado, permitiendo en forma remota que el propio programa se comunique con la máquina y cargue en ella el programa.

2.2. mecaCNC en el proceso CAD/CAM.

Como se ha detallado en el apartado anterior, en la actualidad, el proceso de fabricación trata de minimizar la acción del usuario, y, de manera análoga, consigue que los conocimientos requeridos por éste para la realización de un mecanizado sean menores.

Tal es así que el objeto de cualquier mecanizado, en base al proceso anterior, únicamente exige al usuario el manejo del software CAD asumiendo cierta destreza en el modelado, indicando los materiales y medidas del elemento que se va a manufacturar, dejando a la automatización el cálculo de las trayectorias y la generación del programa ISO.

A diferencia de este común denominador actual, mecaCNC persigue el objetivo de enseñar al usuario a programar en ISO, brindándole herramientas que, en tiempo real, le muestren el resultado de cada instrucción insertada, omitiendo la generación automática del código y dejando, a un segundo plano opcional, la carga del programa ISO resultante en la máquina-herramienta.

Por tanto, la funcionalidad de mecaCNC podrá incluirse dentro de la tercera fase del proceso CAD/CAM, aproximando el programa a dicho proceso tal y como muestra el siguiente diagrama:

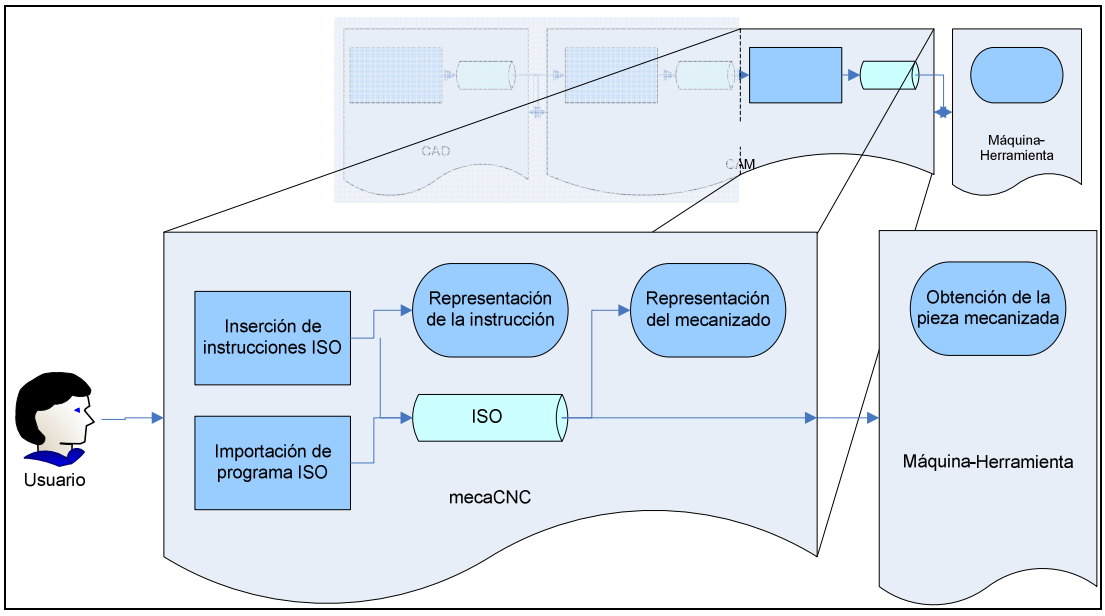


Figura 2.3: mecaCNC en el proceso CAD/CAM.

Este hecho nos lleva a una conclusión: el estado del arte en función del proceso de fabricación CAD/CAM no es comparable con el proyecto que nos ocupa, ya que este último únicamente irá orientado a una de las fase del proceso, si bien se ha considerado oportuno describir dicho proceso y analizar soluciones completas del proceso CAD/CAM, ya que, implícitamente, disponen de la funcionalidad necesaria para resolver la tercera fase del proceso, permitiendo simular gráficamente la ejecución de un programa ISO-6983.

Como se podrá comprobar en los próximos apartados, del resultado del análisis comparativo, se pueden extraer conclusiones muy satisfactorias que podrían ser tomadas en cuenta por las empresas desarrolladoras, tanto de software CAD/CAM capaz de manejar todo el proceso como simuladores CNC aplicables únicamente a la tercera fase.

2.3. Análisis comparativo.

Considerando que la producción en cadena asume gran parte del PIB de los países industrializados, se han hecho grandes avances en referencia al software simulador de procesos de mecanizado, convirtiéndose en un sector de gran competencia entre las empresas de desarrollo de aplicaciones orientadas a los procesos de producción.

Es por ello que, a día de hoy, podemos encontrar grandes soluciones complejas que permiten simular con gran precisión, no sólo el resultado final en base a la geometría de la pieza o el cálculo de trayectorias de la máquina, sino que se incluyen aspectos tales como la detección de colisiones, la temperatura del husillo y la pieza, la dureza del material, y una gran lista de características que permiten al usuario obtener una aproximación muy cercana a la realidad.

De entre todas las soluciones que se han encontrado en el mercado actual, se han seleccionado aquellas que, bien por su calidad y número de usuarios, o bien por contemplar alguna característica que las hace únicas, se han considerado dignas de análisis, pese a que cabe destacar la labor de la práctica totalidad de ellas por tratarse de un ámbito en la ingeniería de gran valor, complejidad e importancia.

2.3.1. Variables de comparación.

Dado el gran número de simuladores de mecanizados que podemos encontrar en la actualidad, se hace necesario que el objeto de este análisis comparativo se particularice en aquellas variables que definen características principales de mecaCNC.

Considerando la clara orientación docente, la limitación en el software adaptado a la máquina EMCO PC Mill 125 y la característica multiplataforma como factores de principal relevancia, las variables de comparación a evaluar serán:

1. **Multiplataforma:** múltiples arquitecturas y sistemas operativos capaces de ejecutar la solución (principalmente Windows, MacOS y Linux).
2. **Ejecución en tiempo real:** se considerará la capacidad del programa de representar el resultado de cada instrucción insertada.
3. **Adaptación a la máquina EMCO PC MILL 125:** se comprobará si se incluyen las características y funcionalidades del caso particular de esta máquina.

4. **Orientación docente:** cumplirán esta característica aquellos programas con una clara orientación docente, donde un alumno sin conocimiento alguno pueda aprender a programar procesos de mecanizado ISO.

5. **En idioma español:** se considerarán aquellos programas cuya interfaz esté en español, o bien sean multilenguaje incluyendo este idioma.

6. **Gratuidad:** se considerarán aquellas soluciones gratuitas o libres para la docencia.

7. **Open Source:** se considerarán aquellos programas de código abierto o ampliables por el propio usuario, donde alumnos de ingeniería informática puedan ampliar o mejorar las funcionalidades del producto.

2.3.2. Soluciones completas CAD/CAM incluidas en el análisis.

Como se ha comentado anteriormente, el software CAD/CAM actual supone programas de muy alta calidad y muy completos que se han convertido en una pieza indispensable en el proceso industrial.

Entre las múltiples funcionalidades de este tipo de software, una de las partes vitales que implementa sigue siendo la simulación del mecanizado, por tanto, se ha considerado adecuado incluir este tipo de soluciones completas en el análisis que nos ocupa, teniendo siempre presente que es inadecuado comparar el software CAD/CAM que dispone de funcionalidad de simulado de alta calidad, con un simulador CNC, que únicamente cubrirá una fase concreta de todas las que puede asumir el CAD/CAM.

De entre las múltiples soluciones encontradas, se ha creído conveniente destacar las siguientes:

2.3.2.1. CATIA.

Desarrollado por la empresa Dassault Systèmes en 1981, CATIA *(Computer-Aided Three Dimensional Interactive Application)* nace inicialmente para dar soporte a la industria aeronáutica.

CATIA provee una solución integrada para la fabricación de piezas mediante CAD/CAM, pudiendo, incluso, simular el mecanizado en plantas de fabricación virtuales. La versión actual del programa es la V6.

La gran capacidad y los múltiples ámbitos de aplicación de CATIA lo han convertido en un software referencia para la industria, conociendo su uso en sectores cómo:

- **Automóvil:** Empresas como Grupo VolksWagen, BMW, Renault, Peugeot, Daimler AG, Chrysler, Smart y Porsche.

- **Aeronáutica**: Boeing, Airbus, Bobardier, EMBRAER, Vought, AgustaWestland y las fuerzas aéreas Norteamericanas, Indias y Chinas.

- **Manufactura**: CATIA tiene una fuerte presencia en la industria de la manufactura, pudiendo destacar su uso en empresas como Schuler, Metso, Claas, Alstom Power y ABB Group entre muchas otras.

- **Arquitectura:** aunque en menor medida, CATIA también ha sido utilizado para diseñar estructuras tales como los *Curvilinear Buildings* del arquitecto Frank Gehry.

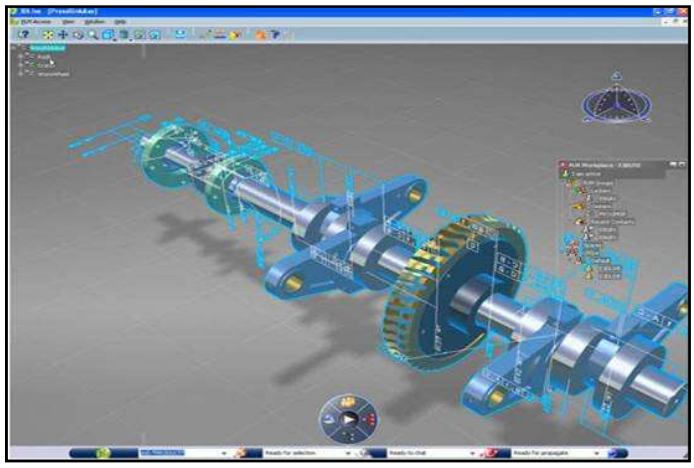


Figura 2.4: CATIA V6.

2.3.2.2. MasterCAM.

Creado por CNC Software Inc., una de las empresas pioneras en el desarrollo de software CAD/CAM en el año 1983, MasterCAM nace para satisfacer las necesidades fabricación asistida por computador mediante un, inicial, sistema CAD en 2D. Su última versión es MasterCAM X5 MU1.

Actualmente, MasterCAM se ha convertido en el software CAD/CAM de mayor uso y predilección entre usuarios del entorno de la manufactura, dada la calidad de su simulación tridimensional y los múltiples módulos y herramientas disponibles: Taladrado y fresado de hasta 7 ejes, torneado, electroerosión por hilo entre otros.

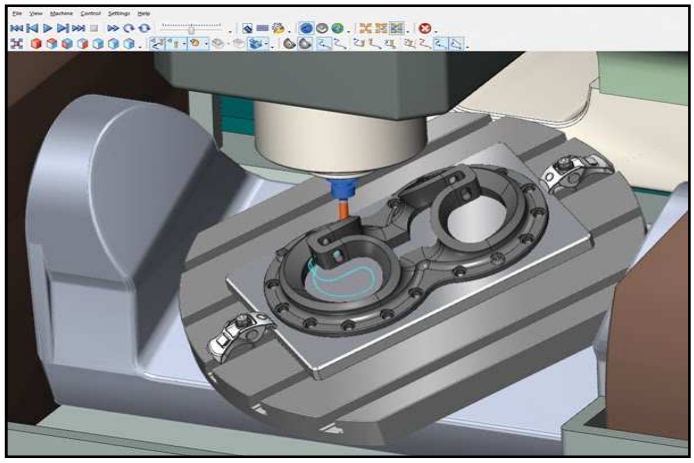


Figura 2.5: MasterCAM X5 MU1.

2.3.2.3. SolidCAM.

Desarrollado por la empresa SolidCam en 1984, SolidCAM ofrece múltiples módulos de software CAD/CAM orientados al fresado 2.5D y 3D, mecanizado de alta velocidad, fresado indexial multicara de 5.4 ejes, fresado simultáneo en 5 ejes, torneado, torneado-fresado de hasta 7 ejes y electroerosión por hilo. Su versión actual es SolidCAM 2011.

El uso de SolidCAM abarca sectores de la manufactura mecánica, electrónica, medicina, productos de consumo, diseño de máquinas, automoción y aeronáutica tanto en la creación de moldes y herramientas como en el prototipado rápido. SolidCAM dispone de más de 15000 implantaciones en la industria y la educación, a través de su red de venta en 50 países de todo el mundo.

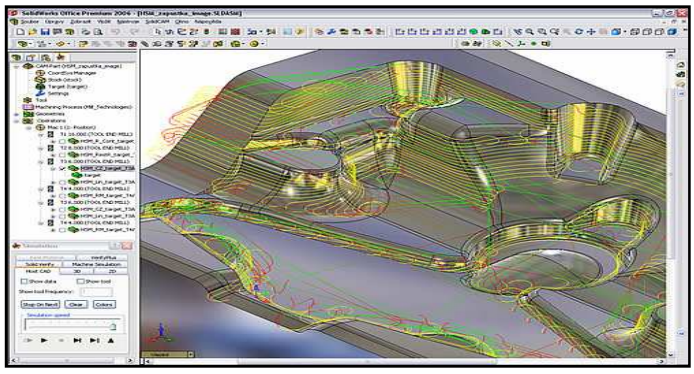


Figura 2.6: SolidCAM 2011.

2.3.3. Simuladores CNC incluidos en el análisis.

El segundo subconjunto de programas incluidos en el análisis está compuesto por programas simuladores CNC. Este tipo de software, al igual que mecaCNC, únicamente implementa funcionalidad para la tercera fase del proceso CAD/CAM, salvo en el caso de EMC, que incluye una función de importación de imágenes vectoriales 2D (importación del modelo CAD) y establecimiento manual de trayectorias de mecanizado, asumiendo de forma opcional parte de la fase dos del proceso CAD/CAM descrito anteriormente.

Como se verá a continuación, el software reviste una menor complejidad y presenta interfaces más simples, pero de igual modo, desempeñan correctamente de su misión como simuladores CNC.

2.3.3.1. CNCSimulator.

CNCSimulator consiste en un simulador de programas de mecanizado en código ISO de libre distribución. Permite la simulación multivista de operaciones de fresado/taladrado y torneado. CNCSimulator permite la conexión RS232 con máquinas-herramienta, pudiendo así enviar y recibir programas de mecanizados ISO de la máquina CNC.

Su interfaz y funcionalidades, evidentemente, son mucho más simples que las soluciones mencionadas anteriormente, pero cumple a la perfección su función como simulador de mecanizados, pudiendo obtener una aproximación tridimensional de la pieza resultante del proceso.

CNCSimulator nace en 2009, y actualmente distribuye la versión 4.53 del programa, revisada en el mes de abril de 2011. A modo de exigencia al usuario que descarga el programa, éste funcionará durante un periodo de un mes, tras el que el usuario deberá volver a la página web del programa y solicitar una nueva clave que le permitirá su uso durante un mes más.

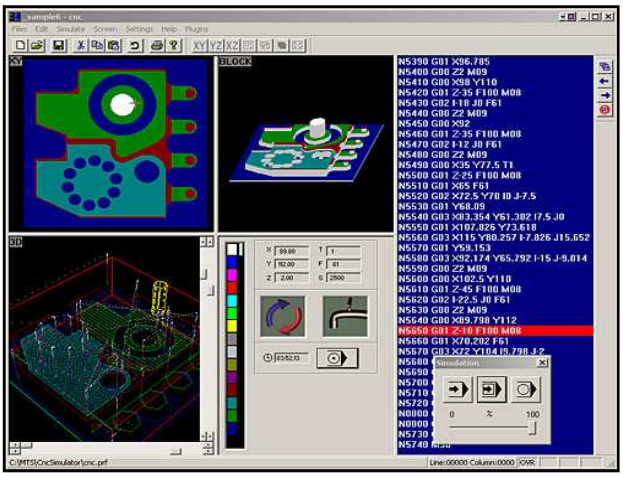


Figura 2.7: CNCSimulator v4.53f.

2.3.3.2. Swansoft CNC Simulator.

Desarrollado por Nanjing Swansoft Technology Company, Swansoft CNC Simulator en su última versión (6.70) ofrece un entorno de simulación y verificación de programas ISO en base a la máquina-herramienta utilizada, con una clara orientación a la docencia mediante las herramientas de consulta de su página web.

Para ello, según la máquina-herramienta o centro de mecanizado seleccionado, Swansoft CNC Simulator ofrece una representación tridimensional de la máquina, los paneles de control y los dispositivos de salida así como el resultado progresivo del mecanizado sobre la pieza. Actualmente, Swansoft CNC Simulator permite la simulación en máquinas-herramienta de FANUC, SINUERIK, MITSUBISHI, FAGOR, HAAS, PA, Romi, GSK, HNC, KND, DASEN, WA, GREAT, SANYING, RENHE, SKY y JNC.

Además, ofrece la posibilidad de la creación de programas en Código G en su propio editor, con operaciones de resaltado y detección y corrección de errores de código.

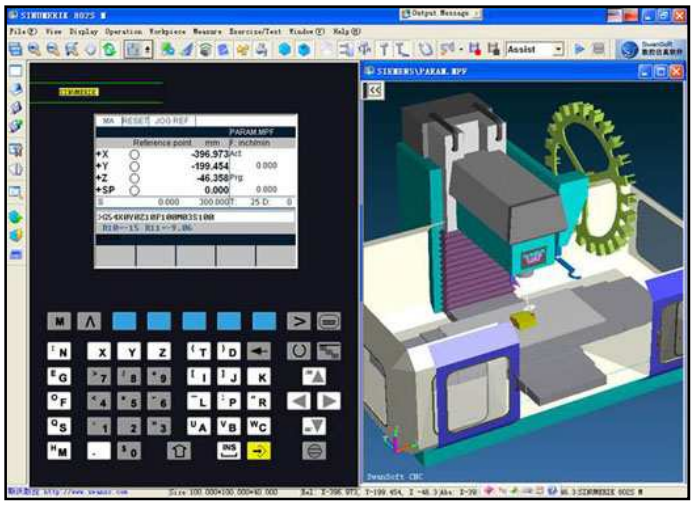


Figura 2.8: Swansoft CNC Simulator 6.70.

2.3.3.3. Enhanced Machine Controller (EMC).

EMC, desarrollado por la comunidad LinuxCNC, supone un programa de control de máquinas-herramienta mediante programación y simulación ISO para entornos UNIX. EMC es un software libre y Open Source bajo licencia GNU GPLv2.

EMC muestra, mediante interfaz gráfica, la simulación de trayectorias y mecanizado en base al código ISO importado o generado desde aplicaciones CAD compatibles, además de un intérprete de Código G. Soporta la simulación y control

en tiempo real de dispositivos de hasta 9 ejes con una gran variedad de interfaces de conexión.

Dado su carácter Open Source, EMC abarca una gran comunidad de desarrolladores, ya que la naturaleza del software hace idóneo su uso para máquinas-herramienta CNC de fabricación casera.

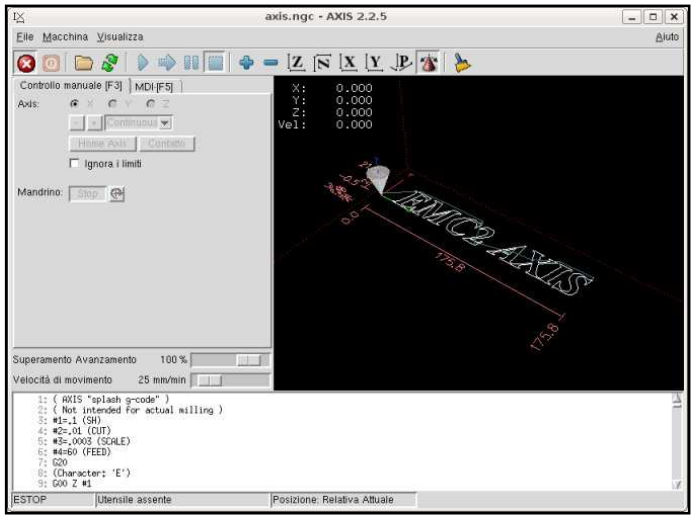


Figura 2.9: LinuxCNC EMC2.

2.3.3.4. EMCO WinCAM.

Desarrollado por EMCO, WinCAM ofrece un entorno de desarrollo CAD/CAM en 2D que permite generar el programa ISO en base al diseño en plano de la pieza a mecanizar.

WinCAM, en su última versión 2.43, permite la conexión directa con la máquina-herramienta a través de un puerto RS232. La simulación permite la obtención tridimensional del resultado del mecanizado, detectando posibles situaciones erróneas mediante control de trayectoria.

WinCAM está concebido y perfectamente adaptado para su uso con el centro de mecanizado EMCO PC Mill 125, siendo actualmente el software utilizado en el laboratorio de mecanizado del DISA en la Universidad Politécnica de Valencia para aproximar al alumno a los sistemas de mecanizado mediante software CAD/CAM.

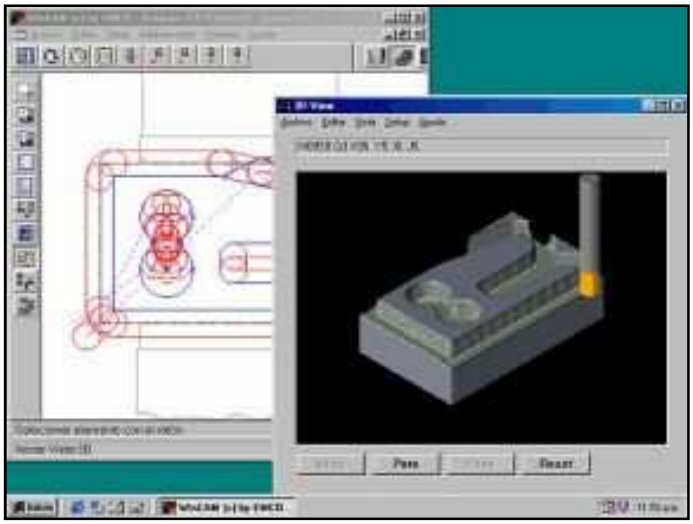


Figura 2.10: EMCO WinCAM 2.43.

2.4. Resultados del análisis.

Tal y como se concluyó en el apartado 2.3 del presente documento, dada la multitud de programas existentes, tanto simuladores CNC como CAD/CAM con capacidades de simulación, se convierte en una difícil tarea la realización de un análisis completo, máxime si se considera el carácter propietario de la mayoría de las soluciones, cuestión que impide la comprobación de toda la funcionalidad ofrecida.

Es por ello que los resultados de este análisis no persiguen voluntad alguna de comparación entre las distintas soluciones y mecaCNC, puesto que muchas de ellas se han quedado fuera del objeto de análisis y, evidentemente, un software completo CAD/CAM con función de simulación reviste una mayor complejidad y calidad que un programa cuya única funcionalidad sea la simulación CNC.

Programa	Multiplataforma			Tiempo real	Adaptado a EMCO PC Mill 125	Orientación docente	Español	Gratuito	Open Source
	Windows	MacOS X	Linux						
mecaCNC	✓	✓	✓	✓	✓	✓	✓	✓	✓
CATIA v6	✓	×	×	×	×	×	×	×	×
MasterCAM X5	✓	×	×	×	×	×	✓	×	×
SolidCAM 2011	✓	×	×	×	×	×	✓	×	×
CNCSimulator 4.53f	✓	×	×	×	×	×	×	✓	×
Swansoft CNC Simulator 6.70	✓	×	×	×	×	✓	✓	×	×
EMC2	×	×	✓	×	×	×	×	✓	✓
WinCAM 2.43	✓	×	×	×	✓	✓	×	×	×

Tabla 2.1: Resultados del análisis comparativo en función de las variables consideradas.

De los resultados del análisis, tanto de los programas muestreados como otros analizados que no han sido incluidos en el estudio, se puede concluir que **no existe ningún software multiplataforma (que permita su ejecución en Windows, MacOS X y Linux), CAD/CAM con función de simulación o simulador CNC, Open Source y gratuito, capaz de mostrar, en tiempo real, el resultado de la inserción de instrucciones ISO-6983 que componen un programa de mecanizado.**

Esta afirmación irá sujeta a tres realidades constatables:

- La estandarización del proceso de fabricación en base al uso de software CAD/CAM actual, hace innecesaria la inserción de código ISO, ya que será el propio software el encargado de generar dicho código en base al modelado de la figura, si bien es cierto que **mecaCNC persigue el objetivo de enseñar a programar mecanizados desde bajo nivel**, tarea necesaria en ingeniería para haber podido llegar al software CAD/CAM actual.

- Existen simuladores CNC capaz de ejecutar el programa ISO paso por paso, pero en ningún caso se ha encontrado un simulador capaz de ejecutar y simular las instrucciones en tiempo real. Dada la gran cantidad de software disponible, en caso de que dicha funcionalidad existiera, no podría sumarse a las condiciones multiplataforma, gratuito y Open Source.

- La adaptación del software a la máquina EMCO PC Mill 125 queda relegada a un segundo plano, ya que la máquina actualmente no se fabrica, si bien se ha considerado esta característica por tratarse de un centro de mecanizado de gran

implantación en el sistema docente, incluyendo el laboratorio al que va destinado el uso de mecaCNC en la Universidad Politécnica de Valencia.

CAPÍTULO 3

LAS MÁQUINAS DE CONTROL NUMÉRICO

3.1. Evolución histórica de las máquinas de control numérico.

Se tiene constancia que, desde civilizaciones ancestrales como la egipcia, el ser humano, motivado por la necesidad de producir ciertos objetos de la manera más fácil y efectiva posible, ha hecho uso de máquinas básicas de producción, como es el caso del torno egipcio, del que supone su uso desde el **año 1000 a.C.** Gracias a un movimiento rotatorio impulsado por el propio operario de la máquina, se conseguía producir objetos tales como los ejes o radios de los carruajes con un menor esfuerzo, en menor tiempo y con una mejor calidad y homogeneidad.

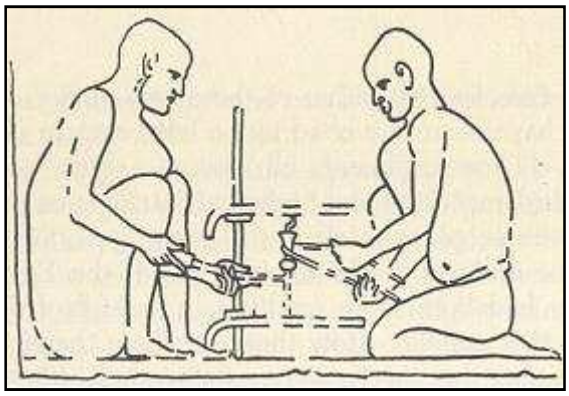


Figura 3.1: Pintura egipcia del siglo III a.C. de una operación de torneado.

Analizando la historia de las civilizaciones, la expansión de los pueblos conlleva la exportación del uso de las máquinas-herramienta a distintos puntos del planeta. En Europa, **a finales de la Edad Media**, el uso del torno a pedal con arco era una realidad frecuente, donde un solo operario podía realizar operaciones de torneado impulsando el movimiento rotatorio con el pié, pudiendo así trabajar la pieza con ambas manos.

Figura 3.2: Grabado de 1435 de un torno accionado por arco

La llegada de la máquina de vapor a finales del siglo XVII, convierte la producción en serie en una realidad. Incorporando el motor de combustión interna que propició la Revolución Industrial, las máquinas-herramienta se convierten en imprescindibles para la producción a gran escala, reduciendo el número de operarios necesarios y generando piezas en tiempo mucho menor. Se crean nuevas máquinas-herramienta tales como la máquina de cilindrado, la mandrinadora y se perfecciona el torno, llegando a fabricarse tornos con cambios de velocidades.

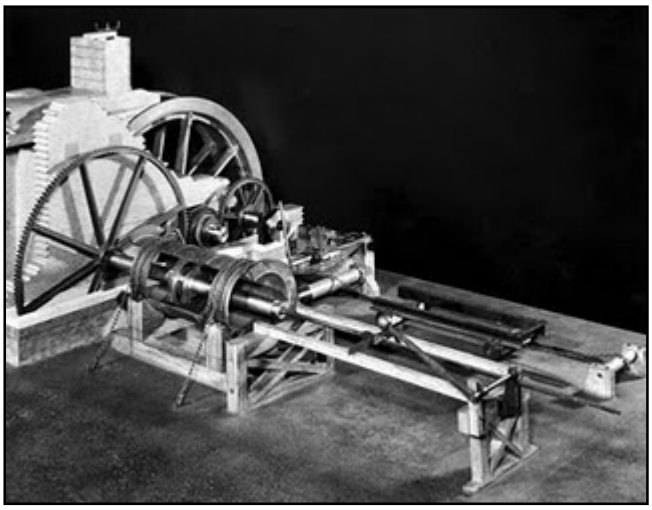


Figura 3.3: Mandrinadora de J. Wilkinson fabricada en 1775.

El descubrimiento y producción de materiales abrasivos artificiales durante el siglo XIX, conlleva la creación de un nuevo tipo de máquina-herramienta: las muelas o amoladoras, que permitían devastar, pulir o afilar metales en un menor tiempo y con una mayor precisión.

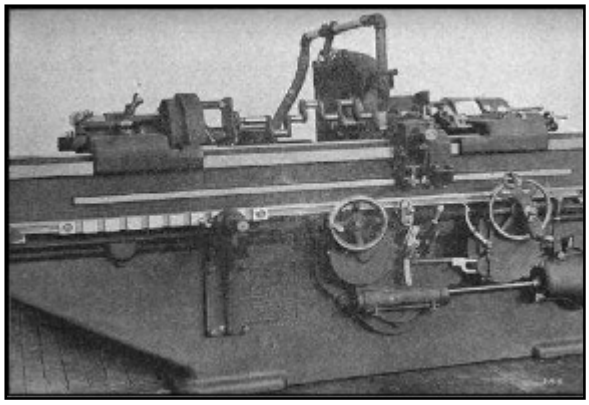
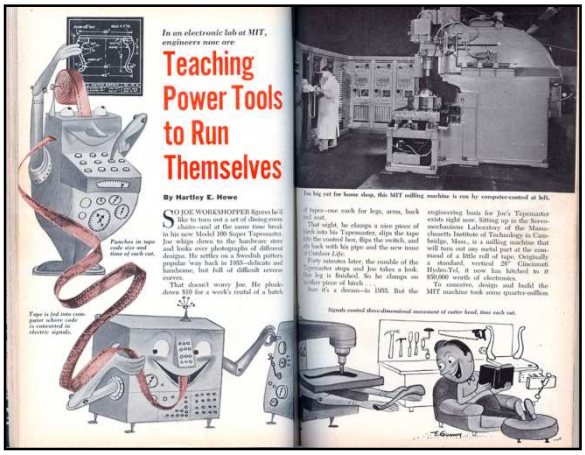


Figura 3.4: Rectificadora de Charles H. Norton construida en 1896.

Tras esta breve reseña de la historia de las máquinas-herramienta, llegamos al **siglo XX**, etapa decisiva en la creación de las *máquinas de Control Numérico por Computador*.

La llegada de las computadoras, automáticas y programables, inspira a John Thoren Parsons a la invención del control numérico en el año 1946. Parsons adaptó a una fresadora una máquina de contar IBM que, mediante el uso de tarjetas perforadas, era capaz de realizar los cálculos y medidas que facilitaban la operación de fresado sobre tres ejes. Por primera vez en la historia, se realiza un proceso automático de fabricación en serie por control numérico. La llegada del control numérico supone, para muchos, el inicio de la II Revolución Industrial.

La invención del control numérico deriva en la investigación y desarrollo de prototipos y experimentos para la producción a gran escala, a pesar de que el elevado coste de las máquinas, su gran volumen y su limitada precisión en relación a las necesidades de producción, suponían un obstáculo para la implementación generalizada. En 1955, tras comprobar su potencial en una feria de Chicago, la Fuerza Aérea Norteamericana adquiere 170 máquinas-herramienta por control numérico, gobernadas por tarjetas y cintas perforadas, que suponen la primera planta de producción automatizada de la historia.



In an electronic lab at MIT, engineers now are

Teaching Power Tools to Run Themselves

By Hartley E. Howe

Figura 3.5: Artículo de la revista Popular Science de 1955 donde figura la primera máquina de control numérico.

La investigación y esfuerzos para la creación de nuevas máquinas CNC crece exponencialmente de forma análoga a los sistemas informáticos, hasta llevarnos a la época contemporánea a partir de los años 60, donde la micro-computación resuelve el problema del volumen y coste de creación de las máquinas-herramienta, naciendo finalmente el concepto de máquina de control numérico por computador.

3.2. Las máquinas de control numérico por computador.

Como se trataba en el apartado anterior, la creación de la micro-electrónica en los años 60 implica un cambio radical en la concepción y fabricación de máquinas-herramienta, ya que, gracias a la integración de computadoras de reducidas dimensiones, cualquier máquina-herramienta era capaz de producir con una alta precisión y sin necesidad de operario.

La reducción de costes y volumen, conlleva la implantación del sistema de producción en serie por máquina-herramienta de control numérico por computador. Las fresadoras, taladros, mandrinadoras, etc. pasan a ser automatizadas gracias a la integración de micro-computadoras.

3.2.1. Estructura informática de una máquina-herramienta.

1. **CPU:** estructura informática dotada de un micro-procesador encargado de realizar los cálculos de posición y desplazamiento de los ejes y accionar las distintas funcionalidades y modos de la máquina.

2. **Periféricos de entrada:** son los elementos que permiten suministrar la información requerida a la CPU. Entre los dispositivos característicos, destacan el teclado, los sensores ópticos y posicionadores y la propia conexión a un ordenador externo desde el que se dirige el flujo de información a la máquina CNC.

3. **Unidades de almacenamiento:** sistemas de memoria de datos que puede residir en la propia máquina-herramienta o en un ordenador externo conectado a ella.

4. **Periféricos de salida:** elementos que reciben la información de respuesta de la máquina-herramienta. Podemos englobar en esta categoría tanto los periféricos destinados a informar sobre el estado de la máquina y el proceso (por ejemplo el monitor, integrado o en ordenador externo) como los distintos dispositivos de control de movimiento de los ejes y el husillo.

5. **PLC's (Controlador Lógico Programable):** autómatas programables que sirven de enlace entre la CPU y los distintos elementos que transfieren el movimiento, tales como motores.

3.2.2. Tipos de máquinas-herramienta CNC.

Entre las múltiples máquinas-herramienta existentes, podemos realizar dos clasificaciones:

3.2.2.1. Según su función:

Máquina	Función	Imagen
Tornos	Corte de viruta por revolución de la pieza	
Fresadoras	Arranque de viruta por aproximación de pieza rotativa de varios filos (fresa).	
Taladradoras	Producción de agujeros cilíndricos por aproximación de pieza rotativa helicoidal (broca).	
Roscadoras	Producción de roscas por laminado o arranque de viruta.	
Muelas	Pulido, abrillantado, afilado y mejora en el acabado por aproximación de material abrasivo rotativo.	
Mortajadoras	Limado por arranque de viruta en base a movimientos rectilíneos alternativos.	
Cepilladoras	Alisado de superficies (principalmente madera) por fricción con cuchillas.	

Rectificadoras	Desgaste por abrasión de piezas metálicas.	
Prensas	Troquelación y moldeado de piezas metálicas por presión.	
Cizallas	Corte vertical por presión de la cuchilla sobre el material.	
Cortadoras	Corte de material (principalmente metal) por aumento de la temperatura localizada en un punto o por chorro de agua a presión.	
Mandrinadoras	Creación de agujeros por aumento de diámetro en base al corte de la pieza rotatoria.	
Puntedoras	Soldadura por puntos de metales.	

Tabla 3.1: Máquinas-herramienta CNC según su función.

3.2.2.2. Según el control numérico:

1. **Punto a punto:** el control determina, antes de iniciarse el movimiento, el camino a recorrer y se posiciona sin importar la trayectoria de forma rápida y precisa.

2. **Contorneo:** El control determina, para la posición final, la trayectoria de la interpolación en cada instante, requiriendo una total sincronización en el movimiento de los ejes.

3.3. Los centros de mecanizado.

Las capacidades de la nueva tecnología propician, hacia los años 80, el nacimiento de un nuevo concepto de máquina-herramienta que aúna las funcionalidades de distintas máquinas-herramienta independientes: el centro de mecanizado.

Según la definición de L. Muñoz, *"un centro de mecanizado es una máquina altamente automatizada capaz de realizar múltiples operaciones de maquinado en una instalación bajo CNC (control numérico computarizado) con la mínima intervención humana. Las operaciones típicas son aquellas que usan herramientas de corte rotatorio como cortadores y brocas. Comparando este sistema de mecanizado con los sistemas tradicionales, se destacan la velocidad de producción como ventaja y los altos costos como desventaja."* [2]

Un centro de mecanizado ofrece la flexibilidad y versatilidad de realizar distintas operaciones sobre una misma pieza, incluyendo la capacidad de cambio automático de herramienta. Son reconfigurables y, según se ha ido avanzando en su fabricación, su precisión y acabado superan en gran medida a las capacidades del ser humano, con una producción uniforme en tiempos muy bajos.

Gracias a la computarización de los centros de mecanizado, se ha conseguido simplificar la propia máquina delegando en el propio sistema informático ciertas capacidades, y dotando así a la máquina de un mejor control y precisión sobre el posicionamiento como consecuencia del movimiento simultáneo de los ejes.

En el ámbito académico, por tanto, tiene sentido el uso de centros de mecanizado capaces de realizar distintas operaciones sobre una misma pieza, aproximando al usuario a la realidad de un mecanizado completo.

3.4. El centro de mecanizado EMCO PC Mill 125.

El proyecto que nos ocupa consiste en la simulación del mecanizado particularizando en las características y funcionalidades del centro de mecanizado EMCO PC Mill 125.

Figura 3.6: Centro de mecanizado EMCO PC Mill 125.

Este centro de mecanizado está claramente orientado al entrenamiento docente, pero sus funcionalidades y la programación de los mecanizados, no distan en gran medida de las correspondientes a un centro de mecanizado industrial, si bien es cierto que la fabricación de este modelo de centro de mecanizado se realizó aproximadamente en el año 2000, y en la actualidad existen sistemas más avanzados, pero siempre conservando la misma base y principios de funcionamiento.

Entre las características principales del EMCO PC Mill 125, destacan:

1. Movimientos paralelos en 3 ejes (X, Y, Z), con área de trabajo de 185, 125 y 200mm respectivamente. Los motores de paso son de alta resolución y dispone de husillos de bolas pre-cargados para los 3 ejes.
2. Carrusel con almacén de 10 herramientas y lógica direccional.
3. Motor principal asíncrono de potencia 0.6 - 0.7 kW, con control de velocidad y gama entre 150 y 5000 revoluciones por minuto.
4. Sus dimensiones son 1730 x 875 x 1900 mm y pesa 560 Kg.
5. El control numérico está basado en un PC montado y cableado a la máquina, con sistema de almacenamiento y dispositivos de entrada/salida. Dispone de un panel de mandos integrado y una pantalla de 14”.

Como la práctica totalidad de los centros de mecanizado y máquinas-herramienta CNC, el EMCO PC Mill 125 permite la inserción de programas en Código G, pudiendo ser directamente importados desde el sistema de almacenaje del PC, convirtiéndo, por tanto, el proyecto que nos ocupa, en un posible paso previo al mecanizado real en la máquina.

3.5. Previsión de futuro.

Durante toda la historia de las máquinas-herramienta, se ha tratado el mecanizado como el proceso por excelencia para la producción en serie de piezas uniformes, de gran tirada y en bajos tiempos, ya que la producción trataba de satisfacer la gran demanda existente.

La realidad actual es distinta. El avance exponencial tecnológico está obligando a adaptar los diseños a un mercado en constante cambio, donde la flexibilidad y escalabilidad de las máquinas es fundamental para poder generar productos nuevos sin cambios sustanciales en las plantas de producción cada cortos periodos de tiempo.

Por otra parte, la dependencia de la mano humana en el proceso de producción está siendo prácticamente innecesaria. Las células de mecanizado consiguen la realización completa de la pieza resultado en una sola sujeción, mediante el uso de sensores y robótica avanzada, no se contempla únicamente el mecanizado de la pieza, sino el almacenaje, desplazamiento de la pieza al espacio de trabajo, acabado, empaquetado, etc. consiguiendo por tanto una mayor rentabilidad de la planta.

Respecto a los diseños, la investigación oscila en las necesidades que han sido imperantes a lo largo de la historia: máquinas más rápidas, más precisas, más rígidas y de menor coste. La investigación y avance en estructuras cinemáticas paralelas de tipo hexápodo, a pesar de considerarse todavía difíciles de desarrollar debido a sus grados de libertad, son un hecho, y reúnen características suficientes para convertirse en un modelo común cuando se superen las barreras que todavía impiden su implantación masiva en el mercado.

Figura 3.7: Fresadora de estructura cinemática paralela Seyanka

En referencia al alcance de la mecanización, la electrónica y la robótica, estamos asistiendo a una etapa sin precedentes que puede cambiar la estructura social y atendiendo al ritmo vertiginoso con un avance exponencial en tecnología, se podría conjeturar, aún a riesgo de parecer una osadía, que el trabajo humano está dejando de ser progresivamente necesario.

Un posible ejemplo que abale esta afirmación consiste en el proyecto *Contour Crafting* (www.contourcrafting.org), desarrollado por el Dr. Behrokh Khoshnevis, que consiste en un proceso de fabricación de edificios por capas de contorno haciendo uso de una máquina-herramienta de gran escala. El uso de esta máquina derivaría en un ahorro de energía y recursos, tanto materiales como humanos, y se considera que en un futuro próximo, *Contour Crafting* podría ser capaz de construir una casa de 185 metros cuadrados en tan solo un día.

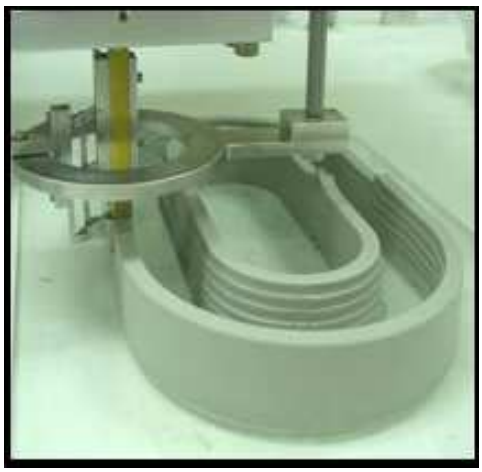
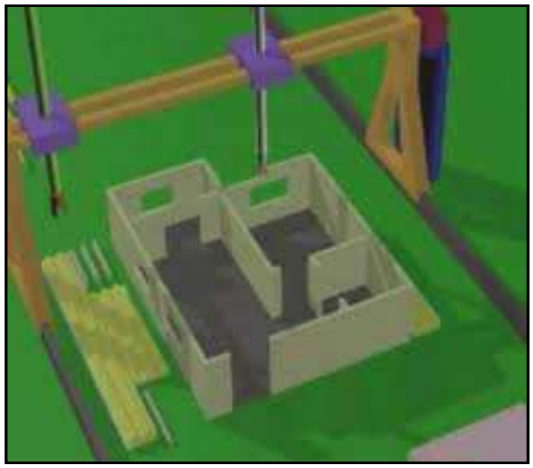



Figura 3.8 (izquierda): Prototipo de Contour Crafting en funcionamiento.
Figura 3.9 (derecha): Animación del uso de Contour Crafting en la edificación.

Otro de los ejemplos que actualmente supone una realidad, aproxima la mecanización al alcance de los hogares. El proyecto *Reprap* (www.reprap.org) consiste en un Open Source Hardware que documenta la construcción de una máquina autómata capaz de construir piezas tridimensionales con hilo plástico fundido. La característica que convierte a esta impresora 3D en excepcional consiste en su auto-reproducción: gran parte de las piezas de la máquina están hechas de plástico, por tanto, al disponer de una máquina *Reprap*, se puede imprimir dichas piezas que forman otra máquina idéntica. Este tipo de proyecto impulsa a pensar que la realidad de los sistemas de mecanizado en los hogares puede estar relativamente próxima.

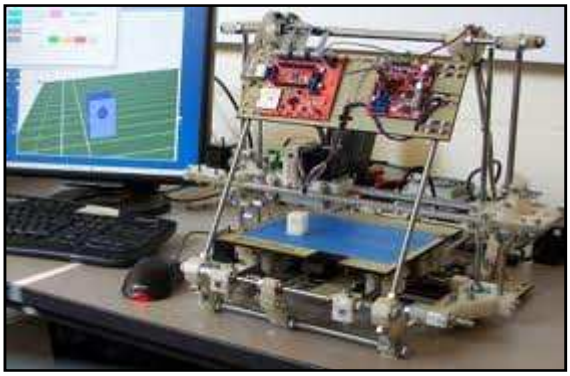


Figura 3.10: Máquina Reprap.

CAPÍTULO 4

DESCRIPCIÓN DEL DESARROLLO

4.1. ¿Qué es mecaCNC?

mecaCNC consiste en un programa simulador de mecanizados en entorno de ventanas para el ámbito docente, que emula las funcionalidades y características del centro de mecanizado EMCO PC Mill 125, permitiendo la inserción de instrucciones en Código G que compondrán el mecanizado, y mostrando su resultado tridimensional en tiempo real.

Dada su aplicación docente, mecaCNC dispone de un completo sistema de ayuda integrado que permitirá al alumno aproximarse a la lógica de la programación de mecanizados desde cero, además de un sistema de acceso directo a documentación en PDF que facilitará el aprendizaje previo de los conceptos y sistemas CAD/CAM, la programación CNC y el uso real del centro de mecanizado EMCO PC Mill 125.

Además, mecaCNC permite la exportación del Código G insertado en el simulador a la máquina real, pudiendo suponer así un paso previo a la mecanización de piezas reales, conociendo de antemano el resultado tridimensional del mecanizado.

4.1.1. Características principales.

1. **Multiplataforma:** mecaCNC puede ser ejecutado en entornos Windows (i586 y x64), MacOS (PowerPC e Intel), Linux (i586 y x64) y Solaris (i586, Sparc y SparcV9).
2. **Ejecución en tiempo real:** a diferencia de otros simuladores, mecaCNC mostrará el resultado de cada instrucción de forma instantánea al ser insertada.
3. **Multiventana:** mecaCNC permite abrir múltiples instancias del programa, pudiendo así realizar comparativas en el diseño de mecanizados similares.
4. **Representación tridimensional:** la ventana de representación del mecanizado permite variar la posición de la cámara, ofreciendo, además, la posibilidad de seleccionar vistas predefinidas.
5. **Exportación e importación de Código G:** mecaCNC permite exportar el Código G insertado a archivos interpretables por la máquina real. Así mismo, mecaCNC permitirá importar dicho código de archivos de texto generados a mano o por otros simuladores, siempre que respeten la estructura e instrucciones de la máquina EMCO PC Mill 125.

6. **Edición en tiempo real:** mecaCNC permite la edición del código que compone el mecanizado, observando los resultados de dichos cambios de manera inmediata.

7. **Sistema de auto-aprendizaje:** el simulador integra un completo sistema de ayuda por bloques, índice y búsqueda de términos que incluye la estructura e instrucciones de un programa en Código G. El sistema de ayuda permite imprimir de manera directa los elementos que se considere. Además, se ofrece acceso directo a los usuarios a distintos documentos de interés en PDF.

8. **Open Source:** el código del programa quedará a disposición de todo aquel que quiera hacer uso, revisión o ampliación.

4.2. Cuestiones preliminares.

El presente apartado detalla ciertas cuestiones relevantes previas al desarrollo del proyecto que influirán a posteriori en la metodología de la programación y la estimación de costes.

4.2.1. Lenguajes de programación utilizados.

Para la realización del programa se han utilizado diversos lenguajes de programación en función de su aplicación al entorno:

4.2.1.1. Java.

Comprende el grueso de la programación del entorno de ventanas, eventos, entrada/salida, funciones matemáticas y auxiliares.

Java es un lenguaje de alto nivel orientado a objetos, derivado de C y C++. Java suprime herramientas de bajo nivel propias de C que suelen conducir a errores, presentando una sintaxis mucho más flexible y cómoda para el programador.

La implementación original del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por *Sun Microsistems* en el año 1995. El control de las especificaciones, desarrollo y evolución del lenguaje fueron asumidas por *Sun* a través del *Java Comunity Process*, quienes creyeron conveniente, entre diciembre de 2006 y mayo de 2007, liberar la práctica totalidad del código en licencia GNU GPL, manteniendo en código cerrado la biblioteca de clases necesaria para la ejecución de programas desarrollados en Java. Finalmente, *Oracle* compró en abril de 2009 *Sun Microsistems*, asumiendo el

control de la tecnología Java, Solaris y MySQL (originalmente creadas o adquiridas por Sun). Esta acción creó reticencias en la comunidad de desarrolladores de software libre, dada la posible mercantilización por parte de Oracle de sistemas tan populares y utilizados actualmente en la comunidad como lo son Java o MySQL.

Inicialmente, Sun determina la distribución de Java en base a plataformas específicas que puedan cubrir las necesidades de desarrollo en base al ámbito de uso, surgiendo, por tanto las plataformas J2SE (Java 2 Standard Edition) orientada a un desarrollo generalizado, J2EE (Java 2 Enterprise Edition) orientada al desarrollo empresarial y J2ME (Java 2 Micro Edition) orientada a la creación de programas en dispositivos de bajas capacidades. A partir de su versión 6, Sun cambia la nomenclatura de sus plataformas, pudiendo obtenerse así la versión que ha sido utilizada para el desarrollo de este proyecto: Java SE 6.0 (Java Standard Edition 6.0). La plataforma Java incluye:

- El lenguaje de programación en sí mismo.
- La Máquina Virtual de Java (JVM) que permite la ejecución multiplataforma del *Bytecode* generado por el compilador.
- El API consistente en un conjunto de paquetes que proporcionan una interfaz común para el desarrollo de programas en distintas plataformas. Incluye las librerías necesarias para el desarrollo en distintos entornos: consola, entorno de ventanas, aplicaciones cliente/servidor, entre otras.

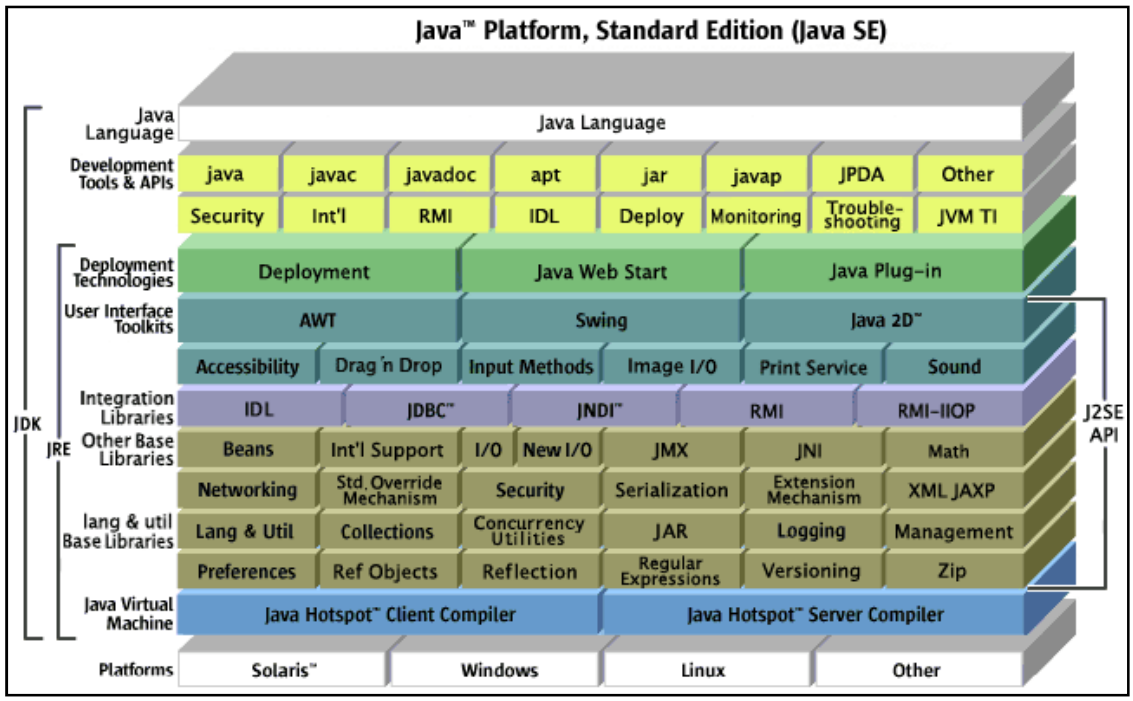


Figura 4.1: Arquitectura Java Platform Standard Edition (Java SE).

Al tratarse de un lenguaje de programación orientado a objetos, permite la definición de clases y sus atributos y son accesibles de una manera muy cómoda.

Gracias a su ejecución en máquina virtual, los archivos generados son encapsulados e interpretados por la propia Java Virtual Machine, reduciendo por tanto el consumo de recursos y facilitando en gran medida la característica multiplataforma del proyecto que nos ocupa.

4.2.1.2. OpenGL.

En su adaptación JOGL (Java-OpenGL). Utilizado para el sistema de representación tridimensional de las trayectorias del mecanizado y el control de la cámara.

JOGL es un conjunto de librerías Open Source, de manera nativa programadas en C, y adaptadas para su uso en Java. Su integración, propiciada por la inclusión de la librería en el propio NetBeans, se convierte en una tarea fácil gracias al control SWING ("GLJPanel"), que ofrece un marco gráfico con un 'GL Event Listener' que reproduce las operaciones gráficas llevadas a cabo por el módulo correspondiente, funcionando, por tanto, de igual manera que OpenGL en su versión para C.

Cabe destacar el uso conjunto de la librería GLU (GL Utilities), integrada en NetBeans, que ofrece herramientas para el manejo de la cámara en sistemas de representación tridimensional, así como facilita la creación de sólidos y alámbricos tridimensionales.

La elección de OpenGL frente a otros lenguajes de programación gráfica reside, además de sus características multiplataforma, multilenguaje y Open Source, en que supone el API de programación gráfica de mayor uso en la comunidad de desarrolladores, pudiendo, por tanto, encontrar gran cantidad de información en multitud de soportes y, dada su relevancia, habiendo sido estudiado en múltiples asignaturas de la carrera.

4.2.1.3. HTML.

Usado para la elaboración de la estructura y elementos del sistema de ayuda integrado. Para la composición y generación del sistema se ha hecho uso de la librería JavaHelp 2.0.05.

HTML supone el lenguaje de marcado, descriptivo y en alto nivel de mayor uso dada su implantación en la práctica totalidad de páginas web existentes. Su característica de programación por etiqueta descriptiva lo convierte

en el lenguaje idóneo para su uso en sistemas de texto estructurado tales como el sistema de ayuda del proyecto que nos ocupa.

4.2.2. Entorno de programación.

El entorno de programación escogido ha sido NetBeans en su versión 6.9. Este *IDE* (Integrated Development Envoriment) multiplataforma, sponsorizado y recomendado por la propia Oracle, añade, a la propia flexibilidad del lenguaje, multitud de características que facilitan la creación, depuración y compilación de cualquier programa.

El entorno de desarrollo de la interfaz gráfica que integra es WYSIWYG (What You See Is What You Get), y permite el diseño de interfaces gráficas complejas mediante tecnología drag&drop de una completa paleta de controles de la librería SWING.

NetBeans (originalmente llamado Xelfi) nace en la República Checa en 1996 como un proyecto estudiantil para la creación de un IDE de Java similar a Delphi. En el año 1999, el deseo de Sun Microsistems de tener una herramienta de desarrollo mejor, lleva a un acuerdo por el cual adquiere Netbeans y en el año 2000 libera su código y lanza Netbeans.org.

Una de las características de mayor importancia en Netbeans reside en su propia comunidad de desarrolladores. Al tratarse de un software Open Source, se pueden encontrar multitud de plugins y add-on's librerías que facilitan el diseño y desarrollo de aplicaciones.

Además, a pesar de que Netbeans está concebido originalmente como un entorno de desarrollo para Java, la posible incorporación al IDE de paquetes oficiales adicionales permite el desarrollo en otros lenguajes de programación tales como PHP, Phyton, C y C++, Ruby, entre otros.

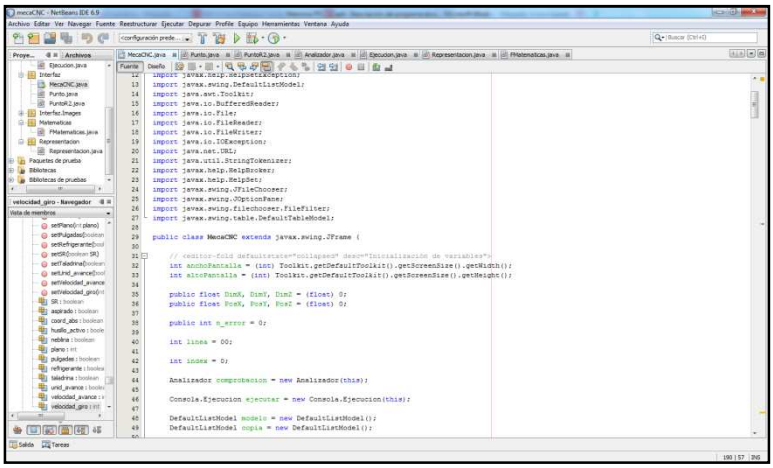


Figura 4.2: Ventana principal de código de mecaCNC sobre NetBeans 6.9

4.2.3. Material necesario para el desarrollo.

El desarrollo del proyecto será llevado a cabo mediante el uso de distintas herramientas dependiendo de la naturaleza del área a desarrollar. Dicho material se puede dividir según su tipo:

4.2.3.1. Herramientas hardware.

Asume el soporte físico sobre el que se hará uso de las distintas herramientas software. El equipo hardware sobre el que se ha desarrollado el proyecto cumple las siguientes características:

- Equipo Apple MacBook Pro 15” (serie 5.3)
  - Procesador Intel Core 2 Duo 2.66 GHz.
  - 8GB de memoria RAM DDR3 con bus a 1333MHz.

Si bien es cierto que disponer de un buen equipo proporciona fluidez y conlleva una comodidad extra en el desarrollo, el proyecto podría haber sido desarrollado en equipos de menor precio y características, aunque sí se considera adecuado el uso de un equipo Apple que integre el sistema operativo MacOSX y permita la instalación de los restantes sistemas operativos haciendo uso de herramientas software de virtualización.

4.2.3.2. Herramientas Software.

Al contrario que el caso anterior, han sido bastantes las herramientas software utilizadas para el desarrollo del proyecto, máxime considerando la característica multiplataforma del programa, que hace necesaria la presencia de múltiples sistemas operativos para realizar las pruebas oportunas.

Herramienta	Uso
Microsoft Word 2007	Redacción del presente documento
Microsoft Visio 2003	Creación de diagramas de flujo
Adobe Illustrator CS4	Creación de gráficos vectoriales (principalmente los contenidos en el capítulo 5)
Gantt Project	Elaboración del diagrama de Gnatt
VMWare Fusion 3.0.0 - Microsoft Windows 7 - Microsoft Windows XP - Ubuntu 11.04 - Solaris 10	Virtualización de los distintos sistemas operativos indicados
NetBeans 6.9 - Java 6 SE - OpenGL Pack - JavaHelp	Implementación
Notepad++	Implementación

Tabla 4.1: Herramientas software empleadas en el desarrollo.

4.3. Planificación.

Uno de los aspectos esenciales en la elaboración de un proyecto complejo radica en la necesidad de establecer un plan de acción para abordar el desarrollo de una forma efectiva y ordenada, respetando los tiempos y cumpliendo en la entrega.

La metodología de desarrollo de un proyecto en base a la ingeniería del software supone un conjunto de herramientas y técnicas que facilita la estructuración y ejecución secuencial de las tareas, teniendo en consideración el ciclo de vida del producto y la precedencia de las fases del desarrollo para obtener unos buenos resultados de manera incremental y progresiva.

Entre los modelos que se ofrecen desde las escalas teóricas de la ingeniería del software, cabe destacar el modelo en espiral, que determina los segmentos de la fase de desarrollo para distintos escenarios, obteniendo versiones beta-incrementales que evolucionan y se amplían en cada ciclo del modelo.

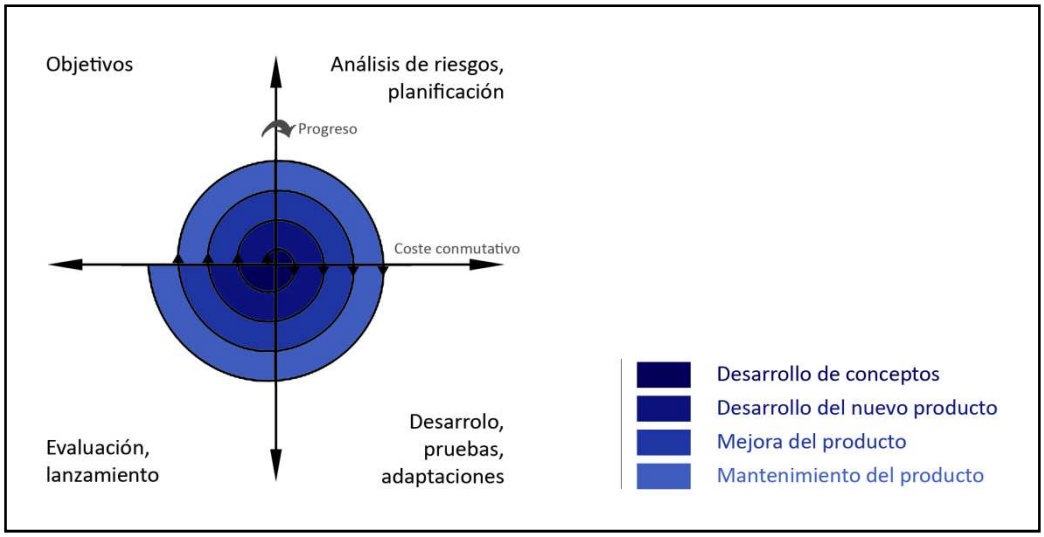


Figura 4.3: Modelo en espiral de Boehm en cuatro fases.

En este modelo, sintetizado en 4 fases, podemos establecer dos ejes de medición: por una parte el avance angular indicará el avance en el proyecto, por otra el radial que supondrá los costes conmutativos. A medida que avancemos angularmente en el desarrollo del proyecto, el nivel de especificaciones, objetivos, implementación, etc. aumentará, aumentando en consecuencia el coste asociado al desarrollo.

Además, cada sector angular etiquetado con distintos tonos de azul, identificará un tipo de proyecto de ingeniería del software identificable en la propia leyenda de la Figura 4.3.

Las cuatro fases asociadas al modelo son:

Objetivos: se establecerá qué se pretende del proyecto y de su desarrollo, considerando la funcionalidad que deberá ejecutar, características principales, intervalo de costes y tiempos de revisión, evaluación y entrega.

Análisis de riesgos y planificación: una vez definidos los objetivos, se formalizarán y analizarán los requisitos para el correcto cumplimiento, y del resultado del análisis se podrán establecer los tiempos de ejecución. Completada esta fase, el desarrollador estará en disposición de elaborar un dossier contractual donde se indicarán los distintos aspectos a cumplir en forma y plazo.

Desarrollo, pruebas y adaptaciones: En cada nuevo ciclo del modelo, será necesario desarrollar y probar las funcionalidades asociadas a los nuevos objetivos, mostrando al cliente los resultados de las beta-pruebas y adaptando, en caso de que fuera necesario, aquellos aspectos cuya implementación no se ajuste por completo a la realidad demandada.

Evaluación y lanzamiento: una vez realizadas todas las pruebas y adaptaciones necesarias, se dispondrá de una versión *release* completa del proyecto en base a los objetivos de la línea angular correspondiente. Si la evaluación de dicho *release* es correcta, el proyecto estará listo para ser lanzado/entregado al cliente.

La aproximación concreta de este modelo al proyecto que nos ocupa identifica al director de proyecto de final de carrera como cliente, el alumno será el desarrollador y el proyecto de ingeniería de software en sí, junto a su documentación, el producto. Durante todo este proceso, existe una comunicación constante y directa entre el cliente (director de proyecto) y el desarrollador (alumno), asegurando de esta forma la fluidez y buen término de la propuesta.

4.3.1.1. Sub-tareas.

Para conseguir una buena planificación, se considera conveniente dividir el grueso del desarrollo del proyecto en sub-tareas más simples, concretizando los objetivos principales en objetivos intermedios y estructurándolas, en la medida de lo posible, por bloques de implementación.

Considerando este pretexto, se realiza la siguiente subdivisión de tareas:

ID	Tarea
0	Recopilación de documentación de consulta
1	Análisis de objetivos, requisitos y riesgos
2	Diseño en alto nivel
3	Diseño en bajo nivel
4	Planificación
5	Implementación, pruebas y adaptaciones iterativas
6	Obtención y prueba de la versión final
7	Redacción de la documentación (memoria y manuales)
8	Evaluación de resultados y entrega

Tabla 4.2: Relación de sub-tareas.

4.3.1.2. Hitos.

Los hitos suponen los objetivos intermedios que se asignarán a cada sub-tarea. La correcta finalización de todos los hitos llevará implícita la consecución de los objetivos finales, y por tanto, la finalización exitosa del proyecto.

Tarea	Hitos
0	Recopilar documentación sobre procesos de mecanizado, programación CNC en código G, sistemas CAD/CAM y estado del arte.
	Recopilar documentación sobre la máquina EMCO PC Mill 125
	Recopilar documentación asociada a la programación en Java y el uso de NetBeans.

	Recopilar documentación sobre programación gráfica OpenGL y su integración en programas Java.
	Recopilar documentación sobre usabilidad en aplicaciones orientadas a la docencia.
1	Establecer objetivos principales.
	Establecer requisitos funcionales y recursos materiales necesarios.
	Analizar requisitos y recursos y evaluar la capacidad de consecución del proyecto en base a los conocimientos, documentación y recursos disponibles.
	Considerar las posibles limitaciones.
2	Definir y analizar variables relevantes del diseño de la interfaz.
	Crear Wireframe del entorno de ventanas.
	Recopilar, crear iconos e imágenes a emplear.
	Crear el logo de la aplicación.
	Analizar conjunto de instrucciones de código G interpretables por el simulador
	Crear diagramas de flujo de interacción usuario ↔ interfaz ↔ funcionalidad
3	Definir paquetes/módulos que serán necesarios y mantendrán una estructura coherente en el programa.
	Establecer clases principales y relaciones entre ellas.
	Crear analizador léxico-sintáctico-semántico para la ejecución en tiempo real.
4	Establecer planificación.
5	Crear interfaz gráfica y añadir componentes.
	Reconocer automáticamente resolución de pantalla del equipo y adaptar la ventana a dicha resolución.
	Crear paquetes e importar librerías necesarias.
	Crear el conjunto de clases.
	Programar los métodos (funciones y eventos de componentes).
	Programar el analizador léxico-sintáctico-semántico.
	Programar la relación entre clases y la funcionalidad asociada a cada instrucción en código G.
	Programar el sistema de ayuda.
	Optimizar el código.
	Realizar sucesivas pruebas iterativas con cada implementación.
	Adaptar la programación al resultado de las pruebas iterativas.
6	Realizar pruebas finales.
	Obtener versión final y encapsularla correctamente.
7	Redactar la memoria del proyecto.
	Redactar el manual de usuario.
8	Evaluar resultados finales.
	Realizar la entrega y presentación.

Tabla 4.3: Relación de hitos.

Cabe destacar que el orden de consecución de los hitos puede ser diferente al mostrado en la *Tabla 4.3*, si bien es cierto que no se podrá

completar una tarea sin haber completado todas las tareas anteriores, lo cual implica la necesidad de haber conseguido todos los hitos de dichas tareas previas.

4.3.1.3. Diagrama de Gantt.

Otra de las herramientas de uso generalizado y gran implantación en el establecimiento y especificación de los tiempos de ejecución de un proyecto consiste en la elaboración de un diagrama de Gantt.

El diagrama de Gantt muestra, por cada fila, una tarea, dividiendo el conjunto de columnas en dos sub-conjuntos: la parte izquierda mostrará los atributos de cada tarea (generalmente, identificador, nombre, duración y fechas de inicio y fin) y la parte derecha (que ocupa la mayor parte del diagrama), mostrará la línea de tiempo, estableciendo una forma rectangular para cada tarea de longitud la duración y las posibles relaciones entre tareas.

Para obtener una buena referencia de los resultados de la planificación, se mostrarán dos diagramas:

El primero *(Figura 4.4)* hará referencia a la estimación de la planificación que se tiene previo al inicio del desarrollo del proyecto, estableciendo los tiempos que se consideran adecuados en base al análisis de objetivos y requisitos previo a la fase de implementación.

El segundo *(Figura 4.5)*, detallará los tiempos reales en los que se ha concluido una vez se haya desarrollado completamente el proyecto, evaluando si se ha conseguido respetar los tiempos iniciales o se ha modificado la planificación temporal por cualquier circunstancia.

Para el caso que se está analizando, el planteamiento inicial sufrió una modificación en los tiempos de ejecución debido a limitaciones propias de la incompatibilidad con el horario laboral. Dicha circunstancia propició, además, que a diferencia de la habitual jornada de 8 horas diarias, en meses de 20 días, se redujera considerablemente el tiempo disponible de dedicación al proyecto, fluctuando en base a la carga laboral, desplazamientos, etc.

Este hecho conlleva la necesidad de ampliar los tiempos de ejecución previstos, principalmente, para las tareas de diseño, implementación, pruebas y obtención de la versión final.

Por el contrario, la redacción de la documentación sufre una variación en el tiempo de ejecución por motivos distintos a los expuestos anteriormente, ya que en este caso, simplemente se consideró una estimación de tiempo menor a la realmente necesaria, pues durante este periodo no existía el compromiso laboral, empleándose una media de 8 o más horas diarias.

En definitiva, la variación resulta en un retraso en la conclusión del proyecto de 2 meses y 9 días, equivalentes a 47 días laborables.

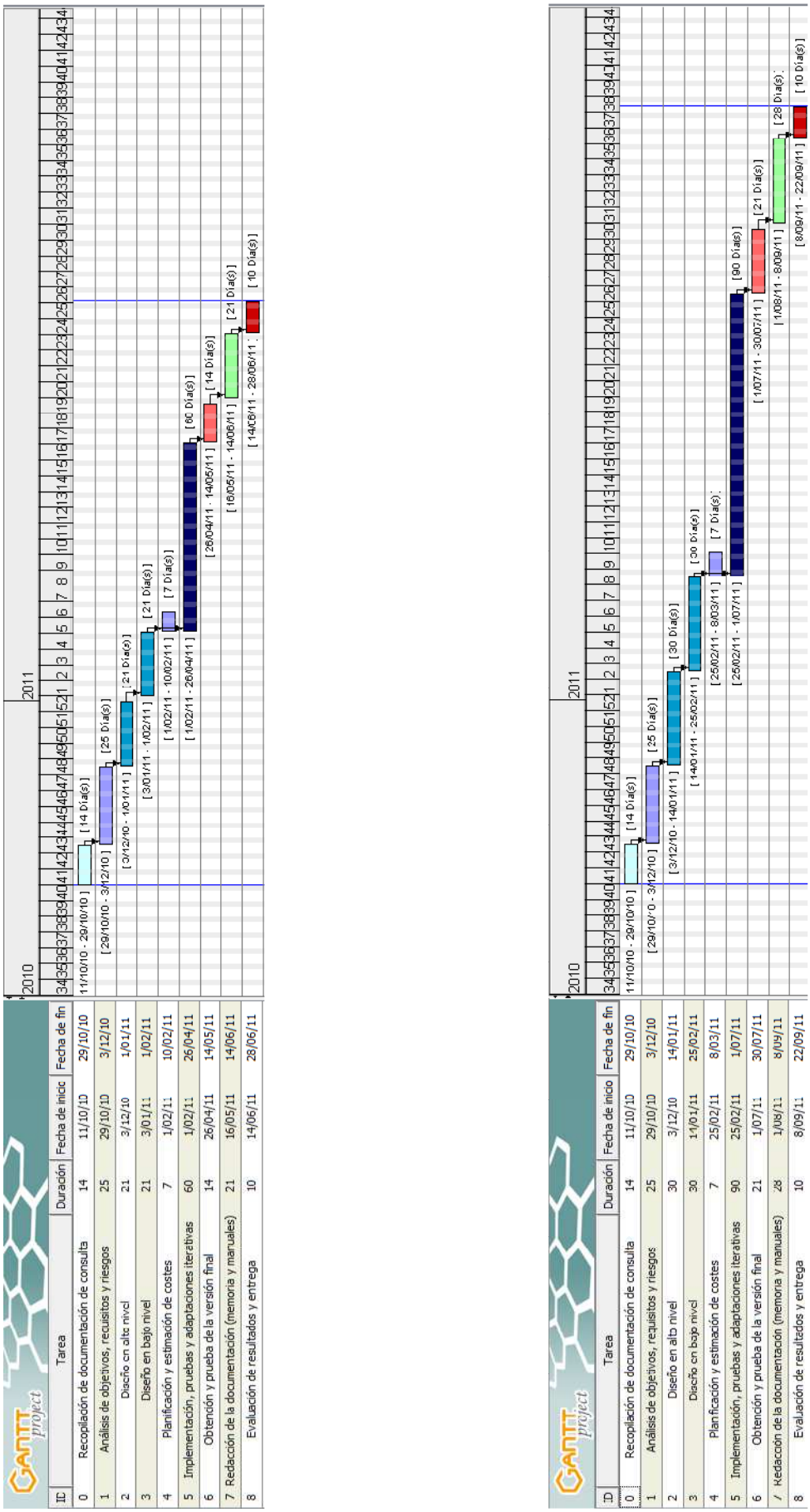


Figura 4.4: Diagramas de Gantt previo y resultado

4.4. Estructura del programa.

En el proceso de desarrollo de un proyecto de ingeniería del software, será necesario distinguir entre dos tipos de niveles o fases del diseño, íntimamente ligados y que mantienen una jerarquía específica que permite concretizar, implementar y optimizar los requisitos funcionales del programa: el diseño en alto y bajo nivel.

A modo de resumen introductorio de los siguientes sub-apartados, se podría considerar que el **diseño en alto nivel** supone el *"qué"* de la aplicación, definiendo cual va a ser la funcionalidad de la aplicación y que se espera de la interacción usuario ↔ programa, en un lenguaje claro y preferiblemente adjunto a gráficos y diagramas de flujo, detallando el *"por qué"* se considera adecuada dicha estructura en base al *target* y los objetivos.

Se ha considerado adecuado incluir el diseño de la **interfaz gráfica** en un nuevo apartado entre los diseños de ambos niveles debido a la estrecha relación existente entre la forma en que el usuario usa el programa (intuyendo a simple vista cuál es la función de cada componente) y el aspecto y estructura de la interfaz que se le presenta. Por ello, además se ha analizado el perfil de uso en función del target, el contenido y el contexto del programa.

Por otra parte, se podría considerar el **diseño en bajo nivel** como el *"cómo"* se va a implementar y ejecutar la estructura definida en la fase de diseño anterior, indicando las estructuras de datos y métodos a emplear para cumplir los requisitos funcionales, también aportando el *"por qué"* se ha considerado adecuada esa solución frente a otras implementaciones posibles.

Por último, dada la relevancia que supone en el proyecto, haciendo posible la ejecución en tiempo real, se incluirá un apartado donde se detalle el **analizador léxico-sintáctico-semántico.**

4.4.1. Diseño en alto nivel.

En la fase de diseño en alto nivel definiremos la funcionalidad asociada al programa y los subsistemas que, siguiendo una estructura lógica y ordenada que facilite la implementación y modificación en la siguiente fase, permitan llevar a cabo las acciones necesarias para alcanzar los objetivos funcionales programa.

4.4.1.1. Arquitectura general.

Gracias a un análisis previo de los requisitos, podemos obtener las partes diferenciadas que llevarán a cabo las distintas funciones del programa.

El enlace de estos módulos de la arquitectura general del programa con la fase posterior del diseño, permite definir cuáles serán los paquetes Java a implementar.

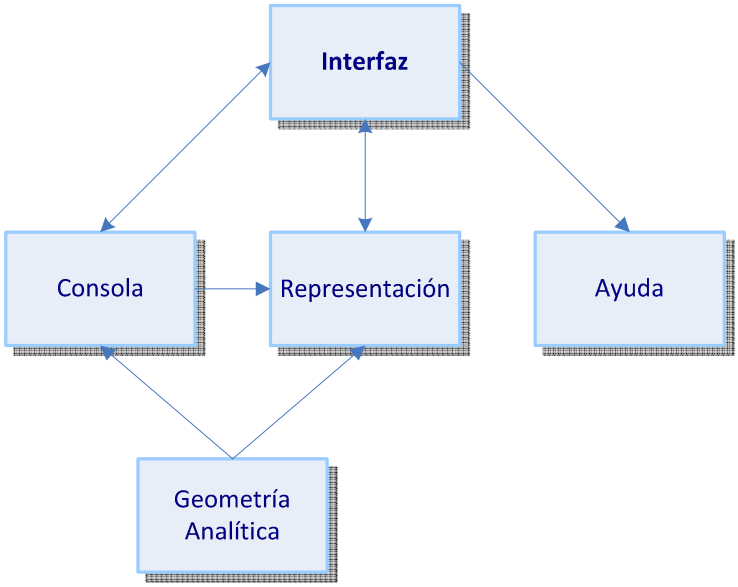


Figura 4.5: Diagrama de arquitectura general.

De la *Figura 4.5* podemos extraer como módulo principal el subsistema ***Interfaz***, el cual manejará la interactuación del usuario con los componentes de la interfaz, generando distintos eventos que darán lugar a un determinado flujo de ejecución entre subsistemas.

Los subsistemas intermedios ***Consola*** y ***Representación*** llevarán a cabo las operaciones necesarias para la ejecución de instrucciones y posterior representación en tiempo real, mostrando en la interfaz el resultado en caso de éxito o error en caso contrario. Ambos subsistemas realizarán la toma de datos apoyándose, en caso necesario, en el subsistema que ***Geometría Analítica*** que contendrá las funciones matemáticas necesarias.

Por último, el subsistema ***Ayuda*** será llamado desde el módulo *Interfaz* pero dispondrá de su propio manejador de eventos para las interacciones con el usuario.

4.4.1.2. Interacciones entre módulos.

Del diagrama anterior podemos extraer las relaciones entre módulos en base a la funcionalidad del programa que percibe el usuario.

Para ello, se presentará el siguiente diagrama *(Figura 4.7)* que muestra los distintos módulos (en desarrollo vertical), y las líneas horizontales etiquetadas con cada funcionalidad, que suponen el paso de la línea de ejecución por los módulos correspondientes.

Para el caso de la apertura de documentación en PDF, la interfaz llamará al programa predeterminado de apertura de archivos PDF en el equipo, que será externo a mecaCNC.

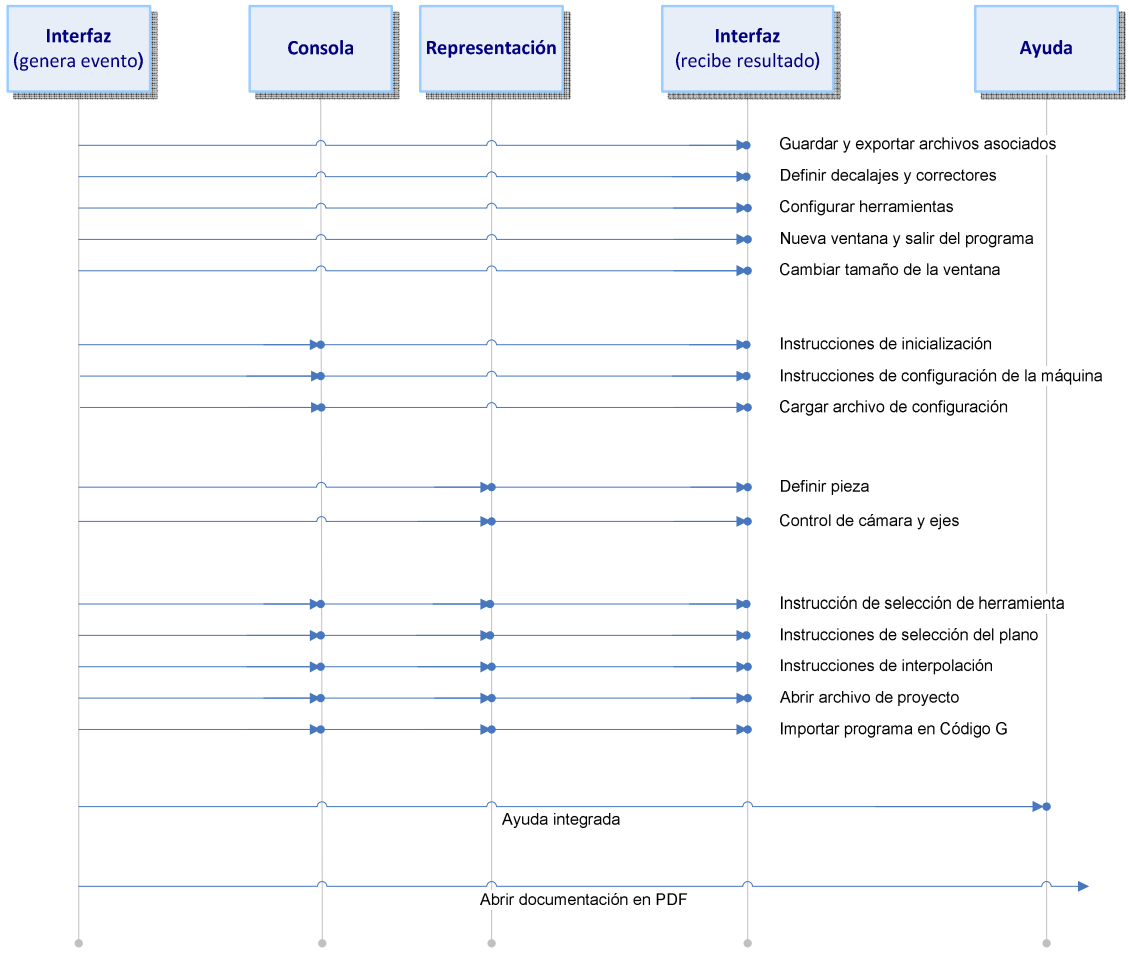


Figura 4.6: Interacciones entre módulos en base a la funcionalidad.

4.4.1.3. Flujo de ejecución de una instrucción ISO-6983 en mecaCNC.

Obtenidas las interacciones entre módulos, restará considerar uno de los aspectos más importantes de este nivel en la ejecución en tiempo real de las instrucciones en Código G insertadas: el diagrama de flujo de dichas instrucciones.

El flujo de ejecución de una instrucción en Código G dependerá del conjunto al que pertenezca dicha instrucción, pudiendo suponer instrucciones de

inicialización, control de trayectoria o control de máquina, manteniendo acciones comunes a todas ellas en el proceso de ejecución.

Para comprender mejor el camino seguido para completar la ejecución de una instrucción en código G, se recomienda al lector comprobar la lista de instrucciones interpretables por el simulador y su funcionalidad asociada, disponible en el *Anexo B* de este documento.

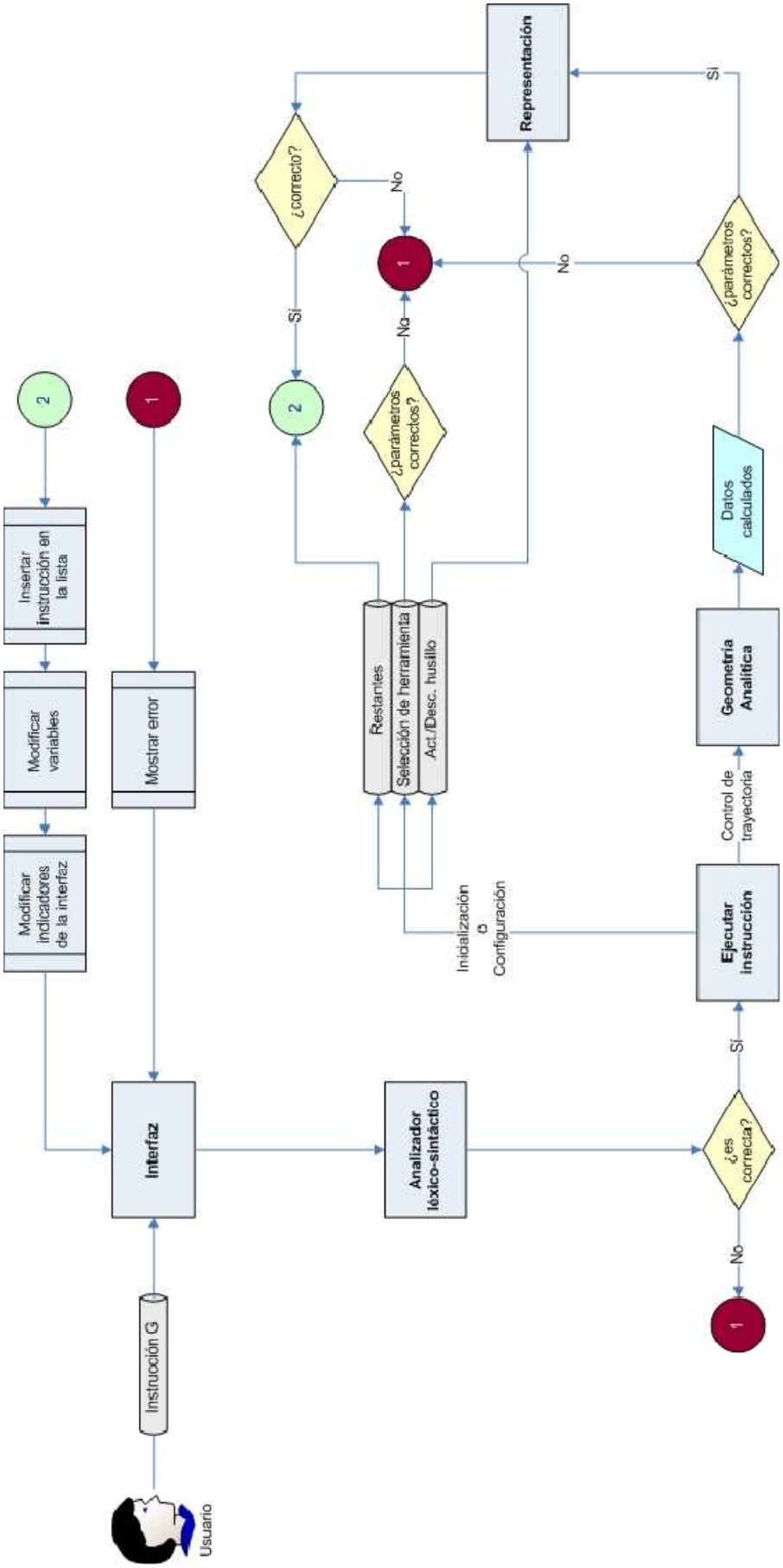


Figura 4.7: Diagrama de flujo de la inserción de una instrucción en Código G.

4.4.2. Interfaz gráfica.

Un aspecto esencial en el análisis de un programa en entorno de ventanas es su interfaz gráfica. Para el proyecto que nos ocupa, eminentemente especializado y orientado a la docencia, se ha tratado de mantener un equilibrio entre los siguientes factores en relación a la interfaz gráfica:

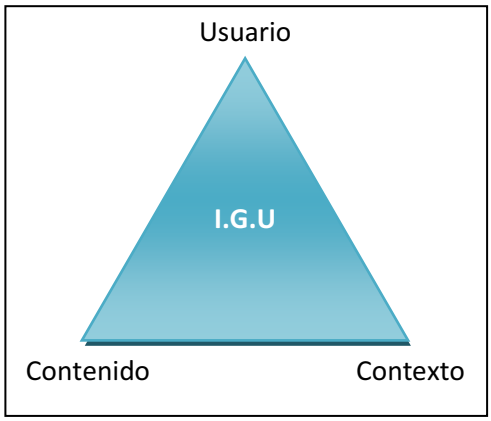


Figura 4.8: Variables de análisis en la creación de la interfaz gráfica.

4.4.2.1. Usuario.

El usuario tipo de la aplicación comprende a una persona altamente cualificada, previsiblemente un estudiante de ingeniería informática o industrial, con grandes capacidades de asimilación y manejo de un programa informático.

A pesar de ser recomendados los conocimientos en sistemas de mecanizado, el simulador pretende ser una herramienta de auto-aprendizaje, permitiendo al usuario aprender desde cero los conceptos de los sistemas de mecanizado y programación CNC en Código G.

Aún tratándose de un programa especializado, el usuario tipo está acostumbrado al uso de programas cuyo diseño sea depurado y de fácil manejo.

4.4.2.2. Contenido.

El simulador dispondrá la funcionalidad en base a aquellas acciones que el usuario realiza de forma física en la máquina real, y aquellas que corresponden a la inserción de instrucciones que modifican el comportamiento de la máquina.

Para ello, se tratará de dividir la ventana principal y ventanas secundarias en subtareas perfectamente diferenciadas que distingan cada una de

las acciones realizadas por el usuario (definición, configuración, inserción, representación, etc.).

Las funcionalidades han de ser específicas y accesibles en el menor número acciones posibles, facilitándose así un completo conjunto de atajos de teclado, barra de menú y acceso por ratón a la completa funcionalidad del programa.

4.4.2.3. Contexto.

El uso del simulador se presupone en el entorno académico, más concretamente en laboratorios de mecanizado. La naturaleza educativa del simulador hace necesaria una interfaz que facilite el acceso al auto-aprendizaje y el uso indistinto de usuarios con conocimientos avanzados en sistemas de mecanizado y en aquellos que inician su educación en este ámbito.

Se considera adecuado aproximar todas aquellas acciones y eventos a los correspondientes en la máquina EMCO PC Mill 125, la cual se da por hecho podrá ser accesible por el alumno en los laboratorios.

4.4.2.4. Primera aproximación: Wireframe.

Considerando el análisis anterior, se decide la elaboración de un primer boceto de la interfaz sobre la cual se realizará el desarrollo del proyecto.

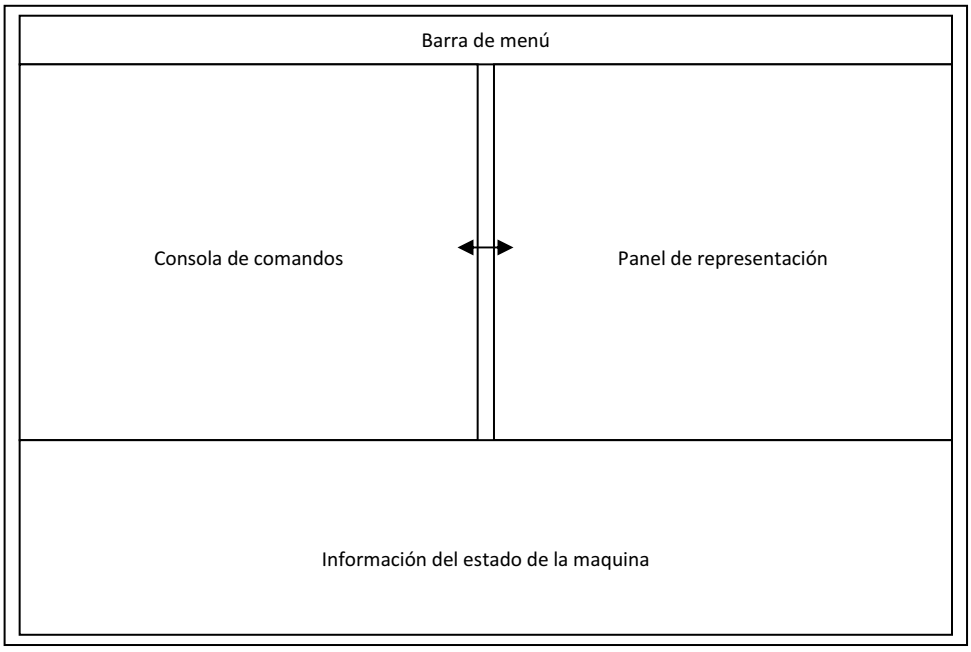


Figura 4.9: Wireframe del interfaz grafico de mecaCNC.

4.4.2.5. Resultado.

A partir del wireframe anterior, se construye la interfaz de la ventana principal, que de manera incremental, va integrando funcionalidades, componentes, iconos, atajos de teclado y ratón y ventanas secundarias.

En este apartado se desarrollará únicamente aquellas ventanas cuya interfaz gráfica sea susceptible de análisis, obviando aquellas ventanas que hacen referencia a tablas y cuya interfaz sea simple, no obstante, el *Anexo A* contiene todas las ventanas y elementos junto a la explicación de sus respectivas funciones.

4.4.2.5.1. Ventana principal.

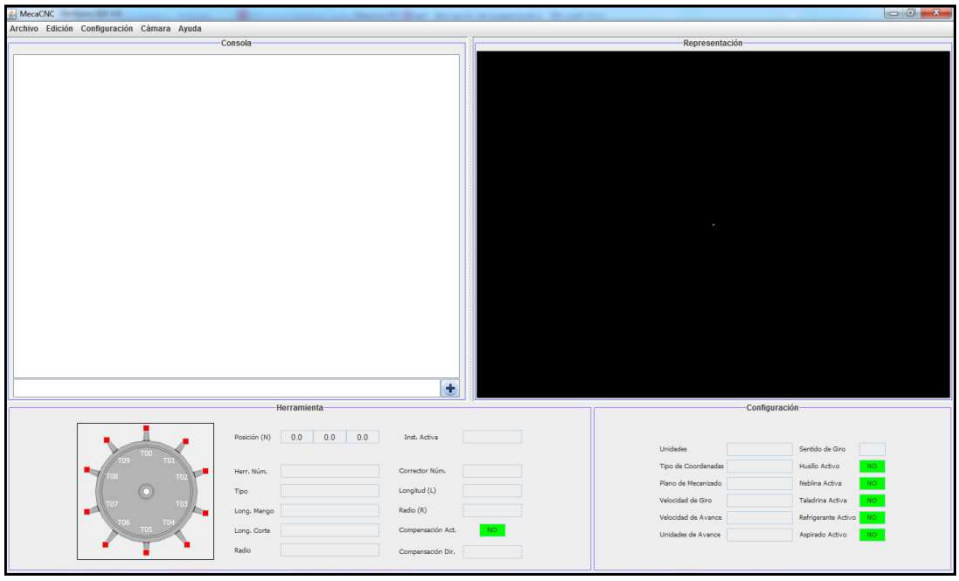


Figura 4.10: Interfaz gráfica de la ventana principal de mecaCNC.

Como se puede comprobar, durante el diseño se decide dividir el panel inferior correspondiente a la información de la máquina en dos sub-paneles: por una parte, el panel Herramienta que mostrará el estado de la herramienta activa, la posición del husillo y la instrucción de interpolación activa; por otra, el panel Configuración con los indicadores correspondientes a las instrucciones G que activan ciertas funcionalidades de la máquina.

De esta forma, se pretende dividir aquellas características de la máquina en las que es necesaria la intervención física del usuario (una herramienta se ha de cargar manualmente en el porta-brocas) o están sujetas a cambios constantes (cada interpolación modifica la posición del husillo) y aquellas que se activan en función de la programación del mecanizado

(mediante instrucciones en Código G, sin intervención física del usuario) y suelen mantenerse constantes durante todo el proceso (salvo el husillo).

Por otra parte, dada la importancia de la representación del mecanizado, se decide emplear un panel móvil que contenga ambos paneles superiores, pudiendo así variar las dimensiones de la consola o el marco de representación tridimensional del mecanizado.

Por cuestiones evidentes de usabilidad e incidencia en el recuerdo, se decide añadir atajos de teclado a cada una de las acciones de la barra de menú y asociarles un icono que identifique la acción que llevará a cabo.

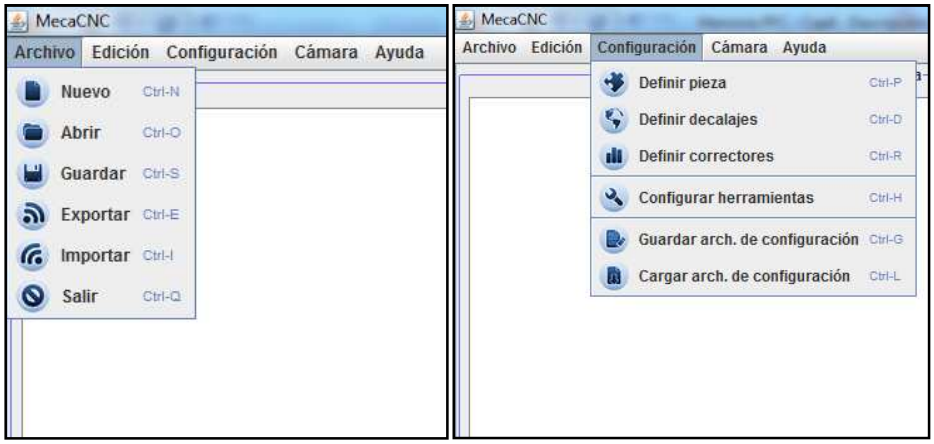


Figura 4.11 (izquierda): Detalle del elemento de menú Archivo.
Figura 4.12 (derecha): Detalle del elemento de menú Configuración.

4.4.2.5.2. Ventana Definir Pieza.

Esta ventana secundaria trata de mostrar de forma visual, la distribución del banco de trabajo y la dirección de los ejes en las operaciones de movimiento.

En los sistemas reales, la ubicación de la mordaza se realiza de forma automática, en base a las dimensiones de la pieza, pero mediante este sistema de determinación de la ubicación se pretende que el alumno entienda la asignación del punto Cero Pieza en el sistema de coordenadas de la máquina.

Una vez realizada la definición de los distintos valores de dimensión de pieza y ubicación de la mordaza, las etiquetas identificativas cambiarán su nombre por su valor, ofreciendo así una mejor comprensión de cada una de las medidas.

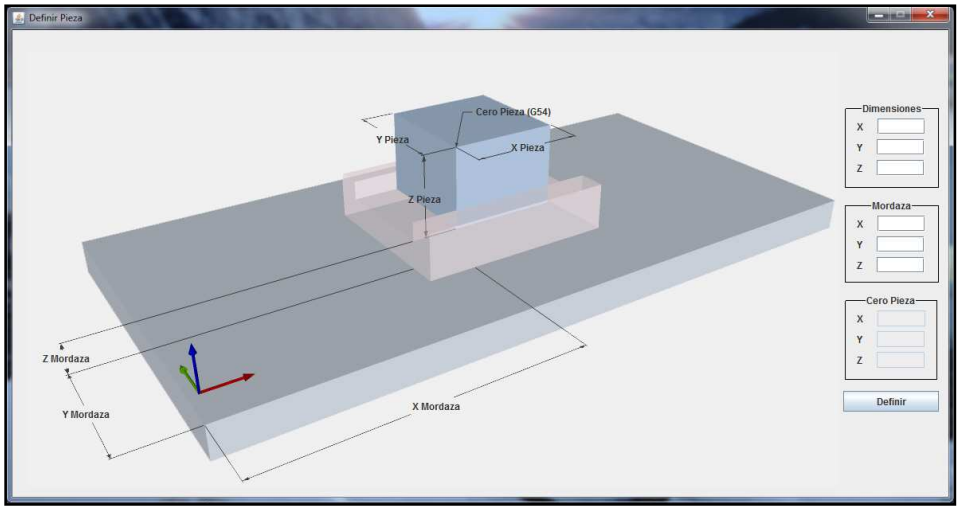


Figura 4.13: Ventana secundaria Definir Pieza.

4.4.2.5.3. Ventana Configurar Herramientas.

En el caso de la ventana Configurar Herramientas, se ha tratado de mantener el principio docente, tratando de mostrar al alumno las medidas por las que se rige cualquier herramienta en un proceso de mecanizado.

Además, la selección e instalación de herramientas en el porta-brocas consiste en un trabajo manual, por tanto, se ha querido seguir un sistema de asignación de entre las múltiples herramientas posibles (incluyendo las que puede definir el usuario), se seleccionará e instalará en el carrusel porta-herramientas únicamente aquellas que serán utilizadas en el mecanizado. El código de colores del carrusel porta-herramientas identificará en todo momento los porta-brocas que disponen de una herramienta asignada. Cabe destacar la intencionalidad de diseñar un frame identificativo que aproxime a la realidad del carrusel porta-herramientas.

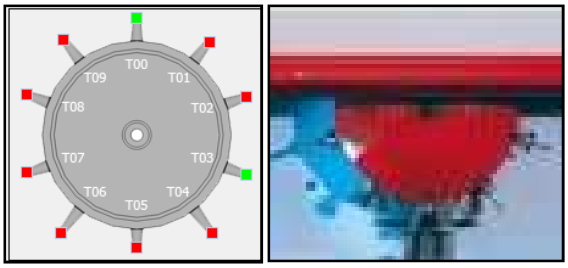


Figura 4.14: Carrusel virtual vs. carrusel real.

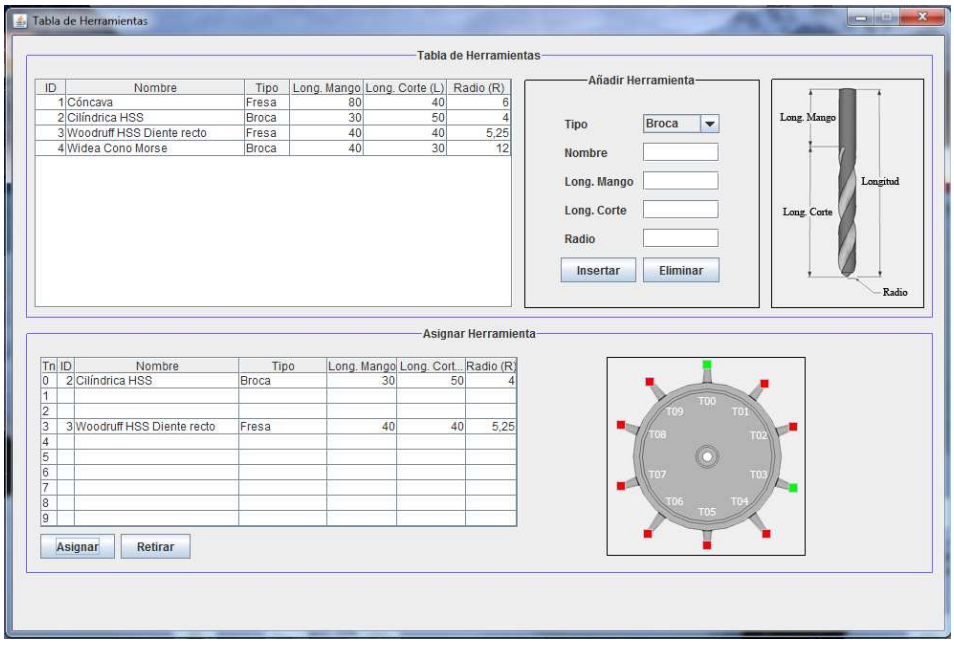


Figura 4.15: Ventana secundaria Configurar Herramientas.

Como se puede comprobar, la ventana Configurar Herramientas se divide en dos paneles: en el panel superior se establece la tabla de herramientas disponibles, obteniendo cuatro herramientas por defecto y pudiendo añadir dinámicamente, nuevas herramientas indicando sus parámetros; el panel inferior identifica el estado del carrusel porta-herramientas, pudiendo asignar las herramientas de la tabla superior en las 10 ranuras porta-brocas.

4.4.3. Diseño en bajo nivel.

Gracias al diseño en alto nivel, se sabe que el programa se dividirá en 5 bloques o paquetes Java diferenciados y enlazados entre sí mediante instanciación. Cada paquete contiene una o varias clases y los archivos auxiliares necesarios tales como imágenes o archivos HTML.

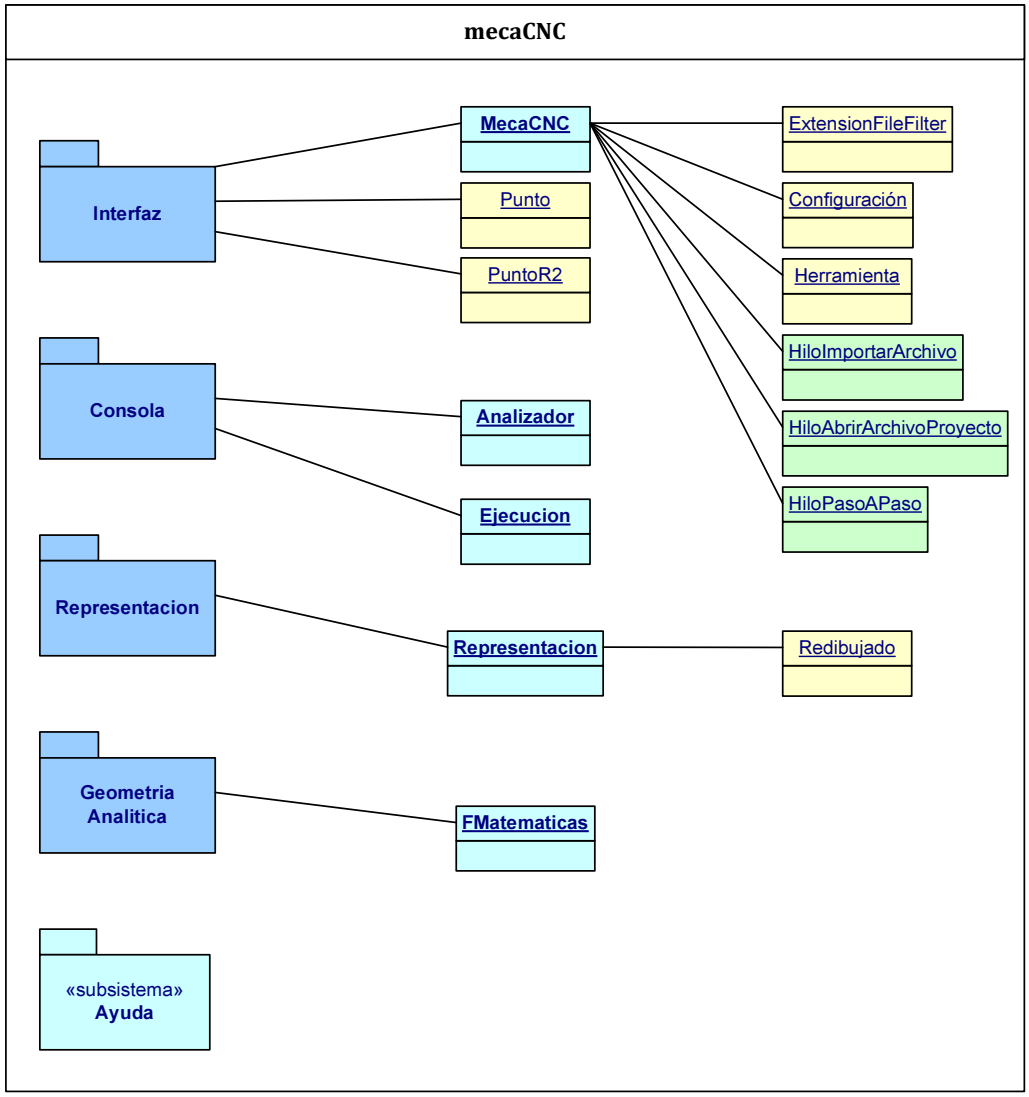


Figura 4.16: Diagrama de paquetes y clases de mecaCNC.

En la *Figura 4.17*, pueden identificarse los paquetes de color azul, las clases principales de color azul celeste, los hilos en color verde y las clases secundarias de color amarillo.

4.4.3.1. Paquete Interfaz.

Módulo principal del programa. Describe el entorno de ventanas y brinda el acceso al resto de funciones del simulador.

El paquete implementa tres clases:

4.4.3.1.1. Clase MecaCNC.

1. Clase principal del programa.

2. Define los marcos y componentes que darán lugar al entorno de ventanas.

3. Define las variables globales que se utilizarán y modificarán en las llamadas al resto de módulos. Para ello, se pasará la instancia activa de la clase al resto de módulos, permitiendo así la modificación en los valores de dichas variables o componentes de la interfaz.

4. Implementa los eventos que describen la funcionalidad al hacer uso de cada uno de los componentes de la interfaz.

5. Es la encargada de la apertura y creación de los archivos de proyecto, configuración y Código G.

6. Implementa la subclase *ExtensionFileFilter*, encargada de manejar las extensiones de los archivos propios del programa en las acciones de apertura, guardado, importación y exportación: *'.mcnc'* para archivos de proyecto, *'.mcfg'* para archivos de configuración y *'.txt'* y *'.g'* para archivos de Código G.

7. Implementa la subclase *Configuracion*, que incorpora las variables y métodos para el control del estado de la máquina.

8. Implementa la subclase *Herramienta*, que contiene las variables y métodos que gestionan el estado de la máquina en relación a la herramienta activa.

9. Implementa los hilos de ejecución *HiloImportarArchivo* e *HiloAbrirArchivoProyecto*, encargados de realizar la lectura del Código G contenido en ellos. Considerando que el volumen de instrucciones de este tipo de archivos puede ser muy elevado, la implementación multihilo supone un factor de importante de optimización y eliminación de posibles cuellos de botella.

10. Implementa el hilo de ejecución *HiloPasoAPaso* encargado de manejar la ejecución cuando se activa el controlador de ejecución paso a paso desde el menú contextual de la lista de instrucciones insertadas. De manera análoga a los hilos detallados anteriormente, el volumen de instrucciones insertadas puede ser elevado y este hecho no ha de alterar el correcto manejo de eventos de la interfaz.

4.4.3.1.2. Clase Punto.

A pesar de suponer una subclase del paquete, la clase Punto reviste una gran importancia, dado que será la encargada de definir todos aquellos aspectos del programa basados en puntos del espacio tridimensional.

Esto incluye, por una parte, los puntos utilizados por OpenGL para el dibujado de las interpolaciones que vaya describiendo la máquina, y por otra, todos los puntos de referencia propios de un mecanizado: posición del husillo (N), decalajes, etc.

4.4.3.1.3. Clase PuntoR2.

Subclase empleada para la realización de cálculos matemáticos en interpolaciones circulares. Este tipo de interpolación se realizará sobre el plano, variando únicamente sobre dos ejes, por tanto, de cara al cálculo de distancias, centros y puntos de corte, se consideró adecuado generar una nueva clase punto sobre R2.

4.4.3.2. Paquete Consola.

El módulo Consola implementa el analizador léxico-sintáctico de Código G y describe la funcionalidad de cada una de las instrucciones G interpretables por el programa.

4.4.3.2.1. Clase Analizador.

Implementa el analizador léxico-sintáctico encargado de manejar los posibles errores tanto en el léxico de los tokens de las instrucciones G insertadas, como en la construcción sintáctica.

El analizador léxico-sintáctico-semántico se analizará con más detalle en el apartado 4.4.4 del presente capítulo.

4.4.3.2.2. Clase Ejecucion.

Clase encargada de realizar la funcionalidad asociada a cada una de las instrucciones interpretables por el simulador.

En el caso de que el analizador haya comprobado el léxico y la sintaxis de la instrucción, la clase *Ejecucion* comprobará la semántica (en caso de que sea necesario) y realizará las acciones asociadas a dicha instrucción, pudiendo ser el cambio en el valor de alguna variable, clase o indicador del interfaz, o asociando los puntos de origen y destino, y realizando las llamadas pertinentes a la clase *FMatematicas* y *Representacion*.

4.4.3.3. Paquete Representacion.

Implementa las funciones de manejo de la representación gráfica del mecanizado.

4.4.3.3.1. Clase Representacion.

La clase Representacion define las variables y métodos derivados del uso de la librería gráfica OpenGL.

En el proyecto que nos ocupa, la clase Representacion es la encargada de dibujar, en el control SWING GLJPanel (que contiene un GLEventListener sobre esta clase), la pieza y las trayectorias del mecanizado, así como otros aspectos relacionados tales como el dibujado de los ejes o el control de la cámara.

Como bien es sabido, OpenGL no permite el trazado directo de arcos, por tanto, en el caso de interpolaciones circulares, será también esta clase la encargada de calcular y trazar los pequeños tramos de recta que compondrán el arco.

El uso de OpenGL implica la implementación de las funciones propias:

1. ***init***: contiene todas las instrucciones de inicialización del marco gráfico deseadas. En el caso que nos ocupa, se establece el color de fondo y la corrección de la perspectiva y el antialiasing de las líneas.
2. ***display***: función principal en la que se realiza el dibujado de las trayectorias tras insertar una instrucción de interpolación o se hace uso de alguna función relacionada con la representación gráfica.
3. ***reshape***: gestiona las relaciones de aspecto en la representación si se produce un cambio en las dimensiones del control o de la ventana que lo contiene.

4. ***displaychanged***: función ligada a posibles cambios en el hardware gráfico (por ejemplo, cambio de monitor). Generalmente, suele ser irrelevante y no se programa.

4.4.3.3.2. Subclase Redibujado.

A fin de optimizar la velocidad en la representación, se crea la subclase Redibujado que contiene la lista de puntos origen y destino, el color, el plano de mecanizado y el radio de la herramienta, del conjunto de instrucciones insertadas relacionadas con el movimiento de la máquina (G00, G01, G02 y G03).

Haciendo uso de esta clase, no será necesaria la repetición de cálculos relacionados con las trayectorias de las interpolaciones, y bastará con recorrer los elementos de la clase, asignando, en cada caso, sus valores de color, plano, etc.

4.4.3.4. Paquete GeometriaAnalitica.

Módulo que asume la implementación de las funciones para diversos cálculos matemáticos relacionados con la geometría euclidiana en la representación gráfica del mecanizado.

4.4.3.4.1. Clase FMatematicas.

Clase que contiene los métodos necesarios para el cálculo de aspectos esenciales en el trazado de interpolaciones circulares.

Dada la posibilidad de insertar interpolaciones circulares definiendo el radio del arco, o bien el punto centro, es necesaria la implementación de funciones que faciliten el cálculo de ciertos puntos necesarios para trazar el arco.

Las funciones matemáticas utilizadas en esta clase serán explicadas con mayor detenimiento en el capítulo 5 de este documento.

4.4.3.5. Paquete Ayuda.

El paquete Ayuda integra en sí mismo un subsistema diferente al del simulador, ya que, haciendo uso de la librería JavaHelp System, la interfaz y funcionalidad del sistema de ayuda corre a cargo de la propia librería.

Para implementar el sistema de ayuda, ha sido necesario programar el archivo de mapeado que hace referencia a cada uno de los archivos de información escritos en HTML, definir la estructura del TOC para la ayuda por contenidos así como el índice, e indexar cada uno de los términos que aparecen en los archivos de información para que sean accesibles en la búsqueda por término.

4.4.3.6. Variables y métodos principales por clase.

La siguientes tablas muestra el tipo y nombre de las variables principales empleadas para la comunicación entre clases, representación y control de flujo del programa, así como la relación de métodos principales asociados a cada clase, indicando a su vez la clase de la que hereda y especificando su función en el programa.

Paquete	Clase	Hereda de	Variables principales	
			Tipo	**Nombre**
Interfaz	MecaCNC	javax.swing.JFrame	Analizador	comprobacion
			Ejecucion	ejecutar
			boolean	camara, dec_act, ejes, eliminar, instact, ioconfig, restablecer
			float	angulo, camaraX, camaraY, camaraZ, desplazamiento_x, desplazamiento_y, DimX, Dim Y, Dim Z, last_x, last_y, miraX, miraY, miraZ, PosX, PosY, PosZ, radio, rotacion_x, rotacion_y
			int	compensacion, modificado, n_error, n_instruccion, n_vista, zoom
			javax.help.HelpSet	helpset
			javax.media.opengl.GLJPanel	gLJPanel1
			javax.swing.DefaultListModel	copia, modelo
			javax.swing.JList	jList1
			javax.swing.JTable	jTableAsign, jTableCorr, jTable Herr
			javax.swing.JTextField	jTaspirado, jTCompAct, jTCompDir, jTCorrLong, jTCorrNum, jTCorrRadio, jTHerrLCorte, jTHerrLMango, jTHerrNum, jTHerrRadio, jTHerrTipo, jThusillo, jTInstAct, jTneblina, jTplano, jTrefrigerante, jTsentido, jTtaladrina, jTtipoCoord, jTudavance, jTunidades, jTvelavance, jTvelgiro, posActX, posActY, posActZ
			MecaCNC.Configuracion	config
			MecaCNC.Herramienta	herr
			Punto	centro, origen, fin, auxiliar, deca54, deca55, deca56, deca57, N
			PuntoR2	centro_r2, fin_r2, origen_r2
	Punto	java.lang.Object	float	x, y, z
	PuntoR2	java.lang.Object	float	x, y
Consola	Analizador	java.lang.Object	MecaCNC	padre

	Ejecucion	java.lang.Object	MecaCNC	padre
			FMatematicas	Fmat
Representacion	Representacion	java.lang.Object	com.sun.opengl.util.GLUT	glut
			FMatematicas	Fmat
			javax.media.opengl.glu.GLU	gku
			java.util.ArrayList<Redibujado>	redibujar
			MecaCNC	padre
	Redibujado	java.lang.Object	float	blue, green, radio, red
			int	plano
			java.util.ArrayList<Punto>	puntos
Matematicas	FMatematicas	java.lang.Object	MecaCNC	padre

Tabla 4.4: Relación de clases y variables principales.

Paquete	Clase	Métodos	Función
Interfaz	mecaCNC	void limpiar()	Establece al valor por defecto cada uno de los indicadores de la interfaz
		void ponerDatosN(Punto pto)	Escribe las componentes de la ubicación (N) en el indicador de la interfaz
		74través74 void initComponents()	Genera los componentes
		private void formWindowActivated(WindowEvent evt)	Evento al abrir la ventana principal
		private void formWindowClosing(WindowEvent evt)	Evento al pulsar el botón de cierre de ventana (aspa)
		private void gLJPanel1MouseDragged(MouseEvent evt)	Evento al arrastrar ratón sobre panel *Representación*
		private void gLJPanel1MousePressed(MouseEvent evt)	Evento al pulsar botón del ratón sobre panel *Representación*
		private void gLJPanel1MouseReleased(MouseEvent evt)	Evento al soltar botón del ratón sobre panel *Representación*
		private void gLJPanel1MouseWheelMoved(MouseWheelEvent evt)	Evento al mover la rueda sobre el panel *Representación*
		private void jTextFieldDespXKeyPressed(KeyEvent evt)	Evento de control de pulsación de tecla *retorno* en cuadros de texto de *Definir pieza*
		private void jTinstruccionKeyPressed(KeyEvent evt)	Evento de control de pulsación de tecla en cuadro de inserción de instrucción G
		private void jTinstruccionKeyTyped(KeyEvent evt)	Evento de control de escritura automática en mayúsculas en el cuadro de inserción de instrucción G
		private void swap(int a, int b)	Intercambia elementos a y b de una lista
		static void main(java.lang.String[] args)	Función principal
		private void *<74través74ac>*ActionPerformed(ActionEvent evt)	Eventos de acción sobre cada uno de los componentes de la interfaz
	Punto – PuntoR2	public float getX(), getY(), getZ()	Devuelve el valor real de la componente correspondiente

		public void igualar(Punto p2)	Iguala las componentes del punto a las del parámetro Punto p2
		public void setCero()	Establece a 0 las componentes del punto
		public void setX(float x), setY(float y), setZ(float z)	Establece al valor entero especificado en el parámetro la componente correspondiente del punto
Consola	Analizador	public 75ravés75 ComprobarSintaxis(String cadena)	Analiza el léxico y la sintaxis del parámetro cadena, devolviendo 'true' o 'false' si es correcto o no.
		Public 75ravés75 esNumerico(String cadena)	Comprueba si el parámetro cadena corresponde a un valor numérico entero o real.
	Ejecucion	public boolean EjecutarInstruccion(String cadena)	Analiza la semántica del parámetro cadena y si es correcta, ejecuta la funcionalidad asociada. Devolverá 'true' o 'false' en función de la ejecución correcta o no.
Representacion	Representacion	public void display(GLAutoDrawable g)	Función de representación de OpenGL
		public void displayChanged(GLAutoDrawable g, 75ravés75 arg1, 75ravés75 arg2)	Función de evento por cambio en el hardaware gráfico de OpenGL
		public void init(GLAutoDrawable g)	Función de inicialización de OpenGL
		public void reshape(GLAutoDrawable g, int x, int y, int width, int height)	Función de redimensión en el marco de representación de OpenGL
	Redibujado	public void asignar_color(float red, float green, float blue)	Asigna los valores RGB del color de la trayectoria
		public void asignar_color(Redibujado objeto)	Asigna el color definido en el parámetro objeto
		public void asignar_plano(int pl)	Define el plano en base al parámetro pl.
		public void asignar_punto(Punto pto)	Inserta el punto correspondiente al parámetro pto
		public void asignar_radio(float rad)	Asigna el radio de la línea en base al parámetro rad
		public int getPlano()	Devuelve el valor entero correspondiente al plano de mecanizado
Geometria Analitica	FMatematicas	public float CalculaAngulo(Punto R2 origen, PuntoR2 fin, float radio, int plano)	Calcula la coordenada polar del punto origen.
		Public PuntoR2 calculaCentro(PuntoR2 origen, PuntoR2 fin, float radio)	Calcula el centro de la interpolación circular en base al radio y a los puntos origen y fin
		public float calculaDistancia(PuntoR2 origen, PuntoR2 fin)	Calcula la distancia entre los puntos origen y fin
		public float calculaRadio(PuntoR2 origen, PuntoR2 centro)	Calcula el radio de la interpolación circular dado el punto origen y el centro.

		Public Punto calculaPuntoMedio(Punto origen, Punto fin, int tipo_compensacion, float valor_compensacion)	Calcula el punto medio del parámetro fin en base a la pendiente de la recta y el valor de la compensación
		public void CalculaPuntosCompensadosRecta (Punto origen, Punto fin, Punto c_origen, Punto c_medio1, int 76través76ación, float valor_compensacion, int plano)	Calcula los puntos compensados de la trayectoria compensada simple de la interpolación lineal, devolviéndolos a 76través de c_origen y c_medio1.
		Public Punto CalculaInterseccionEntreRectas(Punto origen, Punto medio1, Punto medio2, Punto fin, int plano)	Calcula la intersección entre dos rectas.

Tabla 4.5: Relación de clases y sus métodos.

4.4.4. Analizador léxico-sintáctico-semántico.

Uno de los elementos fundamentales para la correcta ejecución de las instrucciones en Código G consiste en el analizador léxico-sintáctico-semántico adaptado a las necesidades de mecaCNC.

Entre los requisitos fundamentales vistos a lo largo de este documento, se encuentra la ejecución en tiempo real de las instrucciones. Esta característica no podría ser llevada a cabo sin la implementación de un analizador de instrucciones, ya que, considerando que la programación ISO-6983 y el resto de lenguajes empleados para el desarrollo del proyecto son completamente diferentes, es necesario transformar cada instrucción en el conjunto de instrucciones Java y OpenGL que simularán la ejecución de la instrucción G en el simulador.

Por otra parte, el analizador tiene una funcionalidad implícita que consiste en el análisis de posibles errores léxicos, sintácticos o semánticos en las instrucciones G insertadas, evitando así errores de ejecución y manteniendo al sistema y al usuario informado cuando sea conveniente, indicando al dónde está el error y de qué manera se ha de corregir.

A nivel de análisis, podemos identificar las partes de una instrucción como identificadores (léxico), tokens (sintáctico) e instrucciones (semántico).

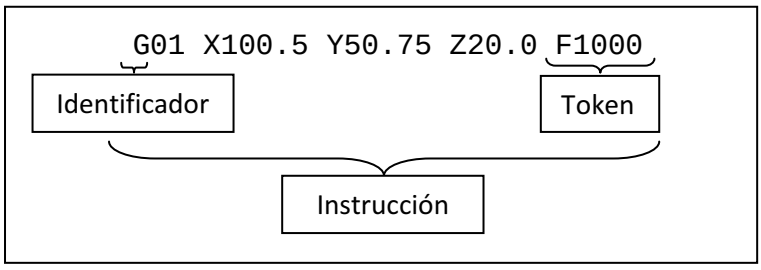


Figura 4.17: Partes de una instrucción ISO a nivel de análisis léxico-sintáctico-semántico

Resulta evidente comprobar que, al fin y al cabo, una instrucción ISO supone una cadena de caracteres separados por espacios en blanco, finalizando en un retorno de línea. Para realizar un análisis de este tipo, Java brinda una potente herramienta en forma de clase, el *StringTokenizer*, que recibirá como parámetro una cadena de caracteres y la dividirá en tokens delimitados por los espacios en blanco y el retorno de línea. Además, cada token es en sí otra cadena de caracteres y, por tanto, podremos acceder a sub-cadenas dentro del propio token.

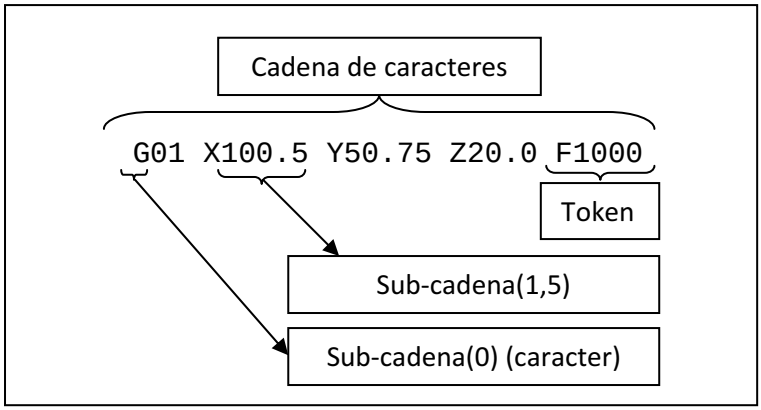


Figura 4.18: El uso de la clase StringTokenizer.

4.4.4.1. Análisis léxico.

El análisis léxico evalúa a nivel de identificador (que en lenguaje común consistiría en la primera letra), que para el caso que nos ocupa, supondrá una letra mayúscula. La fase de análisis léxico será la más simple de las tres.

Las condiciones necesarias a nivel léxico para que la instrucción no devuelva error serán:

Número	Condición
1	Toda instrucción ISO deberá empezar por el identificador 'N'
2	El conjunto de identificadores posibles es {D,F,I,J,K,G,M,N,R,S,T,X,Y}

Tabla 4.6: Condiciones en el análisis léxico.

La implementación del analizador léxico se llevará a cabo a trabes de la clase *Analizador* del paquete consola.

4.4.4.2. Análisis sintáctico.

El análisis sintáctico evalúa a nivel de token (en el lenguaje común supondría una palabra). Sus condiciones serán:

Número	Condición
1	No puede existir más de un token con el mismo identificador en una instrucción.
2	Tras el identificador en un toquen, únicamente puede figurar un valor numérico entero o real.
3	Tras el identificador 'T' ha de figurar un número natural entre 0 y 9, correspondiente al portabrocas, y dicho portabrocas ha de tener una herramienta asignada.
4	Tras el identificador 'D' ha de figurar un número natural entre 0 y 9, y en la tabla de correctores deberá haber un corrector definido en dicha fila.
5	Tras el identificador 'F', ha de figurar un entero positivo entre 0 y 4000.
6	Tras el identificador 'S', ha de figurar un natural entre 150 y 5000
7	Únicamente serán válidos los tokens {X<x>, Y<y>, Z<z>, I<i>, J<j>, K<k>, R<r>, S<s>, F<f>, D<d>, G00, G01, G02, G03, G70, G71, G90, G91, G17, G18, G19, G94, G95, M06, G54, G55, G56, G57, M03, M04, M05, M07, M08, M09, M71, M72, G40, G41, G42, M02, M30}

Tabla 4.7: Condiciones en el análisis sintáctico.

Dada la complejidad en la comprobación de todas las condiciones especificadas en la *Tabla 4.7*, la implementación del analizador sintáctico se llevará a cabo entre la clase *Analizador* y la clase *Ejecucion*, facilitando la fluidez en la ejecución, reduciendo el coste computacional y mejorando la cohesión entre los distintos módulos.

4.4.4.3. Análisis semántico

El análisis semántico supone la evaluación a nivel de instrucción. El objetivo de este análisis radica en la coherencia y cohesión de la instrucción en base a su tipo y sus parámetros. El analizador semántico recogerá todas las condiciones que no hayan sido evaluadas por los anteriores sistemas.

Número	Condición
1	La instrucción G00 recibirá como parámetro, al menos un token del conjunto {X<x>, Y<y>, Z<z>}
2	La instrucción G01 recibirá como parámetro, al menos un token del conjunto {X<x>, Y<y>, Z<z>} y opcionalmente el token 'F<f>'
3	Las instrucciones G02 y G03 recibirán como parámetro, al menos un token del conjunto {X<x>, Y<y>, Z<z>} y el token 'R<r>' o al menos un token del conjunto {I<i>, J<j>, K<k>}, en el orden especificado.
4	La instrucción 'T<t>' deberá tener como parámetros el token 'D<d>' y 'M06' en este orden.
5	Las instrucciones X<x>, Y<y> y Z<z>, únicamente podrán emplearse si existe una instrucción de interpolación activa.

Tabla 4.8: Condiciones en el análisis semántico.

CAPÍTULO 5

GEOMETRÍA ANALÍTICA

En el presente apartado, se ha considerado la necesidad de describir los métodos matemáticos en los cuales se apoya mecaCNC para resolver y obtener la representación gráfica de las instrucciones implicadas en el trazado de trayectorias.

Dejando a un lado la trivialidad del trazado de una interpolación lineal simple, que se construirá utilizando funciones nativas de OpenGL en base al punto inicial y final, se llevará al estudio las interpolaciones circulares así como la traslación de las trayectorias cuando se activan las funciones de compensación de radio de la máquina.

Tal y como se ha visto en apartados anteriores, el lenguaje ISO-6983 contempla dos instrucciones principales de movimiento: la interpolación lineal y la interpolación circular en sentido horario (arco cuyo centro se sitúa a la derecha de la trayectoria, en adelante *SR*) y anti-horario (arco cuyo centro se sitúa a la izquierda de la trayectoria, en adelante *SCR*).

5.1. Interpolación lineal.

En el caso de la interpolación lineal, su trazado en el panel *Representación* es simple: en la propia instrucción se indicará el punto final, y el punto inicial o de partida consistirá en la ubicación actual del usillo (N), por tanto, basta con definir dichos puntos como vértices inicial y final, y trazar la recta que los une. Esto se consigue gracias a las funciones nativas de OpenGL *glVertex3f(float x, float y, float z)*, indicando que dichos vértices componen una recta o secuencia de rectas con *glBegin(GL_LINE_STRIPS);*

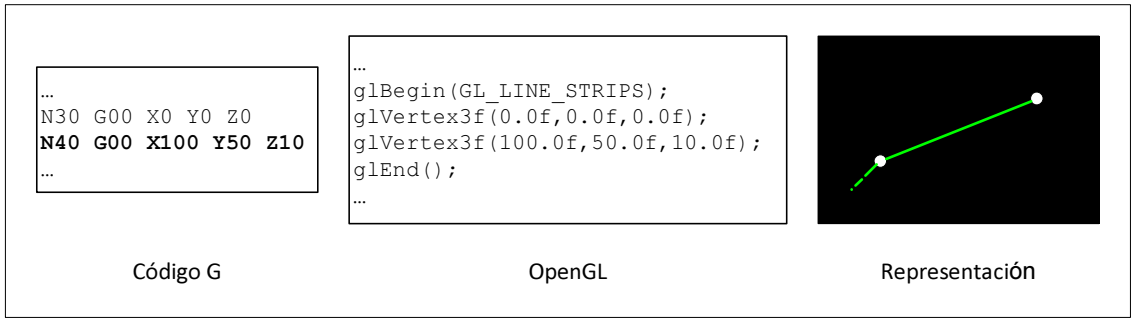


Figura 5.1: Interpolación lineal.

5.2. Interpolación circular.

Para el caso de las interpolaciones circulares, la representación será más compleja.

Simplificando al extremo, cualquier representación gráfica se podrá definir como el conjunto de puntos que la componen. OpenGL concibe, de modo generalista, toda representación gráfica como el conjunto de puntos (GL_POINTS), rectas (GL_LINES) o polígonos (GL_TRIANGLES y GL_QUADS) que podrán dar lugar a cualquier tipo de forma.

Como se puede comprobar, en esta ordenación de las primitivas básicas de OpenGL quedan fuera aquellas formas curvas, tales como los arcos que describen las instrucciones de interpolación circular del código ISO-6983, siendo por tanto necesaria la aproximación de dicha forma a un conjunto de segmentos, de una longitud suficientemente pequeña, como para ser imperceptibles al ojo humano, interpretándose visualmente como una curva.

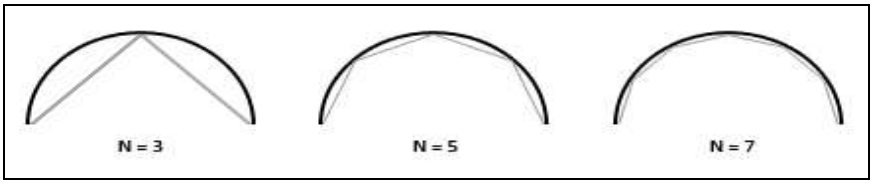


Figura 5.2: Aproximación al arco por N puntos y N-1 segmentos.

Observando la Figura 5.2, podemos afirmar que cuantos más segmentos definan el arco, menor será la longitud de cada uno de ellos y, consecuentemente, obtendremos una mejor aproximación a la curva.

Cabe destacar que **las instrucciones de interpolación circular únicamente trazarán trayectorias en el plano de mecanizado**, sin cambiar la altura entre los puntos inicial y final, y, por tanto, para facilitar el desarrollo y el cálculo, se realizará una conversión previa de puntos $\mathbb{R}3 \rightarrow$ (X, Y, Z) a puntos en $\mathbb{R}2 \rightarrow$ (X, Y), cuyas coordenadas dependerán del plano de mecanizado en uso: XY, XZ o YZ.

5.2.1. Cálculo de la aproximación al arco mediante coordenadas polares.

Como se ha tratado en el apartado anterior, la representación gráfica del arco consiste en una aproximación a la curva en segmentos. Dada la naturaleza curvilínea del arco, se considera oportuna la resolución de dicha aproximación mediante el uso del sistema de coordenadas polares.

Según R.G. Brown, *"el sistema de coordenadas polares es un sistema de coordenadas bidimensional en el cual cada punto o posición del plano se determina por un ángulo y una distancia.*

De manera más precisa, todo punto del plano corresponde a un par de coordenadas (r, θ) donde r es la distancia del punto al origen o polo y θ es el ángulo positivo en sentido anti-horario medido desde el eje polar 0 (equivalente al eje x del sistema cartesiano).” [3]

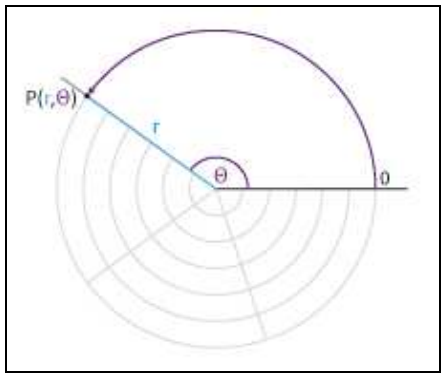


Figura 5.3: Representación en el sistema polar del punto P.

Siguiendo esta notación, se podrá recorrer el arco angularmente en incrementos adecuados (para el proyecto que nos ocupa, se ha hecho uso de incrementos $\Delta = 0.01$), realizando una posterior conversión punto del sistema polar al cartesiano.

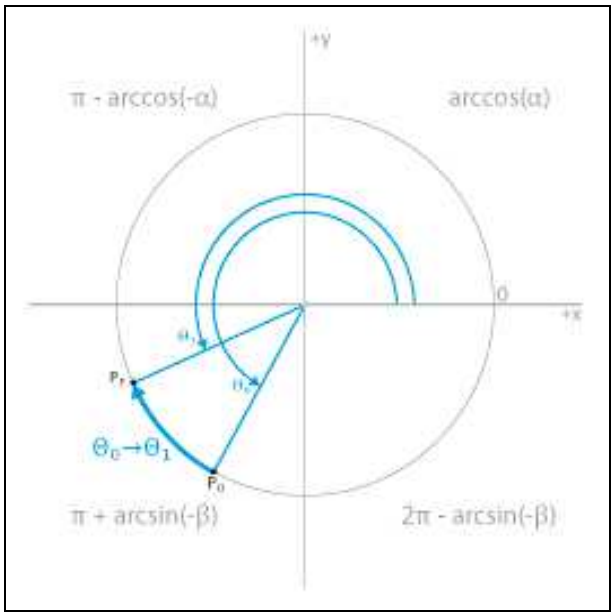


Figura 5.4: Recorrido del arco SR en el sistema polar para plano XY.

Como puede comprobarse en la figura 5.4, el recorrido angular en sentido horario implicará calcular el ángulo (θ) para los puntos inicial y final, obteniendo los valores θ_0 y θ_1, por tanto, el recorrido angular se realizará entre los valores θ_0 y θ_1 en

incrementos $\Delta = 0.01$. Si el ángulo inicial θ_0 fuera menor que el ángulo final θ_1, será necesario recorrer desde θ_0 hasta $\theta_1+2\pi$ (θ_1 + una vuelta completa), análogamente al ejemplo de la *Figura 5.5*, pero en sentido inverso.

Obtenidos dichos puntos auxiliares en el sistema polar, bastará con convertirlos al sistema cartesiano, trasladando el centro de coordenadas al centro del arco:

$$\left.\begin{aligned} auxiliar_x &= centro_x + radio \cdot \cos(i_j) \\ auxiliar_y &= centro_y + radio \cdot \sin(i_j) \end{aligned}\right\} \forall i_j \in [\theta_0, \theta_1] \,/\, i_j = \theta_0 + j \cdot 0.01$$

Fórmula 5.1: Cálculo de los vértices de los segmentos que aproximan al arco.

Para el caso de la interpolación circular SCR, el proceso será análogo al anterior, variando ciertos aspectos: si $\theta_0 > \theta_1$, recorreremos el arco desde θ_0 hasta θ_1 decrementando en $\nabla = 0.01$, y en caso contrario ($\theta_0 < \theta_1$) desde θ_0 hasta $\theta_1-2\pi$ también en decrementos $\nabla = 0.01$.

Si se aplica el ejemplo anterior a la interpolación circular SCR, obtendremos:

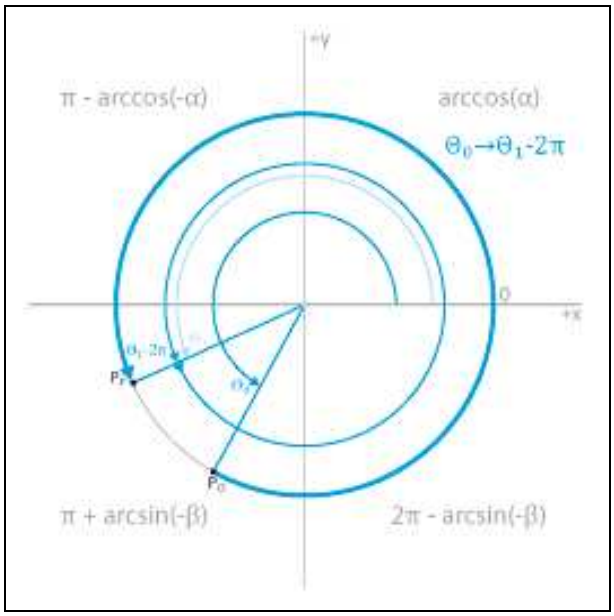


Figura 5.5: Recorrido del arco SCR en el sistema polar para plano XY.

Nótese que el cálculo del ángulo desde el origen polar hasta el punto en sentido anti-horario implica una comprobación adicional para establecer correctamente el intervalo angular: será necesario conocer el cuadrante en el que se ubica el punto, y de esta forma hacer uso de la ecuación de conversión cartesiano-polar adecuada.

Además, dichas ecuaciones de cálculo del ángulo en base al cuadrante también serán distintas en función del plano de mecanizado (XY, XZ o YZ), pero su determinación subyace del mismo desarrollo en distinto plano, por tanto, se omitirá la explicación para cada caso, pero se indicará dichos valores en las siguientes figuras:

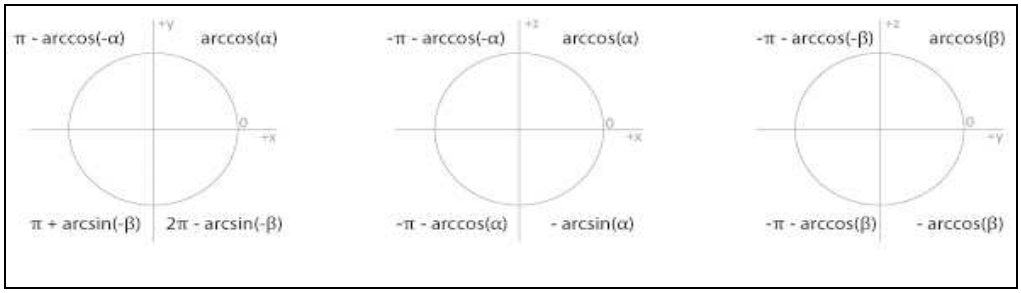


Figura 5.6: Operaciones a aplicar en base al cuadrante y plano de mecanizado.

$$\alpha = \frac{(origen_x - centro_x)}{radio} \qquad \beta = \frac{(origen_y - centro_y)}{radio}$$

Fórmula 5.2: Cálculo del cuadrante en base al plano.

5.2.2. Cálculo del centro del arco dados los puntos origen y final y el radio.

Considerando que los parámetros de la instrucción de interpolación consistan en el punto final y el radio, será necesario calcular el centro del arco, que variará a su vez en función de si la interpolación circular es SR o SCR.

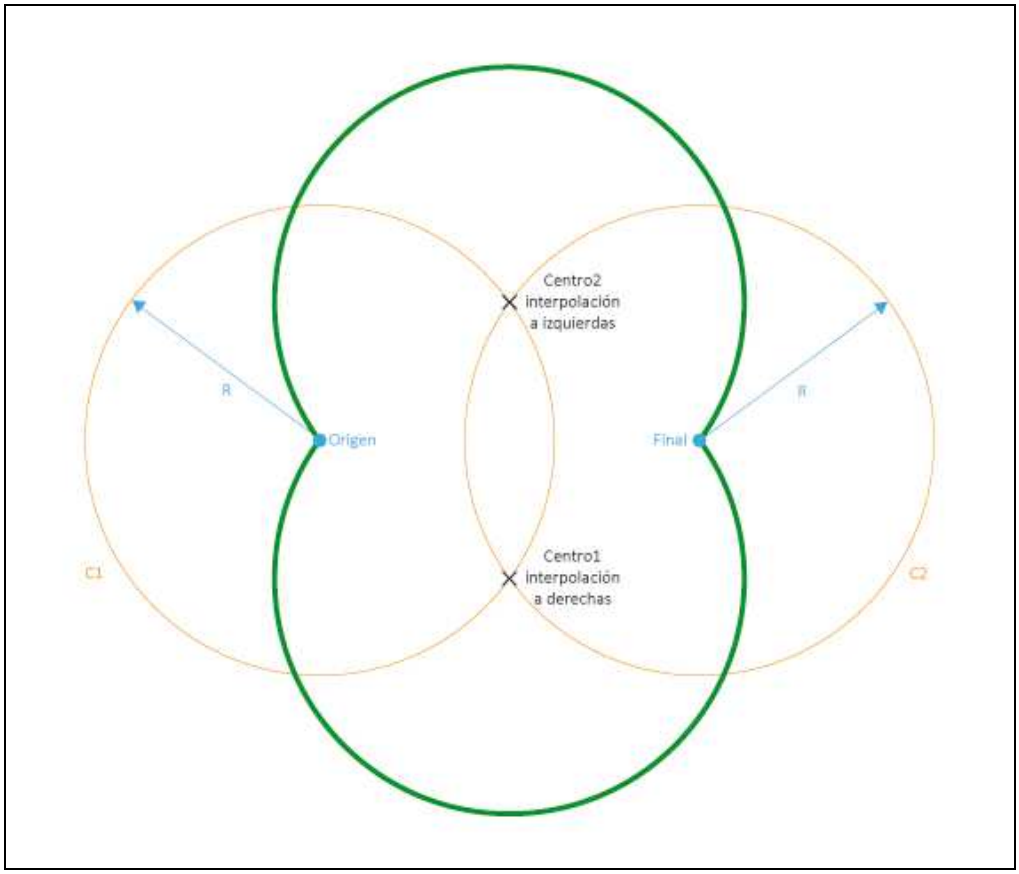


Figura 5.7: Cálculo del centro del arco en función del radio.

Para ello, el proceso que se ha implementado consiste en calcular, para el punto ***Origen*** y ***Final***, la ecuación de la circunferencia (***C1*** y ***C2***) con centro en cada punto y radio ***R***.

$$C1 \Rightarrow (x - Origen_x)^2 + (y - Origen_y)^2 = R^2$$

Fórmula 5.3: Ecuación de la circunferencia C1.

Como se puede comprobar en la figura, dichas circunferencias se cortan en los puntos ***Centro1*** y ***Centro2***, los cuales supondrán el punto centro del arco para la interpolación circular SR o SCR respectivamente.

Una vez hallados los posibles centros, quedaría por determinar cuál es la pendiente de la recta que une el centro con el origen, y en base a dicha pendiente devolveríamos el punto *Centro1* o *Centro2*.

5.2.3. Cálculo del radio dados los puntos inicial y final y el centro del arco.

Si los parámetros de la instrucción de interpolación circular comprenden el punto final y el centro del arco, será necesario calcular el radio para poder transformar las coordenadas cartesianas en polares.

Dicho cálculo es trivial, ya que el radio equivaldrá a la distancia entre cualquiera de los dos puntos (en mecaCNC se ha tomado el origen) y el centro del arco.

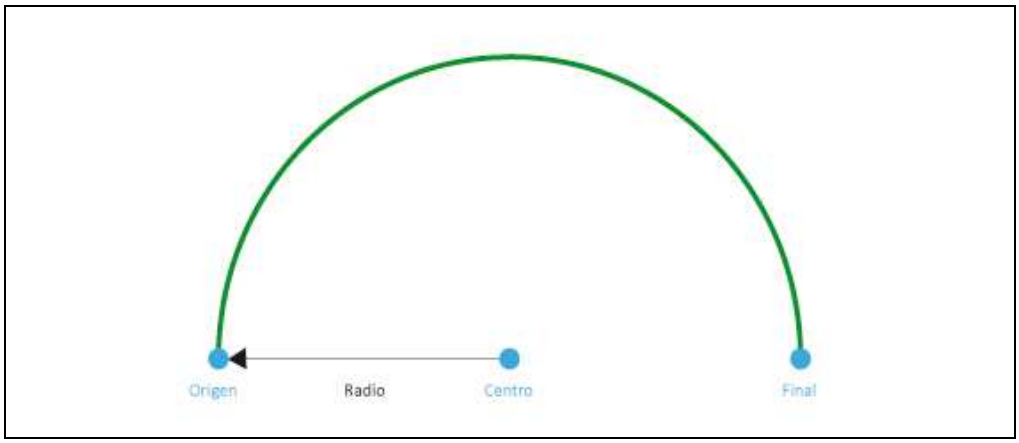


Figura 5.8: Cálculo del radio en función del centro y el punto origen.

La distancia entre ambos puntos se calcula mediante la operación:

$$Radio = \sqrt{(Centro_x - Origen_x)^2 + (Centro_y - Origen_y)^2}$$

Fórmula 5.4: Cálculo del radio en función del punto origen y el centro.

5.2.4. Interpolación circular en base al centro del arco.

En la instrucción se especifica el punto final (X, Y, Z) y el punto centro del arco (I, J, K) a describir, tomando como punto inicial la ubicación del usillo.

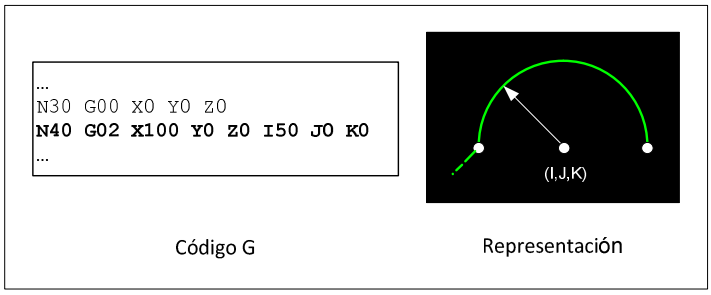


Figura 5.9: Interpolación circular SR en base al centro del arco.

En base al desarrollo de los apartados anteriores, para representar gráficamente dicho arco haremos uso del sistema de coordenadas polares, el cual requiere el radio, por tanto, se calculará dicho valor haciendo uso de los parámetros punto origen (ubicación del usillo N(X,Y,Z)) y el centro (I, J, K).

5.2.5. Interpolación circular en base al radio.

En la instrucción se especifica el punto final (X, Y, Z) y el radio (R), tomando como punto inicial la ubicación del actual del usillo

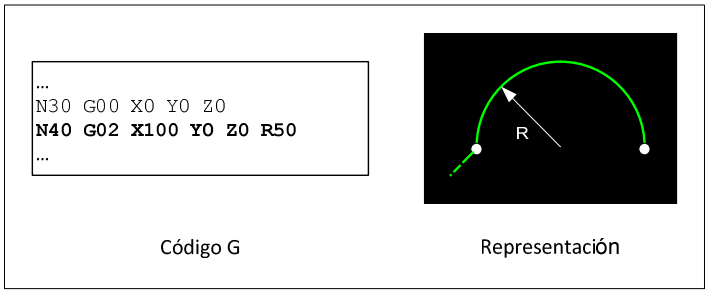


Figura 5.10: Interpolación circular SR en base al radio.

En este caso, calcularemos el centro del arco haciendo uso del radio y los puntos origen y final, y trazaremos el arco de la forma desarrollada en apartados anteriores.

5.3. Compensación de radio.

Otro caso susceptible de análisis matemático consiste en el trazado de trayectorias con la compensación de radio activa.

La activación de la compensación de radio supone la traslación del trazado de la trayectoria en base al radio (R) y desgaste de radio (I) indicado en el corrector asociado a la

herramienta activa. Es decir, si la compensación está activa, se deberá desplazar la herramienta una distancia equivalente a la suma de los parámetros del corrector (R+I).

El desplazamiento se realizará a izquierdas o derechas de la trayectoria sin compensación, según se haya introducido la instrucción de compensación de radio a izquierdas o derechas.

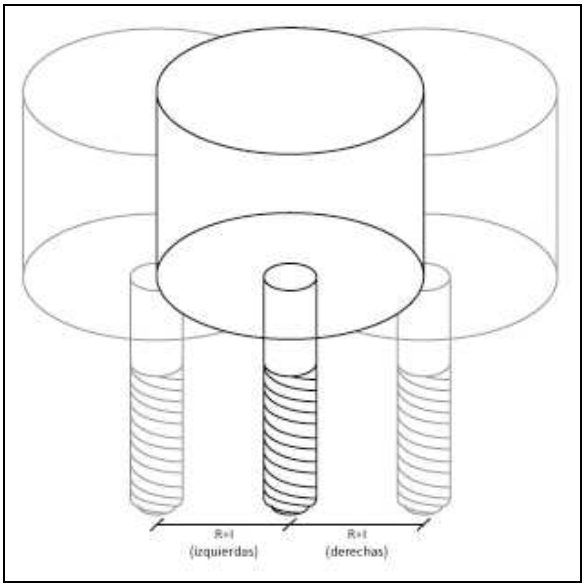


Figura 5.11: Compensación a izquierdas y derechas.

Por último, cabe destacar que **la compensación de radio únicamente es aplicable al plano de mecanizado XY, manteniéndose Z constante.**

5.3.1. Cálculo de la trayectoria compensada en interpolaciones lineales.

Considerando las definiciones previas anteriores, para el caso de instrucciones de interpolación lineal se podrá calcular la trayectoria compensada, trasladada a derecha o izquierda una distancia de R+I milímetros, de la siguiente forma:

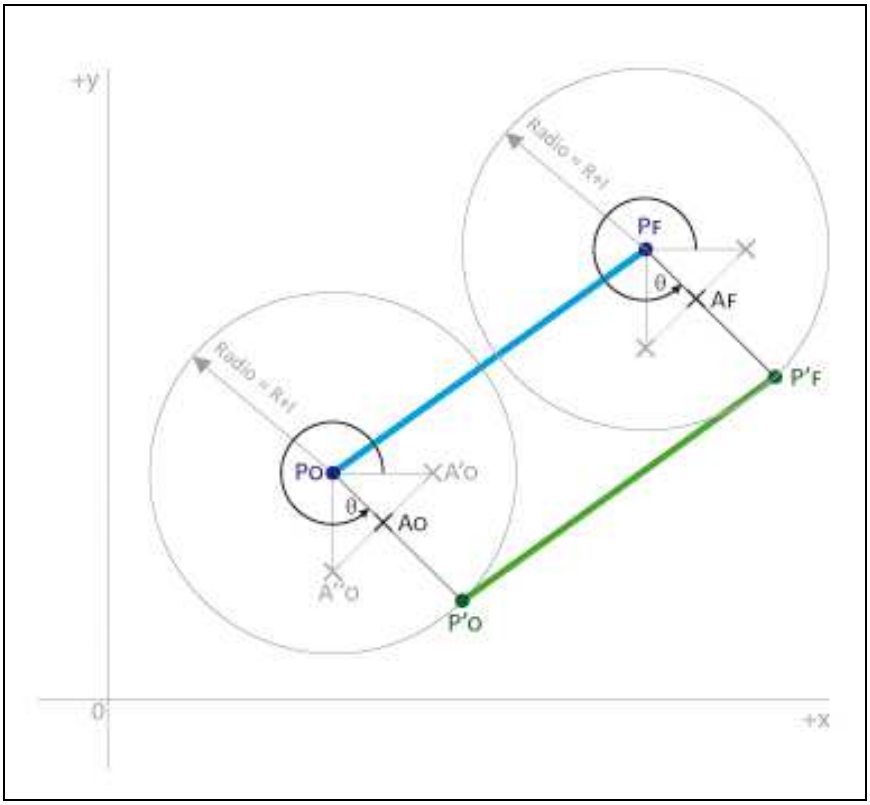


Figura 5.12: Cálculo de los puntos origen y final de la trayectoria compensada.

En la figura podemos identificar los puntos origen (P_O) y final (P_F) así como el radio total de la compensación (*Radio*) resultado de la suma de los parámetros *R* y *I* del corrector asociado. Estos son los datos de que disponemos y que nos permitirán calcular los puntos P'_O y P'_F, correspondientes a la trayectoria compensada.

Para ello, calculamos el punto auxiliar A_O, que consiste en el punto medio entre $A_O' = (P_{Ox}+Radio, P_{Oy})$ y $A_O'' = (P_{Ox}, P_{Oy}-Radio)$, y de manera análoga, el punto A_F.

Si analizamos la *Figura 5.12*, podemos comprobar a simple vista que los puntos auxiliares y los puntos de la trayectoria compensada tienen el mismo ángulo θ, que se podrá calcular haciendo uso de la función θ = *CalculaAngulo*(A_O, P_O, *Radio*, *plano de mecanizado)* descrita en el apartado 5.3.1.

Además, se conoce el valor del *Radio*, por tanto, podemos obtener la ecuación en el sistema polar, y, mediante conversión al sistema cartesiano y desplazamiento del centro del sistema polar a los puntos P_O y P_F, se obtendrá el valor de los puntos P'_O y P'_F:

$$P'_{Ox} = P_{Ox} + (Radio \cdot \cos(\theta)) \qquad P'_{Fx} = P_{Fx} + (Radio \cdot \cos(\theta))$$
$$P'_{Oy} = P_{Oy} + (Radio \cdot \sin(\theta)) \qquad P'_{Fy} = P_{Fy} + (Radio \cdot \sin(\theta))$$

Fórmula 5.5: Cálculo de los puntos Origen y Final de la trayectoria compensada.

Una vez calculados, podremos representar la trayectoria compensada a distancia ***Radio*** trazando la recta entre los puntos P'_O y P'_F.

5.3.2. Cálculo de la trayectoria compensada en interpolaciones circulares.

Tal y como se ha venido haciendo hasta ahora, el cálculo de la trayectoria compensada para la interpolación circular se realizará mediante el uso del sistema polar.

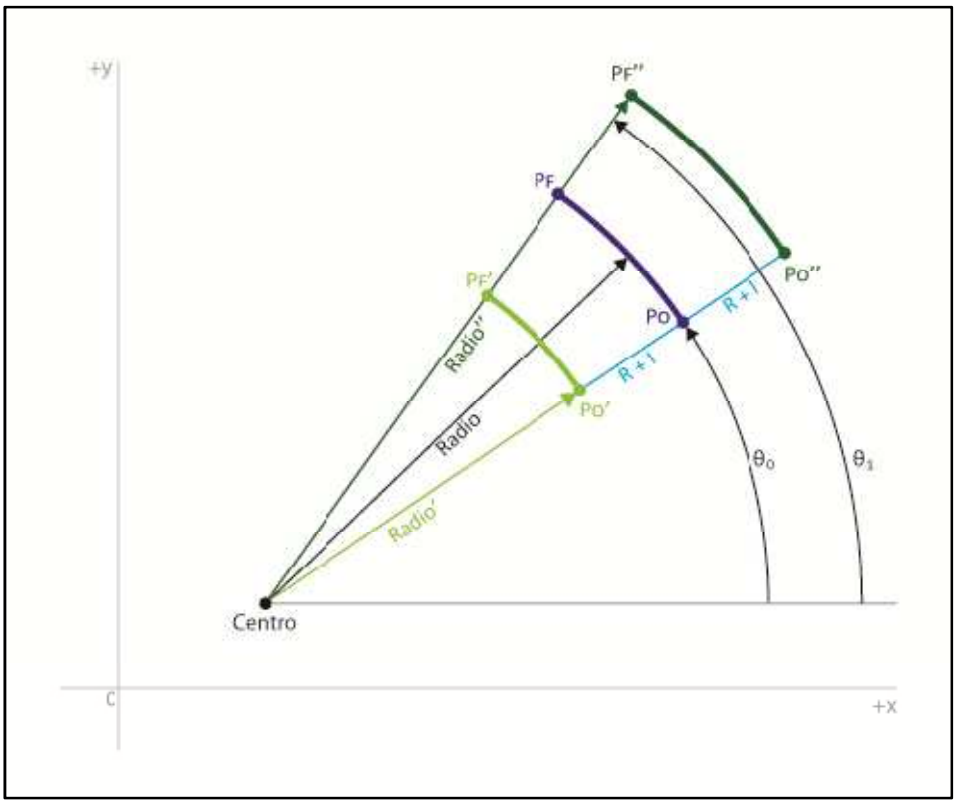


Figura 5.13: Cálculo de las trayectorias compensadas de la instrucción de interpolación circular SCR.

En la *Figura 5.13* podemos observar los datos de que disponemos: el ***Centro*** del arco, los puntos Origen (P_O) y Final (P_F) y la distancia de la compensación ***(R + I)***. Mediante estos parámetros y siguiendo los procedimientos anteriores estaremos en disposición de obtener el ***Radio*** de la trayectoria sin compensar ($P_O \rightarrow P_F$).

Mediante la función *CalculaAngulo* sobre los parámetros P_O, ***Centro*** y ***Radio***, obtenemos el ángulo θ_0, que como podemos comprobar, será idéntico para los puntos P_O' y P_O''. De forma análoga, obtendremos el ángulo θ_1 para los puntos finales P_F' y P_F''.

Para obtener la representación polar de los puntos compensados, únicamente restará por calcular los radios (***radio'*** para la compensación a izquierdas y ***radio''*** para derechas). Gráficamente, se puede observar que estos valores se obtendrán mediante ***Radio*** $\pm$ ***(R + I)***, dependiendo del tipo de compensación e interpolación (izquierdas o derechas, SR o SCR).

Para concluir, realizaremos una conversión de los puntos del sistema polar al cartesiano, trasladando el centro del sistema polar al punto ***Centro***, y obteniendo así los puntos inicial y final de la trayectoria compensada:

$$P_O'(x) = Centro(x) + radio' \cdot \cos(\theta_0) \qquad P_O''(x) = Centro(x) + radio'' \cdot \cos(\theta_0)$$
$$P_O'(y) = Centro(y) + radio' \cdot \sin(\theta_0) \qquad P_O''(y) = Centro(y) + radio'' \cdot \sin(\theta_0)$$
$$P_F'(x) = Centro(x) + radio' \cdot \cos(\theta_1) \qquad P_F''(x) = Centro(x) + radio'' \cdot \cos(\theta_1)$$
$$P_F'(y) = Centro(y) + radio' \cdot \sin(\theta_1) \qquad P_F''(y) = Centro(y) + radio'' \cdot \sin(\theta_1)$$

Fórmula 5.6: Cálculo de los puntos Origen y Final para trayectorias compensadas a izquierda y derecha en interpolaciones circulares

Obtenidos los puntos, trazamos la trayectoria haciendo uso de la función de interpolación circular. La interpolación circular SR seguirá el mismo procedimiento.

5.4. Cálculo de la trayectoria compensada en una secuencia de interpolaciones.

Uno de los puntos clave y que reviste mayor complejidad en el proyecto que nos ocupa consiste en el cálculo de la trayectoria compensada de una secuencia de interpolaciones.

En los sistemas de mecanizado reales, el análisis y cálculo de las trayectorias de mecanizado permite predecir la intencionalidad del programa de mecanizado, es decir, una interpolación compensada calculará el punto adecuado en base a la dirección de la siguiente instrucción.

Este hecho nos lleva a una limitación en el diseño del programa: en el cálculo de la trayectoria de una secuencia de interpolaciones compensadas, será necesario anticipar la siguiente instrucción para obtener los puntos correctos de dicha trayectoria, por tanto, la representación gráfica de la trayectoria compensada no podrá ser en tiempo real.

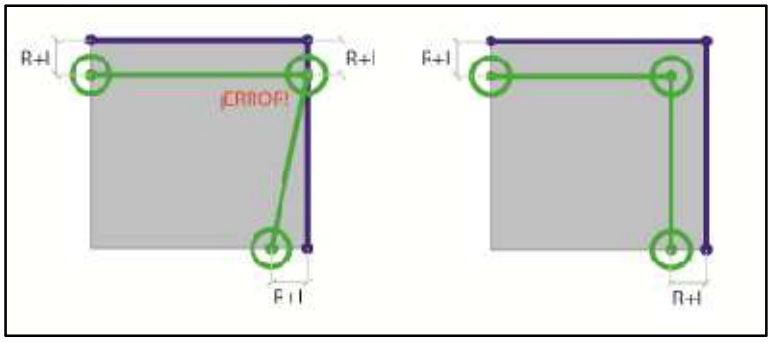


Figura 5.14: Compensación sin anticipación vs. con anticipación.

A fin de resolver esta limitación, se acuerda que, al detectar la activación de la compensación y hasta detectar la instrucción de desactivación, se representará la instrucción de interpolación n-1, pudiendo así disponer del conjunto de puntos adecuado para poder estimar la trayectoria compensada correcta.

Por otra parte, existen ciertos casos especiales que aportan un grado aún mayor de dificultad: **los casos conflictivos**. Se considerarán casos conflictivos aquellos que, por la geometría de la trayectoria, su cálculo lleve a varias posibles soluciones.

Figura 5.15: Tipos de resolución de conflictos en la compensación.

Actualmente, no existe un consenso sobre qué solución adoptar en caso de conflicto, dependiendo únicamente del control numérico y como este haya sido programado, por tanto, mostrando resultados dispares en base a la máquina CNC, si bien es cierto que la mayor parte de éstas resuelven el conflicto mediante el corte de las trayectorias compensadas (primer gráfico de la *Figura 5.15*).

En el proyecto que nos ocupa, existen tres factores decisivos para adoptar como solución el corte de trayectorias compensadas:

1. La naturaleza docente del programa permite al usuario entender, adoptando esta solución, que la compensación consiste en el desplazamiento de los puntos que está insertando una distancia R+I a izquierda o derecha, generando así un escalado uniforme de las interpolaciones programadas.

2. Esta solución es la que respeta en mayor medida la geometría de la trayectoria, ya que se obtiene una trayectoria idéntica desplazada en el factor de compensación.

3. La máquina EMCO PC Mill 125 resolverá los conflictos empleando esta metodología.

Determinada por tanto la solución a implementar, se calcularán los puntos que definen la trayectoria compensada en una secuencia de interpolaciones hallando los puntos de corte de las trayectorias compensadas simples, detalladas en los apartados *5.3.1* y *5.3.2* del presente capítulo.

5.4.1. Cálculo de la trayectoria compensada en una secuencia de interpolaciones lineales.

El primer caso de análisis consistirá en la inserción de varias instrucciones de interpolación lineal con la compensación activada.

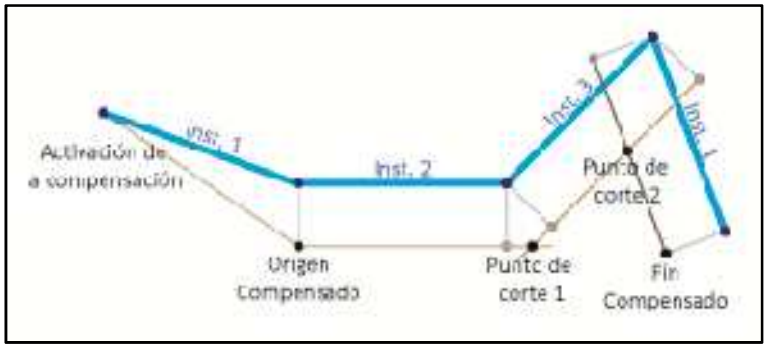


Figura 5.16: Cálculo de los puntos de corte de las trayectorias compensadas simples a derechas en interpolaciones lineales.

En la *Figura 5.16* podemos observar, en distintos tonos de marrón, lo que supondría la trayectoria compensada simple para cada una de las instrucciones de interpolación (***Inst.1-4***), calculadas según el procedimiento descrito en el apartado *5.3.1.*

Obtenidos los puntos de cada trayectoria compensada simple, podremos definir la ecuación de cada una de las rectas que componen, pasando por ambos puntos.

$$recta \rightarrow \frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \,/\, (x_1, y_1), (x_2, y_2)\ puntos\ compensados$$

Fórmula 5.7: Ecuación de la recta que pasa por los puntos compensados simples.

Por cada par de puntos, obtenemos la ecuación de su recta. Igualando dichas rectas y despejando las componentes (se ha considerado despejar '*y*' en función de '*x*'), obtendremos el punto de corte entre ambas rectas que supondrá el punto de la trayectoria mecanizada compensada. Este conjunto de operaciones vendrá definido en una única función ***CalculaInterseccionEntreRectas***, que recibirá como parámetros los 2 puntos que forman cada recta y el plano en el que se está mecanizando.

El procedimiento para la compensación a izquierdas será idéntico al descrito, considerando que los puntos que componen la trayectoria compensada simple tendrán distinta ubicación.

Cabe destacar que, como se puede observar en la *Figura 5.16*, una vez activada la compensación, la primera aproximación se realizará del punto origen al primer punto compensado calculado. El fin de la compensación realizará la operación inversa, desplazándose del punto compensado al punto final sin compensar.

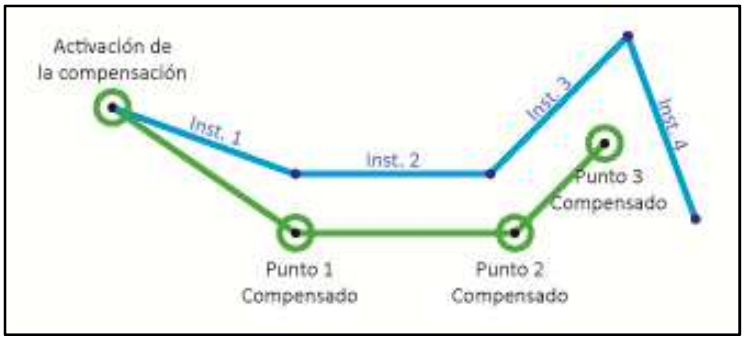


Figura 5.17: Representación gráfica resultado de la trayectoria lineal compensada.

Nótese que en el resultado se representan ***Inst.***-1 tramos compensados, ya que, tal y cómo se explicó anteriormente, mecaCNC no podrá conocer el *Punto 4 Compensado* hasta que se conozca la dirección del último tramo tras insertarse la instrucción ***Inst. 5***.

5.4.2. Cálculo de la trayectoria compensada en una secuencia de interpolaciones circulares.

En el caso de una secuencia de interpolaciones circulares, haciendo uso de funciones de cálculo analizadas en apartados anteriores, podemos obtener fácilmente los puntos de corte que supondrán los puntos conflictivos del mecanizado compensado.

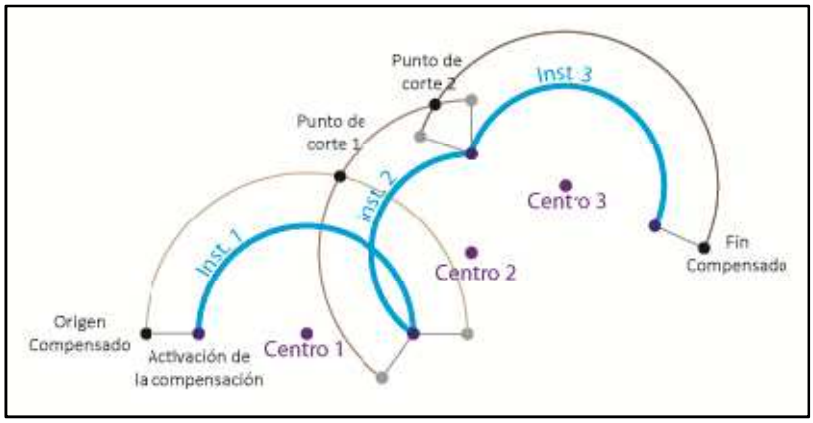


Figura 5.18: Cálculo de los puntos de corte de las trayectorias compensadas simples a izquierdas en interpolaciones circulares SR.

Observando la *Figura 5.18*, se puede comprobar que los puntos de corte se podrán calcular mediante la función descrita en el apartado 5.2.2, estableciendo como puntos inicial y final los puntos *Centro n* y *Centro n+1* y el valor del radio igual al radio de la interpolación más el factor de compensación *R+I*. De esta forma, se puede aprovechar la función de cálculo de centros de la interpolación circular para hallar los puntos de corte que supondrán los puntos de la trayectoria compensada.

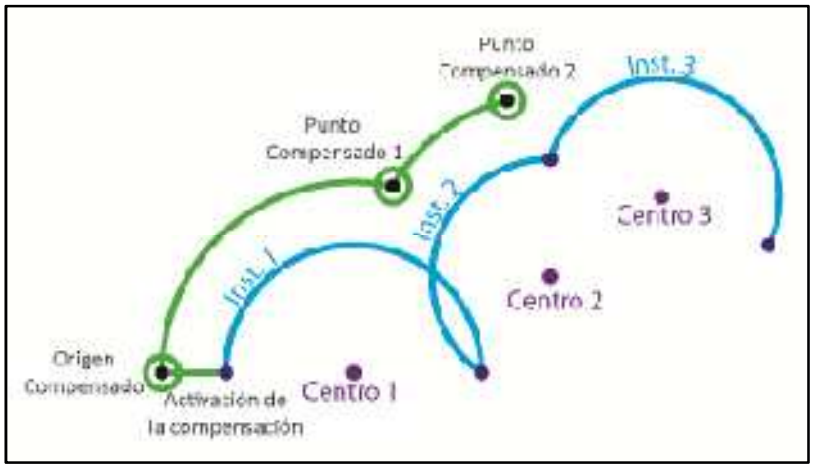


Figura 5.19: Representación gráfica resultado de la trayectoria circular compensada.

En el caso de secuencias de interpolaciones circulares compensadas, para obtener una mayor fidelidad a la geometría de los puntos insertados, en caso de que se inicie la compensación en una instrucción de interpolación circular, la aproximación al *Origen Compensado* se realizará en línea recta desde el punto *Origen* de la interpolación.

Al igual que ocurría en el caso anterior, mecaCNC estará en disposición de representar gráficamente el tramo ***Inst.***-1.

En el caso que las trayectorias compensadas simples no se crucen, el programa devolverá error, tal y como sucedería en el caso presentado en la *Figura 5.19* si la compensación fuera a derechas, al igual que sucedería si se especificara una instrucción de interpolación circular cuyos puntos inicial y final, en conjunción con el radio, conformasen una geometría imposible de resolver.

5.4.3. Cálculo de la trayectoria compensada en una secuencia de interpolaciones mixta.

El último caso de análisis supondrá aquella secuencia de interpolaciones que alterne entre lineales y circulares, que, en adelante, se considerará una trayectoria mixta.

Para este caso, será necesario calcular el punto de corte entre una recta y una circunferencia, lo que conlleva una función de cálculo adicional a las ya vistas hasta ahora.

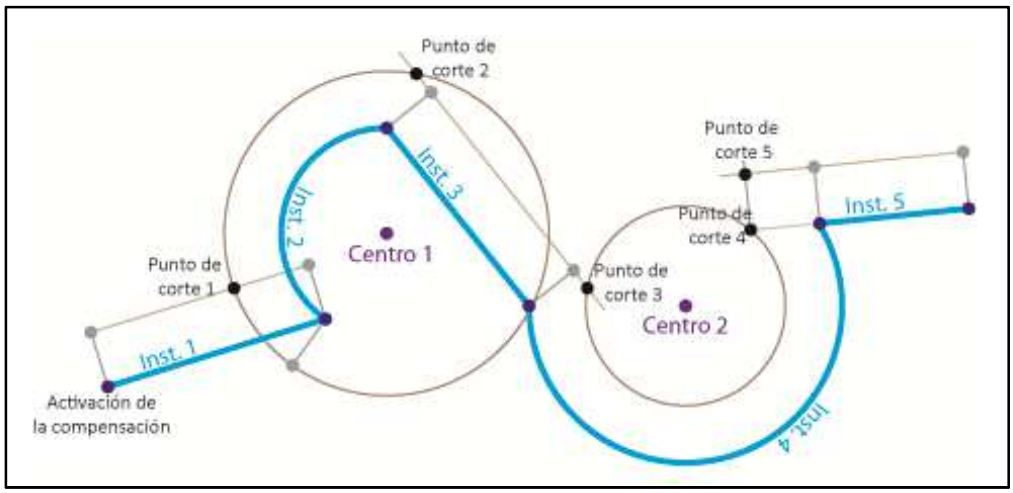


Figura 5.20: Cálculo de los puntos de corte de las trayectorias compensadas simples a izquierdas en interpolaciones mixtas.

Para calcular el punto de corte entre la trayectoria compensada de la instrucción de interpolación lineal y la circular, obtendremos, en primera instancia, la ecuación de la recta y de la circunferencia compensada mediante los métodos descritos anteriormente:

$$recta \rightarrow \frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \,/\, (x_1, y_1), (x_2, y_2)\ puntos\ compensados$$

$$circunferencia \rightarrow (x - Centro_x)^2 + (y - Centro_y)^2 = Radio + (R + I)$$

Fórmula 5.8: Sistema de ecuaciones formado por la recta y la circunferencia compensada.

A continuación, se resolverá el sistema de ecuaciones formado por las ecuaciones anteriores, para el que se obtendrán dos soluciones: el punto de la trayectoria compensada será aquel cuya distancia al punto compensado correspondiente de la interpolación lineal sea menor.

En el caso de que las trayectorias compensadas simples de ambas interpolaciones no se corten, se comprobará si la circunferencia compensada se corta con la trayectoria lineal sin compensar: en caso afirmativo, este punto se añadirá a la trayectoria compensada; en caso contrario, la función devolverá error. Podemos observar este caso entre las instrucciones 4 y 5 de la *Figura 5.20.*

Para la compensación a derechas, se realizará una resolución análoga considerando la nueva ubicación de los puntos.

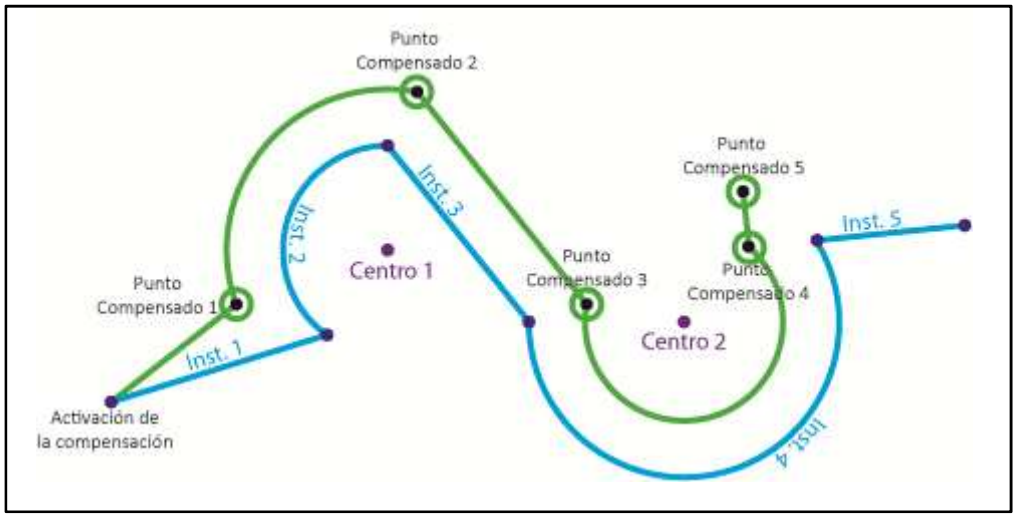


Figura 5.21: Representación gráfica resultado de la trayectoria mixta compensada.

CAPÍTULO 6

EJEMPLO DE USO

En el presente capítulo se establecerá un ejemplo de uso extraído de un ejercicio real de la asignatura Fabricación Asistida por Computador y se resolverá haciendo uso de mecaCNC.

6.1. Enunciado.

Contorno exterior:

· Fresa Ø 4mm. (long. mango = 11.25 mm, long. corte = 33.75mm).

· Profundidad de mecanizado = 10mm.

Contornos interiores:

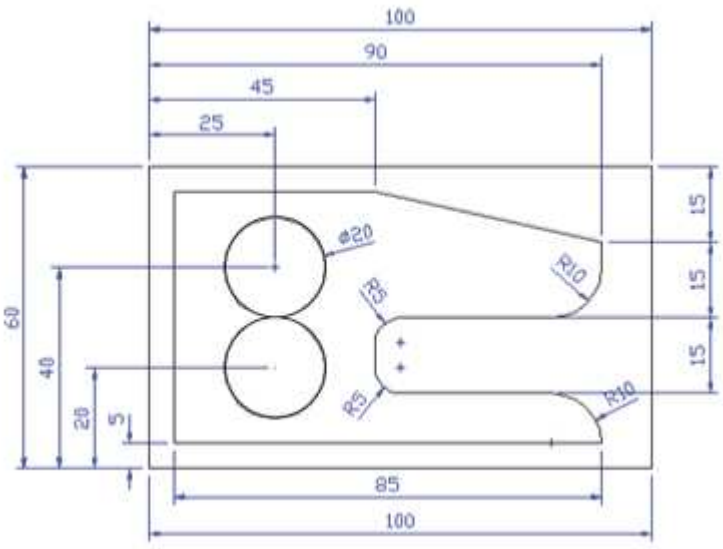
· Fresa Ø 2mm. (long. mango = 9.5 mm, long. corte = 28.5mm).

· Profundidad de mecanizado = 5mm.

Dimensiones de la pieza:

· 100 x 60 x 30

Todas las medidas en milímetros.



6.2. Resolución mediante mecaCNC.

1. En primer lugar, se definirán las herramientas especificadas en el enunciado. Para ello, se accederá a *"Configuración > Configurar Herramientas" (Ctrl+H)*. Una vez mostrada la ventana *Configurar Herramientas,* definiremos las dos fresas, obteniendo la siguiente tabla de herramientas:

ID	Nombre	Tipo	Long. Mango	Long. Corte (L)	Radio (R)
1	Cóncava	Fresa	80	40	6
2	Cilíndrica HSS	Broca	30	50	4
3	Woodruff HSS Diente recto	Fresa	40	40	5,25
4	Widea Cono Morse	Broca	40	30	12
5	Fresa1	Fresa	11,25	33,75	2
6	Fresa2	Fresa	9,5	28,5	1

Figura 6.1: Tabla de herramientas resultante.

2. Asignamos *Fresa1* al primer porta-brocas y *Fresa2* al segundo. Tras la asignación, se podrá cerrar la ventana.

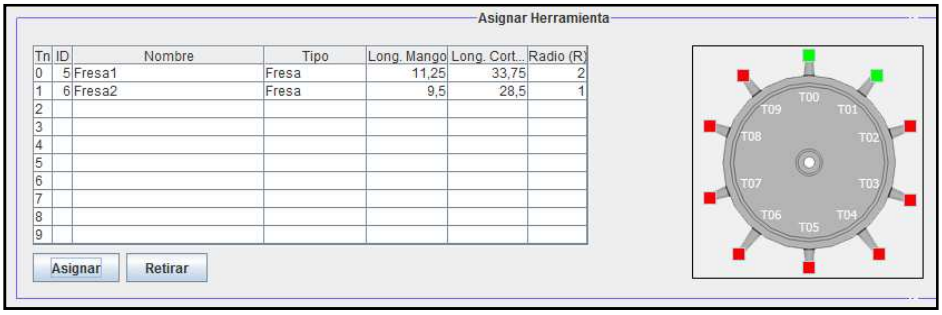


Figura 6.2: Panel Asignar Herramienta resultante.

3. Se define la pieza. Para ello, se accederá a "Configuración > Definir Pieza" (Ctrl+P). Como no se indican datos de la ubicación de la mordaza en el banco de trabajo, dichos valores se establecerán a 0. Al presionar sobre "Definir", de forma automática se establecerá el decalaje G54 en el punto Cero Pieza = (0.0,0.0,30.0).

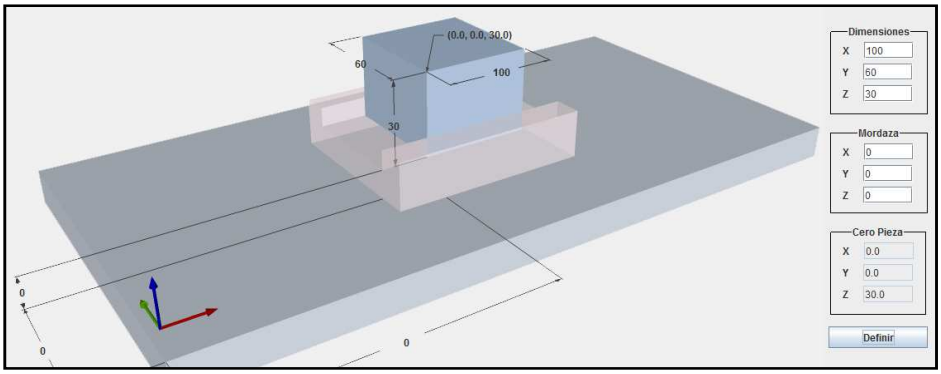


Figura 6.3: Ventana Definir Pieza resultante.

4. Como se puede comprobar en el esquema del enunciado, las medidas parten de dos puntos: el Cero Pieza y el punto (10,10,30). A fin de facilitar la programación, se definirá G55 = (10,10,30), donde 30 equivaldrá a la dimensión de la pieza en el eje Z. Para ello, se accederá a *"Configuración > Definir decalajes" (Ctrl+D).*

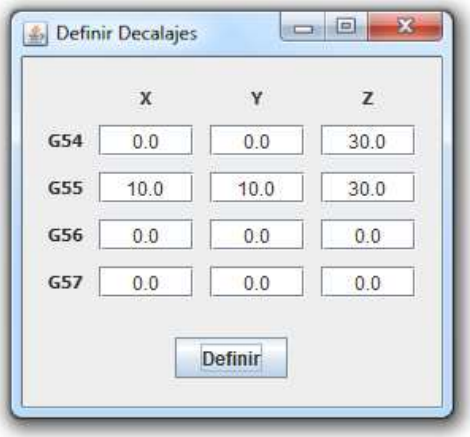


Figura 6.4: Ventana Definir Decalajes resultante.

5. Si se considera adecuado, se podrá guardar un archivo de configuración que incluirá todos las definiciones y asignaciones realizadas hasta este punto desde *"Configuración > Guardar arch. configuración" (Ctrl+G).*

6. A partir de este punto, y al no indicarse valor de correctores, se inicia la inserción de código ISO. En primer lugar, se han de establecer las instrucciones de inicialización que definirán unidades de medida, plano de mecanizado, velocidades, etc. El usuario puede obtener la lista completa de instrucciones y su funcionalidad desde *"Ayuda > Mostrar Ayuda" (F1)* y en la nueva ventana, en la pestaña de ayuda

por contenidos, *"Sistema de ayuda ↳ Código G (ISO-6983) ↳ Lista de instrucciones ISO-6983"*. Insertaremos las instrucciones:

Instrucción	Función
G71	Se establecen las unidades de medida en milímetros
G90	Coordenadas absolutas
G17	Plano de mecanizado XY
G94	Unidades de avance en mm/min.
S5000	Establecemos la velocidad máxima del husillo
T00 D0 M06	Seleccionamos la herramienta para el contorno exterior
G55	Activamos el decalaje G55 =(10.0,10.0,30.0)
M03	Activamos el giro del husillo en dirección SR

Tabla 6.1: Relación de instrucciones de inicialización.

Una vez insertadas las instrucciones de la *Tabla 6.1,* los paneles de la ventana principal mostrarán los siguientes resultados:

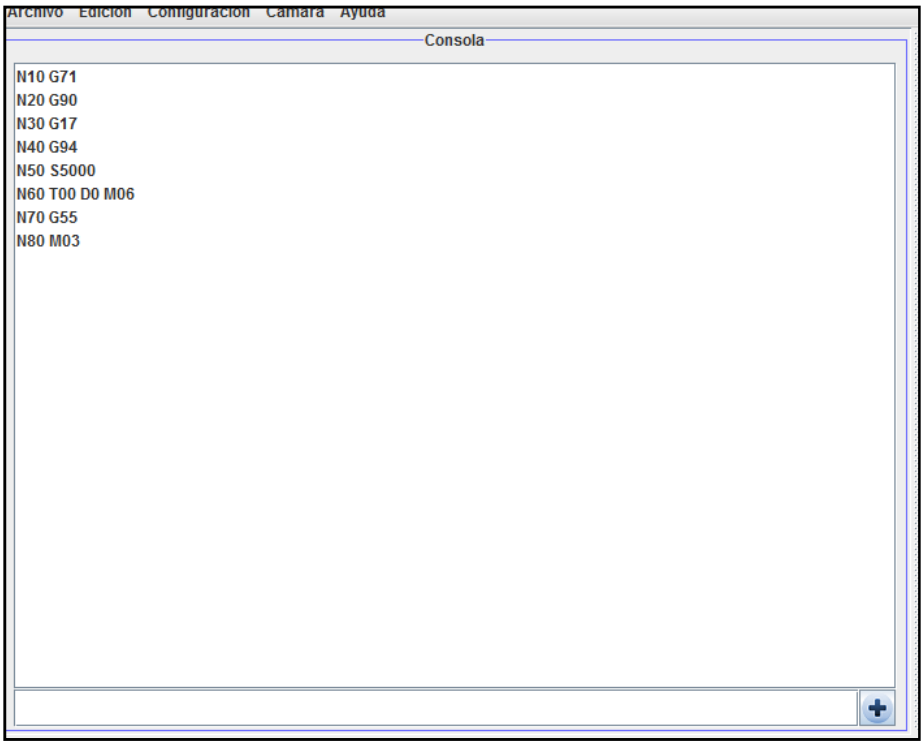


Figura 6.5: Panel Consola resultante.

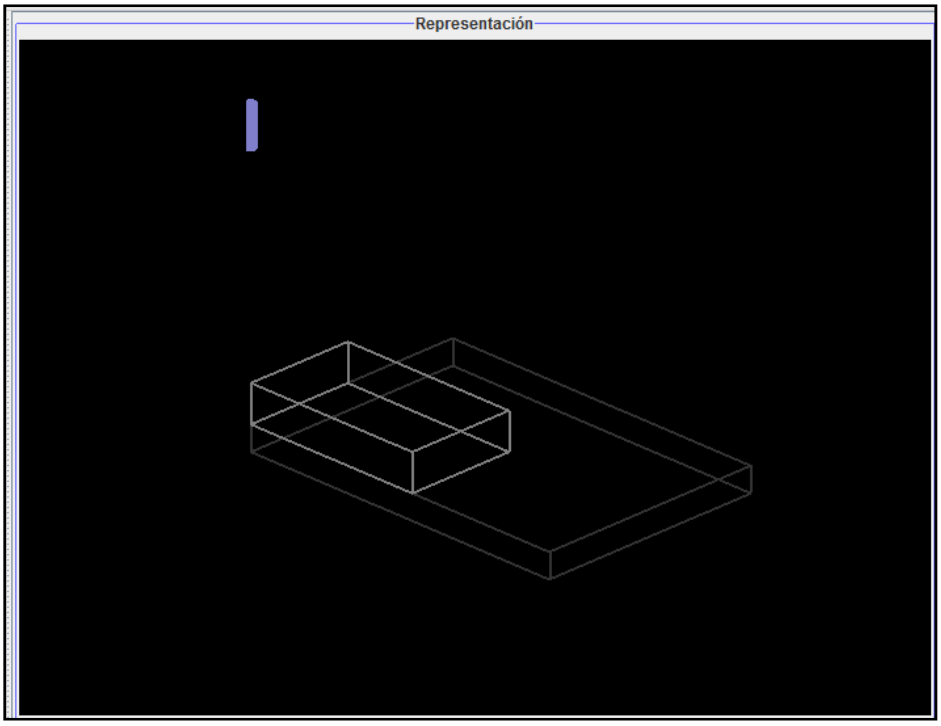


Figura 6.6: Panel Representación resultante.

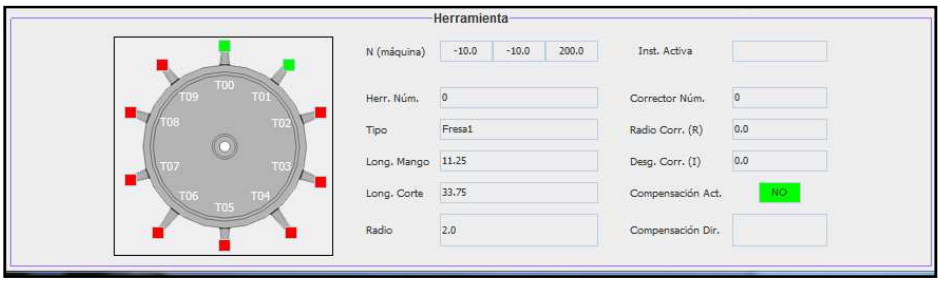


Figura 6.7: Panel Herramienta resultante.

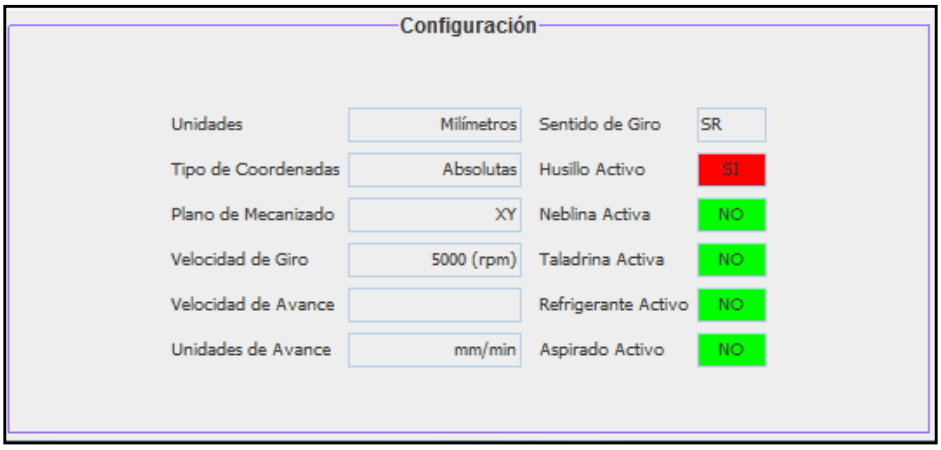


Figura 6.8: Panel Configuración resultante.

7. A partir de este punto, la máquina CNC estará configurada y lista para recibir las instrucciones de control de trayectoria que compondrán el mecanizado, a falta de indicar la velocidad de avance que podrá indicarse haciendo uso de la instrucción G01. La lista de instrucciones que generan el contorno exterior serán:

Instrucción	Función
G01 X0 Y0 Z0 F1000	Aproximamos el husillo al punto (0,0,0). Al haber activado el decalaje G55, realmente nos estaremos aproximando al punto (10,10,30).
Z-10	Nos dirigimos a la profundidad definida en el enunciado para el contorno exterior. Al permanecer activa la instrucción G01, únicamente será necesario indicar la componente que varía, obteniendo un resultado análogo a G01 X0 Y0 Z-10.
X85	Desplazamos en el eje X.
G03 X75 Y10 I-10	Interpolación circular a izquierdas con centro (75,0,-10)
G01 X50 Y10	Interpolación lineal a (50,10,-10)
G02 X45 Y15 R5	Interpolación circular a derechas al punto (45,15,-10) con radio 5
G01 Y20	Interpolación lineal a (45,20,-10)
G02 X50 Y25 R5	Interpolación circular a derechas al punto (50,25,-10) con radio 5
G01 X75	Interpolación lineal a (75,25,-10)
G03 X85 Y35 R10	Interpolación circular a izquierdas al punto (85,35,-10) con radio 10
G01 X85 Y40	Interpolación lineal al punto (85,40,-10)
X40 Y45	Interpolación lineal al punto (40,50,-10)
X0	Interpolación lineal al punto (0,45,-10)
Y0	Interpolación lineal al punto (0,0,-10)
Z10	Interpolación lineal al punto (0,0,10), donde establecemos la posición de reposo antes de continuar con el contorno interior.

Tabla 6.2: Secuencia de instrucciones G para mecanizar el contorno exterior del enunciado.

Tras esta secuencia de instrucciones, obtenemos la siguiente representación:

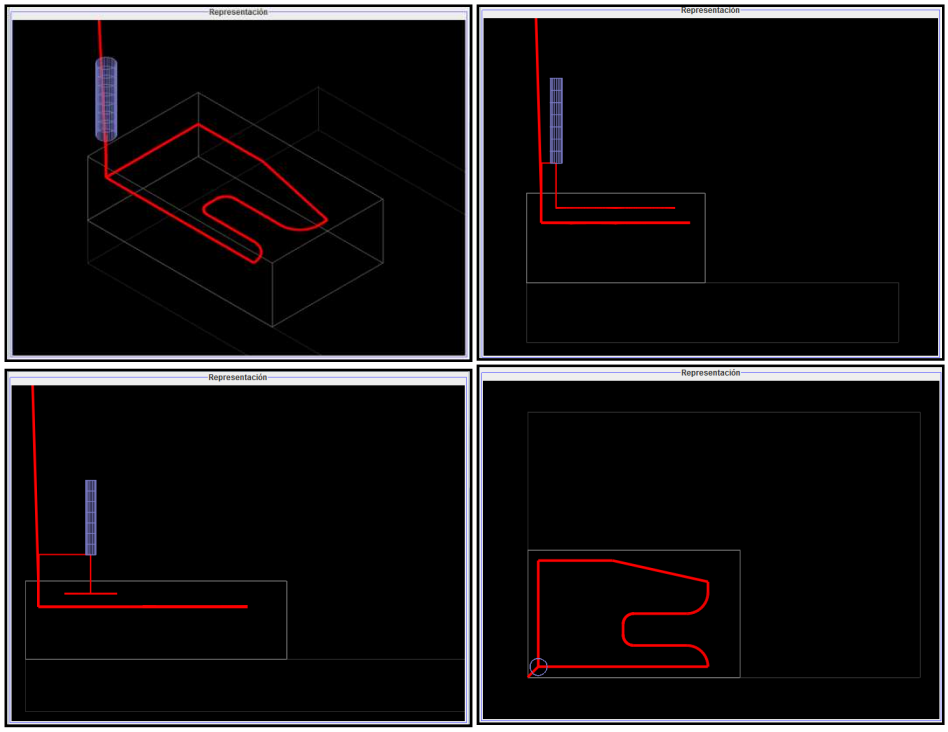


Figura 6.9: Representación multivista del mecanizado del perfil exterior.

8. A continuación, se insertarán las instrucciones para trazar el perfil interior.

Instrucción	Función
G54	Como podemos observar en la figura del enunciado, las medidas del perfil interior toman como referencia el punto Cero Pieza, por tanto, transportamos el sistema de coordenadas a dicho punto mediante el decalaje G54.
T01 D0 M06	Seleccionamos la herramienta definida para el perfil interior.
G01 X25 Y30	Desplazamos al punto (25,30,-5)
G01 Z-5	Desplazamos a Z-5 profundizando en la pieza
G02 Y30 J10	Interpolación circular a derechas indicando el mismo origen y fin (25,30,-5) y el centro (25,40,-5), generando, por tanto, una circunferencia completa.
G03 Y30 J-10	Análogo a la anterior con centro (25,20,-5), en este caso se ha usado interpolación a izquierdas para comprobar que el resultado es el mismo.
G01 Z10	Se abandona el volumen de la pieza desplazando linealmente a Z10
M05	Se desactiva el husillo
G56	Se activa el decalaje G56 = (0,0,0) que permite volver al sistema de coordenadas de la máquina.
G00 X0 Y0 Z200	Se desplaza al punto de reposo (0,0,200) en el sistema de coordenadas de la máquina.
M02	Fin del programa

Tabla 6.3: Secuencia de instrucciones G para mecanizar el contorno interior del enunciado.

Con la anterior secuencia de operaciones, se habrá obtenido el resultado final requerido por el enunciado:

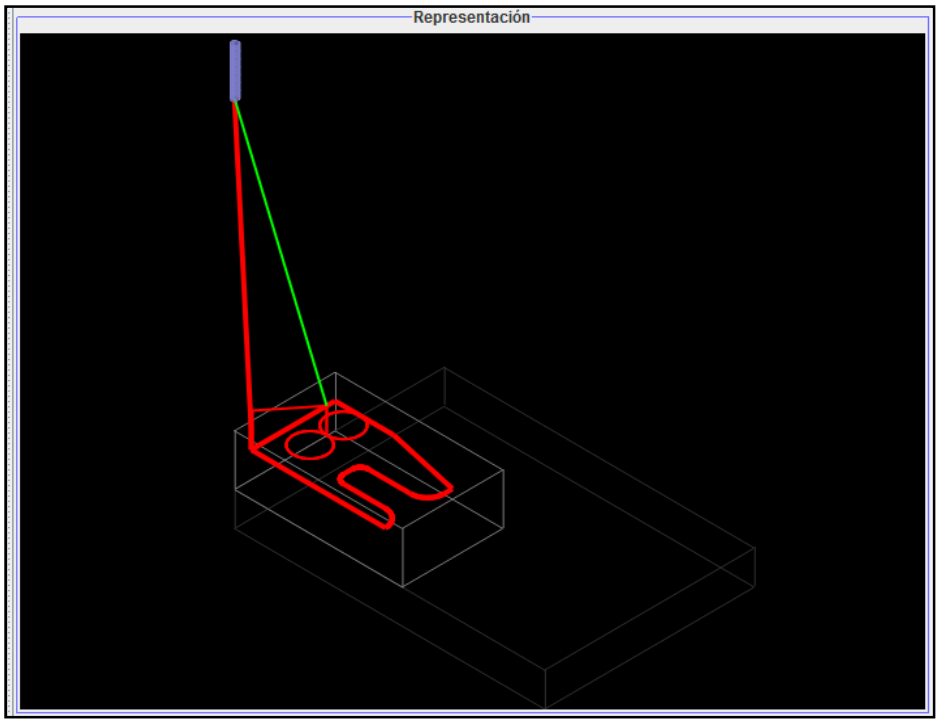


Figura 6.10: Resultado final del mecanizado en perspectiva.

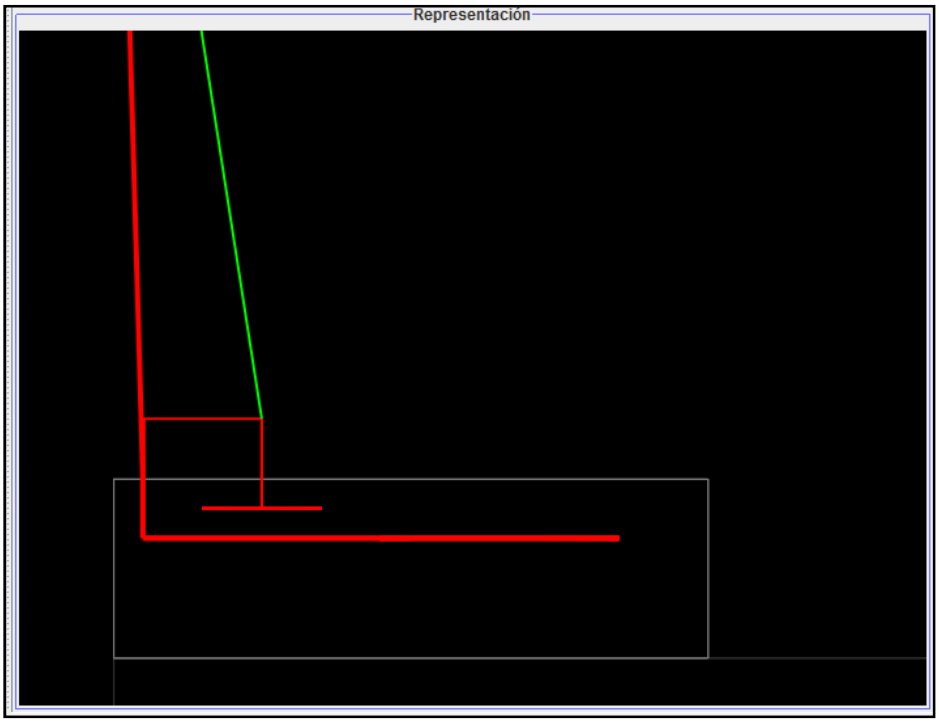


Figura 6.11: Resultado final del mecanizado en perfil.

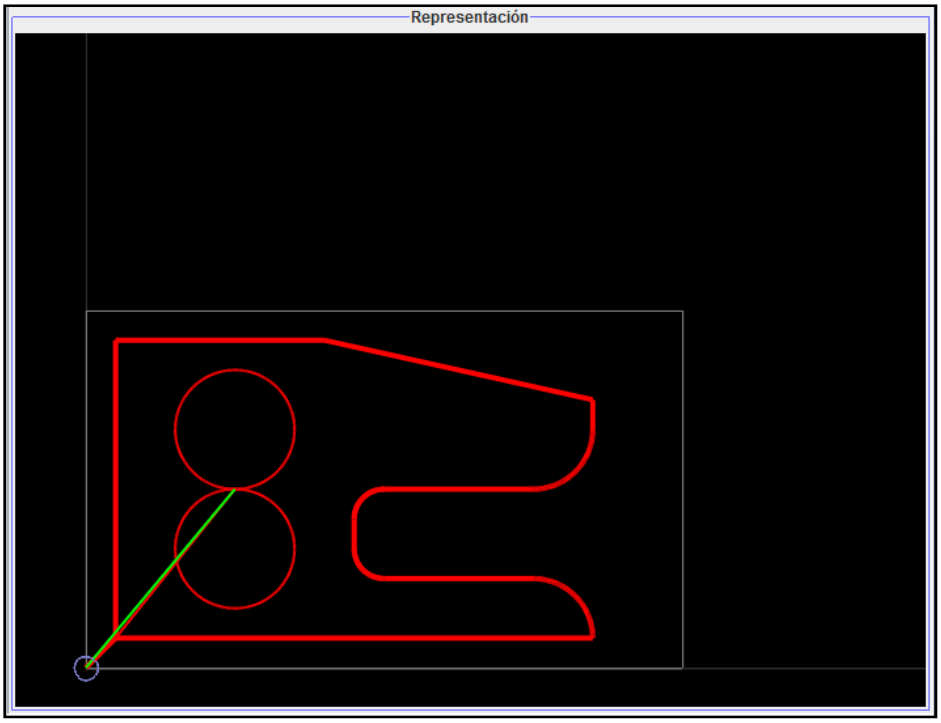


Figura 6.12: Resultado final del mecanizado en planta.

Figura 6.13: Resultado final del mecanizado en alzado

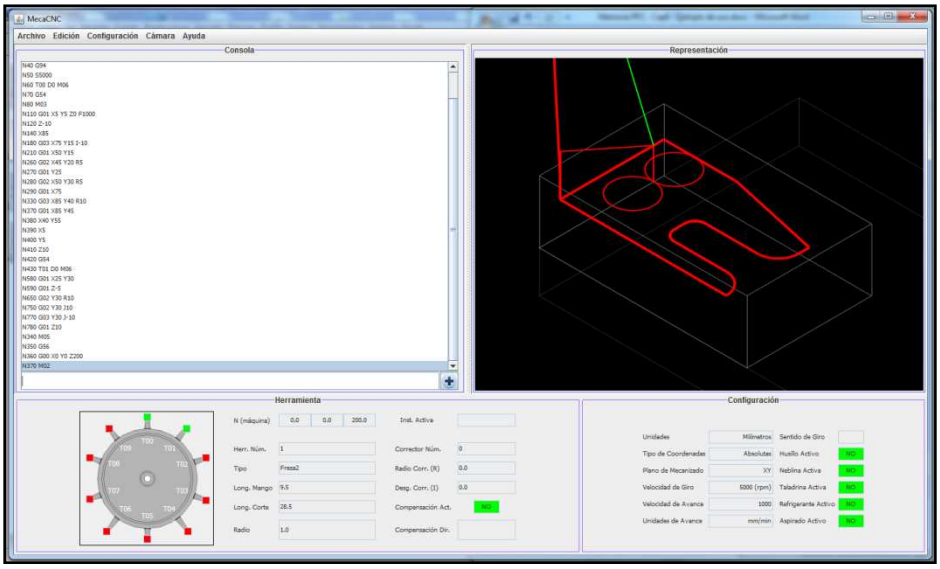


Figura 6.14: Ventana de mecaCNC tras finalizar el mecanizado.

9. Cabe destacar que, durante el proceso de inserción de código, se ha hecho uso de las utilidades de edición disponibles en el menú contextual del panel *Consola (botón derecho sobre instrucción)*, así como las funciones *Deshacer (Ctrl+Z)* y *Rehacer (Ctrl+Y),* pudiendo así flexibilizar la inserción de código en caso de insertar instrucciones incorrectas.

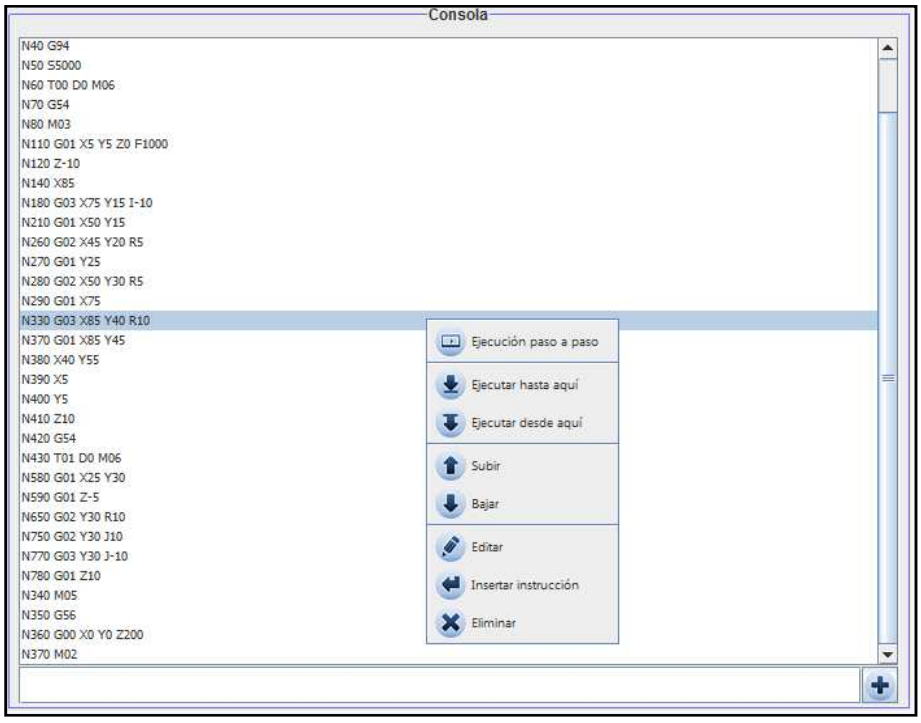


Figura 6.15: Utilidades de edición de código en panel Consola.

10. Así mismo, si el usuario desconoce o no recuerda la sintaxis de alguna instrucción del código ISO-6983, se recuerda que se puede acceder al sistema de ayuda interactiva integrada (*"Ayuda > Mostrar ayuda"* o tecla *F1*) que dispone de la lista completa de instrucciones G interpretables por el simulador y su funcionalidad asociada.

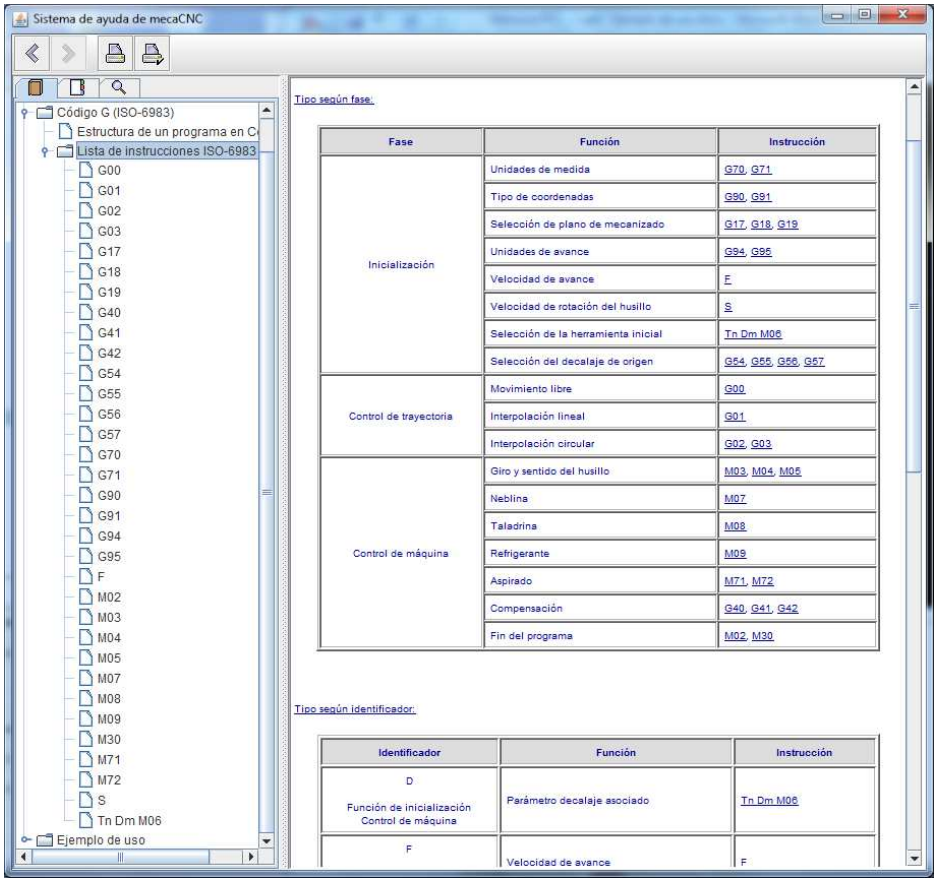


Figura 6.16: Ventana del sistema de ayuda con lista de instrucciones ISO-6983 interpretables por mecaCNC.

11. Una vez finalizado el mecanizado, se podrán realizar distintas acciones:

En primer lugar, se recomienda **guardar** *(Ctrl+S)* un archivo de proyecto ".mcnc" que contendrá las definiciones, asignaciones e instrucciones correspondientes a todas las acciones realizadas durante la ejecución.

También se podrá **exportar** *(Ctrl+E)* el código a un archivo ".txt" o ".g" que contendrá únicamente el programa ISO-6983 del código insertado. Posteriormente, este archivo podrá ser ejecutado en la máquina EMCO PC Mill 125, obteniendo la pieza análoga a la representación mostrada.

Se podrá hacer uso de la ***Ejecución paso a paso*** *(menú contextual del panel Consola)*, que permite comprobar el resultado de la ejecución hasta la instrucción

seleccionada, o bien reproducir la ejecución con un retardo específico que permita identificar la función de cada instrucción.

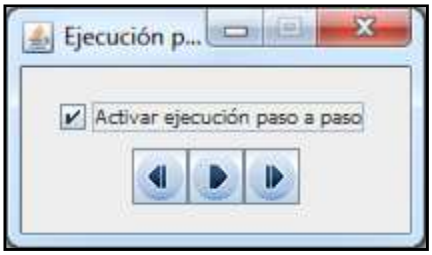


Figura 6.17: Ventana de control de ejecución paso a paso.

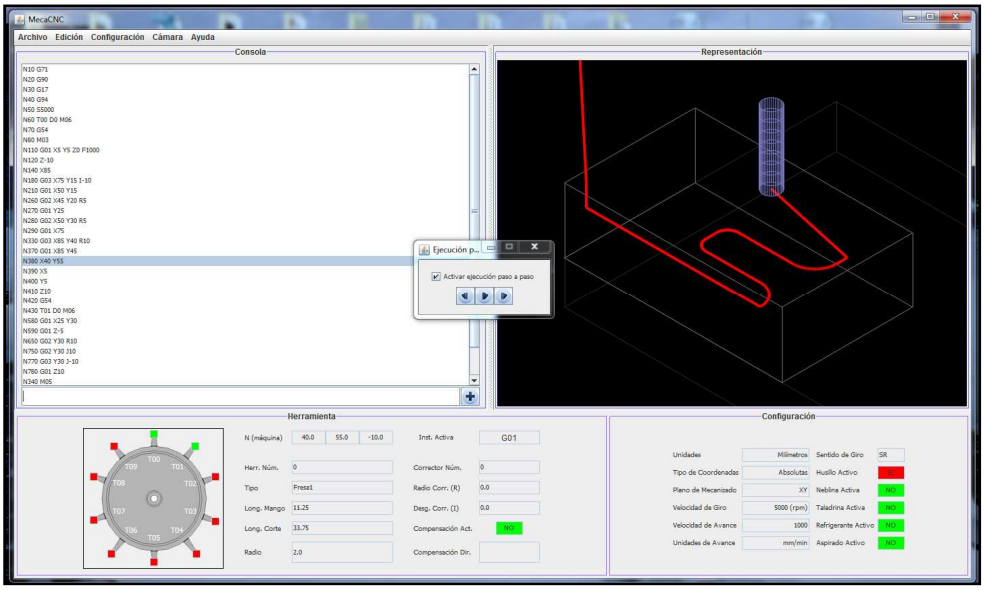


Figura 6.18: Ejecución paso a paso sobre resultado.

12. Para finalizar, cabe recordar que haciendo uso de los atajos de ratón, el usuario podrá modificar a su antojo la cámara en el panel representación, pudiendo comprobar la pieza desde distintos ángulos.

Finalizado el proceso, podremos abrir un nuevo proyecto *"Archivo > Nuevo" (Ctrl+N)* o salir del programa "*Archivo > Salir" (Ctrl+Q)*.

CAPÍTULO 7

CONCLUSIONES

En el desarrollo de un proyecto como el que nos ocupa, una parte fundamental del mismo radica en las conclusiones que se hayan podido extraer.

Se puede considerar en parte éste capítulo como el inverso a la introducción, por tanto, será necesario analizar los resultados obtenidos, comprobando el correcto cumplimiento de todos los objetivos finales e intermedios e indicando aquellos factores no considerados inicialmente y que, a posteriori, se ha creído conveniente añadir al proyecto para enriquecer su funcionalidad, contenidos, optimalidad o usabilidad.

Por otra parte, se considera que una de las mayores virtudes de las ingenierías proviene de encontrarse con errores, dificultades y problemas durante el desarrollo, ya que gracias a este conjunto de situaciones complejas que no se hayan tenido en cuenta y hayan llevado al ingeniero a situaciones que requieran un sobresfuerzo, se enriquecerá su experiencia, conocimientos y capacidad resolutiva. Es por ello que se ha estimado conveniente mostrar al lector los principales problemas encontrados durante el desarrollo y cómo éstos han sido resueltos.

Por último, la elaboración de un proyecto de tal relevancia en la vida de un ingeniero como lo es su proyecto de final de carrera, máxime si se trata de la creación de un programa complejo, genera en la persona una serie de inquietudes, ideas, posibles aplicaciones y proyecciones de futuro en la línea de aquello que se ha desarrollado. En el proyecto que nos ocupa, se ha creído conveniente analizar la proyección de futuro, estableciendo una relación de ideas generadas a raíz del desarrollo.

7.1. Análisis de resultados.

En primer lugar, será conveniente recordar al lector los objetivos especificados en la introducción de este documento. Estos son:

3. **Desarrollo de un programa docente, consistente en un simulador de mecanizados mediante lenguaje ISO-6983**, donde el alumno pueda aprender a programar mecanizados en máquinas CNC, de forma interactiva y en tiempo real, mostrando el resultado del mecanizado en 3D y con capacidad de exportación a la máquina real, adaptando la funcionalidad del simulador a la máquina EMCO PC Mill 125. Para ello se requerirá como requisitos u objetivos intermedios:

a. Que el programa sea **multiplataforma**.

b. La inserción y ejecución de instrucciones será en **tiempo real**.

c. La ejecución en tiempo real hará necesaria la elaboración de un **analizador léxico-sintáctico-semántico** que controle la sintaxis y convierta cada instrucción ISO-6983 al conjunto de instrucciones que implementen su funcionalidad.

d. El simulador será **multiventana**, permitiendo al usuario abrir múltiples instancias del mismo.

e. **El simulador emulará las funciones de la máquina EMCO PC Mill 125**.

f. El alumno ha de recibir una herramienta de auto-aprendizaje, con los contenidos necesarios accesibles desde el propio programa que le permitan aprender desde cero a programar mecanizados en código ISO-6983. Es por ello que se creará un **sistema de ayuda integrada** interactivo.

g. El programa será sostenido por una **interfaz gráfica en entorno de ventanas fácil e intuitiva**, adaptable a múltiples resoluciones de pantalla.

Deteniéndonos en este primer objetivo, podemos concluir que se ha cumplido los requisitos:

- ✓ **Multiplataforma:** el objetivo se cumple, pudiendo ejecutarse la solución en sistemas MacOSX, Windows, Linux y Solaris en distintas arquitecturas, gracias al lenguaje Java y su máquina virtual.

- ✓ **Ejecución en tiempo real:** el simulador representa la ejecución de la instrucción en tiempo real, mediante la activación de los indicadores oportunos en la interfaz, o mediante la representación tridimensional.

- ✓ **Analizador léxico-sintáctico-semántico:** para cada instrucción insertada, se analizan los posibles errores en la construcción, avisando del origen del error en su caso o tomando los datos necesarios y ejecutando la secuencia de instrucciones que darán lugar a la funcionalidad asociada.

- ✓ **Multiventana:** pueden abrirse varias instancias del programa gracias a la jerarquía de clases del lenguaje de programación orientado a objetos.

- ✓ **Adaptación a EMCO PC Mill 125:** el desarrollo del simulador ha sido adaptado a la geometría de la máquina EMCO PC Mill 125, disponiendo del mismo número de herramientas en el carrusel porta-brocas, mismos intervalos en velocidades, mismos parámetros de configuración y mismo conjunto de instrucciones ISO

admitidas. Además, el programa permite exportar el código del programa ISO y se ejecutado en la máquina real.

- ✓ **Sistema de ayuda integrada:** el simulador implementa un subsistema de ayuda integrada por contenidos, índice o búsqueda de palabras clave, incorporando la descripción de cada uno de los elementos y funcionalidades del programa, la estructura de un programa de mecanizado y el listado completo de las instrucciones ISO-6983 interpretables por el simulador junto a su funcionalidad asociada, además de un ejemplo de uso paso a paso. El sistema de ayuda es interactivo y permite la impresión de los documentos que se considere necesarios.

- ✓ **Interfaz gráfica fácil e intuitiva:** se genera una interfaz en base al estudio del usuario, contenido y contexto, tratando de facilitar los accesos en el menor número de acciones posibles, mostrando nombres, imágenes, esquemas e iconos identificativos de cada acción asociada. La interfaz se adapta a la resolución de pantalla partiendo desde un nivel mínimo necesario.

Por tanto, del cumplimiento de los requisitos y objetivos intermedios, alcanzamos la correcta realización del objetivo principal, obteniendo el simulador de mecanizados orientado a la docencia, multiplataforma, en tiempo real y adaptado a la máquina EMCO PC Mill 125.

Además, sobre estos objetivos, se han añadido distintas funcionalidades adicionales que, en tiempo de desarrollo, se han considerado convenientes para potenciar ciertos aspectos del objetivo.

- ➢ Se crea una función de importación de programas ISO-6983 en formato *'.txt'* o *'.g'*, pudiendo obtener dichos programas de las propias máquinas CNC o de la generación de Código G de soluciones CAD/CAM.

- ➢ Se añade un apartado de documentación en la ayuda, que enlazará a distintos documentos de interés en PDF, incluyendo el manual de la máquina EMCO PC Mill 125.

- ➢ Se añade un controlador paso a paso de la ejecución, permitiendo controlar el flujo de ejecución de las instrucciones que han sido insertadas en la lista, obteniendo una representación de la instrucción o intervalo de instrucciones deseadas.

- ➢ Se añaden múltiples hilos de ejecución para asumir operaciones de gran volumen de datos que puedan suponer cuellos de botella, como la importación y apertura.

- ➢ Se implementan funciones de pila de instrucciones insertadas, tales como insertar en un determinado punto, editar, borrar, deshacer y rehacer.

- Se crea una opción de guardado de archivos de configuración, que permitirá preestablecer configuraciones específicas, útiles por ejemplo para la realización de ejercicios o el trabajo recurrente sobre el mismo conjunto de herramientas.

4. Implantación del simulador como herramienta docente en los laboratorios de mecanizado del DISA y el resto de departamentos que se considere.

Tras la culminación del proyecto, se acuerda implantar el programa como herramienta de aprendizaje para la asignatura *Fabricación Asistida por Computador (FAC)*, impartida por Eduardo Vendrell en los laboratorios de mecanizados del Departamento de Ingeniería de Sistemas y Automática (DISA) de la Universidad Politécnica de Valencia.

Debido a los tiempos de finalización del proyecto en relación a los meses en los que se imparte la asignatura, este objetivo no será llevado a cabo hasta el inicio de las clases de la asignatura.

El simulador ha sido sometido a constantes pruebas iterativas (incluyendo importación de archivos en Código G de aproximadamente 14000 líneas de código), y ha sido usado para resolver una batería de ejercicios propuestos en la asignatura, pero se considera una oportunidad única y excepcional su uso por el conjunto de alumnos de FAC.

Pese a que el simulador en el momento de su implantación ya habrá sido finalizado y presentado, se acuerda establecer un seguimiento de su implantación como herramienta docente.

7.2. Dificultades encontradas durante la elaboración del proyecto.

Durante el periodo de desarrollo del proyecto, cabe subrayar ciertos contratiempos que, por su complejidad o relevancia de cara al cumplimiento de plazos, sean susceptibles de mención.

En primer lugar, el proyecto se inició haciendo uso del entorno de programación Eclipse. Como se ha comentado a lo largo de la memoria, el lenguaje Java no implementa funciones nativas de manejo de OpenGL, siendo necesaria la instalación de dichas librerías como un plugin o complemento al entorno. El requerimiento multiplataforma en conjunción con un cambio de versión de Eclipse, derivó en la necesidad de cambio de entorno, ya que no se pudo obtener las bibliotecas Java OpenGL adecuadas para que el programa funcionase en sistemas MacOSX.

Este hecho llevó a la consideración de otros entornos de programación, estudiándose las características que ofrecía NetBeans y comprobándose que el propio entorno es capaz de generar el programa para múltiples plataformas, adaptando las librerías que importa el

programa a cada uno de esos sistemas, y disponiendo de un plugin de fácil instalación que añadía al programa las librerías y componentes necesarios para enlazar OpenGL con la interfaz.

Por ello, se decidió cambiar de entorno a NetBeans y la importación del proyecto Eclipse a NetBeans no ofrecía los resultados deseados en relación a la interfaz gráfica, por tanto, se decidió reprogramar la interfaz gráfica desde cero.

Por otra parte, algunos componentes *SWING* ofrecidos por Java presentan ciertas deficiencias de cara a la ejecución en tiempo real. En concreto, el componente *GLJPanel* de Java que define el marco de representación gráfica de MecaCNC presenta unos tiempos de respuesta inferiores al *GLCanvas*, también para el manejo de la representación de gráficos OpenGL. Sin embargo, este último presentaba ciertos problemas en determinadas plataformas y se decidió hacer uso del componente *GLJPanel*, lo cual creó la necesidad de una mayor optimización del código para obtener mejores tiempos de respuesta del componente.

Por último, uno de los últimos errores que limitaron el diseño conceptual inicial viene determinado por un *bug* en el componente *SWING JList*. Dicho componente presenta un comportamiento inestable si se realizan, de una a una y en tiempo de ejecución, sucesivas inserciones de cadenas de caracteres en la lista (por ejemplo, desde un bucle *for*) para un conjunto relativamente grande de inserciones, y se pretende que el componente muestre siempre la última cadena insertada (mediante el uso del método *JList.ensureIndexIsVisible(int)*).

Esta implementación (considerada necesaria en la importación y apertura de archivos), mostraba deficiencias de inestabilidad en el redibujado del componente, presentando en el redibujado del componente un efecto “flash”, caracteres extraños durante la inserción en bucle y movimientos alternativos del indicador de la barra de desplazamiento del componente. Para solucionar este problema (del que la comunidad Java ya tiene constancia), se consideró:

- Por una parte, para la función de importación y apertura de archivos de proyecto, es conveniente que esta se haga en el menor tiempo posible, por tanto, se insertan todas las instrucciones y posteriormente se actualiza el componente y se representa gráficamente su ejecución línea por línea.

- Por otra parte, en el caso de la ejecución paso a paso, el objetivo consiste en que el usuario compruebe qué es lo que hace cada instrucción, por tanto, no es necesaria la velocidad del caso anterior, permitiendo así realizar pausas de 0.5 segundos entre operaciones *JList.ensureIndexIsVisible(int),* reduciendo así las deficiencias del redibujado del componente.

7.3. Proyección de futuro.

La realización de este proyecto ha proporcionado una serie de inquietudes relacionadas con el mismo y con la ingeniería del software en materia de programas que sirvan para su aplicación docente.

Además, maximizar el ciclo de vida de un programa como mecaCNC supone considerar la elaboración de revisiones y cambios de versión que aporten nuevas funcionalidades, ampliación de las ya existentes o implementación de necesidades que puedan ir surgiendo durante su uso prolongado y que no hayan sido consideradas inicialmente.

De la conjunción resultante entre estas inquietudes, la motivación adquirida y el deseo de maximizar el ciclo de vida del simulador, se han tenido ciertas ideas de futuro para próximas versiones del programa:

- Se ha considerado la generalización del simulador a otras máquinas CNC mediante módulos programables por el propio usuario que determinen las características físicas y funcionales de la adaptación de mecaCNC a dichas máquinas.

- Actualmente, ya se ha analizado y se considera un siguiente paso en el desarrollo, añadir la representación en sólido del resultado del mecanizado. Entre las posibles soluciones a dicha mejora, se considera el uso del Stencil Buffer de OpenGL para implementar las operaciones booleanas que permitan la representación sólida en base a CSG.

- La tecnología Web 2.0 lleva a considerar la adaptación de la próxima versión de mecaCNC a su versión online, pudiendo así llegar a una mayor masa crítica de usuarios del simulador.

- Aunque no se haya considerado como plataforma objetivo, la compilación del lenguaje Java para su uso en plataformas Android es relativamente fácil mediante el uso de la herramienta '*dx*', incluida en el propio SDK de Android, la cual permite convertir archivos Java al formato DEX interpretable por la máquina virtual Dalvik de Android.

- Por último, cabe mencionar que, durante el análisis y posterior diseño del simulador, se ha echado en falta una librería Java que implementase algunos de los métodos matemáticos descritos en el capítulo 5, por tanto, se considera la posibilidad de traducir las funciones desarrolladas para mecaCNC al inglés y compartir la librería resultante con la comunidad.

Como se puede observar, la realización del proyecto ha resultado en un deseo de continuidad en proyectos del ámbito de la ingeniería de sistemas y automática, tanto en su aplicación docente como profesional, señal inequívoca de la satisfacción de los conocimientos adquiridos y el gusto por la materia.

CAPÍTULO 8

REFERENCIAS

[1] "Sistemas CAM"
E. Vendrell Vidal.
Fabricación Asistida por Computador.
Departamento de Ingenierías de Sistemas y Automática.
2009

[2] "Centros de mecanizado - características y conceptos"
L. Muñoz.
URL: *http://www.ib.cnea.gov.ar/~mater2/MATERIALESII/CentroMec.pdf*
Materiales II.
Instituto Balseiro.
2008

[3] "Advanced Mathematics: Precalculus with Discrete Mathematics and Data Analysis."
R.G. Brown.
ED: McDougal Littell.
ISBN 0-395-77114-5.
1997

CAPÍTULO 9

BIBLIOGRAFÍA

1. **"*Apuntes de la asignatura Fabricación Asistida por Computador*"**
   Responsable de la asignatura: E. Vendrell Vidal.
   Departamento de Ingeniería de Sistemas y Automática (DISA).
   Universidad Politécnica de Valencia, 2009.

2. ***"Apuntes de la asignatura Ingeniería de la Programación"***
   Responsable de la asignatura: J. Sánchez Díaz.
   Departamento de Sistemas Informáticos y Computación (DSIC).
   Universidad Politécnica de Valencia, 2008.

3. ***"Apuntes de la asignatura Metodología y Tecnología de la Programación"***
   Responsable de la asignatura: F. J. Jaén Martínez.
   Departamento de Sistemas Informáticos y Computación (DSIC).
   Universidad Politécnica de Valencia, 2003.

4. ***"Apuntes de la asignatura Interfaz Gráfica de Usuario"***
   Responsable de la asignatura: M.C. Juan Lizandra.
   Departamento de Sistemas Informáticos y Computación (DSIC).
   Universidad Politécnica de Valencia, 2004.

5. ***"Apuntes de la asignatura Gráficos Por Computador"***
   Responsable de la asignatura: R.A. Vivó Hernando.
   Departamento de Sistemas Informáticos y Computación (DSIC).
   Universidad Politécnica de Valencia, 2009.

6. ***"Apuntes de la asignatura Gráficos Por Computador"***
   Responsable de la asignatura: R.A. Vivó Hernando.
   Departamento de Sistemas Informáticos y Computación (DSIC).
   Universidad Politécnica de Valencia, 2009.

7. ***"Thinking in Java (4th edition)"***
   Bruce Eckel.
   Editorial Prentice Hall.
   2007.

8. ***"Materiales II: Centros de Mecanizado – Características y Conceptos"***
   L. Muñoz.
   Instituto Balseiro, 2008.

9. ***Tesis doctoral "Estudio numérico de los fenómenos de contacto en el mecanizado"***
R. Cheriguene.
Departamento de Ingeniería Mecánica y Organización Industrial.
Universidad Carlos III de Madrid, 2009.

10. ***Tesis doctoral "Modelo Superficie-Trayectoria. Un modelo geométrico para el diseño y fabricación de objetos tridimensionales"***
R. Molina Carmona
Departamento de Ciencia de la Computación e Inteligencia Artificial.
Universidad de Alicante, 2002.

11. ***"JavaHelp™ 2.0 System User's Guide"***
Sun Microsystems, Inc.
2004

12. ***"Consejos para el desarrollo de tu proyecto final de carrera"***
F.J. Abad Cerdá
Departamento de Sistemas Informáticos y Computación (DSIC).
Universidad Politécnica de Valencia, 2010.

13. ***"Evolución técnica de la máquina-herramienta. Reseña histórica"***
P. Aldabaldetrecu
Interempresas.net, 2002
URL: http://www.interempresas.net/MetalMecanica/Articulos/1435-Evolucion-tecnica-de-la-maquina-herramienta-Resena-historica.html

14. **Java SE 6 Documentation.**
URL: http://download.oracle.com/javase/6/docs/index.html#NewFeature

15. **OpenGL Software Development Kit Documentation**
URL: http://www.opengl.org/sdk/

16. **NeHe OpenGL Tutorials**
URL: http://nehe.gamedev.net/

17. **Wikipedia. La enciclopedia libre.**
URL: http://www.wikipedia.org/

ANEXO A

MANUAL DE USUARIO

Simulador mecaCNC v1.0.

El presente documento consiste en el manual de usuario del simulador de instrucciones ISO-6983 mecaCNC en su primera versión. En este manual se tratarán todos los aspectos funcionales de la aplicación, permitiendo al usuario un máximo aprovechamiento de sus funcionalidades.

Introducción.

mecaCNC consiste en un simulador de mecanizados que interpreta y representa en tiempo real, las instrucciones en código ISO-6983 insertadas. mecaCNC está especialmente concebido para su uso en la docencia, ofreciendo características que faciliten el auto-aprendizaje tanto del uso de la aplicación como de las bases y procedimientos de la programación CNC en los procesos de mecanizado.

A fin de obtener una base de conocimiento que aproxime al proceso de mecanizado real, el simulador mecaCNC está adaptado al centro de mecanizado EMCO PC Mill 125, tratando de emular su funcionalidad e interpretando el mismo conjunto de instrucciones que dicho centro de mecanizado admite como propias.

El uso de mecaCNC está fuertemente ligado a la docencia, permitiendo al usuario afianzar conceptos y comprobar, mediante una representación tridimensional adjunta a indicadores del estado de la máquina, que supone en el proceso de mecanizado real la ejecución de cada una de las instrucciones ISO insertadas.

Requisitos del programa.

Requisitos software.

- mecaCNC es multiplataforma, pudiendo ser ejecutado en los sistemas operativos:
    - Windows XP, Windows Vista, Windows 7 (arquitecturas x86 y x64).
    - MacOS X [10.0 - 10.7] (arquitecturas PowerPC e Intel x86 y x64).
    - Linux (arquitecturas x86 y x64).
    - Solaris (arquitecturas SPARC, SPARCv9 e i586, x86 y x64).
- Para la ejecución de mecaCNC será necesaria la instalación de Java Runtime Environment (JRE), preferiblemente en su última versión (Java 6).
    - Puede descargar Java Runtime Environment (JRE) gratuitamente desde la web http://www.java.com.
- Para la lectura de la documentación asociada, será necesario Adobe Reader, preferiblemente en su última versión (Adobe Reader X).
    - Puede descargar Adobe Reader gratuitamente desde la web http://www.adobe.com/es/products/reader.html.

Requisitos hardware.

- 128 MB RAM (recomendado 512MB).
- 100 MB de espacio libre en disco (recomendado 300MB).
- Tarjeta de aceleración gráfica de al menos 32MB de VRAM (recomendado 64MB).
- Resolución mínima del dispositivo de salida de imagen de 1024 x 768 píxeles.

Instalación y apertura del programa.

mecaCNC se ejecuta sobre Máquina Virtual de Java (JVM), pudiendo, por tanto, ser ejecutado sin instalación previa.

Para obtener una copia de mecaCNC en su disco local, únicamente será necesario descomprimir en la ubicación deseada el archivo Zip correspondiente a su sistema operativo y distribución.

Para ejecutar mecaCNC:

Windows:

- Diríjase a la carpeta <Unidad>:\<Carpeta de descompresión>\mecaCNC. (Por ejemplo, "C:\Archivos de programa\mecaCNC").

- Haga doble clic sobre el archivo "mecaCNC.jar".
    * Puede ejecutar mecaCNC desde consola de comandos mediante la instrucción "java –jar <Unidad>:\<Carpeta de descompresión>\mecaCNC\mecaCNC.jar".

MacOS X - Solaris:

- Diríjase a la carpeta /<Carpeta de descompresión>/mecaCNC. (Por ejemplo, "/Users/usuario1/Simulador/mecaCNC").

- Haga doble clic sobre el archivo "mecaCNC.jar".
    * Puede ejecutar mecaCNC desde el terminal mediante la instrucción "java –jar /<Carpeta de descompresión>/mecaCNC/mecaCNC.jar".

Linux:

- Abrir un terminal.

- Para la correcta ejecución de mecaCNC, será necesario indicar la ubicación de las librerías de JRE. Esta operación puede realizarse, considerando la ubicación por defecto de la instalación de Java 6, mediante la instrucción:
    - export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/jvm/java-6-sun

- Diríjase a la carpeta /<Carpeta de descompresión>/mecaCNC. (Por ejemplo, "/Users/usuario1/Simulador/mecaCNC").

- Ejecutar mecaCNC desde el terminal mediante la instrucción "java –jar ./mecaCNC.jar".

Guía rápida de uso.

1. Abrir mecaCNC según el procedimiento explicado en el apartado anterior.

2. Definir el volumen inicial de la pieza desde *Barra de menú > Configuración > Definir Pieza*.

   2.1. Establecer los valores propios de *Dimensiones* y la ubicación de la *Mordaza*.

   2.2. Presionar el botón "*Definir*".

3. Asignar las herramientas deseadas al carrusel porta-herramientas desde *Barra de menú > Configuración > Configurar herramientas*.

   3.1. Insertar nuevas herramientas definiendo sus parámetros de *Nombre, Longitud de Mango, Longitud de Corte* y *Radio*. Presionar el botón "*Insertar"*.

   3.2. Seleccionar una herramienta en la tabla del panel *Tabla de Herramientas* y seleccionar una ranura libre en la tabla *Asignar Herramienta.* Presionar el botón "*Asignar"*.

4. En caso de uso, definir los valores de los decalajes desde *Barra de menú > Configuración > Definir decalajes.* De forma automática se habrá establecido el decalaje G54 en el punto Cero Pieza al definir el volumen de la pieza.

5. En caso de uso, definir los valores de los correctores desde *Barra de menú > Configuración > Definir correctores.*

6. Insertar, desde cuadro de texto del panel *Consola,* las instrucciones ISO de configuración inicial de la máquina. Una vez insertada la instrucción, presionar la tecla Retorno o el botón "⊕".

7. Insertar las instrucciones ISO de interpolación y configuración que compondrán el mecanizado.

8. Observar el resultado de la inserción de instrucciones en el panel *Representación, Herramienta* y *Configuración.* Puede hacer uso del ratón para cambiar la vista tridimensional en el panel *Representación*.

9. Una vez obtenido el resultado del mecanizado deseado, podrá:

   9.1. Guardar el proyecto desde *Barra de menú > Archivo > Guardar.*

   9.2. Exportar el código ISO a un archivo ".txt" o ".g" desde *Barra de menú > Archivo > Exportar.*

Componentes y funcionalidad.

En el presente apartado de este manual, se describirán exhaustivamente cada uno de los componentes y elementos junto a la funcionalidad asociada que componen el simulador mecaCNC.

1. Ventana principal.

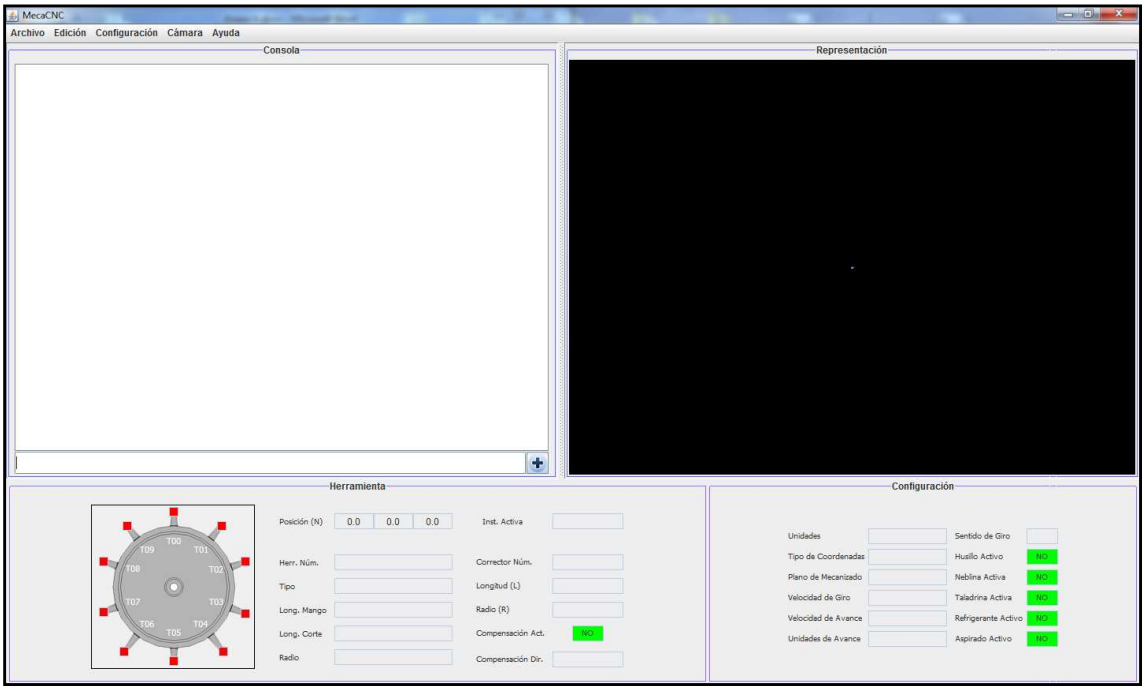


Figura Anexo A.1: Ventana principal.

La ventana principal supone el componente que se presentará ante el usuario al abrir la aplicación. Desde esta ventana se accede al resto de componentes y funcionalidad del programa.

La apertura del programa implica la adaptación de la ventana principal a la resolución del equipo donde se ejecute, siendo recomendada una resolución mínima de 1024 x 768 píxeles.

Para una mejor descripción, dividiremos la ventana en 5 elementos diferenciados que se analizarán por separado:

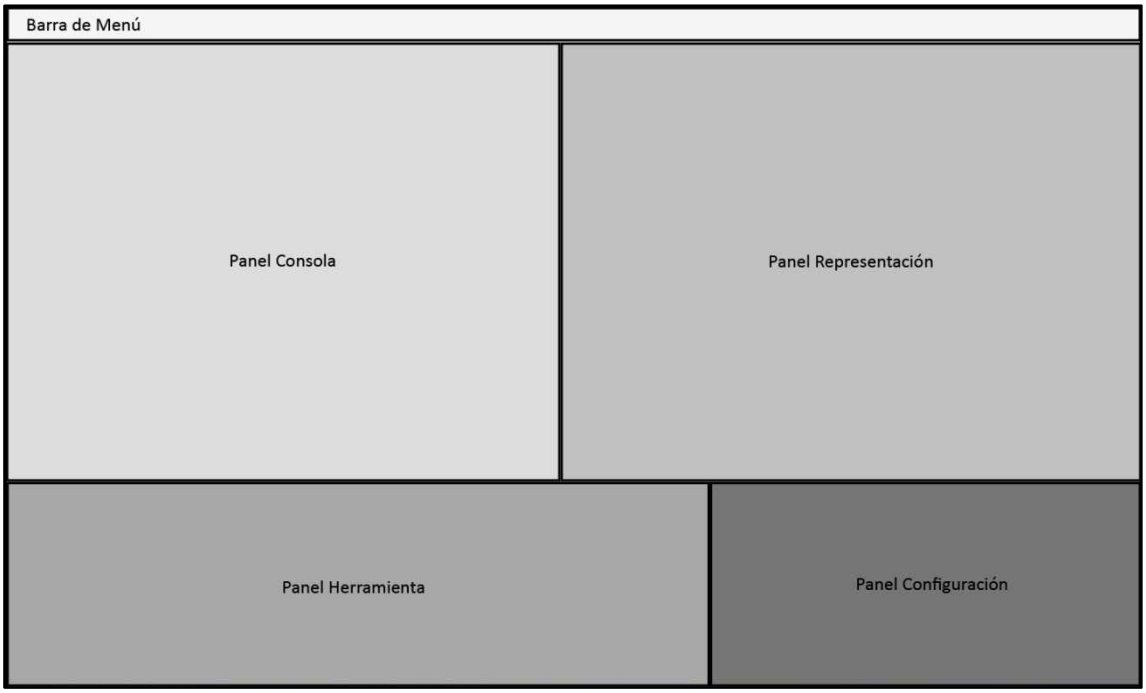


Figura Anexo A.2: Elementos de la ventana principal.

1.1. Barra de menú.

La barra de menú brinda acceso a las múltiples funciones del programa.

Figura Anexo A.3: Barra de menú.

1.1.1. Archivo.

Elemento de la barra de menú donde se ubican las operaciones de creación, guardado y apertura de los distintos tipos de archivos asociados a mecaCNC, así como la función de cierre del programa.

Figura Anexo A.4: Menú Archivo

1.1.1.1. Nuevo.

Abre una nueva ventana principal de mecaCNC.

Dada la cualidad multiventana de mecaCNC, la creación de una nueva ventana de proyecto no implica el cierre de la ventana padre, pudiendo ejecutarse en paralelo múltiples instancias del programa.

Acceso por atajo de teclado: Ctrl + N

1.1.1.2. Abrir.

Permite la apertura de archivos de proyecto mecaCNC (.mcnc), previamente guardados.

Al presionar *Abrir*, se mostrará la ventana con el árbol de directorios donde el usuario deberá seleccionar la ruta y el archivo de proyecto.

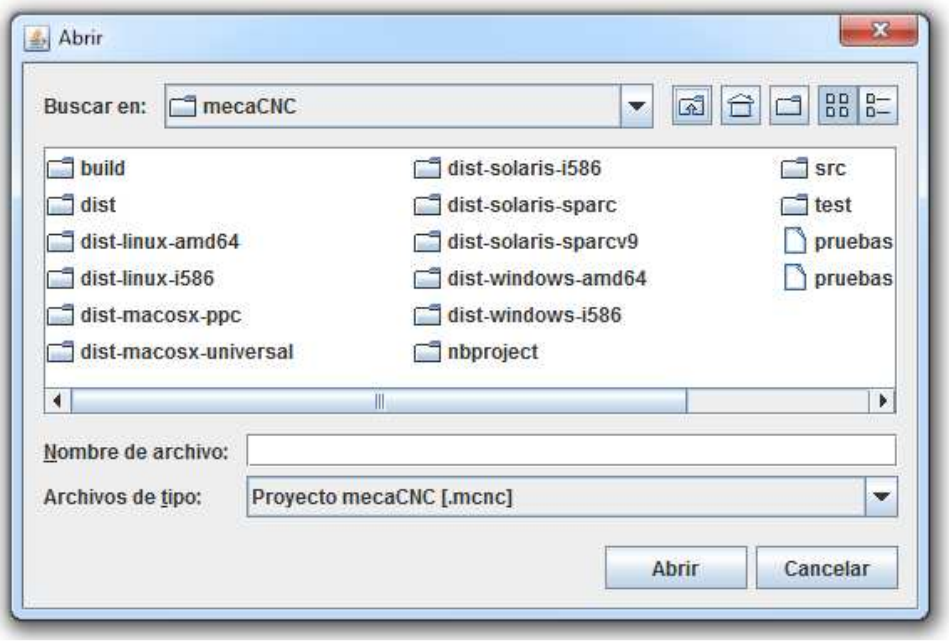


Figura Anexo A.5: Ventana Abrir.

Los archivos de proyecto mecaCNC especifican:

- Dimensiones de la pieza y ubicación de la mordaza.
- Definición de decalajes.
- Definición de nuevas herramientas.

- Asociaciones de herramientas al porta-herramientas.
- Definición de correctores.
- Código G insertado.

Nótese que al abrir un nuevo archivo de proyecto, se perderán todos los cambios no guardados del proyecto en ejecución.

Acceso por atajo de teclado: Ctrl + O

1.1.1.3. Guardar.

Guarda un archivo de proyecto mecaCNC (.mcnc).

Al presionar *Guardar*, se mostrará la ventana con el árbol de directorios donde el usuario deberá seleccionar la ruta y el nombre del archivo.

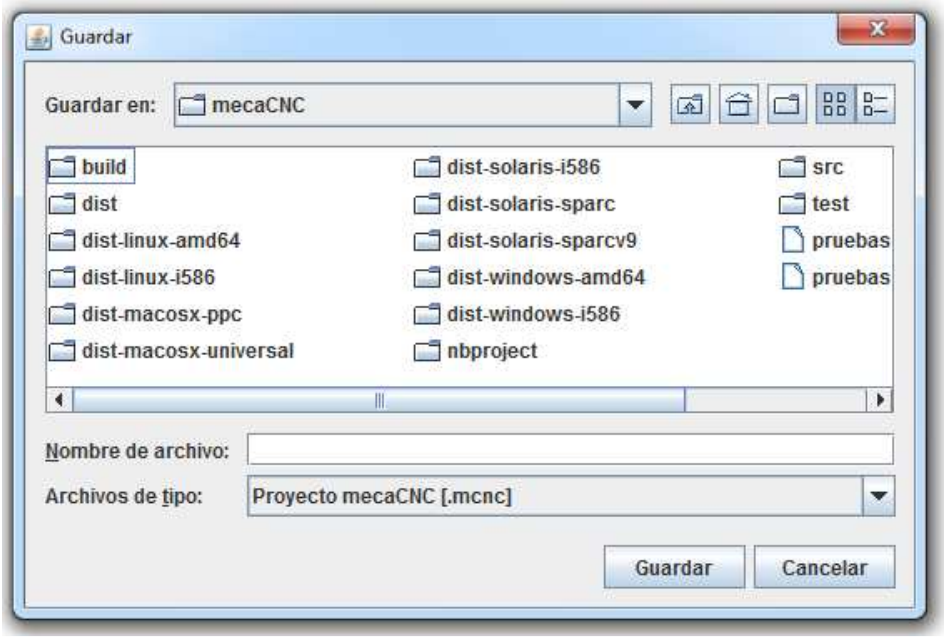


Figura Anexo A.6: Ventana Guardar.

Al presionar en "*Guardar*" en la ventana emergente, el programa generará un archivo de proyecto con extensión "*.mcnc*" cuyo contenido describe:

- Dimensiones de la pieza y ubicación de la mordaza.
- Definición de decalajes.
- Definición de nuevas herramientas.
- Asociaciones de herramientas al porta-herramientas.

- Definición de correctores.
- Código G insertado.

Nótese que si se guarda el proyecto sin alguno de estos datos definidos, estos se evaluarán a <nulo>, su valor por defecto.

Acceso por atajo de teclado: Ctrl + S

1.1.1.4. Exportar.

Exporta el conjunto de instrucciones en Código G introducidas en mecaCNC a un archivo con extensión ".g" o ".txt".

Al presionar *Exportar*, se mostrará la ventana con el árbol de directorios donde el usuario deberá seleccionar la ruta y el nombre del archivo.

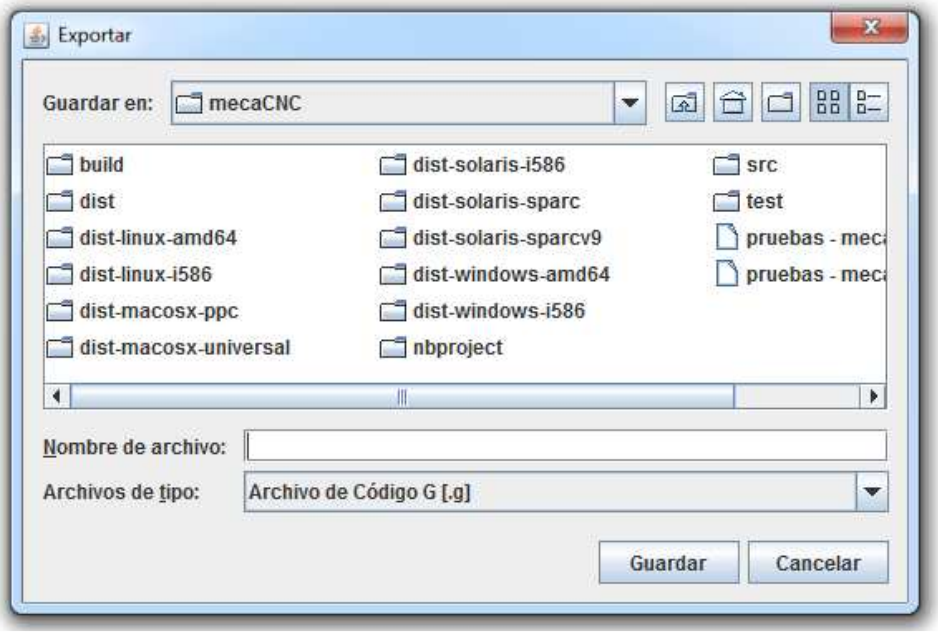


Figura Anexo A.7: Ventana Exportar.

Se podrá seleccionar el tipo de archivo a exportar seleccionándolo desde el desplegable *"Archivos de tipo:"*, pudiendo seleccionar entre *"Archivo de Código G [.g]"* o *"Archivo de Texto [.txt]".*

La exportación únicamente contempla el Código G introducido, ignorando las dimensiones de la pieza, decalajes, correctores, asignaciones, etc.

Este archivo de programa ISO exportado podrá ser cargado en el centro de mecanizado real EMCO PC Mill 125 y en máquinas-herramienta y otros programas simuladores y CAD/CAM compatibles con la sintaxis ISO-6983.

Acceso por atajo de teclado: Ctrl + E

1.1.1.5. Importar

Función de importación de archivos de programas ISO-6983.

Al presionar *Importar*, se mostrará la ventana con el árbol de directorios donde el usuario deberá seleccionar la ruta y el archivo de código G o archivo de texto.

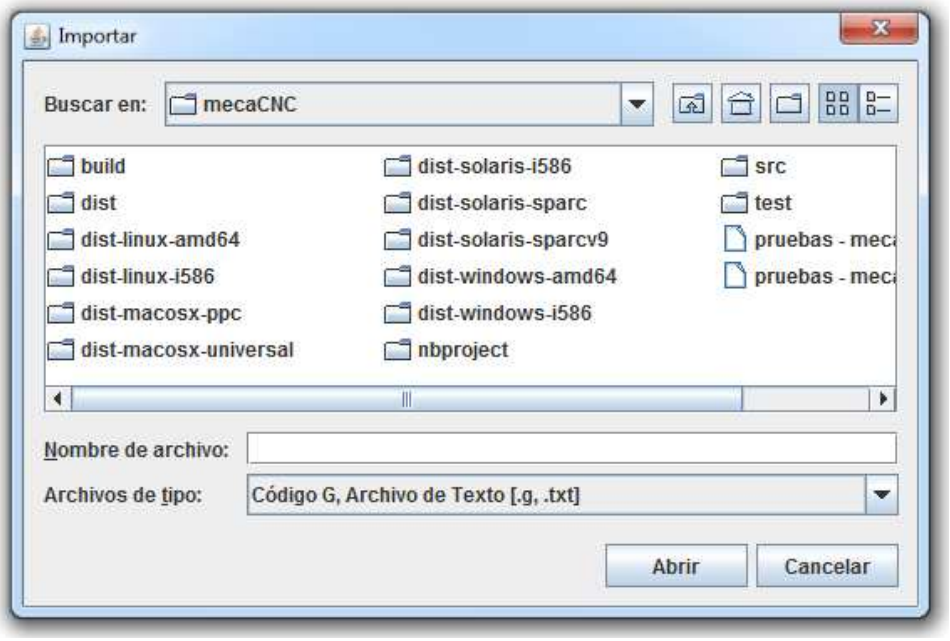


Figura Anexo A.8: Ventana Importar.

mecaCNC leerá las instrucciones contenidas en archivos de instrucciones en Código G (".g" y ".txt"), insertándolas y ejecutándolas en el proyecto activo.

Gracias a la función de importado, el programa podrá leer los archivos exportados de otros programas o incluso instrucciones insertadas a mano en un archivo de texto, siempre que su sintaxis sea la correcta y las instrucciones pertenezcan al conjunto de instrucciones de Código G interpretables por mecaCNC.

mecaCNC ignorará aquellas líneas que comiencen por el carácter "(", pudiendo ser usado a modo de comentario en el código fuente.

Acceso por atajo de teclado: Ctrl + I

1.1.1.6. Salir.

Cierra el programa.

Nótese que haciendo uso de la opción "*Salir*" se cerrarán todas las ventanas abiertas del programa, incluso las instancias múltiples de la ventana principal generadas a partir de la función "*Nuevo*".

Acceso por atajo de teclado: Ctrl + Q

1.1.2. Edición.

Elemento de la barra de menú que brinda acceso a las operaciones de deshacer y rehacer en relación a la inserción de instrucciones ISO.

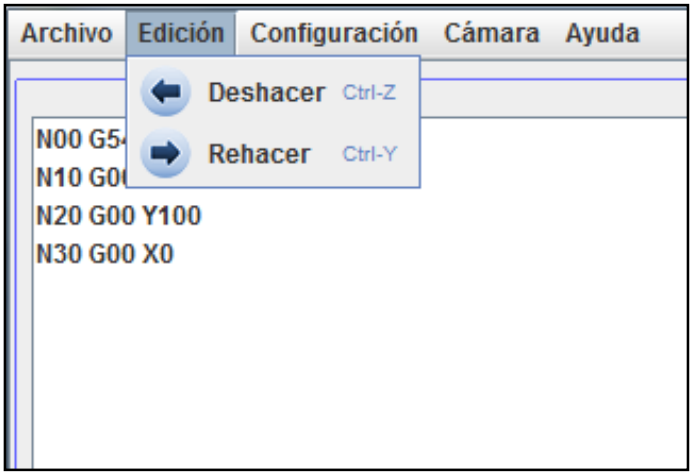


Figura Anexo A.9: Menú Edición.

Las funciones "*Deshacer*" y "*Rehacer*" figurarán inicialmente deshabilitadas, siendo necesario insertar una instrucción para habilitar *Deshacer* y presionar al menos una vez en *Deshacer* para habilitar *Rehacer* (operación inversa).

Así mismo, *Deshacer* volverá a deshabilitarse cuando no existan instrucciones ISO en la lista de instrucciones insertadas y *Rehacer* se deshabilitará al haber retomado todos los cambios generados por *Deshacer*.

1.1.2.1. Deshacer.

Elimina la última instrucción en Código G introducida y re-ejecuta el conjunto de instrucciones resultantes.

Acceso por atajo de teclado: Ctrl + Z

1.1.2.2. Rehacer.

Restaura las instrucciones en Código G eliminadas al hacer uso de Deshacer y re-ejecuta el conjunto de instrucciones resultantes (operación inversa a deshacer).

Acceso por atajo de teclado: Ctrl + Y

1.1.3. Configuración.

Elemento de la barra de menú que proporciona acceso a distintas funciones de configuración del mecanizado y la máquina herramienta.

Figura Anexo A.10: Menú Configuración.

1.1.3.1. Definir pieza.

Muestra la ventana de definición del volumen de la pieza y ubicación de la mordaza.

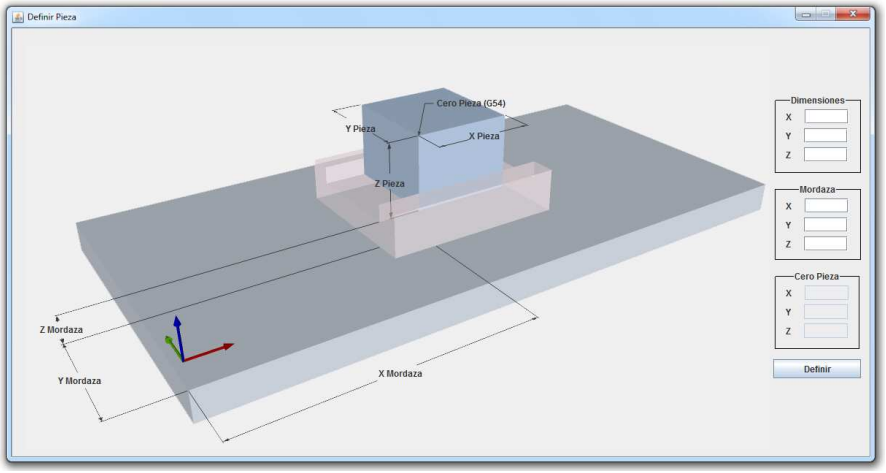


Figura Anexo A.11: Ventana Definir pieza (apariencia inicial).

La ventana mostrará, en un estado previo a la definición inicial de la pieza, una representación gráfica aproximada del banco de trabajo con los indicadores de las medidas a definir.

La definición consistirá en establecer los valores en milímetros de la longitud de la pieza en base a sus tres componentes (x, y, z), así como la ubicación de la mordaza en base a la distancia de ésta al punto Cero Máquina. Si se establece a 0 la ubicación de la mordaza, la ubicación de la pieza en el banco de trabajo vendrá definida únicamente por su geometría.

Nótese que, según el código de colores de los ejes de coordenadas representados en el punto Cero Máquina, la componente "X" corresponderá al color rojo, verde para la componente "Y" y azul para la componente "Z".

Para facilitar la inserción de datos por teclado, se ha habilitado la tecla tabulador que permite navegar entre campos.

Una vez insertados los valores de las medidas, presionaremos el botón "*Definir*" o pulsaremos la tecla *retorno*.

Como se puede comprobar, al presionar "*Definir*" de forma automática se calcula el punto Cero Pieza y es asignado al decalaje G54. Dicho decalaje podrá ser editado a posteriori desde la función *Barra de menú > Configuración > Definir decalajes.*

Al definir los valores, estos se mostrarán en la acotación de las medidas.

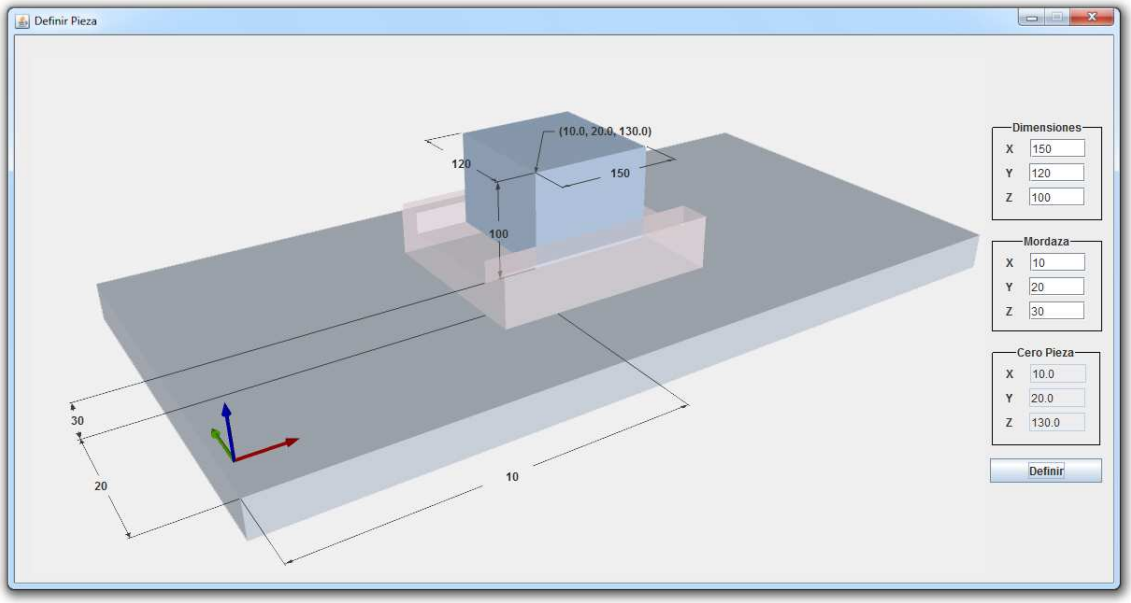


Figura Anexo A.12: Ventana Definir pieza (apariencia tras definición de medidas)

Por último, cabe recordar que la redefinición de la pieza implica la re-ejecución automática del Código G insertado hasta entonces.

Acceso por atajo de teclado: Ctrl + P

1.1.3.2. Definir decalajes.

Muestra la tabla de definición de decalajes.

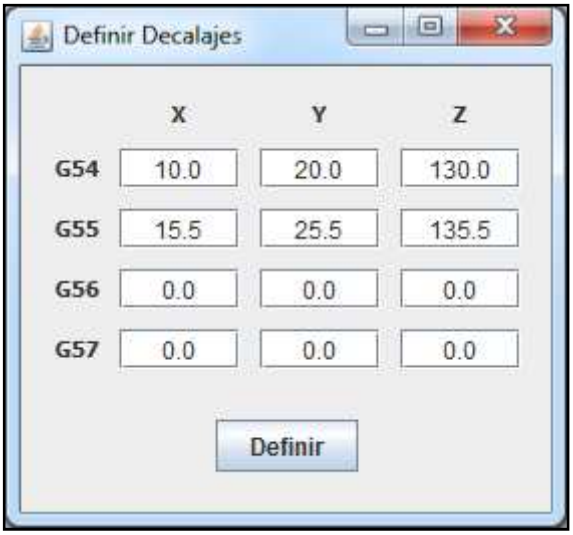


Figura Anexo A.13: Ventana Definir decalajes.

Los decalajes desplazan el sistema de coordenadas de la máquina de la posición Cero Máquina (M) a la posición determinada por el decalaje.

Existen 4 decalajes, (G54, G55, G56, G57), y de forma automática, una vez definida la pieza y la mordaza, el programa establecerá el primer decalaje (G54) en el punto Cero Pieza.

Una vez definidos, para activar un decalaje se hará uso de las instrucciones en Código G G54, G55, G56 o G57 correspondientes a cada uno de los decalajes.

Acceso por atajo de teclado: Ctrl + D

1.1.3.3. Definir correctores.

Muestra la tabla de definición de correctores.

Un corrector determina las variaciones de radio o longitud de una herramienta, pudiendo aplicar dichas variaciones en forma de desplazamientos de la posición de la herramienta en la máquina (N).

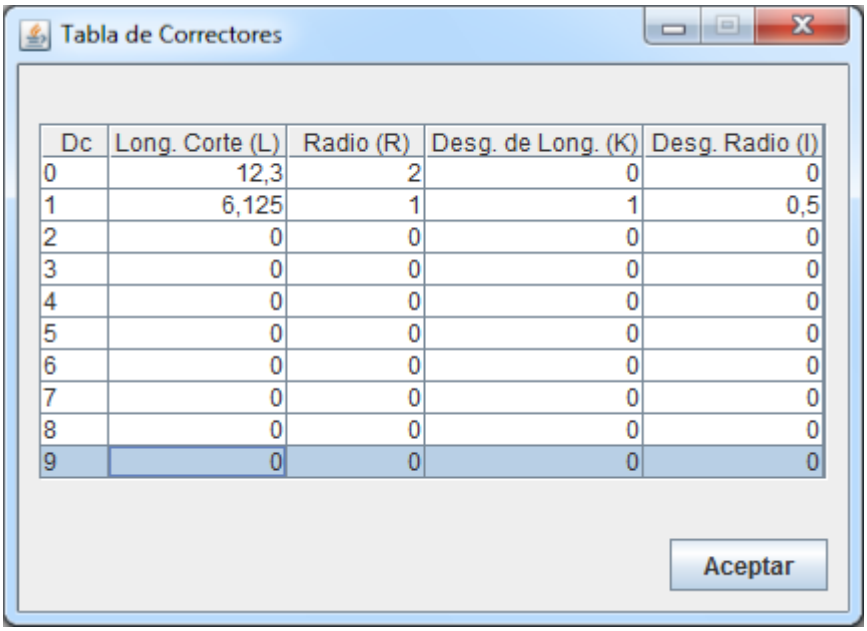


Figura Anexo A.14: Ventana Definir correctores.

El uso de correctores se hace necesario cuando, debido al desgaste de una broca o fresa, su radio o longitud merma, pudiendo compensar el desgaste de la herramienta mediante los desplazamientos adecuados, y en consecuencia, realizando una re-calibración de la herramienta.

Así mismo, se puede hacer uso de correctores para introducir pequeñas variaciones en interpolaciones similares, como por ejemplo, en el caso de la creación de los perfiles de una pieza.

El programa muestra una tabla con 10 correctores, cuyos parámetros son:

Parámetro	Función
Identificador (Dn)	Identifica el número de corrector (con "n" de 0 a 9), y será necesario para indicar, mediante Código G, el corrector asociado a una herramienta.
Longitud de corte (L)	Redefine la longitud de la herramienta.
Radio (R)	Redefine el radio de la herramienta.
Desgaste de longitud (K)	Determina el desplazamiento de longitud (vertical) de la posición de la máquina herramienta.
Desgaste de radio (I)	Determina el desplazamiento de radio (horizontal) de la posición de la máquina herramienta.

Tabla Anexo A.1: Parámetros de un corrector.

Un corrector irá siempre asociado a una herramienta, y se indicará a la máquina mediante la instrucción en Código G Tn Dc M06, donde la herramienta de identificador "n" quedará asociada con el corrector de identificador "c".

También mediante Código G, activaremos o desactivaremos la compensación de los correctores mediante las instrucciones G40, G41, G42 y G43.

Acceso por atajo de teclado: Ctrl + R

1.1.3.4. Configurar herramientas.

Muestra la ventana de configuración del carrusel porta-herramientas.

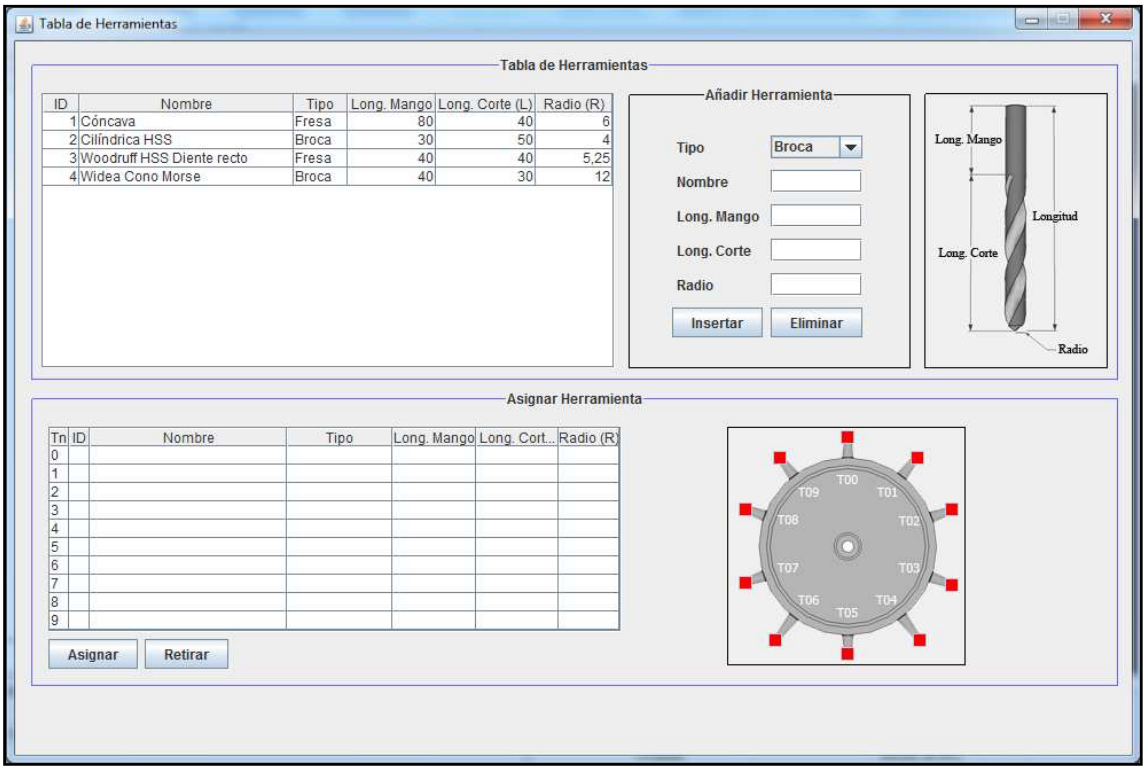


Figura Anexo A.15: Ventana Configurar herramientas.

Tal y cómo se realizaría en el mecanizado real, es necesario que el usuario asigne al porta-brocas de la máquina las brocas y fresas que va a utilizar.

En el panel podemos observar dos marcos diferenciados:

Tabla de herramientas:

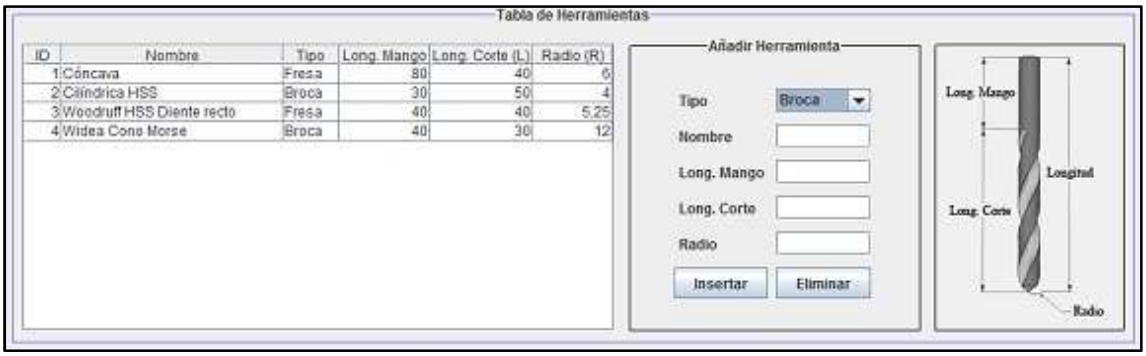


Figura Anexo A.16: Panel Tabla de Herramientas en ventana Configurar Herramientas.

El marco tabla de herramientas muestra las brocas y fresas disponibles.

Además de las cuatro herramientas que el programa muestra por defecto, podrán **definirse nuevas brocas y fresas**. Para ello, se especificará el tipo de herramienta (broca o fresa), nombre, longitud de mango y corte y radio desde el panel "Añadir herramienta", y presionando el botón "Añadir" la nueva broca se mostrará en la tabla de brocas disponibles.

Para facilitar la definición de nuevas herramientas, tras la selección del tipo de herramienta se mostrará un gráfico indicativo de las distintas medidas de la herramienta según su tipo.

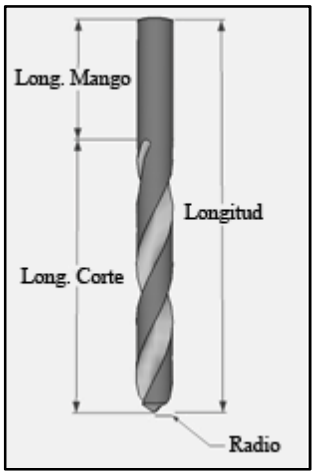


Figura Anexo A.17: Gráfico indicativo de las medidas de la broca.

Así mismo, podemos **eliminar herramientas de la tabla** de herramientas disponibles seleccionando la bronca mediante un clic izquierdo de ratón, y presionando el botón "Eliminar" del panel "Añadir Herramienta".

Asignar herramienta:

Figura Anexo A.17: Panel Asignar Herramienta en ventana Configurar Herramientas.

El marco asignar herramientas facilita la asignación de la herramienta en el porta-brocas de la máquina herramienta.

Cada una de las filas de la tabla, etiquetadas de 0 a 9, identifican el porta-brocas correspondiente en la máquina herramienta (T00 a T09).

Para asignar una determinada herramienta a un porta-brocas, se seleccionará la herramienta en la tabla superior, el porta-brocas en la tabla inferior y se presionará el botón "Asignar" del panel "Asignar Herramienta"

Una vez se haya realizado la asignación de la herramienta, el porta-brocas correspondiente se mostrará en color verde en el gráfico identificativo de la máquina herramienta, indicando que la ranura tiene una herramienta asignada.

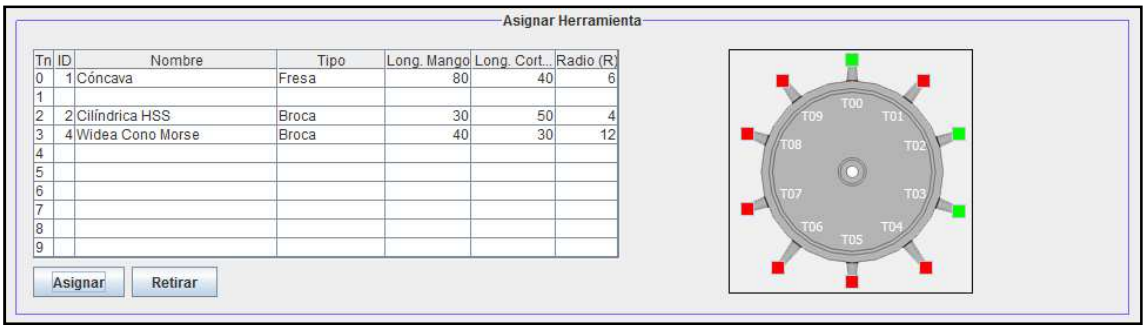


Figura Anexo A.18: Panel Asignar Herramienta tras asignación.

Para retirar una broca del porta-brocas, será necesario seleccionar el porta-brocas deseado en la tabla inferior y presionar el botón "Retirar" del panel "Añadir Herramienta". Al retirar la broca, el porta-brocas volverá a mostrarse en color rojo, indicando que dicha ranura queda libre.

Acceso por atajo de teclado: Ctrl + H

1.1.3.5. Guardar archivo de configuración.

Guarda un archivo de configuración mecaCNC (*.mcfg*).

Una vez presionado "*Guardar arch. de configuración*", se mostrará la ventana con el árbol de directorios donde el usuario deberá seleccionar la ruta y el nombre del archivo.

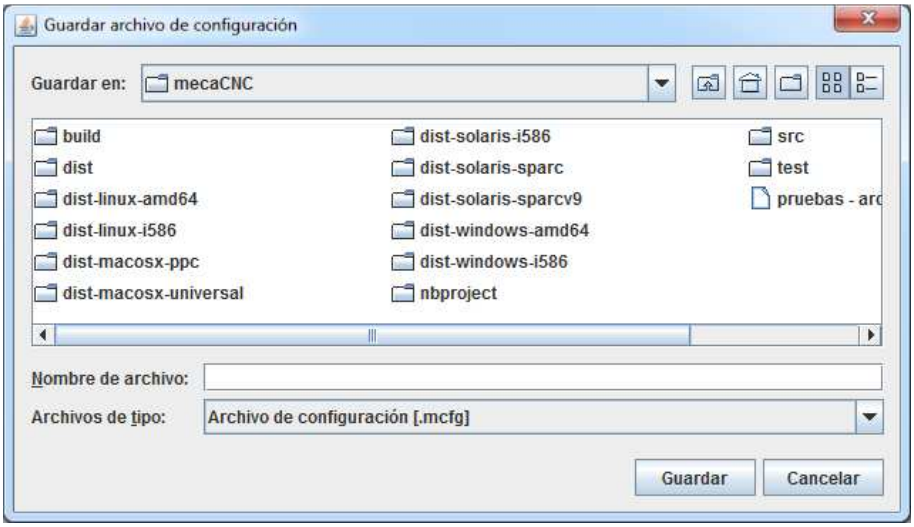


Figura Anexo A.19: Ventana Guardar archivo de configuración.

Al presionar en "*Guardar*", el programa generará un archivo de configuración con extensión "*.mcfg*" cuyo contenido describe:

- Dimensiones de la pieza y ubicación de la mordaza.
- Definición de decalajes.
- Definición de nuevas herramientas.
- Asociaciones de herramientas al porta-herramientas.
- Definición de correctores.

Nótese que si se guarda el archivo de configuración sin alguno de estos datos definidos, estos se evaluarán a <nulo>, su valor por defecto.

Acceso por atajo de teclado: Ctrl + G

1.1.3.6. Cargar archivo de configuración.

Carga un archivo de configuración previamente guardado, cuya extensión será "*.mcfg*".

Al presionar *Cargar arch. de configuración*, se mostrará la ventana con el árbol de directorios donde el usuario deberá seleccionar la ruta y el archivo de configuración deseado.

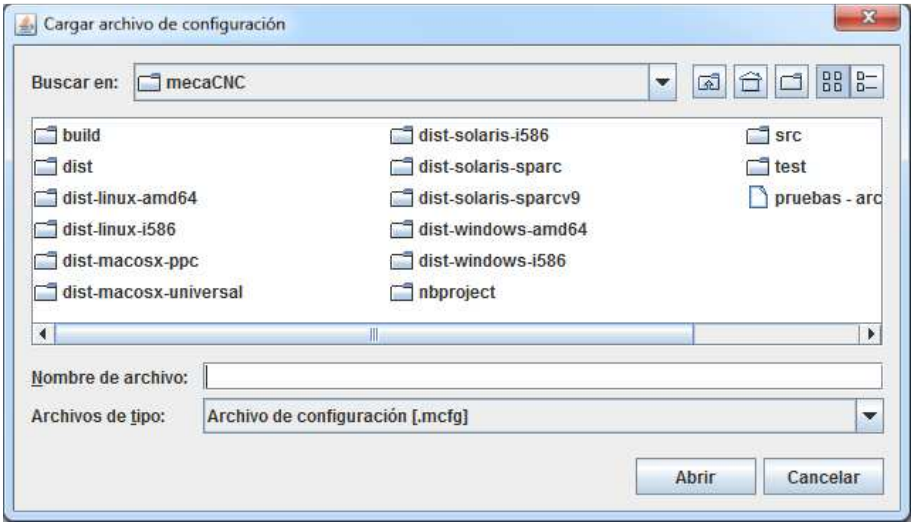


Figura Anexo A.20: Ventana Cargar archivo de configuración.

Dicho archivo establecerá en mecaCNC, en caso de que existan:

- Dimensiones de la pieza y ubicación de la mordaza.
- Definición de decalajes.
- Definición de nuevas herramientas.
- Asociaciones de herramientas al porta-herramientas.
- Definición de correctores.

Acceso por atajo de teclado: Ctrl + L

1.1.4. Cámara.

Elemento de la barra de menú que permite el acceso al control de la cámara del panel de representación.

Figura Anexo A.21: Menú Cámara.

1.1.4.1. Mostrar/Ocultar Ejes.

Muestra u oculta del panel *Representación*, los ejes de coordenadas en el punto Cero Pieza.

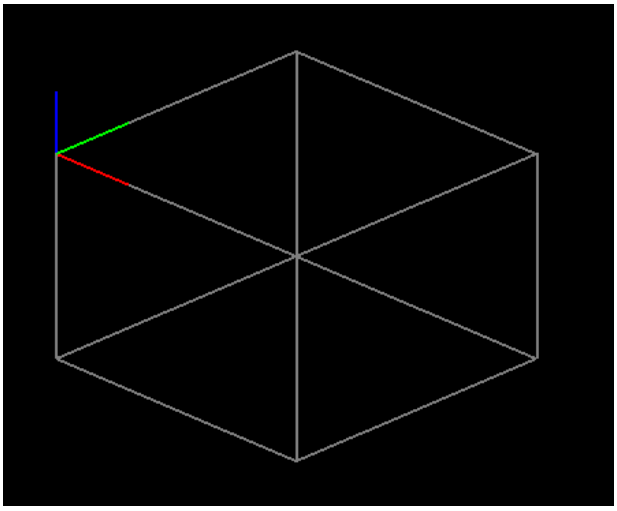


Figura Anexo A.22: Representación gráfica de los ejes de coordenadas.

El código de colores referencia, de color rojo la componente "+x", en verde la componente "+y" y en azul la componente "+z".

Acceso por atajo de teclado: Ctrl + J

1.1.4.2. Restablecer.

Emplaza la cámara del *panel Representación* a su posición inicial.

Acceso por atajo de teclado: Ctrl + 0

1.1.4.3. Vista.

Muestra los accesos directos a las distintas vistas principales predefinidas. La elección de una vista implicará la modificación de la cámara en el panel *Representación*.

Podrá seleccionar la vista del conjunto:

- Perspectiva isométrica (por defecto, posición inicial)
- Alzado
- Planta

- Perfil

Nótese que haciendo clic izquierdo del ratón sobre el panel *Representación* y arrastrando, podrá modificarse la cámara libremente.

Acceso por atajo de teclado: Ctrl + 1, Ctrl + 2, Ctrl + 3 y Ctrl + 4

1.1.4.4. Zoom

Elemento de menú que permite el acceso directo a las funciones de zoom (alejar o acercar) sobre la cámara del panel Representación.

Nótese que esta misma funcionalidad es accesible mediante la rueda del ratón sobre el panel Representación.

Acceso por atajo de teclado: Ctrl + + y Ctrl + -

1.1.5. Ayuda.

Elemento de la barra de menú que permite el acceso al sistema de ayuda, documentación e información del programa.

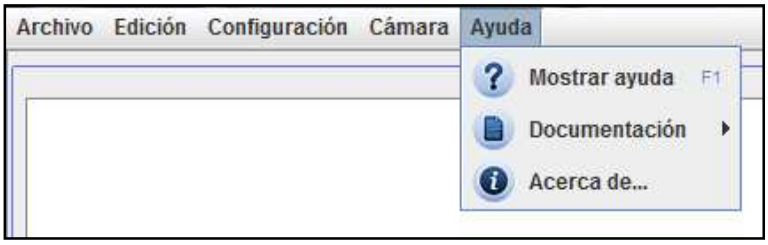


Figura Anexo A.23: Menú Ayuda.

1.1.5.1. Mostrar ayuda.

Muestra la aplicación correspondiente al sistema de ayuda integrada de mecaCNC.

La aplicación brinda acceso a la ayuda por contenidos, índice o búsqueda de palabras clave en base a la pestaña seleccionada, permitiendo imprimir aquellos documentos que se considere oportunos.

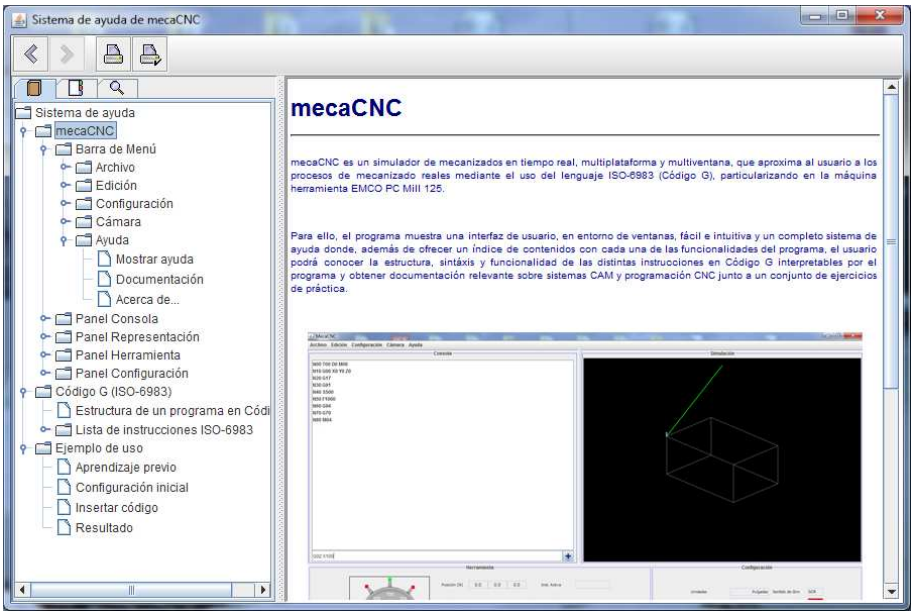


Figura Anexo A.24: Ventana del sistema de ayuda integrada.

Como se puede comprobar en la figura *Anexo A.24*, la ventana se divide en tres bloques:

- Barra superior: botones de control.
- Panel izquierdo: selección de tipo de sistema de ayuda y documento.
- Panel derecho: contenido del documento seleccionado.

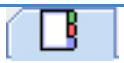
Icono	Función
	Acceder a la tabla de contenidos.
	Acceder al índice de ayuda.
	Acceder a la búsqueda por palabras clave.
	Volver al contenido anterior.
	Ir al siguiente contenido.
	Configurar la impresión.
	Imprimir el contenido actual.

Tabla Anexo A.2: Elementos del sistema de ayuda.

Según la pestaña de selección del tipo de sistema de ayuda, obtendrá:

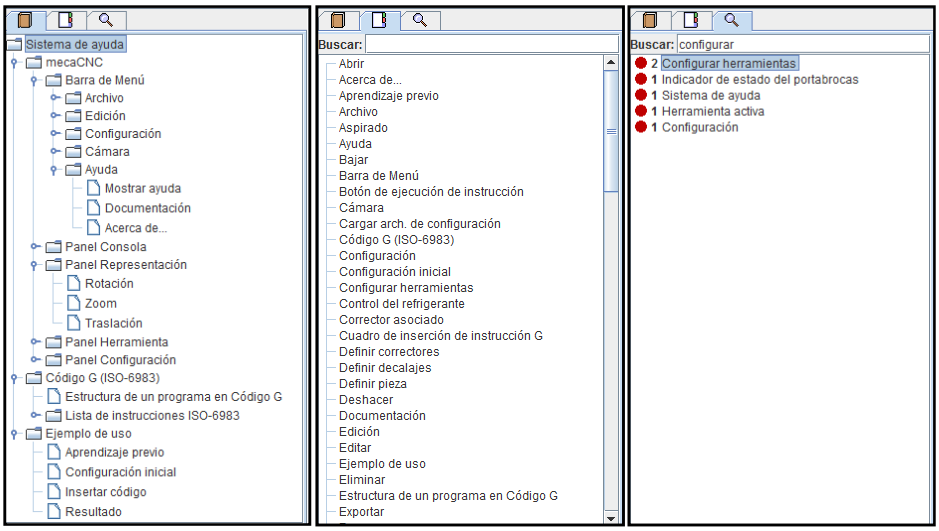


Figura Anexo A.25: (De izquierda a derecha) Sistema de ayuda en modo tabla de contenidos, índice de ayuda y búsqueda por palabras clave.

Tabla de contenidos:

Podrá navegar por los distintos elementos haciendo clic en el árbol de contenidos. En este se mostrarán elementos agrupados por carpetas de la misma temática.

Índice de ayuda:

Se disponen ordenados alfabéticamente todos los archivos de ayuda contenidos en este sistema. Puede buscar un término concreto escribiendo el término en el cuadro de texto "*Buscar:*" y presionando la tecla *retorno*.

Búsqueda por palabras clave:

Introduzca el término o términos que desee buscar en el cuadro de texto "*Buscar*:" y presione la tecla *retorno*. En el panel aparecerán todos aquellos documentos relacionados con su búsqueda, ordenados por el número de coincidencias de los términos especificados en el documento.

1.1.5.2. Documentación.

Brinda acceso directo a distintos documentos de auto-aprendizaje en formato PDF:

- Conceptos CAM.

- Sistemas CAM.
- Programación CNC (1) y (2).
- Manual EMCO PC Mill 125.
- Ejercicios de programación de máquinas-herramienta.

1.1.5.3. Acerca de…

Muestra la ventana de información de mecaCNC, incluyendo su versión e información del autor.

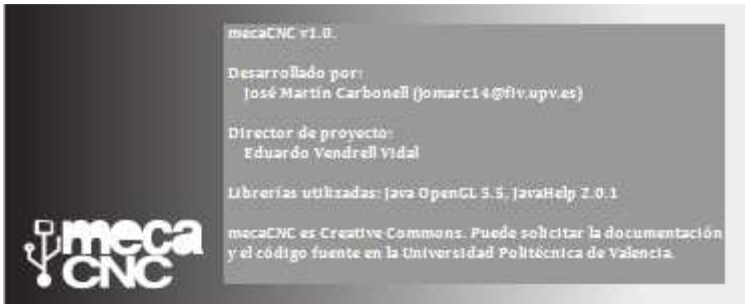


Figura Anexo A.26: Ventana Acerca de…

Para cerrar la ventana "Acerca de…" bastará con hacer clic con el ratón en cualquier parte de ella, salvo el texto.

1.2. Panel *Consola*.

El panel *Consola* permite la inserción, ejecución y registro de las instrucciones en Código G del mecanizado.

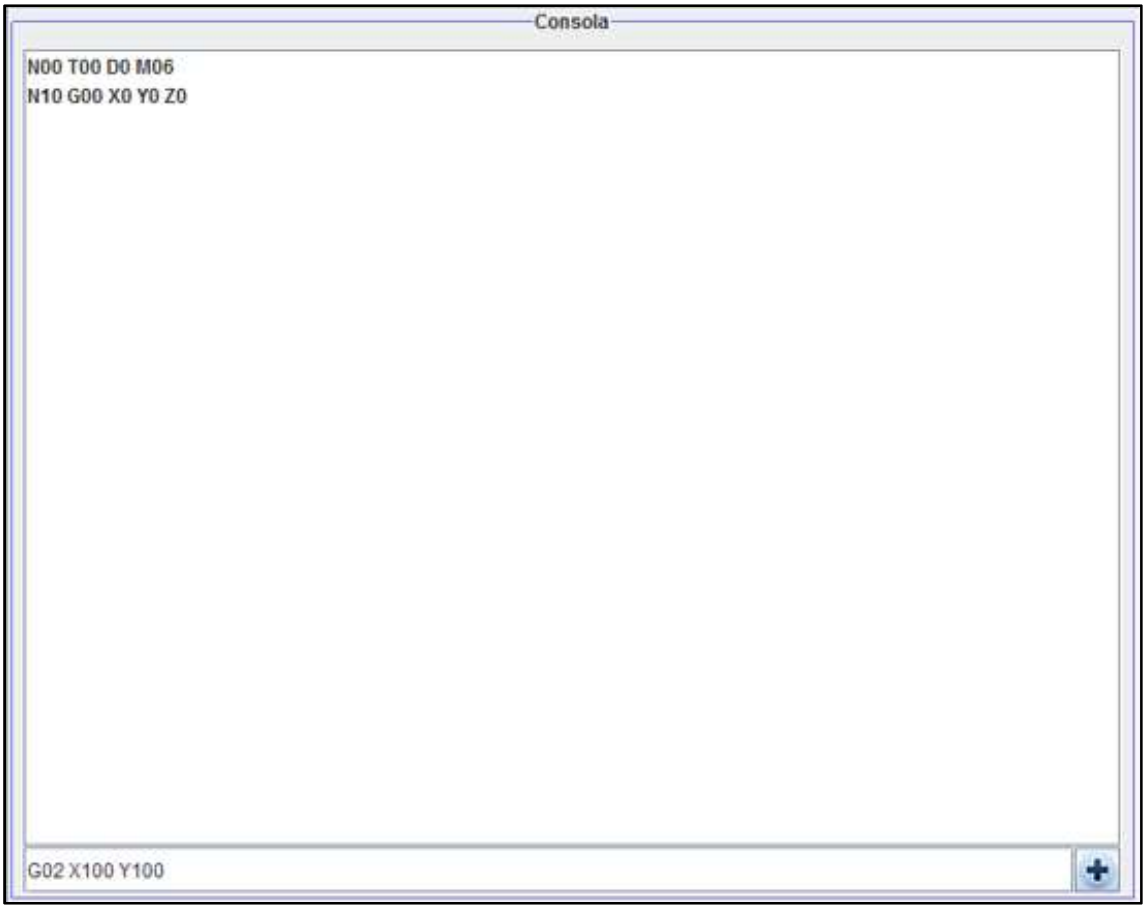


Figura Anexo A.27: Panel Consola.

El panel *Consola* se divide en 3 elementos principales:

1.2.1. Lista de instrucciones insertadas.

Elemento que comprende la práctica totalidad del panel, supone la lista de instrucciones ISO que el usuario ha insertado y su ejecución ha sido correcta, componiendo en su conjunto el programa de mecanizado.

Las operaciones posibles sobre la lista de instrucciones se realizarán mediante el menú contextual asociado, al cual se accede seleccionando una instrucción con el botón izquierdo del ratón, y una vez seleccionada, presionando con el botón derecho del ratón sobre ella.

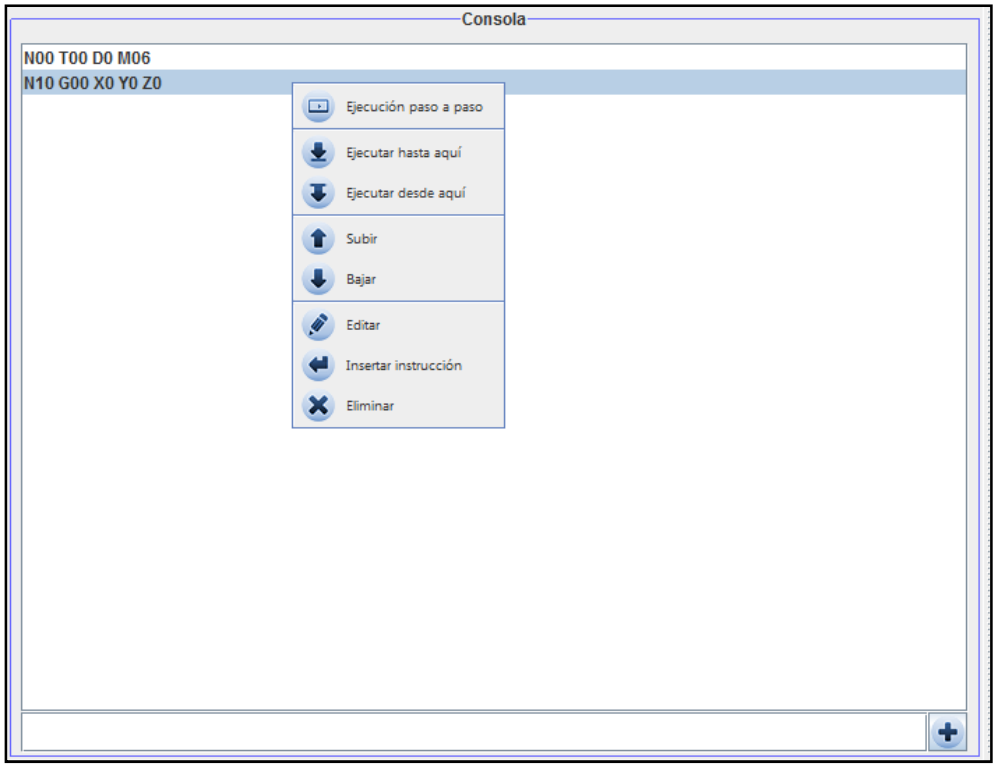


Figura Anexo A.28: Menú contextual en Lista de instrucciones ISO.

Cabe recordar, para todo el conjunto de acciones sobre la lista de instrucciones mediante el menú contextual, que cualquier variación en el orden, edición, inserción o eliminación de una instrucción en un programa ISO puede comportar un malfuncionamiento en el mecanizado, pudiendo llevar a resultados no deseados.

1.2.1.1. Ejecución paso a paso.

Muestra la ventana de control de la ejecución del código insertado.

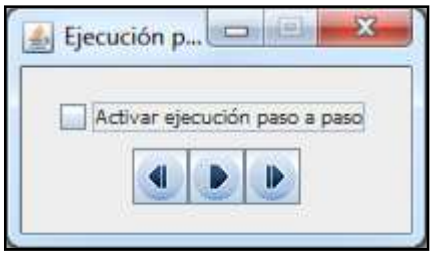


Figura Anexo A.29: Ventana de ejecución paso a paso.

Una vez activada la casilla de verificación que establece el modo de ejecución paso a paso, al seleccionarse una instrucción de la lista de instrucciones insertadas, se ejecutará el programa hasta dicha instrucción. Además, se ofrece la posibilidad de ir avanzando instrucción por instrucción mediante los botones instrucción anterior e instrucción siguiente de la ventana de ejecución paso a paso o mediante las flechas arriba y abajo del teclado.

Así mismo, el botón central de la ventana de ejecución paso a paso iniciará la ejecución, instrucción por instrucción desde la instrucción seleccionada, manteniendo un intervalo de medio segundo de retardo entre instrucciones para que el usuario pueda comprobar su efecto en el simulador.

Al cerrar la ventana o desactivar la casilla de verificación, se detendrá el modo de ejecución paso a paso, mientras tanto, cualquier selección de instrucción de la lista conllevará el comportamiento descrito.

1.2.1.2. Ejecutar hasta aquí.

Ejecuta el intervalo de instrucciones desde la primera instrucción hasta la instrucción seleccionada.

1.2.1.3. Ejecutar desde aquí.

Ejecuta el intervalo de instrucciones desde la instrucción seleccionada hasta la última instrucción insertada.

1.2.1.4. Subir.

Modifica el orden de la instrucción seleccionada, subiéndola a la línea anterior, y re-ejecutando la lista de instrucciones resultante.

1.2.1.5. Bajar.

Modifica el orden de la instrucción seleccionada, bajándola a la línea siguiente, y re-ejecutando la lista de instrucciones resultante.

1.2.1.6. Editar.

Permite modificar la instrucción seleccionada.

Se ha de tener en cuenta que es muy recomendable respetar el número de línea de la instrucción (Nx) para evitar posibles fallos en el mecanizado.

1.2.1.7. Insertar instrucción.

Permite insertar una instrucción a continuación de la seleccionada.

Estas instrucciones intermedias irán numeradas siguiendo el esquema de programación ISO-6981, incrementando en una unidad el número de línea de la instrucción insertada respecto a la anterior (contrariamente al habitual incremento de 10 unidades al insertar una instrucción desde la propia consola).

1.2.1.8. Eliminar.

Elimina la instrucción seleccionada y re-ejecuta la lista de instrucciones resultante.

1.2.2. Cuadro de inserción de instrucción G.

Permite al usuario la escritura de las instrucciones en Código G a ejecutar.

Figura Anexo A.30: Cuadro de inserción de instrucciones con instrucción de ejemplo "G02 X100 Y100..."

De manera análoga al resto de cuadros de texto, el cuadro de inserción permite copiar, cortar o pegar cadenas de texto.

Una vez escrita la instrucción a insertar, el usuario podrá presionar la tecla retorno para analizar y ejecutar dicha instrucción.

Además, se ha añadido al cuadro de inserción, la funcionalidad de instrucción anterior y siguiente (al estilo Terminal Linux) que permite, mediante la pulsación de las flechas arriba (▲) y abajo (▼) del teclado, obtener la instrucción previa o siguiente del conjunto de instrucciones registradas en la Lista de instrucciones insertadas.

1.2.3. Botón de ejecución de instrucción.

A pesar de que la inserción de instrucciones está concebida para ser ejecutada directamente a través de la tecla retorno, se habilita un botón de ejecución de la instrucción escrita en el cuadro de inserción.

Figura Anexo A.31: Botón de ejecución de instrucción.

1.3. Panel *Representación.*

El panel de representación muestra, en tiempo real y en tres dimensiones, la representación en alámbrico de la pieza y la herramienta, así como las líneas que definen las interpolaciones del mecanizado en curso.

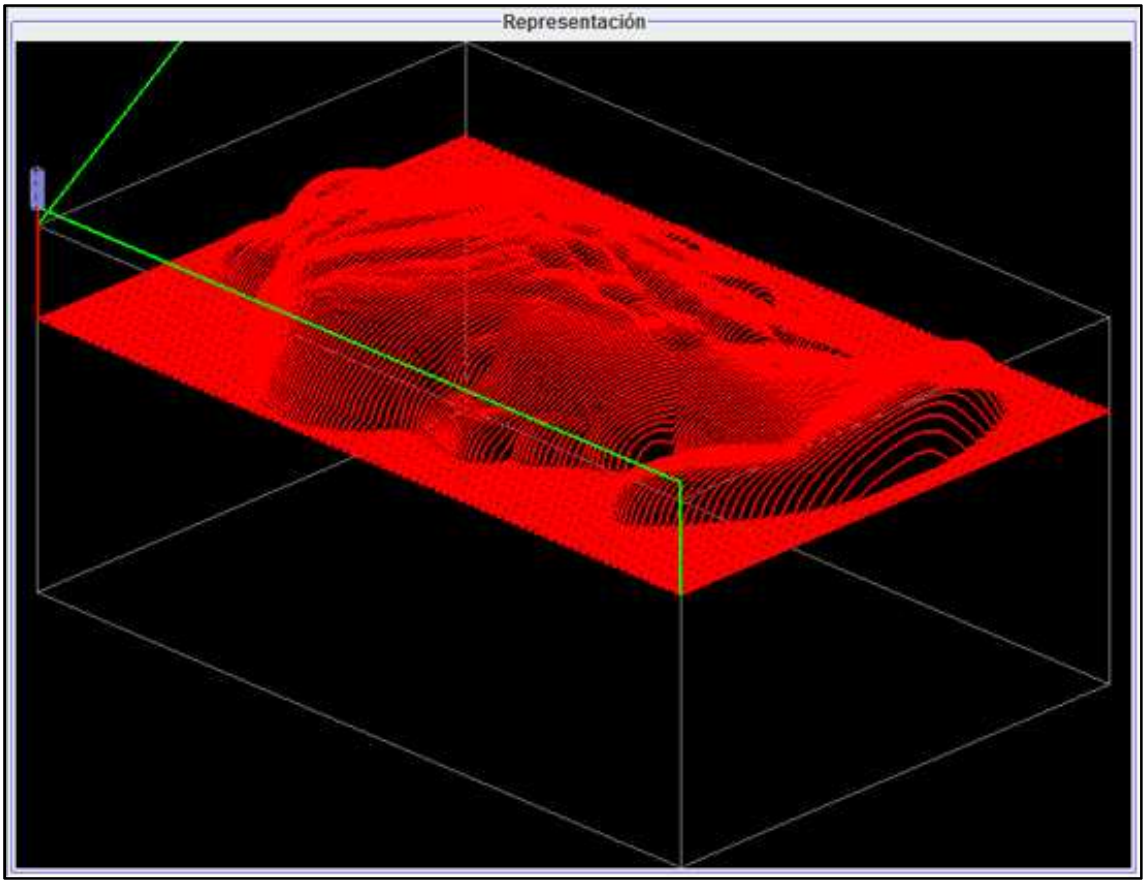


Figura Anexo A.32: Panel Representación con ejemplo de programa de mecanizado

Para obtener una mejor representación del mecanizado, se ha seguido un determinado código de colores:

- La herramienta se mostrará en color **morado**, y el cilindro que la representa variará su tamaño en función de los parámetros radio y longitud de la broca o fresa.
- La pieza se mostrará en color **gris**, y su tamaño dependerá del tamaño establecido en la definición inicial de la pieza.
- Las interpolaciones resultantes del mecanizado (trayectorias de la herramienta) se mostrarán en color **verde** en caso de que el husillo esté inactivo, y **rojo** en caso de que se encuentre el husillo activo.
- En el caso de que la compensación esté activada, se trazará, además de la trayectoria de la máquina en el código de colores descrito anteriormente, una segunda trayectoria (en color blanco) equivalente a la posición en base a las coordenadas que el usuario haya introducido, análoga a la trayectoria que se seguiría si la compensación estuviera desactivada.

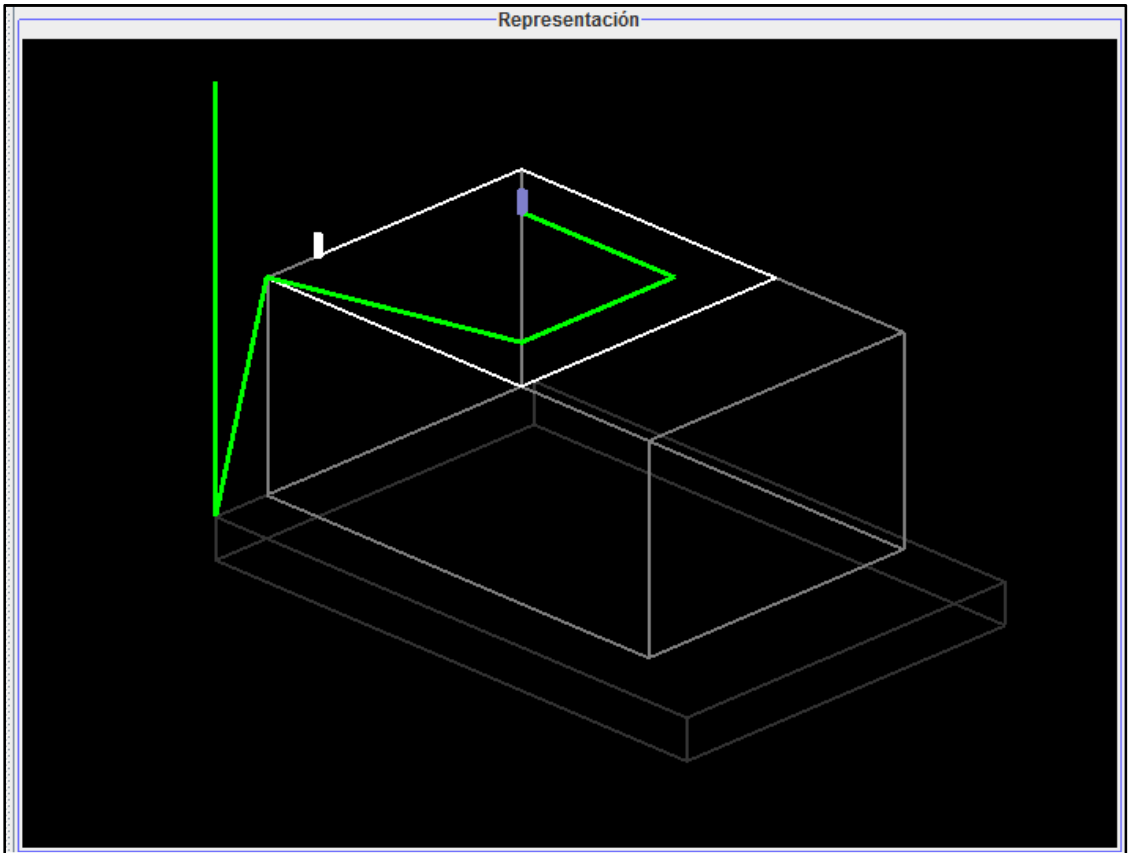


Figura Anexo A.33: Representación gráfica con compensación activa.

El panel de representación es interactivo, permitiendo al usuario variar la cámara o punto de vista mediante el uso del ratón. Las funciones de ratón para el panel de representación son:

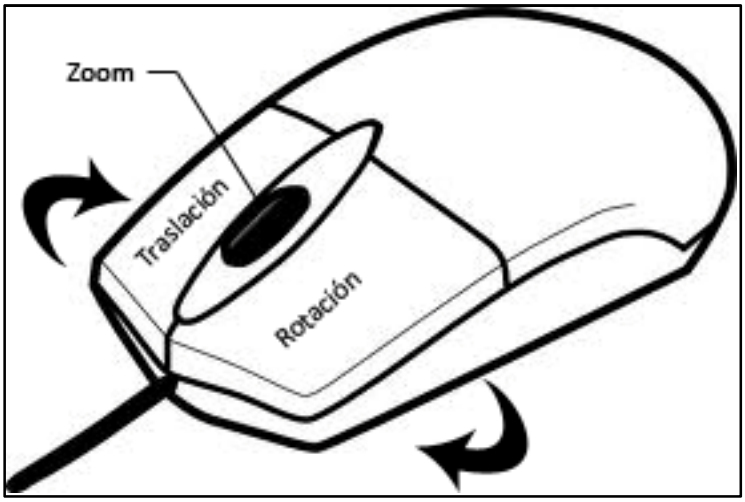


Figura Anexo A.34: Uso de ratón para modificar la cámara en la representación.

1.3.1. Rotación.

El movimiento de rotación de la representación irá asociado al movimiento de **arrastre del ratón tras pulsar y mantener el botón izquierdo** del sobre el panel *Representación*.

La rotación será vertical, horizontal o una combinación de ambas según la trayectoria de movimiento que el usuario haga con el ratón.

1.3.2. Traslación.

El movimiento de traslación de la representación irá asociado al movimiento de **arrastre del ratón tras pulsar y mantener el botón derecho** del sobre el panel *Representación*.

La traslación será vertical, horizontal o una combinación de ambas según la trayectoria de movimiento que el usuario haga con el ratón.

1.3.3. Zoom.

Para realizar zoom sobre la pieza, se deberá hacer **girar la rueda del ratón** sobre el panel *Representación*.

El giro de la rueda del ratón hacia arriba implica acercarse a la pieza y de forma contraria, el giro hacia abajo aleja la cámara de la pieza.

1.4. Panel *Herramienta*.

El panel *Herramienta* muestra el estado actual de la máquina-herramienta en relación a su posición, la instrucción activa (en caso de interpolaciones), las herramientas y correctores asignados y seleccionados y la compensación de radio.

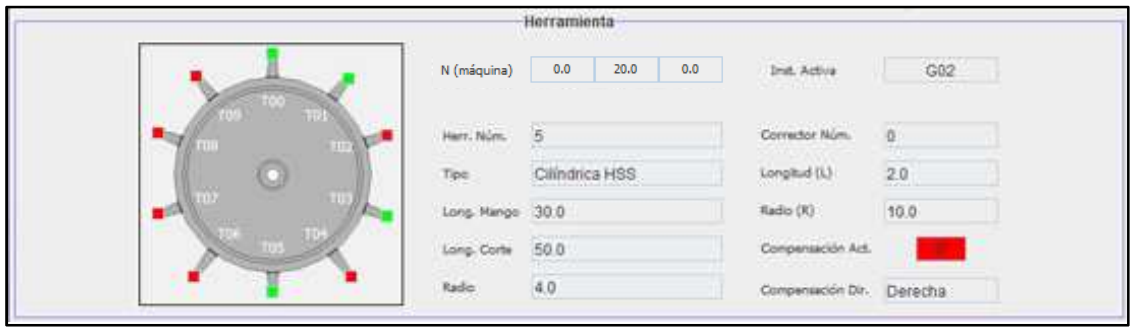


Figura Anexo A.35: Ejemplo de panel Herramienta con mecanizado en curso.

Los datos del panel se irán completando dinámicamente y en tiempo real en función de las instrucciones insertadas y la configuración establecida. Estos son:

1.4.1. Indicador de estado del carrusel porta-brocas.

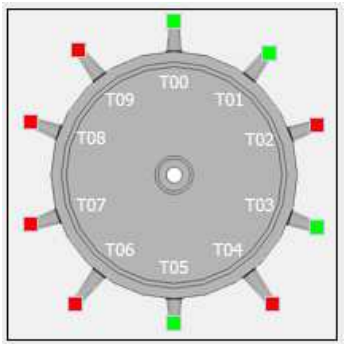


Figura Anexo A.36: Indicador de estado del carrusel porta-brocas.

Muestra aquellos porta-brocas en los que ha sido asignada una herramienta.

El cuadro muestra los 10 porta-brocas del carrusel (T00-T09) indicando mediante un cuadro verde si existe una broca asignada, y rojo en caso contrario.

Se accederá a la asignación y definición de brocas desde *Barra de menú > Configuración > Configurar Herramientas*, explicado en detalle en el apartado 1.1.3.4 de este manual.

1.4.2. N (máquina).

Muestra la posición actual de la herramienta activa.

Cabe destacar que en el caso de instrucciones de interpolación (G00, G01, G02 y G03), si se omite algún parámetro correspondiente a una coordenada, se tomará el valor de dicha componente en N (máquina).

1.4.3. Instrucción activa.

Indica la instrucción en caso de que la última instrucción sea una interpolación.

Las instrucciones de interpolación (G00, G01, G02 y G03), permiten la programación de bloques contiguos del mismo tipo de instrucción, únicamente especificando sus nuevos parámetros. Para ello, el programa se apoya en el indicador instrucción activa, realizando la funcionalidad correspondiente a la instrucción activa sobre los nuevos parámetros.

1.4.4. Herramienta activa.

Muestra los datos de la herramienta activa.

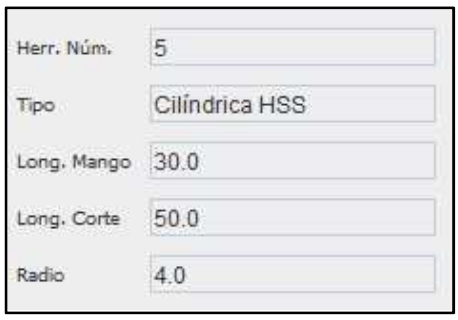


Figura Anexo A.37: Datos identificativos de la herramienta activa en el panel Herramienta.

Para activar una herramienta previamente asociada se hará uso de la instrucción en Código G *Tn Dc M06*.

Puede obtener más información sobre la asignación de herramientas en el apartado 1.1.3.4 del presente manual.

1.4.5. Corrector asociado.

Indica las características del corrector asociado a la broca activa.

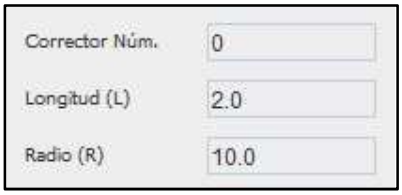


Figura Anexo A.38: Datos identificativos del corrector asociado en el panel Herramienta.

De forma análoga a la activación de la herramienta, la asociación de un corrector se realizará mediante la instrucción en Código G *Tn Dc M06*.

Puede obtener más información sobre correctores en el apartado 1.1.3.3 del presente manual.

1.4.6. Indicador de compensación.

Indica si la compensación radio se encuentra activa, y en caso de estar activa, hacia qué lado compensará.

Figura Anexo A.39: Datos identificativos de la compensación de radio en el panel Herramienta.

Para activar la compensación se hará uso de las instrucciones G41 y G42, mientras que G40 desactivará la compensación.

Una vez activada la compensación, se añadirá al panel *Herramienta* el campo correspondiente a las coordenadas insertadas frente a la posición real de la máquina.

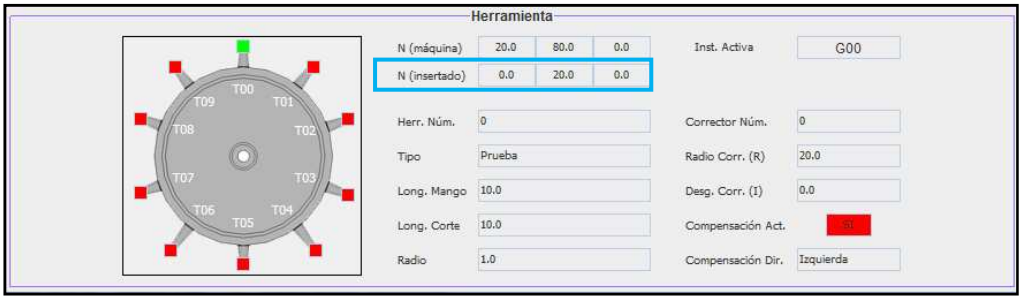


Figura Anexo A.40: Panel Herramienta con compensación activa.

Puede obtener más información sobre el uso de la compensación en el apartado 1.1.3.3 del presente manual.

1.5. Panel *Configuración.*

El panel *Configuración* muestra el estado actual de la máquina herramienta en relación a sus unidades de medida, velocidades, plano de mecanizado, husillo y control del refrigerante.

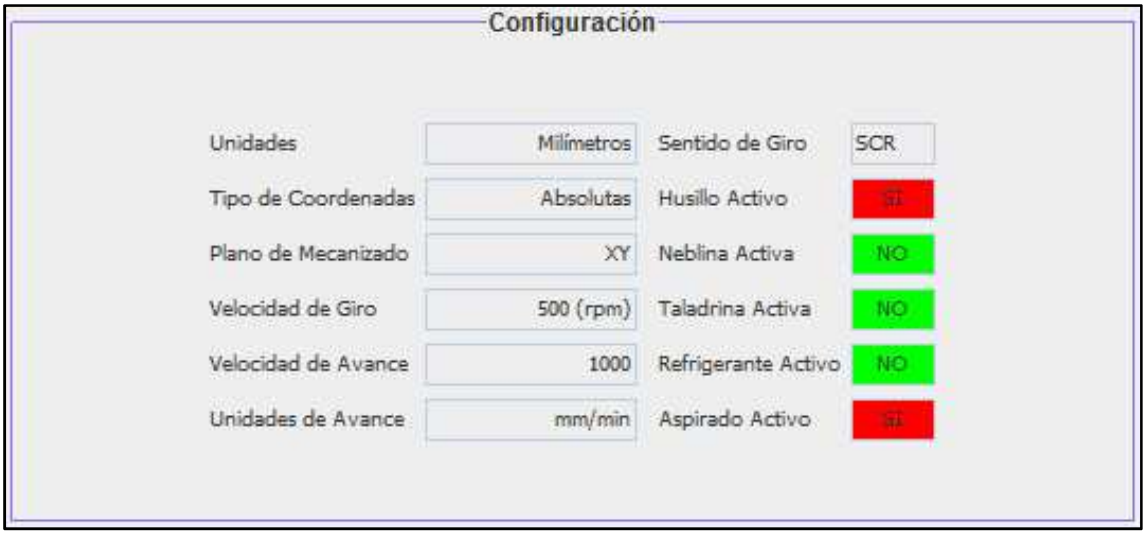


Figura Anexo A.41: Ejemplo de panel configuración con mecanizado en curso.

Los datos del panel se irán completando dinámicamente y en tiempo real en función de las instrucciones insertadas. Estos son:

Indicador	Descripción	Posibles valores	Instrucciones G
Unidades	unidades de medida que serán utilizadas en la programación	Pulgadas	G70
		Milímetros	G71
Tipo de coordenadas	Especifica el tipo de coordenadas	Absolutas	G90
		Incrementales	G91
Plano de mecanizado	Especifica sobre cuál de los tres planos de la pieza, se realizará el mecanizado	XY	G17
		XZ	G18
		YZ	G19
Velocidad de giro	Muestra la velocidad de giro del husillo en rpm	x ∈ [150 - 5000]	Sx
Velocidad de avance	Muestra la velocidad de desplazamiento de la máquina herramienta	x ∈ [0 - 4000]	Fx
Unidades de avance	Indica el tipo de unidades de medida seleccionadas en referencia al desplazamiento de la máquina herramienta	milímetros por minuto (mm/min)	G94
		milímetros por revolución (mm/rev)	G95
Sentido de giro	Indica el sentido en el que el husillo está girando	SR (horario)	M03
		SCR (anti-horario)	M04
Husillo activo	Indica si el husillo está activo.	SI	M03, M04
		NO	M05
Neblina activa	Filtro de eliminación de neblinas de aceite.	SI	M07
		NO	Por defecto
Taladrina activa	Al ser activado, se inyecta taladrina al punto de mecanizado.	SI	M08
		NO	Por defecto
Refrigerante activo	Al ser activado, se inyecta refrigerante al punto de mecanizado.	SI	M09
		NO	Por defecto
Aspirado activo	Al ser activado, inicia la función de aspirado de la viruta generada por el mecanizado.	SI	M71
		NO	M72

Tabla Anexo A.3: Indicadores del panel configuración e instrucciones G asociadas.

ANEXO B

EL CÓDIGO ISO-6983 EN MECACNC

Consideraciones previas.

Para la correcta creación y ejecución de un programa en Código G, será necesario cumplir un conjunto de reglas que compondrán la estructura del programa. El presente documento únicamente contempla la estructura del programa CNC, dándose por hecho que:

Proceso real	Simulador
La pieza ha sido ubicada y fijada en la mesa de trabajo	Se ha definido la pieza y la ubicación de la mordaza desde *"Configuración > Definir Pieza"*
Se han emplazado las herramientas a utilizar en los porta-brocas	Se han definido las herramientas y se han asignado a los porta-brocas desde "*Configuración > Configurar Herramientas"*
Se han definido los decalajes	Se han definido los decalajes en la tabla de decalajes desde *"Configuración > Definir Decalajes"*
Se han definido los correctores	Se ha definido la tabla de correctores desde *"Configuración > Definir Correctores"*

Tabla Anexo B.1: Acciones requeridas en el proceso real y su equivalente en mecaCNC.

En programación ISO-6983, **las instrucciones irán numeradas siguiendo un formato compuesto por el identificador "N" y un número natural creciente.** La tarea de enumeración de las instrucciones será llevada a cabo automáticamente por mecaCNC, siendo innecesaria la especificación de la numeración en cada inserción. La auto-numeración se realizará en incrementos de diez unidades, permitiendo así la inserción de hasta nueve instrucciones intermedias en caso de que fuera necesario.

La programación ISO-6983 permite, en ciertos casos, encadenar instrucciones de distinto tipo y funcionalidad. En el caso de mecaCNC, se ha considerado oportuno no permitir encadenamiento de instrucciones salvo en el caso de "G01 Xx Yy Zz [Ff]", de esta forma, el usuario conocerá el resultado de la funcionalidad asociada a cada instrucción por separado.

La lectura de archivos que incluyan Código G desde mecaCNC implica que el simulador no considere aquellas líneas que empiecen con el caracter *"("* o *"%"*. Se ha considerado conveniente esta notación debido a que, por una parte, el resultado de la exportación de Código G llevada a cabo por otros programas simuladores, incluye instrucciones orientadas a la configuración delimitadas por *"()"*. Por otra parte, en la exportación del Código G de mecaCNC

a la máquina EMCO PC Mill 125, se insertará la instrucción de control *"%1000MPF"* que indica a la máquina que se trata de un programa ISO-6983.

Fases del programa ISO-6983.

1. **Inicialización.**

   Fase previa al inicio del mecanizado. Supone el conjunto de instrucciones que determinan las unidades de medida, tipos de coordenadas, plano de mecanizado, velocidades de rotación y avance, selección de la herramienta inicial y decalaje de origen.

2. **Control de trayectoria:**

   Fase activa en el mecanizado. Supone el conjunto de instrucciones (interpolaciones lineales y circulares) que controlan el movimiento de la máquina herramienta.

3. **Control de máquina:**

   Fase activa y final del mecanizado. Conjunto de instrucciones que controlan el husillo, las compensaciones, la selección de una nueva herramienta, el refrigerante y la terminación del programa.

Lista de instrucciones interpretables.

Fase	Función	Instrucción
Inicialización	Unidades de medida	G70, G71
	Tipo de coordenadas	G90, G91
	Selección de plano de mecanizado	G17, G18, G19
	Unidades de avance	G94, G95
	Velocidad de avance	F
	Velocidad de rotación del husillo	S
	Selección de la herramienta inicial	Tn Dm M06
	Selección del decalaje de origen	G54, G55, G56, G57
Control de trayectoria	Movimiento libre	G00
	Interpolación lineal	G01
	Interpolación circular	G02, G03
Control de máquina	Giro y sentido del husillo	M03, M04, M05
	Neblina	M07
	Taladrina	M08
	Refrigerante	M09
	Aspirado	M71, M72
	Compensación	G40, G41, G42
	Fin del programa	M02, M30

Tabla Anexo B.2: Lista de instrucciones interpretables.